Network Working Group
Internet-Draft
Intended status: Informational
Expires: July 21, 2014

M. Smith
R. Kandula
Cisco Systems
January 17, 2014

Source-Group Tag eXchange Protocol (SXP)
draft-smith-kandula-sxp-00

Abstract

This document discusses source-group tag exchange protocol (SXP), a control protocol to propagate IP address to Source Group Tag (SGT) binding information across network devices.

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].

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 July 21, 2014.

Copyright Notice

Table of Contents

1. Introduction

SXP stands for the Source Group Tag (SGT) eXchange Protocol. Source groups are the endpoints connecting to the network that have common network policies. Each source group is identified by a unique SGT value. The SGT to which an endpoint belongs can be assigned statically or dynamically, and the SGT can be used as a classifier in network policies. SXP is a control-plane mechanism used to transport an endpoint's SGT along with the IP address from one SGT-aware network device to another. The data that SXP transports is referred to as the IP-SGT binding in the rest of the document.

1.1. Terminology

This document frequently uses the following terms:

SGT Source Group Tag

SXP Source-Group Tag (SGT) eXchange Protocol

IP-SGT The IP Address to SGT binding that is exchanged over SXP connection

SXP Speaker The peer that sends the IP-SGT bindings over the SXP connection

SXP Listener The peer that receives the IP-SGT bindings over the SXP connection

Binding Database The database of IP-SGT bindings that the SXP Speaker uses to export to the peer

2. SXP Overview

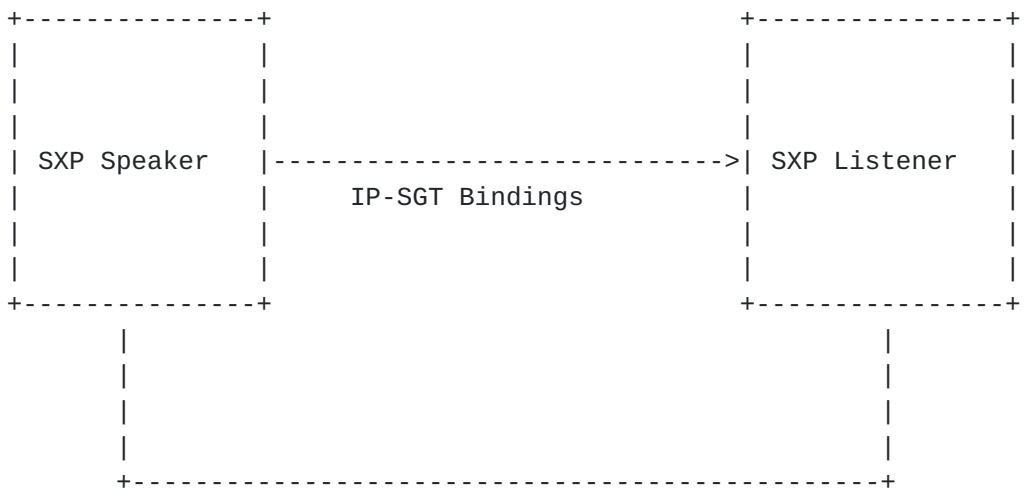
SXP uses TCP as its transport protocol to set up SXP connection between 2 separate network devices. Each SXP connection has one peer designated as SXP speaker and the other peer as SXP listener. The peers can also be configured in a bi-directional (both) mode where each of them act as both speaker & listener. Connections can be initiated by either peers but binding information is always propagated from a speaker to a listener.

At a very high level, SXP comprises of the following -

- SXP connection establishment

- IP-SGT bindings propagation

A typical deployment scenario of SXP is illustrated below -



SXP Connection

3. SXP Operation

The SXP connections are point-to-point and uses TCP as the underlying transport protocol. The summary of operations is illustrated in each of the sections below.

3.1. SXP Connection Management

An SXP connection consists of the following parts:

3.1.1. SXP Connection

- The peer IP address, source IP address (the best IP address of local device used to reach the specified peer IP address)
  - Used to construct TCP connection
- The connection mode
  - An SXP connection can be configured for uni-directional or bi-directional communication. For each direction, one peer must act as an sxp speaker and the other act as an sxp listener.

  * The SXP speaker is responsible for sending the IP-SGT bindings.

  * The SXP listener is responsible for collecting the IP-SGT bindings received from the speaker peer

- SXP connection password

  * If SXP data integrity and authentication are required, then both the peer devices must have same SXP password.

3.1.2. SXP Message integrity/authenticity

The connection peers on the 2 network devices supply the same SXP password to the TCP layer which will authenticate all further messages using the MD5 algorithm (RFC 2385). The SXP payload is not encrypted because the main requirement is for the receiving device to determine that the message originated from a valid source and not to prevent snooping of message (discussed in security considerations section) payload. SXP uses the underlying TCP MD5 option for message integrity/authenticity. The TCP MD5 is exchanged (negotiated) during the initial TCP 3 way handshake. Both sides (peers) are pre-configured with the keys (password) which will be used for forming the MD5 digest. Each packet (segment) needs to be "signed" with the MD5 digest (16 bytes) formed using the password as the key.

Whenever the password needs to be changed for an existing TCP connection, the peer resets the exising TCP connection and sets up a new TCP connection with the new password. Changing the default password will cause all SXP connections using the default password to be re-established.

3.1.3. SXP Connectivity Discovery and Connection Recovery

SXP uses the TCP keep alive mechanism with the default behavior to maintain SXP connectivity. A SXP device attempts to maintain connectivity with each peer. In case of failures, the SXP device continues to retry connection setup with all the peers with which the SXP connections have not been established. This continues until either a connection is established or the peer is removed from the peer set by a change in configuration. This mechanism ensures automatic recovery of the SXP connection once the connectivity issue is resolved. The SXP connection can be re-established by either device and hence the connection can be brought up sooner without relying on the retry timers of the peer devices. Note that this is just an optimization and not a requirement for SXP connection recovery. If both ends of an SXP connection set up the TCP connection at the same time, the end with source IP address higher

than the peer IP address wins: i.e. the TCP connection initiated from that end is kept and the other TCP connection is torn down.

3.1.4. SXP Connection Setup Sequence

When a SXP connection is configured, it initiates a TCP connection setup request. SXP communication starts after TCP connection has been established between the two network devices. The end that initiates the TCP connection starts the exchange by sending the OPEN message to negotiate SXP version, SXP mode, and if that end is a listener, SXP capabilities attribute. Capabilities advertisement and Capabilities Attribute are described with additional details in section 5.4.1. The receiving device responds with an OPEN_RESP message that includes similar attributes. The connection can only be setup with one end configured as a speaker and the other end configured as a listener. See the following sections regarding version negotiation and capability exchange process descriptions. Along with version negotiation, the hold time required for the connection is also negotiated. The following sections explain the hold time negotiation in detail.

After SXP connection is established, binding information can be sent from speaker to listener with UPDATE message. Whenever the SXP connection has to be closed the underlying TCP connection is closed.

3.1.5. SXP Connection States

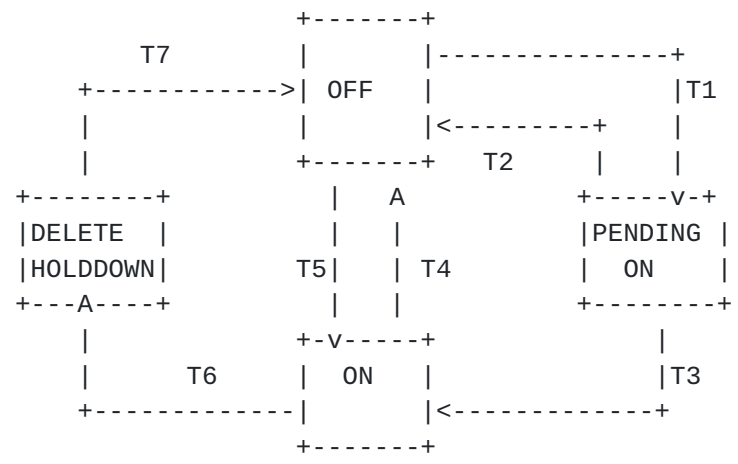
An SXP connection maintains following states: OFF, PENDING_ON, DELETE_HOLD_DOWN, ON.

OFF: In this state, a connection initiation has not been started. This is the only state that an SXP connection will retry establishing the TCP connection.

PENDING_ON: In this state, an SXP OPEN message has been sent to the peer and a response from the peer SXP is expected.

DELETE_HOLD_DOWN: In this state, a connection that was previously in ON state has been terminated. Only a listener can be in this state.

ON: SXP is in ON state when SXP OPEN or SXP OPEN RESP message has been received. SXP connection has been setup successfully. An SXP connection will only propagates bindings in the ON state.



T1-T7: State Transitions

T1: The connection sends a TCP connect request and an SXP OPEN message to the peer. The connection then transits to the PENDING ON state.

T2: TCP connection setup failure;

T3: The connection received an SXP OPEN RESPONSE or SXP OPEN message. If the delete hold down timer is running, the timer is stopped and the reconciliation timer is started.

T4: TCP connection failure; SXP message process failure; SXP configuration change.

T5: The TCP connection is setup and an SXP OPEN message is received. If the delete hold down timer is running, the timer is stopped and the reconciliation timer is started.

T6: TCP connection failure when a connection is in listener mode. The delete hold down timer is started with this state transition.

T7: The delete hold down timer expired or TCP connection setup failure.

In failure scenarios, the connection transits to OFF state and an attempt is made to re-establish the connection.

3.2. Binding Database

A system should have a master binding database for reconciling binding information from multiple SXP connections. The database should consist of a single binding per IP address and there should be a mechanism to handle same bindings from multiple SXP connections.

3.2.1. SXP Learned IP-SGT Binding recovery

When an SXP connection goes down on a network device, SXP continues to learn/advertise IP-SGT bindings on other SXP connections that are alive. If the device has learnt any bindings from the peer where the SXP connection goes down, a delete hold down timer is started for that connection:

- Once the timer expires, all binding entries that were learned from the failed peer are deleted. The deletion of the bindings for which the failed peer was the only source are reported to the master binding database.

- If the connection recovers before the delete hold down timer expiry, a reconcile timer is started to clean up old bindings that didn't get informed to be removed because of the loss of connectivity.

4. Message Formats

The SXP Messages and their formats are described in detail in this section.

4.1. Bit and Octet Numbering Convention

Throughout this section, the following numbering convention is used to depict the SXP message formats:

Each diagram depicts the format and size of each field in bits. Implementations MUST send the bits in each diagram as they are shown, traversing the diagram from top to bottom and then from left to right within each line (which represents a 32-bit quantity). Multi-byte fields representing numeric values must be sent in network (big endian) byte order. Descriptions of bit field (e.g., flag) values are described referring to the position of the bit within the field. These bit positions are numbered from the most significant bit through the least significant bit, so a 1-octet field with only bit 0 set has the value 0x80.

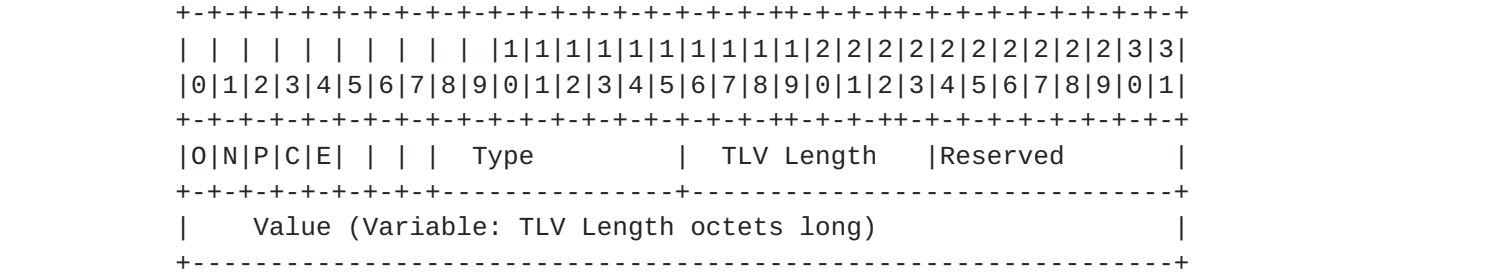
4.2. SXP Message Header

Field	Length	Description
SXP Message Length	4	This 4-octet unsigned integer indicates the total length of the message, including the header but excluding authentication tag. The maximum message length is 4096
SXP Message Type	4	This 4-octet unsigned integer indicates the type code of the SXP message. The following type codes are defined: 1 - OPEN 2 - OPEN_RESP 3 - UPDATE 4 - ERROR 5 - PURGE_ALL 6 - KEEPALIVE
SXP Payload	Variable	SXP message payload is an optional field. Its content when present depends on the message type

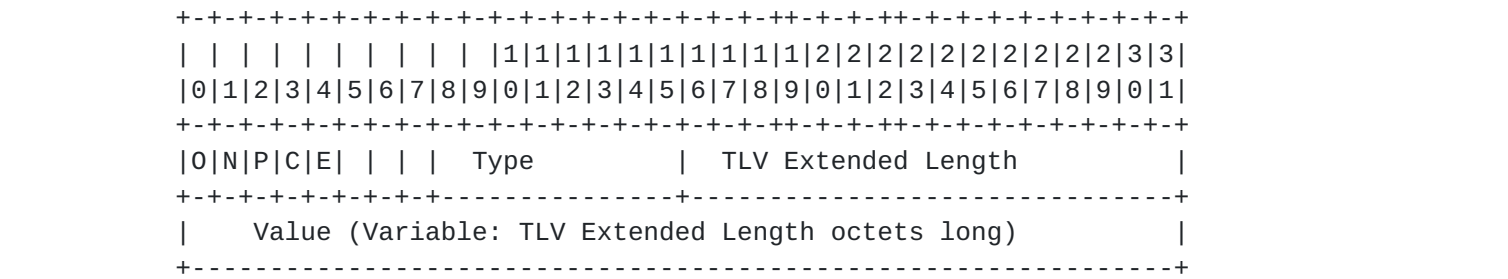
4.3. Attribute Formats

SXP messages can contain a variable number of fields. Some fields have pre-defined format that every implementation shall understand. Other fields are called attributes and they have an extended {Type, Length, Value} (TLV) format in order to allows for gradual introduction of a new feature without requiring a protocol version change. Each attribute is tagged and includes a length field. This allows for newer versions of SXP to add capabilities and coexist with old versions of SXP in the same deployment. Each attribute includes a Flags field to control processing and transitive processing of optional attributes. The format of an attribute {flags, type, length, value} (TLV) is encoded as follows:

Compact TLV



Compact Extended Length TLV



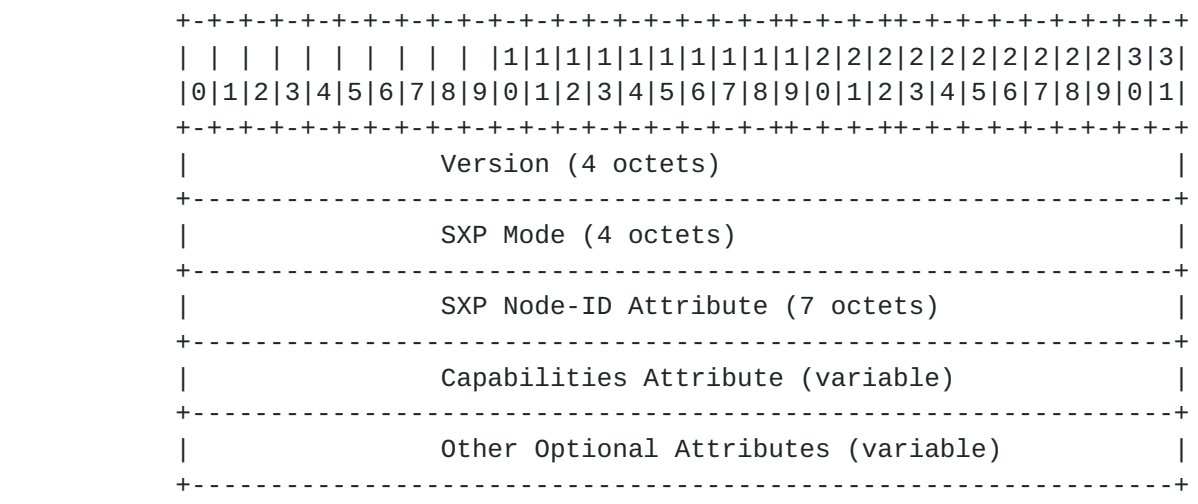
Field	Bits	Description
Optional (O)	0	1 - Attribute is optional and could be ignored 0 - Attribute is well-known and must be recognized
Non-Transitive (N)	1	0 - Attribute shall be forwarded even if it is not recognized 1 - Attribute shall not be forwarded
Partial (P)	2	1 - When a transitive attribute is forwarded but not recognized 0 - Otherwise
Compact (C)	3	1 - TLV is using compact Type and Length encoding 0 - TLV is encoded using entire 4 octets long words for Type and Length. All other flags are 0
Extended Length (E)	4	0 - Length is a single octet long 1 - Length is 2 octets long
Reserved	5-7	Reserved for future use. Must be transmitted as 0 and ignored on receive

Field	Length	Description
Flags	1	Attribute Flags
Type	1 or 4	TLV Type
Length	1, 2 or 4	Length (in octets) of the TLV Value field (i.e. not including the 3, 4, or 8-octets attribute header. Unsigned integer in the following range: [0..255] - For Compact non-Extended Length attribute. [256..4084] - For Compact Extended Length attribute.
Value	Variable	TLV values, which depend on the TLV type

Attribute Length field of Compact attributes SHOULD be only as large as necessary to hold an unsigned integer that specifies the number of octets in the attribute Value field. This implies that Extended-Length field SHOULD contain a value greater than 256 (or the first octet of an Extended-Length field SHOULD be non-zero). The largest allowed value of an Extended-Length field is 4084, considering maximum SXP message length of 4096. The largest allowed value of a non-Compact attribute Length field is thus 4080. SXP Listener MUST accept and process Compact Extended-Length attributes with Length field in the range [0..255] without issuing any errors. In the rest of this document diagrams showing usage of Compact normal and/or Extended-Length attribute are only illustration of expected usage. They do not imply that only the illustrated class of attribute or attribute Length is supported.

Attribute Value field consists of well defined sub-fields which are defined in this protocol spec. Attribute Value field MAY contain additional, possibly optional, sub-attributes which are encoded like other attributes.

4.4. SXP OPEN and OPEN_RESP Message



Field	Length	Description
Version	4	Unsigned integer value. The SXP Version is set to the highest version that is supported on the network device
SXP Mode	4	Unsigned integer value used to convey the SXP role that the device uses on the SXP connection. The following values are used: 1 - SXP Speaker 2 - SXP Listener
SXP Node-Id	7	REQUIRED non-transitive attribute (in SXP version 4 and above) an SXP Listener SHALL include in OPEN or OPEN_RESP to convey its unique 32 bits Node ID. The attribute type used for Node ID Attribute is 6
Capabilities	Variable	REQUIRED non-transitive attribute (in SXP version 4 and above) an SXP Listener SHALL include in OPEN or OPEN_RESP to convey its capabilities for processing data on the connection

SXP Node-ID with value 0 is reserved for denoting bindings that are received from connections that use older versions of the SXP protocol.

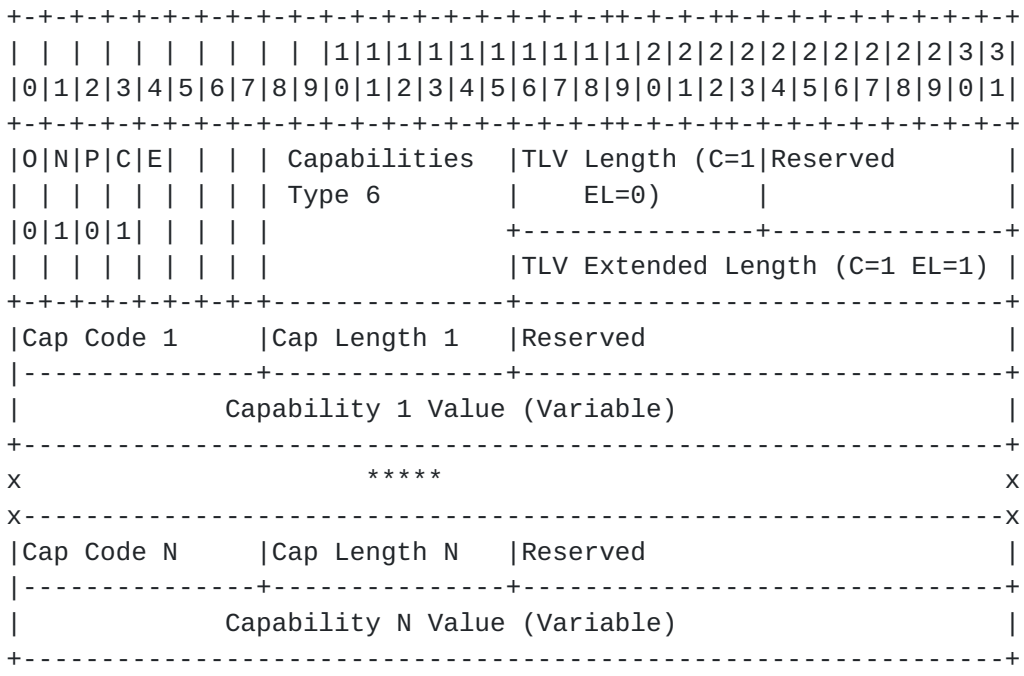
4.4.1. Capabilities Advertisement

SXP Listeners are REQUIRED to include the Capabilities attribute when they send an OPEN message or when they respond to such message with OPEN_RESP message. The Capabilities attribute describes the SXP

features supported by the Listener. This allows the SXP Speaker to tailor the behavior of the connection to what the Listener could process. An SXP speaker receiving Capabilities attribute from an SXP Listener SHALL NOT send to that listener in UPDATE messages information which was not indicated as supported in the Capabilities attribute. If an SXP speaker receives from its listener peer a capability that it does not itself support or recognize, it MUST ignore that capability. In particular, ERROR message MUST NOT be generated and the SXP session MUST NOT be terminated in response to reception of a capability that is not supported by the local SXP speaker.

4.4.1.1. Capabilities Attribute

This is a REQUIRED attribute that is used by an SXP Listener to convey to its peer SXP Speaker the list of capabilities supported by the listener. The attribute contains one or more triples <Capability Code, Capability Length, Capability Values>:



Field	Length	Description
Capability Code	1	Unsigned integer that unambiguously identifies individual capabilities
Capability Length	1	Unsigned integer that contains the length of the Capability Value field in octets
Capability Value	Variable	Variable-length field that is interpreted according to the value of the Capability Code field. Could have zero length

The encoding and meaning of the capabilities defines by this version of the SXP protocol spec are as follows:

Capability Name	Capability Code	Capability Length	Description
IPv4 Unicast	1	0	Listener is capable of receiving and handling IPv4 unicast bindings
IPv6 Unicast	2	0	Listener is capable of receiving and handling IPv6 unicast bindings
Subnet Bindings	3	0	Listener is capable of receiving and handling subnet bindings

4.4.1.2. Handling of Capabilities Attribute

Subnet Bindings - This capability indicates to the speaker that subnet bindings SHOULD be exported as regular bindings without any expansion. A listener that fully supports bindings associated with IP prefixes other than host addresses (IPv4 /32 or IPv6/128) MUST include the Subnet Bindings capability.

A listener that does not support subnet bindings in its forwarding path but has the capability to expand subnet bindings locally MUST send Subnet Bindings capability in order to prevent the listener from unnecessarily expanding subnet bindings. A listener which does not support subnet bindings but might be able to support the volume of expanded bindings MAY include Subnet Bindings capabilities.

A speaker receiving such capability MAY expand subnet bindings exhaustively or selectively and export individual bindings within a subnet. The expansion of subnet bindings COULD be limited by

implementation to a reasonable subnet size with a limit on the total number of bindings expanded. Therefore there could be subnet bindings which are not expanded.

When a speaker does not expand a subnet binding it SHOULD export it unmodified as a subnet binding. This will allow the listener to propagate such bindings even when it cannot use it locally.

When a speaker expands a subnet binding it MUST NOT export it as both subnet and expanded bindings since the listener or any further SXP peers along the SXP propagation path cannot distinguish expanded bindings from other host bindings and relate them to the subnet bindings they are originate from.

IPv6 Unicast - This capability indicates to the speaker that the listener is capable of handling and using bindings associated with IPv6 unicast addresses. A listener that support IPv6 bindings MUST include this capability. When this capability is not included, the speaker MUST NOT send IPv6 bindings within well-known attributes. A speaker, however, SHOULD send IPv6 bindings within Optional Transitive attributes when the IPv6 Unicast capability is omitted. This will allow an SXP node to relay IPv6 bindings even when it could not process them locally.

IPv4 Unicast - This capability should be handled in the same way as IPv6 Unicast capability is handled. It is unlikely for any SXP listener to omit IPv4 Unicast capability at least until pure IPv6 networks would become common.

4.4.2. Keepalive and Hold Time Negotiation

SXP uses TCP-based, keep-alive mechanism to determine if a connection is live. This mechanism has been used in all prior versions of the protocol. SXP version 4 adds an optional negotiated keep-alive mechanism within the protocol itself in order to provide more predictable and timely detection of connection loss.

SXP connections are asymmetric with almost all of the protocol messages (except for OPEN/OPEN_RESP and ERROR) being sent from an SXP speaker to an SXP listener.

SXP listener could also keep potentially large volume of state per connection which includes all the binding information learnt on a connection. Therefore, it is only meaningful to have a keep-alive mechanism that allows a listener to detect the loss of connection with a speaker.

The mechanism is based on two timers:

Hold Timer - Used by a listener for detection of elapsing time without successive KEEPALIVE and/or UPDATE messages from a speaker.

Keep-alive Timer - Used by a speaker to trigger sending of KEEPALIVE messages during intervals when no other information is exported via UPDATE messages.

The keep-alive timer is one-thirds of the negotiated hold-timer value.

Hold Timer for the keep-alive mechanism MAY be negotiated during the OPEN/OPEN_RESP exchange at connection setup.

- A listener MAY have desirable range for Hold Time period locally configured or a default of [90..180] seconds. A value of [0xFFFF..0xFFFF] implies that the keep-alive mechanism is not used.

- A speaker MAY have a minimum acceptable Hold Time period locally configured or a default of 120 seconds. This is the shortest period of time a speaker is willing to send KEEPALIVE messages for keeping the connection alive. Any shorter Hold Time period would require a faster KEEPALIVE rate from the rate the speaker is ready to support. A value of 0xFFFF implies that the keep-alive mechanism is not used

- The negotiation succeeds when the speaker's minimum acceptable Hold Time falls below or within the desirable Hold Time range of the listener. If one end turns off the keep-alive mechanism, the other end should also turn it off to make the negotiation success.

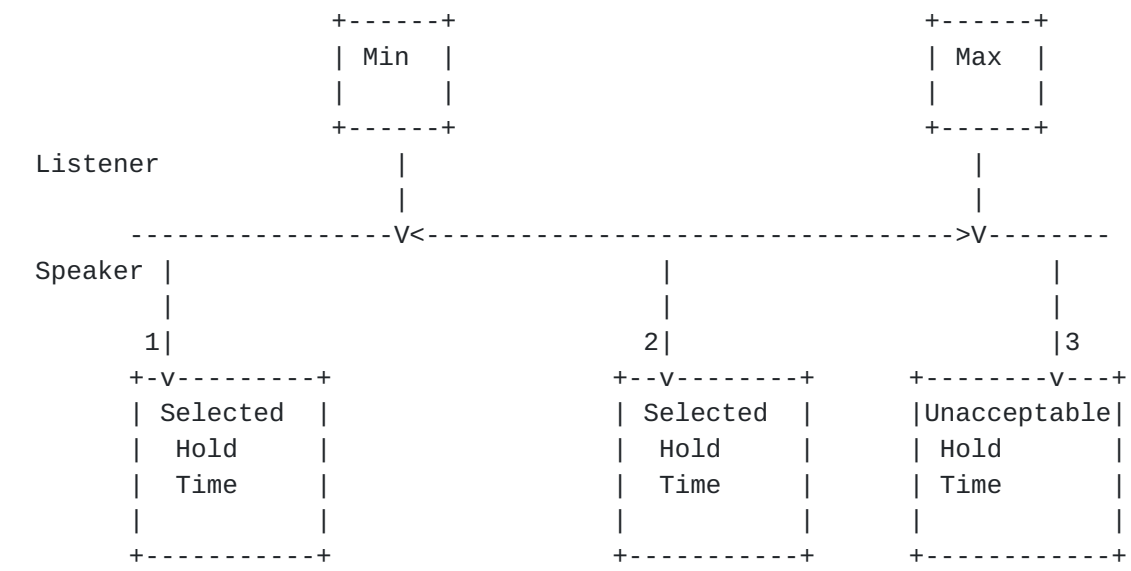
- The negotiation fails when the speaker's minimum acceptable Hold Time is greater than the upper bound of the listener's Hold Time range.

- The selected Hold Time period of a successful negotiation is the maximum of the speaker's minimum acceptable Hold Time and the lower bound of the listener's Hold Time range.

- The speaker calculates the Keep-alive Time to 1/3 of the selected Hold Time by default unless a different Keep-alive Time is locally configured.

- The negotiation process is designed such that the outcome is the same regardless of the mode of the initiator.

The diagram below illustrates the Hold Time negotiation process.



Connection initiated by Listener

- A listener MAY include a Hold-Time Attribute in the OPEN message with minimum and maximum values set to its configured range of Hold Time period. Hold-Time Attribute with just a minimum value set to 0xFFFF would indicate to the speaker that the keep-alive mechanism is not used.

- When a speaker received an OPEN message it will react as follows:

  * If the Hold-Time attribute is not present or if it contains a minimum value that is set to 0xFFFF, the speaker will set its Keepalive Time to 0xFFFF to indicate that keep-alive mechanism is disabled.

  * If the received Hold-Time attribute contains a valid range, the speaker MUST include Hold-Time attribute in its OPEN_RESP message with a minimum value set as follows:

    + 0xFFFF if the speaker does not support the keep-alive mechanism or if the mechanism is supported but disabled due to local configuration which sets the Keep-alive Time to 0xFFFF

    + If the speaker's minimum acceptable Hold Time value is greater than the upper bound of the offered range, the speaker MUST send and Open ERROR message with sub-code set to Unacceptable Hold Time and terminate the connection.

        + Otherwise the speaker's will set the selected Hold Time to the maximum of its minimum acceptable Hold Time value and the lower bound of the offered Hold Time range.

        + The speaker will calculate a new value for its Keep-alive Time as 1/3 of that selected Hold Time.

        + The speaker will set the minimum Hold Time value of the Hold Time attribute to the selected Hold Time.

  o When the listener receives the OPEN_RESP from the speaker it will look for Hold Time Attribute:

    * If Hold-Time attribute is present and contains a minimum Hold Time value of 0xFFFF. The speaker will set its Hold Time value to 0xFFFF to indicate that keep-alive mechanism is not used.

    * If the minimum Hold Time value is within the range offered by the listener, the listener will set its Hold-Time period to the selected value it has received in the OPEN_RESP.

    * If the minimum Hold Time value is outside the offered range, the listener will send Open ERROR message with sub-code set to Unacceptable Hold Time and terminate the connection.

  Connection initiated by Speaker

  o A speaker MAY include a Hold-Time Attribute in the OPEN message with minimum value set to its minimum acceptable Hold Time period. Hold-Time Attribute with just a minimum value set to 0xFFFF would indicate to the listener that the keep-alive mechanism is not used.

  o When a listener receives an OPEN message it will react as follows:

    * If the Hold-Time attribute is not present or if it contains a minimum value that is set to 0xFFFF, the listener will set its Hold Time to 0xFFFF to indicate that keep-alive mechanism is disabled.

    * If the received Hold-Time attribute contains a valid value, the speaker MUST include Hold-Time attribute in its OPEN_RESP message with a minimum value set as follows:

      + 0xFFFF if the listener does not support the keep-alive mechanism or if the mechanism is supported but disabled due to local configuration which sets the Keep-alive Time to 0xFFFF

+ If the received Hold Time value is greater than the upper bound of the listener's configured Hold Time range, the speaker MUST send and Open ERROR message with sub-code set to Unacceptable Hold Time and terminate the connection.

+ If the received Hold Time value falls within the listener's configured Hold Time range, the listener will make it the selected Hold Time.

+ If the received Hold Time value is less than the lower bound of the listener's configured Hold Time range, the listener will set the selected Hold Time to the lower bound of its Hold Time range.

+ The listener will set the minimum Hold Time value of the Hold Time attribute to the selected Hold Time.

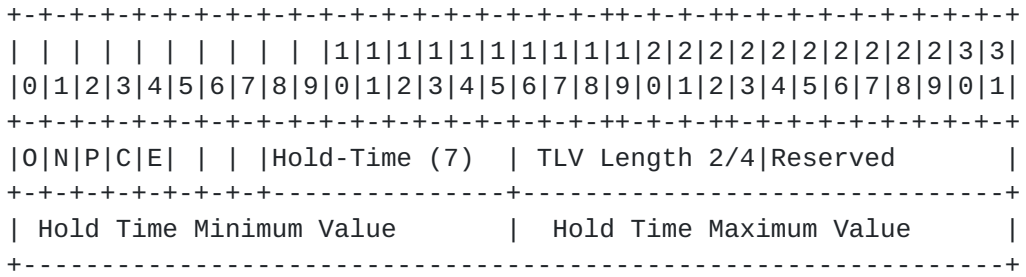
o When the speaker receives the OPEN_RESP from the listener it will look for Hold Time Attribute:

* If Hold-Time attribute is present and contains a minimum Hold Time value of 0xFFFF. The speaker will set its Hold Time value to 0xFFFF to indicate that keep-alive mechanism is not used.

* If the received Hold Time value is greater or equal to the speaker minimum acceptable Hold Time, the speaker will calculate a new value for its Keep-alive Time as 1/3 of that received Hold Time.

* If the received Hold Time value is lower than the minimum acceptable Hold Time, the speaker MUST send and Open ERROR message with sub-code set to Unacceptable Hold Time and terminate the connection.

4.4.2.1. Hold-Time Attribute

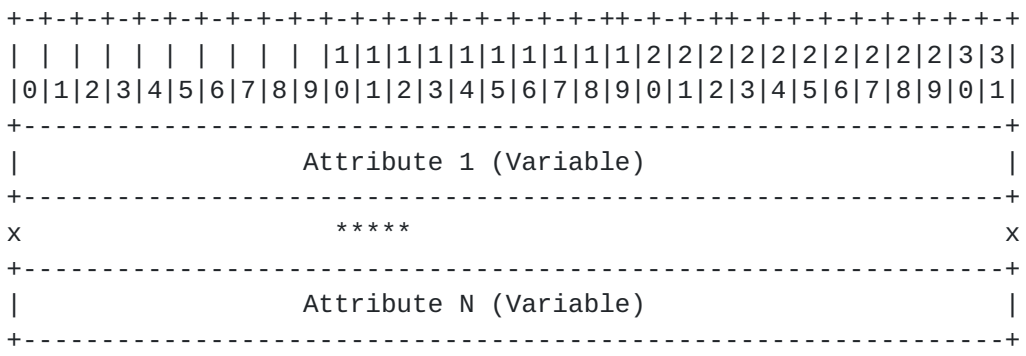


Field	Length	Description
Optional(C)	1 Bit	Value is 0
Non-Transitiv(N)	1 Bit	Value is 0
Partial (P)	1 Bit	Value is 0
Compact (C)	1 Bit	Value is 1
Extended Length (E)	1 Bit	Value is 0
Attribute Type	1	Hold-Time (7)
Attribute Value Length	1	2 - when only Hold Time Minimum Value is present. 4 - when both Hold Time Minimum and Maximum values are present
Hold Time Minimum Value	2	Unsigned integer indicating the number of seconds the sender proposes as a lower bound for the Hold Timer period or the Hold Time period selected by a responder. 0 or at least 3 seconds
Hold Time Maximum Value	2	Unsigned integer indicating the number of seconds the sender proposes as a upper bound for the Hold Timer period. MUST be greater than Hold Time Minimum Value

Hold-Time is a well-known discretionary attribute an initiator of a connection MAY include in an OPEN message and a responder MAY be REQUIRED to include in an OPEN_RESP in order to negotiate the timer periods used for keep-alive mechanism.

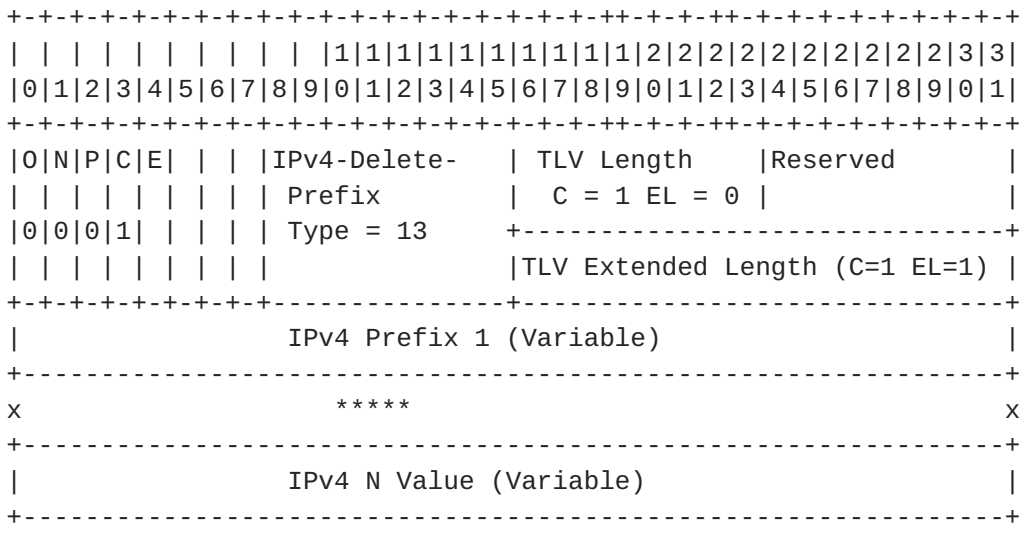
4.5. SXP UPDATE Message

Prior versions of SXP protocol specify that an UPDATE message contains one or more SXP mapping records. This version of SXP protocol generalizes the UPDATE message. An UPDATE message MAY contain one or more attributes. Some attributes are well-known and additional attributes COULD be added in the future without affecting existing implementations of SXP version 4. This document defines the format of the attributes used within UPDATE message and how they are combined to carry the SXP binding information.

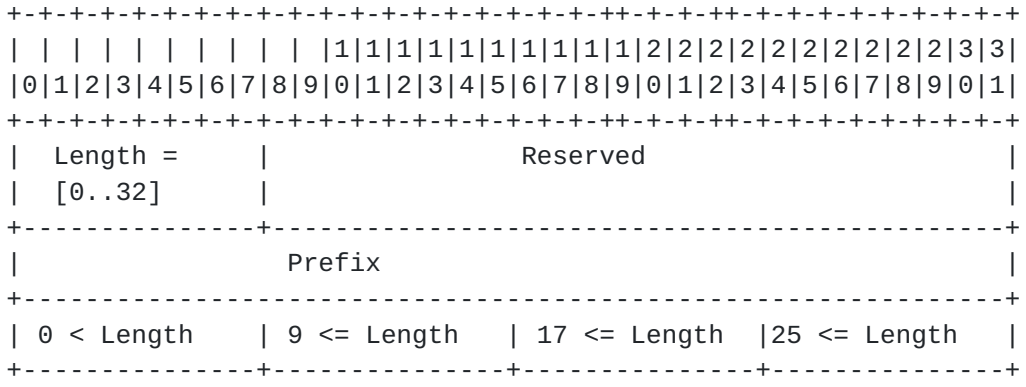


4.5.1. UPDATE Attributes

4.5.1.1. IPv4-Delete-Prefix Attribute

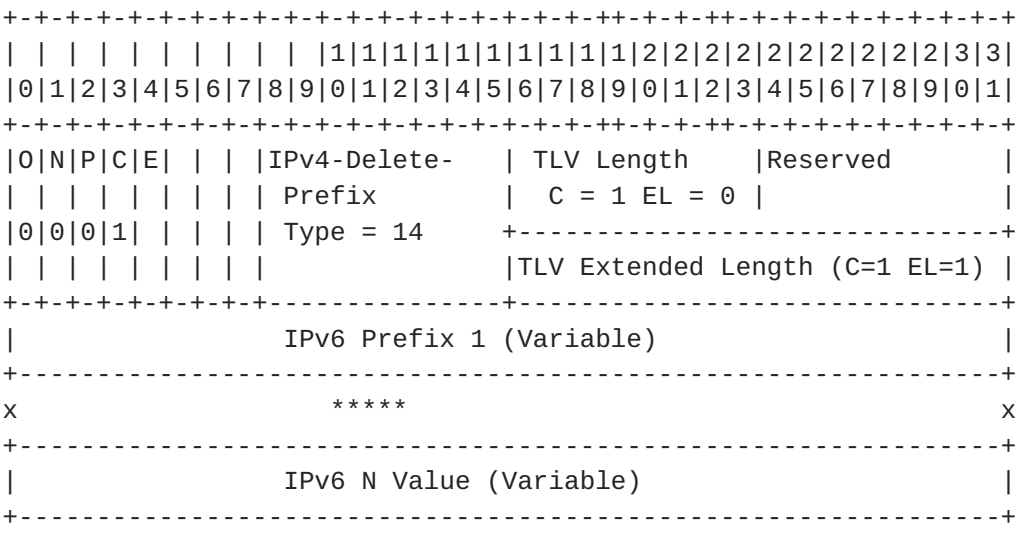


This attribute is used to convey the withdrawal of one or more IPv4 bindings which are unambiguously identified by their IPv4 prefix. This attribute contains a set of IPv4 prefixes. Each IPv4 prefix is encoded as a 2-tuple of the form <length, prefix>, whose fields are encoded as:

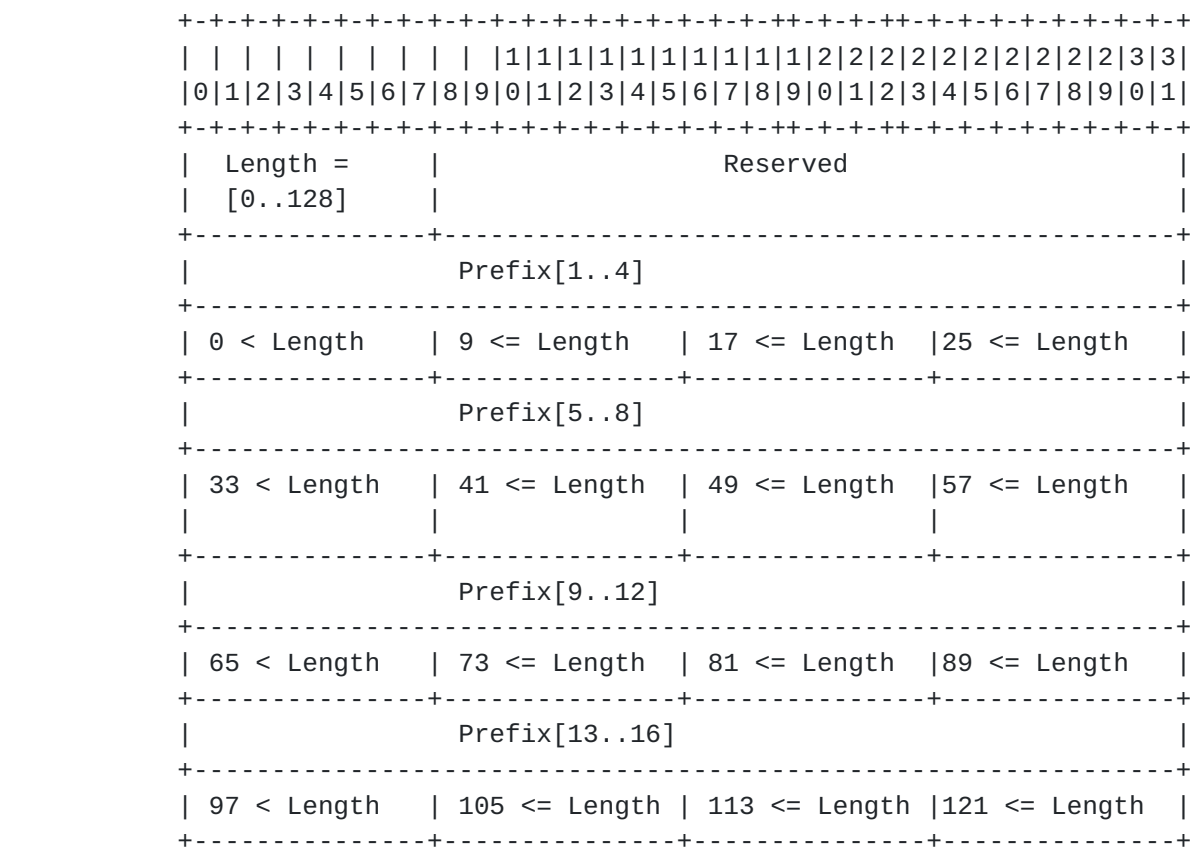


Field	Length (in Octets)	Description
Length	1	Unsigned integer. The length in bits of the IP address prefix. A length of zero indicates a prefix that matches all IP addresses (with Prefix field of zero octets)
Prefix	Variable ⎡Length/8⎤	An IP address prefix, followed by the minimum number of trailing bits needed to make the end of the field fall on an octet boundary. The value of trailing bits is irrelevant

4.5.1.2. IPv6-Delete-Prefix Attribute

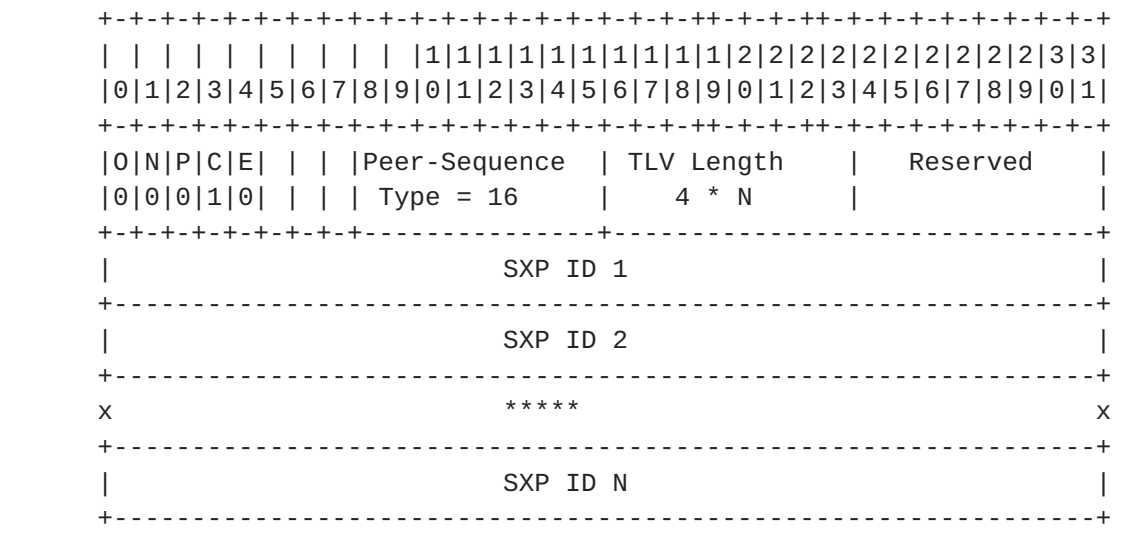


This attribute is used to convey the withdrawal of one or more IPv6 bindings which are unambiguously identified by their IPv6 prefix. This attribute contains a set of IPv6 prefixes. Each IPv6 prefix is encoded as a 2-tuple of the form <length, prefix>, whose fields are encoded as:



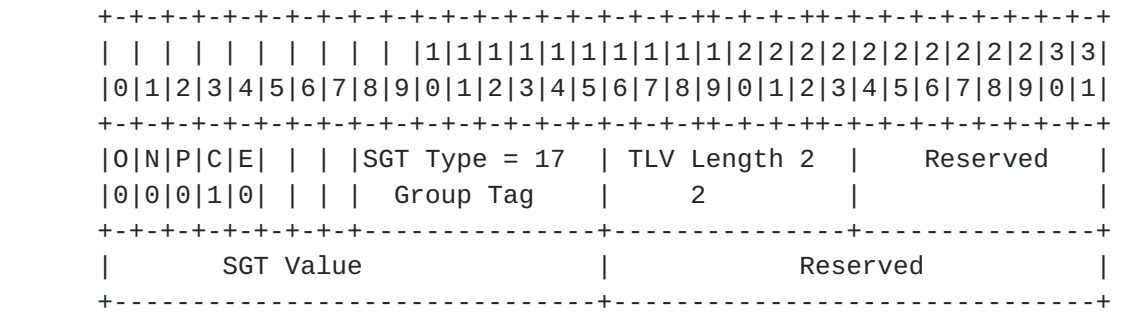
Field	Length (in Octets)	Description
Length	1	Unsigned integer. The length in bits of the IP address prefix. A length of zero indicates a prefix that matches all IP addresses (with Prefix field of zero octets)
Prefix	Variable ⎡Length/8⎤	An IP address prefix, followed by the minimum number of trailing bits needed to make the end of the field fall on an octet boundary. The value of trailing bits is irrelevant

4.5.1.3. Peer-Sequence Attribute



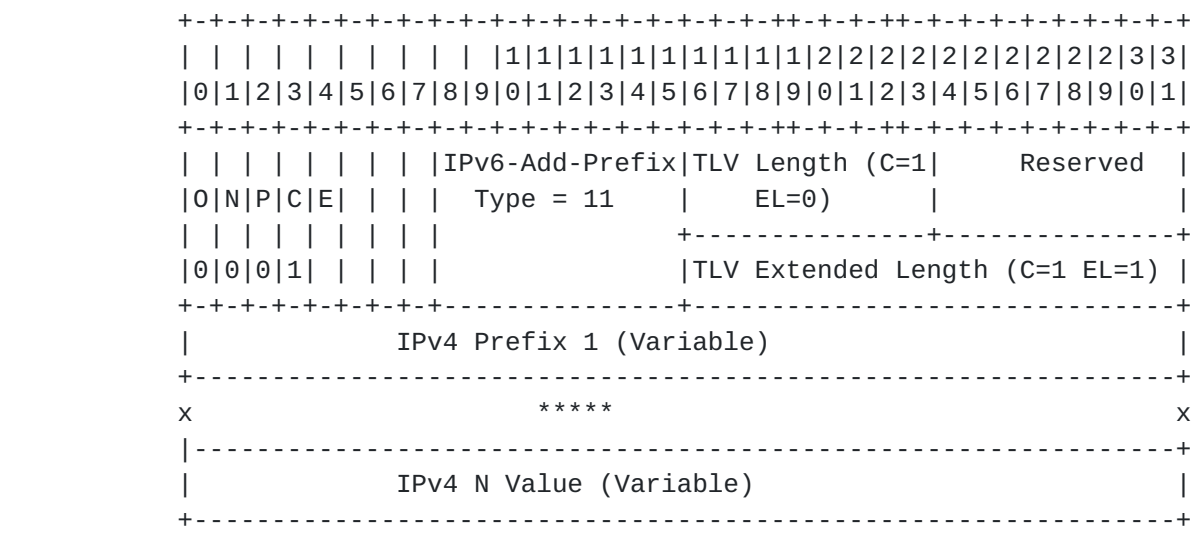
This attribute contains the SXP nodes through which exported bindings has traversed. Each SXP node is identified by a 4 octet SXP ID. SXP ID MUST be unique for every SXP node in an SXP deployment. As exported bindings are propagated by an SXP Speaker, the speaker SHALL prepend its own SXP ID to the sequence of SXP IDs of the Peer-Sequence attribute that was received by an SXP Listener on the same node. If the exported binding is originated by the local node, the Peer-Sequence will contain a single SXP ID identifying the local node. When an UPDATE message is received by an SXP Listener, the first SXP ID within each Peer-Sequence attribute is the SXP node ID of the SXP Speaker on the remote end of the connection the message is received on. The last SXP ID is the node ID of the SXP Speaker that originates the bindings associated with a Peer-Sequence attribute. UPDATE message SHALL contain one Peer-Sequence attribute and MAY contain multiple Peer-Sequence attributes. Each Peer-Sequence attribute is associated with the bindings that follow it up to the next Peer-Sequence attribute if more than one Peer-Sequence attribute is present or up to the end of the UPDATE message.

4.5.1.4. Source-Group-Tag Attribute



Field	Length (in Octets)	Description
Attribute Type	1	Source-Group-Tag (17)
Attribute Value Length	1	
SGT Value	2	Unsigned integer value of an SGT

4.5.1.5. IPv4-Add-Prefix Attribute



This attribute is used to convey the binding of one or more IPv4 prefixes with an SGT value or other information. The encoding of each IPv4 prefix is described above in IPv4-Delete-Prefix Attribute section.

The unbundling of IP prefixes from the information associated with them achieves two major objectives:

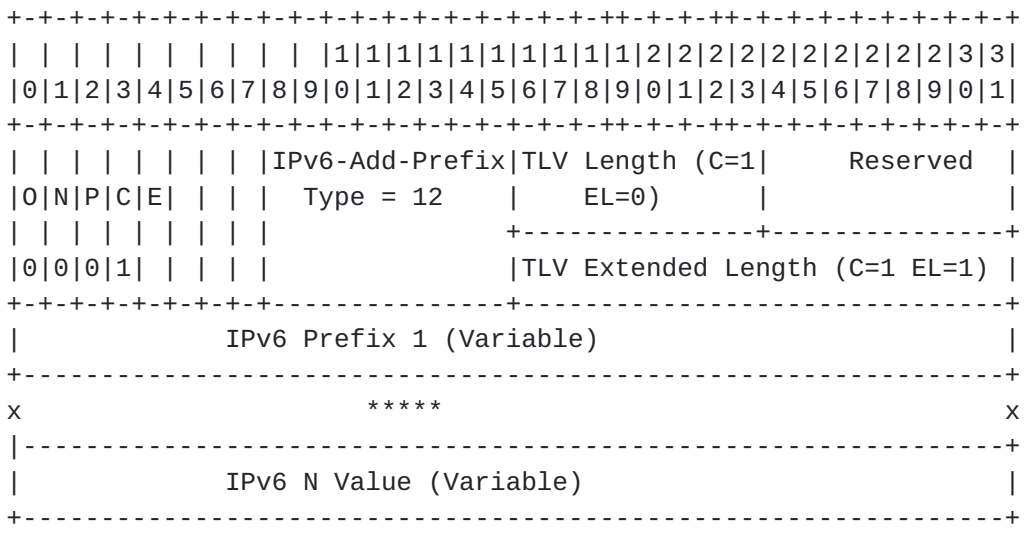
- Allows for efficient sharing of common attributes such as Peer-Sequence, and Source-Group-Tag with multiple IP prefixes.

- Allows for future introduction of new information for association with IP prefixes via optional attributes.

The associated SGT is the value specified by the latest preceding occurrence of Source-Group-Tag attribute within the same UPDATE message. An IPv4-Add-Prefix attribute SHALL be preceded with an occurrence of Peer-Sequence attribute. The latest preceding

occurrence provides the path along which the bindings of all the prefixes contained in the IPv4-Add-Prefix have been traversed.

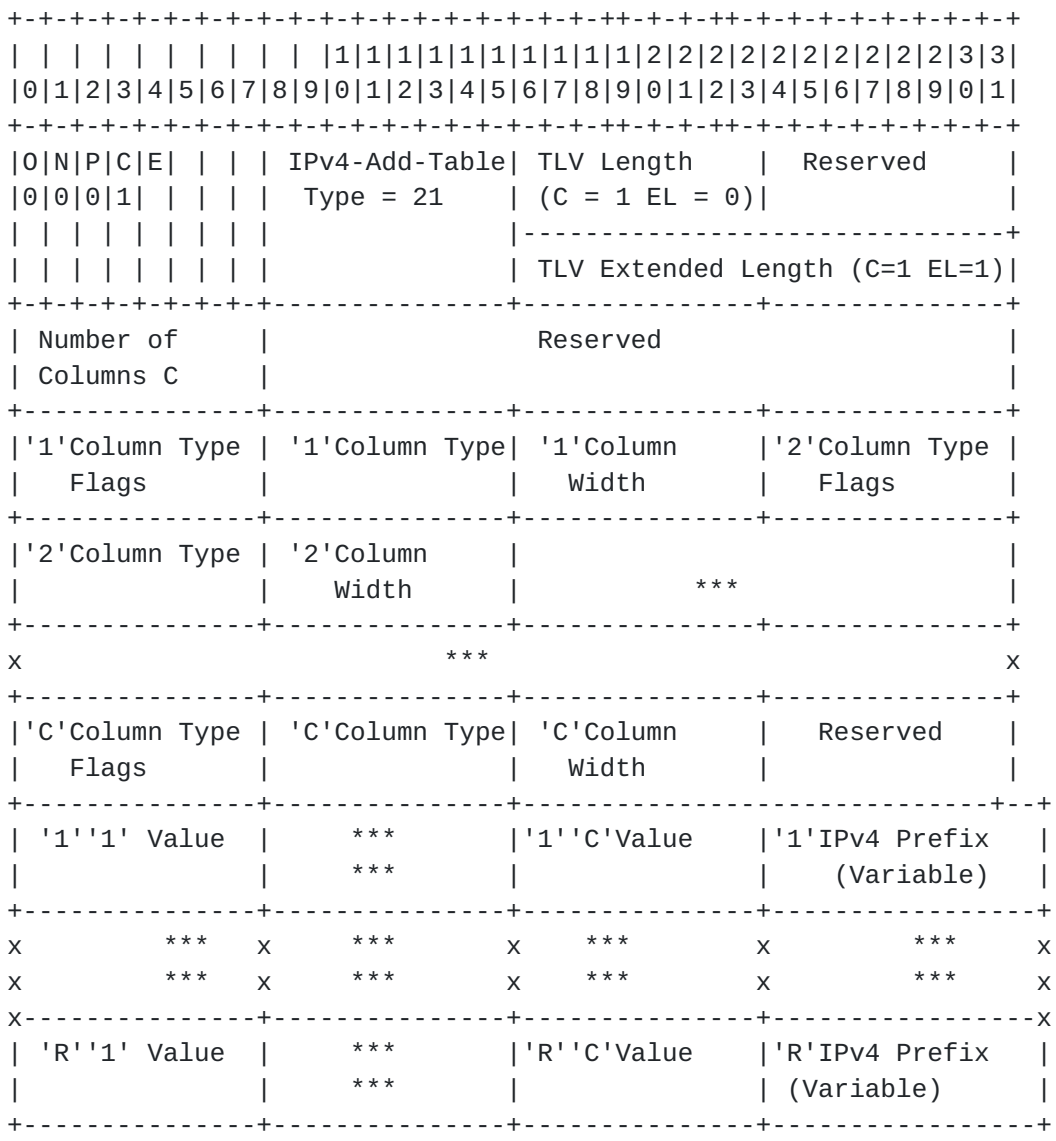
4.5.1.6. IPv6-Add-Prefix Attribute



This attribute is used to convey the binding of one or more IPv6 prefixes with an SGT value or other information. The encoding of each IPv6 prefix is described above in IPv6-Delete-Prefix Attribute section. The handling of this attribute and its relationship with preceding attributes such as Peer-Sequence or Source-Group-Tag are the same as what is described above in IPv4-Add-Prefix section except for a different attribute type (12) and possibly longer IPv6 prefixes.

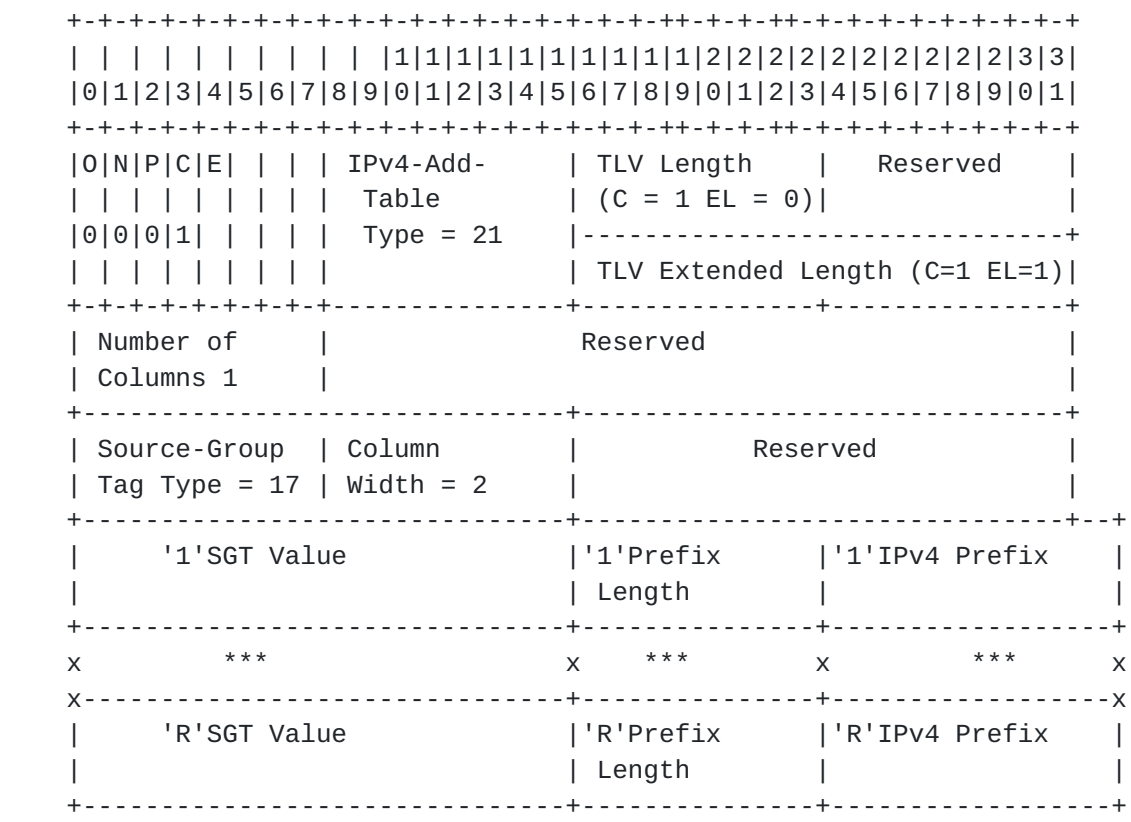
4.5.1.7. IPv4-Add-Table Attribute

This attribute provides a flexible tabular representation of bindings information. It is provided for efficient aggregation of multiple bindings information when an implementation cannot aggregate multiple IP prefixes that are associated with the same SGT or other common attributes.


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
O	N	P	C	E			IPv4-Add-Table	TLV Length	Reserved
0	0	0	1				Type = 21	(C = 1 EL = 0)	
TLV Extended Length (C=1 EL=1)
Number of Columns C
Reserved
'1'Column Type Flags
'1'Column Type
'1'Column Width
'2'Column Type Flags
'2'Column Type
'2'Column Width
***
'C'Column Type Flags
'C'Column Type
'C'Column Width
Reserved
'1''1' Value
'1''C'Value
'1'IPv4 Prefix (Variable)
'R''1' Value
'R''C'Value
'R'IPv4 Prefix (Variable)

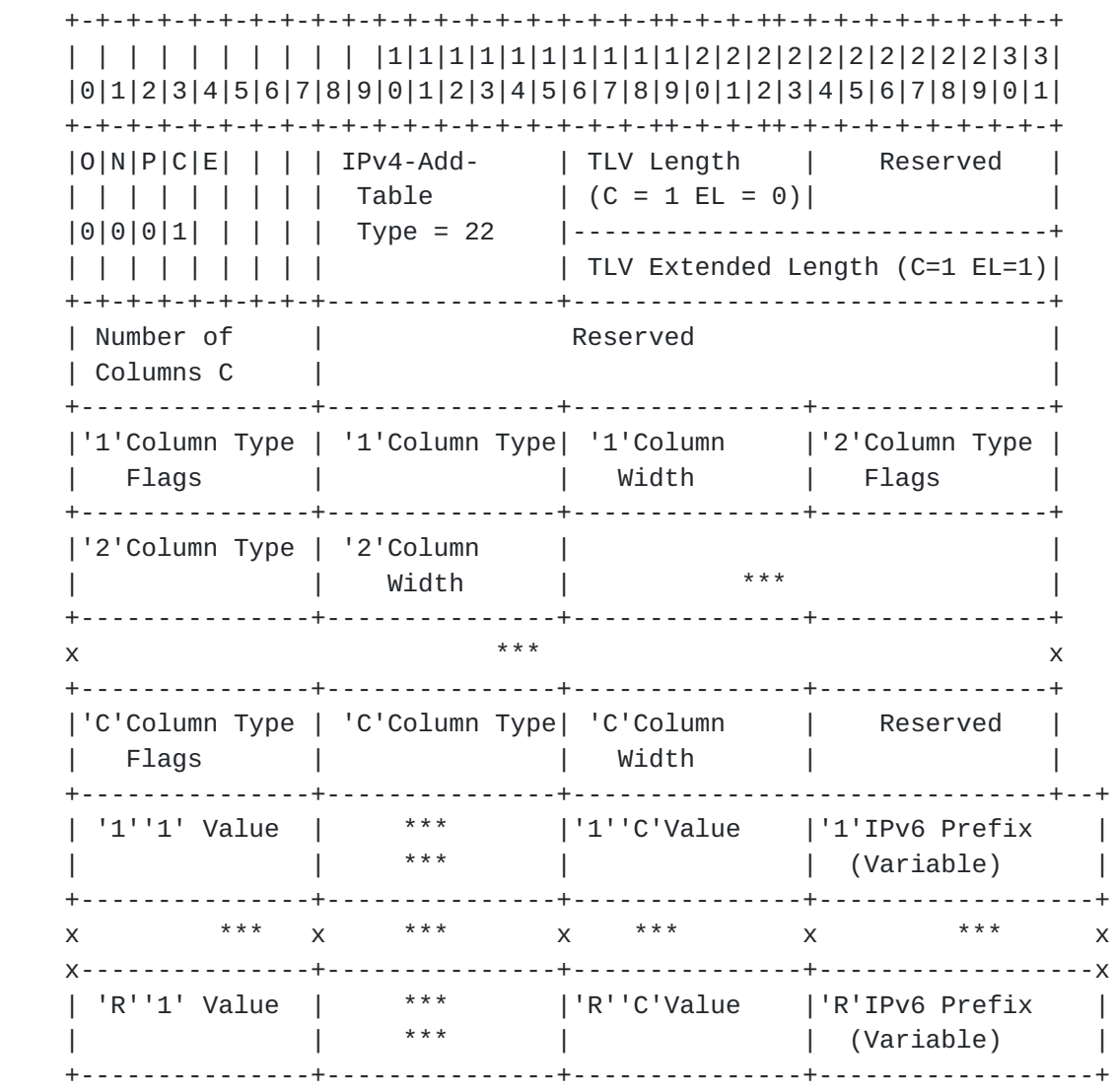
Field	Bits	Description
Number of Columns	1	1 - Unsigned integer which specifies the number of columns of information (not including the trailing IP prefix) associated with each prefix
[i] Column Type Flags	1	0 - An Attribute Type Flags value defining the flag of the attribute in the I'th column on each row
[i] Column Type	1	1 - An Attribute Type value defining what each row have in the i'th column position
[i] Column Width	1	1 - The length in octets of the attribute value in the i'th column on each row
[j] IPv4 Prefix	Variable	0 - An IP prefix which is associated with the C-tuple of attribute values on the j'th row of the table. The format of is the same as the IP prefix field in the IPv4-Add-Prefix attribute

Below is an example of an IPv4-Add-Table attribute which contain a single SGT attribute value on each row.



4.5.1.8. IPv6-Add-Table Attribute

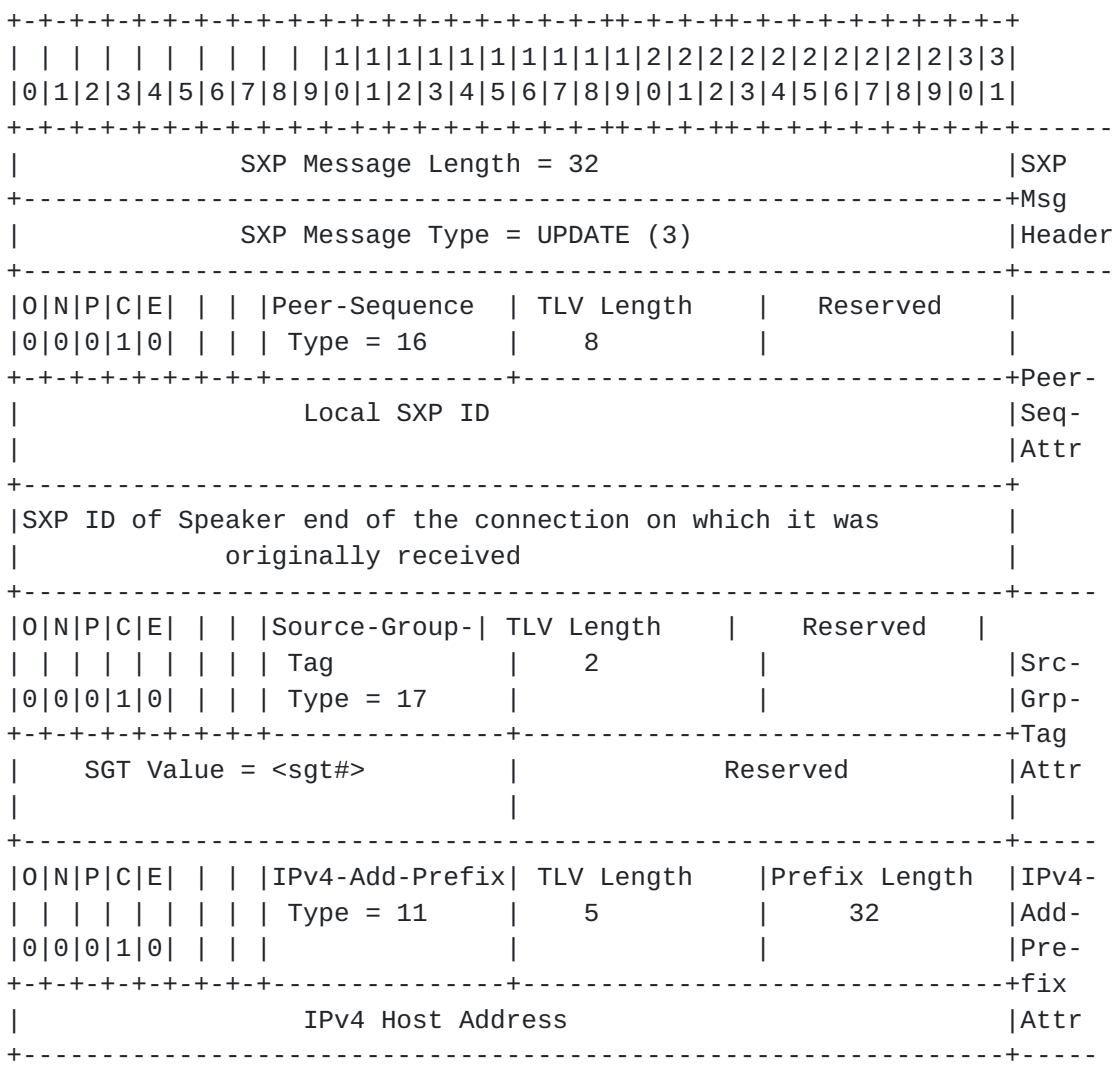
This attribute provides a flexible tabular representation of IPv6 bindings information.



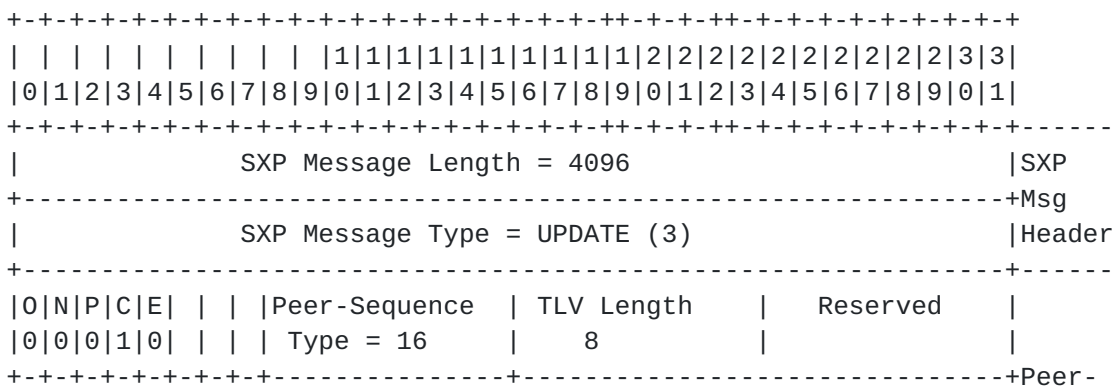
The format and handling of IPv6-Add-Table attribute are the same as what is described above in IPv4-Add-Table section except that the attribute type (22) is used and the trailing IPv6 Prefix field in each row could be potentially longer than the corresponding trailing IPv4 Prefix fields.

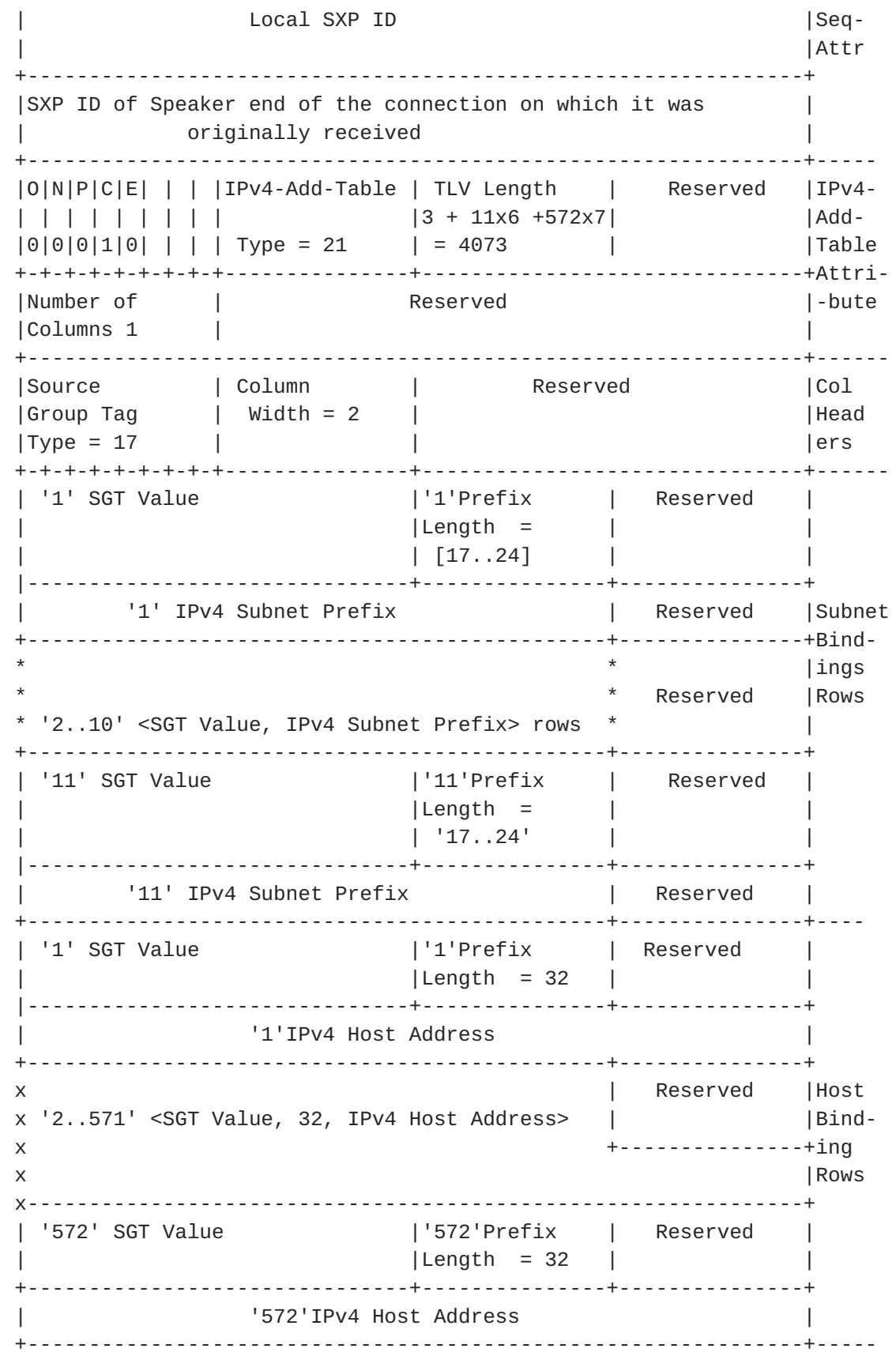
4.5.2. UPDATE Message Samples

Single IPv4 host binding as exported by the first Speaker away from the originating node:

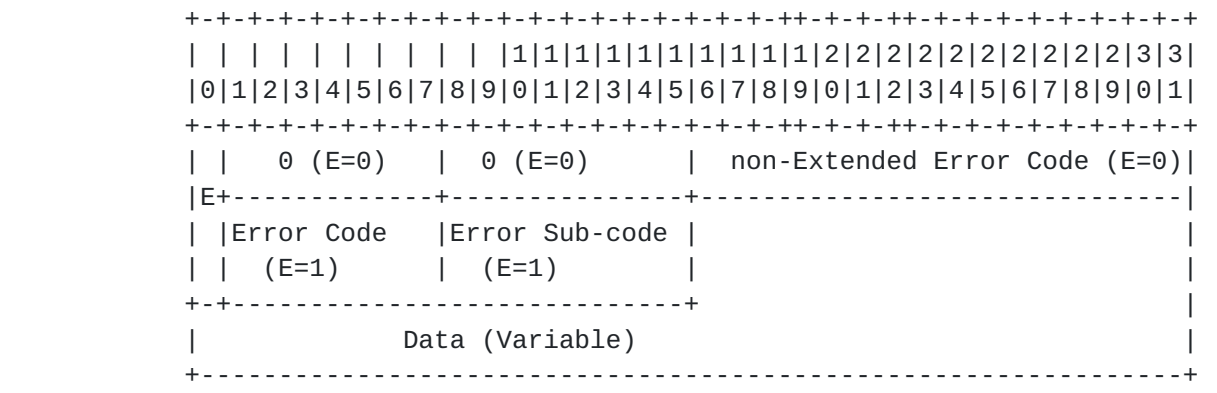


Multiple IPv4 host and subnet bindings sharing the same Peer-Sequence as exported by the first Speaker away from the originating node. 11 subnets bindings of length between 17 and 24 and as many host bindings as SXP total message size allowed are packed into this message:




Local SXP ID
Seq-Attr
SXP ID of Speaker end of the connection on which it was originally received
O|N|P|C|E| | | IPv4-Add-Table Type = 21
TLV Length 3 + 11x6 +572x7 = 4073
Reserved
IPv4-Add-Table Attri-bute
Number of Columns 1
Reserved
Source Group Tag Type = 17
Column Width = 2
Reserved
Col Headers
'1' SGT Value
'1'Prefix Length = [17..24]
Reserved
'1' IPv4 Subnet Prefix
Subnet Bindings Rows
'2..10' <SGT Value, IPv4 Subnet Prefix> rows
'11' SGT Value
'11'Prefix Length = '17..24'
'11' IPv4 Subnet Prefix
'1' SGT Value
'1'Prefix Length = 32
'1'IPv4 Host Address
'2..571' <SGT Value, 32, IPv4 Host Address>
Host Binding Rows
'572' SGT Value
'572'Prefix Length = 32
'572'IPv4 Host Address

4.6. SXP ERROR Message



Field	Bits	Length (in Octets)	Description
Extended	0		1 - Extended Error format, which includes code, sub-code, and variable data 0 - Legacy non-Extended Error
Error Code	[1..7]	0	Unsigned integer in the range [0..127] - indicates the type of ERROR
Error Sub-Code		1	Unsigned integer in the range [0..255] - Provides extended information about the nature of the reported error. Each Error code may have one or more Error Sub-codes associated with it. If no appropriate Error Sub-code is defined, then zero (Unspecified) value is used
non-Extended Error Code		2	Error code values: 0 - No error 1 - Version Mismatch 2 - Message Parse Error
Data		Variable Message Length - 10	Additional data for diagnosing the reason for the ERROR message. The content depends on the values of Error Code and Error Sub-code

4.6.1. Error Codes

Error Code	Error Sub-code	Symbolic Name
1	-	Message Header Error
2	-	OPEN Message Error
3	-	UPDATE Message Error
	1	Malformed Attribute List
	2	Unrecognized Well-known Attribute
	3	Missing Well-known Attribute
	4	Attribute Flags Error
	5	Attribute Length Error
	6	Malformed Attribute
	7	Optional Attribute Error
	8	Unsupported Version Number
	9	Unsupported Optional Attribute
	10	Unacceptable Hold Time

4.7. SXP PURGE-ALL Message

There is no payload corresponding to this message type.

When an SXP connection on which the local SXP node is a speaker is administratively deleted, SXP MUST send a PURGE-ALL message to the SXP listener at the remote end. This will provide the speaker timely knowledge that the connection is about to be torn down and that it is not an intermittent loss of communication. The listener will immediately delete all bindings that were received on that connection and will not go through Delete Hold-Down timer before removal of bindings.

When SXP feature gets administratively disabled, SXP MUST send PURGE-ALL message on all the connections on which the local end is a speaker.

Upon receiving a PURGE-ALL message, an SXP speaker MUST immediately delete all the bindings that were received on that connection. PURGE-ALL message should have the same effect as an UPDATE message with IPv4-Delete-Prefix or IPv6-Delete-Prefix attributes that contain all bindings which has been exported on the connection. A PURGE-ALL is simply an optimization of such delete at-once case.

4.8. SXP KEEPALIVE Message

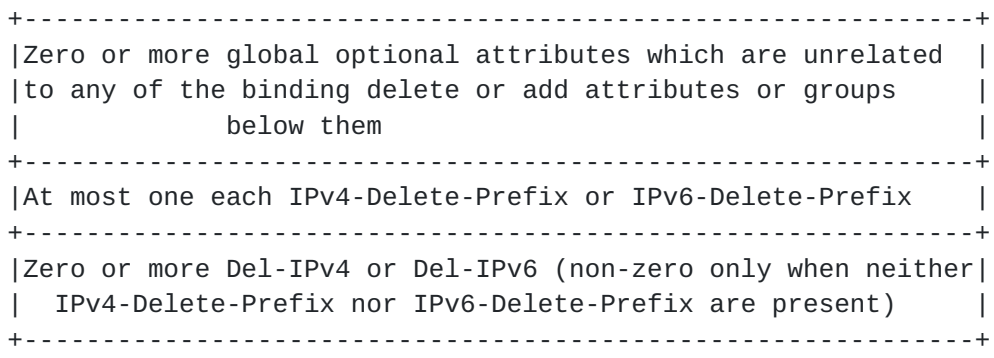
There is no payload corresponding to this message type. This message is send by the speaker to the listener side when KEEPALIVE timer expires.

5. Update Message Handling

An SXP UPDATE message may be received only in the ON state. Receiving an UPDATE message in any other state is an error. When an UPDATE message is received, each field is checked for validity.

5.1. UPDATE Message Validation

UPDATE message SHALL be organized as illustrated below:



All errors detected while processing the UPDATE message MUST be indicated by sending the ERROR message with the Error Code UPDATE Message Error to the SXP speaker from which the UPDATE message was received. The Error Sub-code provides additional information about the error. UPDATE message is a sequence of attributes. Attributes could be further classified as compact, compact with extended length, or non-compact attributes. Each attribute consists of a fixed-size header, 3, 4, or 8 octets long respectively, which contains TLV Length field. The total length of each attribute is thus: Attribute Length = 3/4/8 + TLV Length If the length of any attribute is larger than the SXP Message length or the sum of Attribute Length of all attributes is larger than the SXP Message length, the Error Sub-code MUST be set to Malformed Attribute List. If any recognized attribute has Attribute Flags that conflicts with its Attribute Type, then the Error Sub-code MUST be set to Attribute Flags Error. The Data field MUST contain the erroneous attribute (flags, type, length, and value). If any recognized attribute has Attribute Length that conflicts with the expected length (based on its Attribute Type), then the Error Sub-code MUST be set to Attribute Length Error. The Data field MUST contain the erroneous attribute (flags, type, length, and value).

If any recognized attribute does not conform to the attribute specification in UPDATE Attributes section, then Error Sub-code MUST be set to Malformed Attribute. The Data field MUST contain the erroneous attribute (flags, type, length, and value). Cases of malformed attributes:

o IP Prefix length field is outside of the permitted range [0..32]/ [0..128] for IPv4/IPv6 respectively.

o Implied length of IP Prefix field which extends the prefix beyond the extent of the containing attribute.

If any attribute appears more than once when at most one occurrence is permitted then the Error Sub-code MUST be set to Malformed Attribute List.

If any attribute appears in a location where it is not expected (such as IPv4-Add-Prefix/IPv6-Add-Prefix without preceding Source-Group-Tag) then the Error Sub-code MUST be set to Malformed Attribute List.

If an optional non-transitive attribute is unrecognized, it is quietly ignored.

If an optional transitive attribute is unrecognized, the Partial bit (the third high-order bit) in the attribute flags octet is set to 1, and the attribute is retained for export according to the scope in which the attribute appears. A global attribute is exported along every binding specified in this UPDATE message on all connections for which the local peer is a speaker. A path attribute is exported along the bindings from a single path. A per <path, source-group> attribute is exported along bindings received from a single path which share a single Source-Group-Tag attribute. If an optional attribute is recognized and has a valid value, then, depending on the type of the optional attribute, it is processed locally, retained, and updated, if necessary, for possible export to listener peers. A Peer-Sequence attributes is checked for syntactic correction. If the path is syntactically incorrect (e.g. length is not a multiple of 4), then an ERROR message MUST be sent to the speaker with Error Sub-code set to Malformed Attribute. SXP listener learns the SXP Node-ID from the SXP Node-ID in the OPEN or OPEN_RESP it receives at connection establishment. This Node-ID is used to validate subsequent Peer-Sequence attributes from the same peer. SXP MUST check whether the leftmost (with respect to the position of octets in the protocol message) SXP Node-ID in the Peer-Sequence is equal to the SXP Node-ID of the peer that sent the UPDATE message. If the checks determines that this is not the case, then an ERROR message MUST be sent to the speaker with Error Sub-code set to Malformed Attribute.

5.2. UPDATE Message processing

Processing of an UPDATE message by an SXP listener follows the organization of the message according to the following high level steps:

1. UPDATE Message validation as described in section 6.1.

2. Processing of global optional attributes.

3. Processing of binding delete attributes. Binding delete attributes include any of IPv4-Del-Prefix, IPv6-Del-Prefix, Del-IPv4, or Del-IPv6 attributes.

4. Processing path-groups.

A. Process per-path common optional attributes

B. Add-Prefix groups. Each Add-Prefix group starts with a Source-Group-Tag attribute.

i. Process per <path, SGT> optional attributed.

ii. Process IPv4-Add-Prefix attribute

iii. Process IPv6-Add-Prefix attribute

C. Processing IPv4-Add-Table attribute

D. Processing IPv6-Add-Table attribute

E. Processing Add-IPv4 attributes

F. Processing Add-IPv6 attributes

5. Processing trailing optional non-transitive attributes

Each path-group starts with an instance of Peer-Sequence attribute.

Processing Binding Delete Attributes (step 3)

If the UPDATE message contains a non-empty IPv4-Del-Prefix/IPv6-Del-Prefix or one or more Del-IPv4/Del-IPv6 attributes, the previously received bindings from the remote speaker peer, whose IP/IPv6 addresses are contained in any of those attributes, SHALL be removed from the SXP input bindings database. Additional processing of a deleted binding depends on whether it was the SXP contributed binding for its IPv4/IPv6 address and whether it was the selected and exported binding by the master binding data-base:

- Non-contributed binding - No further processing is needed.

- Contributed but not exported - The binding was the preferred binding by SXP and was reported to the master binding database. However, the binding was not exported by SXP due to higher priority contributors in the master binding database. SXP SHALL select a new contribution for the IPv4/IPv6 address of the deleted binding from the input bindings database, and, upon change of SGT or other associated data that are reported to the master bindings database, report the newly selected binding to the master binding database. The newly selected contribution will not affect the exported binding since an existing higher priority contributor is unaffected.

- Contributed and Exported - The binding was the preferred binding by SXP and was the selected binding for the IPv4/IPv6 address by the master binding database. SXP contribution was either the only binding contributor or the highest priority contributor.

SXP SHALL select a new contribution for the IPv4/IPv6 address of the deleted binding among the bindings for that address from other peers.

SXP SHALL apply the following rules for selecting a binding for reporting to the master binding database:

1. Shortest Path rule

Choose a binding which has the shortest Peer-Sequence among all the bindings for the same IP address. A binding received without Peer-Sequence attribute (i.e. from an earlier version speaker) is considered as being received with a Peer-Sequence that contains a single NULL SXP Node-ID (value 0).

1. Most Recently Received rule

If there are more than one bindings with the shortest Peer-Sequence, the binding which has been most recently received is selected.

If the deleted binding was the only binding for its IP address in the SXP input binding database, SXP SHALL report it to the master binding database. The external outcome when no other contributors to the same IP address is present, is that a deletion of the binding using IPv4-Del-Prefix/IPv6-Del-Prefix or Del-IPv4/Del-IPv6 attribute MUST be exported on all connections on which the local end is a speaker.

If there is one or more additional contributors in the master binding database, a new binding is selected and it MUST be exported using IPv4-Add-Prefix/IPv6-Add-Prefix, IPv4-Add-Table/IPv6-Add-Table, or Add-IPv4/Add-IPv6 and associated with a Peer-Sequence that includes only the SXP Node ID of the local instance.

If the binding selection rule has yielded a newly selected binding from SXP input binding database it SHALL be compared to the binding it is replacing, and

- upon change of SGT, or, other associated data that are reported to the master binding database, report the newly selected binding to the master binding database. The newly selected contribution will become the selected contribution and will trigger export.

- If the SGT of the newly selected binding, and all other associated data that are reported to the master binding database, remain the

same as those of the deleted binding, SXP SHOULD NOT report a new contribution to the master binding database. However, the newly selected binding has a different Peer-Sequence attribute because binding from a different peer is selected. There could also be changes in optional transitive attributes. SXP SHALL export the newly selected binding directly and include all transitive attributes, thus bypassing the usual notification path from the master bindings database.

Processing Binding Add Attributes (step 4)

I. Implicit delete of current bindings

If the UPDATE message contains a non-empty IPv4-Add-Prefix/IPv6 -Add-Prefix/IPv4-Add-Table/IPv6-Add-Table or one or more Add-IPv4/ Add-IPv6 attributes, the previously received bindings from the remote speaker peer (if any), whose IP/IPv6 addresses are contained in any of those attributes, SHALL be removed from the SXP input bindings database and replaced with the new bindings for those IPv4/IPv6 addresses. An added binding is an implicit delete of an existing binding for the same IP address from the same peer. The external behavior of the implicit deletion is as described for the processing of IPv4-Del-Prefix/IPv6-Del-Prefix or Del-IPv4/Del-IPv6 attributes where the deleted binding is the last binding for its IP address.

II. Loop Detection

SXP MUST check the Peer-Sequence associated with each added binding for the presence of loops before accepting the binding as a new binding or as a replacement for an implicitly deleted binding. SXP loop detection is done by scanning the entire Peer-Sequence attribute, and checking that the SXP Node ID of the local system does not appear in the Peer-Sequence.

III. New or Replaced Bindings

SXP SHALL keep the most recently received binding for each IP address in the input binding database. SXP SHALL keep all the mandatory attributes and any optional transitive attributes and MAY keep other non-transitive attributes. A subset of the attributes, such as Source-Group-Tag, are used locally on the system and have to be reported to the master binding database. The master binding database arbitrates among multiple binding contributors for the same IP address. SXP SHALL report new bindings to the master binding database along with the attributes the master database is aware of. The arbitration process in the master binding database yields a selected binding contributor for

each IP address. The binding contributor could be a binding source other than SXP. SXP MUST NOT export any binding it has received in UPDATE message unless it has survived the arbitration process in the master binding database to become the selected binding.

SXP SHALL become aware of the outcome of the arbitration process within the master binding database. The arbitration process could be triggered by binding add or delete reported by SXP or binding changes from other binding contributors. SXP NEED NOT be aware of the presence of other binding contributors, their nature, or how many such contributors exist. SXP is only aware of the possible existence of such binding contributors and only need to know whether a selected binding was contributed by SXP itself or by some other contributor. The mechanism by which SXP becomes aware of changes of selected bindings in the master binding database is an implementation detail.

5.3. Generating UPDATE Message

Binding Removal Event

If SXP becomes aware of a deletion of a binding for an IPv4/IPv6 prefix it then processes the event as follows:

- The IPv4/IPv6 prefix SHALL be included in either an IPv4-Del-Prefix/IPv6-Del-Prefix or a single Del-IPv4/Del-IPv6 attribute of an outgoing UPDATE message. SXP MAY delay exporting of such event for a small and bound duration in order to allow for aggregation as long as such aggregation will not exhibit different external behavior from individually exported events.

Binding Change Event

If SXP becomes aware of a new or changed binding for an IPv4/IPv6 prefix it then processes the event as follows:

- If the selected contributor is not SXP, SXP SHALL associate the binding with a Peer-Sequence that contains the local SXP Node ID as the only peer and export the binding in one of IPv4-Add-Prefix/IPv6-Add-Prefix, IPv4-Add-Table/IPv6-Add-Table, or Add-IPv4/Add-IPv6. SXP SHALL include in the outgoing UPDATE message the attributes that are available from the master binding database.

- If the selected contributor is SXP, SXP MUST locate the contributed binding in its input binding database and extract the Peer-Sequence that was associated with the binding when it was originally received. SXP SHALL prepend the local SXP Node ID to

the Peer-Sequence when it is added to an outgoing UPDATE message. SXP SHALL include in the UPDATE message the mandatory attributes reported by the master binding database and any additional well-known or optional transitive attributes that were associated with the binding upon its arrival. SXP MAY add additional optional transitive or non-transitive attributes.

SXP SHALL flag an SXP originated binding in its input bindings database in order to support UPDATE message processing as specified in this section.

6. SXP Failure Scenarios

The following SXP failure scenarios are analyzed in this section:

- SXP Connection Failure
- Operational (Hardware) Failures

SXP connection can fail for various reasons:

- Peer device offline
- Loss of network connectivity
- Loss message authentication key synchronization with the peer. In this case the device re-attempts message exchange for a fixed number of times and if the problem persists then the connection is torn down.
- Software processing failures, e.g. SXP message decode failure. SXP version mismatch and SXP message parse error will result in the device sending an Error TLV. When sending an error message, the error code is passed as part of the TLV and the action decision is left up to the peer device. This error code COULD be simply logged for debugging purposes. The SXP connection is closed and retries to setup after sending the SXP ERROR message.

7. SXP Timers

There are 5 SXP timers defined in SXP protocol:

- Retry open timer
  - Retry open timer is triggered as long as there is one SXP connection on the device that is not up.

  * The default timer value is 120 seconds. Value 0 means retry timer will not be started.

  * The retry continues until the SXP connection is setup or the retry timer is configured to be 0.

- Delete Hold Down Timer

  * Delete hold down timer is trigger when a connection is on listener side is torn down. The bindings learnt are not deleted immediately but being held off for the delete hold down timer period.

  * The bindings are deleted upon the expiry of this timer.

  * The timer value is set to 120 seconds and it is not configurable.

- Reconciliation Timer

  * If a SXP connection is brought up within the delete hold down timer period, bindings are re-populated from the speaker side. At the same time, the old bindings learnt before the connection goes down still holds.

  * Reconciliation timer starts right after the connection is brought up to wait for the new bindings to be forwarded from the peer.

  * Upon the timer expiry, SXP checks the bindings in its input binding database and delete any stale bindings. Those are bindings that could have been deleted on the remote side while the connection was down.

  * The default timer value is 120 seconds. Value 0 means reconciliation timer will not be started.

- Hold Timer

  * Hold Timer MAY be used by an SXP Listener for detect when a connection is no longer live.

  * The usage of Hold Timer and KEEPALIVE is negotiated during OPEN /OPEN_RESP exchange.

  * The Hold Timer is started when the connection reached ON state. The interval is set to the negotiated Hold Time.

  * The Hold Timer is restarted whenever a listener receives a KEEPALIVE or an UPDATE message.

  * If a listener does not receive successive KEEPALIVE, and/or UPDATE messages within the period negotiated for the Hold Time of a connection, the Hold Timer expires.

  * Upon the timer expiry, SXP MUST send an ERROR message with Hold Timer Expired code and tear down the connection.

  * The rest of the behavior is the same as when TCP indicates to SXP the loss of a connection. The Delete Hold Down timer is restarted for delayed deletion of bindings as described above.

  * The suggested period of the Hold Timer is 90 seconds. SXP implementation MAY allow for local configuration of Hold Time for a listener. The actual value of the Hold Time used for a connection is negotiated.

- Keepalive Timer

  * Keepalive Timer MAY be used by an SXP Listener for triggering sending KEEPALIVE messages to the listener peer which monitors a connection using Hold Timer.

  * The usage of Keepalive Timer and KEEPALIVE messages is negotiated during OPEN/OPEN_RESP exchange.

  * The Keepalive Timer is restarted for the first time when the connection reaches its ON state.

  * The Keepalive Timer is restarted thereafter, when a speaker sends an UPDATE or KEEPALIVE message.

  * Upon timer expiry, SXP speaker MUST send KEEPALIVE message in order to indicate to the listener that the connection remains live.

  * The suggested period of the Keepalive Timer is 30 seconds. SXP implementation MAY allow for local configuration of Keepalive Time for a speaker. The actual value of the Keepalive Time used for a connection is set to 1/3 of the negotiated Hold Time.

  * The Keepalive Timer is restarted with a random jitter. The actual period of the Keepalive Timer is set to a random value between 0.75 to 1.0 of the negotiated Keepalive Time which is re-calculated every time the timer is restarted.

8. SXP Version Negotiation

In a software release, SXP always supports the highest version in that release and any version lower than that. SXP version negotiation is per connection base. On the same device, the connections established between this device and the peer devices may have difference versions, depending on the SXP version running on the peer devices.

If the two ends of a SXP connection run the same SXP version, the running version is used. If the two ends are running different versions, the version of the connection is decided with following process:

When the initiator of a connection sends a SXP OPEN message, the highest supported version is coded in the OPEN message.

On the receiver side, if the version in the OPEN message is higher than the version it can support, it sends back OPEN RESPONSE message with the highest version it supports (which is lower than the version in OPEN message). When the OPEN RESPONSE message is received by the initiator, it accepts the version coded in the OPEN RESPONSE. The receiver side's version is used as the connection's version.

In summary, SXP connection picks the highest version that is supported on both sides.

8.1. SXP Versions

SXP supports three versions starting from version 2 to 4. The details of what is supported in each of the version follows -

Ver	IPv4 Bindings	IPv6 Bindings	Subnet Binding Expansion	Loop Detection	SXP Capability Exchange
2	Yes	Yes	No	No	No
3	Yes	Yes	Yes	No	No
4	Yes	Yes	Yes	Yes	Yes

9. Security Considerations

SXP carries mappings between IP addresses and source groups. An adversary that can modify the data can potentially gain an advantage over the system. This specification defines the use of TCP MD5 Signature Option [RFC5925] to provide integrity protection for the

information. The key used with TCP MD5[RFC1321] should be chosen to be long and have high entropy to provide as much protection from dictionary attacks as possible.

MD5 has been shown to contain weaknesses so this protection may not be sufficient in many environments. In addition, an adversary who can observe the data may be able to use the information learned to identify weak or highly valued targets so it is desirable that confidentiality also be provided. A future revision of this specification will provide more robust security mechanisms such as ones based on IPSEC or TLS.

10. Implementation Note

Current implementations of the protocol support source and peer IP addresses to be IPv4 only. However, the implentation should take care of being able to support IPv6 addresses as well in the future. As such, there are not changes expected to be done in the protocol itself to support IPv6 address based connections.

The SXP password can be upto 80 ASCII characters.

11. IANA Considerations

TBD

12. IPR Disclosure

"By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with Section 6 of BCP 79."

13. Copyright Notice and Disclaimer

"Copyright (C) The IETF Trust (year). This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights." Additional copyright notices are not permitted in IETF Documents except in the case where such document is the product of a joint development effort between the IETF and another standards development organization or the document is a republication of the work of another standards organization. Such exceptions must be approved on an individual basis by the IAB.

"This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY , THE

IETF TRUST AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE."

14. Acknowledgements

Special acknowledgements are due to the following contributors. The protocol was originally developed by Michael Smith, Michael Fine and Awais Nemat. Enhancements have been made over the years by Ronen Arad, Rajesh Bhandari, Lei Fu, and Paddy Nallur. Darrin Miller provided significant input into requirements for later versions of the protocol.

15. Normative References

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

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5925] Touch, J., Mankin, A., and R. Bonica, "The TCP Authentication Option", RFC 5925, June 2010.

Authors' Addresses

Michael Smith
Cisco Systems
210 West Tasman Drive
San Jose, California 95134
United States

Email: michsmit@cisco.com

Rakesh Reddy Kandula
Cisco Systems
Cessna Business Park, Kadubeesanahalli Village
Sarjapur/Marathahalli Outer Ring Road
Bangalore, Karnataka 560103
India

Email: krreddy@cisco.com