

# HEP3 Network Protocol Specification

## (rev. 32 2022-06-21)

Authors (Roland Haenel, Alexandr Dubovikov)

### General protocol structure

HEP3 (Homer Encapsulation Protocol Version 3) transmits packets over UDP/TCP/SCTP connections.

Each packet starts with the HEP3 header:

octet offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0-15	[HEP3 EEP3] ID (0x48455033)				total length		...payload...									
16-31	...payload (continued)...															
...	...payload (continued)...															

The HEP3 header consists of a 4-octet protocol identifier with the fixed value 0x48455033 (ASCII „HEP3“) and a two-octet length value (network byte order). The length value specifies the total packet length including the HEP3 or EEP3 ID, and the length field itself and the payload. It has a possible range of values between 6 and 65535.

After the header, the payload is structured in form of concatenated chunks. Each chunk has the following structure (octet offset relative to the start of the chunk):

octet offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0-15	chunk vendor ID		chunk type ID		chunk length		chunk payload (variable length)									

The chunk type is identified by a two-octet vendor ID and a two-octet type ID (both in network byte order). The vendor ID allows for grouping of chunk types for specific vendors (i.e., a vendor can receive a vendor ID and then define chunk type on its own). The chunk length field (network byte order) specifies the total length of the chunk, including the vendor ID, type ID, length and payload fields.

In combination, the vendor and type ID fields and the length field allows a HEP3 implementation to skip unknown chunks and continue processing of a HEP3 packet.

The chunk payload depends on the type of the chunk and is defined in the following documentation section, or, for vendor specific chunks, by the vendor that maintains the chunk vendor ID. The following payload types are defined

payload type	payload type description
octet-string	arbitrary octet string ("byte array")
utf8-string	UTF8 encoded character sequence
uint8	8 bit unsigned integer number
uint16	16 bit unsigned integer number (network byte order)
uint32	32 bit unsigned integer number (network byte order)
uint64	64 bit unsigned integer number (network byte order)
inet4-addr	4 octet IPv4 address, most significant octet first
inet6-addr	16 octet ipv6 address, most significant octet first

## Generic chunk types

Chunk types with chunk vendor ID 0x0000 are called generic chunk types. The following generic chunk types are defined:

chunk type ID	payload type	chunk type description
0x0001	uint8	IP protocol family
0x0002	uint8	IP protocol ID
0x0003	inet4-addr	IPv4 source address
0x0004	inet4-addr	IPv4 destination address
0x0005	inet6-addr	IPv6 source address
0x0006	inet6-addr	IPv6 destination address
0x0007	uint16	protocol source port (UDP, TCP, SCTP)
0x0008	uint16	protocol destination port (UDP, TCP, SCTP)
0x0009	uint32	timestamp, seconds since 01/01/1970 (epoch)
0x000a	uint32	timestamp microseconds offset (added to timestamp)
0x000b	uint8	protocol type (SIP/H323/RTP/MGCP/M2UA)
0x000c	uint32	capture agent ID (202, 1201, 2033...)
0x000d	uint16	keep alive timer (sec)
0x000e	octet-string	authenticate key (plain text / TLS connection)

0x000f	octet-string	captured packet payload
0x0010	octet-string	captured compressed payload (gzip/inflate)
0x0011	octet-string	Internal correlation id
0x0012	uint16	Vlan ID
0x0013	octet-string	Group ID
0x0014	uint64	Source MAC
0x0015	uint64	Destination MAC
0x0016	uint16	Ethernet Type
0x0017	uint8	TCP Flag [SYN.PUSH...]
0x0018	uint8	IP TOS
.....	.....	.....
0x001F		Reserved
0x0020	uint16	MOS value
0x0021	uint16	R-Factor
0x0022	octet-string	GEO Location
0x0023	uint32	Jitter
0x0024	octet-string	Transaction type [call, registration]
0x0025	octet-string	Payload JSON Keys
0x0026	octet-string	Tags' values
0x0027	uint16	Type of tag
0x0028	uint16	Event type [recording interception]

## Capture protocol types (0x00b)

chunk protocol ID	assigned vendor
0x00	reserved
0x01	SIP
0x02	XMPP

0x03	SDP
0x04	RTP
0x05	RTCP JSON
0x06	MGCP
0x07	MEGACO (H.248)
0x08	M2UA (SS7/SIGTRAN)
0x09	M3UA (SS7/SIGTRAN)
0x0a	IAX
0x0b	H3222
0x0c	H321
0x0d	M2PA
0x22	MOS full report [JSON]
0x23	MOS short report. Please use mos chunk 0x20 [JSON]
0x32	SIP JSON
0x33	RESERVED
0x34	RESERVED
0x35	DNS JSON
0x36	M3UA JSON (ISUP)
0x37	RTSP (JSON)
0x38	DIAMETER (JSON)
0x39	GSM MAP (JSON)
0x3a	RTCP PION
0x3b	RESERVED
0x3c	CDR (can be for call and registration transaction)

## Vendor chunk types

Definition of chunk types for specific vendors is up to the vendor; however the assignment of vendor IDs is specified in this document. The following vendor IDs are assigned:

chunk vendor ID	assigned vendor
-----------------	-----------------

0x0000	No specific vendor, generic chunk types, see above
0x0001	FreeSWITCH ( <a href="http://www.freeswitch.org">www.freeswitch.org</a> )
0x0002	Kamailio/SER ( <a href="http://www.kamailio.org">www.kamailio.org</a> )
0x0003	OpenSIPS ( <a href="http://www.opensips.org">www.opensips.org</a> )
0x0004	Asterisk ( <a href="http://www.asterisk.org">www.asterisk.org</a> )
0x0005	Homer Project ( <a href="http://www.sipcapture.org">http://www.sipcapture.org</a> )
0x0006	SipXecs ( <a href="http://www.sipfoundry.org/">www.sipfoundry.org/</a> )
0x0007	Yeti Switch ( <a href="https://yeti-switch.org/">https://yeti-switch.org/</a> )
0x0008	Genesys ( <a href="https://www.genesys.com/">https://www.genesys.com/</a> )

## Capture protocol types (0x028)

chunk event type	Event name
0x00	reserved
0x01	Recording
0x02	Interception

## Example HEP3 packet

The following packet is decoded as an example of a valid HEP3 packet (hexadecimal octet encoding). The SIP packet payload is shortened for this example.

```

48 45 50 33
    ; HEP3 ID
00 71
    ; total length = 113 octets
00 00 00 01 00 07 02
    ; protocol family = 2 (IPv4)
00 00 00 02 00 07 11
    ; protocol ID = 17 (UDP)
00 00 00 03 00 0a d4 ca 00 01
    ; IPv4 source address = 212.202.0.1
00 00 00 04 00 0a 52 74 00 d3
    ; IPv4 destination address = 82.116.0.211
00 00 00 07 00 08 2e ea

```

```
        ; source port = 12010
00 00 00 08 00 08 13 c4
        ; destination port = 5060
00 00 00 09 00 0a 4e 49 82 cb
        ; seconds timestamp 1313440459 = Mon Aug 15 22:34:19 2011
00 00 00 0a 00 0a 00 01 d4 c0
        ; micro-seconds timestamp offset 120000 = 0.12 seconds
00 00 00 0b 00 07 01
        ; 01 - SIP
00 00 00 0c 00 0a 00 00 00 E4
        ; capture ID (228)
00 00 00 0f 00 14 49 4e 56 49 54 45 20 73 69 70 3a 62 6f 62
        ; SIP payload "INVITE sip:bob" (shortened)
```