Edition 2024

SL DPI

Protocol Dictionary

slinkin.tech

Contents

Protocol dictionary

Physical

None

Data Link

Ethernet

> STATUS

Protocol	RFC	Status	Tags
Ethernet	ieee802.3	Fully	basic, network, internet

> FIELDS

Name	Type	Length	Mask	Multiple	Description
root	**uint8**	**1**	ffffffffffffffff	false	Layer presented flag.
raw_data	**byte-sequence**	**0**	ffffffffffffffff	false	Layer data.
raw_data_length	**uint64**	**8**	ffffffffffffffff	false	Layer data length.
payload_data	**byte-sequence**	**0**	ffffffffffffffff	false	The payload data - data which is placed right after the layer data.
payload_data_length	**uint64**	**8**	ffffffffffffffff	false	The payload data length.
src_mac	**byte-sequence**	**6**	ffffffffffffffff	false	Source MAC address.

dst_mac	**byte-sequence**	**6**	ffffffffffffff	false	Destination MAC address.
ethernet_type	**uint16**	**2**	ffffffffffffff	false	Two-octet field which is used to indicate which protocol is encapsulated in the payload of the frame. 0x0000 – 0x05DC – **IEEE802.3** length Field. 0x0101-0x01FF – experimental.

ARP

> STATUS

Protocol	RFC	Status	Tags
ARP	rfc826	Fully	basic, network, internet

> LAYER DETECTION METHODS

- Explicit detection (Ethernet Type)

> FIELDS

Name	Type	Length	Mask	Multiple	Description
root	**uint8**	**1**	ffffffffffffff	false	Layer presented flag.
raw_data	**byte-sequence**	**0**	ffffffffffffff	false	Layer data.
raw_data_length	**uint64**	**8**	ffffffffffffff	false	Layer data length.
payload_data	**byte-sequence**	**0**	ffffffffffffff	false	The payload data - data which is placed right after the layer data.
payload_data_length	**uint64**	**8**	ffffffffffffff	false	The payload data length.

htype	**uint16**	**2**	ffffffffffffffff	false	Hardware type. (Network link protocol type)
ptype	**uint16**	**2**	ffffffffffffffff	false	Protocol type. Specifies internetwork protocol.
hlen	**uint8**	**1**	ffffffffffffffff	false	Hardware address length. (in octets)
plen	**uint16**	**1**	ffffffffffffffff	false	Protocol length. (Internetwork addresses length; in octets)
op	**uint8**	**2**	ffffffffffffffff	false	Operation. **1**: request, **2**: reply.
sha	**byte-sequence**	**0**	ffffffffffffffff	false	Sender hardware address. In request, indicates the address of the host sending the request. In reply, indicates the address of the host that the request was looking for.
spa	**byte-sequence**	**0**	ffffffffffffffff	false	Sender protocol address. (Internetwork address of the sender)
tha	**byte-sequence**	**0**	ffffffffffffffff	false	Target hardware address. In request, this field is not used. In reply, indicates the address of the host that originated the ARP request.
tpa	**byte-sequence**	**0**	ffffffffffffffff	false	Target protocol address. (Internetwork address of the intended receiver)

RARP

> STATUS

Protocol	RFC	Status	Tags
RARP	rfc903	Fully	basic, network, internet

> LAYER DETECTION METHODS

- Explicit detection (Ethernet Type)

> FIELDS

All **ARP** fields are valid for RARP as well.

VLAN C-TAG

> STATUS

Protocol	RFC	Status	Tags
Vlan C-Tag	ieee802.1q	Fully	basic, network, internet

> LAYER DETECTION METHODS

- Explicit detection (Ethernet Type)

> FIELDS

Name	Type	Length	Mask	Multiple	Description
root	**uint8**	**1**	ffffffffffffff	false	Layer presented flag.
raw_data	**byte-sequence**	**0**	ffffffffffffff	false	Layer data.
raw_data_length	**uint64**	**8**	ffffffffffffff	false	Layer data length.
payload_data	**byte-sequence**	**0**	ffffffffffffff	false	The payload data - data which is placed right after the layer data.
payload_data_length	**uint64**	**8**	ffffffffffffff	false	The payload data length.
tci	**uint16**	**1**	ffffffffffffff	false	Tag control information.
pcp	**16-bit-field**	**2**	e000	false	Priority code point.
dei	**16-bit-field**	**2**	1000	false	Drop eligible indicator.
vid	**16-bit-field**	**2**	fff	false	VLAN identifier.

ethernet_type	**uint16**	**2**	ffffffffffffff	false	Two-octet field which is used to indicate which protocol is encapsulated in the payload of the frame. 0x0000 - 0x05DC - **IEEE802.3** length Field. 0x0101-0x01FF - experimental.

GRE

> STATUS

Protocol	RFC	Status	Tags
GRE	rfc2784	Fully	basic, network, internet

> LAYER DETECTION METHODS

- Explicit detection (IP Protocol Type))

> FIELDS

Name	Type	Length	Mask	Multiple	Description
root	**uint8**	**1**	ffffffffffffff	false	Layer presented flag.
raw_data	**byte-sequence**	**0**	ffffffffffffff	false	Layer data.
raw_data_length	**uint64**	**8**	ffffffffffffff	false	Layer data length.
payload_data	**byte-sequence**	**0**	ffffffffffffff	false	The payload data - data which is placed right after the layer data.
payload_data_length	**uint64**	**8**	ffffffffffffff	false	The payload data length
checksum_flag	**16-bit-field**	**2**	8000	false	If the Checksum Present bit is set to one, then the Checksum and the Reserved1 fields are present and the Checksum field contains valid information.

reserved0	**16-bit-field**	**2**	7ff8	false	A receiver MUST discard a packet where any of bits 1-5 are non-zero, unless that receiver implements rfc1701. Bits 6-12 are reserved for future use.
version	**16-bit-field**	**2**	7	false	The Version Number field MUST contain the value zero.
protocol_type	**uint16**	**2**	ffffffffffffffff	false	The Protocol Type field contains the protocol type of the payload packet. These Protocol Types are defined in rfc1700 as "ETHER TYPES" and in [ETYPES].
checksum	**uint16**	**2**	ffffffffffffffff	false	The Checksum field contains the IP (one's complement) checksum sum of the all the 16 bit words in the GRE header and the payload packet.
reserved0	**uint16**	**2**	ffffffffffffffff	false	The Reserved1 field is reserved for future use, and if present, MUST be transmitted as zero.

Network

IPv4

> STATUS

Protocol	RFC	Status	Tags
IPv4	rfc791	Partly	basic, network, internet

> LAYER DETECTION METHODS

- Explicit detection (Ethernet Type)

> FIELDS

Name	Type	Length	Mask	Multiple	Description
root	**uint8**	**1**	ffffffffffffff	false	Layer presented flag.
raw_data	**byte-sequence**	**0**	ffffffffffffff	false	Layer data.
raw_data_length	**uint64**	**8**	ffffffffffffff	false	Layer data length.
payload_data	**byte-sequence**	**0**	ffffffffffffff	false	The payload data - data which is placed right after the layer data.
payload_data_length	**uint64**	**8**	ffffffffffffff	false	The payload data length.
ip_version	**8-bit-field**	**1**	f0	false	Internet Header Length is the length of the internet header in 32 bit words, and thus points to the beginning of the data.
ihl	**8-bit-field**	**1**	f	false	This field indicates where in the datagram this fragment belongs.
tos	**uint8**	**1**	ffffffffffffff	false	Type of Service provides an indication of the abstract parameters of the quality of service desired.
dscp	**8-bit-field**	**1**	fc	false	Differentiated Services Code Point.
ecn	**8-bit-field**	**1**	3	false	Explicit Congestion Notification.
total_length	**uint16**	**2**	ffffffffffffff	false	**Total Length** is the length of the datagram, measured in octets, including internet header and data.
id	**uint16**	**2**	ffffffffffffff	false	An identifying value assigned by the sender to aid in assembling the fragments of a datagram.
reserved_flag	**16-bit-field**	**2**	8000	false	Reserved bit. Must be zero.
dm_flag	**16-bit-field**	**2**	4000	false	0 (may fragment), 1 (don't fragment).
mf_flag	**16-bit-field**	**2**	2000	false	0 (last fragment), 1 (more fragments).

fragment_ offset	**16-bit-field**	**2**	1fff	false	This field indicates where in the datagram this fragment belongs.
ttl	**uint8**	**1**	fffffffffffffff	false	This field indicates the maximum time the datagram is allowed to remain in the internet system.
protocol	**uint8**	**1**	fffffffffffffff	false	This field indicates the next level protocol used in the data portion of the internet datagram.
checksum	**uint16**	**1**	ffffffffffffffff	false	A checksum on the header only.
src_ip	**uint32**	**4**	ffffffffffffffff	false	The source address.
dst_ip	**uint32**	**4**	fffffffffffffff	false	The destination address.
options	**byte-sequence -field**	**0**	fffffffffffffff	false	The options section.

> LIMITATION

- Options are not supported

ICMP

> STATUS

Protocol	RFC	Status	Tags
ICMP	rfc792	Partly	basic, network, internet

> LAYER DETECTION METHODS

- Explicit detection (IP Protocol Type)

> FIELDS

Name	Type	Length	Mask	Multiple	Description
root	**uint8**	**1**	ffffffffffffff	false	Layer presented flag.
raw_data	**byte-sequence**	**0**	ffffffffffffff	false	Layer data.
raw_data_length	**uint64**	**8**	ffffffffffffff	false	Layer data length.
payload_data	**byte-sequence**	**0**	ffffffffffffff	false	The payload data - data which is placed right after the layer data.
payload_data_length	**uint64**	**8**	ffffffffffffff	false	The payload data length.
type	**uint8**	**1**	ffffffffffffff	false	Type of message.
code	**uint8**	**1**	ffffffffffffff	false	Additional context information for the message.
checksum	**uint16**	**2**	ffffffffffffff	false	The 16-bit ones's complement of the one's complement sum of the ICMP message starting with the ICMP Type.
padding	**uint32**	**4**	ffffffffffffff	false	Four-byte field, contents vary based on the ICMP type and code.

Transport

QUIC

> STATUS

Protocol	RFC	Status	Tags
Quic	rfc9000 rfc9001	Fully	basic, network, internet

> LAYER DETECTION METHODS

- Port-based
- Layer structure test

> PORTS

- 443 (**udp**)

> FIELDS

Name	Type	Length	Mask	Multiple	Description
root	**uint8**	**1**	ffffffffffffffff	false	Layer presented flag.
raw_data	**byte-sequence**	**0**	ffffffffffffffff	false	Layer data.
raw_data_length	**uint64**	**8**	ffffffffffffffff	false	Layer data length.
payload_data	**byte-sequence**	**0**	ffffffffffffffff	false	The payload data - data which is placed right after the layer data.
payload_data_length	**uint64**	**8**	ffffffffffffffff	false	The payload data length.
header	**uint8**	**1**	ffffffffffffffff	false	Quic header. The structure of header can be different between different packet types.

header_form	**uint8**	**1**	80	false	The field specifies header type. It is set to 0 for short headers and is set to 1 for long headers.
retry_unused	**uint8**	**1**	f	false	Unused header bits of retry packet.
version_negotiation_unused	**uint8**	**1**	7f	false	Unused header bits of version negotiation packet.
spin_bit	**uint8**	**1**	20	false	The latency spin bit, which is defined for 1-RTT packets, enables passive latency monitoring from observation points on the network path throughout the duration of a connection.
fixed_bit	**uint8**	**1**	40	false	Packets containing a zero value for this bit are not valid packets in this version and **MUST** be discarded. A value of 1 for this bit allows **QUIC** to coexist with other protocols.
long_packet_type	**uint8**	**1**	30	false	The field specifies packet type in the long header. Initial (**0**), 0-RTT (**1**), Handshake (**2**), Retry (**3**).
protected_reserved_bits	**uint8**	**1**	c	false	Reserved header bits of 0-RTT and Handshake packets. The field is protected.
protected_1rtt_reserved_bits	**uint8**	**1**	18	false	Reserved header bits of 1-RTT packet. The field is protected.
protected_key_phase	**uint8**	**1**	4	false	The field indicates the key phase, which allows a recipient of a packet to identify the packet protection keys that are used to protect the packet. The field is protected.
protected_packet_number_length	**uint8**	**1**	3	false	The field specifies the size of packet number length field. The field is protected. The field is protected.

unprotected_reserved_bits	**uint8**	**1**	c	false	Reserved header bits of 0-RTT and Handshake packets. The field is presented only inside a decrypted layer because the field data is protected.
unprotected_1rtt_reserved_bits	**uint8**	**1**	18	false	Reserved header bits of 1-RTT packet. The field is presented only inside a decrypted layer because the field data is protected.
unprotected_key_phase	**uint8**	**1**	4	false	The field indicates the key phase, which allows a recipient of a packet to identify the packet protection keys that are used to protect the packet. The field is presented only inside a decrypted layer because the field data is protected.
unprotected_packet_number_length	**uint8**	**1**	3	false	The field specifies the size of packet number length field. The field is presented only inside a decrypted layer because the field data is protected.
version	**uint32**	**4**	ffffffffffffffff	false	The QUIC Version is a 32-bit field that follows the first byte. This field indicates the version of **QUIC** that is in use and determines how the rest of the protocol fields are interpreted.
destination_connection_id_length	**byte-sequence**	**0**	ffffffffffffffff	false	The length of destination connection id field.
destination_connection_id	**byte-sequence**	**0**	ffffffffffffffff	false	The destination connection id.
source_connection_id_length	**byte-sequence**	**0**	ffffffffffffffff	false	The length of source connection id field.
source_connection_id	**byte-sequence**	**0**	ffffffffffffffff	false	The source connection id.

token_length	**byte-sequence**	**0**	ffffffffffffff	false	A variable-length integer specifying the length of the Token field, in bytes. This value is 0 if no token is present.
token	**byte-sequence**	**0**	ffffffffffffff	false	The value of the token that was previously provided in a Retry packet or NEW_TOKEN frame.
supported_version	**uint32**	**4**	ffffffffffffff	false	Supported version.
length	**byte-sequence**	**0**	ffffffffffffff	false	This is the length of the remainder of the packet (that is, the Packet Number and Payload fields) in bytes, encoded as a variable-length integer.
packet_data	**byte-sequence**	**0**	ffffffffffffff	false	Packet data section - includes packet number and packet payload fields.
packet_number	**byte-sequence**	**0**	ffffffffffffff	false	This field is 1 to 4 bytes long. The field is presented only inside a decrypted layer because the field data is protected.
protected_data	**byte-sequence**	**0**	ffffffffffffff	false	The "abstract" field which is presented for the data section which cannot be dissected. E.g. when session context or Initial packet of the session are missed.
retry_token	**byte-sequence**	**0**	ffffffffffffff	false	An opaque token that the server can use to validate the client’s address.
retry_integrity_tag	**byte-sequence**	**16**	ffffffffffffff	false	The Retry Integrity Tag is a 128-bit field that is computed as the output of AEAD_AES_128_GCM.
frame	**byte-sequence**	**0**	ffffffffffffff	true	Frame section. The payload of QUIC packets, after removing packet protection, consists of a sequence of complete frames.

frame_type	**byte-sequence**	**0**	ffffffffffffffff	true	Frame type.
padding_data	**byte-sequence**	**0**	ffffffffffffffff	true	The field contains the bytes of padding frame types. The field exists for brevity purposes to not pollute padding fields.
largest_acknowledged	**byte-sequence**	**0**	ffffffffffffffff	true	A variable-length integer representing the largest packet number the peer is acknowledging; this is usually the largest packet number that the peer has received prior to generating the ACK frame.
ack_delay	**byte-sequence**	**0**	ffffffffffffffff	true	A variable-length integer encoding the acknowledgment delay in microseconds.
ack_range_count	**byte-sequence**	**0**	ffffffffffffffff	true	A variable-length integer specifying the number of ACK Range fields in the frame.
first_ack_range	**byte-sequence**	**0**	ffffffffffffffff	true	A variable-length integer indicating the number of contiguous packets preceding the Largest Acknowledged that are being acknowledged.
first_ack_range	**byte-sequence**	**0**	ffffffffffffffff	true	A variable-length integer indicating the number of contiguous packets preceding the Largest Acknowledged that are being acknowledged.
ack_range	**byte-sequence**	**0**	ffffffffffffffff	true	Contains additional ranges of packets that are alternately not acknowledged (Gap) and acknowledged (ACK Range).
gap	**byte-sequence**	**0**	ffffffffffffffff	true	A variable-length integer indicating the number of contiguous unacknowledged packets preceding the packet number one lower than the smallest in the preceding ACK Range.
ack_range_length	**byte-sequence**	**0**	ffffffffffffffff	true	A variable-length integer indicating the number of contiguous acknowledged packets preceding the largest packet number, as determined by the preceding Gap.

ecn_counts	**byte-sequence**	**0**	ffffffffffffffff	true	The three ECN counts. ECN counts are only present when the ACK frame type is 0x03.
ect_0_count	**byte-sequence**	**0**	ffffffffffffffff	true	A variable-length integer representing the total number of packets received with the ECT(0) codepoint in the packet number space of the ACK frame.
ect_1_count	**byte-sequence**	**0**	ffffffffffffffff	true	A variable-length integer representing the total number of packets received with the ECT(1) codepoint in the packet number space of the ACK frame.
ecn_ce_count	**byte-sequence**	**0**	ffffffffffffffff	true	A variable-length integer representing the total number of packets received with the ECN-CE codepoint in the packet number space of the ACK frame.
reset_stream_id	**byte-sequence**	**0**	ffffffffffffffff	true	A variable-length integer encoding of the stream ID of the stream being terminated.
stop_stream_id	**byte-sequence**	**0**	ffffffffffffffff	true	A variable-length integer carrying the stream ID of the stream being ignored.
max_data_stream_id	**byte-sequence**	**0**	ffffffffffffffff	true	The stream ID of the affected stream, encoded as a variable-length integer.
blocked_stream_id	**byte-sequence**	**0**	ffffffffffffffff	true	A variable-length integer indicating the stream that is blocked due to flow control.
stream_id	**byte-sequence**	**0**	ffffffffffffffff	true	A variable-length integer indicating the stream ID of the stream.
reset_application_protocol_error_code	**byte-sequence**	**0**	ffffffffffffffff	true	A variable-length integer containing the application protocol error code that indicates why the stream is being closed.

stop_application_protocol_error_code	**byte-sequence**	**0**	ffffffffffffffff	true	A variable-length integer containing the application-specified reason the sender is ignoring the stream.
final_size	**byte-sequence**	**0**	ffffffffffffffff	true	A variable-length integer indicating the final size of the stream by the RESET_STREAM sender, in units of bytes.
crypto_offset	**byte-sequence**	**0**	ffffffffffffffff	true	A variable-length integer specifying the byte offset in the stream for the data in this CRYPTO frame.
crypto_data_length	**byte-sequence**	**0**	ffffffffffffffff	true	A variable-length integer specifying the length of the Crypto Data field in this CRYPTO frame.
crypto_data	**byte-sequence**	**0**	ffffffffffffffff	true	The cryptographic message data.
maximum_data	**byte-sequence**	**0**	ffffffffffffffff	true	A variable-length integer indicating the maximum amount of data that can be sent on the entire connection, in units of bytes.
blocked_maximum_data	**byte-sequence**	**0**	ffffffffffffffff	true	A variable-length integer indicating the connection-level limit at which blocking occurred.
maximum_stream_data	**byte-sequence**	**0**	ffffffffffffffff	true	A variable-length integer indicating the maximum amount of data that can be sent on the identified stream, in units of bytes.
blocked_maximum_stream_data	**byte-sequence**	**0**	ffffffffffffffff	true	A variable-length integer indicating the maximum amount of data that can be sent on the identified stream, in units of bytes.
cumulative_maximum_streams	**byte-sequence**	**0**	ffffffffffffffff	true	A count of the cumulative number of streams of the corresponding type that can be opened over the lifetime of the connection.
allowed_maximum_streams	**byte-sequence**	**0**	ffffffffffffffff	true	A count of the cumulative number of streams of the corresponding type that can be opened over the lifetime of the connection.

retire_sequence_number	**byte-sequence**	**0**	ffffffffffffffff	true	The sequence number of the connection ID being retired.
new_sequence_number	**byte-sequence**	**0**	ffffffffffffffff	true	The sequence number assigned to the connection ID by the sender, encoded as a variable-length integer.
path_challenge_data	**byte-sequence**	**8**	ffffffffffffffff	true	This 8-byte field contains arbitrary data.
path_response_data	**byte-sequence**	**8**	ffffffffffffffff	true	This 8-byte field contains arbitrary data.
retire_prior_to	**byte-sequence**	**0**	ffffffffffffffff	true	An 8-bit unsigned integer containing the length of the connection ID.
connection_id_length	**uint8**	**1**	ffffffffffffffff	true	An 8-bit unsigned integer containing the length of the connection ID.
connection_id	**byte-sequence**	**0**	ffffffffffffffff	true	A connection ID of the specified length.
stateless_reset_token	**byte-sequence**	**16**	ffffffffffffffff	true	A 128-bit value that will be used for a stateless reset when the associated connection ID is used.
stream_offset	**byte-sequence**	**0**	ffffffffffffffff	true	A variable-length integer specifying the byte offset in the stream for the data in this STREAM frame. This field is present when the **OFF** bit is set to **1**. When the Offset field is absent, the offset is **0**.
stream_data_length	**byte-sequence**	**0**	ffffffffffffffff	true	A variable-length integer specifying the byte offset in the stream for the data in this STREAM frame. This field is present when the OFF bit is set to **1**. When the Offset field is absent, the offset is **0**.
stream_data	**byte-sequence**	**0**	ffffffffffffffff	true	The bytes from the designated stream to be delivered.

error_code	**byte-sequence**	**0**	ffffffffffffffff	true	A variable-length integer that indicates the reason for closing this connection. Error codes for 0x1c and 0x1d frame types have different description.
triggered_frame_type	**byte-sequence**	**0**	ffffffffffffffff	true	A variable-length integer encoding the type of frame that triggered the error. A value of 0 (equivalent to the mention of the PADDING frame) is used when the frame type is unknown. The field is presented only when frame type is 0x1d.
reason_phrase_length	**byte-sequence**	**0**	ffffffffffffffff	true	A variable-length integer specifying the length of the reason phrase in bytes.
reason_phrase	**byte-sequence**	**0**	ffffffffffffffff	true	Additional diagnostic information for the closure. This can be zero length if the sender chooses not to give details beyond the Error Code value.

> FIELD TREE

- **Root**

```
.
├── header
├── version
├──destination_connection_id_length
├── destination_connection_id
├── source_connection_id_length
├── source_connection_id
├── token_length
├── token
├── supported_version
├── length
├── packet_data
├── packet_number
├── protected_data
├── retry_token
├── retry_integrity_tag
└── frame
```

- **Header**

```
.
└── header/
    ├── header_form
    ├── retry_unused (Retry Packet)
    ├── version_negotiation_unused (Version Negotiation Packet)
    ├── spin_bit (1-RTT Packet Only)
    ├── fixed_bit (Handshake 0-RTT and 1-RTT)
    ├── long_packet_type (Handshake and 0-RTT)
    ├── protected_reserved_bits  (Handshake 0-RTT)
    ├── protected_1rtt_reserved_bits (1-RTT)
    ├── protected_key_phase (1-RTT Packet Only)
    ├── protected_packet_number_length (Handshake and 0-RTT)
    ├── unprotected_reserved_bits (Handshake and 0-RTT)
    ├── unprotected_1rtt_reserved_bits (1-RTT)
    ├── unprotected_key_phase (1-RTT Packet Only)
    └── unprotected_packet_number_length (Handshake and 0-RTT)
```

- **Frame**

```
.
└── frame/
    ├── frame_type
    ├── padding_data (Padding)
    ├── largest_acknowledged (Ack)
    ├── ack_delay (Ack)
    ├── ack_range_count (Ack)
    ├── first_ack_range (Ack)
    ├── ack_range (Ack)
    ├── ecn_counts (Ack)
    ├── reset_stream_id (ResetStream)
    ├── stop_stream_id (StopSending)
    ├── max_data_stream_id (MaxStreamData)
    ├── blocked_stream_id (StreamDataBlocked)
    ├── stream_id (Stream)
    ├── reset_application_protocol_error_code (ResetStream)
    ├── stop_application_protocol_error_code (StopSending)
    ├── final_size (ResetStream)
    ├── crypto_offset (Crypto)
    ├── crypto_data_length (Crypto)
    ├── crypto_data (Crypto)
    ├── maximum_data (MaxData)
    ├── blocked_maximum_data (DataBlocked)
    ├── maximum_stream_data (MaxStreamData)
    ├── blocked_maximum_stream_data (StreamDataBlocked)
    ├── cumulative_maximum_streams (MaxStreams)
    ├── allowed_maximum_streams (StreamsBlocked)
    ├── retire_sequence_number (RetireConnectionId)
    ├── new_sequence_number (NewConnectionId)
    ├── path_challenge_data (PathChallenge)
    ├── path_response_data (PathResponse)
    ├── retire_prior_to (NewConnectionId)
    ├── connection_id_length (NewConnectionId)
    ├── connection_id (NewConnectionId)
    ├── stateless_reset_token (NewConnectionId)
    ├── stream_offset (Stream)
    ├── stream_data_length (Stream)
    ├── stream_data (Stream)
    ├── error_code (ConnectionClose)
    ├── triggered_frame_type (ConnectionClose)
    ├── reason_phrase_length (ConnectionClo
    └── reason_phrase_length (ConnectionClo
```

- **AckRange**

```
.
└── ack_range (Ack)
    ├── gap (Ack)
    └── ack_range_length (Ack)
```

- **EcnCounts**

```
.
└── ecn_counts (Ack)
    ├── ect_0_count (Ack)
    ├── ect_1_count (Ack)
    └── ecn_ce_count (Ack)
```

> LIMITATION

- Due to specification, Retry Packet contains Retry Token and Retry Integrity Tag. Because of Retry Token length is not specified explicitly, these 2 fields are combined in one field RetryData

- Encoder from human to byte sequence generates sequence accordingly number value. E.g. if number in [0-63] 0-16383 it generates 1 byte sequence, if [0-16383] 0-16383 it generates 2 bytes sequence, and so on.

Such note is presented here because specification doesn't explain a case when value is less than minimum interval value, e.g. value is 16, but the field length is 2 bytes.

- Quic frames can be dissected only for initial packets (for the rest packets TLS keys are required). Because of that, flow/session cannot be closed when CONNECTION_CLOSE is sent because of it is encrypted. Like that, quic flow/session is closed/released by timeout.

TCP

> STATUS

Protocol	RFC	Status	Tags
TCP	rfc793	Partly	basic, network, internet

> LAYER DETECTION METHODS

- Explicit detection (IP Protocol Type)

> FIELDS

Name	Type	Length	Mask	Multiple	Description
root	**uint8**	**1**	ffffffffffffff	false	Layer presented flag.
raw_data	**byte-sequence**	**0**	ffffffffffffff	false	Layer data.
raw_data_length	**uint64**	**8**	ffffffffffffff	false	Layer data length.
payload_data	**byte-sequence**	**0**	ffffffffffffff	false	The payload data - data which is placed right after the layer data.
payload_data_length	**uint64**	**8**	ffffffffffffff	false	The payload data length.
src_port	**uint16**	**2**	ffffffffffffff	false	The source port number.
dst_port	**uint16**	**2**	ffffffffffffff	false	The destination port number.
sequence_number	**uint32**	**4**	ffffffffffffff	false	The sequence number of the first data octet in this segment (except when SYN is present).
ack_number	**uint32**	**4**	ffffffffffffff	false	If the ACK control bit is set this field contains the value of the next sequence number the sender of the segment is expecting to receive.

data_offset	**8-bit-field**	**1**	f0	false	The number of 32 bit words in the TCP Header.
reserved	**8-bit-field**	**1**	f	false	Reserved for future use. Must be zero.
flags	**uint8**	**1**	ffffffffffffff	false	The field which contains tcp flags which are used to indicate a particular state of connection.
cwr	**8-bit-field**	**1**	80	false	Congestion Window Reduced flag.
ece	**8-bit-field**	**1**	40	false	ECN-Echo flag.
urg	**8-bit-field**	**1**	20	false	Urgent Pointer field significant.
ack	**8-bit-field**	**1**	10	false	Acknowledgment field significant.
psh	**8-bit-field**	**1**	8	false	Push Function.
rst	**8-bit-field**	**1**	4	false	Reset the connection.
syn	**8-bit-field**	**1**	2	false	Synchronize sequence numbers.
fin	**8-bit-field**	**1**	1	false	No more data from sender.
window_size	**uint16**	**2**	ffffffffffffff	false	The number of data octets beginning with the one indicated in the acknowledgment field which the sender of this segment is willing to accept.
checksum	**uint16**	**2**	ffffffffffffff	false	The checksum field is the 16 bit one’s complement of the one’s complement sum of all 16 bit words in the header and text.
syn	**uint16**	**2**	ffffffffffffff	false	This field communicates the current value of the urgent pointer as a positive offset from the sequence number in this segment.

> LIMITATION

- Options are not supported

UDP

> STATUS

Protocol	RFC	Status	Tags
UDP	rfc768	Fully	basic, network, internet

> LAYER DETECTION METHODS

- Explicit detection (IP Protocol Type)

> FIELDS

Name	Type	Length	Mask	Multiple	Description
root	**uint8**	**1**	ffffffffffffff	false	Layer presented flag.
raw_data	**byte-sequence**	**0**	ffffffffffffff	false	Layer data.
raw_data_length	**uint64**	**8**	ffffffffffffff	false	Layer data length.
payload_data	**byte-sequence**	**0**	ffffffffffffff	false	The payload data - data which is placed right after the layer data.
payload_data_length	**uint64**	**8**	ffffffffffffff	false	The payload data length.
src_port	**uint16**	**2**	ffffffffffffff	false	Source port.
dst_port	**uint16**	**2**	ffffffffffffff	false	Destination port.
length	**uint16**	**2**	ffffffffffffff	false	Length is the length in octets of this user datagram including this header and the data.

checksum	**uint16**	**2**	ffffffffffffff	false	Checksum is the 16-bit one's complement of the one's complement sum of a pseudo header of information from the IP header, the UDP header, and the data, padded with zero octets at the end (if necessary) to make a multiple of two octets.

Session

TLS

> STATUS

Protocol	RFC	Status	Tags
TLS	rfc5246 (v1.2) rfc6066 (Extensions) rfc8446 (v1.3) rfc3749 (Compression methods) rfc3943 (LZS compression id) rfc5077 (New Session Ticket) rfc7301 (Application-Layer Protocol Negotiation Extension) rfc4492 (Elliptic Curve Cryptography (ECC)) rfc5289 (Ecdhe Cipher Suites) rfc2246 (v1.0) rfc4346 (v1.1) rfc6101 (ssl v3.0) rfc9001 (QUIC Transport Parameters Extension)	Partly	basic, network, internet

> LAYER DETECTION METHODS

- Port-based
- Try dissect-

> PORTS

- 443 (**tcp**)
- 993 (**tcp**; IMAPS)
- 995 (**tcp**; POP3S)

> FIELDS

Name	Type	Length	Mask	Multiple	Description
root	**uint8**	**1**	ffffffffffffff	false	Layer presented flag.
raw_data	**byte-sequence**	**0**	ffffffffffffff	false	Layer data.
raw_data_length	**uint64**	**8**	ffffffffffffff	false	Layer data length.
payload_data	**byte-sequence**	**0**	ffffffffffffff	false	The payload data - data which is placed right after the layer data.
payload_data_length	**uint64**	**8**	ffffffffffffff	false	The payload data length.
record	**byte-sequence**	**0**	ffffffffffffff	true	Record layer.
record_content_type	**uint8**	**1**	ffffffffffffff	true	The higher-level protocol used to process the enclosed fragment/message.
record_protocol_version	**uint16**	**2**	ffffffffffffff	true	The version of the protocol being employed.
record_protocol_major_version	**uint8**	**1**	ffffffffffffff	true	The major number of protocol version.
record_protocol_minor_version	**uint8**	**1**	ffffffffffffff	true	The minor number of protocol version.
record_message_length	**uint16**	**2**	ffffffffffffff	true	The length (in bytes) of the following TLSPlaintext.fragment. The length MUST NOT exceed 2^14.
record_message	**byte-sequence**	**0**	ffffffffffffff	true	The application data. This data is transparent and treated as an independent block to be dealt with by the higher-level protocol specified by the type field.
heartbeat_message_type	**uint8**	**1**	ffffffffffffff	true	The message type, either heartbeat_request (**1**) or heartbeat_response (**2**).
heartbeat_payload_length	**uint16**	**2**	ffffffffffffff	true	The length of the payload.

heartbeat_ payload	**byte-sequence**	**0**	ffffffffffffff	true	The padding is random content that MUST be ignored by the receiver. The padding_length MUST be at least 16.
signature_ scheme	**uint16**	**2**	ffffffffffffff	true	The field specifies hash and signature algorithm. The field can exist only for tls v1.3 sessions.
change_ci pher_spec _type	**uint8**	**1**	ffffffffffffff	true	The change cipher spec protocol exists to signal transitions in ciphering strategies. The protocol consists of a single message, which is encrypted and compressed under the current (not the pending) connection state. The message consists of a single byte of value 1.
alert_level	**uint8**	**1**	ffffffffffffff	true	Alert message level.
alert_desc ription	**uint8**	**1**	ffffffffffffff	true	Alert message description..
handshake _header	**byte-sequence**	**4**	ffffffffffffff	true	The header of handshake protocol. The TLS Handshake Protocol is one of the defined higher-level clients of the TLS Record Protocol. This protocol is used to neg otiate the secure attributes of a session. Handshake messages are supplied to the TLS record layer, where they are encapsulated within one or more TLSPlaintext structures, which are processed and transmitted as specified by the current active session state.
handshake _type	**32-bit-field**	**4**	ff000000	true	The handshake message type: **0** (hello_request), **1** (clien t_hello), **2** (server_hello), **11** (c ertificate), **12** (server_key_exch ange), **13** (certificate_request), **14** (server_hello_done), **15** (ce rtificate_verify), **16** (client_key _exchange), **20** (finished), **255**.
handshake _message _length	**32-bit-field**	**4**	ffffff	true	The length of handshake message.

client_version	**uint16**	**2**	ffffffffffffffff	true	The version of the TLS protocol by which the client wishes to communicate during this session.
client_major_version	**uint8**	**1**	ffffffffffffffff	true	The major number of client TLS protocol.
client_minor_version	**uint8**	**1**	ffffffffffffffff	true	The minor number of client TLS client protocol.
server_version	**uint16**	**2**	ffffffffffffffff	true	This field will contain the lower of that suggested by the client in the clienthello and the highest supported by the server.
server_major_version	**uint8**	**1**	ffffffffffffffff	true	The major number of server TLS protocol.
server_minor_version	**uint8**	**1**	ffffffffffffffff	true	The minor number of server TLS protocol.
random	**byte-sequence**	**32**	ffffffffffffffff	true	A client/server generated random structure. The structure which is generated by the server MUST be independently generated from the ClientHello.random. (Client/Server handshake header field)
random_gmt_unix_time	**uint32**	**4**	ffffffffffffffff	true	The current time and date in standard UNIX 32-bit format (seconds since the midnight starting Jan 1, 1970, UTC, ignoring leap seconds) according to the sender's internal clock. (Client/Server handshake header field)
random_bytes	**byte-sequence**	**28**	ffffffffffffffff	true	28 bytes generated by a secure random number generator. (Client/Server handshake header field)
session_id_length	**uint8**	**1**	ffffffffffffffff	true	The length of session id. (Client/Server handshake header field)
session_id	**byte-sequence**	**0**	ffffffffffffffff	true	Id of the session corresponding to this connection. (Client/Server handshake header field)

cipher_suites_length	**uint16**	**2**	ffffffffffffffff	true	The length of cipher suites field. (Client handshake header field)
cipher_suites	**uint-16-array**	**0**	ffffffffffffffff	true	This is a list of the cryptographic options supported by the client, with the client’s first preference first. (Client handshake header field)
cipher_suite	**uint16**	**2**	ffffffffffffffff	true	For client: an element of cipher suites. For server: the single cipher suite selected by the server from the client cipher suite list. (Client/Server handshake header field)
compression_methods_length	**uint8**	**1**	ffffffffffffffff	true	The length of compression methods field. (Client handshake header field)
compression_methods	**uint-8-array**	**0**	ffffffffffffffff	true	This is a list of the compression methods supported by the client, sorted by client preference. (Client handshake header field)
compression_method	**uint8**	**1**	ffffffffffffffff	true	For client: an element of compression methods. For server: the single compression algorithm selected by the server from the client compression method list. (Client/Server handshake header field)
extensions	**byte-sequence**	**0**	ffffffffffffffff	true	A list of extensions. Clients MAY request extended functionality from servers by sending data in the extensions field. Note that only extensions offered by the client can appear in the server’s list. (Client/Server handshake header field)
extensions_length	**uint16**	**2**	ffffffffffffffff	true	The length of extensions field.
extension	**byte-sequence**	**0**	ffffffffffffffff	true	Extension record/unit.

extension_length	**uint16**	**2**	ffffffffffffff	true	The length of extension data. (Client/Server handshake header field)
server_name_list_length	**uint16**	**2**	ffffffffffffff	true	The length of server name list.
server_name_list	**byte-sequence**	**0**	ffffffffffffff	true	The list of server name elements.
server_name_type	**uint8**	**1**	ffffffffffffff	true	The type of server name: **0** (host_name), **255**.
server_name	**ascii-string**	**0**	ffffffffffffff	true	The server name string.
protocol_name_list_length	**uint16**	**2**	ffffffffffffff	true	The length of protocol name list.
protocol_name_list	**byte-sequence**	**0**	ffffffffffffff	true	The list contains the list of protocols advertised by the client, in descending order of preference.
protocol_name_length	**uint8**	**1**	ffffffffffffff	true	The length of protocol name.
protocol_name	**ascii-string**	**0**	ffffffffffffff	true	The protocol name string.
supported_versions_length	**uint8**	**1**	ffffffffffffff	true	The length of supported versions.
supported_versions	**byte-sequence**	**0**	ffffffffffffff	true	The list of supported versions in preference order, with the most preferred version first.
supported_version	**uint16**	**2**	ffffffffffffff	true	A supported version.
quic_transport_parameter	**byte-sequenc**	**0**	ffffffffffffff	true	The quic transport parameter section.
quic_transport_parameter_id	**byte-sequence**	**0**	ffffffffffffff	true	The identificator of quic transport parameter.
quic_transport_parameter_length	**byte-sequence**	**0**	ffffffffffffff	true	The field contains the length of the Transport Parameter Value field in bytes.

quic_transport_parameter_value	**uint16**	**2**	ffffffffffffff	true	The length of signature and hash algorithms field.
signature_and_hash_algorithms	**byte-sequence**	**0**	ffffffffffffff	true	Signature and hash algorithm elements.
signature_and_hash_algorithm	**uint16**	**2**	ffffffffffffff	true	The hash and signature algorithm pair.
signature_length	**uint16**	**2**	ffffffffffffff	true	The length of signature field.
signature	**byte-sequence**	**0**	ffffffffffffff	true	A digital signature using algorithms over the contents of the element.
request_update	**uint8**	**1**	ffffffffffffff	true	If the request_update field is set to update_requested (**0**), then the receiver MUST send a KeyUpdate of its own with request_update set to update_not_requested (**1**) prior to sending its next Application Data record.
verify_data	**byte-sequence**	**0**	ffffffffffffff	true	The part of finished message. For tls v1.0 the length is fixed.
md5_hash	**byte-sequence**	**16**	ffffffffffffff	true	The part of finished message. The field can exist only for ssl v3.0 sessions. The length is fixed.
sha_hash	**byte-sequence**	**20**	ffffffffffffff	true	The part of finished message. The field can exist only for ssl v3.0 sessions. The length is fixed.
session_ticket_lifetime	**uint32**	**4**	ffffffffffffff	true	Indicates the lifetime in seconds as a 32-bit unsigned integer in network byte order from the time of ticket issuance.
session_ticket_length	**uint16**	**2**	ffffffffffffff	true	The length of session ticket field.
session_ticket	**byte-sequence**	**0**	ffffffffffffff	true	The session ticket field.

message_hash_data	**byte-sequence**	**0**	ffffffffffffffff	true	The data section of message hash handshake protocol.
certificate_list_length	**byte-sequence**	**3**	ffffffffffffffff	true	The length of certificate list field.
certificate_list	**byte-sequence**	**0**	ffffffffffffffff	true	The certificate list data. The certificate list can contain more than one certificate.
certificate_length	**byte-sequence**	**3**	ffffffffffffffff	true	The length of certificate.
certificate	**byte-sequence**	**0**	ffffffffffffffff	true	The certificate data.
premaster_key_length	**uint16**	**2**	ffffffffffffffff	true	The length of premaster key.
premaster_key	**byte-sequence**	**0**	ffffffffffffffff	true	The value which client generates and sends as encrypted premaster secret message. The field exists only for RSA key agreement.
dh_public_key_length	**uint16**	**2**	ffffffffffffffff	true	Client Diffie-Hellman public value length.
dh_public_key	**byte-sequence**	**0**	ffffffffffffffff	true	Client Diffie-Hellman public value.
dhe_public_key_length	**uint16**	**2**	ffffffffffffffff	true	Client Ephemeral Diffie-Hellman public value length
dhe_public_key	**byte-sequence**	**0**	ffffffffffffffff	true	Client Ephemeral Diffie-Hellman public value.
ecdhe_public_key_length	**uint8**	**1**	ffffffffffffffff	true	Client Elliptic Curve Ephemeral Diffie-Hellman public value length.
ecdhe_public_key	**byte-sequence**	**0**	ffffffffffffffff	true	Client Elliptic Curve Ephemeral Diffie-Hellman public value.
ecdh_public_key_length	**uint8**	**1**	ffffffffffffffff	true	Client/Server Elliptic Curve Diffie-Hellman public value length. For Server Key Exchange maessage that field exists only when curve_type has named_curve(**3**) value.

ecdh_public_key	**byte-sequence**	**0**	ffffffffffffff	true	Client/Server Elliptic Curve Diffie-Hellman public value. For Server Key Exchange maessage that field exists only when curve_type has named_curve (**3**) value.
fortezza_yc_length	**uint8**	**1**	ffffffffffffff	true	The client's Yc value (public key) length.
fortezza_yc	**byte-sequence**	**0**	ffffffffffffff	true	The client's Yc value (public key) for the KEA calculation.
fortezza_rc	**byte-sequence**	**128**	ffffffffffffff	true	The client's Rc value for the KEA calculation.
fortezza_yc_signature	**byte-sequence**	**40**	ffffffffffffff	true	The tsignature of the KEA public key, signed with the client's DSS private key.
fortezza_wrapped_client_write_key	**byte-sequence**	**12**	ffffffffffffff	true	This is the client's write key, wrapped by the TEK.
fortezza_wrapped_server_write_key	**byte-sequence**	**12**	ffffffffffffff	true	This is the server's write key, wrapped by the TEK.
fortezza_client_write_iv	**byte-sequence**	**24**	ffffffffffffff	true	The IV for the client write key.
fortezza_server_write_iv	**byte-sequence**	**24**	ffffffffffffff	true	The IV for the server write key.
fortezza_master_write_iv	**byte-sequence**	**24**	ffffffffffffff	true	This is the IV for the TEK used to encrypt the premaster secret.
fortezza_encrypted_pre_master_secret	**byte-sequence**	**48**	ffffffffffffff	true	A random value, generated by the client and used to generate the master secret.
dh_p_length	**uint16**	**2**	ffffffffffffff	true	The prime modulus field length.
dh_p	**byte-sequence**	**0**	ffffffffffffff	true	The prime modulus used for the Diffie-Hellman operation.

dh_g_length	**uint16**	**2**	ffffffffffffffff	true	The generator field length.
dh_g	**byte-sequence**	**0**	ffffffffffffffff	true	The generator used for the Diffie-Hellman operation.
dh_ys_length	**uint16**	**2**	ffffffffffffffff	true	The server's Diffie-Hellman public value field length.
dh_ys	**byte-sequence**	**0**	ffffffffffffffff	true	The server's Diffie-Hellman public value (g^X mod p).
dh_signature_and_hash_algorithm	**uint16**	**2**	ffffffffffffffff	true	The length of dh signature field.
dh_signature	**byte-sequence**	**0**	ffffffffffffffff	true	The dh signature.
rsa_modulus_length	**uint16**	**2**	ffffffffffffffff	true	The length of rsa modulus field.
rsa_modulus	**byte-sequence**	**0**	ffffffffffffffff	true	The modulus of the server's temporary RSA key.
rsa_exponent_length	**uint16**	**2**	ffffffffffffffff	true	The length of rsa exponent field.
rsa_exponent	**byte-sequence**	**0**	ffffffffffffffff	true	The public exponent of the server's temporary RSA key.
fortezza_rs	**byte-sequence**	**128**	ffffffffffffffff	true	Server random number for FORTEZZA KEA (Key Exchange Algorithm).
curve_type	**uint8**	**1**	ffffffffffffffff	true	The field identifies the type of the elliptic curve domain parameters.
named_curve	**uint16**	**2**	ffffffffffffffff	true	The field specifies a recommended set of elliptic curve domain parameters. All those values of NamedCurve are allowed that refer to a specific curve.
ecdh_signature_and_hash_algorithm	**uint16**	**2**	ffffffffffffffff	true	The ecdh hash and signature algorithm pair.

ecdh_signature_length	**uint16**	**2**	ffffffffffffff	true	The length of ecdh signature field.
ecdh_signature	**byte-sequence**	**0**	ffffffffffffff	true	The ecdh signature.
ecdh_prime_length	**uint8**	**1**	ffffffffffffff	true	The odd prime value length. The field exists only for explicit_prime curve_type.
ecdh_prime	**byte-sequence**	**0**	ffffffffffffff	true	The odd prime defining the field Fp. The field exists only for explicit_prime curve_type.
ecdh_m	**uint16**	**2**	ffffffffffffff	true	The degree of the characteristic-2 field F2^m. The field exists only for explicit_char2 curve_type.
ecdh_basis	**uint8**	**1**	ffffffffffffff	true	The basis type. Possible values: ec_basis_trinomial (**1**), ec_basis_pentanomial (**2**). The field exists only for explicit_char2 curve_type.
ecdh_k_length	**uint8**	**1**	ffffffffffffff	true	The exponent k value length.
ecdh_k	**byte-sequence**	**0**	ffffffffffffff	true	The exponent k for the trinomial basis representation x^m | x^k | 1. The field exists for explicit_char2 curve_type and ec_trinomial basis.
ecdh_k1_length	**uint8**	**1**	ffffffffffffff	true	The exponent k1 value length.
ecdh_k1	**byte-sequence**	**0**	ffffffffffffff	true	The exponents for the pentanomial representation x^m | x^k3 | x^k2 | x^k1 | 1 (such that k3 > k2 > k1). The field exists only for explicit_char2 curve_type and ec_pentanomial basis.
ecdh_k2_length	**uint8**	**1**	ffffffffffffff	true	The exponent k2 value length.

ecdh_k2	**byte-sequence**	**0**	ffffffffffffff	true	The exponents for the pentanomial representation x^m | x^k3 | x^k2 | x^k1 | 1 (such that k3 > k2 > k1). The field exists only for explicit_char2 curve_type and ec_pentanomial basis.
ecdh_k3_length	**uint8**	**1**	ffffffffffffff	true	The exponent k value length.
ecdh_k3	**byte-sequence**	**0**	ffffffffffffff	true	The exponents for the pentanomial representation x^m | x^k3 | x^k2 | x^k1 | 1 (such that k3 > k2 > k1). The field exists only for explicit_char2 curve_type and ec_pentanomial basis.
ecdh_curve	**byte-sequence**	**0**	ffffffffffffff	true	The field specifies the coefficients a and b of the elliptic curve E. The field exists for explicit_prime or explicit_char2 curve_type.
ecdh_curve_a_length	**uint8**	**1**	ffffffffffffff	true	The a value of the elliptic curve length. The field exists for explicit_prime or explicit_char2 curve_type.
ecdh_curve_a	**byte-sequence**	**0**	ffffffffffffff	true	The a value of the elliptic curve. The field exists for explicit_prime or explicit_char2 curve_type.
ecdh_curve_b_length	**uint8**	**1**	ffffffffffffff	true	The b value of the elliptic curve length. The field exists for explicit_prime or explicit_char2 curve_type.
ecdh_curve_b	**byte-sequence**	**0**	ffffffffffffff	true	The b value of the elliptic curve. The field exists for explicit_prime or explicit_char2 curve_type.
ecdh_base_length	**uint8**	**1**	ffffffffffffff	true	The field specifies the base point G value length. The field exists for explicit_prime or explicit_char2 curve_type.

ecdh_base	**byte-sequence**	**0**	ffffffffffffff	true	The field specifies the base point G on the elliptic curve. The field exists for explicit_prime or explicit_char2 curve_type.
ecdh_order_length	**uint8**	**1**	ffffffffffffff	true	The field specifies the order n of the base point value length. The field exists for explicit_prime or explicit_char2 curve_type.
ecdh_order	**byte-sequence**	**0**	ffffffffffffff	true	The field specifies the order n of the base point. The field exists for explicit_prime or explicit_char2 curve_type.
ecdh_cofactor_length	**uint8**	**1**	ffffffffffffff	true	The field specifies the cofactor h value length. The field exists for explicit_prime or explicit_char2 curve_type.
ecdh_cofactor	**byte-sequence**	**0**	ffffffffffffff	true	The field specifies the cofactor h = #E(Fq)/n, where #E(Fq) represents the number of points on the elliptic curve E defined over the field Fq (either Fp or F2^m). The field exists for explicit_prime or explicit_char2 curve_type.

> FIELD TREE

- **Root**

```
.
└── record/
    ├── record_content_type
    ├── record_protocol_version
    ├── record_protocol_major_version
    ├── record_protocol_minor_version
    ├── record_message_length
    └── record_message/
        ├── [Heartbeat]
        ├── heartbeat_message_type
        ├── heartbeat_payload_length
        ├── heartbeat_payload
        ├── heartbeat_padding
        ├── [Certificate Verify]
        ├── signature_scheme
        ├── [Change Cipher Spec Type]
        ├── change_cipher_spec_type
        ├── [Alert]
        ├── alert_level
        ├── alert_description
        ├── [Handshake]
        ├── handshake_header
        ├── handshake_type
        ├── handshake_message_length
        └── handshake_message
```

- **Handshake Header**

```
.
└── handshake_message/
    ├── handshake_type
    └── handshake_message_length
```

- **Handshake Message**

```
.
└── handshake_message/
    ├── [ClientHello only]
    ├── client_version
    ├── client_major_version
    ├── client_minor_version
    ├── [ServerHello only]
    ├── server_version
    ├── server_major_version
    ├── server_minor_version
    ├── [ClientHello and ServerHello]
    ├── random
    ├── random_gmt_unix_time
    ├── random_bytes
    ├── session_id_length
    ├── session_id
    ├── cipher_suites_length
    ├── cipher_suites
    ├── compression_methods_length
    ├── compression_methods
    ├── compression_method
    ├── extensions (also can belong Handshake Enctypted Extensions message)
    └── extensions_length
```

- **Extensions**

```
.
└── extensions/
    ├── extensions_length
    └── extension/
        ├── extension_type
        ├── extension_length
        └── ... (extension related fields)
```

or

```
.
└── extensions/
    ├── extensions_length
    ├── extension_type
    └── extension_length
```

when extension size cannot be defined.

- **Quic Transport Parameter extension**

```
.
└── quic_transport_parameter/
    ├── quic_transport_parameter_id
    ├── quic_transport_parameter_length
    └── quic_transport_parameter_value
```

- **EcDhCurve**

```
.
└── ecdh_curve/
    ├── ecdh_curve_a_length
    ├── ecdh_curve_a
    ├── ecdh_curve_b_length
    └── ecdh_curve_b
```

> SUB-PROTOCOL SUPPORT LIST

- [x] Alert protocol
- [x] Application Data protocol
- [x] Heartbeat protocol
- [ ] Handshake protocol
    - [x] HelloRequest
    - [x] ClientHello
    - [x] ServerHello
    - [x] NewSessionTicket (rfc5077)
    - [x] EndOfEarlyData (rfc8446)
    - [x] EncryptedExtensions (rfc8446)
    - [ ] Certificate
    - [x] ServerKeyExchange
    - [ ] CertificateRequest (1.2 and 1.3 have differences)
    - [x] ServerHelloDone
    - [x] CertificateVerify
    - [x] ClientKeyExchange
    - [x] Finished
    - [x] KeyUpdate (rfc8446)
    - [x] MessageHash (rfc8446)
- [x] Change Cipher Spec protocol
- [ ] Extensions
    - [x] ServerName
    - [ ] MaxFragmentLength
    - [ ] ClientCertificateUrl
    - [ ] TrustedCaKeys
    - [ ] TrustedHmac
    - [ ] StatusRequest
    - [ ] SupportedGroups
    - [ ] SignatureAlgorithms
    - [ ] UseSrtp
    - [ ] Heartbeat
    - [ ] ApplicationLayerProtocolNegotiation
    - [ ] SignedCertificateTimestamp
    - [ ] ClientCertificateType
    - [ ] ServerCertificateType
    - [ ] Padding
    - [ ] Reserved0
    - [ ] PreSharedKey
    - [ ] EarlyData

- [x] SupportedVersions
- [ ] Cookie
- [ ] PskKeyExchangeModes
- [ ] Reserved1
- [ ] CertificateAuthorities
- [ ] OidFilters
- [ ] PostHandshakeAuth
- [ ] SignatureAlgorithmsCert
- [ ] KeyShare
- [ ] ConnectionId (rfc9146)
- [ ] QuicTransportParameters (rfc9001)

- [ ] Not enumerated extensions (have to be found in rfc):
  - [ ] EcPointFormats
  - [ ] SignedCertificateTimestamp
  - [ ] RenegotiationInfo
  - [ ] SessionTicket
  - [ ] NextProtocolNegotiation
  - [ ] ExtendMasterSecret

> LIMITATION

- TLSPGP is not supported (rfc5081)
- New Session Ticket is implemented related to rfc5077 (rfc8446 has a different structure)

> NOTES

- CertificateVerify message contains SignatureAndHashAlgorithm (rfc52 46) field which is the same with SignatureScheme (rfc8446). Since that field is different between tls1.2 and tls1.3 – it has to be interpreted depending on flow context (tls version).

  SignatureAndHashAlgorithm::HashAlgorithm a nd SignatureAndHashAlgorithm::SignatureAlg orithm are not used in dissection to save that field universal for 1.2 and 1.3 versions.

- CertificateRequest and Certificate message dissection are not supported. For extracting base fields of certificates – use tls_certificate in-built extension.

- If the TLS layer has QUIC transport – it leads to handshake only dissection.

DTLS

> STATUS

Protocol	RFC	Status	Tags
TLS	rfc6347 (v1.2) rfc9146(Compression Identifier) rfc9147 (v1.3)	Partly	basic, network, internet

> LAYER DETECTION METHODS

- Port-based
- Try dissect

> PORTS

- 443 (**udp**)

> FIELDS

Name	Type	Length	Mask	Multiple	Description
record_connection_id	**byte-sequence**	**0**	ffffffffffffff	true	The connection identificator. The field is presented when fixed bits (3 higher bits) of record content type is equal to 1 and when previous session packets have been processed.
record_numbers_length	**uint16**	**2**	ffffffffffffff	true	The record numbers length of ACK record.
record_numbers	**byte-sequence**	**0**	ffffffffffffff	true	A list of the records containing handshake messages in the current flight which the endpoint has received and either processed or buffered, in numerically increasing order.

record_number	**byte-sequence**	**16**	ffffffffffffff	true	The structure: epoch:sequence_number. Each field is occupied 64 bits. This 128-bit value is used in the ACK message as well as in the "record_sequence_number" input to the Authenticated Encryption with Associated Data (AEAD) function.
record_epoch	**uint16**	**2**	ffffffffffffff	true	A counter value that is incremented on every cipher state change.
record_sequence_number	**byte-sequence**	**0**	ffffffffffffff	true	The sequence number for this record. The length of the field depends on record content type.
server_protocol_minor_version	**uint8**	**1**	ffffffffffffff	true	The minor number of protocol version. HelloVerifyRequest message.
server_protocol_major_version	**uint8**	**1**	ffffffffffffff	true	The major number of protocol version. HelloVerifyRequest message.
handshake_message_sequence	**64-bit-field**	**8**	ffff000000000000	true	The message sequence number.
handshake_fragment_offset	**64-bit-field**	**8**	ffffff000000	true	The fragment offset.
handshake_fragment_length	**64-bit-field**	**8**	ffffff	true	The fragment length.
server_protocol_version	**uint16**	**2**	ffffffffffffff	true	The version of the protocol being employed. HelloVerifyRequest message.
server_cookie_length	**uint8**	**1**	ffffffffffffff	true	The server cookie value length. HelloVerifyRequest message.
server_cookie	**byte-sequence**	**0**	ffffffffffffff	true	The server cookie value. HelloVerifyRequest message.
connection_id_length	**uint8**	**1**	ffffffffffffff	true	The length of connection_id field data.
num_cids	**uint8**	**1**	ffffffffffffff	true	The number of CIDs desired.

connection_id	**byte-sequence**	**0**	ffffffffffffff	true	The connection identificator.
cids_length	**uint16**	**2**	ffffffffffffff	true	The length of cids field data.
cids	**byte-sequence**	**0**	ffffffffffffff	true	Indicates the set of CIDs that the sender wishes the peer to use.
new_connection_id_length	**uint8**	**1**	ffffffffffffff	true	The length of connection_id field data. The part of NewConnectionId message.
new_connection_id	**byte-sequence**	**0**	ffffffffffffff	true	The connection identificator. The part of NewConnectionId message.
connection_id_usage	**uint8**	**1**	ffffffffffffff	true	Indicates whether the new CIDs should be used immediately or are spare. If usage is set to "cid_immediate", then one of the new CIDs MUST be used immediately for all future records. If it is set to "cid_spare", then either an existing or new CID MAY be used.

> FIELD TREE

- **Root**

```
.
└── record/
    ├── ...
    ├── record_connection_id
    ├── record_epoch
    ├── record_sequence_number
    └── ...
```

- **Handshake Header**

```
.
└── handshake_header/
    ├── handshake_message_sequence
    ├── handshake_fragment_offset
    ├── handshake_fragment_length
    └── handshake_message_length
```

- **Handshake Message**

```
.
└── handshake_message/
    ├── server_protocol_version/
    ├── server_cookie_length
    ├── server_cookie
    ├── ...
    ├── cookie_length
    ├── cookie
    ├── handshake_header/
    ├── ...
    ├── record_numbers_length
    ├── record_numbers
    ├── record_number
    ├── ...
    ├── num_cids
    ├── cids_length
    ├── cids
    ├── new_connection_id_length
    ├── new_connection_id
    ├── connection_id_usage
    └── ...
```

- **Server Protocol Version**

```
.
└── server_protocol_version/
    ├── server_protocol_major_version
    ├── server_protocol_minor_version
    └── server_protocol_minor_version
```

- **Connection Id Extension**

```
.
└── extension/
    ├── connection_id_length
    └── connection_id
```

> LIMITATION

- The protocol implementation supports storing Connection ID from Connection ID extension, but doesn't store the new ones from NewConnectionId session messages.
- The protocol might be detected only over UDP. RFC mentions about DTLS over TCP and SCTP as well, but these transport layers are not supported for DTLS.
- If the DTLS layer is created without previous layer - it leads to handshake only dissection. The transport layer specifies how the next layer has to be dissected. If the previous layer is absent, the library assumes the DTLS layer is assembled (fragmented message). Fragmented messages are possible only for handshake protocol.
- Assembled DTLS messages are separated to different packet frames.

Presentation

None

Application

Telnet

> STATUS

Protocol	RFC	Status	Tags
Telnet	rfc854	Fully	basic, network, internet

> LAYER DETECTION METHODS

- Port-based

> PORTS

- 23 (tcp)

> FIELDS

Name	Type	Length	Mask	Multiple	Description
root	**uint8**	**1**	ffffffffffffff	false	Layer presented flag.
raw_data	**byte-sequence**	**0**	ffffffffffffff	false	Layer data.
raw_data_length	**uint64**	**8**	ffffffffffffff	false	Layer data length.
payload_data	**byte-sequence**	**0**	ffffffffffffff	false	The payload data - data which is placed right after the layer data.
payload_data_length	**uint64**	**8**	ffffffffffffff	false	The payload data length.
data	**byte-sequence**	**0**	ffffffffffffff	false	Stream data.

DNS

> STATUS

Protocol	RFC	Status	Tags
DNS	rfc1035 rfc3596 rfc2874	Fully	basic, network, internet

> LAYER DETECTION METHODS

- Port-based
- Try Dissect

> PORTS

- 53 (udp/tcp)

> FIELDS

Name	Type	Length	Mask	Multiple	Description
root	**uint8**	**1**	ffffffffffffff	false	Layer presented flag.
raw_data	**byte-sequence**	**0**	ffffffffffffff	false	Layer data.
raw_data_length	**uint64**	**8**	ffffffffffffff	false	Layer data length.
payload_data	**byte-sequence**	**0**	ffffffffffffff	false	The payload data - data which is placed right after the layer data.
payload_data_length	**uint64**	**8**	ffffffffffffff	false	The payload data length.
dns_message_length	**uint16**	**2**	ffffffffffffff	false	Dns message length - is presented only for tcp transport.

id	**uint16**	**2**	ffffffffffffff	false	A **16** bit identifier assigned by the program that generates any kind of query. This identifier is copied the corresponding reply and can be used by the requester to match up replies to outstanding queries.
qr	**uint16**	**2**	8000	false	Bit specifies message type. Query (**0**), response (**1**).
opcode	**uint16**	**2**	7800	false	A four bit field that specifies kind of query in this message. This value is set by the originator of a query and copied into the response. The values are: **0** (standard query), **1** (inverse query), **2** (server status request), **3**-**15** (reserved for future use).
aa	**uint16**	**2**	400	false	Authoritative Answer - this bit is valid in responses, and specifies that the responding name server is an authority for the domain name in question section.
tc	**uint16**	**2**	200	false	TrunCation - specifies that this message was truncated due to length greater than that permitted on the transmission channel.
rd	**uint16**	**2**	100	false	Recursion Desired - this bit may be set in a query and is copied into the response. If RD is set, it directs the name server to pursue the query recursively. Recursive query support is optional.
ra	**uint16**	**2**	80	false	Recursion Available - this be is set or cleared in a response, and denotes whether recursive query support is available in the name server.
z	**uint16**	**2**	70	false	Reserved for future use. Must be zero in all queries and responses.

rcode	**uint16**	**2**	f	false	Response code – this 4 bit field is set as part of responses. The values have the following interpretation: 0 (No error condition), **1** (Format error – The name server was unable to interpret the query), **2** (Server failure – The name server was unable to process this query due to a problem with the name server), **3** (Name Error – Meaningful only for responses from an authoritative name server, this code signifies that the domain name referenced in the query does not exist), **4** (Not Implemented – The name server does not support the requested kind of query), **5** (Refused – The name server refuses to perform the specified operation for policy reasons. For example, a name server may not wish to provide the information to the particular requester, or a name server may not wish to perform a particular operation (e.g. zone transfer) for particular data), **6-15** (Reserved for future use).
qdcount	**uint16**	**2**	ffffffffffffff	false	An unsigned 16 bit integer specifying the number of entries in the question section.
ancount	**uint16**	**2**	ffffffffffffff	false	An unsigned 16 bit integer specifying the number of resource records in the answer section.
nscount	**uint16**	**2**	ffffffffffffff	false	An unsigned 16 bit integer specifying the number of name server resource records in the authority records section.
arcount	**uint16**	**2**	ffffffffffffff	false	An unsigned 16 bit integer specifying the number of resource records in the additional records section.
queries	**byte-sequence**	**0**	ffffffffffffff	false	Question section.
query	**byte-sequence**	**0**	ffffffffffffff	true	Query record.

qname	**byte-sequence**	**0**	ffffffffffffffff	true	A domain name represented as a sequence of labels, where each label consists of a length octet followed by that number of octets. The domain name terminates with the zero length octet for the null label of the root. Note that this field may be an odd number of octets; no padding is used.
qtype	**uint16**	**2**	ffffffffffffffff	true	A two octet code which specifies the type of the query. The values for this field include all codes valid for a TYPE field, together with some more general codes which can match more than one type of RR.
qclass	**uint16**	**2**	ffffffffffffffff	true	A two octet code that specifies the class of the query. For example, the QCLASS field is IN for the Internet.
answers	**byte-sequence**	**0**	ffffffffffffffff	false	Answer section.
authority_records	**byte-sequence**	**0**	ffffffffffffffff	false	Authority records section.
additional_records	**byte-sequence**	**0**	ffffffffffffffff	false	Resource record.
domain_name	**byte-sequence**	**0**	ffffffffffffffff	true	A domain name to which this resource record pertains.
domain_name_label_length	**uint8**	**1**	ffffffffffffffff	true	A domain name label length.
domain_name_label	**ascii-string**	**0**	ffffffffffffffff	true	A domain name label.
domain_name_pointer	**uint16**	**2**	ffffffffffffffff	true	A domain name pointer.
domain_name_offset	**uint16**	**2**	3fff	true	A domain name offset.
rdata_type	**uint16**	**2**	ffffffffffffffff	true	Specifies the meaning of the data in the rdata.
rdata_class	**uint16**	**2**	ffffffffffffffff	true	Specifies the class of the data in the rdata.

ttl	**uint32**	**4**	ffffffffffffffff	true	Specifies the time interval (in seconds) that the resource record may be cached before it should be discarded.
rd_length	**uint16**	**2**	ffffffffffffffff	true	Specifies the length in octets of the rdata.
rdata	**byte-sequence**	**0**	ffffffffffffffff	true	A variable length string of octets that describes the resource.
nsdname	**byte-sequence**	**0**	ffffffffffffffff	true	A domain name which specifies a host which should be authoritative for the specified class and domain.
mb_madname	**byte-sequence**	**0**	ffffffffffffffff	true	A domain name which specifies a host which has a mail agent for the domain which should be able to deliver mail for the domain.
mf_madname	**byte-sequence**	**0**	ffffffffffffffff	true	A domain name which specifies a host which has a mail agent for the domain which will accept mail for forwarding to the domain.
cname	**byte-sequence**	**0**	ffffffffffffffff	true	A domain name which specifies the canonical or primaryname for the owner.
mgname	**byte-sequence**	**0**	ffffffffffffffff	true	A domain name which specifies a mailbox which is a member of the mail group specified by the domain name.
newname	**byte-sequence**	**0**	ffffffffffffffff	true	A domain name which specifies a mailbox which is the proper rename of the specified mailbox.
ptrdname	**byte-sequence**	**0**	ffffffffffffffff	true	A domain name which points to some location in the domain name space.
preference	**uint16**	**2**	ffffffffffffffff	true	Specifies the preference given to this RR among others at the same owner.
exchange	**byte-sequence**	**0**	ffffffffffffffff	true	A domain name which specifies a host willing to act as mail exchange for the owner name.

rmailbx	**byte-sequence**	**0**	ffffffffffffffff	true	A domain name which specifies a mailbox which is responsible for the mailing list or mailbox.
emailbx	**byte-sequence**	**0**	ffffffffffffffff	true	A domain name which specifies a mailbox which is to receive error messages related to the mailing list or mailbox specified by the owner of the MINFO RR.
txt_length	**uint8**	**1**	ffffffffffffffff	true	Txt string length.
txt	**ascii-string**	**0**	ffffffffffffffff	true	One or more character string(s). are used to hold descriptive text. the semantics of the text depends on the domain where it is found.
mname	**byte-sequence**	**0**	ffffffffffffffff	true	A domain name of the name server that was the original or primary source of data for this zone.
rname	**byte-sequence**	**0**	ffffffffffffffff	true	A domain name which specifies the mailbox of the person responsible for this zone.
serial	**uint32**	**4**	ffffffffffffffff	true	Version number of the original copy of the zone. zone transfers preserve this value.
refresh	**uint32**	**4**	ffffffffffffffff	true	Time interval before the zone should be refreshed.
retry	**uint32**	**4**	ffffffffffffffff	true	Time interval that should elapse before a failed refresh should be retried.
expire	**uint32**	**4**	ffffffffffffffff	true	Time value that specifies the upper limit on the time interval that can elapse before the zone is no longer authoritative.
minimum	**uint32**	**4**	ffffffffffffffff	true	Minimum ttl field that should be exported with any RR from this zone.
address	**uint32**	**4**	ffffffffffffffff	true	Internet address.

null_data	**byte-sequence**	**0**	ffffffffffffffff	true	Any data. Null section.
cpu_length	**uint8**	**1**	ffffffffffffffff	true	CPU string length.
cpu	**ascii-string**	**0**	ffffffffffffffff	true	A character-string which specifies the cpu type.
os_length	**uint8**	**1**	ffffffffffffffff	true	OS string length.
os	**ascii-string**	**0**	ffffffffffffffff	false	A character-string which specifies the operating system type.
wks_address	**uint32**	**4**	ffffffffffffffff	false	Internet address.
protocol	**uint8**	**1**	ffffffffffffffff	false	IP protocol number.
bit_mask	**byte-sequence**	**0**	ffffffffffffffff	true	Bit map has one bit per port of the specified protocol.
aaaa	**byte-sequence**	**16**	ffffffffffffffff	true	A **128** bit IPv6 address in network byte order (high-order byte first).
prefix_length	**uint8**	**1**	ffffffffffffffff	true	A prefix length, encoded as an eight-bit unsigned integer with value between 0 and 128 inclusive.
address_suffix	**byte-sequence**	**16**	ffffffffffffffff	true	An IPv6 address suffix, encoded in network order (high-order octet first).
prefix_name	**byte-sequence**	**0**	ffffffffffffffff	true	The name of the prefix, encoded as a domain name.

> FIELD TREE

- **Queries**

```
.
└── queries/
    ├── qname/
    |    ├── label-0
    |    ├── label-1
    |    ├── ...
    |    ├── qtype
    |    └── qclass
    └── qname/
        ├── label-0
        ├── ...
        ├── qtype
        └── qclass
```

- **Resource Records**

```
.
└── answers, authority_records, additional_records,
     └── record/
        ├── domain_name/
        |    ├── domain_name_label_length
        |    ├── domain_name_label
        |    ├── domain_name_pointer
        |    └── domain_name_offset
        ├── rdata_type
        ├── rdata_class
        ├── ttl
        ├── rd_length
        └── rdata
```

- **RData**

RData (Mb, Md, Mf, CName, Mg, Mr, Ptr)

```
.
└── rdata/
    └── mb_madname, md_madname, mf_madname, cname, mgname, newname, ptrdname/
        ├── domain_name_label_length
        ├── domain_name_label
        ├── domain_name_pointer
        └── domain_name_offset
```

RData Mx

```
.
└── rdata/
    ├── preference
    └── exchange/
        ├── domain_name_label_length
        ├── domain_name_label
        ├── domain_name_pointer
        └── domain_name_offset
```

RData MInfo

```
.
└── rdata/
    ├── rmailbx/
    |   ├── domain_name_label_length
    |   ├── domain_name_label
    |   ├── domain_name_pointer
    |   └── domain_name_offset
    ├── emailbx/
    |   ├── domain_name_label_length
    |   ├── domain_name_label
    |   ├── domain_name_pointer
    |   └── domain_name_offset
```

RData Txt

```
.
└── rdata/
    ├── txt_length
    └── txt
```

RData Soa

```
.
└── rdata/
    ├── mname
    ├── rname
    ├── serial
    ├── refresh
    ├── Retry
    ├── expire
    └── minimum
```

RData A

```
.
└── rdata/
    └── address
```

RData Null

```
.
└── rdata/
    └── null_data
```

RData HInfo

```
.
└── rdata/
    ├── cpu_length
    ├── cpu
    ├── os_length
    └── os
```

RData Wks

```
.
└── rdata/
    ├── wks_address
    ├── protocol
    └── bit_mask
```

> NOTES

- Resource Records have Name field in **rfc1035**. Our engine uses DomainName field.
- NsDname, MadName, CName, MgName, NewName, PtrDname have **domain name** structure. It means that fields have children fields such as DomainNameLabelLength, DomainNameLabel, DomainNamePointer, DomainNameOffset.
- Due to **rfc1035**, MadName field is presented in the following RData sections: Mb, Md, Mf. To don't use the same name for different RData sections and don't check the parent objects to detect a type, our engine uses MbMadName, MdMadName, MfMadName field names.
- The following malformed reasons are suitable for all fields which have **domain-name** structure:
  - DnsDomainNameHasInvalidFormat
  - DnsDomainNameFirstOctetCannotBeDissected
  - DnsDomainNamePointerCannotBeDissected
  - DnsResourceRecordHeaderCannotBeDissected
  - DnsResourceDataLengthExceedDataLength
  - DnsDomainNameLabelLengthExceedDataLength
  - DnsDomainNameOffsetExceedDataLength

MDNS

> STATUS

Protocol	RFC	Status	Tags
MDNS	rfc6762	Fully	basic, network, internet

> LAYER DETECTION METHODS

- Port-based
- Try dissect

> PORTS

- 5353 (udp/tcp)

> FIELDS

All **DNS fields** are valid for **MDNS** as well. But there are new fields which are presented below.

Name	Type	Length	Mask	Multiple	Description
unicast_response	**16-bit-field**	**2**	8000	true	1 bit unicast response flag. When this bit is set in a question, it indicates that the querier is willing to accept unicast replies in response to this specific query, as well as the usual multicast responses.
cache_flush	**16-bit-field**	**2**	8000	true	Announcements to flush outdated cache entries.

> FIELD TREE

All **DNS field tree structures** are also valid for **MDNS**. The main difference is unicast_response is used instead of qclass and cache_flush is used instead of qclass

> NOTES

- From the dissection point of view, **MDNS** is almost the same as **DNS**. All **DNS** notes belong to **MDNS** as well.

HTTP

> STATUS

Protocol	RFC	Status	Tags

HTTP	rfc1945(HTTP/1.0)
	rfc2616(HTTP/1.1)
	rfc7231(HTTP/1.1)
		Fully	basic, network, internet

> LAYER DETECTION METHODS

- Port-based
- Patterns
- Try dissect

> PORTS

- 80 (tcp)

> PATTERNS

HTTP request methods are used for protocol pattern detection:

- GET
- POST
- HEAD
- PUT
- DELETE
- CONNECT
- OPTIONS
- TRACE
- COPY
- LOCK
- MKCOL
- MOVE
- PROPFIND
- PROPPATCH
- SEARCH
- UNLOCK
- BIND
- REBIND
- UNBIND
- ACL
- REPORT
- MKACTIVITY
- CHECKOUT
- MERGE
- PATCH
- PURGE
- MKCALENDAR
- LINK
- UNLINK
- SOURCE

> FIELDS

Name	Type	Length	Mask	Multiple	Description
root	**uint8**	**1**	ffffffffffffff	false	Layer presented flag.
raw_data	**byte-sequence**	**0**	ffffffffffffff	false	Layer data.
raw_data_length	**uint64**	**8**	ffffffffffffff	false	Layer data length.
payload_data	**byte-sequence**	**0**	ffffffffffffff	false	The payload data - data which is placed right after the layer data.
payload_data_length	**uint64**	**8**	ffffffffffffff	false	The payload data length.
method	**ascii-string**	**0**	ffffffffffffff	false	The method token indicates the method to be performed on the resource identified by the Request-URI.
uri	**ascii-string**	**0**	ffffffffffffff	false	The part of HTTP request line which describes the exact location of a page, post, file, or other asset.
status_code	**ascii-string**	**0**	ffffffffffffff	false	The part of HTTP response status line which is presented as a 3-digit integer number of the attempt to understand and satisfy the request.
reason_phrase	**ascii-string**	**0**	ffffffffffffff	false	The part of HTTP response status line which describes status code.
version	**ascii-string**	**0**	ffffffffffffff	false	The version of an HTTP message.
header	**ascii-string**	**0**	ffffffffffffff	true	An HTTP header consists of its case-insensitive name followed by a colon (:), then by its value. The fields pass additional context and metadata about the request or response.
body	**ascii-string**	**0**	ffffffffffffff	false	HTTP message body.

chunk	**ascii-string**	**0**	ffffffffffffff	true	The part of HTTP body. The field is presented when Transfer-Encoding header has 'chunked' value.
chunk_size	**ascii-string**	**0**	ffffffffffffff	true	The string of hex digits indicating the size of the chunk.
chunk_extension	**ascii-string**	**0**	ffffffffffffff	true	The part of chunk size line. Optional field.
chunk_data	**ascii-string**	**0**	ffffffffffffff	true	The data part of chunk.
trailer	**ascii-string**	**0**	ffffffffffffff	true	The trailer field allows the sender to include additional HTTP header fields at the end of the message.

> LIMITATIONS

- HTTP **1.0** RFC suggests few default request methods. HTTP **1.1** RFC has a bit more default methods. Since each of specification allows to extend HTTP method - dissection process doesn't validate method name is suitable for specific HTTP version.
- HTTP **1.0** has Simple-Response format of response. It doesn't have any HTTP RFC patterns and requires to cache HTTP request. If client sends Simple-Request server must to reply with Simple-Response. Packet library doesn't cache any data and because of that such answers cannot be properly dissected.
- HTTP **1.1** requires to have Host header in requests. Packet library doesn't check that. It dissects just a structure of HTTP message.
- Packet library strongly follows RFC. If RFC describes only 1 **SP** char between tokens - dissection process will expect only 1 **SP**. Not duplicated, not any count of **LWS**.
- If Content-Length has invalid value - the rest of data is dissected as HTTP Body.
- Calling GetNextLayer of HTTP layer doesn't dissect a layer, but it doesn't validate chunk-extension and trailer parts of chunked body. First HTTP line and HTTP header are validating.
- Dissect expects only one space separator in HTTP first line, but GetNextLayer allows any space char count between tokens.
- HTTP Body decode is not supported.
- It is possible to have empty http layer (stub layer where data length is 0 and no dissected fields, e.g. ethernet.ipv4.tcp.http.http2. It might happen, because during the session HTTP/1.* can be switched to HTTP/2 protocol

HTTP/2

> STATUS

Protocol	RFC	Status	Tags
DNS	rfc9113 rfc7541	Fully	basic, network, internet

> LAYER DETECTION METHODS

- Port-based
- Patterns
- Try dissect

> PORTS

- 80 (tcp)

> PATTERNS

HTTP/2 PREFACE string:

- PRI * HTTP/2.0\r\n\r\nSM\r\n\r\n

> FIELDS

Name	Type	Length	Mask	Multiple	Description
root	**uint8**	**1**	ffffffffffffff	false	Layer presented flag.
raw_data	**byte-sequence**	**0**	ffffffffffffff	false	Layer data.
raw_data_length	**uint64**	**8**	ffffffffffffff	false	Layer data length.
payload_data	**byte-sequence**	**0**	ffffffffffffff	false	The payload data - data which is placed right after the layer data.

payload_data_length	**uint64**	**8**	ffffffffffffff	false	The payload data length.
preface	**ascii-string**	**0**	ffffffffffffff	false	The client connection preface.
frame	**byte-sequence**	**0**	ffffffffffffff	false	The frame data.
frame_length	**byte-sequence**	**3**	ffffffffffffff	true	The length of the frame payload expressed as an unsigned 24-bit integer in units of octets. The 9 octets of the frame header are not included in this value.
frame_type	**uint8**	**1**	ffffffffffffff	true	The 8-bit type of the frame. The frame type determines the format and semantics of the frame.
frame_flags	**uint8**	**1**	ffffffffffffff	true	An 8-bit field reserved for boolean flags specific to the frame type. Flags are assigned semantics specific to the indicated frame type. Unused flags are those that have no defined semantics for a particular frame type.
data_unused0_flags	**8-bit-field**	**1**	f0	true	The unused bits.
data_padded_flag	**8-bit-field**	**1**	8	true	When set, the PADDED flag indicates that the Pad Length field and any padding that it describes are present.
data_unused1_flags	**8-bit-field**	**1**	6	true	The unused bits.
data_end_stream_flag	**8-bit-field**	**1**	1	true	When set, the END_STREAM flag indicates that this frame is the last that the end point will send for the identified stream.
headers_unused0_flags	**8-bit-field**	**1**	c0	true	The unused bits.
data_unused1_flags	**8-bit-field**	**1**	6	true	The unused bits.

data_end_stream_flag	**8-bit-field**	**1**	1	true	When set, the END_STREAM flag indicates that this frame is the last that the end point will send for the identified stream.
headers_unused0_flags	**8-bit-field**	**1**	c0	true	The unused bits.
headers_priority_flag	**8-bit-field**	**1**	20	true	When set, the PRIORITY flag indicates that the Exclusive, Stream Dependency, and Weight fields are present.
headers_unused1_flag	**8-bit-field**	**1**	10	true	The unused bits.
headers_padded_flag	**8-bit-field**	**1**	8	true	When set, the PADDED flag indicates that the Pad Length field and any padding that it describes are present.
headers_end_headers_flag	**8-bit-field**	**1**	4	true	When set, the END_HEADERS flag indicates that this frame contains an entire field block and is not followed by any CONTINUATION frames. HEADERS frame without the END_HEADERS flag set MUST be followed by a CONTINUATION frame for the same stream.
headers_unused2_flag	**8-bit-field**	**1**	2	true	The unused bits.
headers_end_stream_flag	**8-bit-field**	**1**	1	true	When set, the END_STREAM flag indicates that the field block is the last that the endpoint will send for the identified stream. A HEADERS frame with the END_STREAM flag set signals the end of a stream. However, a HEADERS frame with the END_STREAM flag set can be followed by CONTINUATION frames on the same stream. Logically, the CONTINUATION frames are part of the HEADERS frame.
settings_unused_flags	**8-bit-field**	**1**	fe	true	The unused bits.

settings_ack_flag	**8-bit-field**	**1**	1	true	The ack flag.
push_promise_unused0_flags	**8-bit-field**	**1**	f0	true	The unused bits.
push_promise_padded_flag	**8-bit-field**	**1**	8	true	An 8-bit field containing the length of the frame padding in units of octets. This field is only present if the PADDED flag is set.
push_promise_end_header_flag	**8-bit-field**	**1**	4	true	When set, the END_HEADERS flag indicates that this frame contains an entire field block and is not followed by any CONTINUATION frames. A PUSH_PROMISE frame without the END_HEADERS flag set MUST be followed by a CONTINUATION frame for the same stream.
push_promise_unused1_flags	**8-bit-field**	**1**	3	true	The unused bits.
ping_unused_flags	**8-bit-field**	**1**	fe	true	The unused bits.
ping_ack_flag	**8-bit-field**	**1**	1	true	When set, the ACK flag indicates that this PING frame is a PING response. An end point MUST set this flag in PING responses. An endpoint MUST NOT respond to PING frames containing this flag.
continuation_unused0_flags	**8-bit-field**	**1**	f8	true	The unused bits.
continuation_end_header_flag	**8-bit-field**	**1**	4	true	When set, the END_HEADERS flag indicates that this frame ends a field block.
continuation_unused1_flags	**8-bit-field**	**1**	3	true	The unused bits.
frame_stream_identifier_data	**uint32**	**4**	ffffffffffffffff	true	The field covers reserved and stream identifier fields.

frame_stream_identifier	**32-bit-field**	**4**	fffffffe	true	A stream identifier expressed as an unsigned 31-bit integer. The value 0x00 is reserved for frames that are associated with the connection as a whole as opposed to an individual stream.
data_pad_length	**uint8**	**1**	ffffffffffffffff	true	An 8-bit field containing the length of the frame padding in units of octets. This field is conditional and is only present if the PADDED flag is set.
data_data	**byte-sequence**	**0**	ffffffffffffffff	true	Application data. The amount of data is the remainder of the frame payload after subtracting the length of the other fields that are present.
data_padding	**byte-sequence**	**0**	ffffffffffffffff	true	Padding octets that contain no application semantic value. Padding octets MUST be set to zero when sending.
headers_pad_length	**uint8**	**1**	ffffffffffffffff	true	An 8-bit field containing the length of the frame padding in units of octets. This field is only present if the PADDED flag is set.
headers_exclusive	**32-bit-field**	**4**	80000000	true	A single-bit flag. This field is only present if the PRIORITY flag is set. Priority signals in HEADERS frames are deprecated.
headers_stream_dependency	**32-bit-field**	**4**	7fffffff	true	A 31-bit stream identifier. This field is only present if the PRIORITY flag is set.
headers_weight	**uint8**	**1**	ffffffffffffffff	true	An unsigned 8-bit integer. This field is only present if the PRIORITY flag is set.
headers_field_block_fragment	**byte-sequence**	**0**	ffffffffffffffff	true	A field block fragment.
headers_padding	**byte-sequence**	**0**	ffffffffffffffff	true	Padding octets that contain no application semantic value. Padding octets MUST be set to zero when sending.

priority_exclusive	**32-bit-field**	**4**	80000000	true	A single-bit flag.
priority_weight	**uint8**	**1**	ffffffffffffffff	true	An unsigned 8-bit integer.
rst_stream_error_code	**uint32**	**4**	ffffffffffffffff	true	The error code indicates why the stream is being terminated.
settings_setting	**byte-sequence**	**0**	ffffffffffffffff	true	The setting data. The field includes id and value fields.
setting_identifier	**uint16**	**2**	ffffffffffffffff	true	A 16-bit setting identifier.
setting_value	**uint32**	**2**	ffffffffffffffff	true	A 32-bit value for the setting.
push_promise_pad_length	**uint8**	**1**	ffffffffffffffff	true	An 8-bit field containing the length of the frame padding in units of octets. This field is only present if the PADDED flag is set.
push_promise_reserved	**32-bit-field**	**4**	80000000	true	The reserved bit.
push_promise_stream_id	**32-bit-field**	**4**	7fffffff	true	An unsigned 31-bit integer that identifies the stream that is reserved by the PUSH_PROMISE. The promised stream identifier MUST be a valid choice for the next stream sent by the sender.
push_promise_field_block_fragment	**byte-sequence**	**0**	ffffffffffffffff	true	A field block fragment containing the request control data and a header section.
push_promise_padding	**byte-sequence**	**0**	ffffffffffffffff	true	Padding octets that contain no application semantic value. Padding octets MUST be set to zero when sending.
ping_opaque_data	**byte-sequence**	**0**	ffffffffffffffff	true	Opaque data. A sender can include any value it chooses and use those octets in any fashion.

go_away_reserved	**32-bit-field**	**4**	80000000	true	The reserved bit.
go_away_last_stream_id	**32-bit-field**	**4**	7fffffff	true	The last stream identifier in the GOAWAY frame contains the highest-numbered stream identifier for which the sender of the GOAWAY frame might have taken some action on or might yet take action on.
go_away_error_code	**uint32**	**4**	ffffffffffffffff	true	A 32-bit error code that contains the reason for closing the connection.
go_away_additional_debug_data	**byte-sequence**	**0**	ffffffffffffffff	true	Additional debug data is intended for diagnostic purposes only and carries no semantic value. Debug information could contain security- or privacy-sensitive data. Logged or otherwise persistently stored debug data MUST have adequate safeguards to prevent unauthorized access.
window_update_reserved	**32-bit-field**	**4**	ffffffffffffffff	true	The reserved bit.
window_update_window_size_increment	**32-bit-field**	**4**	ffffffffffffffff	true	The window size increment.
continuation_field_block_fragment	**byte-sequence**	**0**	ffffffffffffffff	true	A field block fragment.

> FIELD TREE

- **Frame**

```
.
└── frame/
    ├── frame_length
    ├── frame_type
    ├── frame_flags/
    ├── frame_stream_identifier_data/
    ├── frame_payload/
```

- **Data frame flags**

```
.
└── frame/
    ├── ...
    ├── frame_flags/
    |   ├── data_unused0_flags
    |   ├── data_padded_flag
    |   ├── data_unused1_flags
    |   ├── data_end_stream_flag
    ├── ...
```

- **Headers frame flags**

```
.
└── rame/
    ├── ...
    ├── frame_flags/
    |   ├── headers_unused0_flags
    |   ├── headers_priority_flag
    |   ├── headers_unused1_flag
    |   ├── headers_padded_flag
    |   ├── headers_end_headers_flag
    |   ├── headers_unused2_flag
    |   └── headers_end_stream_flag
    └── ...
```

- **Settings frame flags**

```
.
└── frame/
    ├── ...
    ├── frame_flags/
    |   ├── settings_unused_flags
    |   └── settings_ack_flag
    ├── ...
```

- **PushPromise frame flags**

```
.
└── frame/
    ├── ...
    ├── frame_flags/
    |   ├── push_promise_unused0_flags
    |   ├── push_promise_padded_flag
    |   ├── push_promise_end_header_flag
    |   └── push_promise_unused1_flags
    └── ...
```

- **Ping frame flags**

```
.
└── frame/
    ├── ...
    ├── frame_flags/
    |   ├── ping_unused_flags
    |   └── ping_ack_flag
    └── ...
```

- **Continuation frame flags**

```
.
└── frame/
    ├── ...
    ├── frame_flags/
    |   ├── continuation_unused0_flags
    |   ├── continuation_end_header_flag
    |   └── continuation_unused1_flags
    └── ...
```

- **Frame Stream Identifier Data**

```
.
└── frame/
    ├── ...
    ├── frame_stream_identifier_data/
    |   ├── frame_reserved_bit
    |   └── frame_stream_identifier
    └── ...
```

- **Frame Payload**

```
.
└── frame/
    ├── ...
    └── frame_payload/
        ├── data_pad_length
        ├── data_data
        ├── data_padding
        ├── headers_pad_length
        ├── headers_exclusive
        ├── headers_stream_dependency
        ├── headers_weight
        ├── headers_field_block_fragment
        ├── headers_padding
        ├── priority_exclusive
        ├── priority_stream_dependency
        ├── priority_weight
        ├── rst_stream_error_code
        ├── settings_setting
        ├── push_promise_pad_length
        ├── push_promise_reserved
        ├── push_promise_stream_id
        ├── push_promise_field_block_fragment
        ├── push_promise_padding
        ├── ping_opaque_data
        ├── go_away_reserved
        ├── go_away_last_stream_id
        ├── go_away_error_code
        ├── go_away_additional_debug_data
        ├── window_update_reserved
        ├── window_update_window_size_increment
        └── continuation_field_block_fragment
```

- **Setting Frame**

```
.
└── frame/
    ├── ...
    └── frame_payload/
        ├── settings_setting
        │   ├── setting_identifier
        │   └── setting_value
        └── ...
```

SSDP

> STATUS

Protocol	RFC	Status	Tags
Ssdp	draft-cai-ssdp-v1-02 draft-cai-ssdp-v1-03	Fully	basic, network, internet

> LAYER DETECTION METHODS

- Port-based
- Patterns
- Layer structure test

> PORTS

- 1900 (udp)

> PATTERNS

SSDP request methods are used for protocol pattern detection:

- M-SEARCH
- NOTIFY
- SUBSCRIBE
- SSDPC

> FIELDS

All **HTTP fields** are valid for SSDP as well.

Dropbox

> STATUS

Protocol	RFC	Status	Tags
Dropbox Lan Sync Dropbox Lan Sync Discovery	Absent	Fully	basic, network, internet

> LAYER DETECTION METHODS

- Port-based

> PORTS

- 17500 (udp/tcp)

> FIELDS

Dropbox Lan Sync Discovery

Name	Type	Length	Mask	Multiple	Description
root	**uint8**	**1**	ffffffffffffffff	false	Layer presented flag.
raw_data	**byte-sequence**	**0**	ffffffffffffffff	false	Layer data.

raw_data_length	**uint64**	**8**	ffffffffffffff	false	Layer data length.
payload_data	**byte-sequence**	**0**	ffffffffffffff	false	The payload data - data which is placed right after the layer data.
payload_data_length	**uint64**	**8**	ffffffffffffff	false	The payload data length.
data	**byte-sequence**	**0**	ffffffffffffff	false	Data payload (it should have json format with 'host_int', 'version', 'displayname', 'port', 'namespaces' fields).

Dropbox Lan Sync

Name	Type	Length	Mask	Multiple	Description
root	**uint8**	**1**	ffffffffffffff	false	Layer presented flag.
raw_data	**byte-sequence**	**0**	ffffffffffffff	false	Layer data.
raw_data_length	**uint64**	**8**	ffffffffffffff	false	Layer data length.
payload_data	**byte-sequence**	**0**	ffffffffffffff	false	The payload data - data which is placed right after the layer data.
payload_data_length	**uint64**	**8**	ffffffffffffff	false	The payload data length.
type	**uint8**	**1**	ffffffffffffff	false	The type field. Configuration (0x16), Data (0x17).
magic	**uint16**	**2**	ffffffffffffff	false	Magic number. Usually has 0x0301 value.
data_length	**uint16**	**2**	ffffffffffffff	false	Layer data length.
data	**byte-sequence**	**0**	ffffffffffffff	false	The payload data.

slinkin.tech