**3GPP TSG-SA5 Meeting #158 *S5-247076***

Orlando, USA, 18 – 22 November 2024

**Source: Ericsson, Vodafone, Deutsche Telekom, Telecom Italia, Rakuten**

**Title: Signalling traffic monitoring Report Format of the drafted TS28.abc**

**Document for: Approval**

**Agenda Item: 6.19.22**

# 1 Decision/action requested

***For approval.***

# 2 References

[1] S5-245981 new WID signalling monitoring

[2] S5-245336, initial skeleton of draft TS28.abc signalling monitoring

# 3 Rationale

Defines Signalling traffic monitoring management report format

# 4 Detailed proposal

\*\*\* START OF NEXT CHANGE \*\*\*

# 2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non‑specific.

- For a specific reference, subsequent revisions do not apply.

- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document*.

[1] 3GPP TR 21.905: "Vocabulary for 3GPP Specifications".

[x7] IETF RFC8086: GRE-in-UDP Encapsulation

[x8] IETF draft-ietf-opsawg-pcapng-04: PCAP Next Generation (pcapng) Capture File Format

[x9] Extensible Encapsulation Protocol, https://sipcapture.org

\*\*\* START OF NEXT CHANGE \*\*\*

## 7.1 Reporting format

### 7.1.1 Protocol

The STM report shall be send as payload of either UDP, TCP, or SCTP based on operator’s policy.

### 7.1.2 Format of the STM reports

The STM report may be formated based on operator policy by UDP – GRE – PCAPng encapsulation or by the Extensible Encapsulation Protocol.

#### 7.1.2.1 UDP-GRE-PCAPng Encapsulation

Encoding of STM Payloads shall be performed using GRE in UDP Encapsulation as specified in IETF RFC8086 [x7]. The packet is encapsulated as IP header, UDP header, GRE header, and STM Payload.

The encapsulated STM Payload is shown in Figure 7.1.2-1.



Figure 7.1.2-1 STM encapsulation

|  |
| --- |
|  |
|  |
|  |  |  |

The STM payload contains the PCAPNG payload (IETF draft-ietf-opsawg-pcapng-04 [x8]). The PCAPNG payload contains PCAPNG header and the collected signalling messages. The STM payload has following attributes, as specified in the Table 7.1.2-2.

Table 7.1.2.1-2: STM Payload

|  |  |
| --- | --- |
| STM Payload attribute name | Description |
|  |  |
| PCAPNG Payload (M) | PCAPNG header and the collected signalling messages.The collected signalling message is sent before security encapsulation, or received after security decapsulation. |

Editor’s note: the identification of the peer NFs of the message is FFS.

#### 7.1.2.2 General Type – Length- Value encoding

Depending on operator policy the STM payload shall be transferred by UDP/IP, TCP/IP, or SCTP/IP.

The STM payload is formatted as general Type - Lenght - Value (TLV) encoding.

Each message starts with four bytes protocol ID, followed by two bytes to indicate the number of bytes of the whole message (including protocol ID and length). The rest of the message consists of a non-empty list of data chunks.

Each data chunk is composed by a chunk type, the total lenghts of the chunk, and the payload of the chunk.

\*\*\* START OF NEXT CHANGE \*\*\*

# Annex A (informative) Sets of types for TLV encoding

## A.1 Extensible Encapsulation Protocol

The Extensible Encapsulation Protocol [x9] defines a a set of types to be used for type - length – encoding of captured signalling messages used in telecommunication systems.

### A.1.1 Type to define the protocol messages:

Protocol Id = EEP

### A.1.2 Chunk types

The chunk type is composed by a two byte vendor ID and a two byte generic chunk type. Table A.1.2-1 lists the currently defined generi chunk types. In future additional generic chunk types may be defined.

Table A.1.2-1: Generic chunk types

|  |  |  |
| --- | --- | --- |
| **Generic Chunk Type** |  **Payload Type** | **Description** |
| 0x0001 |  uint8 |  IP protocol family |
| 0x0002 |  uint8 |  IP protocol ID |
| 0x0003 |  inet4-addr |  IPv4 source address |
| 0x0004 |  inet4-addr |  IPv4 destination address |
| 0x0005 |  inet6-addr |  IPv6 source address |
| 0x0006 |  inet6-addr |  IPv6 destination address |
| 0x0007 |  uint16 |  protocol source port (UDP, TCP, SCTP) |
| 0x0008 |  uint16 |  protocol destination port (UDP, TCP, SCTP) |
| 0x0009 |  uint32 |  timestamp, seconds since 01/01/1970 (epoch) |
| 0x000a |  uint32 |  timestamp microseconds offset (added to timestamp) |
| 0x000b |  uint8 |  protocol type (SIP/H323/RTP/MGCP/M2UA) |
| 0x000c |  uint32 |  capture agent ID (202, 1201, 2033...) |
| 0x000d |  uint16 |  keep alive timer (sec) |
| 0x000e |  octet-string |  authenticate key (plain text / TLS connection) |
| 0x000f |  octet-string |  captured packet payload |
| 0x0010 |  octet-string |  captured compressed payload (gzip/inflate) |
| 0x0011 |  octet-string |  Internal correlation id |
| 0x0012 |  uint16 |  Vlan ID |
| 0x0013 |  octet-string |  capture agent ID (“node1”, “node2”, “node3”...) |
| 0x0014 |  uint64 |  Source MAC |
| 0x0015 |  uint64 |  Destination MAC |
| 0x0016 |  uint16 |  Ethernet Type |
| 0x0017 |  uint8 |  TCP Flag [SYN.PUSH...] |
| 0x0018 |  uint8 |  IP TOS |
| 0x001F |  Reserved |   |
| 0x0020 |  uint16 |  MOS value |
| 0x0021 |  uint16 |  R-Factor |
| 0x0022 |  octet-string |  GEO Location |
| 0x0023 |  uint32 |  Jitter |
| 0x0024 |  octet-string |  Transaction type [call, registration] |
| 0x0025 |  octet-string |  Payload JSON Keys |
| 0x0026 |  octet-string |  Tags’ values |
| 0x0027 |  uint16 |  Type of tag |
| 0x0028 |  uint16 |  Event type [recording|interception| |
| 0x0029 |  octet-string |  Group ID |

### A.1.2 Captured protocol types

Following the generic chunk type 0x000b "protocol type" the payload contains parts of the captured signalling message.

Table A.1.2-2 defines the types of the protocol for which the signalling message was captured in the payload of the chunk. In future additional protocol types may be added.

Table A.1.2-2: Captured protocol types

|  |  |
| --- | --- |
| **Chunk Protocol ID** |  **Captured Protocol** |
| 0x00 |  Reserved |
| 0x01 |  SIP |
| 0x02 |  XMPP |
| 0x03 |  SDP |
| 0x04 |  RTP |
| 0x05 |  RTCP JSON |
| 0x06 |  MGCP |
| 0x07 |  MEGACO (H.248) |
| 0x08 |  M2UA (SS7/SIGTRAN) |
| 0x09 |  M3UA (SS7/SIGTRAN) |
| 0x0a |  IAX |
| 0x0b |  H3222 |
| 0x0c |  H321 |
| 0x0d |  M2PA |
| 0x22 |  MOS full report [JSON] |
| 0x23 |  MOS short report. Please use mos chunk 0x20 [JSON] |
| 0x32 |  SIP JSON |
| 0x33 |  RESERVED |
| 0x34 |  RESERVED |
| 0x35 |  DNS JSON |
| 0x36 |  M3UA JSON (ISUP) |
| 0x37 |  RTSP (JSON) |
| 0x38 |  DIAMETER (JSON) |
| 0x39 |  GSM MAP (JSON) |
| 0x3a |  RTCP PION |
| 0x3b |  RESERVED |
| 0x3c |  CDR (can be for call and registration transaction) |

\*\*\* START OF NEXT CHANGE \*\*\*

# Annex B (informative):Plant UML source code

## B.4 STM encapsulation

The following PlantUML source code is used to describe STM encapsulation. As depicted by Figure 7.1.2.-1:

@startuml

frame "STM encapsulation" {

rectangle "GRE header" as GRE

rectangle "STM Payload" as Payload

rectangle "UDP header" as UDP

rectangle "IP header" as IP

IP-[hidden]>UDP

UDP-[hidden]>GRE

GRE-[hidden]>Payload

}

@enduml

\*\*\* END OF CHANGE \*\*\*