**3GPP TSG-SA5 Meeting #129-e *S5-201418rev1***

**Online, , 24th Feb 2020 - 4th Mar 2020**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *CR-Form-v12.0* | | | | | | | | |
| **CHANGE REQUEST** | | | | | | | | |
|  | | | | | | | | |
|  | **32.423** | **CR** | **0101** | **rev** | **-** | **Current version:** | **15.1.0** |  |
|  | | | | | | | | |
| *For* [***HE******LP***](http://www.3gpp.org/3G_Specs/CRs.htm#_blank)*on using this form: comprehensive instructions can be found at* [*http://www.3gpp.org/Change-Requests*](http://www.3gpp.org/Change-Requests)*.* | | | | | | | | |
|  | | | | | | | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***Proposed change affects:*** | UICC apps |  | ME |  | Radio Access Network | **X** | Core Network | **X** |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | | | |
| ***Title:*** | Add streaming format for Trace Record Reporting | | | | | | | | | |
|  |  | | | | | | | | | |
| ***Source to WG:*** | Oy LM Ericsson AB | | | | | | | | | |
| ***Source to TSG:*** | S5 | | | | | | | | | |
|  |  | | | | | | | | | |
| ***Work item code:*** | OAM\_RTT | | | | |  | ***Date:*** | | | 2020-02-24 |
|  |  | | | |  | |  | | |  |
| ***Category:*** | B |  | | | | | ***Release:*** | | | Rel-16 |
|  | *Use one of the following categories:* ***F*** *(correction)* ***A*** *(mirror corresponding to a change in an earlier release)* ***B*** *(addition of feature),* ***C*** *(functional modification of feature)* ***D*** *(editorial modification)*  Detailed explanations of the above categories can be found in 3GPP [TR 21.900](http://www.3gpp.org/ftp/Specs/html-info/21900.htm). | | | | | | | | *Use one of the following releases: Rel-8 (Release 8) Rel-9 (Release 9) Rel-10 (Release 10) Rel-11 (Release 11) Rel-12 (Release 12)* *Rel-13 (Release 13) Rel-14 (Release 14) Rel-15 (Release 15) Rel-16 (Release 16)* | |
|  |  | | | | | | | | | |
| ***Reason for change:*** | | 3GPP lacks a standardized streaming format for Trace | | | | | | | | |
|  | |  | | | | | | | | |
| ***Summary of change:*** | | A stage 3 specification of a Trace Record format is added, along with normative examples. | | | | | | | | |
|  | |  | | | | | | | | |
| ***Consequences if not approved:*** | | Streamed Trace Record format will remain vendor specific. | | | | | | | | |
|  | |  | | | | | | | | |
| ***Clauses affected:*** | | X | | | | | | | | |
|  | |  | | | | | | | | |
|  | | **Y** | **N** |  | | | |  | | |
| ***Other specs*** | |  |  | Other core specifications | | | | TS/TR ... CR ... | | |
| ***affected:*** | |  |  | Test specifications | | | | TS/TR ... CR ... | | |
| ***(show related CRs)*** | |  |  | O&M Specifications | | | | TS/TR ... CR ... | | |
|  | |  | | | | | | | | |
| ***Other comments:*** | |  | | | | | | | | |
|  | |  | | | | | | | | |
| ***This CR's revision history:*** | |  | | | | | | | | |

***First change***

# X Trace streaming format

### X.1 Introduction

This clause defines the format of streamed trace records and reports.

### X.2 Definition of the Trace Record

X.2.1 Introduction

The Trace Record comprises a header and payload as shown in Figure X.2.1.1.

Trace Record

Payload

Header

Vendor specific  
extension

Common  
fields

Vendor Specified  
Content

Size

Figure X.2.1.1: Trace Record

The Header contains common fields plus an optional vendor-specific extension as defined in clause X.1.1.

The Payload contains an optional size field plus vendor-specific content as defined in clause X.1.2.

X.2.2 Definition of the Trace Record Header

The trace record header contains the fields defined in Table X.2.2.1

Table X.2.2.1 : Fields in the trace record header

|  |  |
| --- | --- |
| Trace Record Header field name | Description |
| timeStamp (M) | Time stamp (milliseconds since Epoch) of when the trace record is produced internally in the Producer (64 bit integer) |
| nfInstanceId (M) | Id of the Producer NF instance (String) |
| nfType (M) | Type of the Producer NF (String) |
| traceReference (O) | Trace Reference (String) |
| traceRecordingSessionReference (O) | Trace Recording Session Reference (String) |
| traceRecordTypeId (M) | Identifier of the trace record type (64 bit integer) |
| ranUeId(O) | RAN defined Id of the UE (8 byte octet string) |
| payloadSchemaURI (M) | URI identifying the schema to decode the payload (String) |
| vendorExtension (O) | Vendor-specific extension (Array of String) |

The *traceReference* and *traceRecordingSessionReference* are mandatory for signalling based activation.

The *ranUeId* field is mandatory in UE-related trace records. At present *ranUeId* is only defined for F1 (38.473) and E1 (38.463).

The *vendorExtension* is defined as a generic name-value pair list.

X.2.3 Definition of the trace record payload

The trace record payload comprises the fields defined in Table X.2.3.1.

Table X.2.3.1 : Fields in the trace record payload

|  |  |
| --- | --- |
| Trace Record Payload parameter name | Description |
| payloadSize (O) | Size of payload, in bytes (64 bit integer) |
| payload (M) | Array of bytes |

The *payloadSize* indicates the payload size. It may be omitted if the solution set specific encoding/decoding has its own support for indicating the size.

The *payload* is a sequence of bytes representing the binary encoded data of the specific trace record:

* For example, record content per TS 32.423 “Trace Record Content for  gNB-CU-CP, gNB-CU-UP, gNB-DU (and ng-eNB and  E-UTRAN)” with schema indicated in header field *payloadSchemaURI* required for decoding.

X.2.4 Trace Administrative Record Definitions

#### X.2.4.1 Introduction

The following Trace Record messages are defined to for trace stream management purposes.

These administrative messages use specific enum values in the Trace Record header for the ‘traceRecordTypeId’ field as follows:

0: Trace Session Start  
1: Trace Session Stop  
2: Trace Stream Heartbeat

Such Trace Records may also contain vendor specific header extensions or payload with additional data.

#### X.2.4.2 Trace Session start trace record

The following Trace Record is sent to trace consumer at the start of a streaming trace session.

Table X.2.4.2.1 : Fields in the trace record header

|  |  |
| --- | --- |
| Trace Record Header field name | Description |
| timeStamp (M) | Time stamp (milliseconds since Epoch) of when the trace record is produced internally in the Producer (64 bit integer) |
| nfInstanceId (M) | Id of the Producer NF instance (String) |
| nfType (M) | Type of the Producer NF (String) |
| traceReference (O) | Trace Reference (String) |
| traceRecordingSessionReference (O) | Trace Recording Session Reference (String) |
| traceRecordTypeId (M) | **0 (traceSessionStart)** |
| ranUeId(O) | RAN defined Id of the UE (8 byte octet string) |
| payloadSchemaURI (M) | URI identifying the schema to decode the payload (String) |
| vendorExtension (O) | Vendor-specific extension (Array of String) |

The *traceReference* and *traceRecordingSessionReference* are optional for admin records, including signalling-based.

#### X.2.4.3 Trace Session stop trace record

The following Trace Record is sent to trace consumer at the end of a streaming trace session.

Table X.2.4.3.1 : Fields in the trace record header

|  |  |
| --- | --- |
| Trace Record Header field name | Description |
| timeStamp (M) | Time stamp (milliseconds since Epoch) of when the trace record is produced internally in the Producer (64 bit integer) |
| nfInstanceId (M) | Id of the Producer NF instance (String) |
| nfType (M) | Type of the Producer NF (String) |
| traceReference (O) | Trace Reference (String) |
| traceRecordingSessionReference (O) | Trace Recording Session Reference (String) |
| traceRecordTypeId (M) | **1 (traceSessionStop)** |
| ranUeId(O) | RAN defined Id of the UE (8 byte octet string) |
| payloadSchemaURI (M) | URI identifying the schema to decode the payload (String) |
| vendorExtension (O) | Vendor-specific extension (Array of String) |

The *traceReference* and *traceRecordingSessionReference* are optional for admin records, including signalling-based.

#### X.2.4.4 Trace Stream Heartbeat trace record

The following Trace Record is sent periodically to trace consumer.

|  |  |
| --- | --- |
| Trace Record Header field name | Description |
| timeStamp (M) | Time stamp (milliseconds since Epoch) of when the trace record is produced internally in the Producer (64 bit integer) |
| nfInstanceId (M) | Id of the Producer NF instance (String) |
| nfType (M) | Type of the Producer NF (String) |
| traceReference (O) | Trace Reference (String) |
| traceRecordingSessionReference (O) | Trace Recording Session Reference (String) |
| traceRecordTypeId (M) | **2 (traceStreamHeartbeat)** |
| ranUeId(O) | RAN defined Id of the UE (8 byte octet string) |
| payloadSchemaURI (M) | URI identifying the schema to decode the payload (String) |
| vendorExtension (O) | Vendor-specific extension (Array of String) |

The message is intended to indicate that a trace stream connection is alive and is sent regardless of whether there is an ongoing Trace Session or not.

An interval of at least 60 secs is recommended.

### X.2.5 Trace session management

The configuration and management of trace data, including activation and configuration of equipment and subscriber trace is defined in 3GPP TS 32.422.

### X.3 Trace Record stream transport

### X.3.1 Introduction

Trace Records are carried in stream transport protocol-specific messages comprising a header and payload as defined in Figure X.3.1.1.

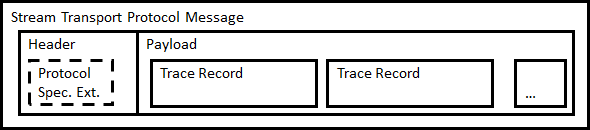


Figure X.3.1.1: Transport of Trace Records

Each trace stream protocol-specific message delivers one or more trace records from the Producer to the Consumer.

The *Header* is transport protocol specific. It may contain protocol specific extensions or options related to the stream.

The *Payload* is Trace Records as specified in sec X.1, with implementation specific encoding.

### X.3.2 Handling of trace stream

Functions such as message routing or filtering performed by a centralized stream collection mechanism, or configuration of client/server extensions would be done using the protocol-specific mechanism.

For streaming trace data, the following is requested to be included in the appropriate transport mechanism:

* Indicator of whether the stream is compressed

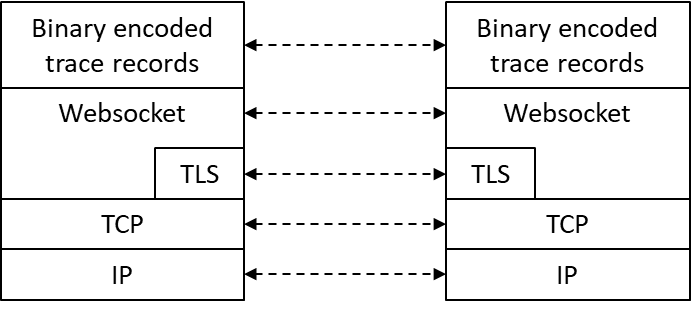
A message can be split across multiple frames if supported by the protocol.

See Annex <x> for an example implementation.

***Next change***

Annex <x1> (Informative):  
Example of protocol stack

Example stream-based Trace Report message (Figure X.3) realized via “The Websocket Protocol” (RFC4255):



Stream metadata defined in X.3.2 via “Compression Extensions for Websocket” (RFC7692).

***Next change***

Annex <x2> (Normative):  
Example Protocol Buffer (GPB) definitions

Example 1: Trace Record schema, defined per clause 1.1 in this specification:

syntax = “proto3”;

/\* Trace Record per 3GPP 32.423 specification.

\* v16

\*/

Message TraceRecordHeader {

required int64 time\_stamp = 1;

required string nf\_instance\_id = 2;

required string nf\_type = 3;

optional string trace\_ref = 4;

required int64 trace\_rec\_type = 5;

optional string trace\_rec\_session\_ref = 6;

optional int64 ran\_ue\_id = 7;

required string payload\_schema\_uri = 8;

optional vendor\_extension = 9;

}

Message TraceRecordPayload {

optional int64 size = 1;

required bytes payload = 2;

}

message TraceRecord {

required TraceRecordHeader header = 1;

required TraceRecordPayload payload = 2;

}

**Example 2: Trace Record payload, 3GPP-based vendor-specific schema:**

syntax="proto3";

/\*\*

\* Description: Event for monitoring RRC Setup procedure.

\* v16

\*/

message CuCpProcRrcEstab {

enum EstablishmentCause{

ESTABLISHMENT\_CAUSE\_EMERGENCY = 1;

ESTABLISHMENT\_CAUSE\_HIGH\_PRIORITY\_ACCESS = 2;

ESTABLISHMENT\_CAUSE\_MT\_ACCESS = 3;

ESTABLISHMENT\_CAUSE\_MO\_SIGNALLING = 4;

ESTABLISHMENT\_CAUSE\_MO\_DATA = 5;

ESTABLISHMENT\_CAUSE\_MO\_VOICE\_CALL = 6;

}

enum RrcConnEstabResult{

RRC\_CONN\_ESTAB\_RESULT\_SUCCESS = 1;

RRC\_CONN\_ESTAB\_RESULT\_UNSPECIFIED = 2;

RRC\_CONN\_ESTAB\_RESULT\_NO\_RADIO\_RESOURCES\_AVAILABLE = 3;

RRC\_CONN\_ESTAB\_RESULT\_FAILURE\_IN\_RADIO\_PROCEDURE = 4;

}

enum PmBooleanType{

PM\_BOOLEAN\_FALSE = 1;

PM\_BOOLEAN\_TRUE = 2;

}

bytes main\_plmn\_id = 1;

int64 gnb\_id = 2;

int64 time\_stamp\_start = 3;

int64 time\_stamp\_stop = 4;

EstablishmentCause establishment\_cause = 5;

bool reattempt\_indicator = 6;

RrcConnEstabResult rrc\_conn\_estab\_result = 7;

bytes ue\_trace\_id = 8;

int64 nci = 9;

int64 gnb\_id\_length = 10;

PmBooleanType reattempt\_indicator\_param = 11;

}

**Example 3: Trace Record payload, 3GPP-based vendor-specific schema, including ASN.1 message:**

syntax="proto3";

/\*\*

\* Description: Event for monitoring EN-DC 3GPP X2-AP SGNB ADDITION REQUEST message.

\*

\* Version: 16

\*/

message CuCpX2SgnbAdditionRequest {

enum SgnbAdditionTriggerIndication{

SGNB\_ADDITION\_TRIGGER\_INDICATION\_ENDC\_SETUP = 1;

SGNB\_ADDITION\_TRIGGER\_INDICATION\_SN\_CHANGE = 2;

SGNB\_ADDITION\_TRIGGER\_INDICATION\_ENB\_HO = 3;

SGNB\_ADDITION\_TRIGGER\_INDICATION\_OTHER\_VALUE = 4;

}

enum MsgDirection{

MSG\_DIRECTION\_SEND = 1;

MSG\_DIRECTION\_RECEIVE = 2;

}

enum ProtocolName{

PROTOCOL\_NAME\_RRC\_36\_331\_BCCH\_BCH\_MESSAGE = 1;

PROTOCOL\_NAME\_RRC\_36\_331\_BCCH\_DL\_SCH\_MESSAGE = 2;

PROTOCOL\_NAME\_RRC\_36\_331\_DL\_CCCH\_MESSAGE = 3;

PROTOCOL\_NAME\_RRC\_36\_331\_DL\_DCCH\_MESSAGE = 4;

PROTOCOL\_NAME\_RRC\_36\_331\_PCCH\_MESSAGE = 5;

PROTOCOL\_NAME\_RRC\_36\_331\_UL\_CCCH\_MESSAGE = 6;

PROTOCOL\_NAME\_RRC\_36\_331\_UL\_DCCH\_MESSAGE = 7;

PROTOCOL\_NAME\_RRC\_38\_331\_BCCH\_BCH\_MESSAGE = 8;

PROTOCOL\_NAME\_RRC\_38\_331\_BCCH\_DL\_SCH\_MESSAGE = 9;

PROTOCOL\_NAME\_RRC\_38\_331\_DL\_CCCH\_MESSAGE = 10;

PROTOCOL\_NAME\_RRC\_38\_331\_DL\_DCCH\_MESSAGE = 11;

PROTOCOL\_NAME\_RRC\_38\_331\_PCCH\_MESSAGE = 12;

PROTOCOL\_NAME\_RRC\_38\_331\_UL\_CCCH\_MESSAGE = 13;

PROTOCOL\_NAME\_RRC\_38\_331\_UL\_DCCH\_MESSAGE = 14;

PROTOCOL\_NAME\_S1AP\_36\_413 = 15;

PROTOCOL\_NAME\_X2AP\_36\_423 = 16;

PROTOCOL\_NAME\_NGAP\_38\_413 = 17;

PROTOCOL\_NAME\_XNAP\_38\_423 = 18;

}

message Asn1Message {

MsgDirection msg\_direction = 1;

bytes msg\_content = 2;

repeated ProtocolName protocol\_name = 3;

}

bytes main\_plmn\_id = 1;

int64 gnb\_id = 2;

int64 nci = 3;

SgnbAdditionTriggerIndication sgnb\_addition\_trigger\_indication = 4;

Asn1Message asn1 = 5;

int64 gnb\_id\_length = 6;

}

***Next change***

Annex <x3> (Informative):  
Example Protocol Buffer (GPB) stream admin messages

The follow examples show trace stream administrative messages as defined in sec X1.2.

The examples are in compact GPB format, using the schema defined in Annex X.1.

**Example, Trace stream start messages:**

Example non-signalling based:

1: 1581525683

2: 100

3: 1

5: 0

8: urn:3gpp:ns:tracestream:start:1.0

Example signalling based:

1: 1581525683

2: 100

3: 1

4: 1000

5: 0

6: 0001

8: urn:3gpp:ns:tracestream:start:1.0

**Example 2, Trace stream stop messages:**

Example non-signalling based:

1: 1581525684

2: 100

3: 1

5: 1

8: urn:3gpp:ns:tracestream:stop:1.0

Example signalling based:

1: 1581525684

2: 100

3: 1

4: 1000

5: 1

6: 0001

8: urn:3gpp:ns:tracestream:stop:1.0

**Example 3, Trace stream heartbeat message:**

1: 1581525685

2: 100

3: 1

5: 2

8: urn:3gpp:ns:tracestream:heartbeat:1.0

***End of changes***