**3GPP TSG-SA3 Meeting #78-LI-e-b *s3i200416***

**Online, 28th Jul 2020 - 29th Jul 2020**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *CR-Form-v12.0* | | | | | | | | |
| **CHANGE REQUEST** | | | | | | | | |
|  | | | | | | | | |
|  | **33.128** | **CR** | **0094** | **rev** | **3** | **Current version:** | **16.3.0** |  |
|  | | | | | | | | |
| *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 |  | Core Network | **X** |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | | | |
| ***Title:*** | Support for PTC Stage 3 | | | | | | | | | |
|  |  | | | | | | | | | |
| ***Source to WG:*** | SA3-LI (OTD) | | | | | | | | | |
| ***Source to TSG:*** | SA3 | | | | | | | | | |
|  |  | | | | | | | | | |
| ***Work item code:*** | LI16 | | | | |  | ***Date:*** | | | 2020-07-23 |
|  |  | | | |  | |  | | |  |
| ***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:*** | | TS 33.128 does not provide an LI capability for LI of Push to Talk over Cellular (PTC) type services. Such an LI capability is essential for carriers to meet their LI obligations. | | | | | | | | |
|  | |  | | | | | | | | |
| ***Summary of change:*** | | Addition of stage three PTC LI events and messages for LI. | | | | | | | | |
|  | |  | | | | | | | | |
| ***Consequences if not approved:*** | | No method for the CSP to deliver LI of PTC services. The CSP cannot meet their obligatory requirements for LI for PTC type services. | | | | | | | | |
|  | |  | | | | | | | | |
| ***Clauses affected:*** | | 2, new clause 7.x, Annex A, Annex C | | | | | | | | |
|  | |  | | | | | | | | |
|  | | **Y** | **N** |  | | | |  | | |
| ***Other specs*** | |  | **x** | Other core specifications | | | | TS/TR ... CR ... | | |
| ***affected:*** | |  | **x** | Test specifications | | | | TS/TR ... CR ... | | |
| ***(show related CRs)*** | |  | **x** | O&M Specifications | | | | TS/TR ... CR ... | | |
|  | |  | | | | | | | | |
| ***Other comments:*** | | Dependency on CR 0088 (s3i200414) | | | | | | | | |
|  | |  | | | | | | | | |
| ***This CR's revision history:*** | | s3i200413; s3i200352; s3i200332 | | | | | | | | |

**\*\*\* Start of First MODIFICATION \*\*\***

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".

[2] 3GPP TS 23.501: "System Architecture for the 5G System".

[3] 3GPP TS 33.126: "Lawful Interception Requirements".

[4] 3GPP TS 23.502: "Procedures for the 5G System; Stage 2".

[5] 3GPP TS 33.127: "Lawful Interception (LI) Architecture and Functions".

[6] ETSI TS 103 120: " Lawful Interception (LI); Interface for warrant information".

[7] ETSI TS 103 221-1: "Lawful Interception (LI); Internal Network Interfaces; Part 1: X1".

[8] ETSI TS 103 221-2: "Lawful Interception (LI); Internal Network Interfaces; Part 2: X2/X3".

[9] ETSI TS 102 232-1: "Lawful Interception (LI); Handover Interface and Service-Specific Details (SSD) for IP delivery; Part 1: Handover specification for IP delivery".

[10] ETSI TS 102 232-7: "Lawful Interception (LI); Handover Interface and Service-Specific Details (SSD) for IP delivery; Part 7: Service-specific details for Mobile Services".

[11] 3GPP TS 33.501: "Security Architecture and Procedures for the 5G System".

[12] 3GPP TS 33.108: "3G security; Handover interface for Lawful Interception (LI)".

[13] 3GPP TS 24.501: "Non-Access-Stratum (NAS) protocol for 5G System (5GS)".

[14] 3GPP TS 24.007: "Mobile radio interface signalling layer 3; General Aspects".

[15] 3GPP TS 29.244: "Interface between the Control Plane and the User Plane nodes".

[16] 3GPP TS 29.502: "5G System; Session Management Services; Stage 3".

[17] 3GPP TS 29.571: "5G System; Common Data Types for Service Based Interfaces; Stage 3".

[18] 3GPP TS 23.040: "Technical realization of the Short Message Service (SMS)".

[19] 3GPP TS 23.003: "Numbering, addressing and identification ".

[20] OMA-TS-MLP-V3\_5-20181211-C: "Open Mobile Alliance; Mobile Location Protocol, Candidate Version 3.5", <https://www.openmobilealliance.org/release/MLS/V1_4-20181211-C/OMA-TS-MLP-V3_5-20181211-C.pdf>.

[21] 3GPP TS 29.540: "5G System; SMS Services; Stage 3".

[22] 3GPP TS 29.518: "5G System; Access and Mobility Management Services; Stage 3".

[23] 3GPP TS 38.413: "NG Application Protocol (NGAP)".

[24] 3GPP TS 29.572: "Location Management Services; Stage 3".

[25] 3GPP TS 29.503: "5G System; Unified Data Management Services".

[26] IETF RFC 815: "IP DATAGRAM REASSEMBLY ALGORITHMS".

[27] IETF RFC 2460: "Internet Protocol, Version 6 (IPv6) Specification".

[28] IETF RFC 793: "TRANSMISSION CONTROL PROTOCOL".

[29] IETF RFC 768: "User Datagram Protocol".

[30] IETF RFC 4340: "Datagram Congestion Control Protocol (DCCP)".

[31] IETF RFC 4960: "Stream Control Transmission Protocol".

[32] IANA (www.iana.org): Assigned Internet Protocol Numbers, "Protocol Numbers".

[33] IETF RFC 6437: "IPv6 Flow Label Specification".

[34] IETF RFC 791: "Internet Protocol".

[35] Open Geospatial Consortium OGC 05-010: "URNs of definitions in ogc namespace".

[36] 3GPP TS 33.107: "3G security; Lawful interception architecture and functions".

[37] 3GPP TS 37.340: "Evolved Universal Radio Access (E-UTRA) and NR-Multi-connectivity; Stage 2".

[QQ] 3GPP TS 24.379: "Mission Critical Push to Talk (MCPTT) call control; Protocol specification".

[WW] OMA-TS-PoC\_System\_Description-V2\_1-20110802-A: "Push to talk over Cellular (PoC) System Description".

[ZZ] 3GPP TS 23.179: "Functional Architecture and Information Flows to Support Mission Critical Communication Services; Stage 2".

[XX] IETF RFC 4566: "Session Description Protocol".

**\*\*\* Start of Second MODIFICATION \*\*\***

## 7.x PTC service

### 7.x.1 Introduction

The Stage 3 intercept capabilities defined in this clause for the Push to Talk over Cellular (PTC) service apply when supported by a CSP. The term PTC represents either a Push to Talk over Cellular (PoC) or Mission Critical Push to Talk (MCPTT) type service. The use of the term PTC server represents either a MCPTT function or PoC server.

#### 7.x.1.1 Provisioning over LI\_X1

The IRI-POI present in the PTC server is provisioned over LI\_X1 by the LIPF using the X1 protocol as described in clause 5.2.2 of the present document.

The POI in the PTC Server shall support the identifier types given in Table 7.x.1.1-1.

Table 7.x.1.1-1: TargetIdentifier Types for PTC service

|  |  |  |
| --- | --- | --- |
| Identifier | ETSI TS 103 221-1 TargetIdentifier type | Definition |
| iMPU | IMPU | See ETSI TS 103 221-1 [7] |
| iMPI | IMPI | See ETSI TS 103 221-1 [7] |
| mCPTTID | TargetIdentifierExtension | See XSD schema |
| instanceIdentifierURN | TargetIdentifierExtension | See XSD schema |
| pTCChatGroupID | TargetIdentifierExtension | See XSD schema |

#### 7.x.1.2 Generating xIRI over LI\_X2

The IRI-POI present in the PTC server shall send xIRI over LI\_X2 for each of the events listed in TS 33.127 [5] clause 7.6.3, each of which is described in the following clauses. The IRI events are based on the use of 3GPP MCPTT features as defined in 3GPP TS 24.379 [QQ] and OMA PoC features as defined in OMA-TS-PoC\_System\_Description-V2\_1-20110802-A [WW].

### 7.x.2 IRI events

#### 7.x.2.1 PTC registration

The IRI-POI present in the PTC server shall generate an xIRI containing a PTCRegistration record when the IRI-POI present in the PTC server detects that a PTC target matching one of the PTC target identifiers, referenced in clause 7.x.1.1, provided via LI\_X1 has registered, re-registered, or de-registered for PTC services , regardless of whether it is successful or unsuccessful. Accordingly, the IRI-POI in the PTC server generates the xIRI when the following event is detected:

- when the PTC server receives a SIP REGISTER from a PTC target.

Table 7.x.2.1-1: Payload for PTCRegistration record

|  |  |  |
| --- | --- | --- |
| Field name | Description | M/C/O |
| pTCTargetInformation | Provide PTC target identity. At least one among MCPTT ID, IMPU, IMPI, InstanceIdentifierURN and PTCChatGroupID shall be provided for PTCTargetInformation. | M |
| pTCServerURI | Shall include the identity of the PTC server serving the PTC target. | M |
| pTCRegistrationRequest | Identifies the type of registration request (register, re-register, or de-register). | M |
| pTCRegistrationOutcome | Identifies success or failure of the registration. | M |

#### 7.x.2.2 PTC session initiation

The IRI-POI present in the PTC server shall generate an xIRI containing a PTCSessionInitiation record when the IRI-POI present in the PTC server detects that the PTC target initiates an on-demand session or the target receives an invitation to join an on-demand session regardless of the success or the final disposition of the invitation. The PTCSessionIniation record shall also be reported when a chat group is the PTC target. Accordingly, the IRI-POI in the PTC server generates the xIRI when the following events are detected:

- when the PTC server receives a SIP INVITE from a PTC target.

- when the PTC Server sends a SIP INVITE to the PTC target.

- when the PTC Server hosting a PTC chat group session, where the PTC chat group is the target, receives a SIP INVITE from a participating PTC server to initiate a PTC chat group session.

Table 7.x.2.2-1: Payload for PTCSessionInitiation record

|  |  |  |
| --- | --- | --- |
| Field name | Description | M/C/O |
| pTCTargetInformation | Provide PTC target identity. At least one among MCPTT ID, IMPU, IMPI, InstanceIdentifierURN and PTCChatGroupID shall be provided for PTCTargetInformation. | M |
| pTCDirection | Indicates the direction of the session relative to the target: "toTarget" or "fromTarget." | M |
| pTCServerURI | Shall include the identity of the PTC server serving the PTC target. | M |
| pTCSessionInfo | Shall provide PTC session information such as PTC Session URI and PTC Session type (e.g., on-demand, pre-established, ad-hoc, pre-arranged, group session). | M |
| pTCOriginatingID | Shall identify the originating party. | M |
| pTCParticipants | Shall identify the individual PTC participants of the communication session, when known. | C |
| pTCParticipantPresenceStatus | Shall provide the Participant Presence Status, which is a list of:   * *PresenceID*: Identity of PTC client(s) or PTC group, when known. * *PresenceType*: Identifies type of ID [PTC client(s) or PTC group]. * *PresenceStatus*: Presence state of each ID. True indicates PTC client is available, while false indicates PTC client is unavailable.   Report when the Presence functionality is supported by the PTC server and the PTC server assumes the role of the watcher on behalf of the PTC target or in the case of a target PTC chat group, when the PTC server assumes the role of the watcher on behalf of any member of the chat group. | C |
| location | Shall include the PTC target’s location when reporting of the PTC target’s location information is authorized and available. | C |
| pTCBearerCapability | Shall provide when known the media characteristics information elements of the PTC session, encoded in SDP format as per RFC 4566 [XX] clause 5. | C |
| pTCHost | Shall identify the PTC participant who has the authority to initiate and administrate a PTC session, if known. | C |

#### 7.x.2.3 PTC session abandon attempt

The IRI-POI present in the PTC server shall generate an xIRI containing a PTCSessionAbandon record when the IRI-POI present in the PTC server detects that the PTC Session is not established and the request is abandoned before the PTC session starts. Accordingly, the IRI-POI in the PTC server generates the xIRI when the following events are detected:

- when the PTC server serving the PTC target receives a SIP CANCEL from the PTC target or sends a SIP CANCEL to the PTC target.

Table 7.x.2.3-1: Payload for PTCSessionAbandonAttempt record

|  |  |  |
| --- | --- | --- |
| Field name | Description | M/C/O |
| pTCTargetInformation | Provide PTC target identity. At least one among MCPTT ID, IMPU, IMPI, InstanceIdentifierURN and PTCChatGroupID shall be provided for PTCTargetInformation. | M |
| pTCDirection | Indicates the direction of the session relative to the target: "toTarget" or "fromTarget." | M |
| pTCSessionInfo | Shall provide PTC session information such as PTC Session URI and PTC Session type (e.g., on-demand, pre-established, ad-hoc, pre-arranged, group session). | M |
| location | Shall include the PTC target’s location when reporting of the PTC target’s location information is authorized and available. | C |
| pTCAbandonCause | Shall identify the reason for the abandoned PTC session based on the warning header field code provided in a response to a SIP INVITE per 3GPP TS 24.379 [QQ] clause 4.4.2. | M |

#### 7.x.2.4 PTC session start

The IRI-POI present in the PTC server shall generate an xIRI containing a PTCSessionStart record when the IRI-POI present in the PTC server detects that the PTC Session is initiated and communication begins for both an on-demand and pre-established PTC session. The PTCSessionStart record shall also be reported when a chat group is the PTC target. Accordingly, the IRI-POI in the PTC server generates the xIRI when the following events are detected:

- when the PTC server sends a SIP 200 OK to the PTC target in response to a SIP INVITE from the PTC target for an on-demand PTC session where the PTC target originates the PTC session.

- when the PTC server receives a SIP 200 OK from the PTC target in response to a SIP INVITE for an on-demand PTC session where the PTC target receives an invitation to join a PTC session.

- when the PTC server receives a SIP 200 OK from the participant PTC server in response to a SIP INVITE previously sent to that participating PTC server for PTC sessions initiated by the PTC target with a pre-established PTC session (PTC server sends a TBCP Connect to the PTC target with a pre-established session).

- when the PTC server sends a SIP 200 OK to the participant PTC server in response to a SIP INVITE previously received from that participating PTC server for PTC sessions terminated to the PTC target with a pre-established PTC session (PTC server sends a TBCP Connect to the PTC target with a pre-established session).

- when the PTC server hosting a PTC chat group session, where PTC chat group is the PTC target, sends a SIP 200 OK in response to a SIP INVITE previously received from the participant PTC server to initiate a PTC chat group session.

Table 7.x.2.4-1: Payload for PTCSessionStart record

|  |  |  |
| --- | --- | --- |
| Field name | Description | M/C/O |
| pTCTargetInformation | Provide PTC target identity. At least one among MCPTT ID, IMPU, IMPI, InstanceIdentifierURN and PTCChatGroupID shall be provided for PTCTargetInformation. | M |
| pTCDirection | Indicates the direction of the session relative to the target: "toTarget" or "fromTarget." | M |
| pTCServerURI | Shall include the identity of the PTC server serving the PTC target. | M |
| pTCSessionInfo | Shall provide PTC session information such as PTC Session URI and PTC Session type (e.g., on-demand, pre-established, ad-hoc, pre-arranged, group session). | M |
| pTCOriginatingID | Shall identify the originating party. | M |
| pTCParticipants | Shall identify the individual PTC participants of the communication session, when known. | C |
| pTCParticipantPresenceStatus | Shall provide the Participant Presence Status, which is a list of:   * *PresenceID*: Identity of PTC client(s) or PTC group, when known. * *PresenceType*: Identifies type of ID [PTC client(s) or PTC group]. * *PresenceStatus*: Presence state of each ID. True indicates PTC client is available, while false indicates PTC client is unavailable.   Report when the Presence functionality is supported by the PTC server and the PTC server assumes the role of the watcher on behalf of the PTC target. | C |
| location | Shall include the PTC target’s location when reporting of the PTC target’s location information is authorized and available. | C |
| pTCHost | Shall identify the PTC participant who has the authority to initiate and administrate a PTC Session, if known. | C |
| pTCBearerCapability | Shall provide the media characteristics information elements of the PTC session, encoded in SDP format as per RFC 4566 [XX] clause 5 when known. | C |

#### 7.x.2.5 PTC session end

The IRI-POI present in the PTC server shall generate an xIRI containing a PTCSessionEnd record when the IRI-POI present in the PTC server detects that the PTC session is released for any reason (i.e. normal or abnormal release) and voice communications ends. Accordingly, the IRI-POI in the PTC server generates the xIRI when the following events are detected:

- when the PTC server receives a SIP BYE from the PTC target to end the session.

- when the PTC server receives a SIP 200 OK from the PTC target in response to a SIP BYE.

- when the PTC server sends a SIP BYE to the participating PTC server to end the PTC session of a PTC target with a pre-established PTC session (PTC server also sends a TBCP Disconnect to the PTC target with a pre-established PTC session).

- when the PTC server receives a SIP BYE from the participant PTC server to end the PTC session of a PTC target with a pre-established PTC session (PTC server sends a TBCP Disconnect to the PTC target with a pre-established PTC session).

- when the PTC server hosting a PTC chat group session, where PTC chat group is the PTC target, sends a SIP 200 OK in response to a SIP BYE received from the participating PTC server of the last participant in the PTC chat group session.

- when the PTC server sends a SIP 487 to the PTC target in response to a SIP CANCEL to end the session.

Table 7.x.2.5-1: Payload for PTCSessionEnd record

|  |  |  |
| --- | --- | --- |
| Field name | Description | M/C/O |
| pTCTargetInformation | Provide PTC target identity. At least one among MCPTT ID, IMPU, IMPI, InstanceIdentifierURN and PTCChatGroupID shall be provided for PTCTargetInformation. | M |
| pTCDirection | Indicates the direction of the session relative to the target: "toTarget" or "fromTarget." | M |
| pTCServerURI | Shall include the identity of the PTC server serving the PTC target. | M |
| pTCSessionInfo | Shall provide PTC session information such as PTC Session URI and PTC Session type (e.g., on-demand, pre-established, ad-hoc, pre-arranged, group session). | M |
| pTCParticipants | Shall identify the individual PTC participants of the communication session, when known. | C |
| location | Shall include the PTC target’s location when reporting of the PTC target’s location information is authorized and available. | C |
| pTCSessionEndCause | Shall identify the reason for the PTC session end based on the following events per OMA-TS-PoC\_System\_Description-V2\_1-20110802-A [WW] clause 4.5.7:   * PTC session initiator leaves session * Defined participant leaves session * Number of participants less than certain value * PTC Session timer expired * PTC Speech inactive for specified time * All Media types inactive for specified tiime | M |

#### 7.x.2.6 PTC start of interception

The IRI-POI present in the PTC server shall generate an xIRI containing a PTCStartOfInterception record when a PTC target or a PTC chat group as a target has an active PTC session in progress. If multiple PTC Sessions are active at the start of interception, a PTCStartOfInterception record is generated for each active session. Accordingly, the IRI-POI in the PTC server generates the xIRI when the following event is detected:

- when the PTC server detects that LI is enabled on a PTC participant or a PTC chat group with an active PTC session.

Table 7.x.2.6-1: Payload for PTCStartOfIntercept record

|  |  |  |
| --- | --- | --- |
| Field name | Description | M/C/O |
| pTCTargetInformation | Provide PTC target identity. At least one among MCPTT ID, IMPU, IMPI, InstanceIdentifierURN and PTCChatGroupID shall be provided for PTCTargetInformation. | M |
| pTCDirection | Indicates the direction of the session relative to the target: "toTarget" or "fromTarget." | M |
| pTCPreEstSessionID | Identifies the PTC Pre-Established Session Identity when available. | C |
| pTCOriginatingID | Shall identify the originating party. | M |
| pTCSessionInfo | Shall provide PTC session information such as PTC Session URI and PTC Session type (e.g., on-demand, pre-established, ad-hoc, pre-arranged, group session) when available. | C |
| pTCHost | Shall identify the PTC participant who has the authority to initiate and administrate a PTC session, if known. | C |
| pTCParticipants | Shall identify the individual PTC participants of the communication session, when known. | C |
| location | Shall include the PTC target’s location when reporting of the PTC target’s location information is authorized and available. | C |
| pTCMediaStreamAvail | Shall include this parameter to indicate if the PTC target is able/not able to receive media streams immediately. True indicates available for media, while false indicates not able to accept media. | M |
| pTCBearerCapability | Shall provide when known the media characteristics information elements of the PTC session, encoded in SDP format as per RFC 4566 [XX] clause 5. | C |

#### 7.x.2.7 PTC pre-established session

The IRI-POI present in the PTC server shall generate an xIRI containing a PTCPre-EstablishedSession record when the IRI-POI present in the PTC server detects that a pre-established session is setup/modified/released between the PTC target and the PTC server associated with the PTC target. Accordingly, the IRI-POI in the PTC server generates the xIRI when the following events are detected:

- when the PTC Server receives a SIP INVITE from the PTC target to setup a pre-established session.

- when the PTC Server receives a SIP BYE from the PTC target to release a pre-established session.

- when the PTC Server receives a SIP UPDATE or SIP re-INVITE from the PTC target for a pre-established session to modify the current session.

Table 7.x.2.7-1: Payload for PTCPre-EstablishedSession record

|  |  |  |
| --- | --- | --- |
| Field name | Description | M/C/O |
| pTCTargetInformation | Provide PTC target identity. At least one among MCPTT ID, IMPU, IMPI, InstanceIdentifierURN and PTCChatGroupID shall be provided for PTCTargetInformation. | M |
| pTCServerURI | Shall include the identity of the PTC server serving the PTC target. | M |
| rTPSetting | The IP address and the port number of the PTC target at the PTC server for the RTP Session. | M |
| pTCMediaCapability | The codec(s) and media parameters selected by the PTC server from those contained in the original SDP offer from the PTC target’s SIP REFER and encoded in SDP format as per RFC 4566 [XX] clause 5. | M |
| pTCPreEstSessionID | Identifies the PTC Pre-Established Session Identity. | M |
| pTCPreEstStatus | Indicates if the pre-established session is established (setup completed), modified, or released. | M |
| pTCMediaStreamAvail | Shall include for a pre-established session to indicate if the PTC target's PTC client is able/not able to receive media streams immediately, when the pre-established session is established. True indicates available for media, while false indicates not able to accept media. | M |
| location | Shall include the PTC target’s location when reporting of the PTC target’s location information is authorized and available. | C |
| pTCFailureCode | Provide when the pre-established session cannot be established or modified. | C |

#### 7.x.2.8 PTC instant personal alert

The IRI-POI present in the PTC server shall generate an xIRI containing a PTCInstantPersonalAlert record when the IRI-POI present in the PTC server detects that an Instant Personal Alert (IPA) (i.e. a request for one participant to initiate a one-to-one PTC session) is initiated by or sent to the PTC target. Accordingly, the IRI-POI in the PTC server generates the xIRI when the following events are detected:

- when the PTC server receives a SIP MESSAGE from a PTC target for an IPA.

- when the PTC Server sends a SIP MESSAGE to the PTC target for an IPA.

Table 7.x.2.8-1: Payload for PTCInstantPersonalAlert record

|  |  |  |
| --- | --- | --- |
| Field name | Description | M/C/O |
| pTCTargetInformation | Provide PTC target identity. At least one among MCPTT ID, IMPU, IMPI, InstanceIdentifierURN and PTCChatGroupID shall be provided for PTCTargetInformation. | M |
| pTCIPAPartyID | Identifies the PTC participant that receives or has sent the Instant Personal Alert to the target. | M |
| pTCIPADirection | Identifies the direction (To PTC target or From PTC target) of the Instant Personal Alert. | M |

#### 7.x.2.9 PTC party join

The IRI-POI present in the PTC server hosting the PTC chat group session when the PTC chat group is the PTC target, shall generate an xIRI containing a PTCPartyJoin record when the IRI-POI present in that PTC server detects when a PTC participant joins (or re-joins) an on-going PTC chat group session. The PTCPartyJoin record shall also be generated when the IRI-POI present in the participating PTC server of the PTC target detects when a PTC Participant joins (or re-joins) an on-going PTC chat group session. Accordingly, the IRI-POI in the participating PTC server generates the xIRI when the following event is detected:

- when the PTC server hosting a PTC chat group session sends a SIP 200 OK in response to a SIP INVITE indicating a PTC participant joining the PTC chat group session.

- when the participating PTC server of a PTC target forwards a SIP NOTIFY (received from the PTC server hosting the PTC chat group session) to the PTC target containing information about a PTC participant joining the PTC chat group session.

Table 7.x.2.9-1: Payload for PTCPartyJoin record

|  |  |  |
| --- | --- | --- |
| Field name | Description | M/C/O |
| pTCTargetInformation | Provide PTC target identity. At least one among MCPTT ID, IMPU, IMPI, InstanceIdentifierURN and PTCChatGroupID shall be provided for PTCTargetInformation. | M |
| pTCDirection | Indicates the direction of the session relative to the target: "toTarget" or "fromTarget." | M |
| pTCSessionInfo | Shall provide PTC session information such as PTC Session URI and PTC Session type (e.g., on-demand, pre-established, ad-hoc, pre-arranged, group session). | M |
| pTCParticipants | Shall identify the individual PTC participants of the communication session, when known. | C |
| participantPresenceStatus | Shall provide the Participant Presence Status, which is a list of:   * *PresenceID*: Identity of PTC client(s) or PTC group, when known. * *PresenceType*: Identifies type of ID [PTC client(s) or PTC group]. * *PresenceStatus*: Presence state of each ID. True indicates PTC client is available, while false indicates PTC client is unavailable.   Report when the Presence functionality is supported by the PTC server and the PTC server assumes the role of the watcher on behalf of the PTC target. | C |
| pTCMediaStreamAvail | Shall include this parameter to indicate if the PTC target is able/not able to receive media streams immediately. True indicates available for media, while false indicates not able to accept media. | M |
| pTCBearerCapability | Shall provide when known the media characteristics information elements of the PTC session, encoded in SDP format as per RFC 4566 [XX] clause 5. | C |

#### 7.x.2.10 PTC party drop

The IRI-POI present in the PTC server hosting the PTC chat group session, when the PTC chat group is the PTC target, shall generate an xIRI containing a PTCPartyDrop record when the IRI-POI present in that PTC server detects that a PTC participant leaves the PTC chat group session that still remains active with other PTC participants. The PTCPartyDrop record shall also be generated when the IRI-POI present in the participating PTC server of the PTC target detects when a PTC Participant leaves an on-going PTC chat group session. Accordingly, the IRI-POI in the participating PTC server generates the xIRI when the following event is detected:

- when the PTC server hosting a PTC chat group session, where the PTC chat group is the target, sends a SIP 200 OK in response to a SIP BYE with the PTC chat group session remaining active with other PTC participants.

- when the participating PTC server of a PTC target forwards a SIP NOTIFY (received from the PTC server hosting the PTC chat group session) to the PTC target containing information about a PTC participant leaving the PTC chat group session.

Table 7.x.2.10-1: Payload for PTCPartyDrop record

|  |  |  |
| --- | --- | --- |
| Field name | Description | M/C/O |
| pTCTargetInformation | Provide PTC target identity. At least one among MCPTT ID, IMPU, IMPI, InstanceIdentifierURN and PTCChatGroupID shall be provided for PTCTargetInformation. | M |
| pTCDirection | Indicates the direction of the session relative to the target: "toTarget" or "fromTarget." | M |
| pTCSessionInfo | Shall provide PTC session information such as PTC Session URI and PTC Session type (e.g., on-demand, pre-established, ad-hoc, pre-arranged, group session). | M |
| pTCPartyDrop | Shall provide the identity of the participant that leaves the PTC session. | M |
| pTCParticipantPresenceStatus | Shall provide the Participant Presence Status, which is a list of:   * *PresenceID*: Identity of PTC client(s) or PTC group, when known. * *PresenceType*: Identifies type of ID [PTC client(s) or PTC group]. * *PresenceStatus*: Presence state of each ID. True indicates PTC client is available, while false indicates PTC client is unavailable.   Report when the Presence functionality is supported by the PTC server and the PTC server assumes the role of the watcher on behalf of the PTC target. | C |

#### 7.x.2.11 PTC party hold

The IRI-POI present in the PTC server shall generate an xIRI containing a PTCPartyHold record when the IRI-POI present in the PTC server detects that an on-going PTC session is placed on hold or retrieved from hold by the PTC target or by a PTC participant in a PTC chat group, where the PTC chat group is the PTC target. Accordingly, the IRI-POI in the PTC server generates the xIRI when the following event is detected:

- when the PTC server receives a SIP UPDATE or SIP re-INVITE from the PTC target and returns a SIP 200 OK to the PTC target for hold/resume operations.

- when the PTC server hosting a PTC chat group, where PTC chat group is the PTC target, receives a SIP UPDATE or SIP re-INVITE from a PTC participant for hold/resume operations.

Table 7.x.2.11-1: Payload for PTCPartyHold record

|  |  |  |
| --- | --- | --- |
| Field name | Description | M/C/O |
| pTCTargetInformation | Provide PTC target identity. At least one among MCPTT ID, IMPU, IMPI, InstanceIdentifierURN and PTCChatGroupID shall be provided for PTCTargetInformation. | M |
| pTCDirection | Indicates the direction of the session relative to the target: "toTarget" or "fromTarget." | M |
| pTCSessionInfo | Shall provide PTC session information such as PTC Session URI and PTC Session type (e.g., on-demand, pre-established, ad-hoc, pre-arranged, group session). | M |
| pTCParticipants | Shall identify the individual PTC participants of the communication session, when known. | C |
| pTCHoldID | The identity of the PTC participant that placed the PTC session on hold or retrieved the held PTC session. | M |
| pTCHoldRetrieveInd | Shall indicate the PTC session is put on hold (i.e., deactivate Media Bursts or a PTC session is locked for talking/listening) or retrieved from hold. True indication equals placed on hold, false indication was retrieved from hold. | M |

#### 7.x.2.12 PTC media modification

The IRI-POI present in the PTC server shall generate an xIRI containing a PTCMediaModification record when the IRI-POI present in the PTC server detects that a re-negotiation of the media parameters occurs during a PTC session involving the PTC target. Accordingly, the IRI-POI in the PTC server generates the xIRI when the following event is detected:

- when the PTC server receives a SIP UPDATE or SIP reINVITE to indicate a PTC media modification on a PTC session being intercepted.

Table 7.x.2.12-1: Payload for PTCMediaModification record

|  |  |  |
| --- | --- | --- |
| Field name | Description | M/C/O |
| pTCTargetInformation | Provide PTC target identity. At least one among MCPTT ID, IMPU, IMPI, InstanceIdentifierURN and PTCChatGroupID shall be provided for PTCTargetInformation. | M |
| pTCDirection | Indicates the direction of the session relative to the target: "toTarget" or "fromTarget." | M |
| pTCSessionInfo | Shall provide PTC session information such as PTC Session URI and PTC Session type (e.g., on-demand, pre-established, ad-hoc, pre-arranged, group session). | M |
| pTCMediaStreamAvail | Shall include this parameter to indicate if the PTC target is able/not able to receive media streams immediately. True indicates available for media, while false indicates not able to accept media. | M |
| pTCBearerCapability | Shall provide when known the media characteristics information elements of the PTC session, encoded in SDP format as per RFC 4566 [XX] clause 5. | C |

#### 7.x.2.13 PTC group advertisement

The IRI-POI present in the PTC server shall generate an xIRI containing a PTCGroupAdvertisement record when the IRI-POI present in the PTC server detects when a PTC target sends group advertisement information to a single PTC participant, a list of PTC participants, or to all members of a PTC chat group, as well as when a PTC target receives group advertisement information from a single PTC participant, a list of PTC participants, or from members of a PTC chat group using the group identity. Accordingly, the IRI-POI in the PTC server generates the xIRI when the following events are detected:

- when the PTC server receives a SIP MESSAGE (containing group advertisement information) from a PTC target.

- when the PTC server sends a SIP MESSAGE (containing group advertisement information) to the PTC target.

Table 7.x.2.13-1: Payload for PTCGroupAdvertisement record

|  |  |  |
| --- | --- | --- |
| Field name | Description | M/C/O |
| pTCTargetInformation | Provide PTC target identity. At least one among MCPTT ID, IMPU, IMPI, InstanceIdentifierURN and PTCChatGroupID shall be provided for PTCTargetInformation. | M |
| pTCDirection | Indicates the direction of the session relative to the target: "toTarget" or "fromTarget." | M |
| pTCIDList | Shall provide Identities of each participant from the target's contact list (i.e., individuals) and PTC group list (i.e., list of pre-identified individuals using a group identification) for a group call when available. | C |
| pTCGroupAuthRule | Identifies the action requested by the PTC target to the PTC Group Authorization Rules.   * Report when action requested to the PTC Group Authorization Rules by the PTC target. * Report when the PTC target attempts a change or queries the access control list(s). | C |
| pTCGroupAdSender | Identifies thesender of the group advertisement. | M |
| pTCGroupNickname | The nickname is a human-readable tag (e.g., “display-name” in a SIP header associated with a PTC client or PTC group per OMA-TS-PoC\_System\_Description-V2\_1-20110802-A [WW]). | C |

#### 7.x.2.14 PTC floor control

The IRI-POI present in the PTC server shall generate an xIRI containing a PTCFloorControl record when the IRI-POI present in the PTC server detects when the PTC target requests floor control (i.e., send media), when floor control is granted to PTC target, when floor control request from the PTC target is rejected/released, when the floor becomes open (e.g., idle), when the floor control request from the PTC target is queued, when the floor control request from the PTC target is dequeued, or when the floor control request is revoked. In addition, when the PTC chat group is the PTC target, the IRI-POI present in the PTC server hosting the PTC chat group shall generate an xIRI containing a PTCFloorControl record when the IRI-POI present in the PTC server detects any of the previously mentioned scenarios for all PTC participants participating in the PTC chat group session. Accordingly, the IRI-POI in the PTC server generates the xIRI when the following events are detected:

- when the PTC server receives a TBCP Talk Burst Request from the PTC target.

- when the PTC server hosting the PTC chat group, where the PTC chat group is the PTC target, receives a TBCP Talk Burst Request from a PTC participant.

- when the PTC server sends a TBCP Talk Burst Granted to a PTC target.

- when the PTC server hosting the PTC chat group, where the PTC chat group is the PTC target, sends a TBCP Talk Burst Granted to a PTC participant.

- when the PTC server sends a TBCP Talk Burst Taken to a PTC target.

- when the PTC server hosting the PTC chat group, where the PTC chat group is the PTC target, sends a TBCP Talk Burst Taken to a PTC participant.

- when the PTC server sends a TBCP Talk Burst Deny to a PTC target.

- when the PTC server hosting the PTC chat group, where the PTC chat group is the PTC target, sends a TBCP Talk Burst Deny to a PTC participant.

- when the PTC server sends a TBCP Talk Burst Release to a PTC target.

- when the PTC server hosting the PTC chat group, where the PTC chat group is the PTC target, sends a TBCP Talk Burst Release to a PTC participant.

- when the PTC server sends a TBCP Talk Burst Idle to a PTC target.

- when the PTC server hosting the PTC chat group, where the PTC chat group is the PTC target, sends a TBCP Talk Burst Idle to a PTC participant.

- when the PTC server sends a TBCP Talk Burst Request Queue Status Response to a PTC target.

- when the PTC server hosting the PTC chat group, where the PTC chat group is the PTC target, sends a TBCP Talk Burst Request Queue Status Response to a PTC participant.

- when the PTC server receives a TBCP Talk Burst Cancel from a PTC target.

- when the PTC server hosting the PTC chat group, where the PTC chat group is the PTC target, receives a TBCP Talk Burst Cancel from a PTC participant.

- when the PTC server sends a TBCP Talk Burst Revoke to a PTC target.

- when the PTC server hosting the PTC chat group, where the PTC chat group is the PTC target, sends a TBCP Talk Burst Revke to a PTC participant.

Table 7.x.2.14-1: Payload for PTCFloorControl record

|  |  |  |
| --- | --- | --- |
| Field name | Description | M/C/O |
| pTCTargetInformation | Provide PTC target identity. At least one among MCPTT ID, IMPU, IMPI, InstanceIdentifierURN and PTCChatGroupID shall be provided for PTCTargetInformation. | M |
| pTCDirection | Indicates the direction of the session relative to the target: "toTarget" or "fromTarget." | M |
| pTCSessioninfo | Shall provide PTC session information such as PTC Session URI and PTC Session type (e.g., on-demand, pre-established, ad-hoc, pre-arranged, group session). | M |
| pTCFloorActivity | Sequence of:   1. “TBCP\_Request”: Received by the PTC server to request permission for the PTC target or PTC participant to send a talk burst. 2. “TBCP\_Granted”: Used by the PTC server to notify the PTC target or PTC participant that it has been granted permission to send a talk burst. 3. “TBCP\_Deny”: Used by the PTC server to notify a PTC target or PTC participant that it has been denied permission to send a talk burst. 4. “TBCP\_Idle”: Used by the PTC server to notify the PTC target or PTC participant that no one has the permission to send a Talk Burst at the moment and that it may accept the TBCP talk burst request message. 5. “TBCP\_Taken”: Used by the PTC server to notify the PTC target or PTC participant that another PTC participant has been given permission to send a talk burst. 6. “TBCP\_Revoke”: Used by the PTC server to revoke the media resource from the PTC target or PTC participant and can be used for preemption functionality, but is also used by the system to prevent overly long use of the media resource. 7. “TBCP\_Queued”: Indicates the request to talk is queued, if queued floor control is supported. Include identification of the PTC target or PTC participant that has the queued talk burst, if known. 8. “TBCP\_Release”: Indicates the request to talk has completed. | M |
| pTCFloorSpeakerID | Include identification of the PTC participant that has initiated the talk burst, if known. | C |
| pTCMaxTBTime | Include the maximum duration value for the talk burst before the permission is revoked. This parameter is defined in seconds. Provide when known. | C |
| pTCQueuedFloorControl | Indicates if queuing is supported by the PTC server and the PTC target's device. | C |
| pTCQueuedPosition | Include if queue position in the TBCP is detected by the IRI-POI. | C |
| pTCTalkBurstPriority | If more than one level of priority is supported, indicates the talk burst priority level of the PTC target. | C |
| pTCTalkBurstReason | The reason for the denial or revoke of a Talk Burst. Provide when known. | C |

#### 7.x.2.15 PTC target presence

The IRI-POI present in the PTC server shall generate an xIRI containing a PTCTargetPresence record when the IRI-POI present in the PTC server detects that the PTC server publishes network presence information to the Presence server on behalf of the PTC target. Accordingly, the IRI-POI in the PTC server generates the xIRI when the following event is detected:

- when the PTC server sends a SIP PUBLISH message to the Presence server based on the PTC target’s PTC session involvement.

Table 7.x.2.15-1: Payload for PTCTargetPresence record

|  |  |  |
| --- | --- | --- |
| Field name | Description | M/C/O |
| pTCTargetInformation | Provide PTC target identity. At least one among MCPTT ID, IMPU, IMPI, InstanceIdentifierURN and PTCChatGroupID shall be provided for PTCTargetInformation. | M |
| pTCTargetPresenceStatus | Shall provide the PTC target presence status, which is a list of:   * PresenceID: Identity of PTC client(s) or PTC group, when known. * PresenceType: Identifies type of ID [PTC client(s) or PTC group]. * PresenceStatus: Presence state of each ID. True indicates PTC target is available, while false indicates PTC target is unavailable. | M |

#### 7.x.2.16 PTC participant presence

The IRI-POI present in the PTC server shall generate an xIRI containing a PTCParticipantPresence record when the IRI-POI present in the PTC server (when it supports the Presence functionality and assumes the role of the Watcher on behalf of the PTC target) detects that the PTC server receives presence status notifications from the Presence servers after having subscribed to the PTC presence status of other PTC participants (i.e. participants in communication with the PTC target). Accordingly, the IRI-POI in the PTC server generates the xIRI when the following event is detected:

- when the PTC server receives a SIP NOTIFY in response to a SIP SUBSCRIBE updating presence information for a participant.

Table 7.x.2.16-1: Payload for PTCParticipantPresence record

|  |  |  |
| --- | --- | --- |
| Field name | Description | M/C/O |
| pTCTargetInformation | Provide PTC target identity. At least one among MCPTT ID, IMPU, IMPI, InstanceIdentifierURN and PTCChatGroupID shall be provided for PTCTargetInformation. | M |
| pTCParticipantPresenceStatus | Shall provide the Participant Presence Status, which is a list of:   * *PresenceID*: Identity of PTC client(s) or PTC group, when known. * *PresenceType*: Identifies type of ID [PTC client(s) or PTC group]. * *PresenceStatus*: Presence state of each ID. True indicates PTC client is available, while false indicates PTC client is unavailable.   Report when the Presence functionality is supported by the PTC server and the PTC server assumes the role of the watcher on behalf of the PTC target. | M |

#### 7.x.2.17 PTC list management

The IRI-POI present in the PTC server shall generate an xIRI containing a PTCListManagement record when the IRI-POI present in the PTC server detects that the PTC target attempts to change their contact list/group list(s) or those lists are updated by the network. Accordingly, the IRI-POI in the PTC server generates the xIRI when the following events are detected:

- when the PTC server receives a SIP PUBLISH from a PTC target to change the PTC target’s contact list or group list(s).

- when the PTC server receives a SIP NOTIFY from other PTC participants updating the PTC target’s contact list or group list(s) (e.g., participant reachability).

Table 7.x.2.17-1: Payload for PTCListManagement record

|  |  |  |
| --- | --- | --- |
| Field name | Description | M/C/O |
| pTCTargetInformation | Provide PTC target identity. At least one among MCPTT ID, IMPU, IMPI, InstanceIdentifierURN and PTCChatGroupID shall be provided for PTCTargetInformation. | M |
| pTCDirection | Indicates the direction of the session relative to the target: "toTarget" or "fromTarget." | M |
| pTCListManagementType | The "List Management Attempts" identify the type of list being managed by the target when available.   1. ContactListManagementAttempt 2. GroupListManagementAttempt 3. ContactListManagementResult 4. GroupListManagementResult 5. Request unsuccessful   For example, a) and b) are reported when PTC target attempts changes to their contact list and their PTC group list(s).  The "List Management Results" identify the network response to a modification by the PTC target.  For example, c), d), or e) is reported when the network notifies the PTC target of changes to their contact list or their PTC group list(s). | C |
| pTCListManagementAction | Identifies the action requested by the PTC target to the contact lists or PTC group list(s). Report when PTC target attempts changes to his contact list or PTC group list(s).   1. Create 2. Modify 3. Retrieve 4. Delete 5. Notify   Also report when a notification is sent to the PTC target due to changes occurring to his contact list or PTC group list(s). | C |
| pTCListManagementFailure | Report when list management request is unsuccessful. | C |
| pTCContactID | Identity of the contact in the list. One contact per contact list or PTC group list. Report if known. | C |
| pTCIDList | Shall provide identities of each participant from the PTC target's contact list (i.e., individuals) and PTC group list (i.e., list of pre-identified individuals using a group identification) for a group call. Report if known. | C |
| pTCHost | Identifies the PTC participant who has authority to initiate and administrate an active PTC group session. Provide when known. | C |

#### 7.x.2.18 PTC access policy

The IRI-POI present in the PTC server shall generate an xIRI containing a PTCAccessPolicy record when the IRI-POI present in the PTC server detects when the PTC target attempts to change the access control lists (e.g. PTC user access policy and PTC group authorization rules) located in the PTC XML Document Management Server (XDMS). Accordingly, the IRI-POI in the PTC server generates the xIRI when the following event is detected:

- when the PTC server receives a SIP PUBLISH from a PTC target to change the access control lists.

Table 7.x.2.18-1: Payload for PTCAccessPolicy record

|  |  |  |
| --- | --- | --- |
| Field name | Description | M/C/O |
| pTCTargetInformation | Provide PTC target identity. At least one among MCPTT ID, IMPU, IMPI, InstanceIdentifierURN and PTCChatGroupID shall be provided for PTCTargetInformation. | M |
| pTCDirection | Indicates the direction of the session relative to the target: "toTarget" or "fromTarget." | M |
| pTCAccessPolicyType | Identifies the type of access policy list being managed or queried by the target when known.   1. PTCUserAccessPolicyAttempt 2. GroupAuthorizationRulesAttempt 3. PTCUserAccessPolicyQuery 4. GroupAuthorizationRulesQuery 5. PTCUserAccessPolicyResult 6. GroupAuthorizationRulesResult 7. Request unsuccessful  * Report a), b), c), or d) when the PTC target attempts a change or queries the Access Control list(s). * Report e), f), or g) when the network notifies the target of changes to the access control list(s) or the request was unsuccessful. | C |
| pTCUserAccessPolicy | Identifies the action requested by the PTC target to the PTC user or group access policy.   1. Allow Incoming PTC session request 2. Block Incoming PTC session request 3. Allow Auto Answer Mode 4. Allow Override Manual Answer Mode  * Report when action requested to the PTC user access policy. * Report when the PTC target attempts a change or queries the access control list(s). | C |
| pTCGroupAuthRule | Identifies the action requested by the PTC target to the PTC Group Authorization Rules.   1. Allow Initiating PTC session 2. Block Initiating PTC session 3. Allow Joining PTC session 4. Block Joining PTC session 5. Allow Add Participants 6. Block Add Participants 7. Allow Subscription PTC session state 8. Block Subscription PTC session state 9. Allow Anonymity 10. Forbid Anonymity  * Report when action requested to the PTC group authorization rules by the PTC target. * Report when the PTC target attempts a change or queries the access control List(s). | C |
| pTCContactID | Identity of the contact in the list. One contact per contact list or PTC group list. Report if known. | C |
| pTCAccessPolicyFailure | Reports the reason for failure when access policy request is unsuccessful. | C |

**\*\*\* Start of Third MODIFICATION \*\*\***

Annex A (normative): Structure of both the Internal and External Interfaces

TS33128Payloads

{itu-t(0) identified-organization(4) etsi(0) securityDomain(2) lawfulIntercept(2) threeGPP(4) ts33128(19) r16(16) version3(3)}

DEFINITIONS IMPLICIT TAGS EXTENSIBILITY IMPLIED ::=

BEGIN

-- =============

-- Relative OIDs

-- =============

tS33128PayloadsOID RELATIVE-OID ::= {threeGPP(4) ts33128(19) r16(16) version3(3)}

xIRIPayloadOID RELATIVE-OID ::= {tS33128PayloadsOID xIRI(1)}

xCCPayloadOID RELATIVE-OID ::= {tS33128PayloadsOID xCC(2)}

iRIPayloadOID RELATIVE-OID ::= {tS33128PayloadsOID iRI(3)}

cCPayloadOID RELATIVE-OID ::= {tS33128PayloadsOID cC(4)}

lINotificationPayloadOID RELATIVE-OID ::= {tS33128PayloadsOID lINotification(5)}

-- ===============

-- X2 xIRI payload

-- ===============

XIRIPayload ::= SEQUENCE

{

xIRIPayloadOID [1] RELATIVE-OID,

event [2] XIRIEvent

}

XIRIEvent ::= CHOICE

{

-- Access and mobility related events, see clause 6.2.2

registration [1] AMFRegistration,

deregistration [2] AMFDeregistration,

locationUpdate [3] AMFLocationUpdate,

startOfInterceptionWithRegisteredUE [4] AMFStartOfInterceptionWithRegisteredUE,

unsuccessfulAMProcedure [5] AMFUnsuccessfulProcedure,

-- PDU session-related events, see clause 6.2.3

pDUSessionEstablishment [6] SMFPDUSessionEstablishment,

pDUSessionModification [7] SMFPDUSessionModification,

pDUSessionRelease [8] SMFPDUSessionRelease,

startOfInterceptionWithEstablishedPDUSession [9] SMFStartOfInterceptionWithEstablishedPDUSession,

unsuccessfulSMProcedure [10] SMFUnsuccessfulProcedure,

-- Subscriber-management related events, see clause 7.2.2

servingSystemMessage [11] UDMServingSystemMessage,

-- SMS-related events, see clause 6.2.5

sMSMessage [12] SMSMessage,

-- LALS-related events, see clause 7.3.3

lALSReport [13] LALSReport,

-- PDHR/PDSR-related events, see clause 6.2.3.4.1

pDHeaderReport [14] PDHeaderReport,

pDSummaryReport [15] PDSummaryReport,

-- PTC-related events, see clause 7.x.2

pTCRegistration [36] PTCRegistration,

pTCSessionInitiation [37] PTCSessionInitiation,

pTCSessionAbandon [38] PTCSessionAbandon,

pTCSessionStart [39] PTCSessionStart,

pTCSessionEnd [40] PTCSessionEnd,

pTCStartOfInterception [41] PTCStartOfInterception,

pTCPreEstablishedSession [42] PTCPreEstablishedSession,

pTCInstantPersonalAlert [43] PTCInstantPersonalAlert,

pTCPartyJoin [44] PTCPartyJoin,

pTCPartyDrop [45] PTCPartyDrop,

pTCPartyHold [46] PTCPartyHold,

pTCMediaModification [47] PTCMediaModification,

pTCGroupAdvertisement [48] PTCGroupAdvertisement,

pTCFloorControl [49] PTCFloorControl,

pTCTargetPresence [50] PTCTargetPresence,

pTCParticipantPresence [51] PTCParticipantPresence,

pTCListManagement [52] PTCListManagement,

pTCAccessPolicy [53] PTCAccessPolicy

}

-- ==============

-- X3 xCC payload

-- ==============

-- No explicit payload required in release 15, see clause 6.2.3.5

-- ===============

-- HI2 IRI payload

-- ===============

IRIPayload ::= SEQUENCE

{

iRIPayloadOID [1] RELATIVE-OID,

event [2] IRIEvent,

targetIdentifiers [3] SEQUENCE OF IRITargetIdentifier OPTIONAL

}

IRIEvent ::= CHOICE

{

-- Registration-related events, see clause 6.2.2

registration [1] AMFRegistration,

deregistration [2] AMFDeregistration,

locationUpdate [3] AMFLocationUpdate,

startOfInterceptionWithRegisteredUE [4] AMFStartOfInterceptionWithRegisteredUE,

unsuccessfulRegistrationProcedure [5] AMFUnsuccessfulProcedure,

-- PDU session-related events, see clause 6.2.3

pDUSessionEstablishment [6] SMFPDUSessionEstablishment,

pDUSessionModification [7] SMFPDUSessionModification,

pDUSessionRelease [8] SMFPDUSessionRelease,

startOfInterceptionWithEstablishedPDUSession [9] SMFStartOfInterceptionWithEstablishedPDUSession,

unsuccessfulSessionProcedure [10] SMFUnsuccessfulProcedure,

-- Subscriber-management related events, see clause 7.2.2

servingSystemMessage [11] UDMServingSystemMessage,

-- SMS-related events, see clause 6.2.5

sMSMessage [12] SMSMessage,

-- LALS-related events, see clause 7.3.3

lALSReport [13] LALSReport,

-- PDHR/PDSR-related events, see clause 6.2.3.4.1

pDHeaderReport [14] PDHeaderReport,

pDSummaryReport [15] PDSummaryReport,

-- MDF-related events, see clause 7.3.4

mDFCellSiteReport [16] MDFCellSiteReport,

-- PTC-related events, see clause 7.x.2

pTCRegistration [36] PTCRegistration,

pTCSessionInitiation [37] PTCSessionInitiation,

pTCSessionAbandon [38] PTCSessionAbandon,

pTCSessionStart [39] PTCSessionStart,

pTCSessionEnd [40] PTCSessionEnd,

pTCStartOfInterception [41] PTCStartOfInterception,

pTCPreEstablishedSession [42] PTCPreEstablishedSession,

pTCInstantPersonalAlert [43] PTCInstantPersonalAlert,

pTCPartyJoin [44] PTCPartyJoin,

pTCPartyDrop [45] PTCPartyDrop,

pTCPartyHold [46] PTCPartyHold,

pTCMediaModification [47] PTCMediaModification,

pTCGroupAdvertisement [48] PTCGroupAdvertisement,

pTCFloorControl [49] PTCFloorControl,

pTCTargetPresence [50] PTCTargetPresence,

pTCParticipantPresence [51] PTCParticipantPresence,

pTCListManagement [52] PTCListManagement,

pTCAccessPolicy [53] PTCAccessPolicy

}

IRITargetIdentifier ::= SEQUENCE

{

identifier [1] TargetIdentifier,

provenance [2] TargetIdentifierProvenance OPTIONAL

}

-- ==============

-- HI3 CC payload

-- ==============

CCPayload ::= SEQUENCE

{

cCPayloadOID [1] RELATIVE-OID,

pDU [2] CCPDU

}

CCPDU ::= CHOICE

{

uPFCCPDU [1] UPFCCPDU

}

-- ===========================

-- HI4 LI notification payload

-- ===========================

LINotificationPayload ::= SEQUENCE

{

lINotificationPayloadOID [1] RELATIVE-OID,

notification [2] LINotificationMessage

}

LINotificationMessage ::= CHOICE

{

lINotification [1] LINotification

}

-- ==================

-- 5G AMF definitions

-- ==================

-- See clause 6.2.2.2.2 for details of this structure

AMFRegistration ::= SEQUENCE

{

registrationType [1] AMFRegistrationType,

registrationResult [2] AMFRegistrationResult,

slice [3] Slice OPTIONAL,

sUPI [4] SUPI,

sUCI [5] SUCI OPTIONAL,

pEI [6] PEI OPTIONAL,

gPSI [7] GPSI OPTIONAL,

gUTI [8] FiveGGUTI,

location [9] Location OPTIONAL,

non3GPPAccessEndpoint [10] UEEndpointAddress OPTIONAL

}

-- See clause 6.2.2.2.3 for details of this structure

AMFDeregistration ::= SEQUENCE

{

deregistrationDirection [1] AMFDirection,

accessType [2] AccessType,

sUPI [3] SUPI OPTIONAL,

sUCI [4] SUCI OPTIONAL,

pEI [5] PEI OPTIONAL,

gPSI [6] GPSI OPTIONAL,

gUTI [7] FiveGGUTI OPTIONAL,

cause [8] FiveGMMCause OPTIONAL,

location [9] Location OPTIONAL

}

-- See clause 6.2.2.2.4 for details of this structure

AMFLocationUpdate ::= SEQUENCE

{

sUPI [1] SUPI,

sUCI [2] SUCI OPTIONAL,

pEI [3] PEI OPTIONAL,

gPSI [4] GPSI OPTIONAL,

gUTI [5] FiveGGUTI OPTIONAL,

location [6] Location

}

-- See clause 6.2.2.2.5 for details of this structure

AMFStartOfInterceptionWithRegisteredUE ::= SEQUENCE

{

registrationResult [1] AMFRegistrationResult,

registrationType [2] AMFRegistrationType OPTIONAL,

slice [3] Slice OPTIONAL,

sUPI [4] SUPI,

sUCI [5] SUCI OPTIONAL,

pEI [6] PEI OPTIONAL,

gPSI [7] GPSI OPTIONAL,

gUTI [8] FiveGGUTI,

location [9] Location OPTIONAL,

non3GPPAccessEndpoint [10] UEEndpointAddress OPTIONAL,

timeOfRegistration [11] Timestamp OPTIONAL

}

-- See clause 6.2.2.2.6 for details of this structure

AMFUnsuccessfulProcedure ::= SEQUENCE

{

failedProcedureType [1] AMFFailedProcedureType,

failureCause [2] AMFFailureCause,

requestedSlice [3] NSSAI OPTIONAL,

sUPI [4] SUPI OPTIONAL,

sUCI [5] SUCI OPTIONAL,

pEI [6] PEI OPTIONAL,

gPSI [7] GPSI OPTIONAL,

gUTI [8] FiveGGUTI OPTIONAL,

location [9] Location OPTIONAL

}

-- =================

-- 5G AMF parameters

-- =================

AMFID ::= SEQUENCE

{

aMFRegionID [1] AMFRegionID,

aMFSetID [2] AMFSetID,

aMFPointer [3] AMFPointer

}

AMFDirection ::= ENUMERATED

{

networkInitiated(1),

uEInitiated(2)

}

AMFFailedProcedureType ::= ENUMERATED

{

registration(1),

sMS(2),

pDUSessionEstablishment(3)

}

AMFFailureCause ::= CHOICE

{

fiveGMMCause [1] FiveGMMCause,

fiveGSMCause [2] FiveGSMCause

}

AMFPointer ::= INTEGER (0..63)

AMFRegistrationResult ::= ENUMERATED

{

threeGPPAccess(1),

nonThreeGPPAccess(2),

threeGPPAndNonThreeGPPAccess(3)

}

AMFRegionID ::= INTEGER (0..255)

AMFRegistrationType ::= ENUMERATED

{

initial(1),

mobility(2),

periodic(3),

emergency(4)

}

AMFSetID ::= INTEGER (0..1023)

-- ==================

-- 5G SMF definitions

-- ==================

-- See clause 6.2.3.2.2 for details of this structure

SMFPDUSessionEstablishment ::= SEQUENCE

{

sUPI [1] SUPI OPTIONAL,

sUPIUnauthenticated [2] SUPIUnauthenticatedIndication OPTIONAL,

pEI [3] PEI OPTIONAL,

gPSI [4] GPSI OPTIONAL,

pDUSessionID [5] PDUSessionID,

gTPTunnelID [6] FTEID,

pDUSessionType [7] PDUSessionType,

sNSSAI [8] SNSSAI OPTIONAL,

uEEndpoint [9] SEQUENCE OF UEEndpointAddress OPTIONAL,

non3GPPAccessEndpoint [10] UEEndpointAddress OPTIONAL,

location [11] Location OPTIONAL,

dNN [12] DNN,

aMFID [13] AMFID OPTIONAL,

hSMFURI [14] HSMFURI OPTIONAL,

requestType [15] FiveGSMRequestType,

accessType [16] AccessType OPTIONAL,

rATType [17] RATType OPTIONAL,

sMPDUDNRequest [18] SMPDUDNRequest OPTIONAL

}

-- See clause 6.2.3.2.3 for details of this structure

SMFPDUSessionModification ::= SEQUENCE

{

sUPI [1] SUPI OPTIONAL,

sUPIUnauthenticated [2] SUPIUnauthenticatedIndication OPTIONAL,

pEI [3] PEI OPTIONAL,

gPSI [4] GPSI OPTIONAL,

sNSSAI [5] SNSSAI OPTIONAL,

non3GPPAccessEndpoint [6] UEEndpointAddress OPTIONAL,

location [7] Location OPTIONAL,

requestType [8] FiveGSMRequestType,

accessType [9] AccessType OPTIONAL,

rATType [10] RATType OPTIONAL

}

-- See clause 6.2.3.2.4 for details of this structure

SMFPDUSessionRelease ::= SEQUENCE

{

sUPI [1] SUPI,

pEI [2] PEI OPTIONAL,

gPSI [3] GPSI OPTIONAL,

pDUSessionID [4] PDUSessionID,

timeOfFirstPacket [5] Timestamp OPTIONAL,

timeOfLastPacket [6] Timestamp OPTIONAL,

uplinkVolume [7] INTEGER OPTIONAL,

downlinkVolume [8] INTEGER OPTIONAL,

location [9] Location OPTIONAL

}

-- See clause 6.2.3.2.5 for details of this structure

SMFStartOfInterceptionWithEstablishedPDUSession ::= SEQUENCE

{

sUPI [1] SUPI OPTIONAL,

sUPIUnauthenticated [2] SUPIUnauthenticatedIndication OPTIONAL,

pEI [3] PEI OPTIONAL,

gPSI [4] GPSI OPTIONAL,

pDUSessionID [5] PDUSessionID,

gTPTunnelID [6] FTEID,

pDUSessionType [7] PDUSessionType,

sNSSAI [8] SNSSAI OPTIONAL,

uEEndpoint [9] SEQUENCE OF UEEndpointAddress,

non3GPPAccessEndpoint [10] UEEndpointAddress OPTIONAL,

location [11] Location OPTIONAL,

dNN [12] DNN,

aMFID [13] AMFID OPTIONAL,

hSMFURI [14] HSMFURI OPTIONAL,

requestType [15] FiveGSMRequestType,

accessType [16] AccessType OPTIONAL,

rATType [17] RATType OPTIONAL,

sMPDUDNRequest [18] SMPDUDNRequest OPTIONAL

}

-- See clause 6.2.3.2.6 for details of this structure

SMFUnsuccessfulProcedure ::= SEQUENCE

{

failedProcedureType [1] SMFFailedProcedureType,

failureCause [2] FiveGSMCause,

initiator [3] Initiator,

requestedSlice [4] NSSAI OPTIONAL,

sUPI [5] SUPI OPTIONAL,

sUPIUnauthenticated [6] SUPIUnauthenticatedIndication OPTIONAL,

pEI [7] PEI OPTIONAL,

gPSI [8] GPSI OPTIONAL,

pDUSessionID [9] PDUSessionID OPTIONAL,

uEEndpoint [10] SEQUENCE OF UEEndpointAddress OPTIONAL,

non3GPPAccessEndpoint [11] UEEndpointAddress OPTIONAL,

dNN [12] DNN OPTIONAL,

aMFID [13] AMFID OPTIONAL,

hSMFURI [14] HSMFURI OPTIONAL,

requestType [15] FiveGSMRequestType OPTIONAL,

accessType [16] AccessType OPTIONAL,

rATType [17] RATType OPTIONAL,

sMPDUDNRequest [18] SMPDUDNRequest OPTIONAL,

location [19] Location OPTIONAL

}

-- =================

-- 5G SMF parameters

-- =================

SMFFailedProcedureType ::= ENUMERATED

{

pDUSessionEstablishment(1),

pDUSessionModification(2),

pDUSessionRelease(3)

}

-- =================

-- 5G UPF parameters

-- =================

UPFCCPDU ::= OCTET STRING

-- ==================

-- 5G UDM definitions

-- ==================

UDMServingSystemMessage ::= SEQUENCE

{

sUPI [1] SUPI,

pEI [2] PEI OPTIONAL,

gPSI [3] GPSI OPTIONAL,

gUAMI [4] GUAMI OPTIONAL,

gUMMEI [5] GUMMEI OPTIONAL,

pLMNID [6] PLMNID OPTIONAL,

servingSystemMethod [7] UDMServingSystemMethod

}

-- =================

-- 5G UDM parameters

-- =================

UDMServingSystemMethod ::= ENUMERATED

{

amf3GPPAccessRegistration(0),

amfNon3GPPAccessRegistration(1),

unknown(2)

}

-- ===================

-- 5G SMSF definitions

-- ===================

-- See clause 6.2.5.3 for details of this structure

SMSMessage ::= SEQUENCE

{

originatingSMSParty [1] SMSParty,

terminatingSMSParty [2] SMSParty,

direction [3] Direction,

transferStatus [4] SMSTransferStatus,

otherMessage [5] SMSOtherMessageIndication OPTIONAL,

location [6] Location OPTIONAL,

peerNFAddress [7] SMSNFAddress OPTIONAL,

peerNFType [8] SMSNFType OPTIONAL,

sMSTPDUData [9] SMSTPDUData OPTIONAL

}

-- ==================

-- 5G SMSF parameters

-- ==================

SMSParty ::= SEQUENCE

{

sUPI [1] SUPI OPTIONAL,

pEI [2] PEI OPTIONAL,

gPSI [3] GPSI OPTIONAL

}

SMSTransferStatus ::= ENUMERATED

{

transferSucceeded(1),

transferFailed(2),

undefined(3)

}

SMSOtherMessageIndication ::= BOOLEAN

SMSNFAddress ::= CHOICE

{

iPAddress [1] IPAddress,

e164Number [2] E164Number

}

SMSNFType ::= ENUMERATED

{

sMSGMSC(1),

iWMSC(2),

sMSRouter(3)

}

SMSTPDUData ::= CHOICE

{

sMSTPDU [1] SMSTPDU

}

SMSTPDU ::= OCTET STRING (SIZE(1..270))

-- ==================

-- 5G PTC definitions

-- ==================

PTCRegistration ::= SEQUENCE

{

pTCTargetInformation [1] PTCTargetInformation,

pTCServerURI [2] UTF8String,

pTCRegistrationRequest [3] PTCRegistrationRequest,

pTCRegistrationOutcome [4] PTCRegistrationOutcome

}

PTCSessionInitiation ::= SEQUENCE

{

pTCTargetInformation [1] PTCTargetInformation,

pTCDirection [2] Direction,

pTCServerURI [3] UTF8String,

pTCSessionInfo [4] PTCSessionInfo,

pTCOriginatingID [5] PTCTargetInformation,

pTCParticipants [6] SEQUENCE OF PTCTargetInformation OPTIONAL,

pTCParticipantPresenceStatus [7] MultipleParticipantPresenceStatus OPTIONAL,

location [8] Location OPTIONAL,

pTCBearerCapability [9] UTF8String OPTIONAL,

pTCHost [10] PTCTargetInformation OPTIONAL

}

PTCSessionAbandon ::= SEQUENCE

{

pTCTargetInformation [1] PTCTargetInformation,

pTCDirection [2] Direction,

pTCSessionInfo [3] PTCSessionInfo,

location [4] Location OPTIONAL,

pTCAbandonCause [5] INTEGER

}

PTCSessionStart ::= SEQUENCE

{

pTCTargetInformation [1] PTCTargetInformation,

pTCDirection [2] Direction,

pTCServerURI [3] UTF8String,

pTCSessionInfo [4] PTCSessionInfo,

pTCOriginatingID [5] PTCTargetInformation,

pTCParticipants [6] SEQUENCE OF PTCTargetInformation OPTIONAL,

pTCParticipantPresenceStatus [7] MultipleParticipantPresenceStatus OPTIONAL,

location [8] Location OPTIONAL,

pTCHost [9] PTCTargetInformation OPTIONAL,

pTCBearerCapability [10] UTF8String OPTIONAL

}

PTCSessionEnd ::= SEQUENCE

{

pTCTargetInformation [1] PTCTargetInformation,

pTCDirection [2] Direction,

pTCServerURI [3] UTF8String,

pTCSessionInfo [4] PTCSessionInfo,

pTCParticipants [5] SEQUENCE OF PTCTargetInformation OPTIONAL,

location [6] Location OPTIONAL,

pTCSessionEndCause [7] PTCSessionEndCause

}

PTCStartOfInterception ::= SEQUENCE

{

pTCTargetInformation [1] PTCTargetInformation,

pTCDirection [2] Direction,

preEstSessionID [3] PTCSessionInfo OPTIONAL,

pTCOriginatingID [4] PTCTargetInformation,

pTCSessionInfo [5] PTCSessionInfo OPTIONAL,

pTCHost [6] PTCTargetInformation OPTIONAL,

pTCParticipants [7] SEQUENCE OF PTCTargetInformation OPTIONAL,

pTCMediaStreamAvail [8] BOOLEAN OPTIONAL,

pTCBearerCapability [9] UTF8String OPTIONAL

}

PTCPreEstablishedSession ::= SEQUENCE

{

pTCTargetInformation [1] PTCTargetInformation,

pTCServerURI [2] UTF8String,

rTPSetting [3] RTPSetting,

pTCMediaCapability [4] UTF8String,

pTCPreEstSessionID [5] PTCSessionInfo,

pTCPreEstStatus [6] PTCPreEstStatus,

pTCMediaStreamAvail [7] BOOLEAN OPTIONAL,

location [8] Location OPTIONAL,

pTCFailureCode [9] PTCFailureCode OPTIONAL

}

PTCInstantPersonalAlert ::= SEQUENCE

{

pTCTargetInformation [1] PTCTargetInformation,

pTCIPAPartyID [2] PTCTargetInformation,

pTCIPADirection [3] Direction

}

PTCPartyJoin ::= SEQUENCE

{

pTCTargetInformation [1] PTCTargetInformation,

pTCDirection [2] Direction,

pTCSessionInfo [3] PTCSessionInfo,

pTCParticipants [4] SEQUENCE OF PTCTargetInformation OPTIONAL,

pTCParticipantPresenceStatus [5] MultipleParticipantPresenceStatus OPTIONAL,

pTCMediaStreamAvail [6] BOOLEAN OPTIONAL,

pTCBearerCapability [7] UTF8String OPTIONAL

}

PTCPartyDrop ::= SEQUENCE

{

pTCTargetInformation [1] PTCTargetInformation,

pTCDirection [2] Direction,

pTCSessionInfo [3] PTCSessionInfo,

pTCPartyDrop [4] PTCTargetInformation,

pTCParticipantPresenceStatus [5] PTCParticipantPresenceStatus OPTIONAL

}

PTCPartyHold ::= SEQUENCE

{

pTCTargetInformation [1] PTCTargetInformation,

pTCDirection [2] Direction,

pTCSessionInfo [3] PTCSessionInfo,

pTCParticipants [4] SEQUENCE OF PTCTargetInformation OPTIONAL,

pTCHoldID [5] SEQUENCE OF PTCTargetInformation,

pTCHoldRetrieveInd [6] BOOLEAN

}

PTCMediaModification ::= SEQUENCE

{

pTCTargetInformation [1] PTCTargetInformation,

pTCDirection [2] Direction,

pTCSessionInfo [3] PTCSessionInfo,

pTCMediaStreamAvail [4] BOOLEAN OPTIONAL,

pTCBearerCapability [5] UTF8String

}

PTCGroupAdvertisement ::=SEQUENCE

{

pTCTargetInformation [1] PTCTargetInformation,

pTCDirection [2] Direction,

pTCIDList [3] SEQUENCE OF PTCTargetInformation OPTIONAL,

pTCGroupAuthRule [4] PTCGroupAuthRule OPTIONAL,

pTCGroupAdSender [5] PTCTargetInformation,

pTCGroupNickname [6] UTF8String OPTIONAL

}

PTCFloorControl ::= SEQUENCE

{

pTCTargetInformation [1] PTCTargetInformation,

pTCDirection [2] Direction,

pTCSessioninfo [3] PTCSessionInfo,

pTCFloorActivity [4] SEQUENCE OF PTCFloorActivity,

pTCFloorSpeakerID [5] PTCTargetInformation OPTIONAL,

pTCMaxTBTime [6] INTEGER OPTIONAL,

pTCQueuedFloorControl [7] BOOLEAN OPTIONAL,

pTCQueuedPosition [8] INTEGER OPTIONAL,

pTCTalkBurstPriority [9] PTCTBPriorityLevel OPTIONAL,

pTCTalkBurstReason [10] PTCTBReasonCode OPTIONAL

}

PTCTargetPresence ::= SEQUENCE

{

pTCTargetInformation [1] PTCTargetInformation,

pTCTargetPresenceStatus [2] PTCParticipantPresenceStatus

}

PTCParticipantPresence ::= SEQUENCE

{

pTCTargetInformation [1] PTCTargetInformation,

pTCParticipantPresenceStatus [2] PTCParticipantPresenceStatus

}

PTCListManagement ::= SEQUENCE

{

pTCTargetInformation [1] PTCTargetInformation,

pTCDirection [2] Direction,

pTCListManagementType [3] PTCListManagementType OPTIONAL,

pTCListManagementAction [4] PTCListManagementAction OPTIONAL,

pTCListManagementFailure [5] PTCListManagementFailure OPTIONAL,

pTCContactID [6] PTCTargetInformation OPTIONAL,

pTCIDList [7] SEQUENCE OF PTCIDList OPTIONAL,

pTCHost [8] PTCTargetInformation OPTIONAL

}

PTCAccessPolicy ::= SEQUENCE

{

pTCTargetInformation [1] PTCTargetInformation,

pTCDirection [2] Direction,

pTCAccessPolicyType [3] PTCAccessPolicyType OPTIONAL,

pTCUserAccessPolicy [4] PTCUserAccessPolicy OPTIONAL,

pTCGroupAuthRule [5] PTCGroupAuthRule OPTIONAL,

pTCContactID [6] PTCTargetInformation OPTIONAL,

pTCAccessPolicyFailure [7] PTCAccessPolicyFailure OPTIONAL

}

-- =================

-- 5G PTC parameters

-- =================

PTCRegistrationRequest ::= ENUMERATED

{

register(1),

reRegister(2),

deRegister(3)

}

PTCRegistrationOutcome ::= ENUMERATED

{

success(1),

failure(2)

}

PTCSessionEndCause ::= ENUMERATED

{

initiaterLeavesSession(1),

definedParticipantLeaves(2),

numberOfParticipants(3),

sessionTimerExpired(4),

pTCSpeechInactive(5),

allMediaTypesInactive(6)

}

PTCTargetInformation ::= SEQUENCE

{

identifiers [1] SEQUENCE SIZE(1..MAX) OF PTCIdentifiers

}

PTCIdentifiers ::= CHOICE

{

mCPTTID [1] UTF8String,

instanceIdentifierURN [2] UTF8String,

pTCChatGroupID [3] PTCChatGroupID,

iMPU [4] IMPU,

iMPI [5] IMPI

}

PTCSessionInfo ::= SEQUENCE

{

pTCSessionURI [1] UTF8String,

pTCSessionType [2] PTCSessionType

}

PTCSessionType ::= ENUMERATED

{

ondemand(1),

preEstablished(2),

adhoc(3),

prearranged(4),

groupSession(5)

}

MultipleParticipantPresenceStatus ::= SEQUENCE OF PTCParticipantPresenceStatus

PTCParticipantPresenceStatus ::= SEQUENCE

{

presenceID [1] PTCTargetInformation,

presenceType [2] PTCPresenceType,

presenceStatus [3] BOOLEAN

}

PTCPresenceType ::= ENUMERATED

{

pTCClient(1),

pTCGroup(2)

}

PTCPreEstStatus ::= ENUMERATED

{

established(1),

modified(2),

released(3)

}

RTPSetting ::= SEQUENCE

{

iPAddress [1] IPAddress,

portNumber [2] PortNumber

}

PTCIDList ::= SEQUENCE

{

pTCPartyID [1] PTCTargetInformation,

pTCChatGroupID [2] PTCChatGroupID

}

PTCChatGroupID ::= SEQUENCE

{

groupIdentity [1] UTF8String

}

PTCFloorActivity ::= ENUMERATED

{

tBCPRequest(1),

tBCPGranted(2),

tBCPDeny(3),

tBCPIdle(4),

tBCPTaken(5),

tBCPRevoke(6),

tBCPQueued(7),

tBCPRelease(8)

}

PTCTBPriorityLevel ::= ENUMERATED

{

preEmptive(1),

highPriority(2),

normalPriority(3),

listenOnly(4)

}

PTCTBReasonCode ::= ENUMERATED

{

noQueuingAllowed(1),

oneParticipantSession(2),

listenOnly(3),

exceededMaxDuration(4),

tBPrevented(5)

}

PTCListManagementType ::= ENUMERATED

{

contactListManagementAttempt(1),

groupListManagementAttempt(2),

contactListManagementResult(3),

groupListManagementResult(4),

requestUnsuccessful(5)

}

PTCListManagementAction ::= ENUMERATED

{

create(1),

modify(2),

retrieve(3),

delete(4),

notify(5)

}

PTCAccessPolicyType ::= ENUMERATED

{

pTCUserAccessPolicyAttempt(1),

groupAuthorizationRulesAttempt(2),

pTCUserAccessPolicyQuery(3),

groupAuthorizationRulesQuery(4),

pTCUserAccessPolicyResult(5),

groupAuthorizationRulesResult(6),

requestUnsuccessful(7)

}

PTCUserAccessPolicy ::= ENUMERATED

{

allowIncomingPTCSessionRequest(1),

blockIncomingPTCSessionRequest(2),

allowAutoAnswerMode(3),

allowOverrideManualAnswerMode(4)

}

PTCGroupAuthRule ::= ENUMERATED

{

allowInitiatingPTCSession(1),

blockInitiatingPTCSession(2),

allowJoiningPTCSession(3),

blockJoiningPTCSession(4),

allowAddParticipants(5),

blockAddParticipants(6),

allowSubscriptionPTCSessionState(7),

blockSubscriptionPTCSessionState(8),

allowAnonymity(9),

forbidAnonymity(10)

}

PTCFailureCode ::= ENUMERATED

{

sessionCannotBeEstablished(1),

sessionCannotBeModified(2)

}

PTCListManagementFailure ::= ENUMERATED

{

requestUnsuccessful(1),

requestUnknown(2)

}

PTCAccessPolicyFailure ::= ENUMERATED

{

requestUnsuccessful(1),

requestUnknown(2)

}

-- ===================

-- 5G LALS definitions

-- ===================

LALSReport ::= SEQUENCE

{

sUPI [1] SUPI OPTIONAL,

pEI [2] PEI OPTIONAL,

gPSI [3] GPSI OPTIONAL,

location [4] Location OPTIONAL

}

-- =====================

-- PDHR/PDSR definitions

-- =====================

PDHeaderReport ::= SEQUENCE

{

pDUSessionID [1] PDUSessionID,

sourceIPAddress [2] IPAddress,

sourcePort [3] PortNumber OPTIONAL,

destinationIPAddress [4] IPAddress,

destinationPort [5] PortNumber OPTIONAL,

nextLayerProtocol [6] NextLayerProtocol,

iPv6flowLabel [7] IPv6FlowLabel OPTIONAL,

direction [8] Direction,

packetSize [9] INTEGER

}

PDSummaryReport ::= SEQUENCE

{

pDUSessionID [1] PDUSessionID,

sourceIPAddress [2] IPAddress,

sourcePort [3] PortNumber OPTIONAL,

destinationIPAddress [4] IPAddress,

destinationPort [5] PortNumber OPTIONAL,

nextLayerProtocol [6] NextLayerProtocol,

iPv6flowLabel [7] IPv6FlowLabel OPTIONAL,

direction [8] Direction,

pDSRSummaryTrigger [9] PDSRSummaryTrigger,

firstPacketTimestamp [10] Timestamp,

lastPacketTimestamp [11] Timestamp,

packetCount [12] INTEGER,

byteCount [13] INTEGER

}

-- ====================

-- PDHR/PDSR parameters

-- ====================

PDSRSummaryTrigger ::= ENUMERATED

{

timerExpiry(1),

packetCount(2),

byteCount(3)

}

-- ===========================

-- LI Notification definitions

-- ===========================

LINotification ::= SEQUENCE

{

notificationType [1] LINotificationType,

appliedTargetID [2] TargetIdentifier OPTIONAL,

appliedDeliveryInformation [3] SEQUENCE OF LIAppliedDeliveryInformation OPTIONAL,

appliedStartTime [4] Timestamp OPTIONAL,

appliedEndTime [5] Timestamp OPTIONAL

}

-- ==========================

-- LI Notification parameters

-- ==========================

LINotificationType ::= ENUMERATED

{

activation(1),

deactivation(2),

modification(3)

}

LIAppliedDeliveryInformation ::= SEQUENCE

{

hI2DeliveryIPAddress [1] IPAddress OPTIONAL,

hI2DeliveryPortNumber [2] PortNumber OPTIONAL,

hI3DeliveryIPAddress [3] IPAddress OPTIONAL,

hI3DeliveryPortNumber [4] PortNumber OPTIONAL

}

-- ===============

-- MDF definitions

-- ===============

MDFCellSiteReport ::= SEQUENCE OF CellInformation

-- =================

-- Common Parameters

-- =================

AccessType ::= ENUMERATED

{

threeGPPAccess(1),

nonThreeGPPAccess(2),

threeGPPandNonThreeGPPAccess(3)

}

Direction ::= ENUMERATED

{

fromTarget(1),

toTarget(2)

}

DNN ::= UTF8String

E164Number ::= NumericString (SIZE(1..15))

FiveGGUTI ::= SEQUENCE

{

mCC [1] MCC,

mNC [2] MNC,

aMFRegionID [3] AMFRegionID,

aMFSetID [4] AMFSetID,

aMFPointer [5] AMFPointer,

fiveGTMSI [6] FiveGTMSI

}

FiveGMMCause ::= INTEGER (0..255)

FiveGSMRequestType ::= ENUMERATED

{

initialRequest(1),

existingPDUSession(2),

initialEmergencyRequest(3),

existingEmergencyPDUSession(4),

modificationRequest(5),

reserved(6),

mAPDURequest(7)

}

FiveGSMCause ::= INTEGER (0..255)

FiveGTMSI ::= INTEGER (0..4294967295)

FTEID ::= SEQUENCE

{

tEID [1] INTEGER (0.. 4294967295),

iPv4Address [2] IPv4Address OPTIONAL,

iPv6Address [3] IPv6Address OPTIONAL

}

GPSI ::= CHOICE

{

mSISDN [1] MSISDN,

nAI [2] NAI

}

GUAMI ::= SEQUENCE

{

aMFID [1] AMFID,

pLMNID [2] PLMNID

}

GUMMEI ::= SEQUENCE

{

mMEID [1] MMEID,

mCC [2] MCC,

mNC [3] MNC

}

HomeNetworkPublicKeyID ::= OCTET STRING

HSMFURI ::= UTF8String

IMEI ::= NumericString (SIZE(14))

IMEISV ::= NumericString (SIZE(16))

IMSI ::= NumericString (SIZE(6..15))

Initiator ::= ENUMERATED

{

uE(1),

network(2),

unknown(3)

}

IPAddress ::= CHOICE

{

iPv4Address [1] IPv4Address,

iPv6Address [2] IPv6Address

}

IPv4Address ::= OCTET STRING (SIZE(4))

IPv6Address ::= OCTET STRING (SIZE(16))

IPv6FlowLabel ::= INTEGER(0..1048575)

MACAddress ::= OCTET STRING (SIZE(6))

MCC ::= NumericString (SIZE(3))

MNC ::= NumericString (SIZE(2..3))

MMEID ::= SEQUENCE

{

mMEGI [1] MMEGI,

mMEC [2] MMEC

}

MMEC ::= NumericString

MMEGI ::= NumericString

MSISDN ::= NumericString (SIZE(1..15))

NAI ::= UTF8String

NextLayerProtocol ::= INTEGER(0..255)

NSSAI ::= SEQUENCE OF SNSSAI

PLMNID ::= SEQUENCE

{

mCC [1] MCC,

mNC [2] MNC

}

PDUSessionID ::= INTEGER (0..255)

PDUSessionType ::= ENUMERATED

{

iPv4(1),

iPv6(2),

iPv4v6(3),

unstructured(4),

ethernet(5)

}

PEI ::= CHOICE

{

iMEI [1] IMEI,

iMEISV [2] IMEISV

}

PortNumber ::= INTEGER(0..65535)

ProtectionSchemeID ::= INTEGER (0..15)

RATType ::= ENUMERATED

{

nR(1),

eUTRA(2),

wLAN(3),

virtual(4)

}

RejectedNSSAI ::= SEQUENCE OF RejectedSNSSAI

RejectedSNSSAI ::= SEQUENCE

{

causeValue [1] RejectedSliceCauseValue,

sNSSAI [2] SNSSAI

}

RejectedSliceCauseValue ::= INTEGER (0..255)

RoutingIndicator ::= INTEGER (0..9999)

SchemeOutput ::= OCTET STRING

Slice ::= SEQUENCE

{

allowedNSSAI [1] NSSAI OPTIONAL,

configuredNSSAI [2] NSSAI OPTIONAL,

rejectedNSSAI [3] RejectedNSSAI OPTIONAL

}

SMPDUDNRequest ::= OCTET STRING

SNSSAI ::= SEQUENCE

{

sliceServiceType [1] INTEGER (0..255),

sliceDifferentiator [2] OCTET STRING (SIZE(3)) OPTIONAL

}

SUCI ::= SEQUENCE

{

mCC [1] MCC,

mNC [2] MNC,

routingIndicator [3] RoutingIndicator,

protectionSchemeID [4] ProtectionSchemeID,

homeNetworkPublicKeyID [5] HomeNetworkPublicKeyID,

schemeOutput [6] SchemeOutput

}

SUPI ::= CHOICE

{

iMSI [1] IMSI,

nAI [2] NAI

}

SUPIUnauthenticatedIndication ::= BOOLEAN

TargetIdentifier ::= CHOICE

{

sUPI [1] SUPI,

iMSI [2] IMSI,

pEI [3] PEI,

iMEI [4] IMEI,

gPSI [5] GPSI,

mISDN [6] MSISDN,

nAI [7] NAI,

iPv4Address [8] IPv4Address,

iPv6Address [9] IPv6Address,

ethernetAddress [10] MACAddress

}

TargetIdentifierProvenance ::= ENUMERATED

{

lEAProvided(1),

observed(2),

matchedOn(3),

other(4)

}

Timestamp ::= GeneralizedTime

UEEndpointAddress ::= CHOICE

{

iPv4Address [1] IPv4Address,

iPv6Address [2] IPv6Address,

ethernetAddress [3] MACAddress

}

-- ===================

-- Location parameters

-- ===================

Location ::= SEQUENCE

{

locationInfo [1] LocationInfo OPTIONAL,

positioningInfo [2] PositioningInfo OPTIONAL,

locationPresenceReport [3] LocationPresenceReport OPTIONAL

}

CellSiteInformation ::= SEQUENCE

{

geographicalCoordinates [1] GeographicalCoordinates,

azimuth [2] INTEGER (0..359) OPTIONAL,

operatorSpecificInformation [3] UTF8String OPTIONAL

}

-- TS 29.518 [22], clause 6.4.6.2.6

LocationInfo ::= SEQUENCE

{

userLocation [1] UserLocation OPTIONAL,

currentLoc [2] BOOLEAN OPTIONAL,

geoInfo [3] GeographicArea OPTIONAL,

rATType [4] RATType OPTIONAL,

timeZone [5] TimeZone OPTIONAL,

additionalCellIDs [6] SEQUENCE OF CellInformation OPTIONAL

}

-- TS 29.571 [17], clause 5.4.4.7

UserLocation ::= SEQUENCE

{

eUTRALocation [1] EUTRALocation OPTIONAL,

nRLocation [2] NRLocation OPTIONAL,

n3GALocation [3] N3GALocation OPTIONAL

}

-- TS 29.571 [17], clause 5.4.4.8

EUTRALocation ::= SEQUENCE

{

tAI [1] TAI,

eCGI [2] ECGI,

ageOfLocatonInfo [3] INTEGER OPTIONAL,

uELocationTimestamp [4] Timestamp OPTIONAL,

geographicalInformation [5] UTF8String OPTIONAL,

geodeticInformation [6] UTF8String OPTIONAL,

globalNGENbID [7] GlobalRANNodeID OPTIONAL,

cellSiteInformation [8] CellSiteInformation OPTIONAL

}

-- TS 29.571 [17], clause 5.4.4.9

NRLocation ::= SEQUENCE

{

tAI [1] TAI,

nCGI [2] NCGI,

ageOfLocatonInfo [3] INTEGER OPTIONAL,

uELocationTimestamp [4] Timestamp OPTIONAL,

geographicalInformation [5] UTF8String OPTIONAL,

geodeticInformation [6] UTF8String OPTIONAL,

globalGNbID [7] GlobalRANNodeID OPTIONAL,

cellSiteInformation [8] CellSiteInformation OPTIONAL

}

-- TS 29.571 [17], clause 5.4.4.10

N3GALocation ::= SEQUENCE

{

tAI [1] TAI OPTIONAL,

n3IWFID [2] N3IWFIDNGAP OPTIONAL,

uEIPAddr [3] IPAddr OPTIONAL,

portNumber [4] INTEGER OPTIONAL

}

-- TS 38.413 [23], clause 9.3.2.4

IPAddr ::= SEQUENCE

{

iPv4Addr [1] IPv4Address OPTIONAL,

iPv6Addr [2] IPv6Address OPTIONAL

}

-- TS 29.571 [17], clause 5.4.4.28

GlobalRANNodeID ::= SEQUENCE

{

pLMNID [1] PLMNID,

aNNodeID [2] ANNodeID

}

ANNodeID ::= CHOICE

{

n3IWFID [1] N3IWFIDSBI,

gNbID [2] GNbID,

nGENbID [3] NGENbID

}

-- TS 38.413 [23], clause 9.3.1.6

GNbID ::= BIT STRING(SIZE(22..32))

-- TS 29.571 [17], clause 5.4.4.4

TAI ::= SEQUENCE

{

pLMNID [1] PLMNID,

tAC [2] TAC

}

-- TS 29.571 [17], clause 5.4.4.5

ECGI ::= SEQUENCE

{

pLMNID [1] PLMNID,

eUTRACellID [2] EUTRACellID

}

-- TS 29.571 [17], clause 5.4.4.6

NCGI ::= SEQUENCE

{

pLMNID [1] PLMNID,

nRCellID [2] NRCellID

}

RANCGI ::= CHOICE

{

eCGI [1] ECGI,

nCGI [2] NCGI

}

CellInformation ::= SEQUENCE

{

rANCGI [1] RANCGI,

cellSiteinformation [2] CellSiteInformation OPTIONAL,

timeOfLocation [3] Timestamp OPTIONAL

}

-- TS 38.413 [23], clause 9.3.1.57

N3IWFIDNGAP ::= BIT STRING (SIZE(16))

-- TS 29.571 [17], clause 5.4.4.28

N3IWFIDSBI ::= UTF8String

-- TS 29.571 [17], table 5.4.2-1

TAC ::= OCTET STRING (SIZE(2..3))

-- TS 38.413 [23], clause 9.3.1.9

EUTRACellID ::= BIT STRING (SIZE(28))

-- TS 38.413 [23], clause 9.3.1.7

NRCellID ::= BIT STRING (SIZE(36))

-- TS 38.413 [23], clause 9.3.1.8

NGENbID ::= CHOICE

{

macroNGENbID [1] BIT STRING (SIZE(20)),

shortMacroNGENbID [2] BIT STRING (SIZE(18)),

longMacroNGENbID [3] BIT STRING (SIZE(21))

}

-- TS 29.518 [22], clause 6.4.6.2.3

PositioningInfo ::= SEQUENCE

{

positionInfo [1] LocationData OPTIONAL,

rawMLPResponse [2] RawMLPResponse OPTIONAL

}

RawMLPResponse ::= CHOICE

{

-- The following parameter contains a copy of unparsed XML code of the

-- MLP response message, i.e. the entire XML document containing

-- a <slia> (described in OMA-TS-MLP-V3\_5-20181211-C [20], clause 5.2.3.2.2) or

-- a <slirep> (described in OMA-TS-MLP-V3\_5-20181211-C [20], clause 5.2.3.2.3) MLP message.

mLPPositionData [1] UTF8String,

-- OMA MLP result id, defined in OMA-TS-MLP-V3\_5-20181211-C [20], Clause 5.4

mLPErrorCode [2] INTEGER (1..699)

}

-- TS 29.572 [24], clause 6.1.6.2.3

LocationData ::= SEQUENCE

{

locationEstimate [1] GeographicArea,

accuracyFulfilmentIndicator [2] AccuracyFulfilmentIndicator OPTIONAL,

ageOfLocationEstimate [3] AgeOfLocationEstimate OPTIONAL,

velocityEstimate [4] VelocityEstimate OPTIONAL,

civicAddress [5] CivicAddress OPTIONAL,

positioningDataList [6] SET OF PositioningMethodAndUsage OPTIONAL,

gNSSPositioningDataList [7] SET OF GNSSPositioningMethodAndUsage OPTIONAL,

eCGI [8] ECGI OPTIONAL,

nCGI [9] NCGI OPTIONAL,

altitude [10] Altitude OPTIONAL,

barometricPressure [11] BarometricPressure OPTIONAL

}

-- TS 29.518 [22], clause 6.2.6.2.5

LocationPresenceReport ::= SEQUENCE

{

type [1] AMFEventType,

timestamp [2] Timestamp,

areaList [3] SET OF AMFEventArea OPTIONAL,

timeZone [4] TimeZone OPTIONAL,

accessTypes [5] SET OF AccessType OPTIONAL,

rMInfoList [6] SET OF RMInfo OPTIONAL,

cMInfoList [7] SET OF CMInfo OPTIONAL,

reachability [8] UEReachability OPTIONAL,

location [9] UserLocation OPTIONAL,

additionalCellIDs [10] SEQUENCE OF CellInformation OPTIONAL

}

-- TS 29.518 [22], clause 6.2.6.3.3

AMFEventType ::= ENUMERATED

{

locationReport(1),

presenceInAOIReport(2)

}

-- TS 29.518 [22], clause 6.2.6.2.16

AMFEventArea ::= SEQUENCE

{

presenceInfo [1] PresenceInfo OPTIONAL,

lADNInfo [2] LADNInfo OPTIONAL

}

-- TS 29.571 [17], clause 5.4.4.27

PresenceInfo ::= SEQUENCE

{

presenceState [1] PresenceState OPTIONAL,

trackingAreaList [2] SET OF TAI OPTIONAL,

eCGIList [3] SET OF ECGI OPTIONAL,

nCGIList [4] SET OF NCGI OPTIONAL,

globalRANNodeIDList [5] SET OF GlobalRANNodeID OPTIONAL

}

-- TS 29.518 [22], clause 6.2.6.2.17

LADNInfo ::= SEQUENCE

{

lADN [1] UTF8String,

presence [2] PresenceState OPTIONAL

}

-- TS 29.571 [17], clause 5.4.3.20

PresenceState ::= ENUMERATED

{

inArea(1),

outOfArea(2),

unknown(3),

inactive(4)

}

-- TS 29.518 [22], clause 6.2.6.2.8

RMInfo ::= SEQUENCE

{

rMState [1] RMState,

accessType [2] AccessType

}

-- TS 29.518 [22], clause 6.2.6.2.9

CMInfo ::= SEQUENCE

{

cMState [1] CMState,

accessType [2] AccessType

}

-- TS 29.518 [22], clause 6.2.6.3.7

UEReachability ::= ENUMERATED

{

unreachable(1),

reachable(2),

regulatoryOnly(3)

}

-- TS 29.518 [22], clause 6.2.6.3.9

RMState ::= ENUMERATED

{

registered(1),

deregistered(2)

}

-- TS 29.518 [22], clause 6.2.6.3.10

CMState ::= ENUMERATED

{

idle(1),

connected(2)

}

-- TS 29.572 [24], clause 6.1.6.2.5

GeographicArea ::= CHOICE

{

point [1] Point,

pointUncertaintyCircle [2] PointUncertaintyCircle,

pointUncertaintyEllipse [3] PointUncertaintyEllipse,

polygon [4] Polygon,

pointAltitude [5] PointAltitude,

pointAltitudeUncertainty [6] PointAltitudeUncertainty,

ellipsoidArc [7] EllipsoidArc

}

-- TS 29.572 [24], clause 6.1.6.3.12

AccuracyFulfilmentIndicator ::= ENUMERATED

{

requestedAccuracyFulfilled(1),

requestedAccuracyNotFulfilled(2)

}

-- TS 29.572 [24], clause

VelocityEstimate ::= CHOICE

{

horVelocity [1] HorizontalVelocity,

horWithVertVelocity [2] HorizontalWithVerticalVelocity,

horVelocityWithUncertainty [3] HorizontalVelocityWithUncertainty,

horWithVertVelocityAndUncertainty [4] HorizontalWithVerticalVelocityAndUncertainty

}

-- TS 29.572 [24], clause 6.1.6.2.14

CivicAddress ::= SEQUENCE

{

country [1] UTF8String,

a1 [2] UTF8String OPTIONAL,

a2 [3] UTF8String OPTIONAL,

a3 [4] UTF8String OPTIONAL,

a4 [5] UTF8String OPTIONAL,

a5 [6] UTF8String OPTIONAL,

a6 [7] UTF8String OPTIONAL,

prd [8] UTF8String OPTIONAL,

pod [9] UTF8String OPTIONAL,

sts [10] UTF8String OPTIONAL,

hno [11] UTF8String OPTIONAL,

hns [12] UTF8String OPTIONAL,

lmk [13] UTF8String OPTIONAL,

loc [14] UTF8String OPTIONAL,

nam [15] UTF8String OPTIONAL,

pc [16] UTF8String OPTIONAL,

bld [17] UTF8String OPTIONAL,

unit [18] UTF8String OPTIONAL,

flr [19] UTF8String OPTIONAL,

room [20] UTF8String OPTIONAL,

plc [21] UTF8String OPTIONAL,

pcn [22] UTF8String OPTIONAL,

pobox [23] UTF8String OPTIONAL,

addcode [24] UTF8String OPTIONAL,

seat [25] UTF8String OPTIONAL,

rd [26] UTF8String OPTIONAL,

rdsec [27] UTF8String OPTIONAL,

rdbr [28] UTF8String OPTIONAL,

rdsubbr [29] UTF8String OPTIONAL

}

-- TS 29.572 [24], clause 6.1.6.2.15

PositioningMethodAndUsage ::= SEQUENCE

{

method [1] PositioningMethod,

mode [2] PositioningMode,

usage [3] Usage

}

-- TS 29.572 [24], clause 6.1.6.2.16

GNSSPositioningMethodAndUsage ::= SEQUENCE

{

mode [1] PositioningMode,

gNSS [2] GNSSID,

usage [3] Usage

}

-- TS 29.572 [24], clause 6.1.6.2.6

Point ::= SEQUENCE

{

geographicalCoordinates [1] GeographicalCoordinates

}

-- TS 29.572 [24], clause 6.1.6.2.7

PointUncertaintyCircle ::= SEQUENCE

{

geographicalCoordinates [1] GeographicalCoordinates,

uncertainty [2] Uncertainty

}

-- TS 29.572 [24], clause 6.1.6.2.8

PointUncertaintyEllipse ::= SEQUENCE

{

geographicalCoordinates [1] GeographicalCoordinates,

uncertainty [2] UncertaintyEllipse,

confidence [3] Confidence

}

-- TS 29.572 [24], clause 6.1.6.2.9

Polygon ::= SEQUENCE

{

pointList [1] SET SIZE (3..15) OF GeographicalCoordinates

}

-- TS 29.572 [24], clause 6.1.6.2.10

PointAltitude ::= SEQUENCE

{

point [1] GeographicalCoordinates,

altitude [2] Altitude

}

-- TS 29.572 [24], clause 6.1.6.2.11

PointAltitudeUncertainty ::= SEQUENCE

{

point [1] GeographicalCoordinates,

altitude [2] Altitude,

uncertaintyEllipse [3] UncertaintyEllipse,

uncertaintyAltitude [4] Uncertainty,

confidence [5] Confidence

}

-- TS 29.572 [24], clause 6.1.6.2.12

EllipsoidArc ::= SEQUENCE

{

point [1] GeographicalCoordinates,

innerRadius [2] InnerRadius,

uncertaintyRadius [3] Uncertainty,

offsetAngle [4] Angle,

includedAngle [5] Angle,

confidence [6] Confidence

}

-- TS 29.572 [24], clause 6.1.6.2.4

GeographicalCoordinates ::= SEQUENCE

{

latitude [1] UTF8String,

longitude [2] UTF8String,

mapDatumInformation [3] OGCURN OPTIONAL

}

-- TS 29.572 [24], clause 6.1.6.2.22

UncertaintyEllipse ::= SEQUENCE

{

semiMajor [1] Uncertainty,

semiMinor [2] Uncertainty,

orientationMajor [3] Orientation

}

-- TS 29.572 [24], clause 6.1.6.2.18

HorizontalVelocity ::= SEQUENCE

{

hSpeed [1] HorizontalSpeed,

bearing [2] Angle

}

-- TS 29.572 [24], clause 6.1.6.2.19

HorizontalWithVerticalVelocity ::= SEQUENCE

{

hSpeed [1] HorizontalSpeed,

bearing [2] Angle,

vSpeed [3] VerticalSpeed,

vDirection [4] VerticalDirection

}

-- TS 29.572 [24], clause 6.1.6.2.20

HorizontalVelocityWithUncertainty ::= SEQUENCE

{

hSpeed [1] HorizontalSpeed,

bearing [2] Angle,

uncertainty [3] SpeedUncertainty

}

-- TS 29.572 [24], clause 6.1.6.2.21

HorizontalWithVerticalVelocityAndUncertainty ::= SEQUENCE

{

hspeed [1] HorizontalSpeed,

bearing [2] Angle,

vSpeed [3] VerticalSpeed,

vDirection [4] VerticalDirection,

hUncertainty [5] SpeedUncertainty,

vUncertainty [6] SpeedUncertainty

}

-- The following types are described in TS 29.572 [24], table 6.1.6.3.2-1

Altitude ::= UTF8String

Angle ::= INTEGER (0..360)

Uncertainty ::= INTEGER (0..127)

Orientation ::= INTEGER (0..180)

Confidence ::= INTEGER (0..100)

InnerRadius ::= INTEGER (0..65535)

AgeOfLocationEstimate ::= INTEGER (0..32767)

HorizontalSpeed ::= UTF8String

VerticalSpeed ::= UTF8String

SpeedUncertainty ::= UTF8String

BarometricPressure ::= INTEGER (30000..155000)

-- TS 29.572 [24], clause 6.1.6.3.13

VerticalDirection ::= ENUMERATED

{

upward(1),

downward(2)

}

-- TS 29.572 [24], clause 6.1.6.3.6

PositioningMethod ::= ENUMERATED

{

cellID(1),

eCID(2),

oTDOA(3),

barometricPresure(4),

wLAN(5),

bluetooth(6),

mBS(7)

}

-- TS 29.572 [24], clause 6.1.6.3.7

PositioningMode ::= ENUMERATED

{

uEBased(1),

uEAssisted(2),

conventional(3)

}

-- TS 29.572 [24], clause 6.1.6.3.8

GNSSID ::= ENUMERATED

{

gPS(1),

galileo(2),

sBAS(3),

modernizedGPS(4),

qZSS(5),

gLONASS(6)

}

-- TS 29.572 [24], clause 6.1.6.3.9

Usage ::= ENUMERATED

{

unsuccess(1),

successResultsNotUsed(2),

successResultsUsedToVerifyLocation(3),

successResultsUsedToGenerateLocation(4),

successMethodNotDetermined(5)

}

-- TS 29.571 [17], table 5.2.2-1

TimeZone ::= UTF8String

-- Open Geospatial Consortium URN [35]

OGCURN ::= UTF8String

END

**\*\*\* Start of Fourth MODIFICATION \*\*\***

Annex C (normative): XSD Schema for LI\_X1 extensions

<?xml version="2.0" encoding="utf-8"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"

xmlns="urn:3GPP:ns:li:3GPPX1Extensions"

targetNamespace="urn:3GPP:ns:li:3GPPX1Extensions"

elementFormDefault="qualified">

<xs:complexType name="X1Extensions">

<xs:sequence>

<xs:element name="Extension" type="X1Extension" minOccurs="1" maxOccurs="unbounded"></xs:element>

</xs:sequence>

</xs:complexType>

<xs:complexType name="PTCLIX1TargetIdentifierExtensions">

<xs:sequence>

<xs:element name="PTCLIX1TargetIdentifier" type="PTCLIX1TargetIdentifier" minOccurs="1" maxOccurs="unbounded"></xs:element>

</xs:sequence>

</xs:complexType>

<xs:complexType name="PTCLIX1TargetIdentifier">

<xs:choice>

<xs:element name="MCPTTID" type="MCPTTID"></xs:element>

<xs:element name="InstanceIdentifierURN" type="InstanceIdentifierURN"></xs:element>

<xs:element name="PTCChatGroupID" type="PTCChatGroupID"></xs:element>

</xs:choice>

</xs:complexType>

<xs:simpleType name="MCPTTID">

<xs:restriction base="xs:anyURI"></xs:restriction>

</xs:simpleType>

<xs:simpleType name="InstanceIdentifierURN">

<xs:restriction base="xs:anyURI"></xs:restriction>

</xs:simpleType>

<xs:simpleType name="PTCChatGroupID">

<xs:restriction base="xs:anyURI"></xs:restriction>

</xs:simpleType>

<xs:complexType name="UPFLIT3TargetIdentifierExtensions">

<xs:sequence>

<xs:element name="UPFLIT3TargetIdentifier" type="UPFLIT3TargetIdentifier" minOccurs="1" maxOccurs="unbounded"></xs:element>

</xs:sequence>

</xs:complexType>

<xs:complexType name="UPFLIT3TargetIdentifier">

<xs:choice>

<xs:element name="FSEID" type="FSEID"></xs:element>

<xs:element name="PDRID" type="xs:unsignedInt"></xs:element>

<xs:element name="QERID" type="xs:unsignedInt"></xs:element>

<xs:element name="NetworkInstance" type="xs:hexBinary"></xs:element>

<xs:element name="GTPTunnelDirection" type="GTPTunnelDirection"></xs:element>

<xs:element name="FTEID" type="FTEID"></xs:element>

</xs:choice>

</xs:complexType>

<xs:complexType name="FSEID">

<xs:sequence>

<xs:element name="SEID" type="xs:unsignedLong"></xs:element>

<xs:element name="IPv4Address" type="IPv4Address" minOccurs="0"></xs:element>

<xs:element name="IPv6Address" type="IPv6Address" minOccurs="0"></xs:element>

</xs:sequence>

</xs:complexType>

<xs:complexType name="FTEID">

<xs:sequence>

<xs:element name="TEID" type="xs:unsignedInt"></xs:element>

<xs:element name="IPv4Address" type="IPv4Address" minOccurs="0"></xs:element>

<xs:element name="IPv6Address" type="IPv6Address" minOccurs="0"></xs:element>

</xs:sequence>

</xs:complexType>

<xs:simpleType name="GTPTunnelDirection">

<xs:restriction base="xs:string">

<xs:enumeration value="Outbound"></xs:enumeration>

<xs:enumeration value="Inbound"></xs:enumeration>

</xs:restriction>

</xs:simpleType>

<xs:complexType name="X1Extension">

<xs:choice>

<xs:element name="LALSLILCSTargetProvisioning" type="LALSLILCSTargetProvisioningExtensions"></xs:element>

<xs:element name="LALSLTFProvisioning" type="LALSLTFProvisioningExtensions"></xs:element>

<xs:element name="HeaderReporting" type="PDHRReportingExtensions"></xs:element>

<xs:element name="ServiceScopingOptions" type="ServiceScopingOptions"></xs:element>

</xs:choice>

</xs:complexType>

<xs:complexType name="LALSLILCSTargetProvisioningExtensions">

<xs:sequence>

<xs:element name="PositioningServiceType" type="PositioningServiceType"></xs:element>

<xs:element name="PositioningPeriodicity" type="PositioningPeriodicity" minOccurs="0"></xs:element>

<xs:element name="PositioningParameters" type="PositioningParameters" minOccurs="0"></xs:element>

</xs:sequence>

</xs:complexType>

<xs:simpleType name="PositioningServiceType">

<xs:restriction base="xs:string">

<xs:enumeration value="Immediate"></xs:enumeration>

<xs:enumeration value="Periodic"></xs:enumeration>

</xs:restriction>

</xs:simpleType>

<xs:simpleType name="PositioningPeriodicity">

<xs:restriction base="xs:nonNegativeInteger">

</xs:restriction>

</xs:simpleType>

<xs:complexType name="PositioningParameters">

<xs:sequence>

<xs:element name="RequestedLocationType" type="RequestedLocationType" minOccurs="0"></xs:element>

<xs:element name="RequestedResponseType" type="RequestedResponseType" minOccurs="0"></xs:element>

<xs:element name="MaxLocationAge" type="xs:nonNegativeInteger" minOccurs="0"></xs:element>

<xs:element name="ResponseTimingRequired" type="ResponseTimingRequired" minOccurs="0"></xs:element>

<xs:element name="ResponseTimer" type="xs:nonNegativeInteger" minOccurs="0"></xs:element>

<xs:element name="HorizontalAccuracy" type="NumberWithQOSClass" minOccurs="0"></xs:element>

<xs:element name="AltitudeAccuracy" type="NumberWithQOSClass" minOccurs="0"></xs:element>

<xs:element name="MotionStateRequest" type="EmptyElement" minOccurs="0"></xs:element>

</xs:sequence>

</xs:complexType>

<xs:simpleType name="RequestedLocationType">

<xs:restriction base="xs:string">

<xs:enumeration value="CURRENT"></xs:enumeration>

<xs:enumeration value="CURRENT\_OR\_LAST"></xs:enumeration>

</xs:restriction>

</xs:simpleType>

<xs:simpleType name="RequestedResponseType">

<xs:restriction base="xs:string">

<xs:enumeration value="SYNC"></xs:enumeration>

<xs:enumeration value="ASYNC"></xs:enumeration>

</xs:restriction>

</xs:simpleType>

<xs:simpleType name="ResponseTimingRequired">

<xs:restriction base="xs:string">

<xs:enumeration value="NO\_DELAY"></xs:enumeration>

<xs:enumeration value="LOW\_DELAY"></xs:enumeration>

<xs:enumeration value="DELAY\_TOL"></xs:enumeration>

</xs:restriction>

</xs:simpleType>

<xs:complexType name="NumberWithQOSClass">

<xs:simpleContent>

<xs:extension base="xs:nonNegativeInteger">

<xs:attribute name="qos\_class" type="QOSClass"></xs:attribute>

</xs:extension>

</xs:simpleContent>

</xs:complexType>

<xs:simpleType name="QOSClass">

<xs:restriction base="xs:string">

<xs:enumeration value="ASSURED"></xs:enumeration>

<xs:enumeration value="BEST\_EFFORT"></xs:enumeration>

</xs:restriction>

</xs:simpleType>

<xs:simpleType name="EmptyElement">

<xs:restriction base="xs:string">

<xs:enumeration value=""></xs:enumeration>

</xs:restriction>

</xs:simpleType>

<xs:complexType name="LALSLTFProvisioningExtensions">

<xs:sequence>

<xs:element name="LILCSClientAddress" type="LILCSClientIPAddress"></xs:element>

<xs:element name="PositioningParameters" type="PositioningParameters" minOccurs="0"></xs:element>

</xs:sequence>

</xs:complexType>

<xs:complexType name="LILCSClientIPAddress">

<xs:sequence>

<xs:choice>

<xs:element name="IPv4Address" type="IPv4Address"/>

<xs:element name="IPv6Address" type="IPv6Address"/>

</xs:choice>

</xs:sequence>

</xs:complexType>

<xs:simpleType name="IPv4Address">

<xs:restriction base="xs:token">

<xs:pattern value="((25[0-5]|2[0-4][0-9]|[01]?[0-9]?[0-9])\.){3}(25[0-5]|2[0-4][0-9]|[01]?[0-9]?[0-9])"/>

</xs:restriction>

</xs:simpleType>

<xs:simpleType name="IPv6Address">

<xs:restriction base="xs:token">

<xs:pattern value="([0-9a-f]{4}:){7}([0-9a-f]{4})"/>

</xs:restriction>

</xs:simpleType>

<xs:complexType name="PDHRReportingExtensions">

<xs:sequence>

<xs:element name="PDHType" type="PDHType"></xs:element>

</xs:sequence>

</xs:complexType>

<xs:complexType name="PDHType">

<xs:choice>

<xs:element name="PDHR" type="EmptyElement"></xs:element>

<xs:element name="PDSR" type="PDSRParameters"></xs:element>

</xs:choice>

</xs:complexType>

<xs:complexType name="PDSRParameters">

<xs:sequence>

<xs:element name="PDSRTriggerType" type="PDSRTriggerType"></xs:element>

</xs:sequence>

</xs:complexType>

<xs:complexType name="PDSRTriggerType">

<xs:choice>

<xs:element name="TimerExpiry" type="TimerExpiryInSeconds"></xs:element>

<xs:element name="PacketCount" type="xs:nonNegativeInteger"></xs:element>

<xs:element name="ByteCount" type="xs:nonNegativeInteger"></xs:element>

</xs:choice>

</xs:complexType>

<xs:simpleType name="TimerExpiryInSeconds">

<xs:restriction base="xs:nonNegativeInteger">

</xs:restriction>

</xs:simpleType>

</xs:schema>

**\*\*\* End of All MODIFICATIONS \*\*\***