ETSI TS 129 573 V18.6.0 (2024-05)

TECHNICAL SPECIFICATION

5G; 5G System; Public Land Mobile Network (PLMN) Interconnection; Stage 3 (3GPP TS 29.573 version 18.6.0 Release 18)

Reference

RTS/TSGC-0429573vi60

Keywords

5G

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - APE 7112B
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° w061004871

Important notice

The present document can be downloaded from:
https://www.etsi.org/standards-search

The present document may be made available in electronic versions and/or in print. The content of any electronic and/or print versions of the present document shall not be modified without the prior written authorization of ETSI. In case of any existing or perceived difference in contents between such versions and/or in print, the prevailing version of an ETSI deliverable is the one made publicly available in PDF format at www.etsi.org/deliver.

Users of the present document should be aware that the document may be subject to revision or change of status. Information on the current status of this and other ETSI documents is available at
https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx

If you find errors in the present document, please send your comment to one of the following services:
https://portal.etsi.org/People/CommiteeSupportStaff.aspx

If you find a security vulnerability in the present document, please report it through our
Coordinated Vulnerability Disclosure Program:
https://www.etsi.org/standards/coordinated-vulnerability-disclosure

Notice of disclaimer & limitation of liability

The information provided in the present deliverable is directed solely to professionals who have the appropriate degree of experience to understand and interpret its content in accordance with generally accepted engineering or other professional standard and applicable regulations.
No recommendation as to products and services or vendors is made or should be implied.
No representation or warranty is made that this deliverable is technically accurate or sufficient or conforms to any law and/or governmental rule and/or regulation and further, no representation or warranty is made of merchantability or fitness for any particular purpose or against infringement of intellectual property rights.
In no event shall ETSI be held liable for loss of profits or any other incidental or consequential damages.

Any software contained in this deliverable is provided "AS IS" with no warranties, express or implied, including but not limited to, the warranties of merchantability, fitness for a particular purpose and non-infringement of intellectual property rights and ETSI shall not be held liable in any event for any damages whatsoever (including, without limitation, damages for loss of profits, business interruption, loss of information, or any other pecuniary loss) arising out of or related to the use of or inability to use the software.

Copyright Notification

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The declarations pertaining to these essential IPRs, if any, are publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (https://ipr.etsi.org/).

Pursuant to the ETSI Directives including the ETSI IPR Policy, no investigation regarding the essentiality of IPRs, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

DECT™, **PLUGTESTS™**, **UMTS™** and the ETSI logo are trademarks of ETSI registered for the benefit of its Members. **3GPP™** and **LTE™** are trademarks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners. **oneM2M™** logo is a trademark of ETSI registered for the benefit of its Members and of the oneM2M Partners. **GSM**® and the GSM logo are trademarks registered and owned by the GSM Association.

Legal Notice

This Technical Specification (TS) has been produced by ETSI 3rd Generation Partnership Project (3GPP).

The present document may refer to technical specifications or reports using their 3GPP identities. These shall be interpreted as being references to the corresponding ETSI deliverables.

The cross reference between 3GPP and ETSI identities can be found under https://webapp.etsi.org/key/queryform.asp.

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

Contents

Foreword

This Technical Specification has been produced by the 3rd Generation Partnership Project (3GPP).

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

x the first digit:

1 presented to TSG for information;

2 presented to TSG for approval;

3 or greater indicates TSG approved document under change control.

y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.

z the third digit is incremented when editorial only changes have been incorporated in the document.

In the present document, modal verbs have the following meanings:

shall indicates a mandatory requirement to do something

shall not indicates an interdiction (prohibition) to do something

The constructions "shall" and "shall not" are confined to the context of normative provisions, and do not appear in Technical Reports.

The constructions "must" and "must not" are not used as substitutes for "shall" and "shall not". Their use is avoided insofar as possible, and they are not used in a normative context except in a direct citation from an external, referenced, non-3GPP document, or so as to maintain continuity of style when extending or modifying the provisions of such a referenced document.

should indicates a recommendation to do something

should not indicates a recommendation not to do something

may indicates permission to do something

need not indicates permission not to do something

The construction "may not" is ambiguous and is not used in normative elements. The unambiguous constructions "might not" or "shall not" are used instead, depending upon the meaning intended.

can indicates that something is possible

cannot indicates that something is impossible

The constructions "can" and "cannot" are not substitutes for "may" and "need not".

will indicates that something is certain or expected to happen as a result of action taken by an agency the behaviour of which is outside the scope of the present document

will not indicates that something is certain or expected not to happen as a result of action taken by an agency the behaviour of which is outside the scope of the present document

might indicates a likelihood that something will happen as a result of action taken by some agency the behaviour of which is outside the scope of the present document

might not	indicates a likelihood that something will not happen as a result of action taken by some agency the behaviour of which is outside the scope of the present document

In addition:

is	(or any other verb in the indicative mood) indicates a statement of fact
is not	(or any other negative verb in the indicative mood) indicates a statement of fact

The constructions "is" and "is not" do not indicate requirements.

1 Scope

The present document specifies the stage 3 protocol and data model for the PLMN and/or SNPN interconnection Interface. It provides stage 3 protocol definitions and message flows, and specifies the APIs for the procedures on the PLMN interconnection interface (i.e N32).

The 5G System stage 2 architecture and procedures are specified in 3GPP TS 23.501 [2] and 3GPP TS 23.502 [3].

The Technical Realization of the Service Based Architecture and the Principles and Guidelines for Services Definition are specified in 3GPP TS 29.500 [4] and 3GPP TS 29.501 [5].

The stage 2 level N32 procedures are specified in 3GPP TS 33.501 [6].

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; Stage 2".

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

[4] 3GPP TS 29.500: "5G System; Technical Realization of Service Based Architecture; Stage 3".

[5] 3GPP TS 29.501: "5G System; Principles and Guidelines for Services Definition; Stage 3".

[6] 3GPP TS 33.501: "Security architecture and procedures for 5G system".

[7] IETF RFC 9113: "HTTP/2".

[8] IETF RFC 8259: "The JavaScript Object Notation (JSON) Data Interchange Format".

[9] IETF RFC 9110: "HTTP Semantics".

[10] Void.

[11] IETF RFC 793: "Transmission Control Protocol".

[12] 3GPP TS 29.571: "5G System; Common Data Types for Service Based Interfaces Stage 3".

[13] IETF RFC 7518: "JSON Web Algorithms (JWA)".

[14] IETF RFC 7516: "JSON Web Encryption (JWE)".

[15] IETF RFC 4648: "The Base16, Base32, and Base64 Data Encodings".

[16] IETF RFC 7515: "JSON Web Signature (JWS)".

[17] IETF RFC 6901: "JavaScript Object Notation (JSON) Pointer".

[18] 3GPP TS 29.510: "Network Function Repository Services; Stage 3".

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

[20] 3GPP TR 21.900: "Technical Specification Group working methods".

[21] IETF RFC 7468: "Textual Encodings of PKIX, PKCS, and CMS Structures".

[22] IETF RFC 9457: "Problem Details for HTTP APIs".

[23] IETF RFC 1952: "GZIP file format specification version 4.3".

[24] Void

[25] 3GPP TS 29.518: "5G System; Access and Mobility Management Service; Stage 3".

[26] 3GPP TS 29.503: "5G System; Unified Data Management Services; Stage 3".

[27] OpenAPI: "OpenAPI Specification Version 3.0.0", https://spec.openapis.org/oas/v3.0.0.

[28] 3GPP TS 22.261: "Service requirements for the 5G system; Stage 1".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in 3GPP TR 21.905 [1] and the following apply. A term defined in the present document takes precedence over the definition of the same term, if any, in 3GPP TR 21.905 [1].

c-SEPP: The SEPP that is present on the NF service consumer side is called the c-SEPP.

p-SEPP: The SEPP that is present on the NF service producer side is called the p-SEPP.

NOTE: For the purpose of N32-c procedures, the two interacting SEPPs are called "initiating" SEPP and "responding" SEPP. The c-SEPP and p-SEPP terminology is not used in this specification though it is used in 3GPP TS 33.501 [6].

c-IPX: The IPX on the NF service consumer side.

p-IPX: The IPX of the NF service producer side.

N32-c context: This context is set up at the SEPP after the Security Capability Exchange procedure is finalized. It defines the security capability that is mutually agreed and effective for both the cSEPP and the pSEPP.

Roaming Hub: A type of Roaming Intermediary; as defined by GSMA (see clause 3.1 of 3GPP TS 33.501 [6]).

Roaming Intermediary: an entity that provides roaming related services (see clause 3.1 of 3GPP TS 33.501 [6]).

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in 3GPP TR 21.905 [1] and the following apply. An abbreviation defined in the present document takes precedence over the definition of the same abbreviation, if any, in 3GPP TR 21.905 [1].

GZIP	GNU ZIP
IPX	IP Exchange Service
JOSE	Javascript Object Signing and Encryption
JWE	JSON Web Encryption
JWS	JSON Web Signature
PRINS	PRotocol for N32 INterconnect Security
SEPP	Security and Edge Protection Proxy
TLS	Transport Layer Security
UPU	UE Parameters Update

4 General Description

4.1 Introduction

This clause provides a general description of the interconnect interfaces used between the PLMNs and/or SNPNs for transporting the service based interface message exchanges.

4.2 N32 Interface

4.2.1 General

The N32 interface is used between SEPPs of different PLMNs for both roaming and PLMN interconnect scenarios.

The N32 interface may also be used between SEPPs from an SNPN and another SNPN or PLMN, for SNPN interconnect scenarios (e.g. for SNPN connectivity with a Credentials Holder network, see clause 5.30.2.9.3 of 3GPP TS 23.501 [2]). Unless specified otherwise, references to "PLMN" throughout this specification shall be substituted by "SNPN" for a SEPP that is deployed in an SNPN.

The SEPP that is on the NF service consumer side is called the c-SEPP and the SEPP that is on the NF service producer is called the p-SEPP. The NF service consumer or SCP may be configured with the c-SEPP or discover the c-SEPP by querying the NRF. The NF service producer or SCP may be configured with the p-SEPP or discover the p-SEPP by querying the NRF.

The N32 interface can be logically considered as 2 separate interfaces as given below.

- N32-c, a control plane interface between the SEPPs for performing initial handshake and negotiating the parameters to be applied for the actual N32 message forwarding.

- N32-f, a forwarding interface between the SEPPs which is used for forwarding the communication between the NF service consumer and the NF service producer after applying application level security protection or TLS security protection.

4.2.2 N32-c Interface

The following figure shows the scope of the N32-c interface.

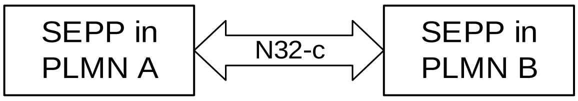

Figure 4.2.2-1: N32-c Interface

The N32-c interface provides the following functionalities:

- Initial handshake procedure between the SEPP in PLMN A (called the initiating SEPP) and the SEPP in PLMN B (called the responding SEPP), that involves capability negotiation and parameter exchange as specified in 3GPP TS 33.501 [6].

4.2.3 N32-f Interface

The following figures shows the scope of the N32-f interface.

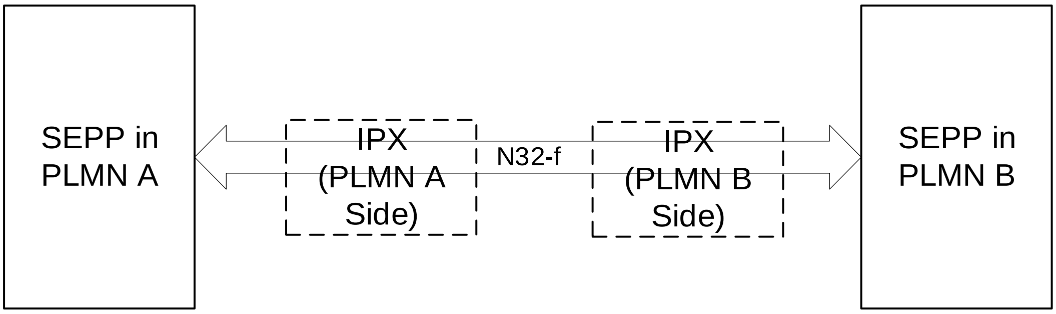

Figure 4.2.3-1a: N32-f Interface with TLS security

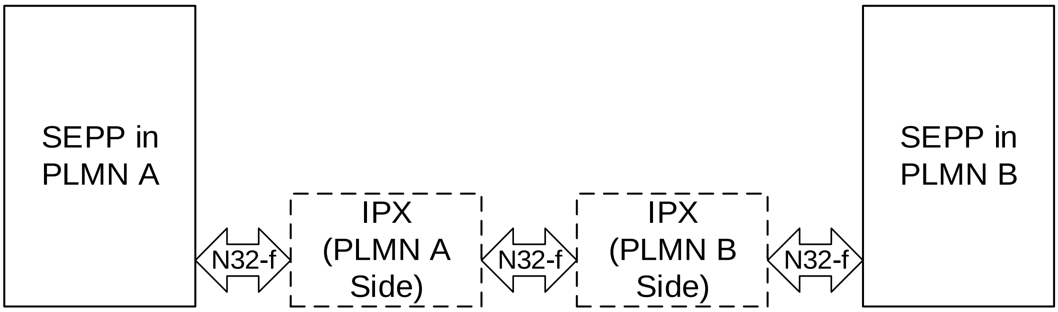

Figure 4.2.3-1b: N32-f Interface with PRINS

The N32-f interface shall be used to forward the HTTP/2 messages of the NF service producers and the NF service consumers in different PLMN, through the SEPPs of the respective PLMN.

If TLS is the negotiated security policy between the SEPP, then the N32-f shall involve only the forwarding of the HTTP/2 messages of the NF service producers and the NF service consumers without any reformatting at the SEPPs and/or the IPXs (see figure 4.2.3-1a).

The application layer security protection functionality of the N32-f is used only if the PRotocol for N32 INterconnect Security (PRINS) is negotiated between the SEPPs using N32-c (see figure 4.2.3-1b).

The N32-f interface provides the following application layer security protection functionalities:

- Message protection of the information exchanged between the NF service consumer and the NF service producer across PLMNs by applying application layer security mechanisms as specified in 3GPP TS 33.501 [6].
- Forwarding of the application layer protected message from a SEPP in one PLMN to a SEPP in another PLMN. Such forwarding may involve IPX providers on path.
- If IPX providers are on the path from SEPP in PLMN A to SEPP in PLMN B, the forwarding on the N32-f interface may involve the insertion of content modification instructions which the receiving SEPP applies after verifying the integrity of such modification instructions.

4.3 Protocol Stack

4.3.1 General

The protocol stack for the N32 interface is shown below in Figure 4.3.1-1.

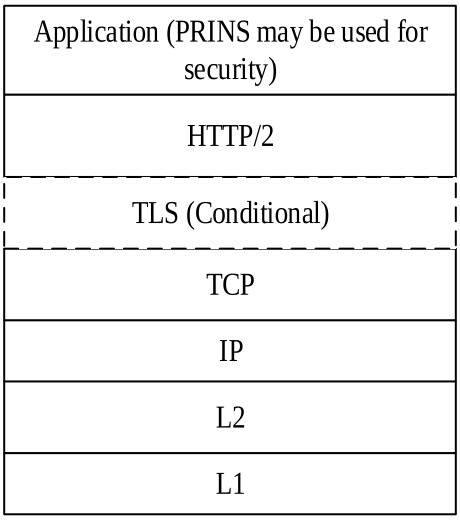

Figure 4.3.1-1: N32 Protocol Stack

The N32 interfaces (N32-c and N32-f) use HTTP/2 protocol (see clause 4.2.2 and 4.2.3, respectively) with JSON (see clause 4.2.4) as the application layer serialization protocol. For the security protection at the transport layer, the SEPPs shall support TLS as specified in clause 13.1.2 of 3GPP TS 33.501 [6].

For the N32-f interface, the application layer (i.e the JSON content) encapsulates the complete HTTP/2 message between the NF service consumer and the NF service producer, by transforming the HTTP/2 headers and the body into specific JSON attributes as specified in clause 6.2. For the scenarios when there are IPX entities between SEPPs, see clause 4.3.2 for TLS/PRINS usage.

4.3.2 HTTP/2 Protocol

4.3.2.1 General

HTTP/2 as described in IETF RFC 9113 [7] shall be used for N32 interface.

4.3.2.2 HTTP standard headers

The HTTP request standard headers and the HTTP response standard headers that shall be supported on the N32 interface are defined in Table 4.2.2.2-1 and in Table 4.2.2.2-2 respectively.

Table 4.3.2.2-1: Mandatory to support HTTP request standard headers

Name	Reference	Description
Accept	IETF RFC 9110 [9]	This header is used to specify response media types that are acceptable.
Accept-Encoding	IETF RFC 9110 [9]	This header may be used to indicate what response content-encodings (e.g gzip) are acceptable in the response.
Content-Length	IETF RFC 9110 [9]	This header is used to provide the anticipated size, as a decimal number of octets, for a potential content.
Content-Type	IETF RFC 9110 [9]	This header is used to indicate the media type of the associated representation.
Via	IETF RFC 9110 [9]	This header is used to indicate the intermediate proxies in the service request path. Please refer to clause 6.10.8 of 3GPP TS 29.500 [4] for encoding of the via header

Table 4.3.2.2-2: Mandatory to support HTTP response standard headers

Name	Reference	Description
Content-Length	IETF RFC 9110 [9]	This header may be used to provide the anticipated size, as a decimal number of octets, for a potential content.
Content-Type	IETF RFC 9110 [9]	This header shall be used to indicate the media type of the associated representation.
Content-Encoding	IETF RFC 9110 [9]	This header may be used in some responses to indicate to the HTTP/2 client the content encodings (e.g gzip) applied to the response body beyond those inherent in the media type.
Via	IETF RFC 9110 [9]	This header is used to indicate the intermediate proxies in the service response path. Please refer to clause 6.10.8 of 3GPP TS 29.500 [4] for encoding of the via header.
Server	IETF RFC 9110 [9]	This header is used to indicate the originator of an HTTP error response.

4.3.2.3 HTTP custom headers

The HTTP custom headers specified in clause 5.2.3 of 3GPP TS 29.500 [4] shall be supported on the N32 interface.

4.3.2.4 HTTP/2 connection management

Each SEPP initiates HTTP/2 connections towards its peer SEPP for the following purposes

- N32-c interface
- N32-f interface

The scope of the HTTP/2 connection used for the N32-c interface is short-lived. Once the initial handshake is completed the connection is torn down as specified in 3GPP TS 33.501 [6]. The HTTP/2 connection used for N32-c is end to end between the SEPPs and does not involve an IPX to intercept the HTTP/2 connection, though an IPX may be involved for IP level routing.

The scope of the HTTP/2 connection used for the N32-f interface is long-lived. The N32-f HTTP/2 connection at a SEPP can be:

- Case A: Towards a SEPP of another PLMN without involving any IPX intermediaries or involving IPX intermediaries where IPX does not require modification or observation of the information; or
- Case B: Towards a SEPP of another PLMN via IPX where IPX requires modification or observation of the information. In this case, the HTTP/2 connection from a SEPP terminates at the next hop IPX with the IPX acting as a HTTP proxy.

For the N32-f interface the HTTP/2 connection management requirements specified in clause 5.2.6 of 3GPP TS 29.500 [4] shall be applicable. The URI scheme used for the N32-f JOSE protected message forwarding API shall be "http". If confidentiality protection of all IEs for the N32-f JOSE protected message forwarding procedure is required, then:

- For case A, the security between the SEPPs shall be ensured by means of an IPSec or TLS connection;
- For case B, hop-by-hop security between the SEPP and the IPXs should be established on N32-f. This hop-by-hop security shall be established using an IPSec or TLS connection.

4.3.3 Transport Protocol

The Transmission Control Protocol as described in IETF RFC 793 [11] shall be used as transport protocol as required by HTTP/2 (see IETF RFC 9113 [7]).

When there is no IPX between the SEPPs or IPX(s) are offering only IP routing service without modification or observation of the content, TLS shall be used for security protection (see clause 13.1.2 of 3GPP TS 33.501 [6]). When there is IPX between the SEPPs and IPX requires modification or observation of the content, TLS or NDS/IP should be used for security protection as specified in clause 13.1.2 of 3GPP TS 33.501 [6].

NOTE: When using TCP as the transport protocol, an HTTP/2 connection is mapped to a TCP connection.

4.3.4 Serialization Protocol

The JavaScript Object Notation (JSON) format as described in IETF RFC 8259 [8] shall be used as the serialization protocol.

5 N32 Procedures

5.1 Introduction

The procedures on the N32 interface are split into two categories:

- Procedures that happen end to end between the SEPPs on the N32-c interface;
- Procedures that are used for the forwarding of messages on the service based interface between the NF service consumer and the NF service producer via the SEPP across the N32-f interface.

Table 5.1-1 summarizes the corresponding APIs defined for this specification.

Table 5.1-1: API Descriptions

Service Name	Clause	Description	OpenAPI Specification File	apiName	Annex
N32 Handshake	6.1	N32-c Handshake Service	TS29573_N32_Handshake.yaml	n32c-handshake	A.2
JOSE Protected Message Forwarding	6.2	N32-f Message Forwarding Service	TS29573_JOSEProtectedMessageForwarding.yaml	n32f-forward	A.3
Nsepp_Telescopic_FQDN_Mapping	6.3	SEPP Telescopic FQDN Mapping	TS29573_SeppTelescopicFqdnMapping.yaml	nsepp-telescopic	A.4

5.2 N32 Handshake Procedures (N32-c)

5.2.1 General

The N32 handshake procedure is used between the SEPPs in two PLMNs to mutually authenticate each other and negotiate the security mechanism to use over N32-f along with associated security configuration parameters.

A HTTP/2 connection shall be established between the initiating SEPP and the responding SEPP end to end over TLS. The following N32 handshake procedures are specified in the clauses below.

- Security Capability Negotiation Procedure
- Parameter Exchange Procedure
- N32-f Context Termination Procedure
- N32-f Error Reporting Procedure

5.2.2 Security Capability Negotiation Procedure

The initiating SEPP shall initiate a Security Capability Negotiation procedure towards the responding SEPP to agree on a security mechanism to use for protecting NF service related signalling over N32-f. An end to end TLS connection shall be setup between the SEPPs before the initiation of this procedure. This procedure may also be used to tear down the N32-f TLS connection if the remote SEPP indicated support of the feature NFTLST during the setup of the N32-c connection. The procedure is described in Figure 5.2.2-1 below.

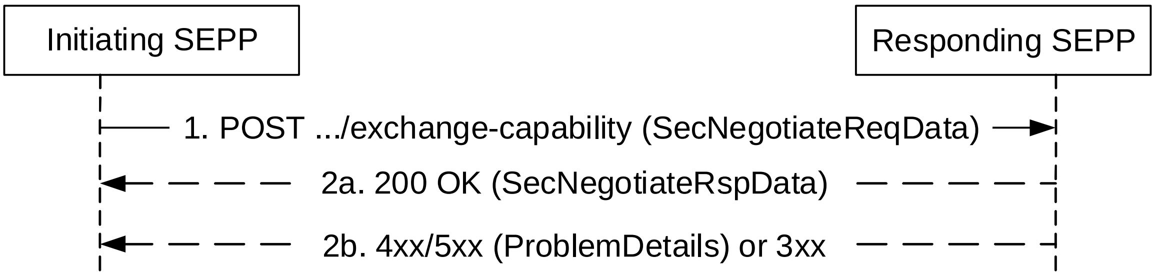
Initiating SEPP
Responding SEPP
1. POST .../exchange-capability (SecNegotiateReqData)
2a. 200 OK (SecNegotiateRspData)
2b. 4xx/5xx (ProblemDetails) or 3xx

Figure 5.2.2-1: Security Capability Negotiation Procedure

1. The initiating SEPP issues a HTTP POST request towards the responding SEPP with the request body containing the "SecNegotiateReqData" IE carrying the following information:

   - Supported security capabilities (i.e., PRINS and/or TLS);
   - Whether the 3gpp-Sbi-Target-apiRoot HTTP header is supported, if TLS security is supported;
   - Sender PLMN ID(s) or SNPN ID(s);
   - Target PLMN ID or SNPN ID;
   - Purpose of the intended usage of N32 connection.
   - The senderN32fFqdn IE, if the initiating SEPP wishes the responding SEPP to establish the N32-f connection towards a specific FQDN (of the initiating SEPP).
   - The senderN32fPortList IE, if the initiating SEPP wishes the responding SEPP to establish the N32-f connection using a specific port number. When present, the list shall contain one port number per supported security capability (i.e., PRINS and/or TLS).

If different PLMNs or SNPNs are represented by different PLMN IDs or SNPN IDs (respectively) supported by a SEPP, then the SEPP shall use separate N32-connections for each pair of local and remote PLMN or SNPN. Both SEPPs shall store the mapping between the N32 connections and their pair of PLMN IDs or SNPN IDs.

NOTE 1: If SEPPs support separate FQDN per PLMN or SNPN, then Target PLMN Id or Target SNPN Id is not required as target PLMN or SNPN can be selected by the FQDN.

To tear down the N32-f connection when negotiated security scheme is TLS, the "SecNegotiateReqData" IE shall contain:

- Supported security capability set to "NONE"

2a. On successful processing of the request, the responding SEPP shall respond to the initiating SEPP with a "200 OK" status code and a POST response body that contains "SecNegotiateRspData" IE carrying the following information:

   - Selected security capability (i.e., PRINS or TLS);
   - Whether the 3gpp-Sbi-Target-apiRoot HTTP header is supported, if TLS security is selected;
   - Sender PLMN ID(s) or SNPN ID(s).
   - Purpose of the accepted usage of N32 connection.
   - The senderN32fFqdn IE, if the responding SEPP wishes the initiating SEPP to establish the N32-f connection towards a specific FQDN (of the responding SEPP).
   - The senderN32fPort IE, if the responding SEPP wishes the initiating SEPP to establish the N32-f connection using a specific port number.

NOTE 2: Same SEPP endpoints can serve all accepted purposes over the same N32-f connection established as the result of request/response messages.

The responding SEPP compares the initiating SEPP's supported security capabilities to its own supported security capabilities and selects, based on its local policy, a security mechanism, which is supported by both the SEPPs. If the selected security capability indicates any other capability other than PRINS, then the HTTP/2 connection initiated between the two SEPPs for the N32 handshake procedures shall be terminated. The negotiated security capability shall be applicable on both the directions. If the selected security capability is PRINS, then the two SEPPs may decide to create (if not available) / maintain HTTP/2 connection(s) where each SEPP acts as a client towards the other (which acts as a server). This may be used for later signalling of N32-f error reporting procedure (see clause 5.2.5) and N32-f context termination procedure (see clause 5.2.4).

If different PLMNs or SNPNs are represented by different PLMN IDs or SNPN IDs (respectively) supported by a SEPP, then the SEPP shall use separate N32-connections for each pair of local and remote PLMN or SNPN. Both SEPPs shall store the mapping between the N32 connections and their pair of PLMN IDs or SNPN IDs.

The SEPP shall select the PLMN or SNPN from the list of supported PLMN(s) or SNPN(s) based on the received Target PLMN ID or SNPN ID, or based on PLMN or SNPN specific FQDN used in the request, and provide the selected PLMN's PLMN Id(s) in the plmnIdList or the selected SNPN's SNPN Id(s) in the snpnIdList.

In case no purposes are exchanged, the receiving SEPP shall assume by default that purposes are for Roaming and inter-PLMN mobility as described in clause 6.1.5.3.9.

The initiating SEPP and/or responding SEPP may enable the establishment of an N32 connection for the purpose of Disaster Roaming only during disaster conditions.

When the request is for tearing down the existing N32-f TLS connection, the "SecNegotiateRspData" IE shall contain:

- Supported security capability set to "NONE"

and, subsequently, both SEPP shall terminate the N32-c and N32-f TLS connection.

If the initiating SEPP receives the senderN32fFqdn IE and/or the senderN32fPort IE from the responding SEPP, the initiating SEPP shall establish the N32-f connection towards the responding SEPP using the received N32-f FQDN and/or the senderN32fPort IE.

If the responding SEPP receives the senderN32fFqdn IE and/or the senderN32fPortList IE from the initiating SEPP, the responding SEPP shall establish the N32-f connection towards the initiating SEPP using the received N32-f FQDN and/or the N32-f port number received in the senderN32fPortList IE corresponding to the selected security capability (i.e., TLS or PRINS).

If the N32-f context exists between the peer SEPPs, and the N32 exchange capability request is not for tearing down the N32-f connections, the responding SEPP shall:

- stop sending any further messages over the N32-f towards the initiating SEPP;
- delete the current N32-f context and terminate any N32-f connection with the initiating SEPP; and
- process the received exchange capability request.

2b. On failure or redirection, the responding SEPP shall respond to the initiating SEPP with an appropriate status code as specified in clause 6.1.4.2.

If the responding SEPP has sent an outgoing Security Capability Negotiation request to the initiating SEPP, the responding SEPP shall compare the FQDN of the initiating SEPP that has been received in the incoming Security Capability Negotiation request message with the FQDN of the responding SEPP that has been sent in the outgoing Security Capability Negotiation request. If the responding SEPP's FQDN lexicographically precedes, it shall reject the incoming HTTP request message with the cause "N32C_EXCHANGE_CAPABILITY_ONGOING" and it shall continue with its initiated procedure and vice versa.

EXAMPLE: assuming SEPP A's FQDN is "sepp.5gc.mnc345.mcc012.3gppnetwork.org" and SEPP B's FQDN is "sepp.5gc.mnc346.mcc012.3gppnetwork.org", then SEPP A's FQDN precedes SEPP B's FQDN and SEPP A proceeds with its exchange capability procedure.

A SEPP may be configured to accept an HTTP request from a given PLMN and not to send an HTTP request for exchange capability towards that PLMN.

5.2.3 Parameter Exchange Procedure

5.2.3.1 General

The parameter exchange procedure shall be executed if the security capability negotiation procedure selected the security capability as PRINS. The parameter exchange procedure is performed to:

- Agree on a cipher suite to use for protecting NF service related signalling over N32-f; and
- Optionally, exchange the protection policies to use for protecting NF service related signalling over N32.

5.2.3.2 Parameter Exchange Procedure for Cipher Suite Negotiation

The parameter exchange procedure for cipher suite negotiation shall be performed after the security capability negotiation procedure if the selected security policy is PRINS. If there is a change in the cipher suite and the SEPP wants to renegotiate it, then the SEPP may reuse the parameter exchange procedure to override what was exchanged before.

The procedure is described in Figure 5.2.3.2-1 below.

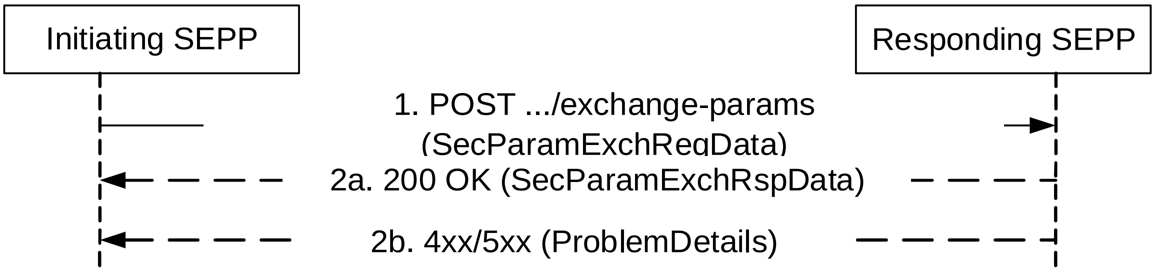
Initiating SEPP
Responding SEPP
1. POST .../exchange-params (SecParamExchReqData)
2a. 200 OK (SecParamExchRspData)
2b. 4xx/5xx (ProblemDetails)

Figure 5.2.3.2-1: Parameter Exchange Procedure for Cipher Suite Negotiation

1. The initiating SEPP issues a HTTP POST request towards the responding SEPP with the request body containing the "SecParamExchReqData" IE carrying the following information

   - Supported cipher suites;

   The supported cipher suites shall be an ordered list with the cipher suites mandated by 3GPP TS 33.501 [6] appearing at the top of the list.

   The initiating SEPP also provides a N32-f context identifier for the responding SEPP to use towards the initiating SEPP for subsequent JOSE Protected Message Forwarding procedures over N32-f (see clause 5.3.3) when the responding SEPP acts as the forwarding SEPP.

2a. On successful processing of the request, the responding SEPP shall respond to the initiating SEPP with a "200 OK" status code and a POST response body that contains the following information

   - Selected cipher suite

   The responding SEPP compares the initiating SEPP's supported cipher suites to its own supported cipher suites and selects, based on its local policy, a cipher suite, which is supported by both the SEPPs. The responding SEPP's supported cipher suites shall be an ordered list with the cipher suites mandated by 3GPP TS 33.501 [6] appearing at the top of the list. The selected cipher suite is applicable for both the directions of communication between the SEPPs.

The responding SEPP also provides a N32-f context identifier for the initiating SEPP to use towards the responding SEPP for subsequent JOSE Protected Message Forwarding procedures over N32-f (see clause 5.3.3) when the initiating SEPP acts as the forwarding SEPP.

If the receiving SEPP already has a previously negotiated cipher suite, the SEPP shall overwrite it with the new one.

2b. On failure, the responding p-SEPP shall respond to the initiating SEPP with an appropriate 4xx/5xx status code as specified in clause 6.1.4.3. If the SEPP already has a previously negotiated cipher suite, the SEPP shall continue to use the same.

NOTE : If a SEPP already has a previously negotiated cipher suite and a new cipher suite is also received, the SEPP starts applying the new cipher suite immediately and also continues with the old cipher suite for a limited time period. This allows messages with old policies to be completed gracefully.

If the initiating SEPP receives a security parameter exchange request from the responding SEPP before receiving a response for its request (i.e security parameter exchange procedure collision), the initiating SEPP shall compare its FQDN that was sent in its request with the FQDN of the responding SEPP that is received in the security parameter exchange request message. If the initiating SEPP's FQDN lexicographically precedes, it shall reject the incoming HTTP request message with the cause "SECURITY_PARAM_EXCHANGE_COLLISION" and it shall continue with its initiated procedure and vice versa.

EXAMPLE: Assuming SEPP A's FQDN is "sepp.5gc.mnc345.mcc012.3gppnetwork.org" and SEPP B's FQDN is "sepp.5gc.mnc346.mcc012.3gppnetwork.org", then SEPP A's FQDN precedes SEPP B's FQDN and SEPP A proceeds with its security parameter exchange procedure.

5.2.3.3 Parameter Exchange Procedure for Protection Policy Exchange

The parameter exchange procedure for protection policy exchange may be performed after the Parameter Exchange Procedure for Cipher Suite Negotiation (see clause 5.2.3.2). If a HTTP/2 connection does not exist towards the peer SEPP at the time of initiating this procedure, the HTTP/2 connection shall be established. If there is a change in the protection policy exchange and the SEPP wants to renegotiate it, then the SEPP may reuse the parameter exchange procedure for the protection policy exchange to override what was exchanged before. If the parameter exchange procedure for the protection policy exchange is not performed, then the protection policies between the SEPP shall be exchanged out of bands.

The procedure is described in Figure 5.2.3.3-1 below.

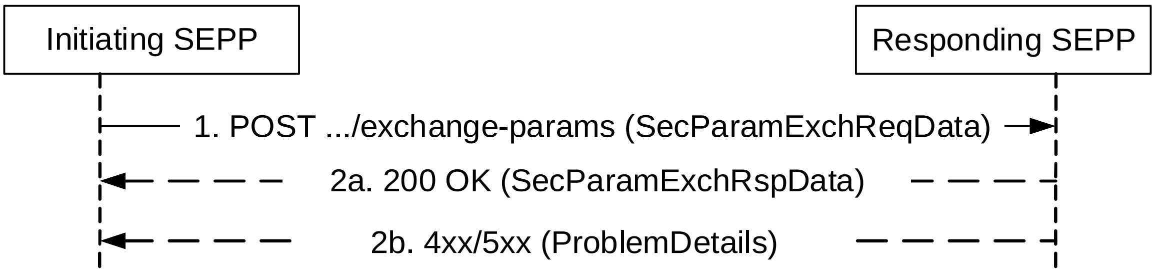
Initiating SEPP
Responding SEPP
1. POST .../exchange-params (SecParamExchReqData)
2a. 200 OK (SecParamExchRspData)
2b. 4xx/5xx (ProblemDetails)

Figure 5.2.3.3-1: Parameter Exchange Procedure for Protection Policy Exchange

1. The initiating SEPP issues a HTTP POST request towards the responding SEPP with the request body containing the "SecParamExchReqData" IE carrying the following information:

   - Protection policy information

The protection policy information contains:

- API to IE mapping containing the mapping information of list of leaf IEs for each:

  - Request/response and Subscribe / Unsubscribe service operation, identified by the API URI and method; and/or
  - Callbacks (e.g Notification service operation), identified by the value of the 3GPP custom HTTP header "3gpp-Sbi-Callback" (see clause 5.2.3 of 3GPP TS 29.500 [4]).
- List of IE types that are to be protected across N32-f (i.e the data type encryption policy as specified in clause 13.2.3.2 of 3GPP TS 33.501 [6]); and
- Modification policy: Against each leaf IE in the API to IE mapping information, a boolean flag indicating whether that IE is allowed to be modified by an IPX on the side of the SEPP sending the protection policy information.

Alternatively, if both the initiating SEPP and the responding SEPP support the PSEPRO feature (PRINS Security Profiles Support, see clause 6.1.7), the initiating SEPP may include a candidate list of security profiles instead of Protection policy information in the parameter exchange request message towards the responding SEPP.

NOTE 1: The definition of security profiles is out of scope of 3GPP.

2a. On successful processing of the request, the responding SEPP shall respond to the initiating SEPP with a "200 OK" status code and a POST response body that contains the following information

- Selected protection policy information

The Selected protection policy information contains the IEs allowed to be modified by an IPX on the side of the responding SEPP. If the responding SEPP connects to several IPXs, an isModifiable IE may be included to indicate an IE is allowed to be modified by all IPX(s) or an map type of isModifiableByIpx IE may be included to indicate an IE is allowed to be modified by an IPX identified by the key of ipxProviderId IE if this IE is allowed to be modified by some of (but not all) the IPX(s), as specified in clause 13.2.3.4 of 3GPP TS 33.501 [6].

The initiating SEPP shall store the modification policy which are sent from responding SEPP in selected protection policy information and the responding SEPP shall store the modification policy which are sent from the initiating SEPP in the protection policy information. The SEPP receiving the subsequent message transfers over N32-f shall check whether the modifications performed by the IPXs were permitted by the respective modification policy.

The SEPPs shall store the encryption policy in selected protection policy information and shall apply this policy for subsequent message transfers over N32-f. The encryption policy in selected protection policy is applicable for both the directions of communication between the SEPPs.

Alternatively, the responding SEPP shall return the selected security profiles to the initiating SEPP, if the responding SEPP supports the PSEPRO feature, and the initiating SEPP sent a candidate list of security profiles in the exchange parameter request message to the responding SEPP.

If the receiving SEPP already has a previously negotiated protection policy information, the SEPP shall overwrite it with the new one.

The HTTP/2 connection used for the N32 handshake procedures may be terminated after the completion of this procedure.

2b. On failure, the responding SEPP shall respond to the initiating SEPP with an appropriate 4xx/5xx status code as specified in clause 6.1.4.3. If the SEPP already has previously negotiated protection policy information, the SEPP shall continue to use the same.

NOTE 2: If a SEPP already has a previously negotiated cipher suite and a new cipher suite is also received, the SEPP starts applying the new cipher suite immediately and also continues with the old cipher suite for a limited time period. This allows messages with old policies to be completed gracefully.

An illustration of how the protection policy is stored and looked up in the SEPP is provided in figure 5.2.3.3-2

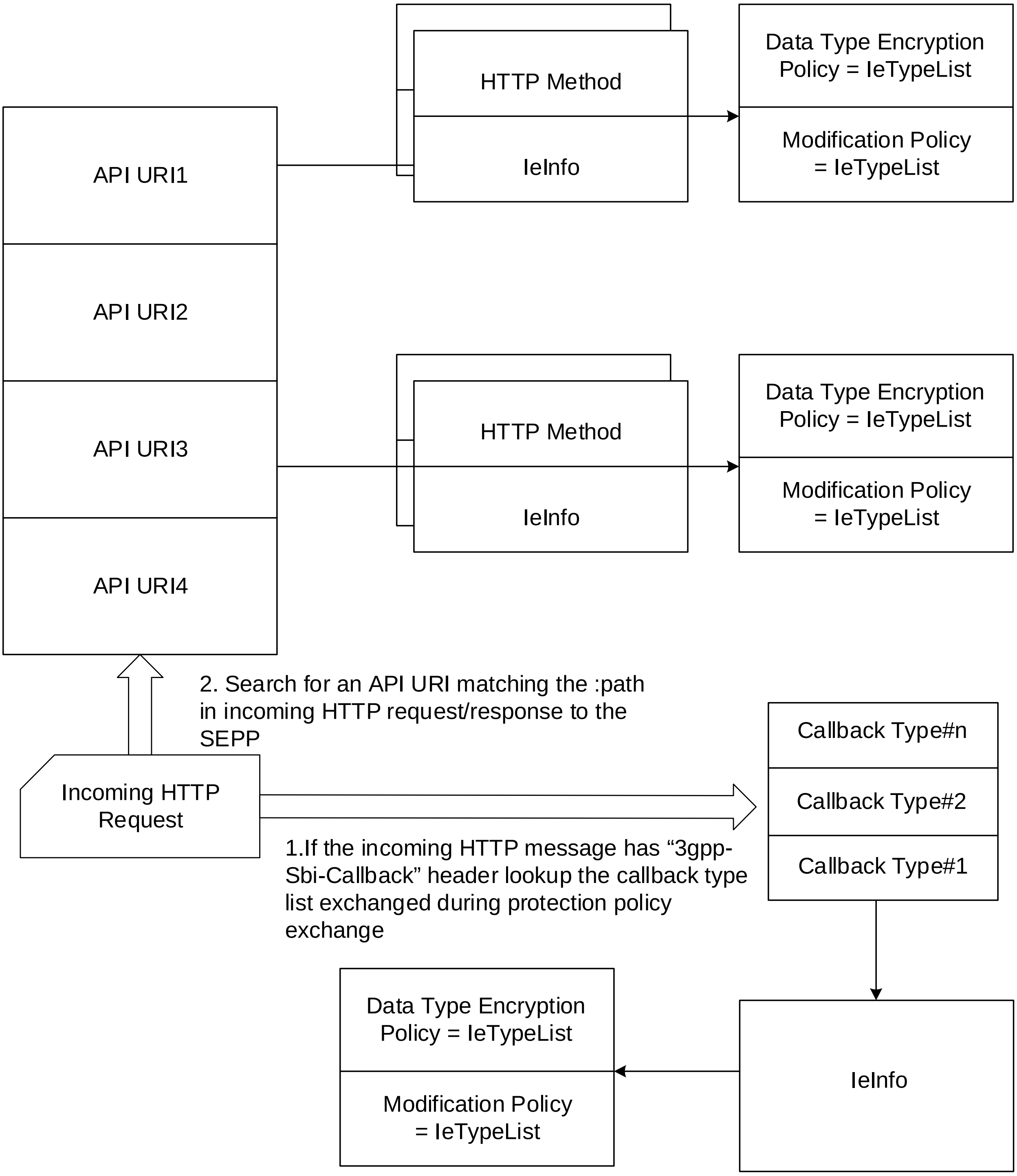
HTTP Method
IeInfo
Data Type Encryption Policy = IeTypeList
Modification Policy = IeTypeList
API URI1
API URI2
API URI3
API URI4
HTTP Method
IeInfo
Data Type Encryption Policy = IeTypeList
Modification Policy = IeTypeList
2. Search for an API URI matching the :path in incoming HTTP request/response to the SEPP
Callback Type#n
Callback Type#2
Callback Type#1
Incoming HTTP Request
1.If the incoming HTTP message has "3gpp-Sbi-Callback" header lookup the callback type list exchanged during protection policy exchange
Data Type Encryption Policy = IeTypeList
Modification Policy = IeTypeList
IeInfo

Figure 5.2.3.3-2: Protection Policy Storage and Lookup in SEPP

During the N32-f message forwarding, the SEPP looks at a HTTP request or response it receives from an NF service consumer or NF service producer and then uses the above tables to decide which IEs and headers in the message it shall cipher and integrity protect and which IEs it shall allow the IPXes to modify.

5.2.3.4 Parameter Exchange Procedure for Security Information list Exchange

The initiating SEPP shall initiate a Security Information list exchange procedure towards the responding SEPP to exchange the Security Information lists that contain information on IPX public keys or certificates that are needed to verify IPX modifications at the receiving SEPP as specified in clause 13.2.2.2 of 3GPP TS 33.501 [6]. If there is a change in the security information list and the SEPP wants to renegotiate it, then the SEPP may reuse the parameter exchange procedure for the security information list exchange to override what was exchanged before.

The procedure is described in Figure 5.2.3.4-1 below.

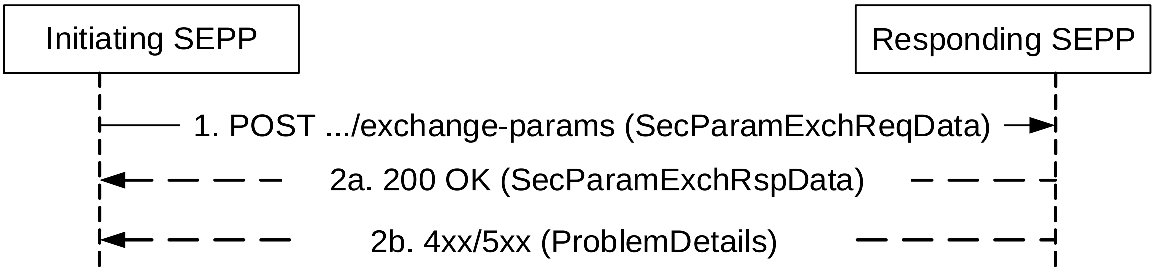
Initiating SEPP
Responding SEPP
1. POST .../exchange-params (SecParamExchReqData)
2a. 200 OK (SecParamExchRspData)
2b. 4xx/5xx (ProblemDetails)

Figure 5.2.3.4-1: Parameter Exchange Procedure for Security Information List exchange

1. The initiating SEPP issues a HTTP POST request towards the responding SEPP with the request body containing the "SecParamExchReqData" IE carrying the following information:

- IPX provider identifier connected to the initiating SEPP;

- List of raw public keys or certificates for that IPX.

2a. On successful processing of the request, the responding SEPP shall respond to the initiating SEPP with a "200 OK" status code and a POST response body that contains the "SecParamExchRspData" IE carrying the following information:

- IPX provider identifier connected to the responding SEPP;

- List of raw public keys or certificates for that IPX.

If the receiving SEPP already has a previously negotiated security information list, the SEPP shall overwrite it with the new one.

2b. On failure, the responding SEPP shall respond to the initiating SEPP with an appropriate 4xx/5xx status code as specified in clause 6.1.4.3. If the SEPP already has previously negotiated security information list, the SEPP shall continue to use the same.

NOTE : If a SEPP already has a previously negotiated cipher suite and a new cipher suite is also received, the SEPP starts applying the new cipher suite immediately and also continues with the old cipher suite for a limited time period. This allows messages with old policies to be completed gracefully.

5.2.4 N32-f Context Termination Procedure

After the completion of the security capability negotiation procedure and/or the parameter exchange procedures, an N32-f context is established between the two SEPPs. The "n32fContextId" of each SEPP is provided to the other SEPP. This context identifier shall be stored in each SEPP until the context is explicitly terminated by the N32-f context termination procedure. The SEPP that is initiating the N32-f context termination procedure shall use the HTTP method POST on the URI: {apiRoot}/n32c-handshake/<apiVersion>/n32f-terminate. If a HTTP/2 connection does not exist towards the receiving SEPP, a HTTP/2 connection shall be created before initiating this procedure. The procedure is shown below in Figure 5.2.4-1.

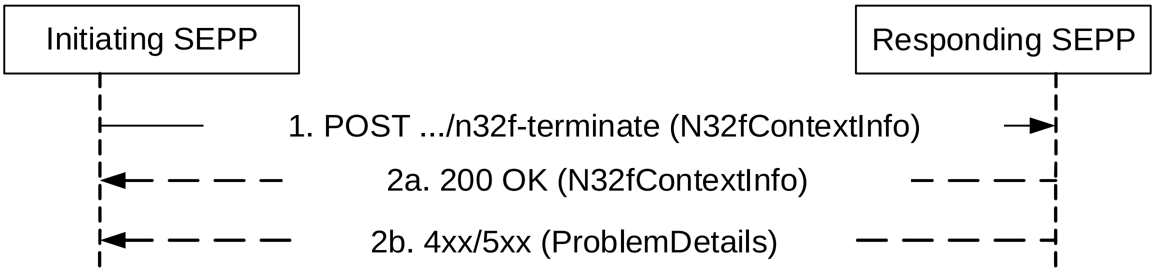
Initiating SEPP
Responding SEPP
1. POST .../n32f-terminate (N32fContextInfo)
2a. 200 OK (N32fContextInfo)
2b. 4xx/5xx (ProblemDetails)

Figure 5.2.4-1: N32f Context Termination Procedure

1. The initiating SEPP issues a HTTP POST request towards the responding SEPP with the request body containing the N32-f context id information that is to be terminated.

2a. On success, the responding SEPP, shall:

- stop sending any further messages over the N32-f towards the initiating SEPP;
- once all the ongoing N32-f message exchanges with the initiating SEPP are completed or timed out, delete the N32-f context identified by the "n32fContextId" provided in the request.

The N32-f HTTP/2 connections from the responding SEPP shall not be deleted if they terminate at an IPX, since that HTTP/2 connection may carry traffic towards other PLMN SEPPs as well. The responding SEPP shall return the status code "200 OK" together with an N32ContextInfo content that carries the "n32fContextId" of the initiating SEPP that the responding SEPP has stored.

The initiating SEPP shall:

- stop sending any further messages over the N32-f towards the responding SEPP;
- once all the ongoing N32-f message exchanges with the responding SEPP are completed or timed out, delete the local N32-f context identified by this "n32fContextId".

If the initiating SEPP receives a N32-f termination request from the responding SEPP before receiving a response for its request (i.e N32-f Context Termination Procedure collision), the initiating SEPP shall process the received N32-f termination request from the responding SEPP and shall return the status code "200 OK" together with an N32ContextInfo content that carries the "n32fContextId" of the responding SEPP that the initiating SEPP has stored. The initiating SEPP shall behave as specified above without waiting for a response from the responding SEPP for its N32-f Context Termination request.

2b. On failure, the responding SEPP shall return an appropriate 4xx/5xx status code together with the "ProblemDetails" JSON body.

5.2.5 N32-f Error Reporting Procedure

When a SEPP is not able to process a message it received over the N32-f interface due to errors, the error information is conveyed to the sending SEPP by using the N32-f error reporting procedure over the N32-c interface. The SEPP that is initiating the N32-f error reporting procedure shall use the HTTP method POST on the URI: {apiRoot}/n32c-handshake/<apiVersion>/n32f-error. If a HTTP/2 connection does not exist towards the receiving SEPP, a HTTP/2 connection shall be created before initiating this procedure. The procedure is shown below in Figure 5.2.5-1.

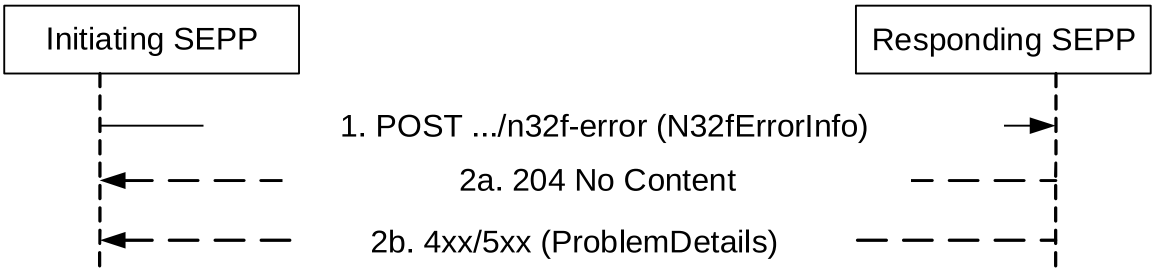
Initiating SEPP
Responding SEPP
1. POST .../n32f-error (N32fErrorInfo)
2a. 204 No Content
2b. 4xx/5xx (ProblemDetails)

Figure 5.2.5-1: N32f Error Reporting Procedure

1. The initiating SEPP issues a HTTP POST request towards the responding SEPP with the request body containing the N32-f error information that is to be reported.

2a. On success, the responding SEPP, shall:

- log that the N32-f request / response message identified by the "n32fMessageId" is not processed by the receiving SEPP;

The responding SEPP shall return the status code "204 No Content".

2b. On failure, the responding SEPP shall return an appropriate 4xx/5xx status code together with the "ProblemDetails" JSON body.

5.3 Message Forwarding Procedure on N32 (N32-f)

5.3.1 Introduction

The N32-f interface is used between two SEPPs for:

- The forwarding of JOSE protected HTTP/2 messages between the NF service consumer and the NF service producer across two PLMNs, when PRINS is the negotiated security policy. The message forwarding on N32-f shall be based on the negotiated security capability and the exchanged security parameters between the two SEPPs (see clause 5.2).
- Forwarding of the HTTP/2 messages between the NF service consumer and the NF service producer without any reformatting and application layer protection, when TLS is the negotiated security policy.

5.3.2 Use of Application Layer Security

5.3.2.1 General

If the negotiated security capability between the two SEPPs is PRINS, one or more HTTP/2 connections between the two SEPPs for the forwarding of JOSE protected message shall be established, which may involve IPX providers on path. The forwarding of messages over the N32-f interface involves the following steps at the sending SEPP:

1. Identification of the protection policy applicable for the API being invoked (i.e either a request/response NF service API or a subscribe/unsubscribe service API or a notification API).
2. Message reformatting as per the identified protection policy.
3. Forwarding of the reformatted message over the N32 interface.

The processing of a message received over the N32-f interface at the receiving IPX provider involves the following steps:

1. Apply the modifications in the "modificationsBlock" appended by the sending IPX provider as JSON patches in the DataToIntegrityProtectBlock (from the decoded "aad" part), if the "modificationsBlock" is received in the message.

2. Determine further modifications required based on modification policy and insert the modification entries in "modificationsBlock".

3. Forwarding the received message with the above inserted modification entries in "modificationsBlock" over the N32 interface.

The processing of a message received over the N32-f interface at the receiving SEPP involves the following steps.

1. Identify the N32-f context using the N32-f context Id received in the message.

2. Verify the integrity protection of the message using the keying material obtained from the TLS layer during the parameter exchange procedure for that N32-f context (see 3GPP TS 33.501 [6]). The TLS connection from which the keying material is obtained is the N32-c TLS connection used for the parameter exchange procedure.

3. Decrypt the ciphertext part of the received JWE message. Decode the "aad" part of the JWE message using BASE64URL decoding.

4. For each entry in the "modificationsBlock" of the received message:

   - First verify the integity protection of that entry using the keying material applicable for the IPX that inserted that block (using the "identity" IE in the "modificationsBlock");

   - Identify the modifications policy exchanged during the parameter exchange procedure with the sending SEPP if the IPX that inserted the modificationsBlock is from the sending SEPP side; else identify the modifications policy applicable for the IPX based on local configuration;

   - Check if the inserted modifications are as per the identified modifications policy;

   - Apply the modifications as a JSON patch in the DataToIntegrityProtectBlock (from the decoded "aad" part).

5. Form the original JSON request / response body from the decrypted ciphertext and the decoded integrity verified "aad" block possibly modified as described in step 4.

6. If the reconstructed HTTP message has an "Authorization" header, then the SEPP shall check whether the service consumer's PLMN ID or SNPN ID is present in the Bearer token contained in the Authorization header (see 3GPP TS 29.510 [18], clause 6.3.5.2.4) and if it matches with the "Remote PLMN ID" or "Remote SNPN ID" of the N32-f context. If they do not match, the SEPP shall respond to the sending SEPP with "403 Forbidden" status code with the application specific cause set as "PLMNID_MISMATCH" or "SNPNID_MISMATCH".

NOTE 1: In this case, the N32-f Error Reporting procedure specified in clause 5.2.5 is not used since the processing of the complete N32-f message fails at the receiving SEPP.

NOTE 2: If the service consumer's PLMN ID or SNPN ID is present in the reconstructed HTTP message, then the receiving SEPP compares this with the sending SEPP's PLMN ID or SNPN ID, which is retrieved from N32f Context (see clause 5.9.3 in 3GPP TS 33.501 [6]). See the above step 6 for the receiving SEPP behaviour. If the service consumer's PLMN ID and SNPN ID are not present, the comparison is not done.

SEPPs and IPX should support gzip coding (see IETF RFC 1952 [23]) in HTTP requests and responses and indicate so in the Accept-Encoding header, as described in clause 6.9 of 3GPP TS 29.500 [4] and clause 6.2.2.2.3.

5.3.2.2 Protection Policy Lookup

When a SEPP receives a HTTP/2 request or response message intended to be routed towards another PLMN, the sending SEPP shall identify the protection policy as given below

1. Identify the target PLMN from the ":aurthority" part of the message using the format specified in clause 6.1.4.3 of 3GPP TS 29.500 [4].

2. Check if the incoming HTTP/2 message has the "3gpp-Sbi-Callback" header. When present, the SEPP shall select the data encryption and modification policy applicable for the specific notification type, identified by the value of the "3gpp-Sbi-Callback" header and the target PLMN, using the notification type list stored as specified in subclase 5.2.3.3.

3. Else, if it is a HTTP/2 request message, then from the ":authority" and ":path" part of the received HTTP/2 request message, form the API URI. For the identified PLMN, check if a protection policy exists for the API URI using the table stored as specified in clause 5.2.3.3.
4. Else, if it is a HTTP/2 response message, then based on the HTTP/2 stream ID on which the response is received, identify the corresponding request that was sent by the SEPP and the protection policy applicable for that request, as specified in step 3.
5. If an entry is not found, then it means that either the particular API has no protection policy exchanged.

Once a protection policy is identified, the SEPP shall apply the application layer security as per the identified protection policy.

5.3.2.3 Message Reformatting

A SEPP on the sending side PLMN applies message reformatting in the following cases:

- When it receives a HTTP/2 request message from an NF service consumer to a an NF service producer in another PLMN;
- When it receives a response HTTP/2 response message from an NF service producer to an NF service consumer in another PLMN.
- When it receives a HTTP/2 notification request message from an NF service producer to an NF service consumer in another PLMN;
- When it receives a HTTP/2 notification response message from an NF service consumer to an NF service producer in another PLMN

The SEPP shall reformat the HTTP/2 message by encapsulating the whole message into the body of a new HTTP POST message. The body of the HTTP POST request / response message shall contain the reformatted original HTTP/2 request/response message respectively. The HTTP POST request/response body shall be encoded as the "N32fReformattedReqMsg"/"N32fReformattedRspMsg" JSON bodies respectively, as specified in clause 6.2.5.

The "N32fReformattedReqMsg"/"N32fReformattedRspMsg" are structured as given below:

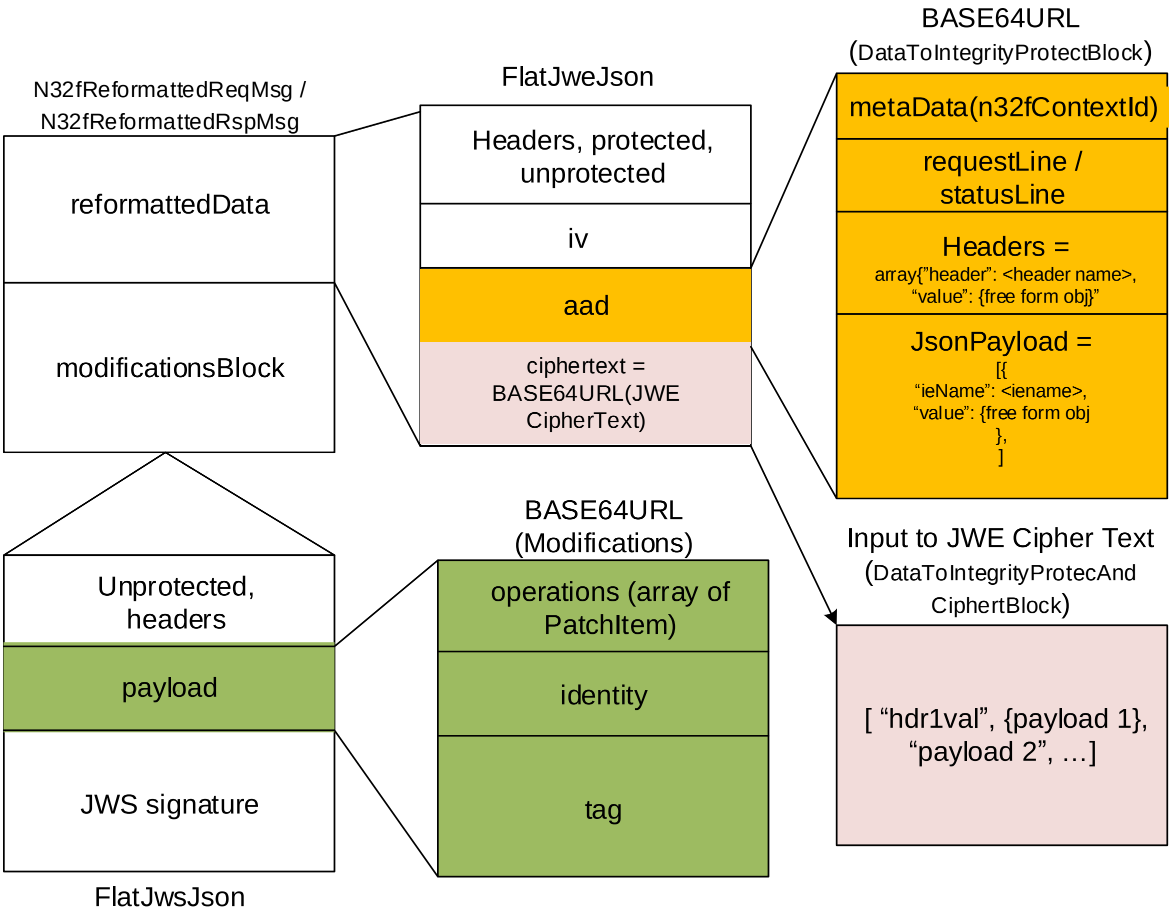

Figure 5.3.2.3-1 JSON representation of a reformatted HTTP message

The "cipherText" part of the reformatted message in FlatJweJson shall be prepared as given below

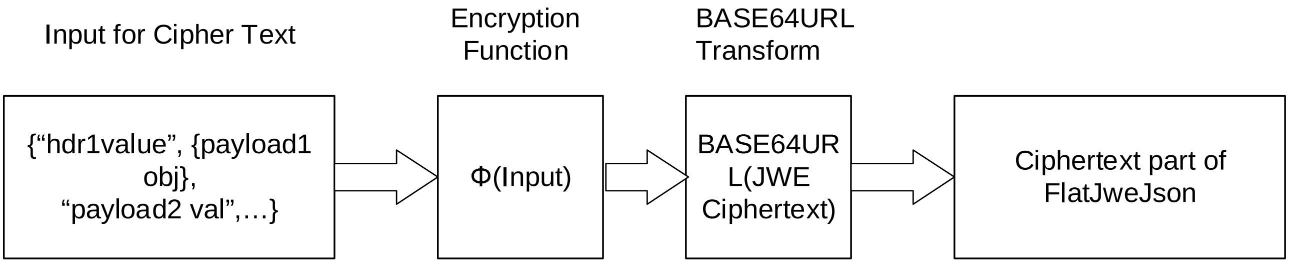
Input for Cipher Text
Encryption Function
BASE64URL Transform
{"hdr1value", {payload1 obj}, "payload2 val",…}
Φ(Input)
BASE64URL(JWE Ciphertext)
Ciphertext part of FlatJweJson

Figure 5.3.2.3-2 Transformation of HTTP Header and Content to Encrypt into CipherText

1. Based on the protection policy exchanged between the SEPPs, the sending SEPP prepares an input for the JWE ciphering and integrity protection as an array of arbitrary types in the "DataToIntegrityProtectAndCipher" block with each entry containing either a HTTP header value or the value of a JSON payload IE of the API message being reformatted. The index value "encBlockIdx" in the contentpart of DataToIntegrityProtectBlock shall point to the index of a header value or IE value in this input array.

2. The input block is fed into an encryption function along with the other required inputs for JWE as specified in IETF RFC 7516 [14].

3. The encryption function outputs the cipher text information. This cipher text is then subjected to BASE64URL transformation as specified in IETF RFC 4648 [15] clause 5.

4. The output of the BASE64URL transform is them encoded as the ciphertext part of FlatJweJson IE specified in clause 6.2.5.2.11.

5.3.2.4 Message Forwarding to Peer SEPP

Once a SEPP reformats the HTTP/2 message into the "N32ReformattedReqMsg"/"N32ReformattedRspMsg" JSON object as specified in clause 5.3.2, the SEPP forwards the message to the receiving SEPP by invoking a HTTP POST method as shown in figure 5.3.2.4-1 below.

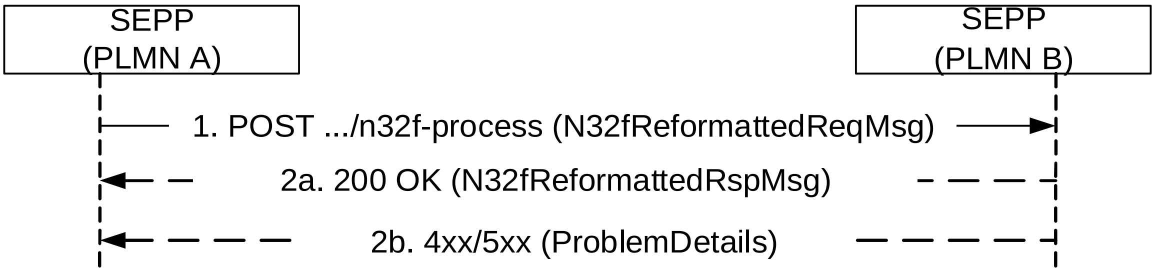
SEPP
(PLMN A)
SEPP
(PLMN B)
1. POST .../n32f-process (N32fReformattedReqMsg)
2a. 200 OK (N32fReformattedRspMsg)
2b. 4xx/5xx (ProblemDetails)

Figure 5.3.2.4-1 Message Forwarding between SEPP on N32-f

1. The initiating SEPP issues a HTTP POST request towards the responding SEPP with the request body containing the "N32ReformattedReqMsg" IE carrying the reformatted HTTP/2 message. The request message shall contain the "n32fContextId" information provided by the responding SEPP to the initiating SEPP earlier during the parameter exchange procedure (see clause 5.2.3). The responding SEPP shall use the "n32fContextId" information to:

   - Locate the agreed cipher suite and protection policy;

   - Locate the n32ContextId to be used in the response.

If the HTTP request/response message to be forwarded over N32-f includes an 3gpp-Sbi-Message-Priority header, the initiating/responding SEPP should additionally insert a 3gpp-Sbi-Message-Priority header in the N32-f message with the same contents as the 3gpp-Sbi-Message-Priority header encoded within the "N32ReformattedReqMsg"/ N32ReformattedRspMsg IE respectively.

NOTE 1: Replicating the information in a N32-f message header enables the receiving SEPP to determine the priority of the forwarded HTTP request/response without having to parse the N32-f message content.

The HTTP request content may be compressed hop by hop over N32-f, if the initiating SEPP or IPX and its next hop (IPX or SEPP) support gzip coding (see IETF RFC 1952 [23]).

2a. On successful processing of the request, the responding SEPP shall:

- decompress the N32-f HTTP request content, if it is compressed;

- reconstruct the HTTP/2 message towards the NF service producer;

- compress the reconstructed HTTP request if the reconstructed HTTP content contains a Content-Encoding header indicating gzip compression;

- forward the reconstructed HTTP/2 message to the NF service producer;

- wait for the response from the NF service producer; and then

- once the response from the NF service producer is received, respond to the initiating SEPP with a "200 OK" status code and a POST response body that contains the "N32ReformattedRspMsg". The "N32ReformattedRspMsg" shall contain the reformatted HTTP response message from the responding PLMN. The response message shall contain the "n32fContextId" information provided by the initiating SEPP to the responding SEPP earlier during the parameter exchange procedure (see clause 5.2.3).

NOTE 2: For unsuccessful processing of the request with "PLMNID_MISMATCH", see clause 5.3.2.1.

The responding SEPP shall be able to map the response received from the NF service producer to the HTTP/2 stream ID for the corresponding response it needs to generate towards the initiating SEPP. The HTTP/2 stream ID and the HTTP/2 connection information on either side shall be used to derive this mapping.

The HTTP response content may be compressed hop by hop over N32-f, if the responding SEPP or IPX and its next hop (IPX or SEPP) support gzip coding (see IETF RFC 1952 [23]).

2b. On failure or unsuccessful processing of the request, the responding SEPP shall respond to the initiating SEPP with an appropriate 4xx/5xx status code, the message body shall contain a ProblemDetails structure with the "cause" attribute set to one of the application error as specified in clause 6.2.4.2. The "cause" attribute shall be set to "UNSPECIFIED", if the responding SEPP fails to process the reconstructed message, and the error is reported by N32f error reporting procedure as specified in clause 5.2.5.

5.3.2.5 JOSE Protected Forwarding Options

The JOSE Protected Forwarding Options is used by the sending SEPP or IPX to discover the communication options supported by its next hop (IPX or SEPP) for N32-f message processing.

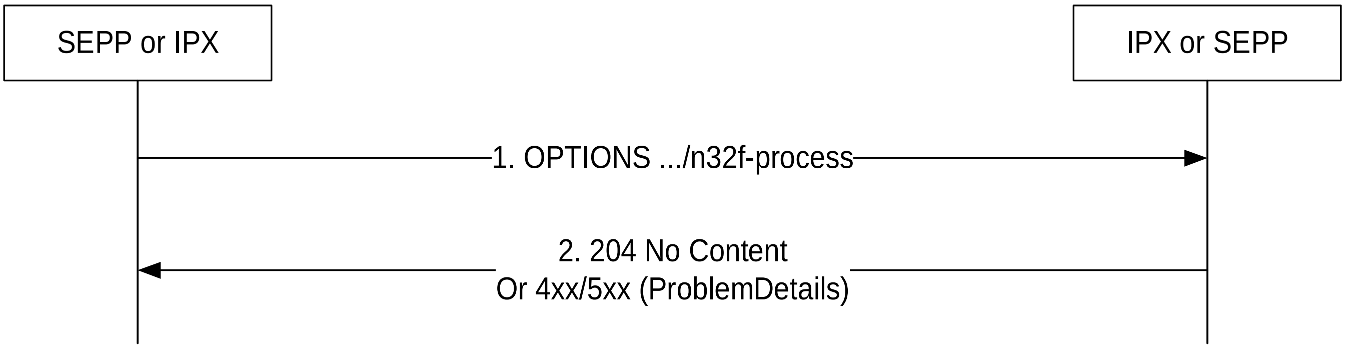

Figure 5.3.2.5-1: Procedure for the discovery of communication options supported by the next hop

1. The sending SEPP or IPX shall send an OPTIONS request to discover the communication options supported by its next hop (IPX or SEPP) for N32-f message processing.
2. If the request is accepted, the next hop (IPX or SEPP) shall respond with the status code 204 No Content and include an Accept-Encoding header (as described in IETF RFC 9110 [9]).

On failure, the next hop shall return one of the HTTP status code listed in Table 6.2.4.3.2.1-3.

5.3.3 Message Forwarding to Peer SEPP when TLS is used

When the negotiated security policy between the SEPPs is TLS, then the procedures described in clause 5.3.2 shall not be applied. Messages shall be forwarded to the peer SEPP as specified in clause 6.1.4.3.4 of 3GPP TS 29.500 [4].

On failure or unsuccessful processing of the incoming N32-f request, the responding SEPP shall respond to the initiating SEPP with an appropriate 4xx/5xx status code including a ProblemDetails structure with the "cause" attribute set to one of the following application errors as specified in Table 5.3.3-1.

Table 5.3.3-1: Protocol and application errors generated by SEPP

Protocol or application Error	HTTP status code	Description
"CONTEXT_NOT_FOUND"	403 Forbidden	The N32-f request which was received over TLS connection is rejected due to having no related N32-c context.

5.3.4 Void

5.4 Nsepp_Telescopic_FQDN_Mapping Service

5.4.1 General

The Nsepp_Telescopic_FQDN_Mapping service is used between any Network Function and the SEPPs in the same PLMN, if TLS protection between the Network Function and the SEPP relies on using telescopic FQDN. See clause 28.5.2 of 3GPP TS 23.003 [19] and clause 6.1.4.3 of 3GPP TS 29.500 [4]) for the definition and use of Telescopic FQDN.

5.4.2 Foreign FQDN to Telescopic FQDN Mapping Procedure

This procedure is initiated by an NF Service Consumer (typically an NRF or an NSSF) that needs to interact with a NF in a foreign PLMN (typically the corresponding NRF or NSSF), and to do so, it needs to build a telescopic FQDN of said NF (i.e. concatenation of the FQDN of the foreign FQDN, and the FQDN of the local SEPP), and then the resulting telescopic FQDN needs to be "flattened" (i.e. the FQDN of the NF in the foreign PLMN needs to be converted to a singel label). The procedure is described in Figure 5.4.2-1 below.

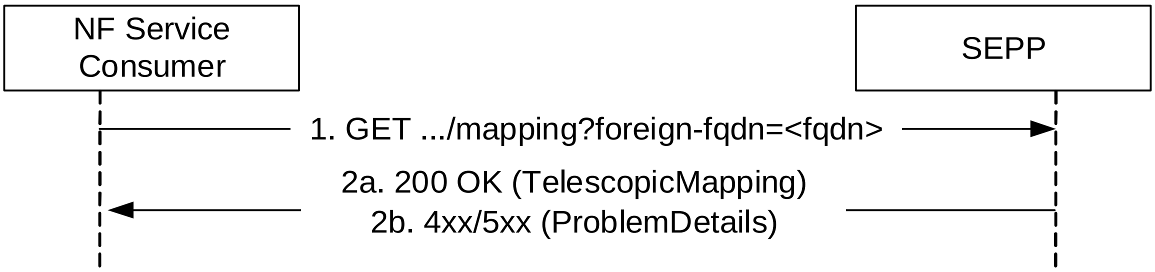
NF Service Consumer
SEPP
1. GET .../mapping?foreign-fqdn=<fqdn>
2a. 200 OK (TelescopicMapping)
2b. 4xx/5xx (ProblemDetails)

Figure 5.4.2-1: Foreign FQDN to Telescopic FQDN Mapping Procedure

1. The NF Service Consumer issues an HTTP GET request towards the local SEPP with a query parameter "foreign-fqdn" containing the FQDN of the NF in the foreign PLMN, that needs to be transformed into a flattened telescopic FQDN.

2a. On successful processing of the request, the responding SEPP shall respond to the NF Service Consumer with a "200 OK" status code and a response body that contains a JSON object of type "TelescopicMapping", containing as attributes the label to be used as first label in the telescopic FQDN, and the domain of the local SEPP to be appended after such first label. The resulting FQDN shall be used by the NF Consumer to setup a TLS session terminated in the local SEPP, where the SEPP shall present a server certificate with a wildcard domain matching the returned telescopic FQDN.

5.4.3 Telescopic FQDN to Foreign FQDN Mapping Procedure

This procedure is initiated by an NF Service Consumer (typically another SEPP) that has received a service request with an unknown first label of a telescopic FQDN. Typically, this SEPP may interact with other SEPPs in the same PLMN in order to determine if there is an existing mapping for a given label to an FQDN of a foreign FQDN; this procedure is only expected to be used when multiple SEPPs are deployed in a PLMN. The procedure is described in Figure 5.4.3-1 below.

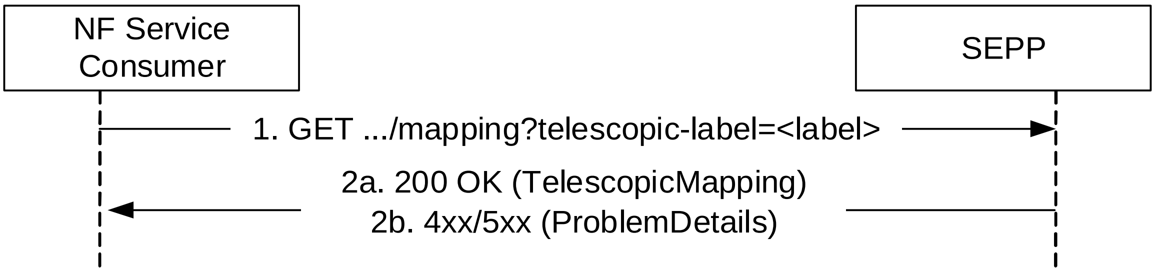
NF Service Consumer
SEPP
1. GET .../mapping?telescopic-label=<label>
2a. 200 OK (TelescopicMapping)
2b. 4xx/5xx (ProblemDetails)

Figure 5.4.3-1: Foreign FQDN to Telescopic FQDN Mapping Procedure

1. The NF Service Consumer issues an HTTP GET request towards another SEPP with a query parameter "telescopic-label" containing the first label of a given telescopic FQDN, whose mapping towards an FQDN of an NF in a foreign PLMN needs to be verified.

2a. On successful processing of the request, the responding SEPP shall respond to the NF Service Consumer with a "200 OK" status code and a response body that contains a JSON object of type "TelescopicMapping", containing as attribute "foreignFqdn", containing the FQDN of the NF in the foreign PLMN.

5.5 Support of Roaming Intermediaries

5.5.1 General

Roaming services providers provide the technical and commercial means to facilitate the deployment and operation of roaming services between a client operator and a set of selected connected operators (see clause 6.45 of 3GPP TS 22.261 [28]).

The communication between two SEPPs may pass up to two Roaming Intermediaries. The changes made by Roaming Intermediaries to messages originated by a SEPP, based on the originating PLMNs policy, shall be identifiable by the receiving SEPP.

NOTE: In this release of the specification, Roaming Intermediary refers to Roaming Hub, only.

5.5.2 N32-c connection establishment via Roaming Intermediaries

5.5.2.1 N32-c connection establishment using HTTP CONNECT

5.5.2.1.1 General

This clause specifies the requirements that apply in scenarios where a PLMN SEPP makes use of Roaming Intermediaries and support messages generated by Roaming Intermediaries as specified in clause 5.9.3.2a of 3GPP TS 33.501 [6].

The N32 Handshake procedures specified in clause 5.2 shall apply between the c-SEPP and p-SEPP with the following additions:

- Prior to establishing the (end-to-end) N32-c TLS connection with the pSEPP, the c-SEPP shall send an HTTP CONNECT Request to the Roaming Intermediary to request the Roaming Intermediary to set up a TCP connection towards the p-SEPP as specified in clause 5.5.2.1.2 (with one Roaming Intermediary between the SEPPs) and clause 5.5.2.1.3 (with two Roaming Intermediaries between the SEPPs); upon receipt of a successful HTTP CONNECT response, the c-SEPP shall establish the N32-c TLS connection and proceed with the N32 Handshake procedures;

- If a Roaming Intermediary disallows the establishment of the N32-c connection, the Roaming Intermediary shall reject the HTTP CONNECT request as specified in clause 5.5.2.2.

5.5.2.1.2 Successful N32-c connection establishment via one Roaming Intermediary

Figure 5.5.2.1.2-1 depicts the successful establishment of the N32-c connection between c-SEPP and p-SEPP via one Roaming Intermediary (RI-A).

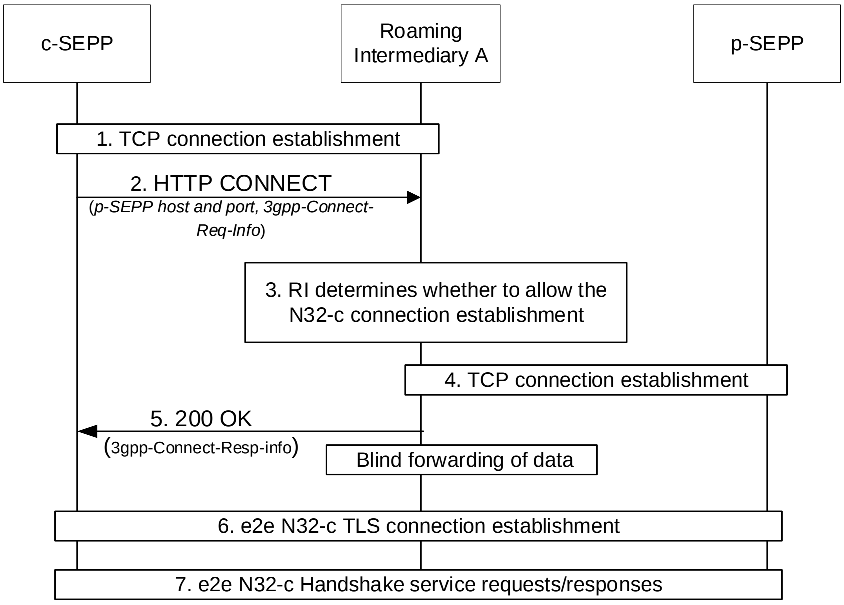
c-SEPP
Roaming Intermediary A
p-SEPP
1. TCP connection establishment
2. HTTP CONNECT
(p-SEPP host and port, 3gpp-Connect-Req-Info)
3. RI determines whether to allow the N32-c connection establishment
4. TCP connection establishment
5. 200 OK
(3gpp-Connect-Resp-info)
Blind forwarding of data
6. e2e N32-c TLS connection establishment
7. e2e N32-c Handshake service requests/responses

Figure 5.5.2.1.2-1: Successful N32-c connection establishment via one Roaming Intermediary

1. The c-SEPP shall establish a TCP connection with the RI-A.
2. The c-SEPP shall send an HTTP CONNECT request to the RI-A to request the RI-A to establish a TCP connection towards the p-SEPP. The authority of the HTTP CONNECT request shall contain the p-SEPP's FQDN.

   The c-SEPP shall include the following information in the 3gpp-Connect-Req-Info header in the HTTP CONNECT request:

   - the HTTP connect purpose set to "n32c" to indicate that the TCP connection requested to be established is to set up an N32-c connection between the c-SEPP and p-SEPP;
   - the c-SEPP's PLMN ID or SNPN ID; and
   - the c-SEPP's FQDN.

   The c-SEPP may also include the following information in the HTTP CONNECT request in the 3gpp-Connect-Req-Info header:

   - the intended N32 purpose(s) of the N32 connection.

The HTTP CONNECT request shall not contain any content (i.e. payload).

3. The Roaming Intermediary shall determine whether to (dis)allow the N32-c connection establishment based on its roaming contractual agreements and the following parameters:

   - the source PLMN ID or SNPN-ID; and
   - the target PLMN ID or SNPN ID.

   The Roaming Intermediary may also consider the following information for the above determination:

   - the HTTP connect purpose, e.g. the Roaming Intermediary may accept an HTTP CONNECT request only for the purpose of establishing a N32-c connection; and/or

- the intended N32 purposes, e.g. the Roaming Intermediary may accept to establish the N32-c connection between the two PLMNs only for specific purposes.

4. If the RI-A allows the establishment of the N32-c connection, it shall establish the TCP connection towards the p-SEPP.

5. On successful processing of the request and establishment of the TCP connection towards the p-SEPP, the RI-A shall respond to the c-SEPP with a "200 OK" status code and may include the following information in the 3gpp-Connect-Resp-Info header in the HTTP CONNECT response:

   - the allowed N32 purposes for the N32 connection, which may be a subset of the N32 purposes signalled in the request; and/or.

   - the p-SEPP FQDN, if the Roaming Intermediary has overwritten the p-SEPP FQDN received from the c-SEPP that the c-SEPP should use for sending its N32-c requests to the p-SEPP.

6 The c-SEPP shall establish the (end to end) N32-c TLS connection with the p-SEPP.

7. The c-SEPP shall proceed with the N32-c Handshake procedures specified in clause 5.2.

5.5.2.1.3 Successful N32-c connection establishment via two Roaming Intermediaries

Figure 5.5.2.1.3-1 depicts the successful establishment of the N32-c connection between c-SEPPs and p-SEPP via two RI’s (RI-A and RI-B).

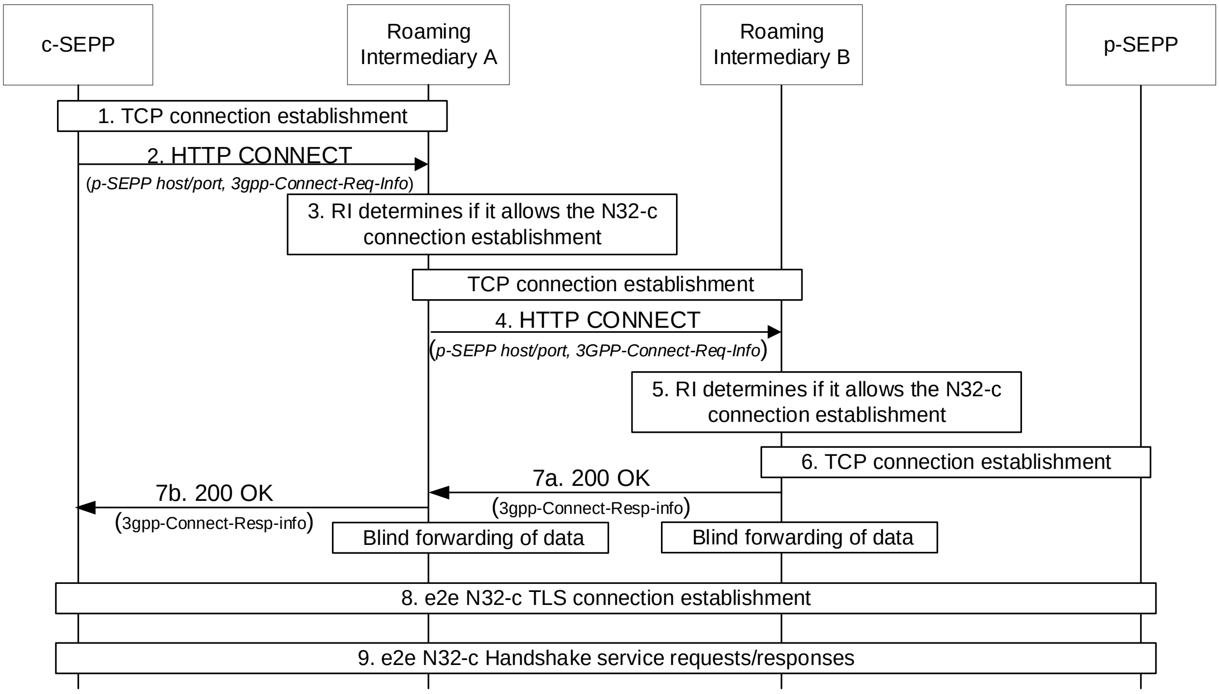
c-SEPP
Roaming Intermediary A
Roaming Intermediary B
p-SEPP
1. TCP connection establishment
2. HTTP CONNECT
(p-SEPP host/port, 3gpp-Connect-Req-Info)
3. RI determines if it allows the N32-c connection establishment
TCP connection establishment
4. HTTP CONNECT
(p-SEPP host/port, 3GPP-Connect-Req-Info)
5. RI determines if it allows the N32-c connection establishment
6. TCP connection establishment
7a. 200 OK
(3gpp-Connect-Resp-info)
7b. 200 OK
(3gpp-Connect-Resp-info)
Blind forwarding of data
Blind forwarding of data
8. e2e N32-c TLS connection establishment
9. e2e N32-c Handshake service requests/responses

Figure 5.5.2.1.3-1: Successful N32-c connection establishment via 2 Roaming Intermediaries

Steps 1 to 3: same as steps 1 to 3 of Figure 5.5.2.1.2-1.

4. If the RI-A allows the establishment of the N32-c connection, it shall establish a TCP connection towards the RI-B and send an HTTP CONNECT request to the RI-B, as described in for step 2 but with the following modification:

   - the RI-A shall include the RI-A's FQDN in the 3gpp-Connect-Req-Info header, instead of including the c-SEPP's FQDN.

steps 5 and 6: same as steps 3 and 4 of Figure 5.5.2.1.2-1.

Steps 7a and 7b: same as step 5 of Figure 5.5.2.1.2-1. The RI-A shall send the response to the c-SEPP only upon receiving the 200 OK response from the RI-B.

Steps 8 and 9: same as steps 6 and 7 of Figure 5.5.2.1.2-1.

5.5.2.2 Error messages originated by Roaming Intermediaries over the N32-c interface

5.5.2.2.1 General

The Roaming Intermediary may reject an N32-c connection establishment request by rejecting the HTTP CONNECT request.

The following error scenarios are supported and further detailed in the following clauses.

1) Errors determined upon receipt of the HTTP CONNECT request

   Examples: Roaming Intermediary rejecting an HTTP CONNECT request due to:

   - the N32-c connection cannot be setup due to contractual reasons;
   - the N32-c connection cannot be setup due to a connectivity issue.

5.5.2.2.2 N32-c connection establishment rejection by Roaming Intermediary A

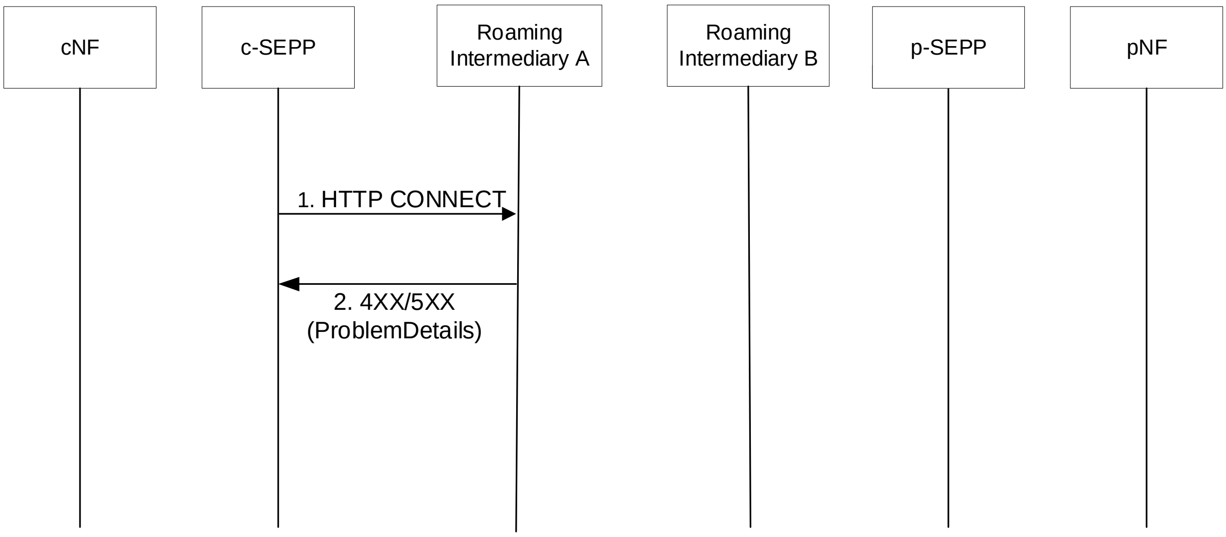

Figure 5.5.2.2.2-1: N32-c connection establishment rejection by Roaming Intermediaries

1. The c-SEPP shall send an HTTP CONNECT request message as specified in step 2 of Figure 5.5.2.1.2-1.
2. On failure, if the Roaming Intermediary A determines that an N32 connection shall not or cannot be established e.g. due to contractual reasons or connectivity issues, the Roaming Intermediary shall return an 4xx or 5xx response with the ProblemDetails providing details on the N32 related error for the c-SEPP.

5.5.2.2.3 N32-c connection establishment rejection by Roaming Intermediary B

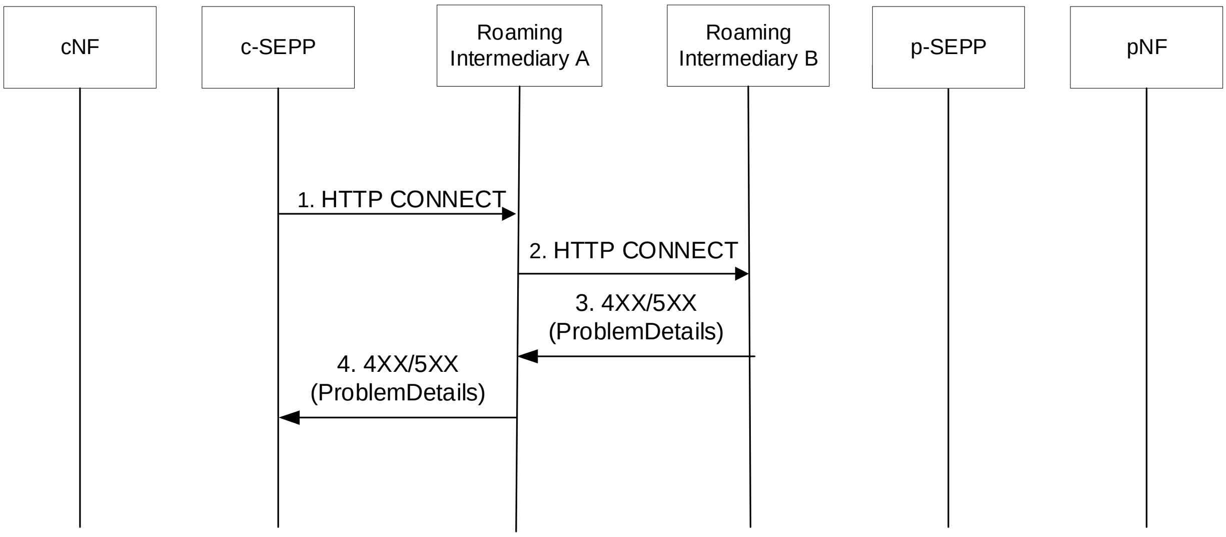

Figure 5.5.2.2.3-1: N32-c connection establishment rejection by Roaming Intermediaries

- Steps 1 and 2: same as steps 2 and 4 of Figure 5.5.2.1.3-1.
- Steps 3 and 4: same as step 2 of Figure 5.5.2.2.2-1. The RI-A shall send the response to the c-SEPP only upon receiving the 4xx or 5xx response from the RI-B.

5.5.3 N32-f messages forwarding or origination via Roaming Intermediaries

5.5.3.1 Error messages originated by (or related to) Roaming Intermediaries over the N32-f interface

5.5.3.1.1 General

Error messages may be originated from either PLMN SEPPs or Roaming Hubs to adjacent Roaming Hubs or adjacent PLMN SEPPs, in an identifiable way. Furthermore, if allowed by the PLMN policy, the SEPP shall be able to send error messages on the N32 interface to a roaming hub via the N32-f. See clause 5.9.3.2 of 3GPP TS 33.501 [6]).

The following error scenarios are supported and further detailed in the following clauses.

1) N32-f related error determined upon receipt of an N32-f request

   Examples: Roaming Intermediary rejecting an N32-f request due to:

   - the N32-f connection cannot be setup due to contractual reasons;
   - the N32-f connection cannot be setup due to a connectivity issue;
   - incompatible encryption/plain information in the request (e.g. an IE is encrypted while it was expected in clear);
   - N32-f request not delivered due contractual reasons.

2) N32-f related error determined upon receipt of an N32-f response

   Example:

   - incompatible encryption/plain information in the N32-f response (e.g. an IE is encrypted while it was expected in clear).

3) Applicative (i.e. SBI related) error determined upon receipt of an N32-f request

Example:

- Roaming Intermediary rejecting a UE Registration on behalf of the involved PLMNs based on roaming agreements.

Editor's Note: It is left FFS how to handle the applicative errors for termination of session and deregistration of the UE by the Roaming Intermediary based on roaming agreements.

5.5.3.2 N32-f related error determined upon receipt of an N32-f request

5.5.3.2.1 Error message originated by Roaming Intermediary via N32-f

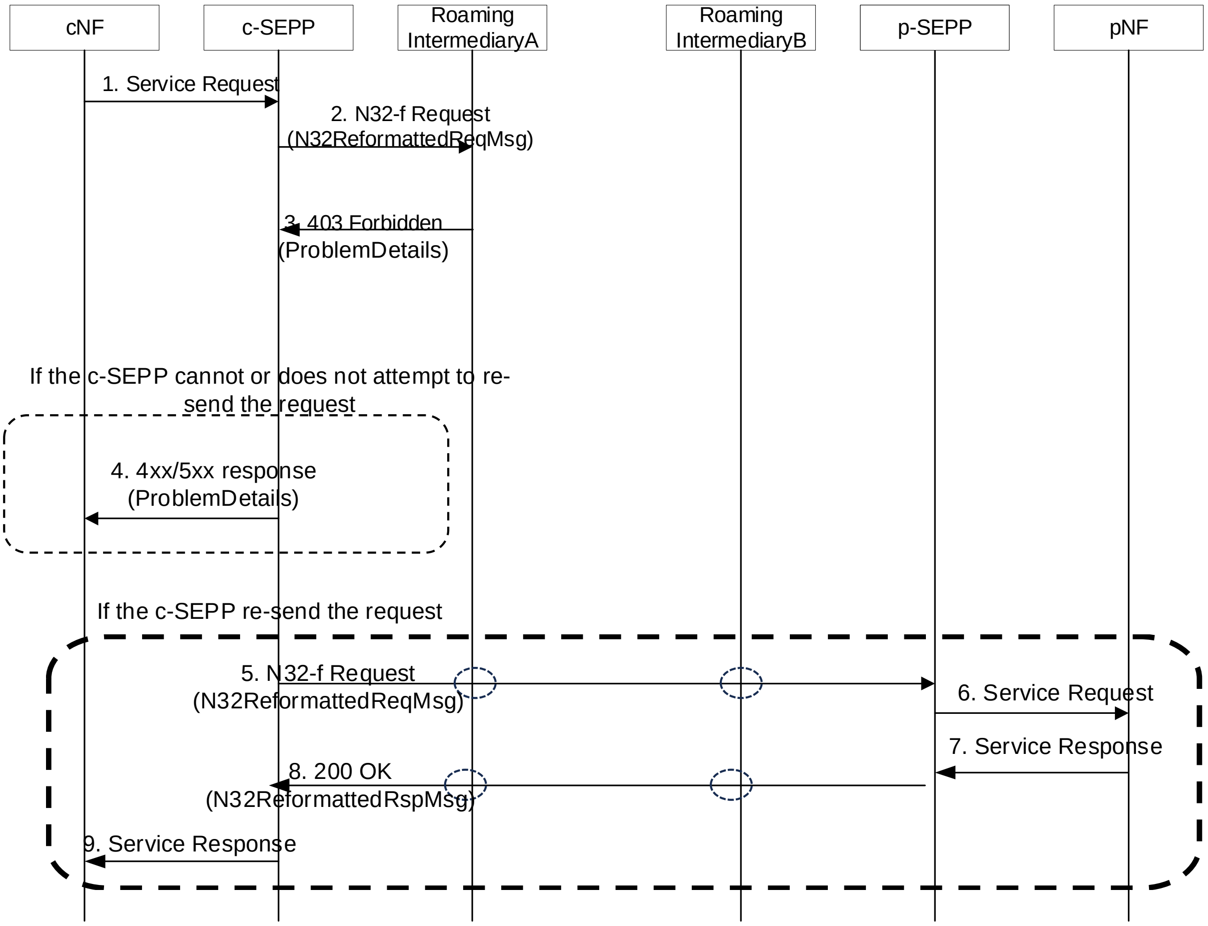
cNF
c-SEPP
Roaming IntermediaryA
Roaming IntermediaryB
p-SEPP
pNF
1. Service Request
2. N32-f Request (N32ReformattedReqMsg)
3. 403 Forbidden (ProblemDetails)
If the c-SEPP cannot or does not attempt to re-send the request
4. 4xx/5xx response (ProblemDetails)
If the c-SEPP re-send the request
5. N32-f Request (N32ReformattedReqMsg)
6. Service Request
7. Service Response
8. 200 OK (N32ReformattedRspMsg)
9. Service Response

Figure 5.5.3.2.1-1: Error message originated by Roaming Intermediary via N32-f

1. The cSEPP receives a service request (HTTP request) message from cNF.

2. The cSEPP sends an N32-f request using PRINS security (i.e. JOSE protected message) to forward the service request message to the pSEPP.

3. The Roaming Intermediary detects an N32-f related error and returns an N32-f error response, e.g. "403 Forbidden" response, with the ProblemDetails data providing details on the N32-f related error for the cSEPP. If the error is due to an encryption policy mismatch, the ProblemDetails may include the invalidParams attribute indicating which IEs were received ciphered when they were expected to be received in clear, and vice-versa. The N32-f error response may additionally contain a suggested status code (e.g. "504 Gateway Timeout") and a suggested application error (e.g. "TARGET_PLMN_NOT_REACHABLE") that the Roaming Intermediary

suggests the cSEPP to forward in the error response to the cNF, if the cSEPP cannot or does not attempt to re-send the N32-f request taking into account the N32-f error information.

4. If the cSEPP cannot or does not attempt to re-send the N32-f request taking into account the N32-f error information, the cSEPP sends an error response to the cNF. The cSEPP may use the suggested status code and/or suggested application error for the error response sent to the cNF (e.g. the cSEPP may send a "504 Gateway timeout" response with the cause "TARGET_PLMN_NOT_REACHABLE" in the ProblemDetails).

5. Alternatively, the cSEPP may re-send the N32-f request taking into account the N32-f error information that was received from the Roaming Intermediary. For instance, if the cSEPP receives an error message with the application error "POLICY_MISMATCH", the cSEPP may change the data type encryption policy to 'Parameter shall be encrypted' or 'Parameter shall not be encrypted', if this is allowed by local policies, and if necessary, re-negotiate the data type encryption policy with the peer SEPP. After that, the cSEPP may re-send the N32-f Request based on the updated data type encryption policy to the Roaming Intermediary.

6-9. The rest of procedures are processed accordingly.

5.5.3.2.2 Error message originated by pSEPP on N32-f (and optionally N32-c)

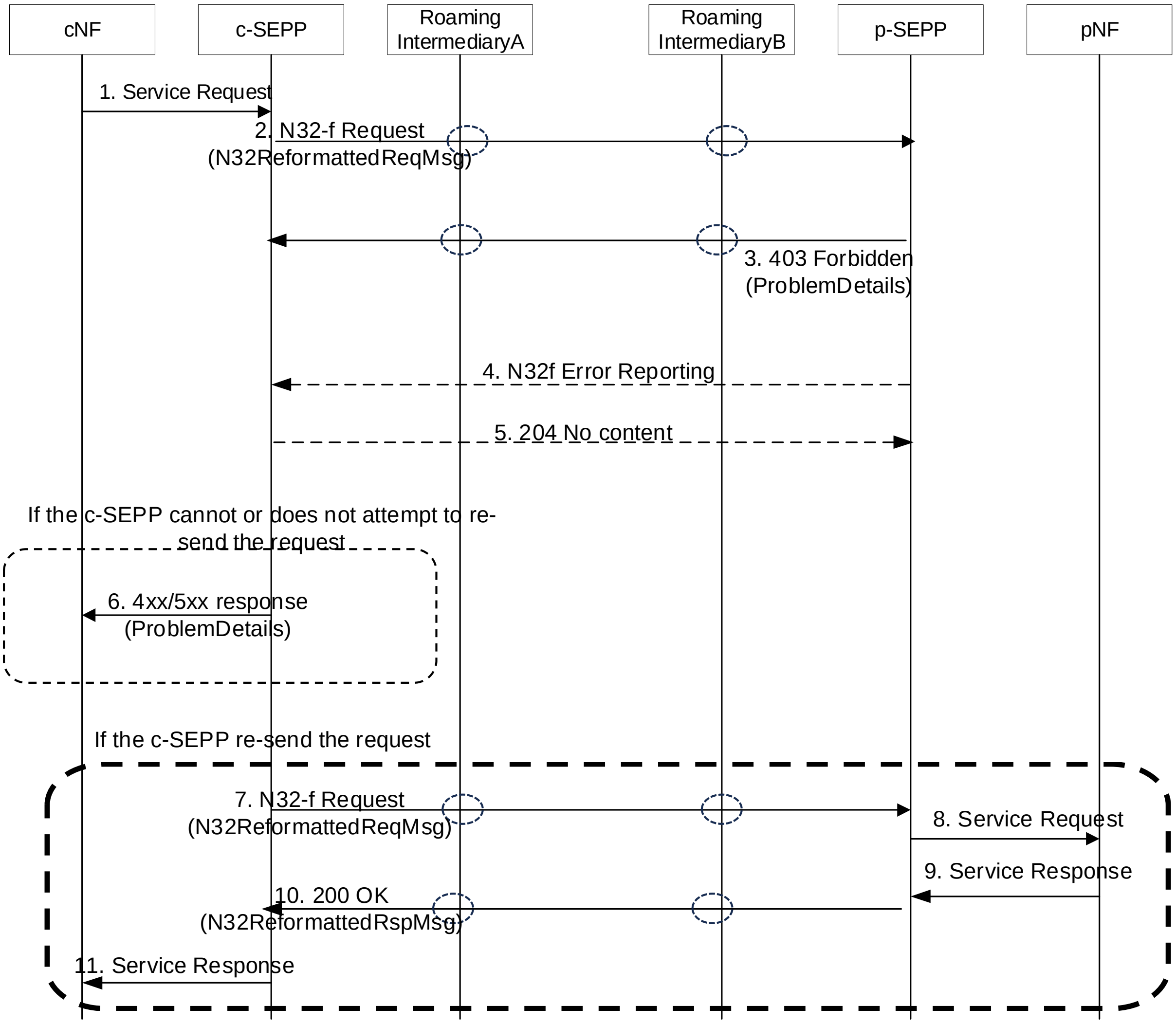
cNF
c-SEPP
Roaming IntermediaryA
Roaming IntermediaryB
p-SEPP
pNF
1. Service Request
2. N32-f Request (N32ReformattedReqMsg)
3. 403 Forbidden (ProblemDetails)
4. N32f Error Reporting
5. 204 No content
If the c-SEPP cannot or does not attempt to re-send the request
6. 4xx/5xx response (ProblemDetails)
If the c-SEPP re-send the request
7. N32-f Request (N32ReformattedReqMsg)
8. Service Request
9. Service Response
10. 200 OK (N32ReformattedRspMsg)
11. Service Response

Figure 5.5.3.2.2-1: Error message originated by pSEPP via N32-f (and optionally N32-c)

1. The cSEPP receives a service request (HTTP request) message from cNF.

2. The cSEPP sends an N32-f request using PRINS security (i.e. JOSE protected message) to forward the service request message to the pSEPP.

3. The pSEPP detects an N32-f related error and returns an N32-f error response, e.g. "403 Forbidden" response, with the ProblemDetails data providing details on the N32-f related error for the cSEPP. If the error is due to an encryption policy mismatch, the ProblemDetails may include the invalidParams attribute indicating which IEs were received ciphered when they were expected to be received in clear, and vice-versa. The N32-f error response may additionally contain a suggested status code (e.g. "504 Gateway Timeout") and a suggested application error (e.g. "TARGET_PLMN_NOT_REACHABLE") that the pSEPP suggests the cSEPP to forward in the error response to the cNF, if the cSEPP cannot or does not attempt to re-send the N32-f request taking into account the N32-f error information.

4. The pSEPP may also send an N32-c request (HTTP POST request) towards the cSEPP with the content containing the N32-f error information that is to be reported (see clause 5.2.5).

5. The cSEPP shall return the status code "204 No Content" as the response to the N32-f Error Reporting. (see clause 5.2.5)

6. If the cSEPP cannot or does not attempt to re-send the N32-f request taking into account the N32-f error information, the cSEPP sends an error response to the cNF. The cSEPP may use the suggested status code and/or suggested application error for the error response to the cNF (e.g. the cSEPP may send a"504 Gateway timeout" response with the cause "TARGET_PLMN_NOT_REACHABLE" in the ProblemDetails).

7. Alternatively, the cSEPP may re-send the N32-f request taking into account the N32-f error information that was received from the pSEPP. For instance, if the cSEPP receives an error message with the application error "POLICY_MISMATCH", the cSEPP may change the data type encryption policy to 'Parameter shall be encrypted' or 'Parameter shall not be encrypted', if this is allowed by local policies, and if necessary, re-negotiate the data type encryption policy with the peer SEPP. After that, the cSEPP may re-send the N32-f Request based on the updated data type encryption policy to the Roaming Intermediary.

8-11. The rest of procedures are processed accordingly.

5.5.3.3 N32-f related error determined upon receipt of an N32-f response

5.5.3.3.1 Error message originated by Roaming Intermediary via N32-f

The procedure below describes the situation in which Roaming Intermediary B detects an error in the response.

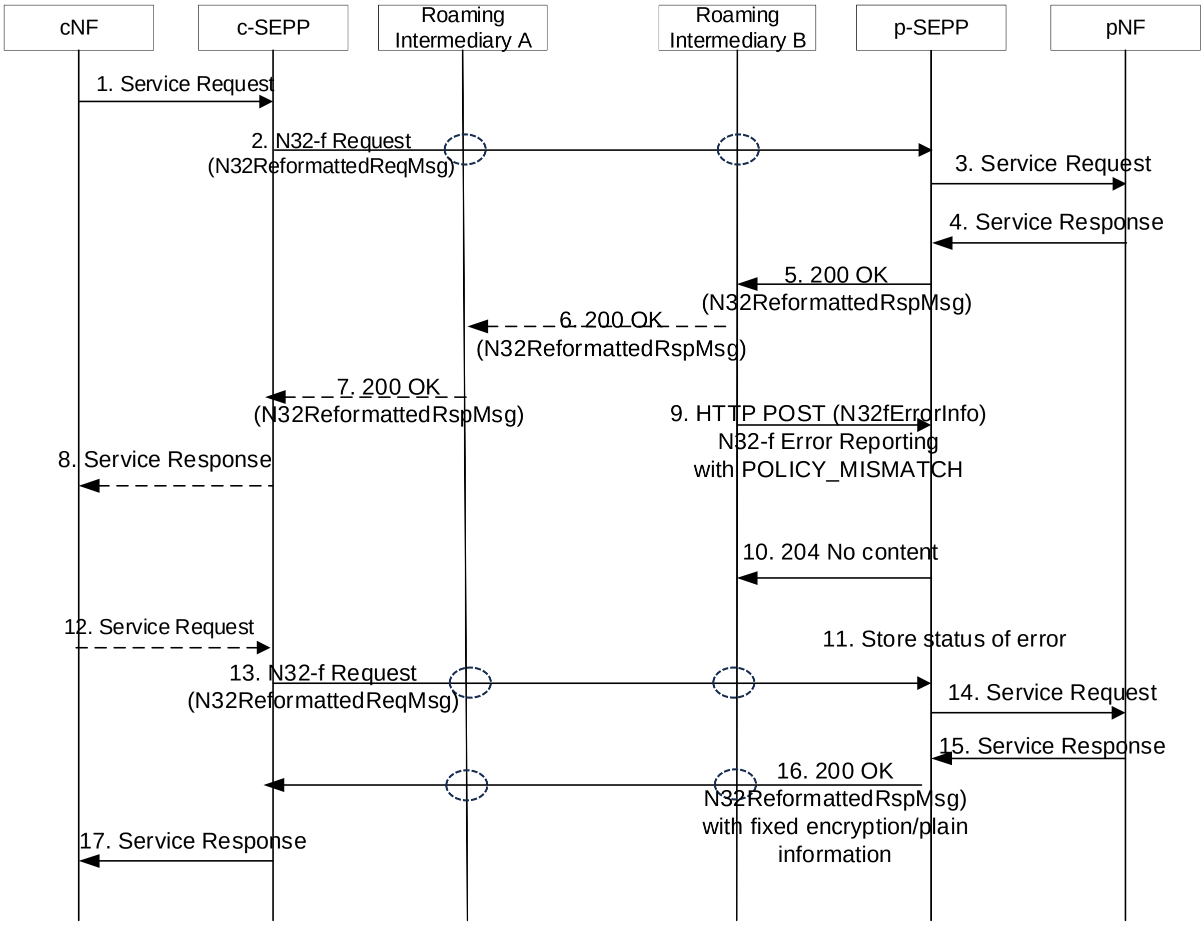
cNF
c-SEPP
Roaming Intermediary A
Roaming Intermediary B
p-SEPP
pNF
1. Service Request
2. N32-f Request (N32ReformattedReqMsg)
3. Service Request
4. Service Response
5. 200 OK (N32ReformattedRspMsg)
6. 200 OK (N32ReformattedRspMsg)
7. 200 OK (N32ReformattedRspMsg)
9. HTTP POST (N32fErrorInfo) N32-f Error Reporting with POLICY_MISMATCH
8. Service Response
10. 204 No content
12. Service Request
11. Store status of error
13. N32-f Request (N32ReformattedReqMsg)
14. Service Request
15. Service Response
16. 200 OK N32ReformattedRspMsg) with fixed encryption/plain information
17. Service Response

Figure 5.5.3.3.1-1: Error message originated by Roaming Intermediary via N32-f Response

1. The c-SEPP receives a service request (HTTP request) message from cNF.

2. The c-SEPP sends an N32-f request using PRINS security (i.e. JOSE protected message) to forward the service request message to the p-SEPP.

3. The pSEPP send the service request to the pNF (see clause 5.3.2.3)

4. The pNF returns the service response (e.g. 200 OK response) to the p-SEPP.

5. The p-SEPP encapsulates the service reponse in an N32-f response (i.e. JOSE protected message) and forwards the message to the c-SEPP (see clause 5.3.2.3).

6-8. As the Roaming Intermediary B detects an N32-f related error (e.g. an IE is received ciphered while it should be in clear), depending on the Roaming Intermediary's policy, the Roaming Intermdiary B may forward the response message (200 OK) encapsulating the service response to the c-SEPP and the c-SEPP sends the Service Response to the cNF.

NOTE: In case the Roaming Intermediary decides not to forward the response message to c-SEPP, NF consumers and NF producers can end up with de-synchronized status in case of a non-safe/idempotent operation. Mechanisms specified for 5GC SBI can be used for handling such situation (e.g. to detect the re-transmitted request).

9. The Roaming Intermediary B sends a new N32-f request encapsulating an N32-c "N32-f Error Reporting request" message towards p-SEPP to report the error, as specified in clause 5.5.3.3.2.

   Upon receipt of an N32-f request encapsulating an N32-c message with a dummy N32-c apiRoot, the receiving N32-f service instance of the p-SEPP looks up for a N32-c service instance that can support the N32-f connection (identified by the N32fContextId received in the N32-f request), substitutes the dummy N32-c apiRoot of the N32-c message with the apiRoot of that N32-c service instance and forwards the N32-c message towards that N32-c service instance.

10. The p-SEPP returns "204 No Content" to the Roaming Intermediary B.

11. The p-SEPP logs the error and, if possible and allowed by local policies, considers it for further N32-f messages the p-SEPP sends towards the c-SEPP (e.g. the p-SEPP may send the reported IE in clear in further messages it forwards towards the c-SEPP).

12. The cNF may repeat its service request in case no response is being received from the c-SEPP.

13. The c-SEPP forwards the (repeated) service request from the cNF, if any. Alternatively, the c-SEPP may resend its N32-f request to the p-SEPP due to no response being received from the p-SEPP.

14. The pSEPP forwards the service request towards the pNF.

15. The pNF returns the service response (e.g. 200 OK response).

16. The p-SEPP encapsulates the service response in an N32-f response (i.e. JOSE protected message) and forwards the message to the c-SEPP, taking into account any error information earlier received from the cSEPP or Roaming Intermediary, if possible and allowed by local policies (e.g. the IE previously reported in error in clear).

17. The c-SEPP send the service response to the cNF.

The procedure is identical if the Roaming Intermediary A detects an error.

5.5.3.3.2 Error message formatting by the Roaming Intermediary

If a Roaming Intermediary needs to generate an N32-f related error message upon receiving an N32-f response, the Roaming Intermediary shall construct a new N32-f request as defined in clause 5.3.2.3 for a SEPP with the following modifications:

- the DataToIntegrityProtectBlock (see Table 6.2.5.2.2-1) shall only contain the MetaData with the N32fContextId and messageId of the N32-f response message for which an error was detected.

- the patch instructions in the modificationBlock (see Table 6.2.5.2.2-1) shall be based on an DataToIntegrityProtectBlock only containing the MetaData with the n32fContextId and messageId.

- the modifications in the "modificationsBlock" shall result in encoding a N32-c request for N32-f Error Reporting, i.e. it shall contain patch instructions:

    - adding the requestLine to form an HTTP POST request "{n32c-apiRoot}/n32c-handshake/v1/n32f-error"; the {n32c-apiRoot} shall be set to a dummy N32-c apiRoot defined as the N32-f apiRoot with the authority part prepended with the label "n32c".

EXAMPLE: If the n32-f apiRoot is https://sepp-n32f.5gc.mnc203.mcc422.3gppnetwork.org:443, the dummy N32-c apiRoot is:

https://n32c.sepp-n32f.5gc.mnc203.mcc422.3gppnetwork.org:443

NOTE: The dummy N32-c apiRoot in the N32-c message encapsulated in the N32-f request needs not contain addressing information of any actual N32-c service instance of the p-SEPP. Instead, this indicates to the receiving N32-f service instance of the p-SEPP that it needs to look up for a N32-c service instance and route the request towards that N32-c service instance. See step 9 of clause 5.5.3.3.1.

    - adding headers, if applicable; and

    - adding the payload that shall be the content of the N32-f Error Reporting Request, i.e N32fErrorInfo.

- the modificationBlock shall contain the JWS signature of the Roaming Intermediary.

The Roaming Intermediary shall then send its N32-f request towards the p-SEPP, (using the same N32-f apiRoot as the one that was used in the original N32-f request sent to the p-SEPP) possibly via another intermediate Roaming Intermediary.

5.5.3.4 Applicative (i.e. SBI related) error determined upon receipt of an N32-f request

5.5.3.4.1 Applicative error originated by Roaming Intermediary via N32-f

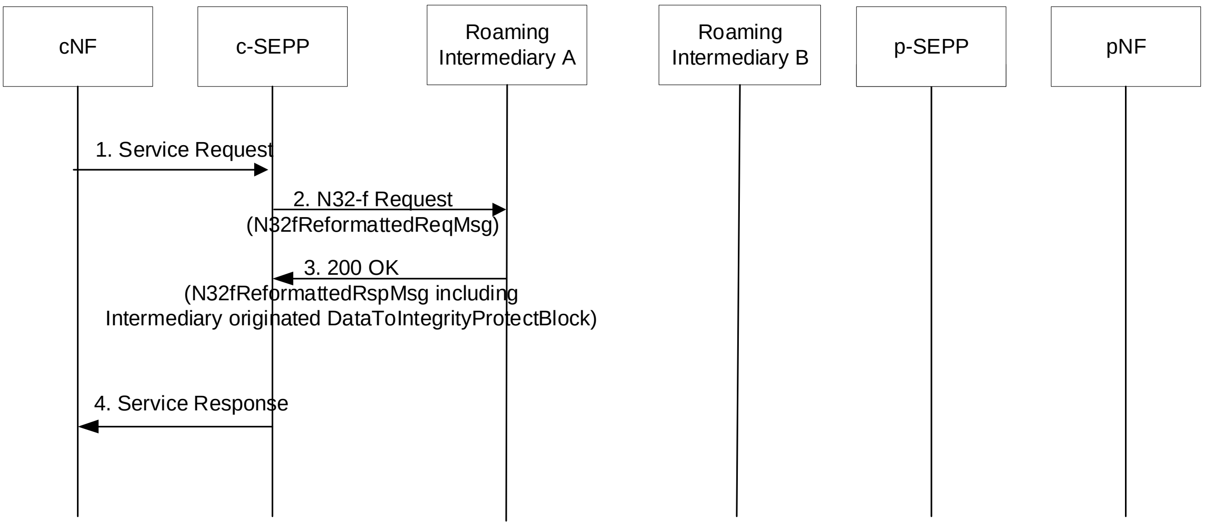
cNF
c-SEPP
Roaming Intermediary A
Roaming Intermediary B
p-SEPP
pNF
1. Service Request
2. N32-f Request (N32fReformattedReqMsg)
3. 200 OK (N32fReformattedRspMsg including Intermediary originated DataToIntegrityProtectBlock)
4. Service Response

Figure 5.5.3.4.1-1: Applicative (i.e. SBI related) error originated by Roaming Intermediary via N32-f

1. The c-SEPP receives a service request (HTTP request) message from cNF.
2. The c-SEPP sends an N32-f request using PRINS security (i.e. JOSE protected message) to forward the service request message to the p-SEPP.
3. The Roaming Intermediary detects an applicative error within the service request encapsulated in the N32-f request, e.g. the UE registration needs to be rejected on behalf of the involved PLMNs. The Roaming Intermediary responds back with a successful N32-f response encapsulating a service error response instead of forwarding the N32-f request to the p-SEPP, as defined in clause 5.5.3.2.
4. The c-SEPP forwards the service error response towards the cNF.

5.5.3.4.2 Error message formatting by the Roaming Intermediary

If a Roaming Intermediary needs to generate a service error message upon receiving an N32-f request, the Roaming Intermediary shall construct a service error response (to be sent within a successful N32-f response) as defined in clause 5.3.2.3 for a SEPP with the following modifications:

- the DataToIntegrityProtectBlock (see Table 6.2.5.2.3-1) shall only contain metadata with N32-f message ID and N32-f context ID of the N32-f request message;
- the patch instructions in the modificationBlock (see Table 6.2.5.2.3-1) shall be based on the intermediary originated DataToIntegrityProtectBlock only containing the MetaData with the N32-f message ID and N32-f context ID.
- the modifications in the "modificationsBlock" shall result in encoding the service error response, i.e. it shall contain patch instructions;
  - adding the statusLine to form the desired service error response (e.g. 403 Forbidden response);
  - adding SBI headers, if applicable; and
  - adding the payload that shall be the content of the service error response (e.g. ProblemDetails with the reason why the registration request is rejected);
- the modificationBlock shall contain the JWS signature of the Roaming Intermediary.

The Roaming Intermediary shall then send its N32-f response towards the c-SEPP, possibly via another intermediate Roaming Intermediary, encapsulating the service error response.

Editor's note: Roaming Intermediary originated error message e.g. to terminate a session is FFS.

5.5.3.5 Handling of applicative events trigger determined by Roaming Intermediary

5.5.3.5.1 Applicative request message originated by Roaming Intermediary via N32-f

The procedure below describes the situation in which Roaming Intermediary detects an event where there are no immediate N32-f messages corresponding to react on. An independent request message is initiated by the Roaming Intermediary.

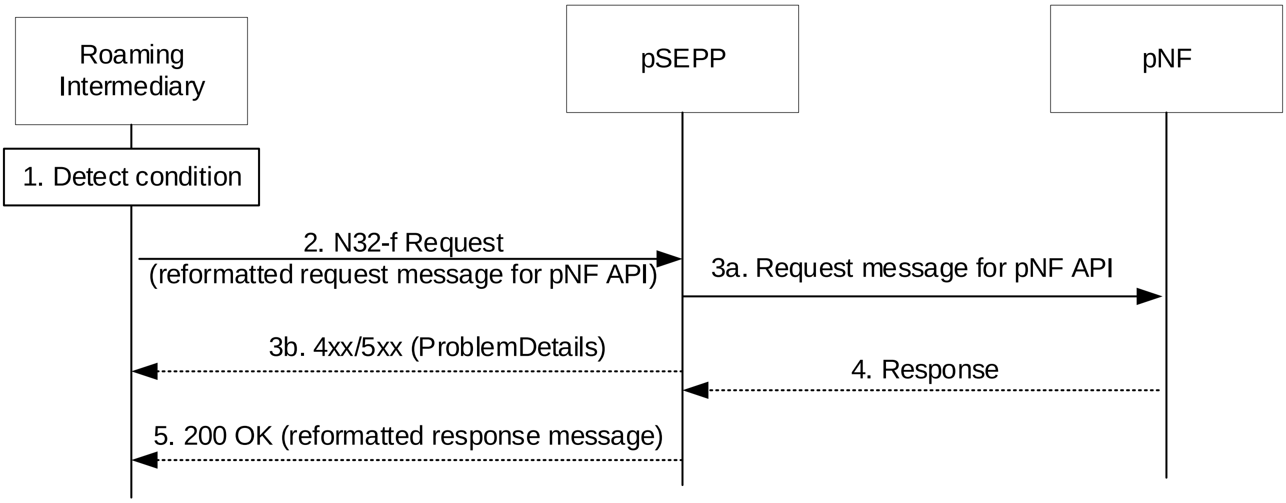
Roaming Intermediary
pSEPP
pNF
1. Detect condition
2. N32-f Request (reformatted request message for pNF API)
3a. Request message for pNF API
3b. 4xx/5xx (ProblemDetails)
4. Response
5. 200 OK (reformatted response message)

Figure 5.5.3.5.1-1: Applicative request message originated by Roaming Intermediary via N32-f

1. The Roaming Intermediary detects an applicative event regarding the status of roaming where the Roaming Intermediary requires to change the status on specific NF.

NOTE 1: How the Roaming Intermediary becomes aware of N32f_contextID is outside the scope of this specification.

2. The Roaming Intermediary shall create a new N32-f request encapsulating a request for the appropriate NF as specified in clause 5.5.3.5.2, and shall send the request towards pSEPP for the message targeted to NF Producer.

Editor's note: It is FFS how to avoid the clash of the message ID created by the Roaming Intermediary and any future messages initiated by the cSEPP.

NOTE 2: The encapsulated message itself is not in scope of N32 definition. Here only the transport is defined, the content is application specific.

3a. The pSEPP shall verify that the Roaming Intermediary is allowed to generate the given request according to operator policy. If allowed by the policy, the pSEPP shall reconstruct the request to be forwarded to the NF and forward the request to the target NF.

3b. If the message is not allowed by the policy as in step 3a, then the responding SEPP shall return an appropriate 4xx/5xx status code together with the "ProblemDetails" in the content of the response.

4. The target NF shall return an appropriate response to the pSEPP.

5. The pSEPP shall forward the response using PRINS security to Roaming Intermediary. Since the Roaming Intermediary is the NF consumer of the request, the response is not forwarded to any other entity from the Roaming Intermediary.

NOTE 3: Step 5 does not prohibit the Roaming Intermediary to initiate any further signalling corresponding to the received HTTP response.

The procedure is analogous if the Roaming Intermediary detects an event that is required on NF beyond the other Roaming Intermediary. In this case the request is routed via the other Roaming Intermediary to the pSEPP.

NOTE 4: As there can only be two Roaming Intermediaries between two SEPPs, the scenario described above refers to the case when the Roaming Intermediary needs to send to the SEPP connected via the other Roaming Intermediary.

5.5.3.5.2 Originated request message formatting by the Roaming Intermediary

If a Roaming Intermediary needs to generate an applicative request message, the Roaming Intermediary shall construct a new N32-f request as defined in clause 5.5.3.3.2, with the difference that the procedure applies to request instead of response. However, the modifications in the "modificationsBlock" shall result in encoding a request for NF API the i.e. it shall contain patch instructions:

- adding the requestLine to form an HTTP POST request for the target NF the request is intended for;
- adding headers, if applicable; and
- adding the content that shall be the content of the requested NF service operation.

NOTE: The present document does not specify how the Roaming Intermediary knows the apiRoot of the NF service.

6 API Definitions

6.1 N32 Handshake API

6.1.1 API URI

The N32 Handshake Procedures shall use the N32 Handshake API.

The API URI of the N32 Handshake API shall be:

{apiRoot}/<apiName>/<apiVersion>

The request URIs used in HTTP requests from the initiating SEPP towards the responding SEPP shall have the Resource URI structure defined in clause 4.4.1 of 3GPP TS 29.501 [5], i.e.:

{apiRoot}/<apiName>/<apiVersion>/<apiSpecificResourceUriPart>

with the following components:

- The {apiRoot} shall be set as described in 3GPP TS 29.501 [5].
- The <apiName> shall be "n32c-handshake".
- The <apiVersion> shall be "v1".
- The <apiSpecificResourceUriPart> shall be set as described in clause 6.1.4.

6.1.2 Usage of HTTP

6.1.2.1 General

HTTP/2, as defined in IETF RFC 9113 [7], shall be used as specified in clause 4.3.2.1.

HTTP/2 shall be transported as specified in clause 4.3.3.

HTTP messages and bodies for the N32 handshake API shall comply with the OpenAPI [27] specification contained in Annex A.

6.1.2.2 HTTP standard headers

6.1.2.2.1 General

The HTTP standard headers as specified in clause 4.3.2.2 shall be supported for this API.

6.1.2.2.2 Content type

The following content types shall be supported:

- the JSON format (see IETF RFC 8259 [8]). The use of the JSON format shall be signalled by the content type "application/json". See also clause 5.3.4.
- the Problem Details JSON Object (see IETF RFC 9457 [22]). The use of the Problem Details JSON object in a HTTP response body shall be signalled by the content type "application/problem+json".

6.1.2.3 HTTP custom headers

6.1.2.3.1 General

In this release of the specification, no specific custom headers are defined for the N32 handshake API.

For 3GPP specific HTTP custom headers used across all service based interfaces, see clause 4.3.2.3.

6.1.3 Resources

6.1.3.1 Overview

There are no resources in this version of the N32 handshake API. All the operations are realized as custom operations without resources.

6.1.4 Custom Operations without Associated Resources

6.1.4.1 Overview

Table 6.1.4.1-1: Custom operations without associated resources

Operation Name	Custom operation URI	Mapped HTTP method	Description
Security Capability Negotiation	/exchange-capability	POST	This is the N32 capability exchange API used to negotiate the security capabilities between SEPPs or tear down the N32-f TLS connection.
Parameter Exchange	/exchange-params	POST	This is the N32 parameter exchange API used to exchange the cipher suites and protection policies.
N32-f Context Terminate	/n32f-terminate	POST	This is the N32-f context termination procedure API.
N32-f Error Reporting	/n32f-error	POST	This is the N32-f error reporting procedure API.

6.1.4.2 Operation: Security Capability Negotiation

6.1.4.2.1 Description

This custom operation is used between the SEPPs to negotiate their security capabilities or to tear down the N32-f connection when negotiated security scheme is TLS. The HTTP method POST shall be used on the following URI:

URI: {apiRoot}/n32c-handshake/<apiVersion>/exchange-capability

This operation shall support the resource URI variables defined in table 6.1.4.2.1-1.

Table 6.1.4.2.1-1: Resource URI variables for this Operation

Name	Data type	Definition
apiRoot	string	See clause 6.1.1.

6.1.4.2.2 Operation Definition

This operation shall support the request data structures and response codes specified in tables 6.2.4.2.2-1 and 6.2.4.2.2-2.

Table 6.1.4.2.2-1: Data structures supported by the POST Request Body

Data type	P	Cardinality	Description
SecNegotiateReqData	M	1	The IE shall contain the security capabilities of the initiating SEPP.

Table 6.1.4.2.2-2: Data structures supported by the POST Response Body on this resource

Data type	P	Cardinality	Response codes	Description
SecNegotiateRspData	M	1	200 OK	This represents the successful processing of the requested security capabilities. The responding SEPP shall provide the security capabilities that it has selected, in the response.
RedirectResponse	O	0..1	307 Temporary Redirect	Temporary redirection. See clause 6.1.8.
ProblemDetails	O	0..1	403 Forbidden	The "cause" attribute may be used to indicate one of the following application errors: - REQUESTED_PURPOSE_NOT_ALLOWED When the receiving SEPP fails to negotiate the security capability, the "cause" attribute shall be set to "NEGOTIATION_NOT_ALLOWED".
ProblemDetails	O	0..1	409 Conflict	The "cause" attribute may be used to indicate one of the following application errors: - N32C_EXCHANGE_CAPABILITY_ONGOING, when the receiving SEPP receives an N32-c exchange capability request from a peer SEPP while it is waiting for an N32-c exchange capability response message from the same peer SEPP as specified in clause 5.2.2.
NOTE: The mandatory HTTP error status codes for the POST method listed in Table 5.2.7.1-1 of 3GPP TS 29.500 [4] other than those specified in the table above also apply, with a ProblemDetails data type (see clause 5.2.7 of 3GPP TS 29.500 [4]).				

Table 6.1.4.2.2-3: Headers supported by the 307 response code on this resource

Name	Data type	P	Cardinality	Description
Location	string	M	1	The URI of the SEPP towards which the request is redirected. See clause 6.1.8.

6.1.4.3 Operation: Parameter Exchange

6.1.4.3.1 Description

This custom operation is used between the SEPPs to exchange the parameters for the N32-f connection. The HTTP method POST shall be used on the following URI:

URI: {apiRoot}/n32c-handshake/<apiVersion>/exchange-params

This operation shall support the resource URI variables defined in table 6.1.4.3.1-1.

Table 6.1.4.3.1-1: Resource URI variables for this Operation

Name	Data type	Definition
apiRoot	string	See clause 6.1.1.

6.1.4.3.2 Operation Definition

This operation shall support the request data structures and response codes specified in tables 6.1.4.3.2-1 and 6.1.4.3.2-2.

Table 6.1.4.3.2-1: Data structures supported by the POST Request Body

Data type	P	Cardinality	Description
SecParamExchReqData	M	1	The IE shall contain the parameters requested by the requesting SEPP.

Table 6.1.4.3.2-2: Data structures supported by the POST Response Body on this resource

Data type	P	Cardinality	Response codes	Description
SecParamExchRspData	M	1	200 OK	This represents the successful processing of the requested parameters. The SEPP shall provide the selected parameters (i.e selected cipher suite and/or selected data type encryption policy) depending on what was requested by the requesting SEPP and what is supported by the responding SEPP, or the SEPP shall provide the modification policy and/or security information lists of the connected IPXs.
ProblemDetails	O	0..1	409 Conflict	The "cause" attribute may be used to indicate one of the following application errors: - REQUESTED_PARAM_MISMATCH - SECURITY_PARAM_EXCHANGE_COLLISION, when the receiving SEPP receives a security parameter exchange request from a peer SEPP while it is waiting for a security parameter exchange response message from the same peer SEPP as specified in clause 5.2.3.2.
NOTE: The mandatory HTTP error status codes for the POST method listed in Table 5.2.7.1-1 of 3GPP TS 29.500 [4] other than those specified in the table above also apply, with a ProblemDetails data type (see clause 5.2.7 of 3GPP TS 29.500 [4]).				

6.1.4.4 Operation: N32-f Context Terminate

6.1.4.4.1 Description

This custom operation is used between the SEPPs to terminate an N32-f context. The HTTP method POST shall be used on the following URI:

URI: {apiRoot}/n32c-handshake/<apiVersion>/n32f-terminate

This operation shall support the resource URI variables defined in table 6.1.4.3.1-1.

Table 6.1.4.4.1-1: Resource URI variables for this Operation

Name	Data type	Definition
apiRoot	string	See clause 6.1.1.

6.1.4.4.2 Operation Definition

This operation shall support the request data structures and response codes specified in tables 6.1.4.4.2-1 and 6.1.4.4.2-2.

Table 6.1.4.4.2-1: Data structures supported by the POST Request Body

Data type	P	Cardinality	Description
N32fContextInfo	M	1	The IE shall contain the information about the N32-f context requested to be terminated by the requesting SEPP.

Table 6.1.4.4.2-2: Data structures supported by the POST Response Body on this resource

Data type	P	Cardinality	Response codes	Description
N32fContextInfo	M	1	200 OK	This represents the successful deletion of the request N32-f context. The responding SEPP shall return the "n32fContextId" it had towards the initiating SEPP, in this IE.
NOTE: The mandatory HTTP error status codes for the POST method listed in Table 5.2.7.1-1 of 3GPP TS 29.500 [4] other than those specified in the table above also apply, with a ProblemDetails data type (see clause 5.2.7 of 3GPP TS 29.500 [4]).				

6.1.4.5 Operation: N32-f Error Reporting

6.1.4.5.1 Description

This custom operation is used between the SEPPs to report errors identified while processing the messages received on N32-f. The HTTP method POST shall be used on the following URI:

URI: **{apiRoot}/n32c-handshake/<apiVersion>/n32f-error**

This operation shall support the resource URI variables defined in table 6.1.4.5.1-1.

Table 6.1.4.5.1-1: Resource URI variables for this Operation

Name	Data type	Definition
apiRoot	string	See clause 6.1.1.

6.1.4.5.2 Operation Definition

This operation shall support the request data structures and response codes specified in tables 6.1.4.5.2-1 and 6.1.4.5.2-2.

Table 6.1.4.5.2-1: Data structures supported by the POST Request Body

Data type	P	Cardinality	Description
N32fErrorInfo	M	1	The IE shall contain the information about the N32-f message that failed to process at the SEPP initiating the N32-f error reporting procedure, together with information related to the nature of the error.

Table 6.1.4.5.2-2: Data structures supported by the POST Response Body on this resource

Data type	P	Cardinality	Response codes	Description
			204 No Content	This represents the successful processing of the N32-f error report at the receiving SEPP.
NOTE: The mandatory HTTP error status codes for the POST method listed in Table 5.2.7.1-1 of 3GPP TS 29.500 [4] other than those specified in the table above also apply, with a ProblemDetails data type (see clause 5.2.7 of 3GPP TS 29.500 [4]).				

6.1.5 Data Model

6.1.5.1 General

This clause specifies the application data model supported by the API.

Table 6.1.5.1-1 specifies the data types defined for the N32 interface.

Table 6.1.5.1-1: N32 specific Data Types

Data type	Clause defined	Description
SecNegotiateReqData	6.1.5.2.2	Defines the security capabilities of a SEPP sent to a receiving SEPP.
SecNegotiateRspData	6.1.5.2.3	Defines the selected security capabilities by a SEPP.
SecurityCapability	6.1.5.3.3	Enumeration of security capabilities.
SecParamExchReqData	6.1.5.2.4	Request data structure for parameter exchange
SecParamExchRspData	6.1.5.2.5	Response data structure for parameter exchange
ProtectionPolicy	6.1.5.2.6	The protection policy to be negotiated between the SEPPs.
ApiIeMapping	6.1.5.2.7	API URI to IE mapping on which the protection policy needs to be applied.
IeInfo	6.1.5.2.8	Protection and modification policy for the IE
ApiSignature	6.1.5.2.9	API URI of the service operation
N32fContextInfo	6.1.5.2.10	N32-f context information
N32fErrorInfo	6.1.5.2.11	N32-f error information.
FailedModificationInfo	6.1.5.2.12	Information on N32-f modifications block that failed to process.
N32fErrorDetail	6.1.5.2.13	Details about the N32f error.
CallbackName	6.1.5.2.14	Callback Name.
IpxProviderSecInfo	6.1.5.2.15	Defines the security information list of an IPX.
IntendedN32Purpose	6.1.5.2.16	Defines the intended N32 establishment purpose.
HttpMethod	6.1.5.3.4	Enumeration of HTTP methods.
IeType	6.1.5.3.5	Enumeration of types of IEs (i.e kind of IE) to specify the protection policy.
IeLocation	6.1.5.3.6	Location of the IE in a HTTP message.
N32fErrorType	6.1.5.3.7	Type of error while processing N32-f message.
FailureReason	6.1.5.3.8	Reason for failure to reconstruct a HTTP/2 message from N32-f message.

Table 6.1.5.1-2 specifies data types re-used by the N32 interface protocol from other specifications, including a reference to their respective specifications and when needed, a short description of their use within the Namf service based interface.

Table 6.1.5.1-2: N32 re-used Data Types

Data type	Reference	Comments
Fqdn	3GPP TS 29.571 [12]	
SupportedFeatures	3GPP TS 29.571 [12]	Used to negotiate the applicability of the optional features defined in table 6.1.7-1.

6.1.5.2 Structured data types

6.1.5.2.1 Introduction

This clause defines the structures to be used in the N32 Handshake API.

6.1.5.2.2 Type: SecNegotiateReqData

Table 6.1.5.2.2-1: Definition of type SecNegotiateReqData

Attribute name	Data type	P	Cardinality	Description
sender	Fqdn	M	1	This IE shall uniquely identify the SEPP that is sending the request. This IE is used to identify and store the negotiated security capability against the right SEPP.
supportedSecCapabilityList	array(SecurityCapability)	M	1..N	This IE shall contain the list of security capabilities that the requesting SEPP supports. To tear down the N32-f TLS connection, this IE shall set SecurityCapability as "NONE".
3GppSbiTargetApiRootSupported	boolean	C	0..1	This IE should be present and indicate that the 3gpp-Sbi-Target-apiRoot HTTP header is supported, if TLS security is supported for N32f message forwarding. When present, it shall indicate if TLS security using the 3gpp-Sbi-Target-apiRoot HTTP header is supported: - true: supported - false (default): not supported (NOTE 1)
plmnIdList	array(PlmnId)	O	1..N	A list of PLMN IDs associated with the SEPP, which is sending the request. The list to be stored by the receiving SEPP in a N32-f Context (see clause 5.9.3 in 3GPP TS 33.501 [6])
snpnIdList	array(PlmnIdNid)	O	1..N	A list of SNPN IDs associated with the SEPP, which is sending the request. The list to be stored by the receiving SEPP in a N32-f Context (see clause 5.9.3 in 3GPP TS 33.501 [6])
targetPlmnId	PlmnId	O	1	When present, this IE shall contain a PLMN ID of the target SEPP. See clause 5.2.2 step 1.
targetSnpnId	PlmnIdNid	O	0..1	When present, this IE shall contain a SNPN ID of the target SEPP. See clause 5.2.2 step 1.
intendedUsagePurpose	array(IntendedN32Purpose)	O	1..N	This attribute notifies the list of requested usage purpose the N32 is established for.
supportedFeatures	SupportedFeatures	C	0..1	This IE shall be present if at least one optional feature defined in clause 6.1.7 is supported
senderN32fFqdn	Fqdn	O	0..1	This IE may be present if the sending SEPP wishes the receiving SEPP to establish the N32-f connection towards a specific FQDN. (NOTE 2)
senderN32fPortList	array(Uinteger)	O	1..N	This IE may be present if the sending SEPP wishes the receiving SEPP to establish the N32-f connection using a specific port number. The N32-f ports list shall contain one port number per security capability encoded in the supportedSecCapabilityList IE and it shall be ordered in the same order as the security capabilities list. For example, if TLS is the first security capability in the supportedSecCapabilityList, then the first N32-f port in the senderN32fPortList shall be for TLS. (NOTE 3)
NOTE 1: The attribute name does not follow the naming conventions specified in 3GPP TS 29.501 [5]. The attribute name is kept though as defined in the current specification for backward compatibility reason. NOTE 2: If the senderN32fFqdn IE is absent, the receiving SEPP establishes the N32-f connection towards the sending SEPP using the N32-c FQDN and/or local configuration. SEPP implementations complying with earlier releases of the specification may not support this IE. NOTE 3: If the senderN32fPortList IE is absent, the receiving SEPP shall use a locally configured port if any, otherwise the default HTTPs port number, i.e., TCP port 443 for "https" URIs as specified in IETF RFC 9113 [7]. SEPP implementations complying with earlier releases of the specification may not support this IE.				

6.1.5.2.3 Type: SecNegotiateRspData

Table 6.1.5.2.3-1: Definition of type SecNegotiateRspData

Attribute name	Data type	P	Cardinality	Description
sender	Fqdn	M	1	This IE shall uniquely identify the SEPP that is sending the response. This IE is used to identify and store the negotiated security capability against the right SEPP.
selectedSecCapability	SecurityCapability	M	1	This IE shall contain the security capability selected by the responding SEPP. When the request is for tearing down the N32-f TLS connection, the responding SEPP shall add SecurityCapability as "NONE".
3GppSbiTargetApiRoot Supported	boolean	C	0..1	This IE should be present and indicate that the 3gpp-Sbi-Target-apiRoot HTTP header is supported, if TLS security is negotiated for N32f message forwarding and the initiating SEPP indicated support of this header. When present, it shall indicate if TLS security using the 3gpp-Sbi-Target-apiRoot HTTP header is supported: - true: supported - false (default): not supported (NOTE 1)
plmnIdList	array(PlmnId)	O	1..N	A list of PLMN IDs of a single PLMN associated with the SEPP, which is sending the response. The list to be stored by the receiving SEPP in a N32-f Context (see clause 5.9.3 in 3GPP TS 33.501 [6]). If different PLMNs are represented by different PLMN IDs supported by a SEPP, then the SEPP shall select the PLMN as specified in clause 5.2.2 step 2a.
snpnIdList	array(PlmnIdNid)	O	1..N	A list of SNPN IDs of a single SNPN associated with the SEPP, which is sending the response. The list to be stored by the receiving SEPP in a N32-f Context (see clause 5.9.3 in 3GPP TS 33.501 [6]). If different SNPNs are represented by different SNPN IDs supported by a SEPP, then the SEPP shall select the SNPN as specified in clause 5.2.2 step 2a.
allowedUsagePurpose	array(IntendedN32Purpose)	O	1..N	This attribute notifies the list of allowed usage purpose the N32 is established for. IntendedN32Purpose shall not include attribute "cause".
rejectedUsagePurpose	array(IntendedN32Purpose)	O	1..N	This attribute notifies the list of rejected usage purpose the N32 is established for. Shall only be present if any of the requested usage purpose is rejected.
supportedFeatures	SupportedFeatures	C	0..1	This IE shall be present if at least one optional feature defined in clause 6.1.7 is supported
senderN32fFqdn	Fqdn	O	0..1	This IE may be present if the sending SEPP wishes the receiving SEPP to establish the N32-f connection towards a specific FQDN. (NOTE 2)
senderN32fPort	Uinteger	O	0..1	This IE may be present if the sending SEPP wishes the receiving SEPP to establish the N32-f connection using a specific port number. (NOTE 3)
NOTE 1: The attribute name does not follow the naming conventions specified in 3GPP TS 29.501 [5]. The attribute name is kept though as defined in the current specification for backward compatibility reason. NOTE 2: If the senderN32fFqdn IE is absent, the receiving SEPP establishes the N32-f connection towards the sending SEPP using the N32-c FQDN and/or local configuration. SEPP implementations complying with earlier releases of the specification may not support this IE. NOTE 3: If the senderN32fPort number is absent, the receiving SEPP shall use a locally configured port, if any, otherwise the default HTTPs port number, i.e., TCP port 443 for "https" URIs as specified in IETF RFC 9113 [7]. SEPP implementations complying with earlier releases of the specification may not support this IE.				

6.1.5.2.4 Type: SecParamExchReqData

Table 6.1.5.2.4-1: Definition of type SecParamExchReqData

Attribute name	Data type	P	Cardinality	Description	Applicability
n32fContextId	string	M	1	This IE shall contain the context identifier to be used by the responding SEPP for subsequent JOSE protected message forwarding procedure over N32-f towards the initiating SEPP. The initiating SEPP shall use this context identifier to locate the cipher suite and protection policy exchanged and agreed to be used with the responding SEPP, for the message forwarding procedure over N32-f. The n32fContextId shall encode a 64-bit integer in hexadecimal representation. Each character in the string shall take a value of "0" to "9" or "A" to "F" and shall represent 4 bits. The most significant character representing the 4 most significant bits of the N32-f context Id shall appear first in the string, and the character representing the 4 least significant bit of the N32-f context Id shall appear last in the string. Pattern: '^[A-Fa-f0-9]{16}$' Example: "0600AD1855BD6007".	
jweCipherSuiteList	array(string)	C	1..N	This IE shall be present during the parameter exchange procedure for cipher suite negotiation (see clause 5.2.3.2). When present, this IE shall contain the ordered list of JWE cipher suites supported by the requesting SEPP. Valid values for the string are as specified in clause 5.1 of IETF RFC 7518 [13].	
jwsCipherSuiteList	array(string)	C	1..N	This IE shall be present during the parameter exchange procedure for cipher suite negotiation (see clause 5.2.3.2). When present, this IE shall contain the ordered list of JWS cipher suites supported by the requesting SEPP. Valid values for the string are as specified in clause 3.1 of IETF RFC 7518 [13].	
protectionPolicyInfo	ProtectionPolicy	C	0..1	Either this IE or the secProfiles IE shall be present during the parameter exchange procedure for protection policy exchange(see clause 5.2.3.3). When present, this IE shall contain the data type encryption policy requested by the requesting SEPP and/or the modification policy supported by the IPX(s) on the side of the requesting SEPP.	
secProfiles	array(string)	C	1..N	Either this IE or the protectionPolicyInfo IE shall be present during the parameter exchange procedure for protection policy exchange(see clause 5.2.3.3). When present, this IE shall indicate the candidate list of security profiles that the initiating SEPP is supporting for PRINS. The list may contain up to 256 profiles.	PSEPRO
ipxProviderSecInfoList	array(IpxProviderSecInfo)	C	1..N	This IE includes the list of IPX security information.	
sender	Fqdn	C	0..1	This IE shall be present if the Parameter Exchange request is sent on a different N32-c HTTP connection than the one used to perform the Security Capability Negotiation procedure. It may be present otherwise. When present, it shall uniquely identify the SEPP that is sending the request. This IE is used to store the exchanged parameters against the right SEPP.	

6.1.5.2.5 Type: SecParamExchRspData

Table 6.1.5.2.5-1: Definition of type SecParamExchRspData

Attribute name	Data type	P	Cardinality	Description	Applicability
n32fContextId	string	M	1	This IE shall contain the context identifier to be used by the initiating SEPP for subsequent JOSE protected message forwarding procedure over N32-f towards the responding SEPP. The responding SEPP shall use this context identifier to locate the cipher suite and protection policy exchanged and agreed to be used with the initiating SEPP, for the message forwarding procedure over N32-f. The n32fContextId shall encode a 64-bit integer in hexadecimal representation. Each character in the string shall take a value of "0" to "9" or "A" to "F" and shall represent 4 bits. The most significant character representing the 4 most significant bits of the N32-f context Id shall appear first in the string, and the character representing the 4 least significant bit of the N32-f context Id shall appear last in the string. Pattern: '^[A-Fa-f0-9]{16}$' Example: "0600AD1855BD6007".	
selectedJweCipherSuite	string	C	1	This IE shall be present during the parameter exchange procedure for cipher suite negotiation (see clause 5.2.3.2). When present, this IE shall contain the JWE cipher suite selected by the responding SEPP.	
selectedJwsCipherSuite	string	C	1	This IE shall be present during the parameter exchange procedure for cipher suite negotiation (see clause 5.2.3.2). When present, this IE shall contain the JWS cipher suite selected by the responding SEPP.	
selProtectionPolicyInfo	ProtectionPolicy	C	0..1	This IE shall be present during the parameter exchange procedure for protection policy exchange (see clause 5.2.3.3) if the initiating SEPP included the protectionPolicyInfo IE in the exchange parameter request message to the responding SEPP. When present, this IE shall contain the data type encryption policy selected by the responding SEPP and/or the modification policy supported by the IPX(s) on the side of the responding SEPP.	
selSecProfiles	array(string)	C	1..N	This IE shall indicate the list of selected security profiles applicable for messages forwarding over N32-f if the initiating SEPP sent a candidate list of security profiles in the exchange parameter request message to the responding SEPP. The list may contain up to 256 profiles.	PSEPRO
ipxProviderSecInfoList	array(IpxProviderSecInfo)	C	1..N	This IE includes the list of IPX security information.	
sender	Fqdn	C	0..1	This IE shall be present if the Parameter Exchange response is sent on a different N32-c HTTP connection than the one used to perform the Security Capability Negotiation procedure. It may be present otherwise. When present, it shall uniquely identify the SEPP that is sending the response. This IE is used to store the exchanged parameters against the right SEPP.	

6.1.5.2.6 Type: ProtectionPolicy

Table 6.1.5.2.6-1: Definition of type ProtectionPolicy

Attribute name	Data type	P	Cardinality	Description
apiIeMappingList	array(ApiIeMapping)	M	1..N	Contains an array of API URI to IE type - IE name mapping. The mapping includes an indication against each IE if that IE is allowed to be modified by the IPX on the side of the SEPP or not.
dataTypeEncPolicy	array(IeType)	C	1..N	This IE shall be present when the SEPPs need to exchange the IE protection policies. When present, this IE shall contain the list of IE types that the SEPP intends to protect by ciphering.

6.1.5.2.7 Type: ApiIeMapping

Table 6.1.5.2.7-1: Definition of type ApiIeMapping

Attribute name	Data type	P	Cardinality	Description
apiSignature	ApiSignature	M	1	This IE shall contain: - The API URI of the NF service operations following request/response semantic; or - The API URI of the subscribe / unsubscribe service operation
apiMethod	HttpMethod	M	1	This IE shall contain the HTTP method used by the API.
IeList	array(IeInfo)	M	1..N	This IE shall contain the array of Ies in the API. (NOTE)
NOTE: The attribute name does not follow the naming conventions specified in 3GPP TS 29.501 [5]. The attribute name is kept though as defined in the current specification for backward compatibility reason.				

6.1.5.2.8 Type: IeInfo

Table 6.1.5.2.8-1: Definition of type IeInfo

Attribute name	Data type	P	Cardinality	Description
ieLoc	IeLocation	M	1	This IE shall contain the location of the IE mentioned in "reqIe" or "rspIe" (i.e Variable in URI path or URI query parameter or HTTP header or JSON body or multipart message)
ieType	IeType	M	1	This IE shall contain the type of the IE, representing the nature of the information the IE is carrying.
reqIe	string	C	0..1	This IE shall be included when the Ies in HTTP/2 request messages of an API need to be protected when forwarded over N32-f. When present, this IE shall contain: - The JSON pointer representation of the IE to be protected, if the "ieLoc" indicates "BODY". Only the JSON pointer of the leaf IEs are included; - The name of the Variable in URI path to be protected, if the "ieLoc" indicates "URI_PATH"; e.g. the name "{ueId}" can be used to protect the UE ID in the following API URI: "/nNf-service/v1/(ueId)/service-operation-1" - The name of the URI query parameter to be protected, if the "ieLoc" indicates "URI_PARAM"; - The name of the HTTP header, if the "ieLoc" indicates "HEADER"; - The JSON pointer representation of the attribute defined with the RefToBinaryData type if the "ieLoc" indicates "MULTIPART_BINARY". It shall be encoded as: <JSON Pointer of the attribute defined with the RefToBinaryData type>/data.
rspIe	string	C	0..1	This IE shall be included when the IEs in HTTP/2 response messages of an API need to be protected when forwarded over N32-f. When present, this IE shall contain: - The JSON pointer representation of the IE to be protected, if the "ieLoc" indicates "BODY". Only the JSON pointer of the leaf IEs are included; - The name of the HTTP header, if the "ieLoc" indicates "HEADER"; - The JSON pointer representation of the attribute defined with the RefToBinaryData type if the "ieLoc" indicates "MULTIPART_BINARY". It shall be encoded as: <JSON Pointer of the attribute defined with the RefToBinaryData type>/data.
isModifiable	boolean	C	0..1	This IE shall be included if the IE is allowed to be modified by all IPX(s) on the side of the SEPP sending the API IE mapping. When present, - true, indicates that the IE is allowed to be modified by all IPX(s) on the side of the SEPP; - false, indicates that the IE is not allowed to be modified by any IPX on the side of the SEPP; - default is false. When the IE is not included, the default value shall be applied. (NOTE)

isModifiableByIpx	map(boolean)	C	0..1	This IE shall be included if the IE is allowed to be modified by some of (but not all) the IPX(s) on the side of the SEPP sending the API IE mapping. The key of the map is the *ipxProviderId* for which the boolean applies. When present, each element carries the isModifiable indication for the IPX indicated by the key. (NOTE)
NOTE: Either isModifiable or isModifiableByIpx may be present, but not both.				

6.1.5.2.9 Type: ApiSignature

Table 6.1.5.2.9-1: Definition of type ApiSignature as a list of "mutually exclusive alternatives"

Data type	Cardinality	Description	Applicability
Uri	1	API URI of a request/response or subscribe/unsubscribe NF service operation as specified in the respective API specification with the variable parts other than {apiVersion} unresolved. Examples: "{apiRoot}/nsmf-pdusession/v1/sm-contexts", for the SMF PDUSession Create SM Context service operation. "{apiRoot}/nsmf-pdusession/v1/sm-contexts/{smContextRef}/modify", for the SMF PDUSession Update SM Context service operation. .	
CallbackName	1	A value identifying the type of callback.	

6.1.5.2.10 Type: N32fContextInfo

Table 6.1.5.2.10-1: Definition of type N32fContextInfo

Attribute name	Data type	P	Cardinality	Description
n32fContextId	string	M	1	This IE shall contain the N32-f context identifier of the receiving SEPP. The n32fContextId shall encode a 64-bit integer in hexadecimal representation. Each character in the string shall take a value of "0" to "9" or "A" to "F" and shall represent 4 bits. The most significant character representing the 4 most significant bits of the N32-f context Id shall appear first in the string, and the character representing the 4 least significant bit of the N32-f context Id shall appear last in the string. Pattern: '^[A-Fa-f0-9]{16}$' Example: "0600AD1855BD6007".

6.1.5.2.11 Type: N32fErrorInfo

Table 6.1.5.2.11-1: Definition of type N32fErrorInfo

Attribute name	Data type	P	Cardinality	Description
n32fMessageId	string	M	1	This IE shall contain the N32-f message identifier received over N32-f (see clause 6.2.5.2.9).
n32fErrorType	N32fErrorType	M	1	This IE shall contain the type of processing error encountered by the SEPP initiating the N32-f error reporting procedure.
n32fContextId	string	C	0..1	This IE shall be present if available. When present, this IE shall contain the n32fContextId of the SEPP receiving N32-f error reporting message, which is exchanged between the SEPPs during the parameter exchange procedure (see clause 5.2.3). The n32fContextId shall encode a 64-bit integer in hexadecimal representation. Each character in the string shall take a value of "0" to "9" or "A" to "F" and shall represent 4 bits. The most significant character representing the 4 most significant bits of the N32-f context Id shall appear first in the string, and the character representing the 4 least significant bit of the N32-f context Id shall appear last in the string. Pattern: '^[A-Fa-f0-9]{16}$' Example: "0600AD1855BD6007".
failedModificationList	array(FailedModificationInfo)	C	1..N	This IE shall be present if the n32ErrorType is "INTEGRITY_CHECK_ON_MODIFICATIONS_FAILED" or "MODIFICATIONS_INSTRUCTIONS_FAILED". When present this IE shall contain a list of FQDNs of the IPX-es whose inserted modifications failed to process at the SEPP initiating the N32-f error reporting procedure, together with the reason for the failure to process.
errorDetailsList	array(N32fErrorDetail)	O	1..N	This IE may be included when the n32ErrorType IE indicates "MESSAGE_RECONSTRUCTION_FAILED ". When present, this IE shall contain a list of JSON pointers to the IEs that failed to process together with the reason for the failure to process that IE.
policyMismatchList	array(InvalidParam)	O	1..N	This IE may be included when n32ErrorType is "POLICY_MISMATCH". When present, this IE shall indicate a list of JSON pointers to the IEs and the type of mismatch. - If the parameter was sent in plain while it should have been encrypted, the value "Parameter shall be encrypted" shall be set as the reason. - If the parameter was sent confidentiality protected when required without confidentially protected, value "Parameter shall not be encrypted" shall be set as the reason.

6.1.5.2.12 Type: FailedModificationInfo

Table 6.1.5.2.12-1: Definition of type FailedModificationInfo

Attribute name	Data type	P	Cardinality	Description
ipxId	Fqdn	M	1	This IE shall identify the IPX.
n32fErrorType	N32fErrorType	M	1	This IE shall contain the type of processing error on the modifications block, encountered by the SEPP initiating the N32-f error reporting procedure. The value shall be one of the following: INTEGRITY_CHECK_ON_MODIFICATIONS_FAILED; MODIFICATIONS_INSTRUCTIONS_FAILED

6.1.5.2.13 Type: N32fErrorDetail

Table 6.1.5.2.13-1: Definition of type N32fErrorDetail

Attribute name	Data type	P	Cardinality	Description
attribute	string	M	1	Contains either a HTTP header name or the JSON pointer of an attribute within the N32-f message that failed to reconstruct. The value shall be one of the values of the iePath attribtue (see clause 6.2.5.2.8) in the received N32-f message.
msgReconstructFailReason	FailureReason	M	1	Indicates the reason for the failure to reconstruct the attribute.

6.1.5.2.14 Type: CallbackName

Table 6.1.5.2.14-1: Definition of type CallbackName

Attribute name	Data type	P	Cardinality	Description
callbackType	string	M	1	This IE shall contain a string identifying the type of callback. The value shall be one of the values specified in 3GPP 29.500 [4], Annex B.

6.1.5.2.15 Type: IpxProviderSecInfo

Table 6.1.5.2.15-1: Definition of type IpxProviderSecInfo

Attribute name	Data type	P	Cardinality	Description
ipxProviderId	Fqdn	M	1	This IE shall uniquely identify the IPX.
rawPublicKeyList	array(string)	C	1..N	This IE includes the list of raw public keys for the IPX. When present, each array item shall contain a raw public key for the IPX, with textual encoding as specified in clause 13 of IETF RFC 7468 [21].
certificateList	array(string)	C	1..N	This IE includes the list of certificates for the IPX. When present, each array item shall contain a certificate for the IPX, with textual encoding as specified in IETF RFC 7468 [21].
NOTE: Either the rawPublicKeyList attribute, or the certificateList attribute, shall be present.				

6.1.5.2.16 Type: IntendedN32Purpose

Table 6.1.5.2.16-1: Definition of type IntendedN32Purpose

Attribute name	Data type	P	Cardinality	Description
usagePurpose	N32Purpose	M	1	This attribute provides the one purpose of the intended N32 establishment.
additionalInfo	String	O	0..1	This attribute provides any additional information necessary to characterize the intention for N32 establishment.
cause	String	O	0..1	This attribute, if present, provided the reason for the reject for the purpose described in other attributes are rejected (e.g. "NO_CONTRACT"). This attribute shall be absent if the intended purpose is allowed.

6.1.5.3 Simple data types and enumerations

6.1.5.3.1 Introduction

This clause defines simple data types and enumerations that can be referenced from data structures defined in the previous clauses.

6.1.5.3.2 Simple data types

The simple data types defined in table 6.1.5.3.2-1 shall be supported.

Table 6.1.5.3.2-1: Simple data types

Type Name	Type Definition	Description

6.1.5.3.3 Enumeration: SecurityCapability

Table 6.1.5.3.3-1: Enumeration SecurityCapability

Enumeration value	Description	Applicability
"TLS"	TLS security.	
"PRINS"	PRotocol for N32 INterconnect Security.	
"NONE"	N32-f TLS connection termination	NFTLST

6.1.5.3.4 Enumeration: HttpMethod

Table 6.1.5.3.4-1: Enumeration HttpMethod

Enumeration value	Description
"GET"	HTTP GET Method.
"PUT"	HTTP PUT Method.
"POST"	HTTP POST Method.
"DELETE"	HTTP DELETE Method.
"PATCH"	HTTP PATCH Method.
"HEAD"	HTTP HEAD Method.
"OPTIONS"	HTTP OPTIONS Method.
"CONNECT"	HTTP CONNECT Method.
"TRACE"	HTTP TRACE Method.

6.1.5.3.5 Enumeration: IeType

Table 6.1.5.3.5-1: Enumeration IeType

Enumeration value	Description
"UEID"	These are IEs which carry the UE identity (i.e. SUPI and GPSI). This also includes the long-lasting identity Charging ID. An example of a UEID IE is gpsi IE defined in 3GPP TS 29.518 [25].
"LOCATION"	These are IEs which carry location information (i.e. cell-id and TAI). An example of a LOCATION IE is ncgi IE defined in 3GPP TS 29.571 [12].
"KEY_MATERIAL"	These are IEs which carry keying material as KSEAF and UPU related information. An example of a KEY_MATERIAL IE is upuInfo IE defined in 3GPP TS 29.503 [26].
"AUTHENTICATION_MATERIAL"	These are IEs which carry authentication material like authentication vectors and EAP payload. An example of an AUTHENTICATION_MATERIAL IE is authenticationVector IE defined in 3GPP TS 29.503 [26].
"AUTHORIZATION_TOKEN"	These are IEs which carry authorization Token. The oauth2 access_token would be of this type. An example of an AUTHORIZATION_TOKEN IE is access_token IE defined in 3GPP TS 29.510 [18].
"OTHER"	These are IEs which do not fall into one of the above types, but they would be considered sensitive, and which protection policies may wish to apply confidentiality protection.
"NONSENSITIVE"	These are IEs which carry information that are not sensitive. A protection policy would not normally encrypt (confidentiality protect) these.

6.1.5.3.6 Enumeration: IeLocation

Table 6.1.5.3.6-1: Enumeration IeLocation

Enumeration value	Description
"URI_PARAM"	IE is located in the URI query parameters.
"HEADER"	IE is located in the HTTP header.
"BODY"	IE is located in the body.
"MULTIPART_BINARY"	IE is located in the message body but encoded as a multipart message information in binary format.
"URI_PATH"	IE is located in the URI path excluding query parameters.

6.1.5.3.7 Enumeration: N32fErrorType

Table 6.1.5.3.7-1: Enumeration N32fErrorType

Enumeration value	Description
"INTEGRITY_CHECK_FAILED"	The integrity check verification on the received N32-f message failed.
"INTEGRITY_CHECK_ON_MODIFICATIONS_FAILED"	The integrity check verification on the modifications block of the received N32-f message failed.
"MODIFICATIONS_INSTRUCTIONS_FAILED"	Failed to apply the JSON patch instructions in the modifications block of the received N32-f message, e.g. the references to encBlockIndex is inserted or relocated by IPX (see clause 5.9.3.2 of 3GPP TS 33.501 [6]).
"DECIPHERING_FAILED"	The deciphering of the encrypted block of the received N32-f message failed.
"MESSAGE_RECONSTRUCTION_FAILED"	The reconstruction of the original HTTP/2 message from the received N32-f message failed.
"CONTEXT_NOT_FOUND"	The n32fContextId is unknown in the receiving SEPP. (NOTE)
"INTEGRITY_KEY_EXPIRED"	The integrity keys in the receiving SEPP have expired.
"ENCRYPTION_KEY_EXPIRED"	The encryption keys in the receiving SEPP have expired.
"POLICY_MISMATCH"	The encryption policy verification on the received N32-f message has failed, e.g. protected IEs are not ciphered, or unprotected IEs are ciphered.
NOTE: This enumeration value is deprecated and shall not be used by N32-f error reporting procedure over the N32-c interface.	

6.1.5.3.8 Enumeration: FailureReason

Table 6.1.5.3.8-1: Enumeration FailureReason

Enumeration value	Description
"INVALID_JSON_POINTER"	The JSON pointer value in iePath attribute (see clause 6.2.5.2.8) is invalid.
"INVALID_INDEX_TO_ENCRYPTED_BLOCK"	The value part of the HttpPayload attribute (see clause 6.2.5.2.8) or HttpHeader attribute (see clause 6.2.5.2.7) is pointing to an invalid index to the encrypted block.
"INVALID_HTTP_HEADER"	The name of the header in the received HttpHeader attribute is invalid.

6.1.5.3.9 Enumeration: N32Purpose

Table 6.1.5.3.9-1: Enumeration N32Purpose

Enumeration value	Description
"ROAMING"	Usage dedicated to roaming
"INTER_PLMN_MOBILITY"	Usage corresponding to any inter-mobility transactions
"SMS_INTERCONNECT"	Usage dedicated to SMS interconnect, e.g. SMS sent between subscribers of two different networks
"ROAMING_TEST"	Usage dedicated to roaming, and allowed only for tests, e.g. to allow traffic for test subscribers/SUPI or sessions
"INTER_PLMN_MOBILITY_TEST"	Usage corresponding to any inter-mobility transactions and allowed only for tests, e.g. to allow traffic for test subscribers/SUPI or sessions
"SMS_INTERCONNECT_TEST"	Usage dedicated to SMS interconnect, e.g. SMS sent between subscribers of two different networks, and allowed only for tests, e.g. to allow traffic for test subscribers/SUPI or sessions
"SNPN_INTERCONNECT"	Usage dedicated to an interconnection with an SNPN
"SNPN_INTERCONNECT_TEST"	Usage corresponding to any interconnection with an SNPN and allowed only for tests, e.g. to allow traffic for test subscribers/SUPI or sessions
"DISASTER_ROAMING"	Usage dedicated to Disaster Roaming (see clause 5.40 of 3GPP TS 23.501 [2]).
"DISASTER_ROAMING_TEST"	Usage dedicated to Disaster Roaming (see clause 5.40 of 3GPP TS 23.501 [2]) and allowed only for tests, e.g. to allow traffic for test subscribers/SUPI or sessions.

6.1.5.4 Binary data

There are no multipart/binary part used on the N32-c API(s) in this release of this specification.

6.1.6 Error Handling

6.1.6.1 General

HTTP error handling shall be supported as specified in clause 5.2.4 of 3GPP TS 29.500 [4].

6.1.6.2 Protocol Errors

Protocol Error Handling shall be supported as specified in clause 5.2.7.2 of 3GPP TS 29.500 [4].

6.1.6.3 Application Errors

The common application errors defined in the Table 5.2.7.2-1 in 3GPP TS 29.500 [4] may also be used for the N32-c Handshake service. The following application errors listed in Table 6.1.6.3-1 are specific for the N32-c Handshake service.

Table 6.1.6.3-1: Application errors

Application Error	HTTP status code	Description
REQUESTED_PARAM_MISMATCH	409 Conflict	This represents a parameter mismatch has been detected by the receiving SEPP, i.e. received data-type encryption or modification policy conflict with the one manually configured for the specific roaming partner, interconnect partner and IPX provider
REQUESTED_PURPOSE_NOT_ALLOWED	403 Forbidden	This represents that all the requested purposes included in the request was rejected by the receiving SEPP.
NEGOTIATION_NOT_ALLOWED	403 Forbidden	This represents a security capability negotiation failure at the receiving SEPP, i.e., the received security capability from the peer SEPP is not configured to be supported at the receiving SEPP.
N32C_EXCHANGE_CAPABILITY_ONGOING	409 Conflict	This represents a security capability negotiation failure at the receiving SEPP, i.e., the SEPP receives an N32-c exchange capability request from a peer SEPP while it is expecting an exchange capability response from the same peer SEPP as specified in clause 5.2.2.
SECURITY_PARAM_EXCHANGE_COLLISION	409 Conflict	This represents a Parameter Exchange Procedure for Cipher Suite Negotiation failure at the receiving SEPP, i.e., the SEPP receives a security parameter exchange request from a peer SEPP while it is expecting a security parameter exchange response from the same peer SEPP as specified in clause 5.2.3.2.

6.1.7 Feature Negotiation

The feature negotiation mechanism specified in clause 6.6 of 3GPP TS 29.500 [4] shall be used to negotiate the optional features applicable between the c-SEPP and the p-SEPP, for the N32 Handshake service, if any.

The c-SEPP shall indicate the optional features it supports for the N32 Handshake service, if any, by including the supportedFeatures attribute in the HTTP POST request message for following service operations:

- Security Capability Negotiation procedure, as specified in clause 5.2.2 to negotiate the security capability;

The p-SEPP shall determine the supported features for the requested network as specified in clause 6.6 of 3GPP TS 29.500 [4] and shall indicate the supported features by including the supportedFeatures attribute in content of the HTTP response for the service operation.

The syntax of the supportedFeatures attribute is defined in clause 5.2.2 of 3GPP TS 29.571 [12].

The following features are defined for the N32 Handshake service.

Table 6.1.7-1: Features of supportedFeatures attribute used by N32 Handshake service

Feature Number	Feature	M/O	Description
1	NFTLST	O	N32-f TLS Connection Termination Support A SEPP that supports this feature shall support handling of Security Capability Negotiation procedure to tear down the N32-f TLS connection as specified in clause 5.2.2).
2	PSEPRO	O	PRINS Security Profiles Support A SEPP that supports this feature shall support the negotiation of security profiles as specified in clause 5.2.3.3.
3	PSIU	M	Protection of Sensitive Information in URI (Path and Query Parameters) A SEPP that complies with this release of the specification shall support this feature, i.e. the protection of sensitive information in URI path and query parameters in HTTP messages to be forwarded.
Feature number: The order number of the feature within the supportedFeatures attribute (starting with 1). Feature: A short name that can be used to refer to the bit and to the feature. M/O: Defines if the implementation of the feature is mandatory ("M") or optional ("O"). Description: A clear textual description of the feature.			

6.1.8 HTTP redirection

An N32 HTTP request may be redirected to a different SEPP service instance located within the same PLMN.

If e.g. a SEPP-A1 in PLMN-A receives a N32 HTTP request from another, e.g. SEPP-B that is in PLMN-B and redirects the request to another SEPP-A2 in PLMN-A, the SEPP-A1 shall send 307 Temporary Redirect response to the SEPP-B and may include a RedirectResponse data structure (see 3GPP TS 29.571 [12]) in the response, where the "cause" attribute shall not be set to "SEPP_REDIRECTION" and the "targetSepp" attribute shall be absent. The Location header shall contain the URI of the SEPP-A2.

NOTE 1: A sender that receives a redirectResponse with the cause "SEPP_REDIRECTION" ignores the Location header as specified in clause 6.10.9.1 in 3GPP TS 29.500 [4], accordingly this cause is not used for redirecting N32 request for which the location header is used to convey the URI of the SEPP to which the request is redirected.

NOTE 2: For non N32 interfaces, if a SEPP receives a service request from a HTTP client e.g. an NF or an SCP, from the same PLMN and redirects the service request to a different SEPP in the same PLMN, the SEPP sends 307 Temporary Redirect or 308 Permanent Redirect response to the HTTP client including a RedirectResponse data structure (see 3GPP TS 29.571 [12]), where the "cause" attribute sets to "SEPP_REDIRECTION" and the "targetSepp" attribute contains the apiRoot of the SEPP towards which the request is redirected. The content of the Location header field is ignored by the receiver. See clause 6.10.9.1 in 3GPP TS 29.500 [4].

6.2 JOSE Protected Message Forwarding API on N32

6.2.1 API URI

The JOSE Protected Message Forwarding Procedure on N32 shall use the JOSE Protected Message Forwarding API on N32-f API.

The API URI of the JOSE Protected Message Forwarding API on N32-f API shall be:

{apiRoot}/<apiName>/<apiVersion>

The request URIs used in HTTP requests from the initiating SEPP towards the responding SEPP shall have the Resource URI structure defined in clause 4.4.1 of 3GPP TS 29.501 [5], i.e.:

{apiRoot}/<apiName>/<apiVersion>/<apiSpecificResourceUriPart>

with the following components:

- The {apiRoot} shall be set as described in 3GPP TS 29.501 [5].
- The <apiName> shall be "n32f-forward".
- The <apiVersion> shall be "v1".
- The <apiSpecificResourceUriPart> shall be set as described in clause 6.2.4.

The apiRoot to use towards a SEPP of the target PLMN shall be configured at the SEPP. The URI scheme of the API shall be "http". The apiName part of the URI shall be as specified here for homogeneity of the API across PLMNs.

6.2.2 Usage of HTTP

6.2.2.1 General

HTTP/2, as defined in IETF RFC 9113 [7], shall be used as specified in clause 4.3.2.1.

HTTP/2 shall be transported as specified in clause 4.3.3.

HTTP messages and bodies for the JOSE protected message forwarding API on N32-f shall comply with the OpenAPI [27] specification contained in Annex A.

6.2.2.2 HTTP standard headers

6.2.2.2.1 General

The HTTP standard headers as specified in clause 4.3.2.2 shall be supported for this API.

6.2.2.2.2 Content type

The following content types shall be supported:

- the JSON format (see IETF RFC 8259 [8]). The use of the JSON format shall be signalled by the content type "application/json". See also clause 5.3.4.
- the Problem Details JSON Object (see IETF RFC 9457 [22]). The use of the Problem Details JSON object in a HTTP response body shall be signalled by the content type "application/problem+json".

6.2.2.2.3 Accept-Encoding

SEPPs and IPX should support gzip coding (see IETF RFC 1952 [23]) in HTTP requests and responses and indicate so in the Accept-Encoding header, as described in clause 5.3.2.1.

6.2.2.3 HTTP custom headers

6.2.2.3.1 General

In this release of the specification, no specific custom headers are defined for the JOSE protected message forwarding API on N32.

For 3GPP specific HTTP custom headers used across all service based interfaces, see clause 4.3.2.3.

6.2.3 Resources

6.2.3.1 Overview

There are no resources in this version of this API. All the operations are realized as custom operations without resources.

6.2.4 Custom Operations without associated resources

6.2.4.1 Overview

Table 6.2.4.1-1: Custom operations without associated resources

Operation Name	Custom operation URI	Mapped HTTP method	Description
JOSE Protected Forwarding	/n32f-process	POST	This is the N32f forwarding API used to forward a reformatted and JOSE protected message to a receiving SEPP.
JOSE Protected Forwarding Options	/n32f-process	OPTIONS	Discover the communication options supported by the next hop (IPX or SEPP) for N32-f message processing.

6.2.4.2 Operation: JOSE Protected Forwarding

6.2.4.2.1 Description

This custom operation is used between the SEPPs to forward the reformatted and JOSE protected HTTP/2 message on N32-f. The HTTP method POST shall be used on the following URI:

URI: **{apiRoot}/n32f-forward/<apiVersion>/n32f-process**

This operation shall support the resource URI variables defined in table 6.1.4.2.1-1.

Table 6.2.4.2.1-1: Resource URI variables for this Operation

Name	Data type	Definition
apiRoot	string	See clause 6.2.1.

6.2.4.2.2 Operation Definition

This operation shall support the request data structures and response codes specified in tables 6.2.4.2.2-1 and 6.2.4.2.2-2.

Table 6.2.4.2.2-1: Data structures supported by the POST Request Body on this resource

Data type	P	Cardinality	Description
N32fReformatted ReqMsg	M	1	This IE shall contain the reformatted HTTP/2 message comprising the plain text part, encrypted information, meta data and modification chain information. See clause 6.2.5.2.2.

Table 6.2.4.2.2-2: Data structures supported by the POST Response Body on this resource

Data type	P	Cardinality	Response codes	Description
N32fReformattedRspMsg	M	1	200 OK	This represents the successful processing of the reformatted JOSE protected message at the responding SEPP. The responding SEPP shall provide the reformatted and JOSE protected content of the corresponding HTTP/2 response message received from the NF service producer, or the HTTP/2 notification response message received from the NF service consumer.

ProblemDetailsMsgForwarding	O	0..1	403 Forbidden	When the receiving SEPP fails to process the reconstructed message due to PLMN ID or SNPN ID verification failure, the "cause" attribute shall be set to "PLMNID_MISMATCH" or "SNPNID_MISMATCH". When the receiving SEPP receives HTTP requests over N32-f with purpose, marked using 3gpp-Sbi-Interplmn-Purpose header as specified in 3GPP TS 29.500 [4], that does not match with any of the purposes exchanged via the Security Capability Negotiation procedure, then the "cause" attribute shall be set to "REQUESTED_PURPOSE_NOT_ALLOWED". When the receiving SEPP fails to process the reconstructed message due to the n32fContextId is unknown, the "cause" attribute shall be set to "CONTEXT_NOT_FOUND". When the receiving SEPP fails to process the reconstructed message, and the error is reported by N32f error reporting procedure as specified in clause 5.2.5, the "cause" attribute shall be set to "UNSPECIFIED". When the Roaming Intermediary or receiving SEPP receives HTTP requests over N32-f and detects an encryption policy mismatch, e.g. protected IEs are not ciphered, or unprotected IEs are ciphered, the "cause" attribute shall be set to "POLICY_MISMATCH". In this case, the ProblemDetails may include the invalidParams attribute indicating which IEs were received ciphered when they were expected to be received in clear, and vice-versa, with the reason attribute for each invalid parameter set to "Parameter shall be encrypted" if the IE was sent without confidentiality protection" or “Parameter shall not be encrypted" if the IE was sent with confidential protection. When the Roaming Intermediary or receiving SEPP receives HTTP requests, but the N32 connection cannot be setup due to contractual reasons, the "cause" attribute shall be set to "NO_CONNECTION_DUE_TO_CONTRACT". When the Roaming Intermediary receives HTTP requests but the N32 connection cannot be setup due to a connectivity issue, the "cause" attribute shall be set to "NO_CONNECTION_DUE_TO_CONNECTIVITY". When the Roaming Intermediary receives HTTP requests over N32-f but the message was not delivered due to contractual reasons, the "cause" attribute shall be set to "MSG_NOT_DELIVERED_DUE_TO_CONTRACT". When the receiving SEPP receives a roaming intermediary generated error reporting request (see clause 5.2.3.3), but the JWS signature of the roaming intermediary is invalid, the "cause" attribute shall be set to "INVALID_RI_JWS_SIGNATURE".
NOTE: The mandatory HTTP error status codes for the POST method listed in Table 5.2.7.1-1 of 3GPP TS 29.500 [4] other than those specified in the table above also apply, with a ProblemDetails data type (see clause 5.2.7 of 3GPP TS 29.500 [4]).				

Table 6.2.4.2.2-3: Headers supported by the POST method on this resource

Name	Data type	P	Cardinality	Description
3gpp-Sbi-Message-Priority	string	O	0..1	3gpp-Sbi-Message-Priority header, defined in 3GPP TS 29.500 [4].

Table 6.2.4.2.2-4: Headers supported by 200 Response Code on this endpoint

Name	Data type	P	Cardinality	Description
3gpp-Sbi-Message-Priority	string	O	0..1	3gpp-Sbi-Message-Priority header, defined in 3GPP TS 29.500 [4].

6.2.4.3 Operation: JOSE Protected Forwarding Options

6.2.4.3.1 Description

This service operation queries the communication options supported by the next hop (IPX or SEPP) for N32-f message processing (see clauses 5.3.2.4 and 5.3.4).

The HTTP method OPTIONS shall be used on the following URI:

URI: **{apiRoot}/n32f-forward/<apiVersion>/n32f-process**

This operation shall support the resource URI variables defined in table 6.2.4.3.1-1.

Table 6.2.4.3.1-1: Resource URI variables for this Operation

Name	Data type	Definition
apiRoot	string	See clause 6.1.1.

6.2.4.3.2 Operation Definition

6.2.4.3.2.1 OPTIONS

This method shall support the URI query parameters specified in table 6.2.4.3.2.1-1.

Table 6.2.4.3.2.1-1: URI query parameters supported by the OPTIONS method

Name	Data type	P	Cardinality	Description
n/a				

This method shall support the request data structures specified in table 6.2.4.3.2.1-2 and the response data structures and response codes specified in table 6.2.4.3.2.1-3.

Table 6.2.4.3.2.1-2: Data structures supported by the OPTIONS Request Body on this resource

Data type	P	Cardinality	Description
n/a			

Table 6.2.4.3.2.1-3: Data structures supported by the OPTIONS Response Body on this resource

Data type	P	Cardinality	Response codes	Description
n/a			204 No Content	
ProblemDetails	O	0..1	405 Method Not Allowed	
ProblemDetails	O	0..1	501 Not Implemented	
NOTE: The mandatory HTTP error status codes for the OPTIONS method listed in Table 5.2.7.1-1 of 3GPP TS 29.500 [4] other than those specified in the table above also apply, with a ProblemDetails data type (see clause 5.2.7 of 3GPP TS 29.500 [4]).				

Table 6.2.4.3.2.1-4: Headers supported by the 204 Response Code on this resource

Name	Data type	P	Cardinality	Description
Accept-Encoding	string	O	0..1	Accept-Encoding, described in IETF RFC 9110 [9]

6.2.5 Data Model

6.2.5.1 General

This clause specifies the application data model supported by the API.

Table 6.2.5.1-1 specifies the data types defined for the N32 interface.

Table 6.2.5.1-1: N32 specific Data Types

Data type	Clause defined	Description
N32fReformattedReqMsg	6.2.5.2.2	Contains the reformatted HTTP/2 request message
N32fReformattedRspMsg	6.2.5.2.3	Contains the reformatted HTTP/2 response message
DataToIntegrityProtectAndCipherBlock	6.2.5.2.4	HTTP header to be encrypted or the value of a JSON attribute to be encrypted
DataToIntegrityProtectBlock	6.2.5.2.5	Data to be integrity protected
RequestLine	6.2.5.2.6	Contains the request line of the HTTP API request being reformatted and forwarded over N32-f
HttpHeader	6.2.5.2.7	Contains the encoding of HTTP headers in the API request / response
HttpPayload	6.2.5.2.8	Contains the encoding of JSON content in the API request / response
MetaData	6.2.5.2.9	Contains the meta data information needed for replay protection
Modifications	6.2.5.2.10	Information on inserting of the modifications entry
FlatJweJson	6.2.5.2.11	Contains the integrity protected reformatted block
FlatJwsJson	6.2.5.2.12	Contains the modification from IPXes on path
IndexToEncryptedValue	6.2.5.2.13	Index to the encrypted value
EncodedHttpHeaderValue	6.2.5.2.14	HTTP header value or index to the HTTP header value

Table 6.2.5.1-2 specifies data types re-used by the N32 interface protocol from other specifications, including a reference to their respective specifications and when needed, a short description of their use within the Namf service based interface.

Table 6.2.5.1-2: N32 re-used Data Types

Data type	Reference	Comments
HttpMethod	6.1.5.3.5	
IeLocation	6.1.5.3.6	
PatchItem	3GPP TS 29.571 [12]	
UriScheme	3GPP TS 29.571 [12]	
Fqdn	3GPP TS 29.571 [12]	

6.2.5.2 Structured data types

6.2.5.2.1 Introduction

This clause defines the structures to be used in the JOSE Protected Message Forwarding API on N32.

6.2.5.2.2 Type: N32fReformattedReqMsg

Table 6.2.5.2.2-1: Definition of type N32fReformattedReqMsg

Attribute name	Data type	P	Cardinality	Description
reformattedData	FlatJweJson	M	1	This IE shall contain the integrity protected reformatted block as well as the ciphered part of the reformatted block of the HTTP/2 request message sent between NF service producer and consumer. The SEPP shall reformat the HTTP/2 request message as: - The part of original HTTP/2 request message headers and the content that needs to be only integrity protected is first reformatted into "DataToIntegrityProtectBlock" and then fed as input for the "aad" parameter of the FlatJweJson after subjecting to BASE64URL encoding. The part of the original HTTP/2 request message headers and content that require integrity protection and ciphering is first reformatted into "DataToIntegrityProtectAndCipherBlock" and then fed as input for JWE ciphering and the JWE ciphered block is then BASE64URL encoded and set into the "ciphertext" parameter of the FlatJweJson.
modificationsBlock	array(FlatJwsJson)	C	1..N	This IE shall be included if the IPXes or Roaming Intermediaries on path are allowed to apply modification policies and if they have any specific modification to be applied on the message contained in the DataToIntegrityProtectBlock. This IE shall also be included if a Roaming Intermediary on the path originates a N32-f Request (see clause 5.5.3).

6.2.5.2.3 Type: N32fReformattedRspMsg

Table 6.2.5.2.3-1: Definition of type N32fReformattedRspMsg

Attribute name	Data type	P	Cardinality	Description
reformattedData	FlatJweJson	M	1	This IE shall contain the integrity protected reformatted block as well as the ciphered part of the reformatted block of the HTTP/2 response message sent between NF service producer and consumer. The SEPP shall reformat the HTTP/2 response message as: - The part of original HTTP/2 response message headers and the content that needs to be only integrity protected is first reformatted into "DataToIntegrityProtectBlock" and then fed as input for the "aad" parameter of the FlatJweJson after subjecting to BASE64URL encoding. - The part of the original HTTP/2 response message headers and content that require integrity protection and ciphering is first reformatted into "DataToIntegrityProtectAndCipherBlock" and then fed as input for JWE ciphering and the JWE ciphered block is then BASE64URL encoded and set into the "ciphertext" parameter of the FlatJweJson.
modificationsBlock	array(FlatJwsJson)	C	1..N	This IE shall be included if the IPXes or the Roaming Intermediaries on path are allowed to apply modification policies and if they have any specific modification to be applied on the message contained in the DataToIntegrityProtectBlock. This IE shall also be included if a Roaming Intermediary on the path originates an N32-f Response (see clause 5.5.3).

6.2.5.2.4 Type: DataToIntegrityProtectAndCipherBlock

Table 6.2.5.2.4-1: Definition of type DataToIntegrityProtectAndCipherBlock

Attribute name	Data type	P	Cardinality	Description
dataToEncrypt	array(Any Type)	M	1..N	This IE shall contain the input for ciphering as a JSON object block containing an array of values with arbitrary types. Each entry of the array shall contain the value of a HTTP header to be encrypted or the value of a JSON attribute to be encrypted.

6.2.5.2.5 Type: DataToIntegrityProtectBlock

Table 6.2.5.2.5-1: Definition of type DataToIntegrityProtectBlock

Attribute name	Data type	P	Cardinality	Description
metaData	MetaData	C	0..1	This IE shall be included if the SEPP encodes additional information for replay protection. When present this IE shall contain the meta data information needed for replay protection. This IE shall also be included if the Roaming Intermediary originates a N32-f Request or an N32-f Response (see clause 5.5.3)
requestLine	RequestLine	C	1	This IE shall be included when a JOSE protected API "request" is forwarded over N32-f. When present, this IE shall contain the request line of the HTTP API request being reformatted and forwarded over N32-f.
statusLine	string	C	0..1	This IE shall be included when a JOSE protected API "response" is forwarded over N32-f. When present, this IE shall contain the status line of the HTTP API response being reformatted and forwarded over N32-f.
headers	array(HttpHeader)	C	1..N	This IE shall be included when a JOSE protected API request / response contains HTTP headers. When present this IE shall contain the encoding of HTTP headers in the API request / response.
payload	array(HttpPayload)	C	1..N	This IE shall be included when a JOSE protected API request / response contains JSON content that needs to be sent in clear text. When present this IE shall contain the encoding of JSON content in the API request / response.

6.2.5.2.6 Type: RequestLine

Table 6.2.5.2.6-1: Definition of type RequestLine

Attribute name	Data type	P	Cardinality	Description
method	HttpMethod	M	1	This IE shall contain the HTTP method of the HTTP request encapsulated in the N32-f Request.
scheme	UriScheme	M	1	This IE shall contain the HTTP scheme of the API.
authority	string	M	1	This IE shall contain the authority part of the URI of the API being invoked.
path	string	M	1	This IE shall contain the path part of the URI of the API being invoked, excluding the query and fragment components. The string value of this IE may contain one or more IndexToEncryptedValue structures (encoded with JSON format) specified in clause 6.2.5.2.13 if there are Variables in URI path to be encrypted.
protocolVersion	string	M	1	This IE shall contain the HTTP protocol version. The version shall be 2 in this release of this specification.
queryFragment	string	C	0..1	This IE shall contain the query component of the URI of the API if available. The string value of this IE may contain one or more IndexToEncryptedValue structures (encoded with JSON format) specified in clause 6.2.5.2.13 if the values of certain URI query parameters are to be encrypted.
pathQueryProtectInd	array(IeLocation)	C	1..2	This IE shall be present when IE(s) with sensitive information in the URI path and/or query parameters were protected with encryption. When present, this IE shall indicate the location(s) of the protected IE(s), i.e. "URI_PATH" and/or "URI_PARAM".

6.2.5.2.7 Type: HttpHeader

Table 6.2.5.2.7-1: Definition of type HttpHeader

Attribute name	Data type	P	Cardinality	Description
header	string	M	1	This IE shall contain the name of the HTTP header to encoded.
value	EncodedHttpHeaderValue	M	1	This IE shall contain the value of the HTTP header. The value of the HTTP header shall be encoded as: - value field of the EncodedHttpHeaderValue structure specified in clause 6.2.5.2.14 if the HTTP header is not to be encrypted. - IndexToEncryptedValue structure specified in clause 6.2.5.2.13 if the value of the HTTP header is to be encrypted.

6.2.5.2.8 Type: HttpPayload

Table 6.2.5.2.8-1: Definition of type HttpPayload

Attribute name	Data type	P	Cardinality	Description
iePath	string	M	1	This IE identifies the JSON pointer representation (see IETF RFC 6901 [17]) of full JSON path of the IE to be encoded. IEs that are of type object shall be flattened into each individual attribute's full JSON path and the HttpPayload IE shall only contain the final leaf attribute IE path and its corresponding value.
ieValueLocation	IeLocation	M	1	This IE shall identify where the IE value is located - i,e in the JSON body or in the multipart message part.

value	object	M	1	This IE shall contain the value of the IE corresponding to "iePath", encoded as a free form object.
				If the value of this IE is encrypted, then the value part shall be encoded as
				{
				"encBlockIndex": <array index in DataToIntegrityProtectAndCipherBlock>
				}
				(see clause 6.2.5.2.4).
				
				If the value of this IE is a RefToBinary data type (see 3GPP TS 29.571 [12], then value shall contain the value of the Content-ID header field of the referenced binary body part.
				
				The referenced binary body part of the multipart/related message shall be either encrypted or not encrypted depending on the protection policy exchanged between the SEPPs.
				
				If the referenced binary body part is required to be encrypted, then the binary part is first base64 encoded into a byte array and then inserted into the "DataToIntegrityProtectAndCipherBlock". Then two HttpPayload instances with the following values shall be added immediately after this HttpPayload instance in the "DataToIntegrityProtectBlock"
				{
				  "iePath": <JSON Pointer of the attribute defined with theRefToBinaryData type >/contenttype
				  "ieValueLocation": "MULTIPART_BINARY"
				  "value": <value of the content type of multipart binary>
				},
				{
				  "iePath": <JSON Pointer of the attribute defined with the RefToBinaryData type>/data,
				  "ieValueLocation": "MULTIPART_BINARY"
				"value": {"encBlockIndex": <array index in DataToIntegrityProtectAndCipherBlock that contains the byte array>}
				}
				
				If the referenced binary body part is not required to be encrypted, then the binary part is first base64 encoded into a byte array and then inserted as new instance of HttpPayload IE in " DataToIntegrityProtectBlock" as
				
				{
				  "iePath": <JSON Pointer of the attribute defined with the RefToBinaryData type>/contenttype
				  "ieValueLocation": "MULTIPART_BINARY"
				  "value": <value of the content type of multipart binary>
				},
				{
				  "iePath": <JSON path of the attribute defined with the RefToBinaryData type>/data,
				  "ieValueLocation": "MULTIPART_BINARY"
				"value": <base64 encoded byte array>
				}
				
				See NOTE 1.

NOTE 1: In this release of this specification only N16 interface has binary content and there is no sensitive information carried over N16 interface. Consequently ciphering of binary part is not required in this release of this specification. The encoding specified here is to provide a N32-f framework in a future proof manner so that if a binary part need to be encrypted in future this structure can be used.				

6.2.5.2.9 Type: MetaData

Table 6.2.5.2.9-1: Definition of type MetaData

Attribute name	Data type	P	Cardinality	Description
n32fContextId	string	M	1	This IE shall contain the n32fContextId of the SEPP receiving the message, which is exchanged between the SEPPs during the parameter exchange procedure (see clause 5.2.3). The n32fContextId shall encode a 64-bit integer in hexadecimal representation. Each character in the string shall take a value of "0" to "9" or "A" to "F" and shall represent 4 bits. The most significant character representing the 4 most significant bits of the N32-f context Id shall appear first in the string, and the character representing the 4 least significant bit of the N32-f context Id shall appear last in the string. Pattern: '^[A-Fa-f0-9]{16}$' Example: "0600AD1855BD6007".
messageId	string	M	1	This IE identifies a particular request that is transformed by the SEPP. The value of this IE shall be encoded in hexadecimal representation of a 64 bit integer. This identifier is used in the N32-f error reporting procedure as specified in clause 6.1.4.5. Pattern: ^[a-fA-F0-9]{1, 16}$
authorizedIpxId	string	M	1	This IE identifies the first hop IPX or the Roaming Intermediary that is authorized to insert modifications block. The identifier of the IPX shall be an FQDN. When there is no IPX that's authorized to update, the value of this IE is set to the string "NULL". This IE shall be set to the empty string "", if the Roaming Intermediary originates a message as defined in clause 5.5.3.

6.2.5.2.10 Type: Modifications

Table 6.2.5.2.10-1: Definition of type Modifications

Attribute name	Data type	P	Cardinality	Description
operations	array(PatchItem)	C	1..N	This IE shall be included if an intermediary IPX inserts modification instructions on the JSON data carried in the "DataToIntegrityProtectBlock" part of the N32-f forwarded message. For the first modifications entry, this IE shall not be included, since the first entry is inserted by the SEPP. This IE shall also be included if a Roaming Intermediary originates a message as defined in clause 5.5.3.
identity	Fqdn	M	1	This IE shall contain the identity of the entity inserting the modifications entry. The identity shall be encoded in the form of an URI.
tag	string	C	0..1	This IE shall be present when the JWE Authentication Tag value is non-empty as specified in IETF RFC 7515 [16]. When present, this IE shall contain the BASE64URL(JWE Authentication Tag).

6.2.5.2.11 Type: FlatJweJson

Table 6.2.5.2.11-1: Definition of type FlatJweJson

Attribute name	Data type	P	Cardinality	Description
protected	string	C	0..1	This IE shall be present if there is a JWE Protected Header part of the JOSE header to encode as specified in IETF RFC 7516 [14]. When present, this IE shall contain the BASE64URL(UTF8(JWE Protected Header)) encoding of the JWE protected header.
unprotected	object	C	0..1	This IE shall be present if there is a JWE unprotected header part of the JOSE header that is shared across recipients, to encode as specified in IETF RFC 7515 [16]. This value is represented as an unencoded free form JSON object, rather than as a string. These Header Parameter values are not integrity protected.
header	object	C	0..1	This IE shall be present if there is a JWE unprotected header part of the JOSE header that is specific for the recipient, to encode as specified in IETF RFC 7515 [16]. This value is represented as an unencoded free form JSON object, rather than as a string. These Header Parameter values are not integrity protected.
encrypted_key	string	C	0..1	This IE shall be present when the JWE Encrypted Key for the recipient is non empty. When present this IE shall contain BASE64URL(JWE Encrypted Key). (NOTE)
aad	string	C	0..1	This IE shall be present when the JWE AAD value is non-empty as specified in IETF RFC 7515 [16]. When present, this IE shall contain BASE64URL encoding of the DataToIntegrityProtectBlock JSON object (see clause 6.2.5.2.5).
iv	string	C	0..1	This IE shall be present when the JWE Initialization Vector is non-empty as specified in IETF RFC 7515 [16]. When present, this IE shall contain the BASE64URL(JWE Initialization Vector).
ciphertext	string	M	1	This IE shall contain BASE64URL(JWE Ciphertext). The input for JWE ciphering is the DataToIntegrityProtecAndCiphertBlock (see clause 6.2.5.2.5).
tag	string	C	0..1	This IE shall be present when the JWE Authentication Tag value is non-empty as specified in IETF RFC 7515 [16]. When present, this IE shall contain the BASE64URL(JWE Authentication Tag).
NOTE: The attribute name does not follow the naming conventions specified in 3GPP TS 29.501 [5]. The attribute name is kept though as defined in the current specification for backward compatibility reason.				

6.2.5.2.12 Type: FlatJwsJson

Table 6.2.5.2.12-1: Definition of type FlatJwsJson

Attribute name	Data type	P	Cardinality	Description
payload	string	M	1	This IE shall contain the BASE64URL encoding of the Modifications JSON object (see clause 6.2.5.2.10).
protected	string	C	0..1	This IE shall be present if there is a JWS Protected Header part of the JOSE header to encode as specified in IETF RFC 7515 [16]. When present, this IE shall contain the BASE64URL(UTF8(JWS Protected Header)) encoding of the JWS protected header.
header	object	C	0..1	This IE shall be present if there is a JWS unprotected header part of the JOSE header to encode as specified in IETF RFC 7515 [16]. This value is represented as an unencoded free form JSON object, rather than as a string. These Header Parameter values are not integrity protected.
signature	string	M	1	This IE shall contain the BASE64URL encoded value of the calculated JWS signature.

6.2.5.2.13 Type: IndexToEncryptedValue

Table 6.2.5.2.13-1: Definition of type IndexToEncryptedHttpHeader

Attribute name	Data type	P	Cardinality	Description
encBlockIndex	Uinteger	M	1	Index to the value in DataToIntegrityProtectAndCipherBlock

6.2.5.2.14 Type: EncodedHttpHeaderValue

Table 6.2.5.2.14-1: Definition of type EncodedHttpHeaderValue as a list of "mutually exclusive alternatives"

Data type	Cardinality	Description	Applicability
string	1	HTTP header value.	
IndexToEncryptedValue	1	Index to encrypted HTTP header in the DataToIntegrityProtectAndCipherBlock	

6.2.5.2.15 Type: ProblemDetailsMsgForwarding

Table 6.2.5.2.15-1: Definition of type ProblemDetailsMsgForwarding as a list of to be combined data types

Data type	Cardinality	Description	Applicability
ProblemDetails	1		
AdditionInfoMsgForwarding	1		

6.2.5.2.16 Type: AdditionInfoMsgForwarding

Table 6.2.5.2.16-1: Definition of type AdditionInfoMsgForwarding

Attribute name	Data type	P	Cardinality	Description
suggestedStatusCode	integer	O	0..1	When present, this IE shall indicate a status code that is suggested to be sent to cNF if the cSEPP cannot or does not resend the N32-f request taking into account the N32-f error information.
suggestedProblemDetails	ProblemDetails	O	0..1	When present, this IE shall indicate suggested ProblemDetails to be sent to cNF if the cSEPP cannot or does not resend the N32-f request taking into account the N32-f error information.

6.2.5.3 Simple data types and enumerations

6.2.5.3.1 Introduction

This clause defines simple data types and enumerations that can be referenced from data structures defined in the previous clauses.

6.2.5.3.2 Simple data types

The simple data types defined in table 6.1.5.3.2-1 shall be supported.

Table 6.2.5.3.2-1: Simple data types

Type Name	Type Definition	Description

6.2.5.3.3 Void

6.2.5.3.4 Void

6.2.6 Error Handling

6.2.6.1 General

HTTP error handling shall be supported as specified in clause 5.2.4 of 3GPP TS 29.500 [4].

6.2.6.2 Protocol Errors

Protocol Error Handling shall be supported as specified in clause 5.2.7.2 of 3GPP TS 29.500 [4].

6.2.6.3 Application Errors

The application errors defined for the JOSE protected message forwarding API on N32-f are listed in Table 6.2.6.3-1.

Table 6.2.6.3-1: Application errors

Application Error	HTTP status code	Description
PLMNID_MISMATCH	403 Forbidden	The PLMN ID in the Bearer token carried in the "Authorization" header of the reconstructed message does not match the PLMN ID of the N32-f context.
SNPNID_MISMATCH	403 Forbidden	The SNPN ID in the Bearer token carried in the "Authorization" header of the reconstructed message does not match the SNPN ID of the N32-f context.
REQUESTED_PURPOSE_NOT_ALLOWED	403 Forbidden	The purpose indicated in 3gpp-Sbi-Interplmn-Purpose header as specified in 3GPP TS 29.500 [4] of the reconstructed message does not match with any of the purposes exchanged via the Security Capability Negotiation procedure.
CONTEXT_NOT_FOUND	403 Forbidden	The n32fContextId is unknown in the receiving SEPP.
UNSPECIFIED	403 Forbidden	The receiving SEPP fails to process the reconstructed message, and the error is reported by N32f error reporting procedure as specified in clause 5.2.5.
POLICY_MISMATCH	403 Forbidden	The encryption policy verification on the received N32-f message has failed, e.g. protected IEs are not ciphered, or unprotected IEs are ciphered.
NO_CONNECTION_DUE_TO_CONTRACT	403 Forbidden	The message failed to be delivered as N32 connection cannot be setup due to contractual reasons.
NO_CONNECTION_DUE_TO_CONNECTIVITY	403 Forbidden	The message failed to be delivered as N32 connection cannot be setup due to a connectivity issue.
MSG_NOT_DELIVERED_DUE_TO_CONTRACT	403 Forbidden	The message was not delivered due to contractual reasons.
INVALID_RI_JWS_SIGNATURE	403 Forbidden	The error reporting request generated by the roaming intermediary was not delivered because the JWS signature of the roaming intermediary is invalid.

6.3 Nsepp_Telescopic_FQDN_Mapping API

6.3.1 API URI

The Nsepp_Telescopic_FQDN_Mapping Service shall use the SEPP Telescopic FQDN Mapping API.

The API URI of the SEPP Telescopic FQDN Mapping API shall be:

{apiRoot}/<apiName>/<apiVersion>

The request URIs used in HTTP requests from the NF service consumer towards the SEPP shall have the Resource URI structure defined in clause 4.4.1 of 3GPP TS 29.501 [5], i.e.:

{apiRoot}/<apiName>/<apiVersion>/<apiSpecificResourceUriPart>

with the following components:

- The {apiRoot} shall be set as described in 3GPP TS 29.501 [5].
- The <apiName> shall be "nsepp-telescopic".
- The <apiVersion> shall be "v1".
- The <apiSpecificResourceUriPart> shall be set as described in clause 6.3.3.

6.3.2 Usage of HTTP

6.3.2.1 General

HTTP/2, as defined in IETF RFC 9113 [7], shall be used as specified in clause 5 of 3GPP TS 29.500 [4].

HTTP/2 shall be transported as specified in clause 5.3 of 3GPP TS 29.500 [4].

HTTP messages and bodies for the Nsepp_Telescopic_FQDN_Mapping service shall comply with the OpenAPI [27] specification contained in Annex A.

6.3.2.2 HTTP standard headers

6.3.2.2.1 General

The HTTP standard headers as specified in clause 4.3.2.2 shall be supported for this API.

6.3.2.2.2 Content type

The following content types shall be supported:

- JSON, as defined in IETF RFC 8259 [9]. The use of the JSON format shall be signalled by the content type "application/json". See also clause 5.4 of 3GPP TS 29.500 [4].
- The Problem Details JSON Object (IETF RFC 9457 [22]. The use of the Problem Details JSON object in a HTTP response body shall be signalled by the content type "application/problem+json".

6.3.2.3 HTTP custom headers

6.3.2.3.1 General

In this release of this specification, no custom headers specific to the Nsepp_Telescopic_FQDN_Mapping service are defined. For 3GPP specific HTTP custom headers used across all service-based interfaces, see clause 5.2.3 of 3GPP TS 29.500 [4].

6.3.3 Resources

6.3.3.1 Overview

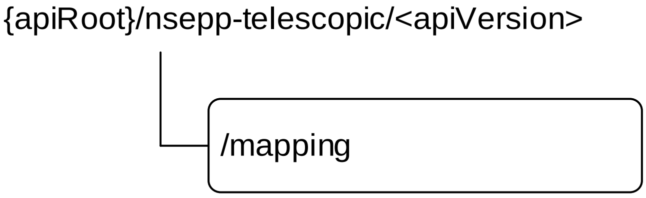

Figure 6.3.3.1-1: Resource URI structure of the nsepp-telescopic API

Table 6.3.3.1-1 provides an overview of the resources and applicable HTTP methods.

Table 6.3.3.1-1: Resources and methods overview

Resource name	Resource URI	HTTP method or custom operation	Description
Mapping	/mapping	GET	Retrieve the mapping between the FQDN in a foreign PLMN and a telescopic FQDN, or viceversa.

6.3.3.2 Resource: Mapping

6.3.3.2.1 Description

This resource represents the mapping between the FQDN of an NF in a foreign PLMN and a telescopic FQDN.

6.3.3.2.2 Resource Definition

Resource URI: {apiRoot}/nsepp-telescopic/<apiVersion>/mapping

This resource shall support the resource URI variables defined in table 6.3.3.2.2-1.

Table 6.3.3.2.2-1: Resource URI variables for this resource

Name	Data type	Definition
apiRoot	string	See clause 6.3.1

6.3.3.2.3 Resource Standard Methods

6.3.3.2.3.1 GET

This method shall support the URI query parameters specified in table 6.3.3.2.3.1-1.

Table 6.3.3.2.3.1-1: URI query parameters supported by the GET method on this resource

Name	Data type	P	Cardinality	Description
foreign-fqdn	Fqdn	O	0..1	This parameter shall contain the FQDN of the NF in the foreign network, that needs to be flattened to a telescopic FQDN in the local network (i.e. an FQDN that points to the local SEPP).
telescopic-label	string	O	0..1	This parameter shall contain the first label used in a telescopic FQDN (i.e. an FQDN that points to the local SEPP) that needs to be mapped to an NF in the foreign network.
NOTE: The parameters "foreign-fqdn" and "telescopic-label" shall not be present simultaneously.				

This method shall support the request data structures specified in table 6.3.3.2.3.1-2 and the response data structures and response codes specified in table 6.3.3.2.3.1-3.

Table 6.3.3.2.3.1-2: Data structures supported by the GET Request Body on this resource

Data type	P	Cardinality	Description
n/a			

Table 6.3.3.2.3.1-3: Data structures supported by the GET Response Body on this resource

Data type	P	Cardinality	Response codes	Description
TelescopicMapping	M	1	200 OK	Upon success, a response body containing a TelescopicMapping object shall be returned
ProblemDetails	O	0..1	404 Not Found	The mapping between a foreign FQDN and a telescopic FQDN could not be found.

6.3.4 Data Model

6.3.4.1 General

This clause specifies the application data model supported by the API.

Table 6.3.4.1-1 specifies the data types defined for the Nsepp_Telescopic_FQDN_Mapping service-based interface protocol.

Table 6.3.4.1-1: Nsepp_Telescopic_FQDN_Mapping specific Data Types

Data type	Clause defined	Description
TelescopicMapping	6.3.4.2.2	Contains the Telescopic mapping data

Table 6.3.4.1-2 specifies data types re-used by the Nsepp_Telescopic_Mapping service-based interface protocol from other specifications.

Table 6.3.4.1-2: Nsepp_Telescopic_FQDN_Mapping re-used Data Types

Data type	Reference	Comments
Fqdn	3GPP TS 29.571 [12]	
ProblemDetails	3GPP TS 29.571 [12]	Common data type for error responses

6.3.4.2 Structured data types

6.3.4.2.1 Introduction

This clause defines the structures to be used in resource representations.

6.3.4.2.2 Type: TelescopicMapping

Table 6.3.4.2.2-1: Definition of type TelescopicMapping

Attribute name	Data type	P	Cardinality	Description
telescopicLabel	string	C	0..1	This parameter shall contain the first label to be used in a telescopic FQDN (i.e. an FQDN that points to the local SEPP) that corresponds to a given NF in the foreign network. In a successful response, this parameter shall be included when the query parameter "foreign-fqdn" is present in the request.
seppDomain	Fqdn	C	0..1	This parameter shall contain the FQDN of the domain of the local SEPP that needs to be appended after the "telescopicLabel" to compose the complete flattened telescopic FQDN. In a successful response, this parameter shall be included when the query parameter "foreign-fqdn" is present in the request.
foreignFqdn	Fqdn	C	0..1	This parameter shall contain the FQDN of the NF in the foreign network. In a successful response, this parameter shall be included when the query parameter "telescopic-label" is present in the request.

6.3.4.3 Simple data types and enumerations

6.3.4.3.1 Introduction

This clause defines simple data types and enumerations that can be referenced from data structures defined in the previous clauses.

6.3.4.3.2 Simple data types

The simple data types defined in table 6.3.4.3.2-1 shall be supported.

Table 6.3.4.3.2-1: Simple data types

Type Name	Type Definition	Description

6.3.5 Error Handling

6.3.5.1 General

HTTP error handling shall be supported as specified in clause 5.2.4 of 3GPP TS 29.500 [4].

6.3.5.2 Protocol Errors

Protocol Error Handling shall be supported as specified in clause 5.2.7 of 3GPP TS 29.500 [4].

6.3.5.3 Application Errors

The common application errors defined in the Table 5.2.7.2-1 in 3GPP TS 29.500 [4] may also be used for the Nsepp_Telescopic_Mapping service, and the following application errors listed in Table 6.3.5.3-1 are specific for the Nsepp_Telescopic_Mapping service.

Table 6.3.5.3-1: Application errors

Application Error	HTTP status code	Description

6.3.6 Feature Negotiation

This API does not currently specify any features.

6.3.7 Security

6.3.7.1 General

This API shall be accessed only from NFs in the same PLMN as the SEPP; it shall not be exposed externally to NFs from another PLMN.

7 Usage of HTTP CONNECT for N32-c connection establishment via Roaming Intermediaries

7.1 General

HTTP/2, as defined in IETF RFC 9113 [7], shall be used.

The Transmission Control Protocol as described in IETF RFC 793 [11] shall be used as transport protocol as required by HTTP/2 (see IETF RFC 9113 [7]).

The HTTP CONNECT method shall be used as defined in clause 9.3.6 of IETF RFC 9110 [9] and clause 8.5 of IETF RFC 9113 [7].

The ":authority" pseudo-header field in the HTTP/2 CONNECT request message shall be set to:

":authority" = uri-host [":" port] as specified in clause 8.3.1 of IETF RFC 9113 [7].

The Uri-host shall indicate the FQDN of the p-SEPP.

The HTTP CONNECT request and the 200 OK status response shall not contain any content.

7.2 HTTP standards headers

The HTTP standard headers defined in Table 7.2-1 shall be supported on the interface between the SEPP and Roaming Intermediary.

Mandatory to support HTTP standard headers does not mean that all the HTTP requests and responses carry the identified request and response headers respectively. It only means it is mandatory to support the processing of the identified headers in request and response message.

Table 7.2-1: Mandatory to support HTTP response standard headers

Name	Reference	Description
Via	IETF RFC 9110 [9]	This header should be inserted by a Roaming Intermediary when relaying an HTTP error response (see clause 6.10.8 of 3GPP TS 29.500 [4]). It may be inserted when relaying other HTTP responses. When inserted by a Roaming Intermediary, the received-protocol portion of the header field value should be set to "HTTP/2.0" or "2.0" and the received-by portion of the header field value should be formatted as follows: - "RI-<RI FQDN>" for a Roaming Intermediary
Server	IETF RFC 9110 [9]	This header should be inserted by the originator of an HTTP error response (see clause 6.10.8 of 3GPP TS 29.500 [4]). It may be inserted otherwise. When inserted by a Roaming Intermediary, the pattern of the header should be formatted as follows: - "RI-<RI FQDN>" for a Roaming Intermediary
NOTE: The inclusion of the Via and Server header in an HTTP CONNECT error response message enables the receiving SEPP to determine which Roaming Intermediary has rejected the request in scenarios with two roaming intermediaries between the SEPPs.		

7.3 HTTP custom headers

7.3.1 3gpp-Connect-Req-Info

The header enables to convey information in the HTTP CONNECT request to the Roaming Intermediary, that may be used by the Roaming Intermediary to determine whether to allow the establishment of the N32-c connection and/or for troiuble-shooting.

The encoding of the header follows the ABNF as defined in IETF RFC 9110 [9].

```
Connect-Req-Info-Header = "3gpp-Connect-Req-Info:" OWS "connect-purpose=" OWS connect-purpose-
value";" OWS orig-network-id ";" OWS sender-fqdn [";" OWS intended-n32-purposes] *( ";" OWS req-
param )

connect-purpose-value = "n32c" / token

orig-network-id = "originating-network-id=" OWS 3DIGIT "-" 2*3DIGIT [ "-" 11HEXDIG ]

sender-fqdn = "sender-fqdn=" OWS 4*( ALPHA / DIGIT / "-" / "." )

intended-n32-purposes = intended-n32-purpose *(";" OWS intended-n32-purpose)
intended-n32-purpose = "intended-n32-purpose=" OWS n32-purpose-value

n32-purpose-value = "ROAMING"
           / "INTER_PLMN_MOBILITY"
           / "SMS_INTERCONNECT"
           / "ROAMING_TEST"
           / "INTER_PLMN_MOBILITY_TEST"
           / "SMS_INTERCONNECT_TEST"
           / "SNPN_INTERCONNECT"
           / "SNPN_INTERCONNECT_TEST"
           / "DISASTER_ROAMING"
           / "DISASTER_ROAMING_TEST"
           / token

req-param = req-param-name "=" OWS req-param-value

req-param-name = token

req-param-value = token
```

EXAMPLE 1: For an HTTP CONNECT request message from the c-SEPP to the RI-A to request establishing the TCP connection towards the p-SEPP:
3gpp-Connect-Req-Info: connect-purpose=n32c; originating-network-id=123-45;sender-fqdn=sepp12.5gc.mnc155.mcc400.3gppnetwork;intended-n32-purpose=ROAMING;intended-32-purpose=SMS_INTERCONNECT

EXAMPLE 2: For an HTTP CONNECT request message from the RI-A to the RI-B to request establishing the TCP connection towards p-SEPP:
3gpp-Connect-Req-Info: connect-purpose=n32c;originating-network-id=123-45;sender-fqdn=ri234.rioperator.com;intended-n32-purpose=ROAMING;intended-n32-purpose=SMS_INTERCONNECT

7.3.2 3gpp-Connect-Resp-Info

The header enables to convey information in the HTTP CONNECT response towards the c-SEPP.

The encoding of the header follows the ABNF as defined in IETF RFC 9110 [9].

```
Connect-Resp-Info-Header = "3gpp-Connect-Resp-Info:" OWS resp-param *( ";" OWS resp-param ) OWS

resp-param = allowed-n32-purposes / p-sepp-fqdn / other-resp-param

allowed-n32-purposes = allowed-n32-purpose *(OWS";"OWS allowed-n32-purpose)

allowed-n32-purpose = "allowed-n32-purpose=" n32-purpose-value

p-sepp-fqdn = "p-sepp-fqdn=" OWS 4*( ALPHA / DIGIT / "-" / "." )

other-resp-param = other-resp-param-name "=" OWS other-resp-param-value

other-resp-param-name = token

other-resp-param-value = token
```

EXAMPLE: 3gpp-Connect-Resp-Info: allowed-n32-purpose=ROAMING;allowed-n32-purpose=SMS_INTERCONNECT;p-sepp-fqdn=sepp2.5gc.mnc203.mcc422.3gppnetwork.org

7.4 Error Handling

7.4.1 General

HTTP/2 connection error and stream error shall be supported as specified in clause 5.4 of IETF RFC 9113 [7].

Upon detecting an error, the Roaming Intermediary shall send an 4XX or 5XX error response. The error response should include a ProblemDetails object providing details on the error and including one of the application errors defined in Table 7.4.2-1.

7.4.2 Application Errors

The application errors defined for the HTTP CONNECT service are listed in Table 7.4.2-1.

Table 7.4.2-1: Application errors

Application Error	HTTP status code	Description
INVALID_MSG_FORMAT	400 Bad Request	It is used when the c-SEPP sends an HTTP request with an invalid format.
CONTRACTUAL_REASON	403 Forbidden	It is used when the SEPP request does not match with the contract between the SEPP and Roaming Intermediary.
INSUFFICIENT_RESOURCES	500 Internal Server Error	It is used when the Roaming Intermediary does not have enough resources to establish the TCP connection toward the destination.
SYSTEM_FAILURE	500 Internal Server Error	The request is rejected due to generic error condition at the Roaming Intermediary.
CONNECTIVITY_ISSUE	502 Bad Gateway	It is used when the Roaming Intermediary cannot establish the connection towards the destination SEPP due to connectivity issue.

Annex A (normative): OpenAPI Specification

A.1 General

This Annex specifies the formal definition of the N32 Handshake API(s) on the N32-c interface. It consists of OpenAPI 3.0.0 specifications, in YAML format.

This Annex takes precedence when being discrepant to other parts of the specification with respect to the encoding of information elements and methods within the API(s).

NOTE: The semantics and procedures, as well as conditions, e.g. for the applicability and allowed combinations of attributes or values, not expressed in the OpenAPI definitions but defined in other parts of the specification also apply.

Informative copies of the OpenAPI specification files contained in this 3GPP Technical Specification are available on a Git-based repository that uses the GitLab software version control system (see 3GPP TS 29.501 [5] clause 5.3.1 and 3GPP TR 21.900 [7] clause 5B).

A.2 N32 Handshake API

```
openapi: 3.0.0

info:
  version: '1.3.0-alpha.6'
  title: 'N32 Handshake API'
  description: |
    N32-c Handshake Service.
    © 2024, 3GPP Organizational Partners (ARIB, ATIS, CCSA, ETSI, TSDSI, TTA, TTC).
    All rights reserved.
servers:
  - url: '{apiRoot}/n32c-handshake/v1'
    variables:
      apiRoot:
        default: https://example.com
        description:  apiRoot as defined in clause 4.4 of 3GPP TS 29.501.
externalDocs:
  description: 3GPP TS 29.573 V18.6.0; 5G System; Public Land Mobile Network (PLMN) Interconnection;
Stage 3
  url: https://www.3gpp.org/ftp/Specs/archive/29_series/29.573/

paths:
  /exchange-capability:
    post:
      summary:  Security Capability Negotiation
      tags:
        - Security Capability Negotiation
      operationId: PostExchangeCapability
      requestBody:
        description: Custom operation for security capability negotiation
        required: true
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/SecNegotiateReqData'
      responses:
        '200':
          description: OK (Successful negitiation of security capabilities)
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/SecNegotiateRspData'
        '307':
          $ref: 'TS29571_CommonData.yaml#/components/responses/307'
        '400':
          $ref: 'TS29571_CommonData.yaml#/components/responses/400'
        '401':
          $ref: 'TS29571_CommonData.yaml#/components/responses/401'
        '403':
          $ref: 'TS29571_CommonData.yaml#/components/responses/403'
        '404':
          $ref: 'TS29571_CommonData.yaml#/components/responses/404'
        '409':
          $ref: 'TS29571_CommonData.yaml#/components/responses/409'
        '411':
          $ref: 'TS29571_CommonData.yaml#/components/responses/411'
        '413':
          $ref: 'TS29571_CommonData.yaml#/components/responses/413'
        '415':
          $ref: 'TS29571_CommonData.yaml#/components/responses/415'
        '429':
          $ref: 'TS29571_CommonData.yaml#/components/responses/429'
        '500':
          $ref: 'TS29571_CommonData.yaml#/components/responses/500'
        '502':
          $ref: 'TS29571_CommonData.yaml#/components/responses/502'
        '503':
          $ref: 'TS29571_CommonData.yaml#/components/responses/503'
        default:
          description: Unexpected error
  /exchange-params:
    post:
      summary: Parameter Exchange
      tags:
        - Parameter Exchange
      operationId: PostExchangeParams
      requestBody:
        description: Custom operation for parameter exchange
        required: true
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/SecParamExchReqData'
      responses:
        '200':
          description: OK (Successful exchange of parameters)
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/SecParamExchRspData'
        '400':
          $ref: 'TS29571_CommonData.yaml#/components/responses/400'
        '401':
          $ref: 'TS29571_CommonData.yaml#/components/responses/401'
        '403':
          $ref: 'TS29571_CommonData.yaml#/components/responses/403'
        '404':
          $ref: 'TS29571_CommonData.yaml#/components/responses/404'
        '409':
          $ref: 'TS29571_CommonData.yaml#/components/responses/409'
        '411':
          $ref: 'TS29571_CommonData.yaml#/components/responses/411'
        '413':
          $ref: 'TS29571_CommonData.yaml#/components/responses/413'
        '415':
          $ref: 'TS29571_CommonData.yaml#/components/responses/415'
        '429':
          $ref: 'TS29571_CommonData.yaml#/components/responses/429'
        '500':
          $ref: 'TS29571_CommonData.yaml#/components/responses/500'
        '502':
          $ref: 'TS29571_CommonData.yaml#/components/responses/502'
        '503':
          $ref: 'TS29571_CommonData.yaml#/components/responses/503'
        default:
          description: Unexpected error
  /n32f-terminate:
    post:
      summary: N32-f Context Terminate
      tags:
        - N32-f Context Terminate
      operationId: PostN32fTerminate
      requestBody:
        description: Custom operation for n32-f context termination
        required: true
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/N32fContextInfo'
      responses:
        '200':
          description: OK (Successful exchange of parameters)
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/N32fContextInfo'
        '400':
          $ref: 'TS29571_CommonData.yaml#/components/responses/400'
        '401':
          $ref: 'TS29571_CommonData.yaml#/components/responses/401'
        '403':
          $ref: 'TS29571_CommonData.yaml#/components/responses/403'
        '404':
          $ref: 'TS29571_CommonData.yaml#/components/responses/404'
        '411':
          $ref: 'TS29571_CommonData.yaml#/components/responses/411'
        '413':
          $ref: 'TS29571_CommonData.yaml#/components/responses/413'
        '415':
          $ref: 'TS29571_CommonData.yaml#/components/responses/415'
        '429':
          $ref: 'TS29571_CommonData.yaml#/components/responses/429'
        '500':
          $ref: 'TS29571_CommonData.yaml#/components/responses/500'
        '502':
          $ref: 'TS29571_CommonData.yaml#/components/responses/502'
        '503':
          $ref: 'TS29571_CommonData.yaml#/components/responses/503'
        default:
          description: Unexpected error
  /n32f-error:
    post:
      summary: N32-f Error Reporting Procedure
      tags:
        - N32-f Error Report
      operationId: PostN32fError
      requestBody:
        description: Custom operation for n32-f error reporting procedure
        required: true
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/N32fErrorInfo'
      responses:
        '204':
          description: successful error reporting
        '400':
          $ref: 'TS29571_CommonData.yaml#/components/responses/400'
        '401':
          $ref: 'TS29571_CommonData.yaml#/components/responses/401'
        '403':
          $ref: 'TS29571_CommonData.yaml#/components/responses/403'
        '404':
          $ref: 'TS29571_CommonData.yaml#/components/responses/404'
        '411':
          $ref: 'TS29571_CommonData.yaml#/components/responses/411'
        '413':
          $ref: 'TS29571_CommonData.yaml#/components/responses/413'
        '415':
          $ref: 'TS29571_CommonData.yaml#/components/responses/415'
        '429':
          $ref: 'TS29571_CommonData.yaml#/components/responses/429'
        '500':
          $ref: 'TS29571_CommonData.yaml#/components/responses/500'
        '502':
          $ref: 'TS29571_CommonData.yaml#/components/responses/502'
        '503':
          $ref: 'TS29571_CommonData.yaml#/components/responses/503'
        default:
          description: Unexpected error
components:
  schemas:
    SecurityCapability:
      description: Enumeration of security capabilities
      anyOf:
        - type: string
          enum:
            - TLS
            - PRINS
            - NONE
        - type: string
    ApiSignature:
      description: API URI of the service operation
      oneOf:
        - $ref: 'TS29571_CommonData.yaml#/components/schemas/Uri'
        - $ref: '#/components/schemas/CallbackName'
    HttpMethod:
      description: Enumeration of HTTP methods
      anyOf:
        - type: string
          enum:
            - GET
            - PUT
            - POST
            - DELETE
            - PATCH
            - HEAD
            - OPTIONS
            - CONNECT
            - TRACE
        - type: string

    IeType:
      description: Enumeration of types of IEs (i.e kind of IE) to specify the protection policy
      anyOf:
        - type: string
          enum:
            - UEID
            - LOCATION
            - KEY_MATERIAL
            - AUTHENTICATION_MATERIAL
            - AUTHORIZATION_TOKEN
            - OTHER
            - NONSENSITIVE
        - type: string

    IeLocation:
      description: Location of the IE in a HTTP message
      anyOf:
        - type: string
          enum:
            - URI_PARAM
            - HEADER
            - BODY
            - MULTIPART_BINARY
            - URI_PATH
        - type: string

    IeInfo:
      description: Protection and modification policy for the IE
      type: object
      required:
        - ieLoc
        - ieType
      properties:
        ieLoc:
          $ref: '#/components/schemas/IeLocation'
        ieType:
          $ref: '#/components/schemas/IeType'
        reqIe:
          type: string
        rspIe:
          type: string
        isModifiable:
          type: boolean
        isModifiableByIpx:
          type: object
          additionalProperties:
            type: boolean
          minProperties: 1
    ApiIeMapping:
      description: API URI to IE mapping on which the protection policy needs to be applied
      type: object
      required:
        - apiSignature
        - apiMethod
        - IeList
      properties:
        apiSignature:
          $ref: '#/components/schemas/ApiSignature'
        apiMethod:
          $ref: '#/components/schemas/HttpMethod'
        IeList:
          type: array
          items:
            $ref: '#/components/schemas/IeInfo'
          minItems: 1
# The attribute name does not follow the naming conventions specified in 3GPP TS 29.501. The
attribute name is kept though as defined in the current specification for backward compatibility
reason.

    ProtectionPolicy:
      description: The protection policy to be negotiated between the SEPPs
      type: object
      required:
        - apiIeMappingList
      properties:
        apiIeMappingList:
          type: array
          items:
            $ref: '#/components/schemas/ApiIeMapping'
          minItems: 1
        dataTypeEncPolicy:
          type: array
          items:
            $ref: '#/components/schemas/IeType'
          minItems: 1

    SecNegotiateReqData:
      description: Defines the security capabilities of a SEPP sent to a receiving SEPP
      type: object
      required:
        - sender
        - supportedSecCapabilityList
      properties:
        sender:
          $ref: 'TS29571_CommonData.yaml#/components/schemas/Fqdn'
        supportedSecCapabilityList:
          type: array
          items:
            $ref: '#/components/schemas/SecurityCapability'
          minItems: 1
        3GppSbiTargetApiRootSupported:
          type: boolean
          default: false
# The attribute name does not follow the naming conventions specified in 3GPP TS 29.501. The
attribute name is kept though as defined in the current specification for backward compatibility
reason.
        plmnIdList:
          type: array
          items:
            $ref: 'TS29571_CommonData.yaml#/components/schemas/PlmnId'
          minItems: 1
        snpnIdList:
          type: array
          items:
            $ref: 'TS29571_CommonData.yaml#/components/schemas/PlmnIdNid'
          minItems: 1
        targetPlmnId:
          $ref: 'TS29571_CommonData.yaml#/components/schemas/PlmnId'
        targetSnpnId:
          $ref: 'TS29571_CommonData.yaml#/components/schemas/PlmnIdNid'
        intendedUsagePurpose:
          type: array
          items:
            $ref: '#/components/schemas/IntendedN32Purpose'
          minItems: 1
        supportedFeatures:
          $ref: 'TS29571_CommonData.yaml#/components/schemas/SupportedFeatures'
        senderN32fFqdn:
          $ref: 'TS29571_CommonData.yaml#/components/schemas/Fqdn'
        senderN32fPort:
          $ref: 'TS29571_CommonData.yaml#/components/schemas/Uinteger'

    SecNegotiateRspData:
      description: Defines the selected security capabilities by a SEPP
      type: object
      required:
        - sender
        - selectedSecCapability
      properties:
        sender:
          $ref: 'TS29571_CommonData.yaml#/components/schemas/Fqdn'
        selectedSecCapability:
          $ref: '#/components/schemas/SecurityCapability'
        3GppSbiTargetApiRootSupported:
          type: boolean
          default: false
# The attribute name does not follow the naming conventions specified in 3GPP TS 29.501. The
attribute name is kept though as defined in the current specification for backward compatibility
reason.
        plmnIdList:
          type: array
          items:
            $ref: 'TS29571_CommonData.yaml#/components/schemas/PlmnId'
          minItems: 1
        snpnIdList:
          type: array
          items:
            $ref: 'TS29571_CommonData.yaml#/components/schemas/PlmnIdNid'
          minItems: 1
        allowedUsagePurpose:
          type: array
          items:
            $ref: '#/components/schemas/IntendedN32Purpose'
          minItems: 1
        rejectedUsagePurpose:
          type: array
          items:
            $ref: '#/components/schemas/IntendedN32Purpose'
          minItems: 1
        supportedFeatures:
          $ref: 'TS29571_CommonData.yaml#/components/schemas/SupportedFeatures'
        senderN32fFqdn:
          $ref: 'TS29571_CommonData.yaml#/components/schemas/Fqdn'
        senderN32fPortList:
          type: array
          items:
            $ref: 'TS29571_CommonData.yaml#/components/schemas/Uinteger'
          minItems: 1

    SecParamExchReqData:
      description: Request data structure for parameter exchange
      type: object
      required:
        - n32fContextId
      properties:
        n32fContextId:
          type: string
          pattern: '^[A-Fa-f0-9]{16}$'
        jweCipherSuiteList:
          type: array
          items:
            type: string
          minItems: 1
        jwsCipherSuiteList:
          type: array
          items:
            type: string
          minItems: 1
        protectionPolicyInfo:
          $ref: '#/components/schemas/ProtectionPolicy'
        secProfiles:
          type: array
          items:
            type: string
          minItems: 1
          maxItems: 256
        ipxProviderSecInfoList:
          type: array
          items:
            $ref: '#/components/schemas/IpxProviderSecInfo'
          minItems: 1
        sender:
          $ref: 'TS29571_CommonData.yaml#/components/schemas/Fqdn'

    SecParamExchRspData:
      description: Response data structure for parameter exchange
      type: object
      required:
        - n32fContextId
      properties:
        n32fContextId:
          type: string
          pattern: '^[A-Fa-f0-9]{16}$'
        selectedJweCipherSuite:
          type: string
        selectedJwsCipherSuite:
          type: string
        selProtectionPolicyInfo:
          $ref: '#/components/schemas/ProtectionPolicy'
        selSecProfiles:
          type: array
          items:
            type: string
          minItems: 1
          maxItems: 256
        ipxProviderSecInfoList:
          type: array
          items:
            $ref: '#/components/schemas/IpxProviderSecInfo'
          minItems: 1
        sender:
          $ref: 'TS29571_CommonData.yaml#/components/schemas/Fqdn'

    N32fContextInfo:
      description: N32-f context information
      type: object
      required:
        - n32fContextId
      properties:
        n32fContextId:
          type: string
          pattern: '^[A-Fa-f0-9]{16}$'
    CallbackName:
      description: Callback Name
      type: object
      required:
        - callbackType
      properties:
        callbackType:
          type: string
    N32fErrorInfo:
      description: N32-f error information
      type: object
      required:
        - n32fMessageId
        - n32fErrorType
      properties:
        n32fMessageId:
          type: string
        n32fErrorType:
          $ref: '#/components/schemas/N32fErrorType'
        n32fContextId:
          type: string
          pattern: '^[A-Fa-f0-9]{16}$'
        failedModificationList:
          type: array
          items:
            $ref: '#/components/schemas/FailedModificationInfo'
          minItems: 1
        errorDetailsList:
          type: array
          items:
            $ref: '#/components/schemas/N32fErrorDetail'
          minItems: 1
        policyMismatchList:
          type: array
          items:
            $ref: 'TS29571_CommonData.yaml#/components/schemas/InvalidParam'
          minItems: 1
    FailedModificationInfo:
      description: Information on N32-f modifications block that failed to process
      type: object
      required:
        - ipxId
        - n32fErrorType
      properties:
        ipxId:
          $ref: 'TS29571_CommonData.yaml#/components/schemas/Fqdn'
        n32fErrorType:
          $ref: '#/components/schemas/N32fErrorType'
    N32fErrorDetail:
      description: Details about the N32f error
      type: object
      required:
        - attribute
        - msgReconstructFailReason
      properties:
        attribute:
          type: string
        msgReconstructFailReason:
          $ref: '#/components/schemas/FailureReason'
    IpxProviderSecInfo:
      description: Defines the security information list of an IPX
      type: object
      required:
        - ipxProviderId
      properties:
        ipxProviderId:
          $ref: 'TS29571_CommonData.yaml#/components/schemas/Fqdn'
        rawPublicKeyList:
          type: array
          items:
            type: string
          minItems: 1
        certificateList:
          type: array
          items:
            type: string
          minItems: 1

    IntendedN32Purpose:
      description: Indicates the intended N32 establishment purpose
      type: object
      required:
        - usagePurpose
      properties:
        usagePurpose:
          $ref: '#/components/schemas/N32Purpose'
        additionalInfo:
          type: string
        cause:
          type: string

    N32fErrorType:
      description: Type of error while processing N32-f message
      anyOf:
        - type: string
          enum:
            - INTEGRITY_CHECK_FAILED
            - INTEGRITY_CHECK_ON_MODIFICATIONS_FAILED
            - MODIFICATIONS_INSTRUCTIONS_FAILED
            - DECIPHERING_FAILED
            - MESSAGE_RECONSTRUCTION_FAILED
            - CONTEXT_NOT_FOUND
            - INTEGRITY_KEY_EXPIRED
            - ENCRYPTION_KEY_EXPIRED
            - POLICY_MISMATCH
        - type: string
    FailureReason:
      description: Reason for failure to reconstruct a HTTP/2 message from N32-f message
      anyOf:
        - type: string
          enum:
            - INVALID_JSON_POINTER
            - INVALID_INDEX_TO_ENCRYPTED_BLOCK
            - INVALID_HTTP_HEADER
        - type: string

    N32Purpose:
      description: Usage purpose of establishing N32 connectivity
      anyOf:
        - type: string
          enum:
            - ROAMING
            - INTER_PLMN_MOBILITY
            - SMS_INTERCONNECT
            - ROAMING_TEST
            - INTER_PLMN_MOBILITY_TEST
            - SMS_INTERCONNECT_TEST
            - SNPN_INTERCONNECT
            - SNPN_INTERCONNECT_TEST
            - DISASTER_ROAMING
            - DISASTER_ROAMING_TEST
        - type: string
```

A.3 JOSE Protected Message Forwarding API on N32-f

```
openapi: 3.0.0

info:
  version: '1.3.0-alpha.3'
  title: 'JOSE Protected Message Forwarding API'
  description: |
    N32-f Message Forwarding Service.
    © 2024, 3GPP Organizational Partners (ARIB, ATIS, CCSA, ETSI, TSDSI, TTA, TTC).
    All rights reserved.
servers:
  - url: '{apiRoot}/n32f-forward/v1'
    variables:
      apiRoot:
        default: https://example.com
        description:  apiRoot as defined in clause 4.4 of 3GPP TS 29.501.
externalDocs:
  description: 3GPP TS 29.573 V18.6.0; 5G System; Public Land Mobile Network (PLMN) Interconnection;
Stage 3
  url: https://www.3gpp.org/ftp/Specs/archive/29_series/29.573/

paths:
  /n32f-process:
    post:
      summary:  N32-f Message Forwarding
      tags:
        - N32-f Forward
      operationId: PostN32fProcess
      parameters:
        - name: Content-Encoding
          in: header
          description: Content-Encoding, described in IETF RFC 9110
          schema:
            type: string
        - name: Accept-Encoding
          in: header
          description: Accept-Encoding, described in IETF RFC 9110
          schema:
            type: string
        - name: 3gpp-Sbi-Message-Priority
          in: header
          description: 3gpp-Sbi-Message-Priority, defined in 3GPP TS 29.500
          schema:
            type: string
      requestBody:
        description: Custom operation N32-f Message Forwarding
        required: true
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/N32fReformattedReqMsg'
      responses:
        '200':
          description: OK (Successful forwarding of reformatted message over N32-f)
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/N32fReformattedRspMsg'
          headers:
            Accept-Encoding:
              description: Accept-Encoding, described in IETF RFC 9110
              schema:
                type: string
            Content-Encoding:
              description: Content-Encoding, described in IETF RFC 9110
              schema:
                type: string
            3gpp-Sbi-Message-Priority:
              description: 3gpp-Sbi-Message-Priority, defined in 3GPP TS 29.500
              schema:
                type: string
        '400':
          $ref: 'TS29571_CommonData.yaml#/components/responses/400'
        '401':
          $ref: 'TS29571_CommonData.yaml#/components/responses/401'
        '403':
          description: Forbidden
          content:
            application/problem+json:
              schema:
                $ref: '#/components/schemas/ProblemDetailsMsgForwarding'
        '404':
          $ref: 'TS29571_CommonData.yaml#/components/responses/404'
        '411':
          $ref: 'TS29571_CommonData.yaml#/components/responses/411'
        '413':
          $ref: 'TS29571_CommonData.yaml#/components/responses/413'
        '415':
          $ref: 'TS29571_CommonData.yaml#/components/responses/415'
        '429':
          $ref: 'TS29571_CommonData.yaml#/components/responses/429'
        '500':
          $ref: 'TS29571_CommonData.yaml#/components/responses/500'
        '502':
          $ref: 'TS29571_CommonData.yaml#/components/responses/502'
        '503':
          $ref: 'TS29571_CommonData.yaml#/components/responses/503'
        default:
          description: Unexpected error

    options:
      summary: Discover communication options supported by next hop (IPX or SEPP)
      operationId: N32fProcessOptions
      tags:
        - N32-f Forward
      responses:
        '204':
          description: No Content
          headers:
            Accept-Encoding:
              description: Accept-Encoding, described in IETF RFC 9110
              schema:
                type: string
        '400':
          $ref: 'TS29571_CommonData.yaml#/components/responses/400'
        '401':
          $ref: 'TS29571_CommonData.yaml#/components/responses/401'
        '403':
          $ref: 'TS29571_CommonData.yaml#/components/responses/403'
        '404':
          $ref: 'TS29571_CommonData.yaml#/components/responses/404'
        '405':
          $ref: 'TS29571_CommonData.yaml#/components/responses/405'
        '429':
          $ref: 'TS29571_CommonData.yaml#/components/responses/429'
        '500':
          $ref: 'TS29571_CommonData.yaml#/components/responses/500'
        '501':
          $ref: 'TS29571_CommonData.yaml#/components/responses/501'
        '502':
          $ref: 'TS29571_CommonData.yaml#/components/responses/502'
        '503':
          $ref: 'TS29571_CommonData.yaml#/components/responses/503'
        default:
          $ref: 'TS29571_CommonData.yaml#/components/responses/default'

components:
  schemas:

#
# STRUCTURED TYPES
#

    FlatJweJson:
      description: Contains the integrity protected reformatted block
      type: object
      required:
        - ciphertext
      properties:
        protected:
          type: string
        unprotected:
          type: object
        header:
          type: object
        encrypted_key:
          type: string
# The attribute name does not follow the naming conventions specified in 3GPP TS 29.501. The
attribute name is kept though as defined in the current specification for backward compatibility
reason.
        aad:
          type: string
        iv:
          type: string
        ciphertext:
          type: string
        tag:
          type: string

    FlatJwsJson:
      description: Contains the modification from IPXes on path
      type: object
      required:
        - payload
        - signature
      properties:
        payload:
          type: string
        protected:
          type: string
        header:
          type: object
        signature:
          type: string

    N32fReformattedReqMsg:
      description: Contains the reformatted HTTP/2 request message
      type: object
      required:
        - reformattedData
      properties:
        reformattedData:
          $ref: '#/components/schemas/FlatJweJson'
        modificationsBlock:
          type: array
          items:
            $ref: '#/components/schemas/FlatJwsJson'
          minItems: 1

    N32fReformattedRspMsg:
      description: Contains the reformatted HTTP/2 response message
      type: object
      required:
        - reformattedData
      properties:
        reformattedData:
          $ref: '#/components/schemas/FlatJweJson'
        modificationsBlock:
          type: array
          items:
            $ref: '#/components/schemas/FlatJwsJson'
          minItems: 1

    DataToIntegrityProtectAndCipherBlock:
      description: HTTP header to be encrypted or the value of a JSON attribute to be encrypted
      type: object
      required:
        - dataToEncrypt
      properties:
        dataToEncrypt:
          type: array
          items: {}
          minItems: 1
    DataToIntegrityProtectBlock:
      description: Data to be integrity protected
      type: object
      properties:
        metaData:
          $ref: '#/components/schemas/MetaData'
        requestLine:
          $ref: '#/components/schemas/RequestLine'
        statusLine:
          type: string
        headers:
          type: array
          items:
            $ref: '#/components/schemas/HttpHeader'
          minItems: 1
        payload:
          type: array
          items:
            $ref: '#/components/schemas/HttpPayload'
          minItems: 1
    RequestLine:
      description: >
        Contains the request line of the HTTP API request being reformatted and forwarded over N32-f
      type: object
      required:
        - method
        - scheme
        - authority
        - path
        - protocolVersion
      properties:
        method:
          $ref: 'TS29573_N32_Handshake.yaml#/components/schemas/HttpMethod'
        scheme:
          $ref: 'TS29571_CommonData.yaml#/components/schemas/UriScheme'
        authority:
          type: string
        path:
          type: string
        protocolVersion:
          type: string
        queryFragment:
          type: string
        pathQueryProtectInd:
          type: array
          items:
            $ref: 'TS29573_N32_Handshake.yaml#/components/schemas/IeLocation'
          minItems: 1
          maxItems: 2
    HttpHeader:
      description: Contains the encoding of HTTP headers in the API request / response
      type: object
      required:
        - header
        - value
      properties:
        header:
          type: string
        value:
          $ref: '#/components/schemas/EncodedHttpHeaderValue'
    HttpPayload:
      description: Contains the encoding of JSON content in the API request / response
      type: object
      required:
        - iePath
        - ieValueLocation
        - value
      properties:
        iePath:
          type: string
        ieValueLocation:
          $ref: 'TS29573_N32_Handshake.yaml#/components/schemas/IeLocation'
        value:
          type: object
    MetaData:
      description: Contains the meta data information needed for replay protection
      type: object
      required:
        - n32fContextId
        - messageId
        - authorizedIpxId
      properties:
        n32fContextId:
          type: string
          pattern: '^[A-Fa-f0-9]{16}$'
        messageId:
          type: string
        authorizedIpxId:
          type: string
    Modifications:
      description: Information on inserting of the modifications entry
      type: object
      required:
        - identity
      properties:
        identity:
          $ref: 'TS29571_CommonData.yaml#/components/schemas/Fqdn'
        operations:
          type: array
          items:
            $ref: 'TS29571_CommonData.yaml#/components/schemas/PatchItem'
          minItems: 1
        tag:
          type: string
    IndexToEncryptedValue:
      description: Index to the encrypted value
      type: object
      required:
        - encBlockIndex
      properties:
        encBlockIndex:
          $ref: 'TS29571_CommonData.yaml#/components/schemas/Uinteger'
    EncodedHttpHeaderValue:
      description: HTTP header value or index to the HTTP header value
      oneOf:
        - type: string
        - $ref: '#/components/schemas/IndexToEncryptedValue'

    ProblemDetailsMsgForwarding:
      allOf:
        - $ref: 'TS29571_CommonData.yaml#/components/schemas/ProblemDetails'
        - $ref: '#/components/schemas/AdditionInfoMsgForwarding'

    AdditionInfoMsgForwarding:
      description: Problem Details extensions for N32-f message forwarding
      properties:
        suggestedStatusCode:
          type: integer
        suggestedProblemDetails:
          $ref: 'TS29571_CommonData.yaml#/components/schemas/ProblemDetails'

#
# SIMPLE TYPES
#

#
# ENUMS
```

```
#
```

A.4 SEPP Telescopic FQDN Mapping API

```
openapi: 3.0.0

info:
  version: '1.2.0-alpha.2'
  title: 'SEPP Telescopic FQDN Mapping API'
  description: |
    SEPP Telescopic FQDN Mapping Service.
    © 2024, 3GPP Organizational Partners (ARIB, ATIS, CCSA, ETSI, TSDSI, TTA, TTC).
    All rights reserved.

servers:
  - url: '{apiRoot}/nsepp-telescopic/v1'
    variables:
      apiRoot:
        default: https://example.com
        description:  apiRoot as defined in clause 4.4 of 3GPP TS 29.501.

externalDocs:
  description: 3GPP TS 29.573 V18.6.0; 5G System; Public Land Mobile Network (PLMN) Interconnection;
Stage 3
  url: https://www.3gpp.org/ftp/Specs/archive/29_series/29.573/

paths:
  /mapping:
    get:
      summary: Maps an FQDN to/from a telescopic FQDN
      operationId: GetTelescopicMapping
      tags:
        - Telescopic Mapping (Document)
      parameters:
        - name: foreign-fqdn
          in: query
          description: FQDN of the NF in the foreign PLMN
          schema:
            $ref: 'TS29571_CommonData.yaml#/components/schemas/Fqdn'
        - name: telescopic-label
          in: query
          description: Telescopic Label
          schema:
            type: string
      responses:
        '200':
          description: Expected response to a valid request
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/TelescopicMapping'
        '400':
          $ref: 'TS29571_CommonData.yaml#/components/responses/400'
        '401':
          $ref: 'TS29571_CommonData.yaml#/components/responses/401'
        '403':
          $ref: 'TS29571_CommonData.yaml#/components/responses/403'
        '404':
          $ref: 'TS29571_CommonData.yaml#/components/responses/404'
        '429':
          $ref: 'TS29571_CommonData.yaml#/components/responses/429'
        '500':
          $ref: 'TS29571_CommonData.yaml#/components/responses/500'
        '502':
          $ref: 'TS29571_CommonData.yaml#/components/responses/502'
        '503':
          $ref: 'TS29571_CommonData.yaml#/components/responses/503'
        default:
          $ref: 'TS29571_CommonData.yaml#/components/responses/default'

components:
  schemas:

#
# STRUCTURED TYPES
#
```

```
    TelescopicMapping:
      description: Contains the Telescopic mapping data
      type: object
      properties:
        telescopicLabel:
          type: string
        seppDomain:
          $ref: 'TS29571_CommonData.yaml#/components/schemas/Fqdn'
        foreignFqdn:
          $ref: 'TS29571_CommonData.yaml#/components/schemas/Fqdn'

#
# SIMPLE TYPES
#

#
# ENUMS
#
```

Annex B (informative): Examples of N32-f Encoding

B.1 General

This Annex provides some example encodings of HTTP/2 request and response messages initiated by NF service consumer / producer when they are reformatted and sent over N32-f

B.2 Input Message Containing No Binary Part

Consider the following example:

- Some headers of the input HTTP/2 message need to be integrity protected and ciphered.
- Some content part of the input HTTP/2 message need to be integrity protected and ciphered.
- The input HTTP/2 message has no multipart/related binary content.
- The headers and content that are not required to be integrity protected and ciphered in the input HTTP/2 message need to be only integrity protected.

The N32fReformattedReqMessage for this example looks like

```
"reformattedData": {
  "protected": BASE64URL(UTF8(JWE Protected Header),
  "unprotected": <non integrity protected shared JOSE headers>,
  "header": <non integrity protected recipient specific JOSE headers>,
  "encrypted_key": BASE64URL(JWE Encrypted Key),
  "aad": BASE64URL(DataToIntegrityProtectBlock),
  "iv": BASE64URL(JWE Initialization Vector),
  "ciphertext": BASE64URL(JWE CipherText(DataToIntegrityProtectAndCipherBlock),
  "tag": BASE64URL(JWE Authentication Tag)
}
```

The DataToIntegrityProtectBlock for this example looks like

```
{
  "metaData":
    {
      "n32fContextId": <the n32fcontext Id of receiving SEPP>,
      "messageId": <Id of the message>,
      "authorizedIpxId": <FQDN of the IPX>
    },
  "requestLine":
    {
      "method": <http method of the NF service API>,
      "scheme": <http scheme of the NF service API>,
      "authority": <authority part of the NF service API URI>,
      "path": <path part of the NF service API URI>,
      "protocolVersion": <HTTP protocol version>,
      "queryFragment": <query fragment of the NF service API, if available>
    },
  "headers":
    [
      {
        "header": <name of HTTP header 1>,
        "value": {"headerval": <string carrying value of the header>}
      },
      {
        "header": <name of HTTP header 2>,
        "value": {"encBlockIndex": 1}
      }
    ],
  "payload":
    [
      {
        "iePath": <JSON Pointer of IE 1>,
        "ieValueLocation": "BODY",
        "value": <value of IE>
      },
      {
        "iePath": <JSON Pointer of IE 2>,
        "ieValueLocation": "BODY",
        "value": {"encBlockIndex": 2}
      }
    ]
}
```

The DataToIntegrityProtectAndCipherBlock for this example looks like

```
{
  "dataToEncrypt":
    [
      <value of HTTP header 2>,
      <value of payload 2>
    ]
}
```

B.3 Input Message Containing Multipart Binary Part

Consider the following example:

- Some headers of the input HTTP/2 message need to be integrity protected and ciphered.
- Some content part of the input HTTP/2 message need to be integrity protected and ciphered.
- The input HTTP/2 message has two multipart/related binary content out of which one binary content needs to be integrity protected and ciphered while the other is only required to be integrity protected.
- The headers and content that are not required to be integrity protected and ciphered in the input HTTP/2 message need to be only integrity protected.

The N32fReformattedReqMessage for this example looks like

```
"reformattedData": {
  "protected": BASE64URL(UTF8(JWE Protected Header),
  "unprotected": <non integrity protected shared JOSE headers>,
  "header": <non integrity protected recipient specific JOSE headers>,
  "encrypted_key": BASE64URL(JWE Encrypted Key),
  "aad": BASE64URL(DataToIntegrityProtectBlock),
  "iv": BASE64URL(JWE Initialization Vector),
  "ciphertext": BASE64URL(JWE CipherText(DataToIntegrityProtectAndCipherBlock),
  "tag": BASE64URL(JWE Authentication Tag)
}
```

The DataToIntegrityProtectBlock for this example looks like

```
{
  "metaData":
    {
      "n32fContextId": <the n32fcontext Id of receiving SEPP>,
      "messageId": <Id of the message>,
      "authorizedIpxId": <FQDN of the IPX>
    },
  "requestLine":
    {
      "method": <http method of the NF service API>,
      "scheme": <http scheme of the NF service API>,
      "authority": <authority part of the NF service API URI>,
      "path": <path part of the NF service API URI>,
      "protocolVersion": <HTTP protocol version>,
      "queryFragment": <query fragment of the NF service API, if available>
    },
  "headers":
    [
      {
        "header": <name of HTTP header 1>,
        "value": {"headerval": <string carrying value of the header>}
      },
      {
        "header": <name of HTTP header 2>,
        "value": {"encBlockIndex": 1}
      }
    ],
  "payload":
    [
      {
        "iePath": <JSON Pointer of IE 1>,
        "ieValueLocation": "BODY",
        "value": <value of IE>
      },
      {
        "iePath": <JSON Pointer of IE 2 - which is an attribute defined with the RefToBinaryData
type>/contentId,
        "ieValueLocation": "BODY",
        "value": <value of the Content ID>
      },
      {
        "iePath": <JSON Pointer of IE 2 - which is an attribute defined with the RefToBinaryData
type>/contenttype,
        "ieValueLocation": "MULTIPART_BINARY",
        "value": <value of the Content Type>
      },
      {
        "iePath": <JSON Pointer of IE 2 - which is an attribute defined with the RefToBinaryData
type>/data,
        "ieValueLocation": "MULTIPART_BINARY",
        "value": <BASE 64 encoded byte array of the binary part>
      }
      {
        "iePath": <JSON Pointer of IE 3 - which is an attribute defined with the RefToBinaryData
type>/contentId,
        "ieValueLocation": "BODY",
        "value": <value of the Content ID>
      },
      {
        "iePath": <JSON Pointer of IE 3 - which is an attribute defined with the RefToBinaryData
type>/contenttype,
        "ieValueLocation": "MULTIPART_BINARY",
        "value": <value of the Content Type>
      },
      {
        "iePath": <JSON Pointer of IE 3 - which is an attribute defined with the RefToBinaryData
type>/data,
        "ieValueLocation": "MULTIPART_BINARY",
        "value": {"encBlockIndex": 2}
      }
    ]
}
```

NOTE: The "iePath" for Content Type or data is a virtual path, which actually refers to the "Content-Type" and "data" in multipart body.

EXAMPLE: If the input HTTP message contains multipart binary part, as:

```
POST /example.com/namf-comm/v1/ue-contexts/{ueContextId}/n1-n2-messages HTTP/2
Content-Type: multipart/related; boundary=----Boundary
Content-Length: xyz

------Boundary
Content-Type: application/json

{
    "n2InfoContainer": {
      "n2InformationClass": "SM",
      "smInfo": {
        "pduSessionId": 5,
        "n2InfoContent": {
          "ngapIeType": "PDU_RES_SETUP_REQ",
          "ngapData": {
            "contentId": "n2msg"
          }
        }
      }
    },
    "pduSessionId": 5
}
------Boundary
Content-Type: application/vnd.3gpp.ngap
Content-Id: n2msg

{ … N2 Information binary data …}
------Boundary
```

the binary content needs to be integrity protected will be formatted, as:

```
"payload":
  [
    {
      "iePath": "/n2InfoContainer/smInfo/n2InfoContent/ngapData/contentId",
      "ieValueLocation": "BODY",
      "value": "n2msg"
    },
    {
      "iePath": "/n2InfoContainer/smInfo/n2InfoContent/ngapData/contenttype",
      "ieValueLocation": "MULTIPART_BINARY",
      "value": "application/vnd.3gpp.ngap"
    },
    {
      "iePath": "/n2InfoContainer/smInfo/n2InfoContent/ngapData/data",
      "ieValueLocation": "MULTIPART_BINARY",
      "value": <BASE 64 encoded byte array of N2 Information binary data >
    }
  ]
```

The DataToIntegrityProtectAndCipherBlock for this example looks like

```
{
  "dataToEncrypt":
    [
      <value of HTTP header 2>,
      <byte array containing BASE 64 encoding of the binary part>
    ]
}
```

B.4 Input Message Containing Sensitive Information in URI Path and/or URI Query Parameters

Consider the following example:

```
POST /nNf-service1/v1/(ueId)/service-operation-1?ue-loc={ueLocation}&query2={value of query
parameter 2}
```

in the above HTTP request,

- One Variable in URI path of the input HTTP/2 message needs to be integrity protected and ciphered, i.e. the UE ID.
- One URI query parameter of the input HTTP/2 message needs to be integrity protected and ciphered, i.e. UE location.
- The headers and content in the input HTTP/2 message need to be only integrity protected.

The N32fReformattedReqMessage for this example looks like

```
"reformattedData": {
  "protected": BASE64URL(UTF8(JWE Protected Header),
  "unprotected": <non integrity protected shared JOSE headers>,
  "header": <non integrity protected recipient specific JOSE headers>,
  "encrypted_key": BASE64URL(JWE Encrypted Key),
  "aad": BASE64URL(DataToIntegrityProtectBlock),
  "iv": BASE64URL(JWE Initialization Vector),
  "ciphertext": BASE64URL(JWE CipherText(DataToIntegrityProtectAndCipherBlock),
  "tag": BASE64URL(JWE Authentication Tag)
}
```

The DataToIntegrityProtectBlock for this example looks like

```
{
  "metaData":
    {
      "n32fContextId": <the n32fcontext Id of receiving SEPP>,
      "messageId": <Id of the message>,
      "authorizedIpxId": <FQDN of the IPX>
    },
  "requestLine":
    {
      "method": <http method of the NF service API>,
      "scheme": <http scheme of the NF service API>,
      "authority": <authority part of the NF service API URI>,
      "path": "/nNf-service1/v1/{"encBlockIndex":1}/service-operation-1",
      "protocolVersion": <HTTP protocol version>,
      "queryFragment": "ue-loc={"encBlockIndex":2}&query2={value of query parameter 2}",
      "pathQueryProtectInd": [ "URI_PATH", "URI_PARAM" ]
    },
  "headers":
    [
      {
        "header": <name of HTTP header 1>,
        "value": {"headerval": <string carrying value of the header>}
      },
      {
        "header": <name of HTTP header 2>,
        "value": {"headerval": <string carrying value of the header>}
      }
    ],
  "payload":
    [
      {
        "iePath": <JSON Pointer of IE 1>,
        "ieValueLocation": "BODY",
        "value": <value of IE>
      },
      {
        "iePath": <JSON Pointer of IE 2>,
        "ieValueLocation": "BODY",
        "value": <value of IE>
      }
    ]
}
```

The DataToIntegrityProtectAndCipherBlock for this example looks like

```
{
  "dataToEncrypt":
    [
      <value of {ueId}>,
      <value of {ueLocation}>
    ]
}
```

Annex C (informative): End to end call flows when SEPP is on path

C.1 General

This Annex provides an informative reference for how the end to end call flow works when the NF service consumer and the NF service producer are in different PLMNs and SEPP is involved on path.

The following clauses explain how the HTTP messages are forwarded between NF services in two PLMNs via the SEPP. In these clauses, the following aspects are not shown to avoid cluttering of the figures and procedure:

- Resolution of FQDN into an IP address using DNS. TCP / TLS connection for sending the HTTP/2 messages is initiated towards the IP address obtained from DNS resolution.

When https URI scheme is used, TLS protection between the Network Function and the SEPP may rely on using telescopic FQDN or 3gpp-Sbi-Target-apiRoot header. See clause 6.1.4.3 of 3GPP TS 29.500 [4].

C.2 TLS security between SEPPs

C.2.1 When http URI scheme is used

C.2.1.1 General

The following figure shows the end to end call flow between an NF service consumer and a NF service producer in different PLMNs when:

- the SEPP in each PLMN acts as a security proxy;
- the negotiated security policy between the SEPPs is TLS;
- "http" scheme URI is used between the NF service consumer and NF service producer; and
- "http" scheme URI is used for accessing NRF's NF discovery service.

NOTE: There may be one or more IPX(s), offering only IP routing serving without content modification or observation of the information, in between the SEPPs.

C.2.1.2 Without TLS protection between NF and SEPP and with TLS security without the 3gpp-Sbi-Target-apiRoot header used over N32f

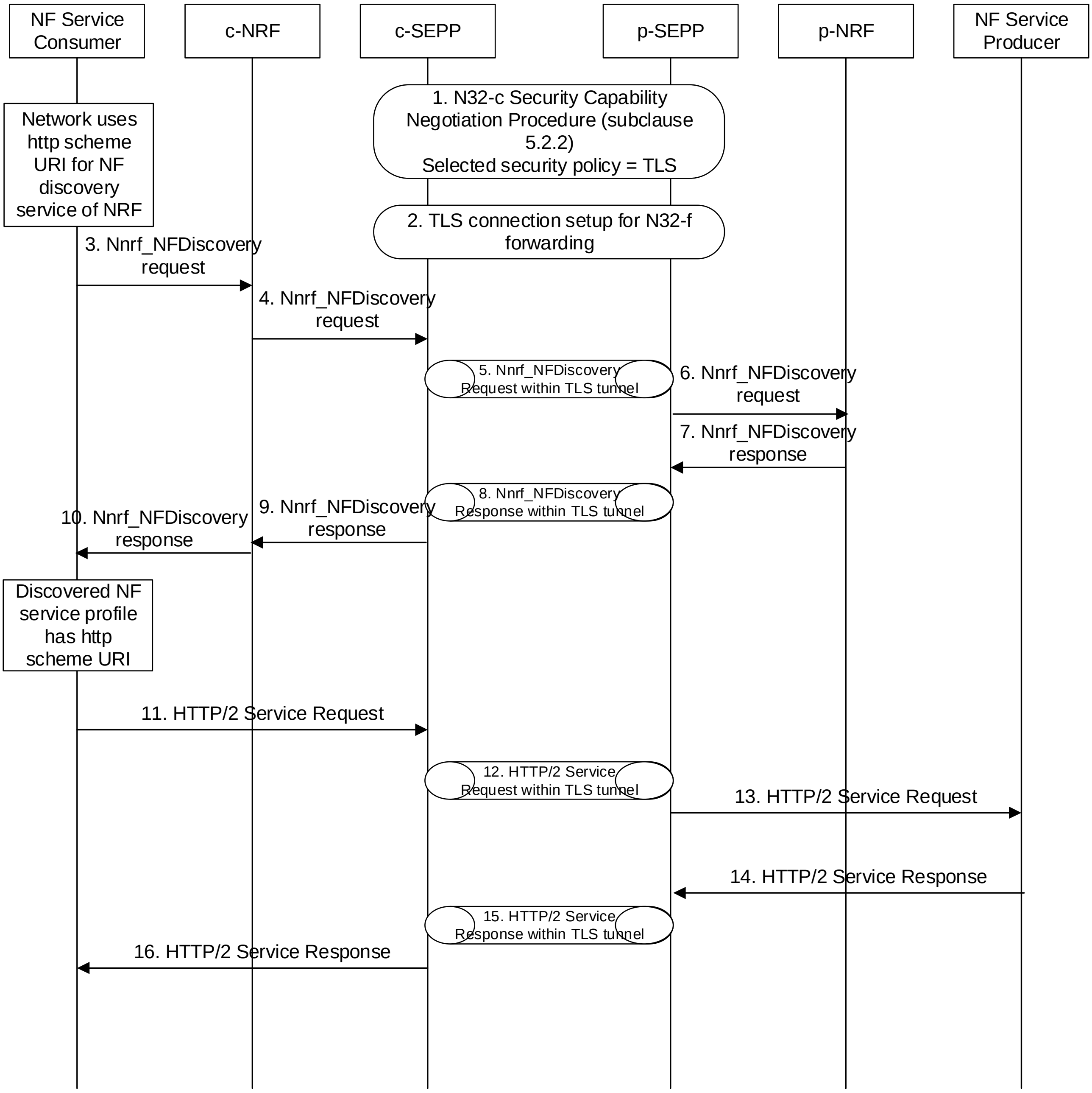
NF Service Consumer
c-NRF
c-SEPP
p-SEPP
p-NRF
NF Service Producer
1. N32-c Security Capability Negotiation Procedure (subclause 5.2.2)
Selected security policy = TLS
Network uses http scheme URI for NF discovery service of NRF
2. TLS connection setup for N32-f forwarding
3. Nnrf_NFDiscovery request
4. Nnrf_NFDiscovery request
5. Nnrf_NFDiscovery Request within TLS tunnel
6. Nnrf_NFDiscovery request
7. Nnrf_NFDiscovery response
8. Nnrf_NFDiscovery Response within TLS tunnel
9. Nnrf_NFDiscovery response
10. Nnrf_NFDiscovery response
Discovered NF service profile has http scheme URI
11. HTTP/2 Service Request
12. HTTP/2 Service Request within TLS tunnel
13. HTTP/2 Service Request
14. HTTP/2 Service Response
15. HTTP/2 Service Response within TLS tunnel
16. HTTP/2 Service Response

Figure C.2.1.2-1: End to end call flow when http scheme URI is used and TLS security without the 3gpp-Sbi-Target-apiRoot header used is used between SEPPs

1. The SEPP on the NF service consumer side (c-SEPP) and the SEPP on the NF service producer side (p-SEPP) negotiate the security capabilities using the procedure specified in clause 5.2.2. The SEPPs mutually negotiate to use TLS as the security policy.

2. A TLS connection is setup between the c-SEPP and the p-SEPP for N32-f forwarding.

3. Before the NF service consumer starts using the API of the NF service producer it needs to discover the NF service profile of the producer by querying the NRF. The NF service consumer uses "http" scheme URI to access the Nnrf_NFDiscovery service.

4. The NRF on the NF service consumer side (c-NRF) needs to further initiate a discovery request to the NRF on the NF service producer side (p-NRF). The c-NRF is configured to route all HTTP messages with inter PLMN FQDN as the "authority" part of the URI via the c-SEPP. The c-SEPP acts as a HTTP proxy.

5. The c-SEPP forwards the NF discovery request within the N32-f TLS tunnel established in step 2.

6. The p-SEPP forwards the NF discovery request to the p-NRF.

7. The p-NRF sends the NF discovery response. The NF service profile contains service URI with "http" scheme. The FQDN of the NF service is an inter PLMN FQDN.

8. The p-SEPP forwards the NF discovery response within TLS tunnel to the c-SEPP.

9. The c-SEPP forwards the NF discovery response to c-NRF.

10. The c-NRF sends the NF discovery response to NF service consumer.

11. The NF service profile received at the NF service consumer contains service URI with "http" scheme. The NF service consumer initiates a HTTP message (as supported by the NF service producer API) using "http" scheme URI. The NF service consumer is configured to route all HTTP messages with inter PLMN FQDN as the "authority" part of the URI via the c-SEPP. The c-SEPP acts as a HTTP proxy.

12. The c-SEPP forwards the HTTP service request within the N32-f TLS tunnel established in step 2.

13. The p-SEPP forwards the HTTP service request to the NF service producer.

14. The NF service producer sends the HTTP service response.

15. The p-SEPP forwards the HTTP service response within TLS tunnel to the c-SEPP.

16. The c-SEPP forwards the HTTP service response to the NF service consumer.

C.2.1.3 Without TLS protection between NF and SEPP and with TLS security with the 3gpp-Sbi-Target-apiRoot header used over N32f

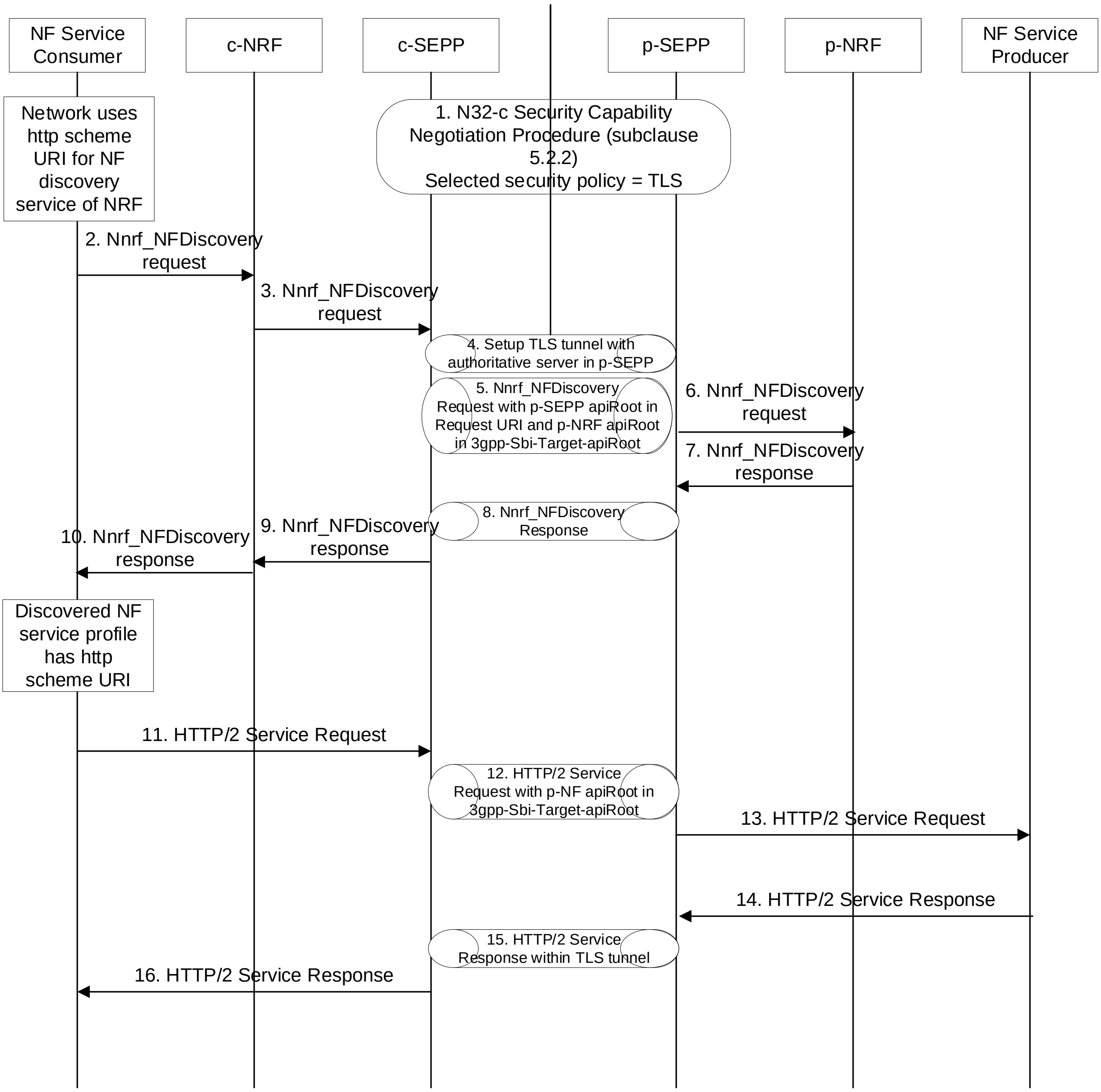
NF Service Consumer
c-NRF
c-SEPP
p-SEPP
p-NRF
NF Service Producer
Network uses http scheme URI for NF discovery service of NRF
1. N32-c Security Capability Negotiation Procedure (subclause 5.2.2) Selected security policy = TLS
2. Nnrf_NFDiscovery request
3. Nnrf_NFDiscovery request
4. Setup TLS tunnel with authoritative server in p-SEPP
5. Nnrf_NFDiscovery Request with p-SEPP apiRoot in Request URI and p-NRF apiRoot in 3gpp-Sbi-Target-apiRoot
6. Nnrf_NFDiscovery request
7. Nnrf_NFDiscovery response
8. Nnrf_NFDiscovery Response
9. Nnrf_NFDiscovery response
10. Nnrf_NFDiscovery response
Discovered NF service profile has http scheme URI
11. HTTP/2 Service Request
12. HTTP/2 Service Request with p-NF apiRoot in 3gpp-Sbi-Target-apiRoot
13. HTTP/2 Service Request
14. HTTP/2 Service Response
15. HTTP/2 Service Response within TLS tunnel
16. HTTP/2 Service Response

Figure C.2.1.3-1: End to end call flow when http scheme URI is used and TLS security with the 3gpp-Sbi-Target-apiRoot header used is used between SEPPs

1. Same as step 1 of Figure C.2.1.2-1.

2. Same as step 3 of Figure C.2.1.2-1

3. Same as step 4 of Figure C.2.1.2-1

4. The c-SEPP setups a TLS connection with the authoritative server for the p-SEPP FQDN (in the apiRoot of the Request URI) and verifies that the certificate presented by the endpoint of the TLS connection belongs to the authoritative server of the p-SEPP. The c-SEPP is configured with the p-SEPP FQDN.

5. The c-SEPP sets the apiRoot in the request URI with the apiRoot of the p-SEPP, inserts the 3gpp-Sbi-Target-apiRoot header set to the apiRoot of the p-NRF, and sends the request towards p-SEPP.

6. The p-SEPP extracts the HTTP message received on the TLS connection, replaces the apiRoot of the p-SEPP FQDN in the request URI with the apiRoot of the p-NRF received in the 3gpp-Sbi-Target-apiRoot header, and then seeing that the URI scheme of the NF discovery service of the p-NRF is "http", the p-SEPP forwards the NF discovery request to the p-NRF.

7 to 11. Same as steps 7 to 11 of Figure C.2.1.2-1.

12. The c-SEPP sets the apiRoot of the p-SEPP FQDN in the request URI, inserts the 3gpp-Sbi-Target-apiRoot header set to the apiRoot of the p-NF, and sends the request towards p-SEPP.

13. The p-SEPP extracts the HTTP message received on the TLS connection, replaces the apiRoot of the p-SEPP FQDN in the request URI with the apiRoot of the p-NF received in the 3gpp-Sbi-Target-apiRoot header and then seeing that the URI scheme of the NF service producer is "http", the p-SEPP forwards the request to the p-NF.

13 to 16. Same as steps 13 to 16 of Figure C.2.1.2-1.

C.2.2 When https URI scheme is used

C.2.2.1 General

The following figures show the end to end call flow between an NF service consumer and a NF service producer in different PLMNs when:

- the SEPP in each PLMN acts as a security proxy;
- the negotiated security policy between the SEPPs is TLS;
- "https" scheme URI is used between the NF service consumer and NF service producer;
- "https" scheme URI is used for accessing NRF's NF discovery service; and
- TLS protection between NF and SEPP relies on using telescopic FQDN or 3gpp-Sbi-Target-apiRoot header.

C.2.2.2 With TLS protection between NF and SEPP relying on telescopic FQDN, and TLS security without the 3gpp-Sbi-Target-apiRoot header used over N32f

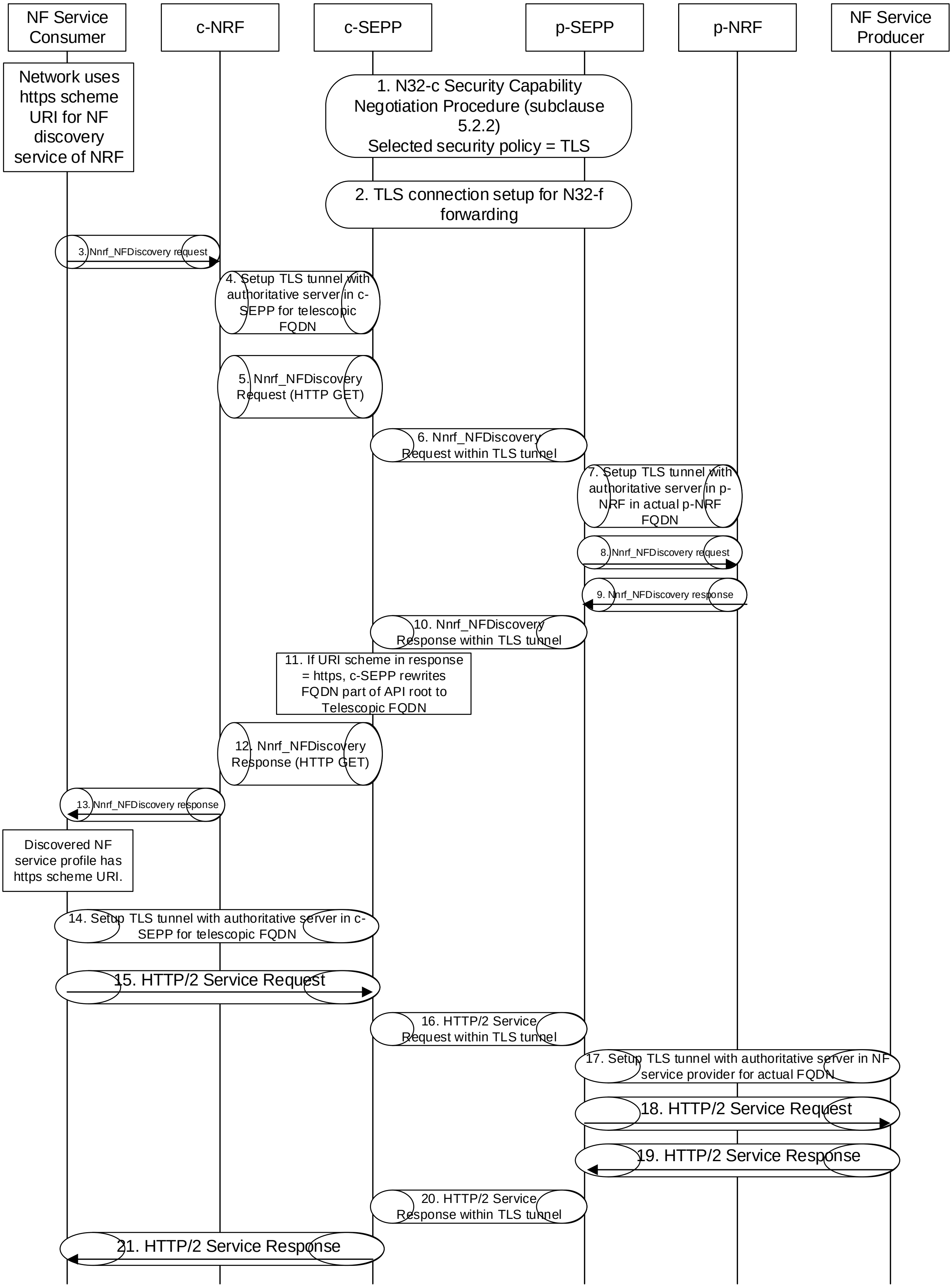
NF Service Consumer
c-NRF
c-SEPP
p-SEPP
p-NRF
NF Service Producer
Network uses https scheme URI for NF discovery service of NRF
1. N32-c Security Capability Negotiation Procedure (subclause 5.2.2) Selected security policy = TLS
2. TLS connection setup for N32-f forwarding
3. Nnrf_NFDiscovery request
4. Setup TLS tunnel with authoritative server in c-SEPP for telescopic FQDN
5. Nnrf_NFDiscovery Request (HTTP GET)
6. Nnrf_NFDiscovery Request within TLS tunnel
7. Setup TLS tunnel with authoritative server in p-NRF in actual p-NRF FQDN
8. Nnrf_NFDiscovery request
9. Nnrf_NFDiscovery response
10. Nnrf_NFDiscovery Response within TLS tunnel
11. If URI scheme in response = https, c-SEPP rewrites FQDN part of API root to Telescopic FQDN
12. Nnrf_NFDiscovery Response (HTTP GET)
13. Nnrf_NFDiscovery response
Discovered NF service profile has https scheme URI.
14. Setup TLS tunnel with authoritative server in c-SEPP for telescopic FQDN
15. HTTP/2 Service Request
16. HTTP/2 Service Request within TLS tunnel
17. Setup TLS tunnel with authoritative server in NF service provider for actual FQDN
18. HTTP/2 Service Request
19. HTTP/2 Service Response
20. HTTP/2 Service Response within TLS tunnel
21. HTTP/2 Service Response

Figure C.2.2.2-1: End to end call flow when https scheme URI is used, telescopic FQDNs are used between NF and SEPP and TLS security without the 3gpp-Sbi-Target-apiRoot header is used between SEPPs

1. The SEPP on the NF service consumer side (c-SEPP) and the SEPP on the NF service producer side (p-SEPP) negotiate the security capabilities using the procedure specified in clause 5.2.2. The SEPPs mutually negotiate to use TLS as the security policy.

2. A TLS connection is setup between the c-SEPP and the p-SEPP for N32-f forwarding.

3. Before the NF service consumer starts using the API of the NF service producer it needs to discover the NF service profile of the producer by querying the NRF. The NF service consumer uses "https" scheme URI to access the Nnrf_NFDiscovery service. This implies that the NF service consumer sets up a TLS connection to the c-NRF and then sends the HTTP request over the TLS connection to the c-NRF.

4. The NRF on the NF service consumer side (c-NRF) needs to further initiate a discovery request to the NRF on the NF service producer side (p-NRF). The c-NRF uses "https" scheme URI to access the NF discovery service of the p-NRF. Since "https" requires setup of TLS connection with the p-NRF and it requires that c-NRF has to verify that the certificate presented by the endpoint of the TLS connection belongs to the authoritative server of the p-NRF, a telescopic FQDN with wildcarded certificate scheme mechanism is specified in 3GPP TS 33.501 [6]. The c-NRF is configured with the telescopic FQDN of the p-NRF with the telescopic FQDN having the FQDN of the c-SEPP as the trailing part. The c-NRF sets up a TLS connection with the authoritative server for the telescopic FQDN (i.e. the c-SEPP).

5. The c-NRF forwards the NF discovery request in this TLS connection.

6. The c-SEPP extracts the NF discovery request from the TLS connection, replaces the telescopic FQDN in the request URI with the FQDN of the p-NRF and sends the request towards p-SEPP in the TLS tunnel setup in step 2. The c-SEPP and the p-SEPP act as a man in the middle proxy in this case.

7. The p-SEPP extracts the HTTP message received on the TLS connection, and then seeing that the URI scheme of the NF discovery service of the p-NRF is in the request URI "https", the p-SEPP sets up a TLS connection with the p-NRF.

8. The p-SEPP forwards the NF discovery request to the p-NRF.

9. The p-NRF sends the NF discovery response within the TLS connection. The NF service profile contains service URI with "https" scheme. The FQDN of the NF service is an inter PLMN FQDN.

10. The p-SEPP forwards the NF discovery response within TLS tunnel setup in step 2 to the c-SEPP. The p-SEPP may replace the inter PLMN FQDN of the NF service producer's API endpoint with a label representing that FQDN. The p-SEPP re-maps the label with the NF service producer's API endpoint in step 17.

11. The c-SEPP upon receiving the HTTP response message for NF discovery response, within the TLS tunnel in step 2, replaces the trailing part of the inter PLMN FQDN of the NF service producer's API endpoint in the NF service profile with the FQDN of the c-SEPP, to form a telescopic FQDN as specified in clause 28.5.2 of 3GPP TS 23.003 [19]. The c-SEPP may replace the label part of the telescopic FQDN with a label of it's own significance. The p-SEPP re-maps the label in step 16.

12. The c-SEPP then forwards the NF discovery response to c-NRF, with the NF service profile containing the telescopic FQDN.

13. The c-NRF sends the NF discovery response to NF service consumer.

14. The NF service profile received at the NF service consumer contains service URI with "https" scheme. The NF service consumer sets up a TLS connection with the authoritative server for the telescopic FQDN (i.e. c-SEPP) received in step 13.

15. The NF service consumer sends the HTTP service request within the TLS connection to the c-SEPP.

16. The c-SEPP extracts the HTTP request from the TLS connection, replaces the telescopic FQDN in the request URI the FQDN of the NF service producer and sends the request towards p-SEPP in the TLS tunnel setup in step 2. The c-SEPP and the p-SEPP act as a man in the middle proxy in this case.

17. The p-SEPP extracts the HTTP message received on the TLS connection, and then seeing that the URI scheme of the NF service producer in the request URI is "https", the p-SEPP sets up a TLS connection with the NF service producer. The p-SEPP also replaces callback URI and link relations within the extracted HTTP message with a telescopic FQDN containing the FQDN of the p-SEPP as the trailing part, as specified in clause 6.1.4.3 of 3GPP TS 29.500 [4].

18. The p-SEPP forwards the HTTP request to the NF service producer.

19. The NF service producer sends the HTTP response within the TLS connection.

20. The p-SEPP forwards the HTTP response within TLS tunnel setup in step 2 to the c-SEPP.

21. The c-SEPP upon receiving the HTTP response message within the TLS tunnel setup in step 2, forwards the response to the NF service consumer. The c-SEPP replaces callback URI and link relations within the extracted HTTP response message with a telescopic FQDN containing the FQDN of the c-SEPP as the trailing part, as specified in clause 6.1.4.3 of 3GPP TS 29.500 [4].

C.2.2.3 With TLS protection between NF and SEPP relying on 3gpp-Sbi-Target-apiRoot header, and TLS security without the 3gpp-Sbi-Target-apiRoot header used over N32f

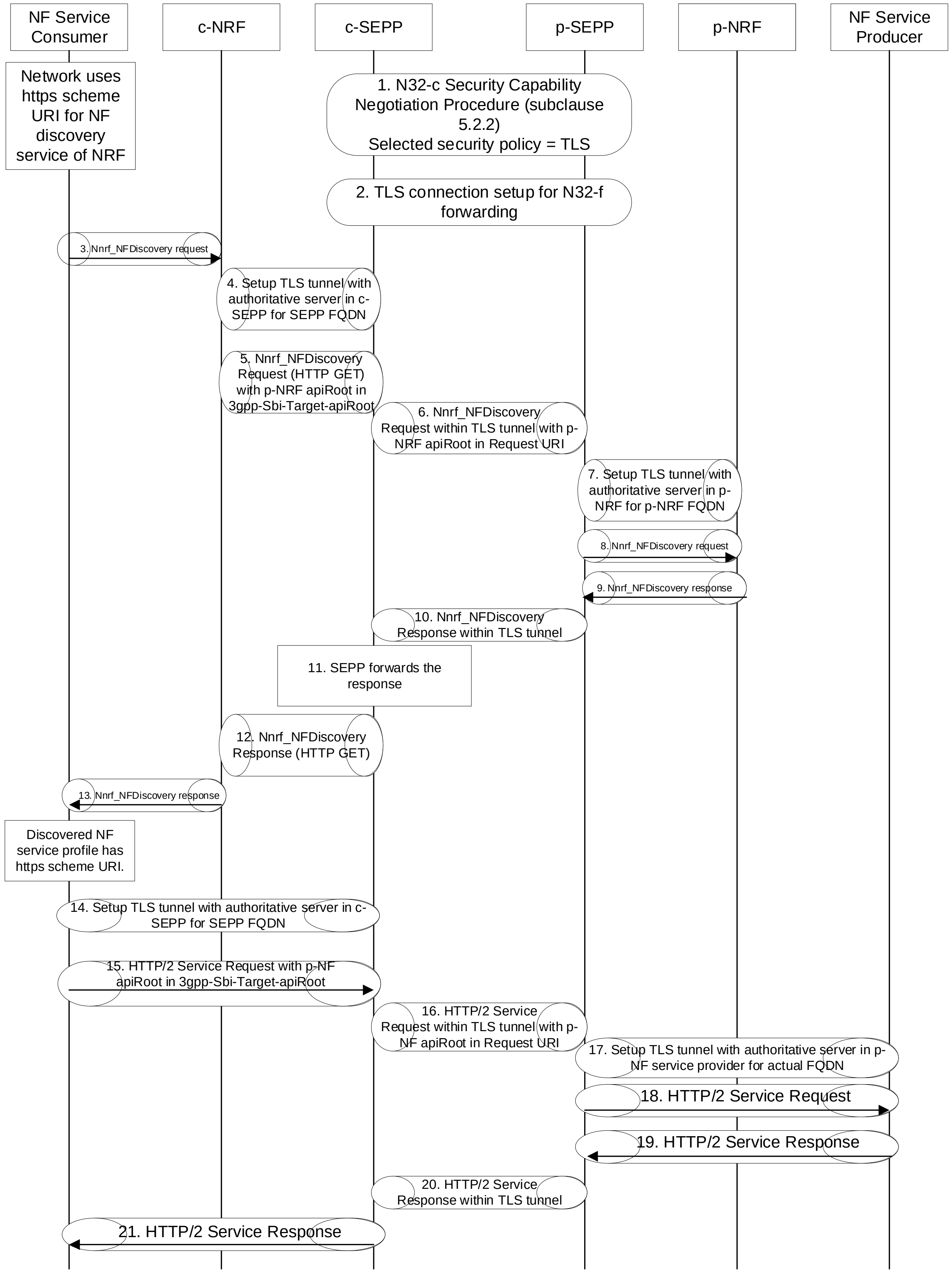
NF Service Consumer
c-NRF
c-SEPP
p-SEPP
p-NRF
NF Service Producer
Network uses https scheme URI for NF discovery service of NRF
1. N32-c Security Capability Negotiation Procedure (subclause 5.2.2) Selected security policy = TLS
2. TLS connection setup for N32-f forwarding
3. Nnrf_NFDiscovery request
4. Setup TLS tunnel with authoritative server in c-SEPP for SEPP FQDN
5. Nnrf_NFDiscovery Request (HTTP GET) with p-NRF apiRoot in 3gpp-Sbi-Target-apiRoot
6. Nnrf_NFDiscovery Request within TLS tunnel with p-NRF apiRoot in Request URI
7. Setup TLS tunnel with authoritative server in p-NRF for p-NRF FQDN
8. Nnrf_NFDiscovery request
9. Nnrf_NFDiscovery response
10. Nnrf_NFDiscovery Response within TLS tunnel
11. SEPP forwards the response
12. Nnrf_NFDiscovery Response (HTTP GET)
13. Nnrf_NFDiscovery response
Discovered NF service profile has https scheme URI.
14. Setup TLS tunnel with authoritative server in c-SEPP for SEPP FQDN
15. HTTP/2 Service Request with p-NF apiRoot in 3gpp-Sbi-Target-apiRoot
16. HTTP/2 Service Request within TLS tunnel with p-NF apiRoot in Request URI
17. Setup TLS tunnel with authoritative server in p-NF service provider for actual FQDN
18. HTTP/2 Service Request
19. HTTP/2 Service Response
20. HTTP/2 Service Response within TLS tunnel
21. HTTP/2 Service Response

Figure C.2.2.3-1 End to end call flow when https scheme URI is used, 3gpp-Sbi-Target-apiRoot header is used between NF and SEPP and TLS security without the 3gpp-Sbi-Target-apiRoot header is used between SEPPs

1. Same as step 1 of Figure C.2.2.2-1.

2. Same as step 2 of Figure C.2.2.2-1.

3. Same as step 3 of Figure C.2.2.2-1

4. The NRF on the NF service consumer side (c-NRF) needs to further initiate a discovery request to the NRF on the NF service producer side (p-NRF). The c-NRF uses "https" scheme URI to access the NF discovery service of the p-NRF. The c-NRF setups a TLS connection with the authoritative server for the SEPP FQDN (in the apiRoot of the Request URI) and verifies that the certificate presented by the endpoint of the TLS connection belongs to the authoritative server of the c-SEPP. The c-NRF is configured with the c-SEPP FQDN, or the c-SEPP registered to the c-NRF (including c-SEPP FQDN in its profile).

5. The c-NRF forwards the NF discovery request in this TLS connection, including an 3gpp-Sbi-Target-apiRoot header set to the apiRoot of the p-NRF.

6. The c-SEPP extracts the NF discovery request from the TLS connection, replaces the apiRoot of the SEPP FQDN in the request URI with the apiRoot of the p-NRF received in the 3gpp-Sbi-Target-apiRoot header and sends the request towards p-SEPP in the TLS tunnel setup in step 2. The c-SEPP and the p-SEPP act as a man in the middle proxy in this case.

7. The p-SEPP extracts the HTTP message received on the TLS connection, and then seeing that the URI scheme of the NF discovery service of the p-NRF is "https", the p-SEPP sets up a TLS connection with the p-NRF.

8. Same as step 8 of Figure C.2.2.2-1

9. Same as step 9 of Figure C.2.2.2-1

10. Same as step 10 of Figure C.2.2.2-1

11, 12. The c-SEPP forwards the NF discovery response to c-NRF.

13. Same as step 13 of Figure C.2.2.2-1

14. The NF service profile received at the NF service consumer contains service URI with "https" scheme. Since the URI of the p-NF contains an authority of a remote PLMN, the NF service consumer sets up a TLS connection with the authoritative server for the SEPP FQDN (i.e. c-SEPP). The c-NF is configured with the c-SEPP FQDN, or the c-NF discovers the c-SEPP FQDN by querying the c-NRF.

15. The NF service consumer sends the HTTP service request within the TLS connection to the c-SEPP, including a 3pp-Sbi-Target-apiRoot header set to the apiRoot of the p-NF.

16. The c-SEPP extracts the HTTP request from the TLS connection, replaces the apiRoot of the SEPP FQDN in the request URI with the apiRoot of the p-NRF received in the 3gpp-Sbi-Target-apiRoot header and sends the request towards p-SEPP in the TLS tunnel setup in step 2. The c-SEPP and the p-SEPP act as a man in the middle proxy in this case.

17. The p-SEPP extracts the HTTP message received on the TLS connection and then seeing that the URI scheme of the NF service producer is "https", the p-SEPP sets up a TLS connection with the NF service producer.

18. Same as step 18 of Figure C.2.2.2-1

19. Same as step 19 of Figure C.2.2.2-1

20. Same as step 20 of Figure C.2.2.2-1

21. The c-SEPP upon receiving the HTTP response message within the TLS tunnel setup in step 2, forwards the response to the NF service consumer.

C.2.2.4 With TLS protection between NF and SEPP relying on telescopic FQDN, and TLS security with the 3gpp-Sbi-Target-apiRoot header used over N32f

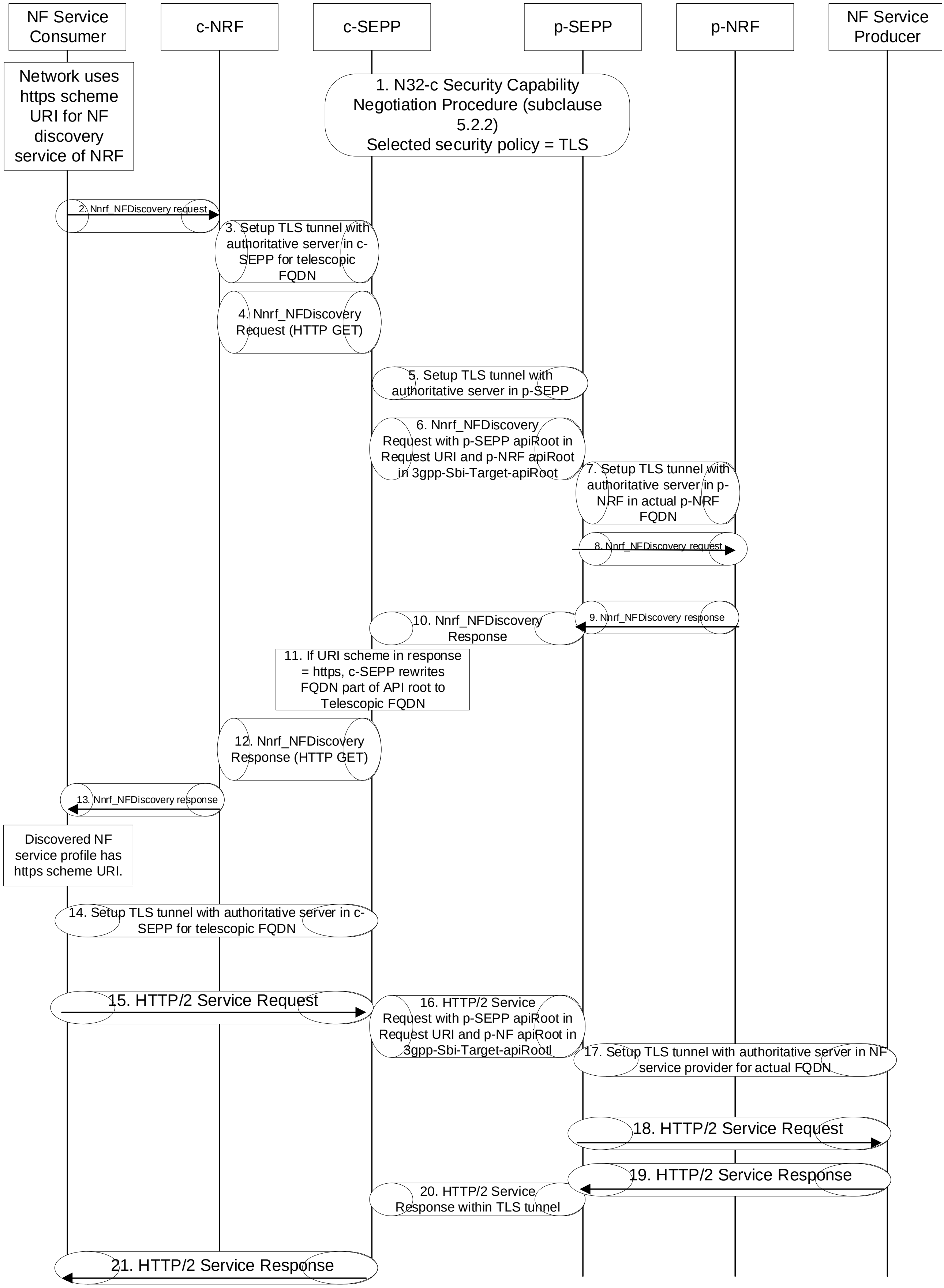
NF Service Consumer
c-NRF
c-SEPP
p-SEPP
p-NRF
NF Service Producer
Network uses https scheme URI for NF discovery service of NRF
1. N32-c Security Capability Negotiation Procedure (subclause 5.2.2) Selected security policy = TLS
2. Nnrf_NFDiscovery request
3. Setup TLS tunnel with authoritative server in c-SEPP for telescopic FQDN
4. Nnrf_NFDiscovery Request (HTTP GET)
5. Setup TLS tunnel with authoritative server in p-SEPP
6. Nnrf_NFDiscovery Request with p-SEPP apiRoot in Request URI and p-NRF apiRoot in 3gpp-Sbi-Target-apiRoot
7. Setup TLS tunnel with authoritative server in p-NRF in actual p-NRF FQDN
8. Nnrf_NFDiscovery request
9. Nnrf_NFDiscovery response
10. Nnrf_NFDiscovery Response
11. If URI scheme in response = https, c-SEPP rewrites FQDN part of API root to Telescopic FQDN
12. Nnrf_NFDiscovery Response (HTTP GET)
13. Nnrf_NFDiscovery response
Discovered NF service profile has https scheme URI.
14. Setup TLS tunnel with authoritative server in c-SEPP for telescopic FQDN
15. HTTP/2 Service Request
16. HTTP/2 Service Request with p-SEPP apiRoot in Request URI and p-NF apiRoot in 3gpp-Sbi-Target-apiRootl
17. Setup TLS tunnel with authoritative server in NF service provider for actual FQDN
18. HTTP/2 Service Request
19. HTTP/2 Service Response
20. HTTP/2 Service Response within TLS tunnel
21. HTTP/2 Service Response

Figure C.2.2.4-1: End to end call flow when https scheme URI is used, telescopic FQDNs are used between NF and SEPP and TLS security with the 3gpp-Sbi-Target-apiRoot header is used between SEPPs

1. Same as step 1 of Figure C.2.2.2-1.

2. Same as step 3 of Figure C.2.2.2-1.

3. Same as step 4 of Figure C.2.2.2-1.

4. Same as step 5 of Figure C.2.2.2-1

5. The c-SEPP setups a TLS connection with the authoritative server for the p-SEPP FQDN (in the apiRoot of the Request URI) and verifies that the certificate presented by the endpoint of the TLS connection belongs to the authoritative server of the p-SEPP. The c-SEPP is configured with the p-SEPP FQDN.

6. The c-SEPP sets the apiRoot in the request URI with the apiRoot of the p-SEPP, inserts the 3gpp-Sbi-Target-apiRoot header set to the apiRoot of the p-NRF derived from the telescopic FQDN received in step 4, and sends the request towards p-SEPP.

7. The p-SEPP extracts the HTTP message received on the TLS connection, replaces the apiRoot of the p-SEPP FQDN in the request URI with the apiRoot of the p-NRF received in the 3gpp-Sbi-Target-apiRoot header, and then seeing that the URI scheme of the NF discovery service of the p-NRF is "https", the p-SEPP sets up a TLS connection with the p-NRF.

8 to 15. Same as steps 8 to 15 of Figure C.2.2.3-1.

16. The c-SEPP extracts the HTTP request from the TLS connection, sets the apiRoot of the p-SEPP FQDN in the request URI, inserts the 3gpp-Sbi-Target-apiRoot header set to the apiRoot of the p-NF derived from the telescopic FQDN received in step 15, and sends the request towards p-SEPP.

17. The p-SEPP extracts the HTTP message received on the TLS connection, replaces the apiRoot of the p-SEPP FQDN in the request URI with the apiRoot of the p-NF received in the 3gpp-Sbi-Target-apiRoot header and then seeing that the URI scheme of the NF service producer is "https", the p-SEPP sets up a TLS connection with the NF service producer.

18 to 21. Same as steps 18 to 21 of Figure C.2.2.2-1

C.2.2.5 With TLS protection between NF and SEPP relying on 3gpp-Sbi-Target-apiRoot header, and TLS security with the 3gpp-Sbi-Target-apiRoot header used over N32f

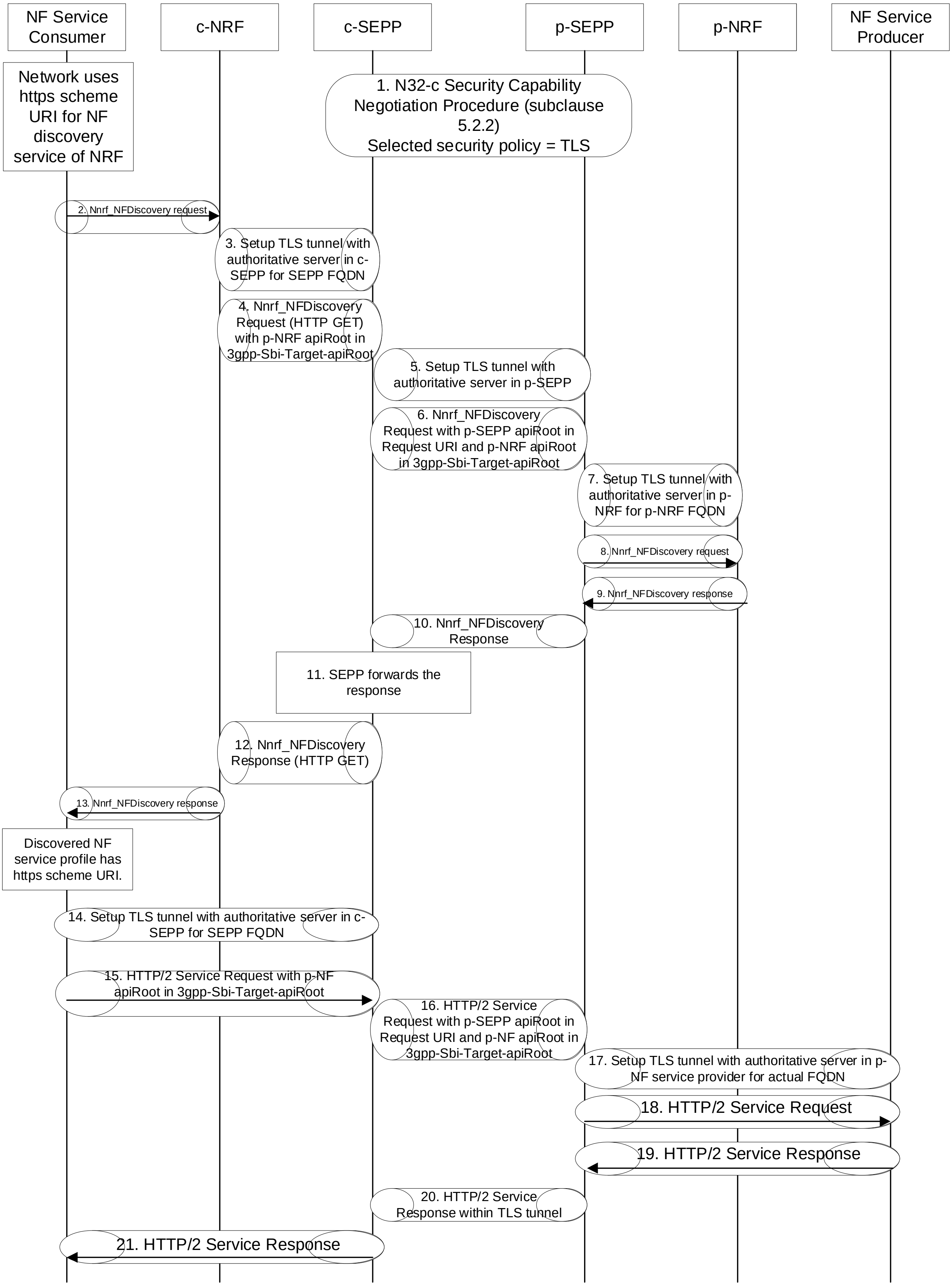
NF Service Consumer
c-NRF
c-SEPP
p-SEPP
p-NRF
NF Service Producer
Network uses https scheme URI for NF discovery service of NRF
1. N32-c Security Capability Negotiation Procedure (subclause 5.2.2)
Selected security policy = TLS
2. Nnrf_NFDiscovery request
3. Setup TLS tunnel with authoritative server in c-SEPP for SEPP FQDN
4. Nnrf_NFDiscovery Request (HTTP GET) with p-NRF apiRoot in 3gpp-Sbi-Target-apiRoot
5. Setup TLS tunnel with authoritative server in p-SEPP
6. Nnrf_NFDiscovery Request with p-SEPP apiRoot in Request URI and p-NRF apiRoot in 3gpp-Sbi-Target-apiRoot
7. Setup TLS tunnel with authoritative server in p-NRF for p-NRF FQDN
8. Nnrf_NFDiscovery request
9. Nnrf_NFDiscovery response
10. Nnrf_NFDiscovery Response
11. SEPP forwards the response
12. Nnrf_NFDiscovery Response (HTTP GET)
13. Nnrf_NFDiscovery response
Discovered NF service profile has https scheme URI.
14. Setup TLS tunnel with authoritative server in c-SEPP for SEPP FQDN
15. HTTP/2 Service Request with p-NF apiRoot in 3gpp-Sbi-Target-apiRoot
16. HTTP/2 Service Request with p-SEPP apiRoot in Request URI and p-NF apiRoot in 3gpp-Sbi-Target-apiRoot
17. Setup TLS tunnel with authoritative server in p-NF service provider for actual FQDN
18. HTTP/2 Service Request
19. HTTP/2 Service Response
20. HTTP/2 Service Response within TLS tunnel
21. HTTP/2 Service Response

Figure C.2.2.5-1: End to end call flow when https scheme URI is used, 3gpp-Sbi-Target-apiRoot header is used between NF and SEPP and TLS security with the 3gpp-Sbi-Target-apiRoot header is used between SEPPs

1. Same as step 1 of Figure C.2.2.3-1.

2. Same as step 3 of Figure C.2.2.3-1

3. Same as step 4 of Figure C.2.2.3-1

4. Same as step 5 of Figure C.2.2.3-1.

5. The c-SEPP setups a TLS connection with the authoritative server for the p-SEPP FQDN (in the apiRoot of the Request URI) and verifies that the certificate presented by the endpoint of the TLS connection belongs to the authoritative server of the p-SEPP. The c-SEPP is configured with the p-SEPP FQDN.

6. The c-SEPP sets the apiRoot in the request URI with the apiRoot of the p-SEPP and sends the request towards p-SEPP including the 3gpp-Sbi-Target-apiRoot header received in step 4.

7. The p-SEPP extracts the HTTP message received on the TLS connection, replaces the apiRoot of the p-SEPP FQDN in the request URI with the apiRoot of the p-NRF received in the 3gpp-Sbi-Target-apiRoot header, and then seeing that the URI scheme of the NF discovery service of the p-NRF is "https", the p-SEPP sets up a TLS connection with the p-NRF.

8 to 15. Same as steps 8 to 15 of Figure C.2.2.3-1.

16. The c-SEPP extracts the HTTP request from the TLS connection, replaces the apiRoot of the c-SEPP FQDN in the request URI with the apiRoot of the p-SEPP, and sends the request towards p-SEPP including the 3gpp-Sbi-Target-apiRoot header received in step 15.

17. The p-SEPP extracts the HTTP message received on the TLS connection, replaces the apiRoot of the p-SEPP FQDN in the request URI with the apiRoot of the p-NF received in the 3gpp-Sbi-Target-apiRoot header and then seeing that the URI scheme of the NF service producer is "https", the p-SEPP sets up a TLS connection with the NF service producer.

18 to 21. Same as steps 18 to 21 of Figure C.2.2.2-1

C.3 Application Layer Security between SEPPs

C.3.1 When http URI scheme is used

The following figure shows the end to end call flow between an NF service consumer and a NF service producer in different PLMNs when:

- the SEPP in each PLMN acts as a security proxy;
- the negotiated security policy between the SEPPs is "PRINS";
- "http" scheme URI is used between the NF service consumer and NF service producer; and
- "http" scheme URI is used for accessing NRF's NF discovery service.

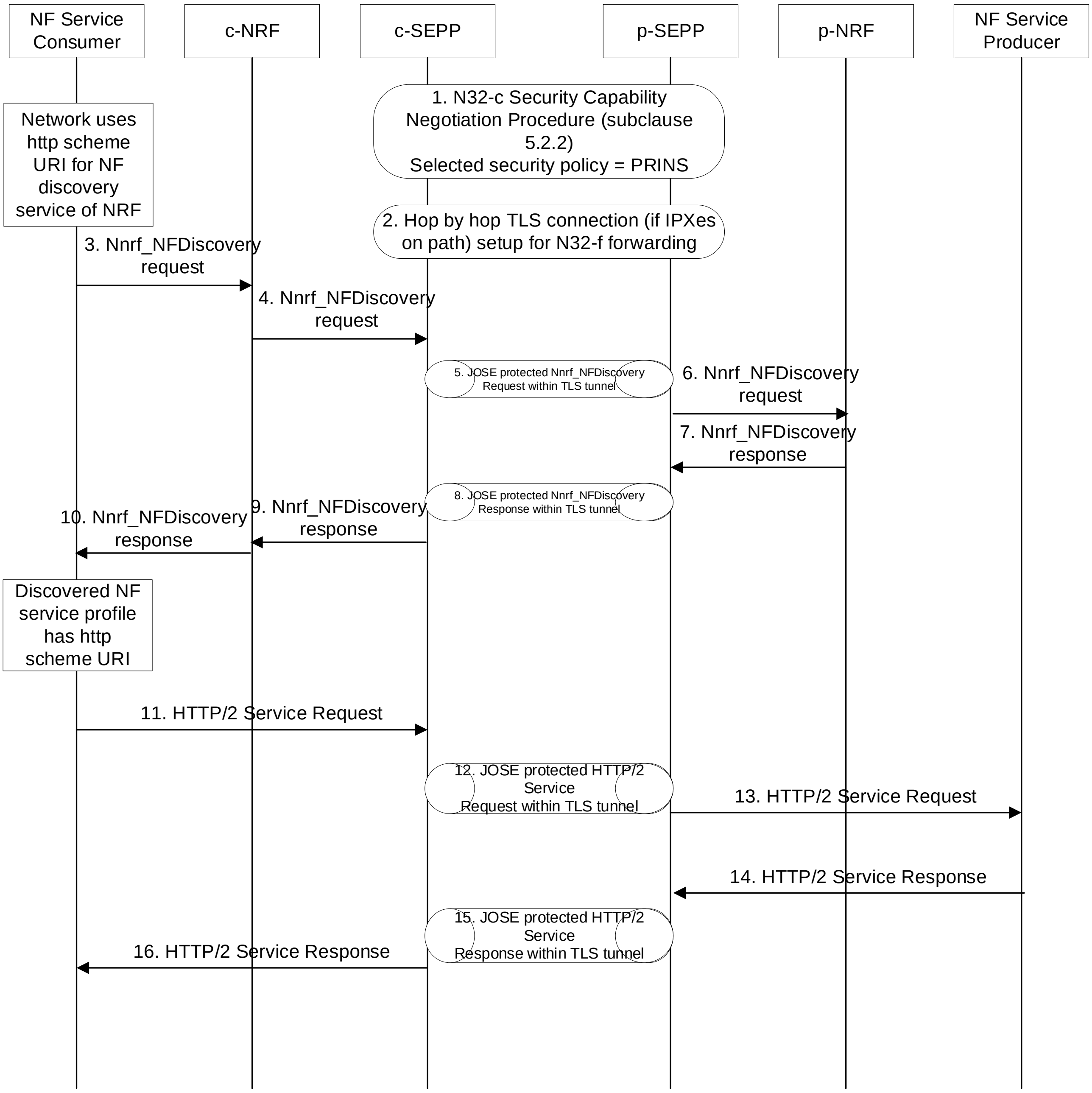
NF Service Consumer
c-NRF
c-SEPP
p-SEPP
p-NRF
NF Service Producer
1. N32-c Security Capability Negotiation Procedure (subclause 5.2.2)
Selected security policy = PRINS
Network uses http scheme URI for NF discovery service of NRF
2. Hop by hop TLS connection (if IPXes on path) setup for N32-f forwarding
3. Nnrf_NFDiscovery request
4. Nnrf_NFDiscovery request
5. JOSE protected Nnrf_NFDiscovery Request within TLS tunnel
6. Nnrf_NFDiscovery request
7. Nnrf_NFDiscovery response
8. JOSE protected Nnrf_NFDiscovery Response within TLS tunnel
9. Nnrf_NFDiscovery response
10. Nnrf_NFDiscovery response
Discovered NF service profile has http scheme URI
11. HTTP/2 Service Request
12. JOSE protected HTTP/2 Service Request within TLS tunnel
13. HTTP/2 Service Request
14. HTTP/2 Service Response
15. JOSE protected HTTP/2 Service Response within TLS tunnel
16. HTTP/2 Service Response

Figure C.3.1-1 End to end call flow when http scheme URI is used and "PRINS" security is used between SEPPs

1. The SEPP on the NF service consumer side (c-SEPP) and the SEPP on the NF service producer side (p-SEPP) negotiate the security capabilities using the procedure specified in clause 5.2.2. The SEPPs mutually negotiate to use "PRINS" as the security policy.

2. A TLS connection is setup between the c-SEPP and the p-SEPP for N32-f forwarding. If IPX-es are deployed between the c-SEPP and p-SEPP, the TLS connection is set up hop by hop with the authoritative server of the next hop.

3. Before the NF service consumer starts using the API of the NF service producer it needs to discover the NF service profile of the producer by querying the NRF. The NF service consumer uses "http" scheme URI to access the Nnrf_NFDiscovery service.

4. The NRF on the NF service consumer side (c-NRF) needs to further initiate a discovery request to the NRF on the NF service producer side (p-NRF). The c-NRF is configured to route all HTTP messages with inter PLMN FQDN as the "authority" part of the URI via the c-SEPP. The c-SEPP acts as a HTTP proxy.

5. The c-SEPP forwards the NF discovery request within the N32-f TLS tunnel established in step 2 and using the JOSE protected message forwarding procedure and API specified in clauses 5.3 and 6.2 respectively. The apiRoot of the Request URI of the HTTP request shall contain the apiRoot of p-SEPP. The HTTP request shall not contain any 3gpp-Sbi-Target-apiRoot header.

6. The p-SEPP forwards the NF discovery request to the p-NRF.

7. The p-NRF sends the NF discovery response. The NF service profile contains service URI with "http" scheme. The FQDN of the NF service is an inter PLMN FQDN.

8. The p-SEPP forwards the NF discovery response within TLS tunnel to the c-SEPP using the JOSE protected message forwarding procedure and API specified in clauses 5.3 and 6.2 respectively.

9. The c-SEPP forwards the NF discovery response to c-NRF.

10. The c-NRF sends the NF discovery response to NF service consumer.

11. The NF service profile received at the NF service consumer contains service URI with "http" scheme. The NF service consumer initiates a HTTP message (as supported by the NF service producer API) using "http" scheme URI. The NF service consumer is configured to route all HTTP messages with inter PLMN FQDN as the "authority" part of the URI via the c-SEPP. The c-SEPP acts as a HTTP proxy.

12. The c-SEPP forwards the HTTP service request within the N32-f TLS tunnel established in step 2 and using the JOSE protected message forwarding procedure and API specified in clauses 5.3 and 6.2 respectively. The apiRoot of the Request URI of the HTTP request shall contain the apiRoot of p-SEPP. The HTTP request shall not contain any 3gpp-Sbi-Target-apiRoot header.

13. The p-SEPP forwards the HTTP service request to the NF service producer.

14. The NF service producer sends the HTTP service response.

15. The p-SEPP forwards the HTTP service response within TLS tunnel to the c-SEPP using the JOSE protected message forwarding procedure and API specified in clauses 5.3 and 6.2 respectively.

16. The c-SEPP forwards the HTTP service response to the NF service consumer.

C.3.2 When https URI scheme is used

C.3.2.1 General

The following figure shows the end to end call flow between an NF service consumer and a NF service producer in different PLMNs when:

- the SEPP in each PLMN acts as a security proxy;
- the negotiated security policy between the SEPPs is "PRINS";
- "https" scheme URI is used between the NF service consumer and NF service producer; and
- "https" scheme URI is used for accessing NRF's NF discovery service; and
- TLS protection between NF and SEPP relies on using telescopic FQDN or 3gpp-Sbi-Target-apiRoot header.

When https URI scheme is used, TLS protection between the Network Function and the SEPP may rely on using telescopic FQDN or 3gpp-Sbi-Target-apiRoot header. See clause 6.1.4.3 of 3GPP TS 29.500 [4].

C.3.2.2 With TLS protection between NF and SEPP relying on telescopic FQDN

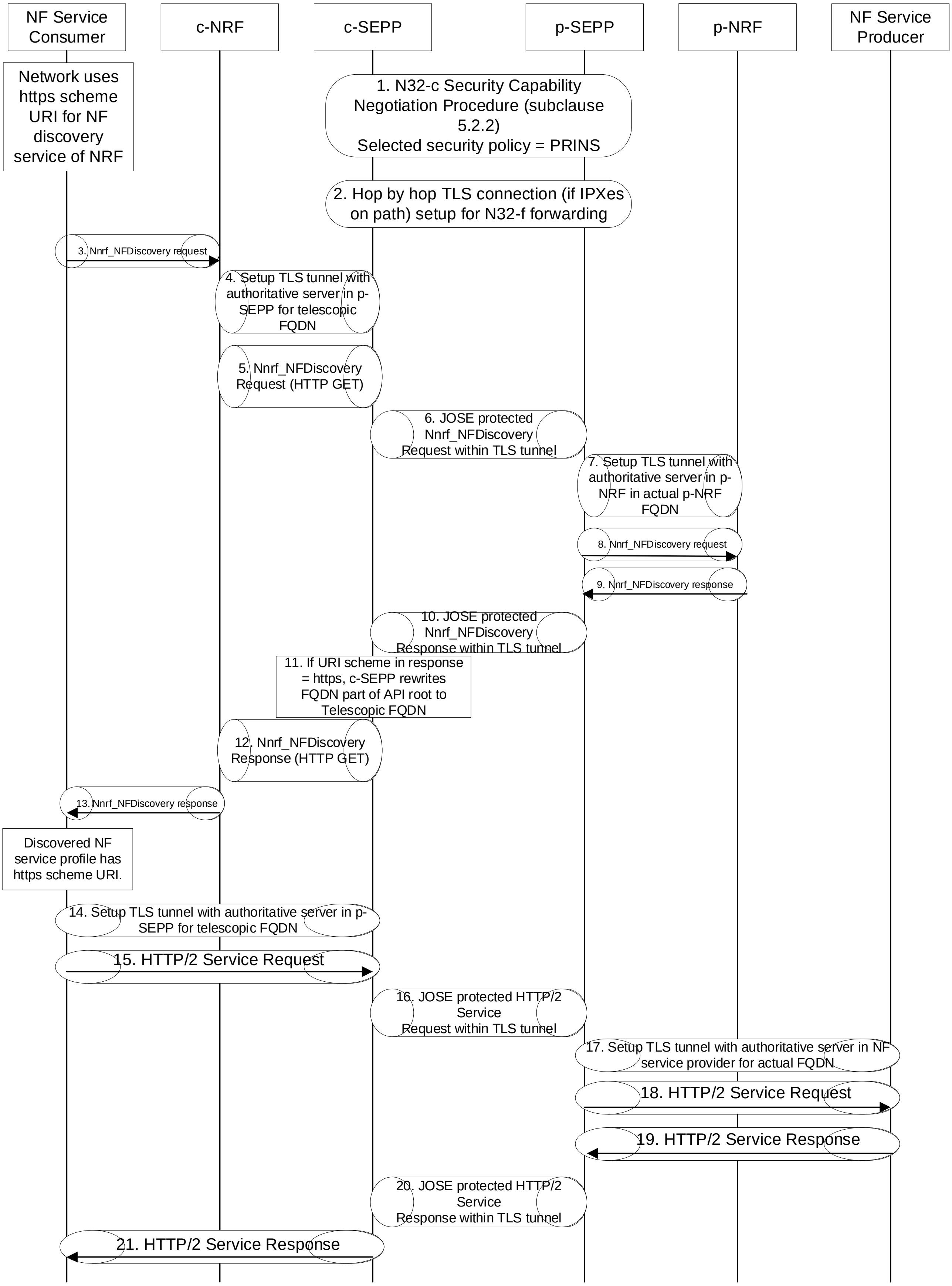
NF Service Consumer
c-NRF
c-SEPP
p-SEPP
p-NRF
NF Service Producer
Network uses https scheme URI for NF discovery service of NRF
1. N32-c Security Capability Negotiation Procedure (subclause 5.2.2)
Selected security policy = PRINS
2. Hop by hop TLS connection (if IPXes on path) setup for N32-f forwarding
3. Nnrf_NFDiscovery request
4. Setup TLS tunnel with authoritative server in p-SEPP for telescopic FQDN
5. Nnrf_NFDiscovery Request (HTTP GET)
6. JOSE protected Nnrf_NFDiscovery Request within TLS tunnel
7. Setup TLS tunnel with authoritative server in p-NRF in actual p-NRF FQDN
8. Nnrf_NFDiscovery request
9. Nnrf_NFDiscovery response
10. JOSE protected Nnrf_NFDiscovery Response within TLS tunnel
11. If URI scheme in response = https, c-SEPP rewrites FQDN part of API root to Telescopic FQDN
12. Nnrf_NFDiscovery Response (HTTP GET)
13. Nnrf_NFDiscovery response
Discovered NF service profile has https scheme URI.
14. Setup TLS tunnel with authoritative server in p-SEPP for telescopic FQDN
15. HTTP/2 Service Request
16. JOSE protected HTTP/2 Service Request within TLS tunnel
17. Setup TLS tunnel with authoritative server in NF service provider for actual FQDN
18. HTTP/2 Service Request
19. HTTP/2 Service Response
20. JOSE protected HTTP/2 Service Response within TLS tunnel
21. HTTP/2 Service Response

Figure C.3.2.2-1 End to end call flow when https scheme URI is used, telescopic FQDNs are used between NF and SEPP and "PRINS" security is used between SEPPs

1. The SEPP on the NF service consumer side (c-SEPP) and the SEPP on the NF service producer side (p-SEPP) negotiate the security capabilities using the procedure specified in clause 5.2.2. The SEPPs mutually negotiate to use "PRINS" as the security policy.

2. A TLS connection is setup between the c-SEPP and the p-SEPP for N32-f forwarding. If IPX-es are deployed between the c-SEPP and p-SEPP, the TLS connection is set up hop by hop with the authoritative server of the next hop.

3. Before the NF service consumer starts using the API of the NF service producer it needs to discover the NF service profile of the producer by querying the NRF. The NF service consumer uses "https" scheme URI to access the Nnrf_NFDiscovery service. This implies that the NF service consumer sets up a TLS connection to the c-NRF and then sends the HTTP request over the TLS connection to the c-NRF.

4. The NRF on the NF service consumer side (c-NRF) needs to further initiate a discovery request to the NRF on the NF service producer side (p-NRF). The c-NRF uses "https" scheme URI to access the NF discovery service of the p-NRF. Since "https" requires setup of TLS connection with the p-NRF and it requires that c-NRF has to verify that the certificate presented by the endpoint of the TLS connection belngs to the authoritative server of the p-NRF, a telescopic FQDN with wildcarded certificate scheme mechanism is specified in 3GPP TS 33.501 [6]. The c-NRF is configured with the telescopic FQDN of the p-NRF with the telescopic FQDN having the FQDN of the c-SEPP as the trailing part. The c-NRF sets up a TLS connection with the authoritative server for the telescopic FQDN (i.e. the c-SEPP).

5. The c-NRF forwards the NF discovery request in this TLS connection.

6. The c-SEPP extracts the NF discovery request from the TLS connection, replaces the telescopic FQDN in the request URI with the FQDN of the p-NRF and sends the request towards p-SEPP in the TLS tunnel setup in step 2 and using the JOSE protected message forwarding procedure and API specified in clauses 5.3 and 6.2 respectively. The apiRoot of the Request URI of the HTTP request shall contain the apiRoot of p-SEPP. The HTTP request shall not contain any 3gpp-Sbi-Target-apiRoot header. The c-SEPP and the p-SEPP act as a man in the middle proxy in this case.

7. The p-SEPP extracts the HTTP message received on the TLS connection, and then seeing that the URI scheme of the NF discovery service of the p-NRF in the request URI is "https", the p-SEPP sets up a TLS connection with the p-NRF.

8. The p-SEPP forwards the NF discovery request to the p-NRF.

9. The p-NRF sends the NF discovery response within the TLS connection. The NF service profile contains service URI with "https" scheme. The FQDN of the NF service is an inter PLMN FQDN.

10. The p-SEPP forwards the NF discovery response within TLS tunnel setup in step 2 using the JOSE protected message forwarding procedure and API specified in clauses 5.3 and 6.2 respectively, to the c-SEPP. The p-SEPP may replace the inter PLMN FQDN of the NF service producer's API endpoint with a label representing that FQDN. The p-SEPP re-maps the label with the NF service producer's API endpoint in step 17.

11. The c-SEPP upon receiving the HTTP response message for NF discovery response, within the TLS tunnel in step 2, replaces the trailing part of the inter PLMN FQDN of the NF service producer's API endpoint in the NF service profile with the FQDN of the c-SEPP, to form a telescopic FQDN as specified in clause 28.5.2 of 3GPP TS 23.003 [19]. The c-SEPP may replace the label part of the telescopic FQDN with a label of it's own significance. The p-SEPP re-maps the label in step 16.

12. The c-SEPP then forwards the NF discovery response to c-NRF, with the NF service profile containing the telescopic FQDN.

13. The c-NRF sends the NF discovery response to NF service consumer.

14. The NF service profile received at the NF service consumer contains service URI with "https" scheme. The NF service consumer sets up a TLS connection with the authoritative server for the telescopic FQDN (i.e. the c-SEPP).

15. The NF service consumer sends the HTTP service request within the TLS connection to the c-SEPP.

16. The c-SEPP extracts the HTTP request from the TLS connection, replaces the the telescopic FQDN in the request URI with the FQDN of the NF service producer and sends the request towards p-SEPP in the TLS tunnel setup in step 2 using the JOSE protected message forwarding procedure and API specified in clauses 5.3 and 6.2 respectively. The apiRoot of the Request URI of the HTTP request shall contain the apiRoot of p-SEPP. The HTTP request shall not contain any 3gpp-Sbi-Target-apiRoot header. The c-SEPP and the p-SEPP act as a man in the middle proxy in this case.

17. The p-SEPP extracts the HTTP message received on the TLS connection, and then seeing that the URI scheme of the NF service producer in the request URI is "https", the p-SEPP sets up a TLS connection with the NF service producer. The p-SEPP also replaces callback URI and link relations within the extracted HTTP message with a telescopic FQDN containing the FQDN of the p-SEPP as the trailing part, as specified in clause 6.1.4.3 of 3GPP TS 29.500 [4].

18. The p-SEPP forwards the HTTP request to the NF service producer.

19. The NF service producer sends the HTTP response within the TLS connection.

20. The p-SEPP forwards the HTTP response within TLS tunnel setup in step 2 to the c-SEPP using the JOSE protected message forwarding procedure and API specified in clauses 5.3 and 6.2 respectively.

21. The c-SEPP upon receiving the HTTP response message within the TLS tunnel setup in step 2, forwards the response to the NF service consumer. The c-SEPP replaces callback URI and link relations within the extracted HTTP response message with a telescopic FQDN containing the FQDN of the c-SEPP as the trailing part, as specified in clause 6.1.4.3 of 3GPP TS 29.500 [4].

C.3.2.3 With TLS protection between NF and SEPP relying on 3gpp-Sbi-Target-apiRoot header

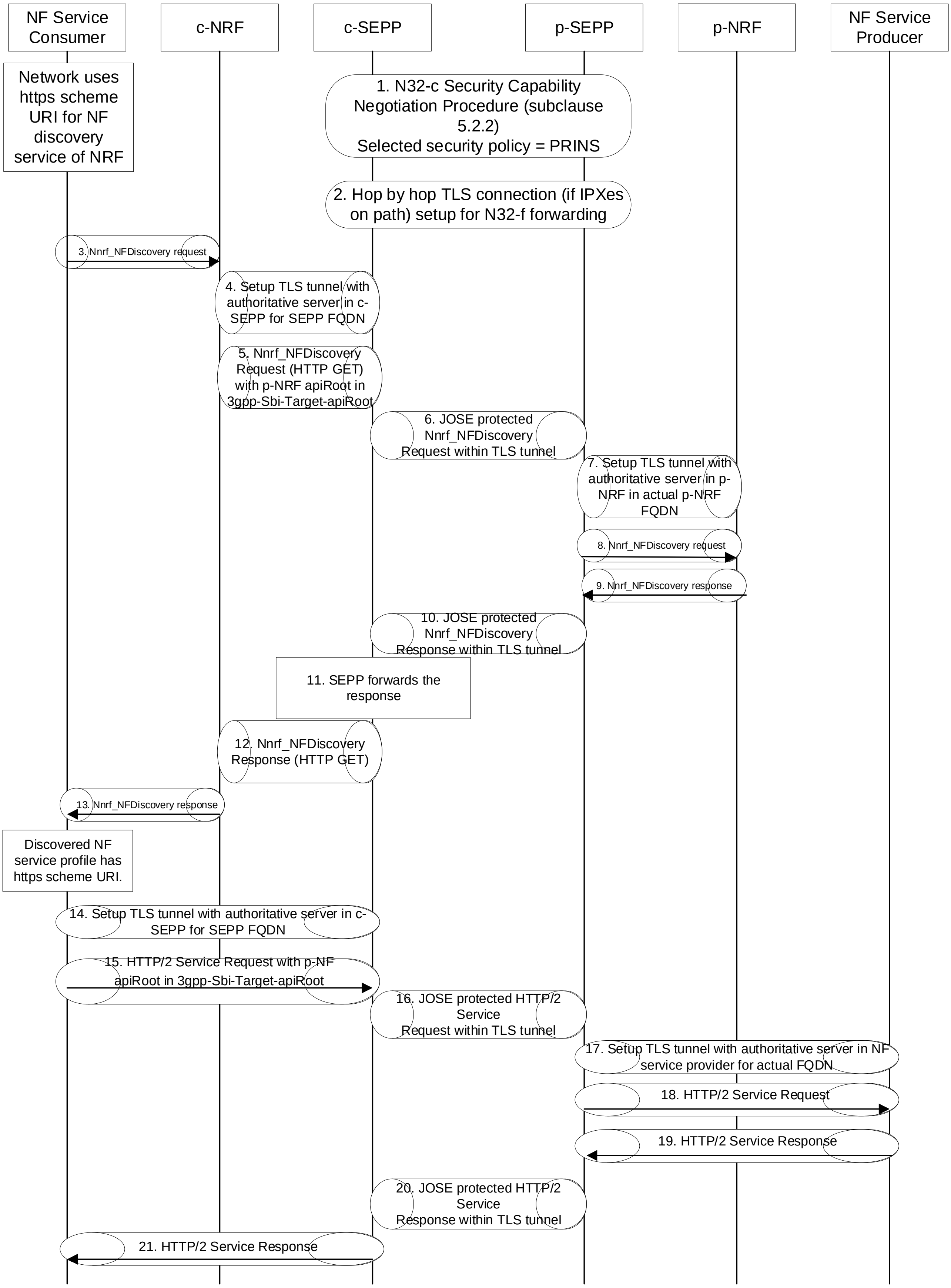
NF Service Consumer
c-NRF
c-SEPP
p-SEPP
p-NRF
NF Service Producer
Network uses https scheme URI for NF discovery service of NRF
1. N32-c Security Capability Negotiation Procedure (subclause 5.2.2)
Selected security policy = PRINS
2. Hop by hop TLS connection (if IPXes on path) setup for N32-f forwarding
3. Nnrf_NFDiscovery request
4. Setup TLS tunnel with authoritative server in c-SEPP for SEPP FQDN
5. Nnrf_NFDiscovery Request (HTTP GET) with p-NRF apiRoot in 3gpp-Sbi-Target-apiRoot
6. JOSE protected Nnrf_NFDiscovery Request within TLS tunnel
7. Setup TLS tunnel with authoritative server in p-NRF in actual p-NRF FQDN
8. Nnrf_NFDiscovery request
9. Nnrf_NFDiscovery response
10. JOSE protected Nnrf_NFDiscovery Response within TLS tunnel
11. SEPP forwards the response
12. Nnrf_NFDiscovery Response (HTTP GET)
13. Nnrf_NFDiscovery response
Discovered NF service profile has https scheme URI.
14. Setup TLS tunnel with authoritative server in c-SEPP for SEPP FQDN
15. HTTP/2 Service Request with p-NF apiRoot in 3gpp-Sbi-Target-apiRoot
16. JOSE protected HTTP/2 Service Request within TLS tunnel
17. Setup TLS tunnel with authoritative server in NF service provider for actual FQDN
18. HTTP/2 Service Request
19. HTTP/2 Service Response
20. JOSE protected HTTP/2 Service Response within TLS tunnel
21. HTTP/2 Service Response

Figure C.3.2.3-1 End to end call flow when https scheme URI is used, 3gpp-Sbi-Target-apiRoot header is used between NF and SEPP and "PRINS" security is used between SEPPs

1. Same as step 1 of Figure C.3.2.2-1.

2. Same as step 2 of Figure C.3.2.2-1.

3. Same as step 3 of Figure C.3.2.2-1.

4. The NRF on the NF service consumer side (c-NRF) needs to further initiate a discovery request to the NRF on the NF service producer side (p-NRF). The c-NRF uses "https" scheme URI to access the NF discovery service of the p-NRF. The c-NRF uses "https" scheme URI to access the NF discovery service of the p-NRF. The c-NRF setups a TLS connection with the authoritative server for the SEPP FQDN (in the apiRoot of the Request URI) and verifies that the certificate presented by the endpoint of the TLS connection belongs to the authoritative server of the c-SEPP. The c-NRF is configured with the c-SEPP FQDN, or the c-SEPP registered to the c-NRF (including c-SEPP FQDN in its profile).

5. The c-NRF forwards the NF discovery request in this TLS connection, including an 3gpp-Sbi-Target-apiRoot header set to the apiRoot of the p-NRF.

6. The c-SEPP extracts the NF discovery request from the TLS connection, replaces the apiRoot of the SEPP FQDN in the request URI with the apiRoot of the p-NRF received in the 3gpp-Sbi-Target-apiRoot header and sends the request towards p-SEPP in the TLS tunnel setup in step 2 and using the JOSE protected message forwarding procedure and API specified in clauses 5.3 and 6.2 respectively. The apiRoot of the Request URI of the HTTP request shall contain the apiRoot of p-SEPP. The HTTP request shall not contain any 3gpp-Sbi-Target-apiRoot header. The c-SEPP and the p-SEPP act as a man in the middle proxy in this case.

7. The p-SEPP extracts the HTTP message received on the TLS connection, and then seeing that the URI scheme of the NF discovery service of the p-NRF is "https", the p-SEPP sets up a TLS connection with the p-NRF.

8. Same as step 8 of Figure C.3.2.2-1.

9. Same as step 9 of Figure C.3.2.2-1.

10. Same as step 10 of Figure C.3.2.2-1.

11, 12. The c-SEPP forwards the NF discovery response to c-NRF.

13. Same as step 13 of Figure C.3.2.2-1.

14. The NF service profile received at the NF service consumer contains service URI with "https" scheme. Since the URI of the p-NF contains an authority of a remote PLMN, the NF service consumer sets up a TLS connection with the authoritative server for the SEPP FQDN (i.e. c-SEPP). The c-NF is configured with the c-SEPP FQDN, or the c-NF discovers the c-SEPP FQDN by querying the c-NRF.

15. The NF service consumer sends the HTTP service request within the TLS connection to the c-SEPP, including a 3pp-Sbi-Target-apiRoot header set to the apiRoot of the p-NF.

16. The c-SEPP extracts the HTTP request from the TLS connection, replaces the apiRoot of the SEPP FQDN in the request URI with the apiRoot of the p-NF received in the 3gpp-Sbi-Target-apiRoot header and sends the request towards p-SEPP in the TLS tunnel setup in step 2 using the JOSE protected message forwarding procedure and API specified in clauses 5.3 and 6.2 respectively. The c-SEPP and the p-SEPP act as a man in the middle proxy in this case. The apiRoot of the Request URI of the HTTP request shall contain the apiRoot of p-SEPP. The HTTP request shall not contain any 3gpp-Sbi-Target-apiRoot header.

17. The p-SEPP extracts the HTTP message received on the TLS connection, and then seeing that the URI scheme of the NF service producer is "https", the p-SEPP sets up a TLS connection with the NF service producer.

18. Same as step 18 of Figure C.3.2.2-1.

19. Same as step 19 of Figure C.3.2.2-1.

20. Same as step 20 of Figure C.3.2.2-1.

21. The c-SEPP upon receiving the HTTP response message within the TLS tunnel setup in step 2, forwards the response to the NF service consumer.

Annex D (informative): Withdrawn API versions

D.1 General

This Annex lists withdrawn API versions of the APIs defined in the present specification. 3GPP TS 29.501 [5] clause 4.3.1.6 describes the withdrawal of API versions.

D.2 N32 Handshake API

The API versions listed in table D.2-1 are withdrawn for the N32 Handshake API.

Table D.2-1: Withdrawn API versions of the N32 Handshake API service

API version number	Reason for withdrawal
1.0.0	A backward incompatible change has been introduced in v1.0.1 to align with related stage 2 specifications. Indeed, the term "ALS" has been replaced by "PRINS" during the handshake procedure. As a consequence, the v1.0.0 must not be used in the field in order to avoid interoperability problem between roaming partners.

Annex E (Normative): ABNF grammar for HTTP custom headers for HTTP CONNECT

E.1 General

This Annex contains a self-contained set of ABNF rules, comprising the re-used rules from IETF RFCs, and the rules defined by the 3GPP custom headers defined in this specification (see clause 7.3).

This grammar may be used as input to existing tools to help implementations to parse 3GPP custom headers.

E.2ABNF definitions (Filename:"TS29573_CustomHeaders.abnf")

```
; ---------------------------------------
;   RFC 5234
; ---------------------------------------

HTAB   = %x09 ; horizontal tab

SP     = %x20

DIGIT  = %x30-39 ; 0-9

ALPHA  = %x41-5A / %x61-7A ; A-Z / a-z

HEXDIG = DIGIT / "A" / "B" / "C" / "D" / "E" / "F"

; ---------------------------------------
;   RFC 9110
; ---------------------------------------

OWS            = *( SP / HTAB )

RWS            = 1*( SP / HTAB )

tchar          = "!" / "#" / "$" / "%" / "&" / "'" / "*" / "+" / "-"
               / "." / "^" / "_" / "`" / "|" / "~" / DIGIT / ALPHA

token          = 1*tchar

; ---------------------------------------
;   3GPP TS 29.573
;
;   Version: 18.6.0 (March 2024)
;
;   (c) 2023, 3GPP Organizational Partners (ARIB, ATIS, CCSA, ETSI, TSDSI, TTA, TTC).
; ---------------------------------------

;
; Header: 3gpp-Connect-Req-Info
;
Connect-Req-Info-Header = "3gpp-Connect-Req-Info:" OWS "connect-purpose=" OWS connect-purpose-value
";" OWS orig-network-id ";" OWS sender-fqdn [";" OWS intended-n32-purposes] *( ";" OWS req-param )

connect-purpose-value =  "n32c" / token

orig-network-id = "originating-network-id=" OWS 3DIGIT "-" 2*3DIGIT [ "-" 11HEXDIG ]

sender-fqdn = "sender-fqdn=" OWS 4*( ALPHA / DIGIT / "-" / "." )

intended-n32-purposes = intended-n32-purpose *(";" OWS intended-n32-purpose)

intended-n32-purpose="intended-n32-purpose=" OWS n32-purpose-value

n32-purpose-value = "ROAMING"
           / "INTER_PLMN_MOBILITY"
           / "SMS_INTERCONNECT"
           / "ROAMING_TEST"
           / "INTER_PLMN_MOBILITY_TEST"
           / "SMS_INTERCONNECT_TEST"
           / "SNPN_INTERCONNECT"
           / "SNPN_INTERCONNECT_TEST"
           / "DISASTER_ROAMING"
           / "DISASTER_ROAMING_TEST"
           / token

req-param = req-param-name "=" OWS req-param-value

req-param-name = token

req-param-value = token

;
; Header: 3gpp-Connect-Resp-Info
;

Connect-Resp-Info-Header = "3gpp-Connect-Resp-Info:" OWS resp-param *( ";" OWS resp-param ) OWS

resp-param = allowed-n32-purposes / p-sepp-fqdn / other-resp-param

allowed-n32-purposes = allowed-n32-purpose *(OWS ";" OWS allowed-n32-purpose)

allowed-n32-purpose = "allowed-n32-purpose=" n32-purpose-value

p-sepp-fqdn = "p-sepp-fqdn=" OWS 4*( ALPHA / DIGIT / "-" / "." )

other-resp-param = other-resp-param-name "=" OWS other-resp-param-value

other-resp-param-name = token

other-resp-param-value = token
```

Annex F (informative): Change history

Change history

Date	Meeting	TDoc	CR	Rev	Cat	Subject/Comment	New version
2018-07	CT4#85bis	C4-185523				TS Skeleton, Scope, General Description and N32 Procedures. Implementation of C4-185531, C4-185353, C4-185352, C4-185469	0.1.0
2018-08	CT4#86	C4-186630				Implementations of PCRs agreed in CT4#86 - C4-186157, C4-186421, C4-186422, C4-186423, C4-186425 and C4-186599	0.2.0
2018-09	CT#81	CP-182082				Presented for information and approval	1.0.0
2018-09	CT#81	CP-182233				Approved in CT#81	15.0.0
2018-12	CT#82	CP-183026	0001	1	F	Resolve the editor's note on HTTP/2 connection management	15.1.0
2018-12	CT#82	CP-183026	0002	1	F	Clarification to N32-f Forwarding Procedure	15.1.0
2018-12	CT#82	CP-183026	0003	2	F	N32-f Error Reporting	15.1.0
2018-12	CT#82	CP-183026	0004	2	F	Resolve editor's notes on identification of notifications	15.1.0
2018-12	CT#82	CP-183026	0005	2	F	Resolve Editor's Notes on RequestId and NextHopId	15.1.0
2018-12	CT#82	CP-183026	0006	2	F	General Cleanup	15.1.0
2018-12	CT#82	CP-183026	0007	1	F	OpenAPI for N32 Handshake API	15.1.0
2018-12	CT#82	CP-183196	0008	2	F	OpenAPI for JOSE Protected Message Forwarding API on N32-f	15.1.0
2018-12	CT#82	CP-183026	0009	1	F	Cardinality	15.1.0
2018-12	CT#82	CP-183026	0010	-	F	Error Handling Clauses	15.1.0
2019-06	CT#84	CP-191043	0011	4	F	PLMN ID verification at receiving SEPP	15.2.0
2019-06	CT#84	CP-191043	0012	1	F	Informative Annex on End to End Call Flow via SEPP	15.2.0
2019-06	CT#84	CP-191043	0013	2	F	Storage of OpenAPI specification files	15.2.0
2019-06	CT#84	CP-191043	0014		F	New name for Application Layer Security protocol	15.2.0
2019-06	CT#84	CP-191043	0015	1	F	Copyright Note in YAML file	15.2.0
2019-06	CT#84	CP-191043	0016		F	3GPP TS 29.573 API version update	15.2.0
2019-09	CT#85	CP-192114	0017		F	ALS renaming to PRINS	15.3.0
2019-09	CT#85	CP-192114	0019	1	F	Add an Annex to Withdrawn N32 Handshake API v1.0.0	15.3.0
2019-09	CT#85	CP-192123	0018	1	B	Telescopic FQDN Mapping Service	16.0.0
2019-09	CT#85	CP-192080	0020	2	B	Exchange IPX security information lists	16.0.0
2019-09	CT#85	CP-192123	0021		F	SecurityNegotiateReqData in the Security Capability Negotiation	16.0.0
2019-09	CT#85	CP-192120	0023		F	3GPP TS 29.573 API Version Update	16.0.0
2019-10						Corrupted references fixed	16.0.1
2019-12	CT#86	CP-193063	0024		F	Certificate and Public key Encoding	16.1.0
2019-12	CT#86	CP-193044	0026		F	3GPP TS 29.573 API version update	16.1.0
2020-03	CT#87	CP-200039	0027	2	F	Add Corresponding API descriptions in clause 5.1	16.2.0
2020-03	CT#87	CP-200016	0028	3	F	Inter-PLMN communication using 3gpp-Sbi-Target-apiRoot	16.2.0
2020-03	CT#87	CP-200047	0030	2	F	Corrections to N32 procedures for PRINS (PRotocol for N32 INterconnect Security)	16.2.0
2020-03	CT#87	CP-200039	0031	2	D	Editorial corrections	16.2.0
2020-03	CT#87	CP-200039	0032	1	F	Correction - formatting consistency	16.2.0
2020-03	CT#87	CP-200140	0033	3	B	29573 CR optionality of ProblemDetails	16.2.0
2020-03	CT#87	CP-200052	0035		F	3GPP TS 29.573 Rel16 API External doc update	16.2.0
2020-07	CT#88	CP-201061	0036		F	Storage of YAML files in ETSI Forge	16.3.0
2020-07	CT#88	CP-201061	0037	1	F	Data type column in Resource URI variables Table	16.3.0
2020-07	CT#88	CP-201061	0038	1	F	Add custom operation Name	16.3.0
2020-07	CT#88	CP-201327	0039	1	F	29.573 Rel-16 API version and External doc update	16.3.0
2020-09	CT#89	CP-202110	0040	1	F	N32f Error Type	16.4.0
2020-09	CT#89	CP-202119	0041	1	F	TLS security with the 3gpp-Sbi-Target-apiRoot header on N32f	16.4.0
2020-09	CT#89	CP-202119	0042		F	Corrections to PRINS call flows	16.4.0
2020-09	CT#89	CP-202023	0044	3	F	Error handling of mismatch of polices at SEPP	16.4.0
2020-09	CT#89	CP-202043	0046	2	F	Correction of flow description	16.4.0
2020-09	CT#89	CP-202096	0047		F	29.573 Rel-16 API version and External doc update	16.4.0
2020-12	CT#90	CP-203048	0049	3	F	PLMN ID handling over N32	16.5.0
2020-12	CT#90	CP-203048	0050	2	F	N32-f payload compression	16.5.0
2020-12	CT#90	CP-203037	0051		F	Update of the metaData	16.5.0
2020-12	CT#90	CP-203037	0052	2	F	Exchange of the modification policy	16.5.0
2020-12	CT#90	CP-203035	0053		F	Storage of YAML files in 3GPP Forge	16.5.0
2020-12	CT#90	CP-203037	0054	2	F	Update the description of IeType	16.5.0
2020-12	CT#90	CP-203036	0055		F	Rel-16 API version and External doc update	16.5.0
2021-03	CT#91-e	CP-210062	0058		F	dataToEncrypt encoding in DataToIntegrityProtectAndCipherBlock	16.6.0
2021-03	CT#91-e	CP-210058	0059	1	F	Error handling for encBlockIndex	16.6.0
2021-03	CT#91-e	CP-210054	0062		F	29.573 Rel-16 API version and External doc update	16.6.0
2021-03	CT#91-e	CP-210034	0060	1	F	OpenAPI Reference	17.0.0
2021-06	CT#92-e	CP-211081	0064	2	A	Annex C.2.2.2 & C.3.2.2 correction of Telescopic FQDN handling in call flow over N32	17.1.0
2021-06	CT#92-e	CP-211028	0065		F	Data Types Descriptions	17.1.0
2021-06	CT#92-e	CP-211028	0066	1	F	Discover the SEPP via NRF	17.1.0
2021-06	CT#92-e	CP-211051	0068	2	F	Clarification to N32 protocol stack	17.1.0
2021-06	CT#92-e	CP-211050	0069		F	29.573 Rel-17 API version and External doc update	17.1.0
2021-09	CT#93-e	CP-212076	0071	3	A	Correction on Parameter Exchange procedure	17.2.0
2021-09	CT#93-e	CP-212076	0073	1	A	Via and server header	17.2.0

2021-09	CT#93-e	CP-212082	0078	1	A	Essential Correction in TLS for N32-f	17.2.0
2021-12	CT#94-e	CP-213085	0083		F	Corrections to the API URI	17.3.0
2022-03	CT#95-e	CP-220025	0089	2	B	Usage indication over N32-c	17.4.0
2022-03	CT#95-e	CP-220032	0086		F	Adding Missing table	17.4.0
2022-03	CT#95-e	CP-220032	0079	2	B	SEPP capability negotation	17.4.0
2022-03	CT#95-e	CP-220080	0088	1	A	SEPP for interconnect scenarios	17.4.0
2022-03	CT#95-e	CP-220080	0075	2	A	PLMN Specific N32-C connection	17.4.0
2022-03	CT#95-e	CP-220066	0090		F	29.573 Rel-17 API version and External doc update	17.4.0
2022-06	CT#96	CP-221027	0092		F	Description of n32fContextId	17.5.0
2022-06	CT#96	CP-221028	0093	1	F	Error response of JOSE Protected Forwarding	17.5.0
2022-06	CT#96	CP-221028	0094	1	F	N32f Error Report	17.5.0
2022-06	CT#96	CP-221027	0095		F	Informative note for attributes not complying with 29.501 naming conventions	17.5.0
2022-06	CT#96	CP-221028	0096	1	F	CONTEXT_NOT_FOUND error report in N32c	17.5.0
2022-06	CT#96	CP-221028	0097	1	F	Reuse of type Fqdn from 29.571	17.5.0
2022-06	CT#96	CP-221036	0098		B	N32 usage with SNPN	17.5.0
2022-06	CT#96	CP-221029	0099		F	Correct the name of AuthenticatedBlock	17.5.0
2022-06	CT#96	CP-221029	0101		F	Informative note for attributes not complying with 29.501 naming conventions	17.5.0
2022-06	CT#96	CP-221069	0103	2	A	Message Containing Multipart Binary	17.5.0
2022-06	CT#96	CP-221069	0107		A	JWE Authentication tag	17.5.0
2022-06	CT#96	CP-221029	0108		B	SMS over N32	17.5.0
2022-06	CT#96	CP-221051	0109		F	Rel-17 API version and External doc update	17.5.0
2022-09	CT#97-e	CP-222051	0116	1	F	Correction on applying modification policies	17.6.0
2022-09	CT#97-e	CP-222071	0115	1	F	Modification policy in IPX	17.6.0
2022-09	CT#97-e	CP-222039	0111		F	Disaster Roaming	17.6.0
2022-09	CT#97-e	CP-222052	0112	3	F	Adding the SBI Message Priority to N32f Message header	17.6.0
2022-09	CT#97-e	CP-222058	0119		F	29.573 Rel-17 API version and External doc update	17.6.0
2022-09	CT#97-e	CP-222021	0113	5	F	FQDN and port number for N32-f connection	18.0.0
2022-09	CT#97-e	CP-222025	0120		F	29.573 Rel-18 API version and External doc update	18.0.0
2022-12	CT#98-e	CP-223028	0121	1	F	Missing mandatory status codes in OpenAPI	18.1.0
2022-12	CT#98-e	CP-223033	0123		F	29.573 Rel-18 API version and External doc update	18.1.0
2023-03	CT#99	CP-230083	0125	1	A	Source SNPN ID verification at the receiving SEPP	18.2.0
2023-03	CT#99	CP-230029	0126	2	B	Redirection support for N32-c	18.2.0
2023-03	CT#99	CP-230029	0127	1	B	Enable senderN32fFqdn and senderN32fPort when PRINS is selected	18.2.0
2023-03	CT#99	CP-230071	0139		F	29.573 Rel-18 API version and External doc update	18.2.0
2023-06	CT#100	CP-231025	0142		F	Remove the Editor Note for N32f FQDN	18.3.0
2023-06	CT#100	CP-231028	0141	1	B	N32-f port list for N32-f connection	18.3.0
2023-06	CT#100	CP-231029	0140	5	F	Location header and missing Redirection clause	18.3.0
2023-06	CT#100	CP-231070	0143		F	29.573 Rel-18 API version and External doc update	18.3.0
2023-09	CT#101	CP-232058	0144	3	F	Major API version	18.4.0
2023-09	CT#101	CP-232058	0145		F	Remove the Editor's Note on RedirectResponse	18.4.0
2023-12	CT#102	CP-233027	0147		F	HTTP RFCs obsoleted by IETF RFC 9110 and 9113	18.5.0
2023-12	CT#102	CP-233056	0149	2	B	General description on Support of Roaming Intermediaries	18.5.0
2023-12	CT#102	CP-233029	0150	2	F	Security capability negotiation procedures collision	18.5.0
2023-12	CT#102	CP-233028	0151		F	N32-f interface with TLS security	18.5.0
2023-12	CT#102	CP-233029/ CP-233031	0152	5	F	N32-c context recovery	18.5.0
2023-12	CT#102	CP-233057	0153	3	B	N32-f related error determined upon receipt of an N32-f request	18.5.0
2023-12	CT#102	CP-233057	0154	2	B	N32-f related error determined upon receipt of an N32-f response	18.5.0
2023-12	CT#102	CP-233057	0155	2	B	Applicative (i.e. SBI related) error upon receipt of an N32-f request	18.5.0
2023-12	CT#102	CP-233057	0156	2	B	Procedures for Roaming Intermediary to reject N32 connection establishment	18.5.0
2023-12	CT#102	CP-233031	0157	1	F	ProblemDetails RFC 7807 obsoleted by 9457	18.5.0
2023-12	CT#102	CP-233027	0158	1	F	N32-f context not found for TLS connection	18.5.0
2023-12	CT#102	CP-233028	0159	1	F	N32-f Context Termination Procedure collision	18.5.0
2023-12	CT#102	CP-233029	0160	1	F	Parameter exchange procedure collision	18.5.0
2023-12	CT#102	CP-233030	0161	2	F	Update N32-f for TLS related sub-clauses	18.5.0
2023-12	CT#102	CP-233060	0162		F	29.573 Rel-18 API version and External doc update	18.5.0
2024-03	CT#103	CP-240053	0167		F	Correct the reference number of IETF RFC 9110	18.6.0
2024-03	CT#103	CP-240053	0178		F	Editorial and Style Corrections	18.6.0
2024-03	CT#103	CP-240053	0174	2	B	Support the negotiation of security profiles	18.6.0
2024-03	CT#103	CP-240053	0175	2	F	Clarification to the description of the queryfragment attribute	18.6.0
2024-03	CT#103	CP-240053	0180	2	F	Protection of Sensitive Information in Request Line	18.6.0
2024-03	CT#103	CP-240055	0163		B	Addressing the limitations per TS 33.501	18.6.0
2024-03	CT#103	CP-240055	0169		F	Editor Note's on handling of error delivery failure over N32-f	18.6.0
2024-03	CT#103	CP-240055	0173		B	Routing the N32-c Error Reporting Request from the Roaming Intermediary	18.6.0

2024-03	CT#103	CP-240240	0170	2	B	N32-c connection establishment to support roaming intermediaries for 5GS roaming	18.6.0
2024-03	CT#103	CP-240055	0171	1	B	N32-c connection establishment rejection by Roaming Intermediaries	18.6.0
2024-03	CT#103	CP-240055	0172	1	B	N32-f message format for RI originated messages and related data types	18.6.0
2024-03	CT#103	CP-240055	0179	1	B	Error Handling for Roaming Intermediary Generated N32-f Related Error Request	18.6.0
2024-03	CT#103	CP-240055	0165	2	B	Applicative request message originated by roaming intermediary over N32-f	18.6.0
2024-03	CT#103	CP-240056	0181		F	29.573 Rel-18 API version and External doc update	18.6.0

History

Document history		
V18.6.0	May 2024	Publication