ETSI TS 129 573 V15.3.0 (2019-10)

5G;
5G System;
Public Land Mobile Network (PLMN) Interconnection;
Stage 3
(3GPP TS 29.573 version 15.3.0 Release 15)

Reference
RTS/TSGC-0429573vf30

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 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is 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 IPR Policy, no investigation, 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.

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 http://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.

1 Scope

The present document specifies the stage 3 protocol and data model for the PLMN 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 7540: "Hypertext Transfer Protocol Version 2 (HTTP/2)".

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

[9] IETF RFC 7231: "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content".

[10] IETF RFC 7230: "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing".

[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".

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.

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].

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

4 General Description

4.1 Introduction

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

4.2 N32 Interface

4.2.1 General

The N32 interface is used between the SEPPs of a VPLMN and a HPLMN in roaming scenarios. 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 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.

4.2.2 N32-c Interface

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

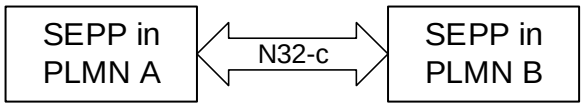

Figure 4.2.2-1: N32-c Interface

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 figure shows the scope of the N32-f interface.

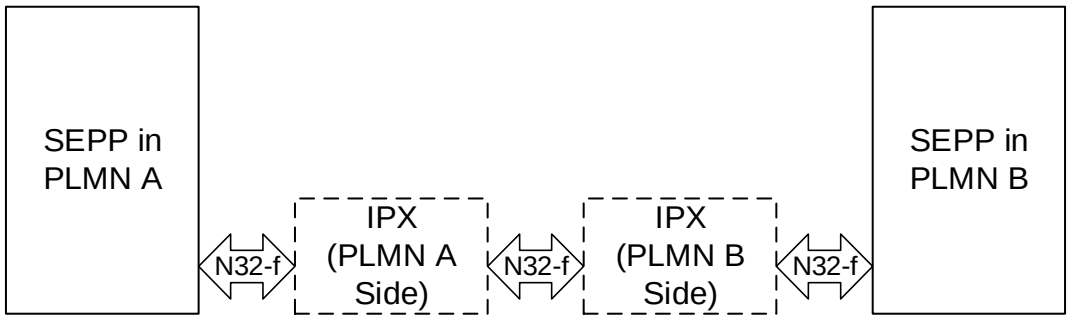

Figure 4.2.3-1: N32-f Interface

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. 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.

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.

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.

4.3 Protocol Stack

4.3.1 General

The protocol stack for the N32 interface is shown below in Figure 4.2.1-1

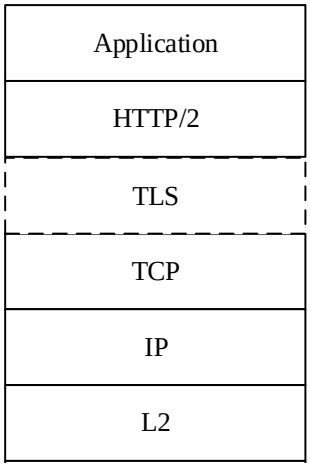

Figure 4.3.1-1: N32 Protocol Stack

The N32 interfaces (N32-c and N32-f) use HTTP/2 protocol (see clause 4.2.2) 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 3GPP TS 33.501 [6].

For the N32-f interface, the application layer (i.e the JSON payload) 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.

4.3.2 HTTP/2 Protocol

4.3.2.1 General

HTTP/2 as described in IETF RFC 7540 [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 7231 [9]	This header is used to specify response media types that are acceptable.
Accept-Encoding	IETF RFC 7231 [9]	This header may be used to indicate what response content-encodings (e.g gzip) are acceptable in the response.
Content-Length	IETF RFC 7230 [10]	This header is used to provide the anticipated size, as a decimal number of octets, for a potential payload body.
Content-Type	IETF RFC 7231 [9]	This header is used to indicate the media type of the associated representation.

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

Name	Reference	Description
Content-Length	IETF RFC 7230 [10]	This header may be used to provide the anticipated size, as a decimal number of octets, for a potential payload body.
Content-Type	IETF RFC 7231 [9]	This header shall be used to indicate the media type of the associated representation.
Content-Encoding	IETF RFC 7231 [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.

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
- Case B: Towards a SEPP of another PLMN via IPX. 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 IPSec or TLS VPN;
- 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 VPN.

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 7540 [7]). When there is no IPX between the SEPPs, TLS shall be used for security protection (see 3GPP TS 33.501 [6]). When there is IPX between the SEPPs, TLS should be used for security protection as specified in 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.

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. The procedure is described in Figure 5.2.2-1 below.

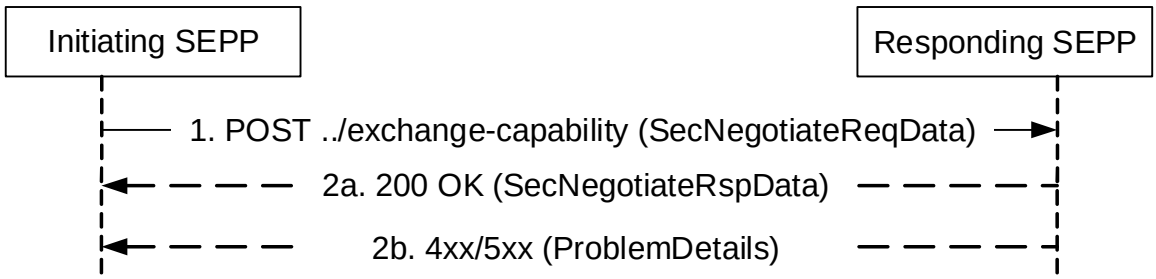

Figure 5.2.2-1: Security Capability Negotiation Procedure

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

- Supported security capabilities (i.e PRINS and/or TLS)

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 security capability (i.e PRINS or TLS)

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).

2b. On failure, the responding SEPP shall respond to the initiating SEPP with an appropriate 4xx/5xx status code as specified in clause 6.1.4.2.

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.

The procedure is described in Figure 5.2.3.2-1 below.

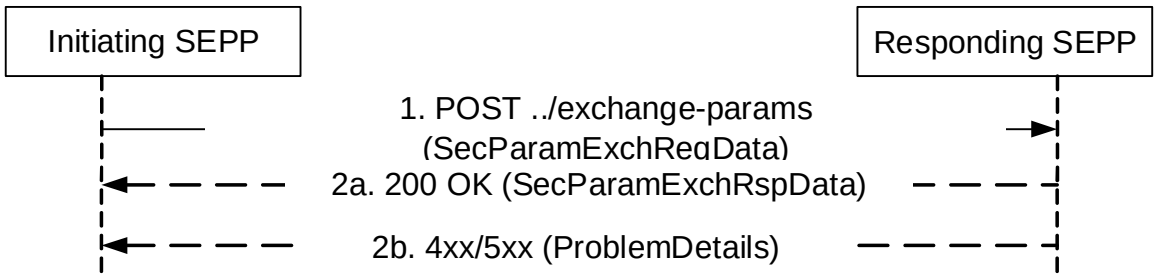

Figure 5.2.3.2-1: Parameter Exchange Procedure for Cipher Suite Negotiation

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.

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.

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 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.

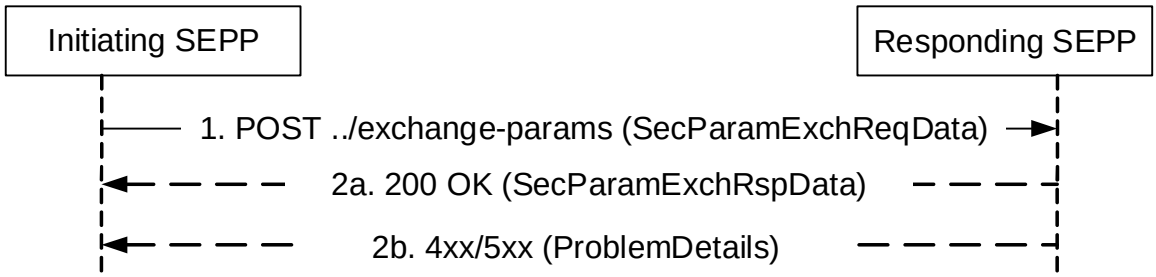

Figure 5.2.3.3-1: Parameter Exchange Procedure for Protection Policy Negotiation

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

- 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
- 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.

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

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.

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

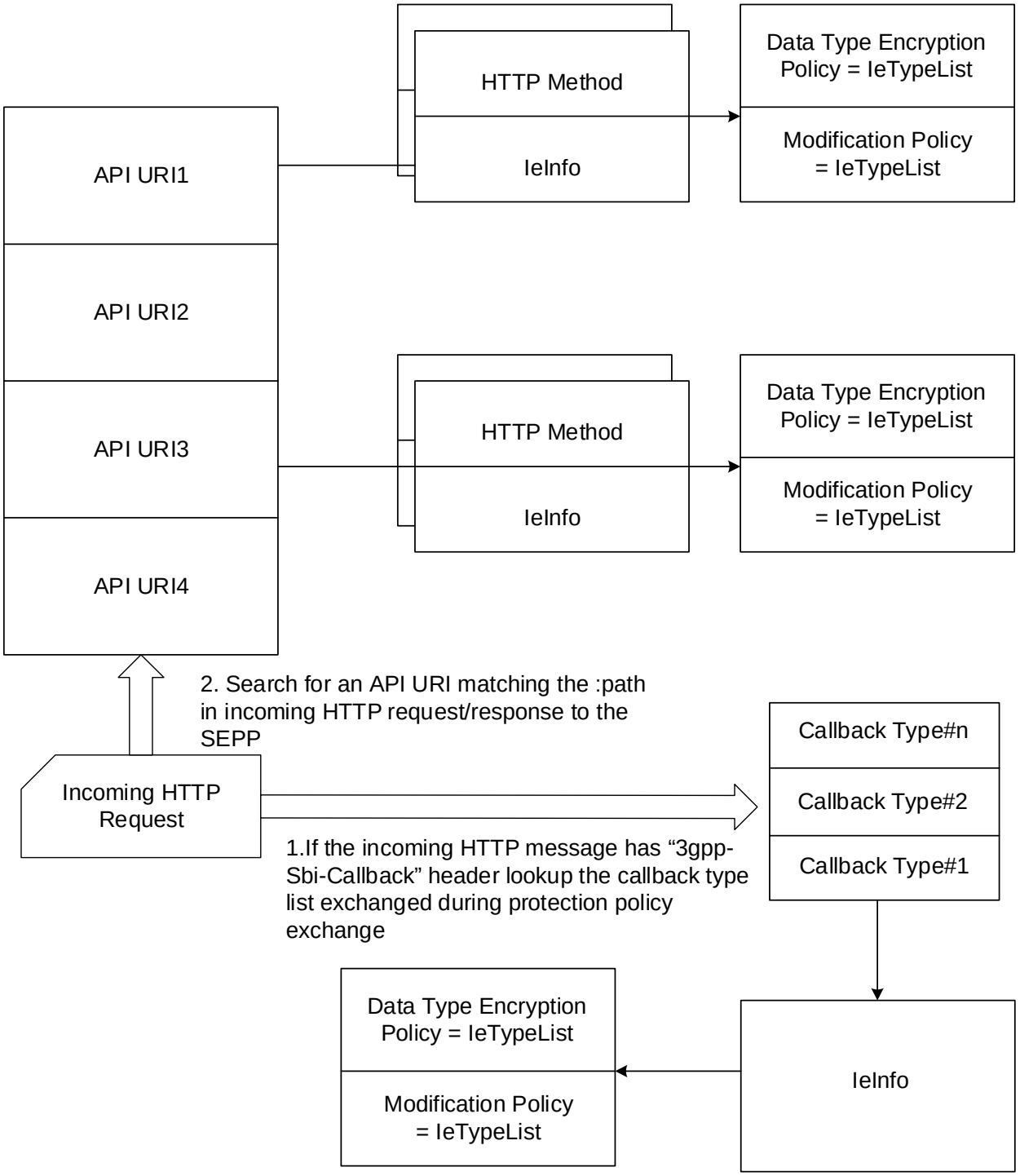

Figure 5.2.3.3-2: Protection Policy Storage and Lookup in SEPP

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.

5.2.4 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/v1/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.

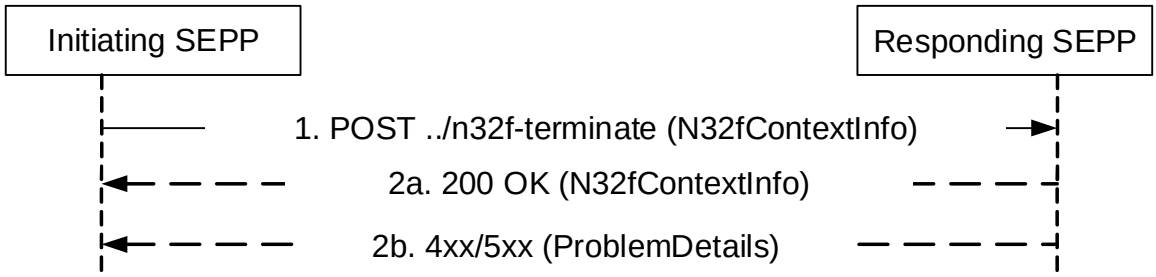

Figure 5.2.4-1: N32f Context Termination Procedure

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 payload body 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".

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/v1/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.

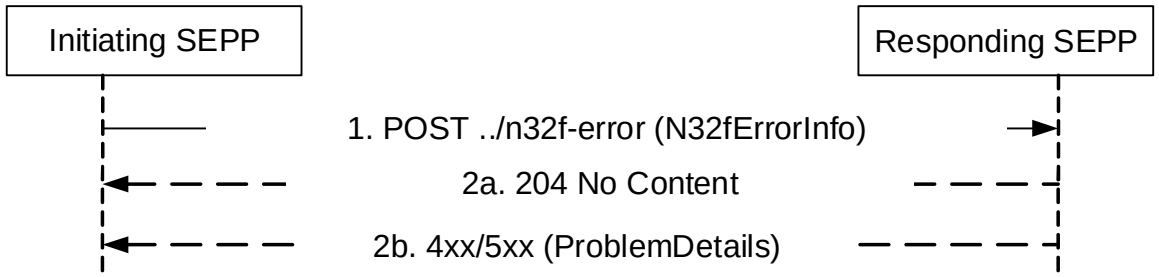

Figure 5.2.5-1: N32f Error Reporting Procedure

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 "messageId" 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 JOSE Protected 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 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. Form the original JSON request / response body from the decrypted ciphertext and the decoded integrity verified "aad" block.

5. 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 over the formed original JSON request / response body from step 4.

6. If the reconstructed HTTP message has a "Authorization" header, then the SEPP shall check whether the service consumer's PLMN 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" 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".

NOTE: 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.

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:

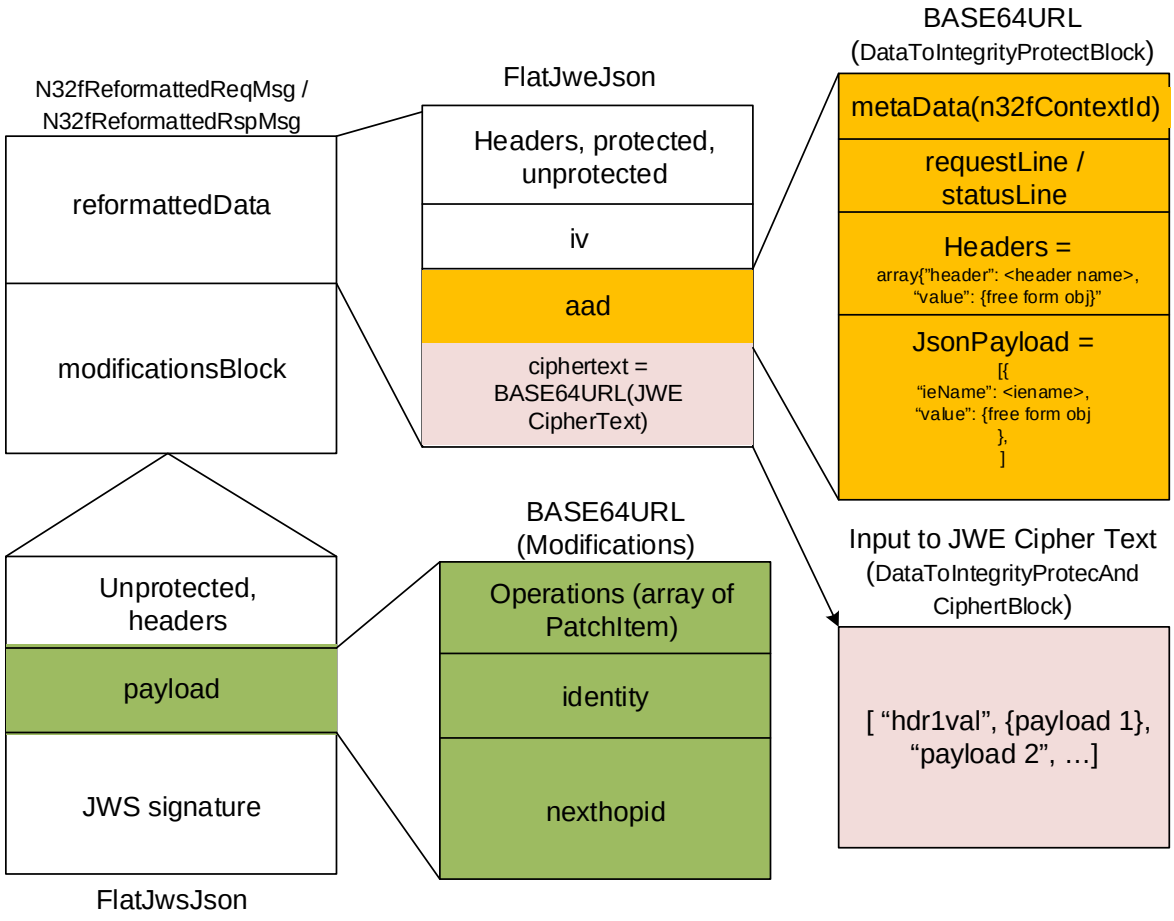

Figure 5.3.2.3-1 JSON representation of a reformatted HTTP message

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

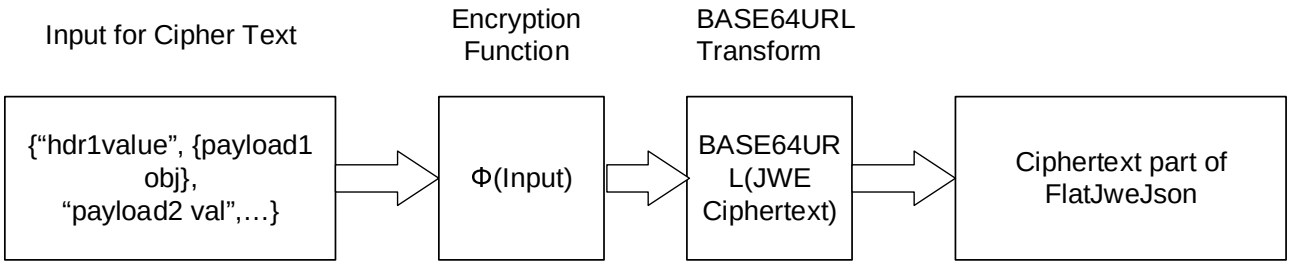

Figure 5.3.2.3-2 Transformation of HTTP Header and Payload to Encrypt into CipherText

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 free form JSON objects 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 payload part 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.

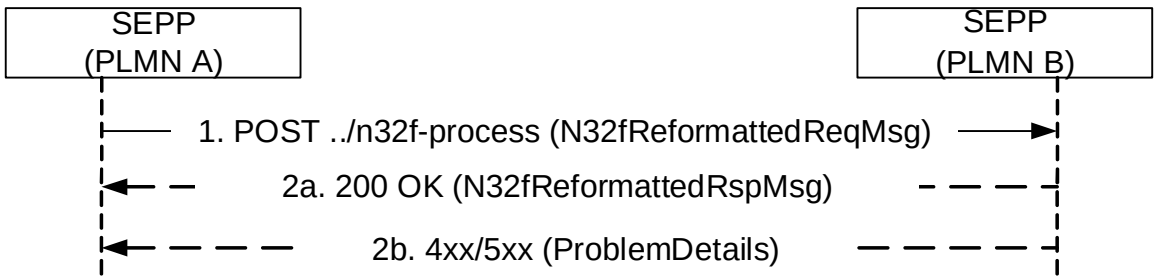

Figure 5.3.2.4-1 Message Forwarding between SEPP on N32-f

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.

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

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

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.

2b. On failure, the responding SEPP shall respond to the initiating SEPP with an appropriate 4xx/5xx status code as specified in clause 6.2.4.2.

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 clauses 5.3.2, 5.3.3 and 5.3.4 shall not be applied. The SEPP shall use a TLS connection towards the other SEPP to forward the HTTP/2 messages sent by the NF service producers and NF service consumers, as is without reformatting.

6 API Definitions

6.1 N32 Handshake API

6.1.1 API URI

URIs of this API shall have the following root:

{apiRoot}/{apiName}/{apiVersion}/

where "apiRoot" is defined in clause 4.4.1 of 3GPP TS 29.501 [5], the "apiName" shall be set to "n32c -handshake" and the "apiVersion" shall be set to "v1" for the current version of this specification.

6.1.2 Usage of HTTP

6.1.2.1 General

HTTP/2, as defined in IETF RFC 7540 [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 [15] 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 JSON format shall be supported. The use of the JSON format (see IETF RFC 8259 [8]) shall be signalled by the content type "application/json" or "application/problem+json". See also clause 5.3.4.

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

Custom operation URI	Mapped HTTP method	Description
{apiRoot}/n32c-handshake/v1/exchange-capability	POST	This is the N32 capability exchange API used to negotiate the security capabilities between SEPPs.
{apiRoot}/n32c-handshake/v1/exchange-params	POST	This is the N32 parameter exchange API used to exchange the cipher suites and protection policies.
{apiRoot}/n32c-handshake/v1/n32f-terminate	POST	This is the N32-f context termination procedure API.
{apiRoot}/n32c-handshake/v1/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. The HTTP method POST shall be used on the following URI:

URI: {apiRoot}/n32c-handshake/v1/exchange-capability

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

Table 6.1.3.2.1-1: URI variables for this Operation

Name	Definition
apiRoot	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.
ProblemDetails	M	1	4xx / 5xx	All the mandatory to support 4xx and 5xx status codes as specified in clause 5.2.7.1 and their corresponding application errors specified in clause 5.2.7.2 of 3GPP TS 29.500 [4] shall be supported.

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/v1/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: URI variables for this Operation

Name	Definition
apiRoot	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 protection policy) depending on what was requested by the requesting SEPP and what is supported by the responding SEPP.
ProblemDetails	M	1	4xx / 5xx	All the mandatory to support 4xx and 5xx status codes as specified in clause 5.2.7.1 and their corresponding application errors specified in clause 5.2.7.2 of 3GPP TS 29.500 [4] shall be supported.

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/v1/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: URI variables for this Operation

Name	Definition
apiRoot	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.
ProblemDetails	M	1	4xx / 5xx	All the mandatory to support 4xx and 5xx status codes as specified in clause 5.2.7.1 and their corresponding application errors specified in clause 5.2.7.2 of 3GPP TS 29.500 [4] shall be supported.

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/v1/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: URI variables for this Operation

Name	Definition
apiRoot	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 errror 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.
ProblemDetails	M	1	4xx / 5xx	All the mandatory to support 4xx and 5xx status codes as specified in clause 5.2.7.1 and their corresponding application errors specified in clause 5.2.7.2 of 3GPP TS 29.500 [4] shall be supported.

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	
ApiSignature	6.1.5.2.9	
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.
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 29.510 [18]	

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 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.

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 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.

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
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.
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	This 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 protection policy requested by the requesting 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
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.
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). When present, this IE shall contain the protection policy selected by the responding 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.

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 "reqBodyIePath" or "rspBodyIePath" (i.e URI parameter 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 URI query attribute 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 RefToBinary IE if the "ieLoc" indicates "MULTIPART_BINARY".
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 URI query attribute 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 RefToBinary IE if the "ieLoc" indicates "MULTIPART_BINARY".
isModifiable	boolean	C	0..1	This IE shall be included if the IE is allowed to be modified by an IPX on the side of the SEPP sending the API IE mapping. When present, - true, indicates that the IE is allowed to be modified by an IPX on the side of the SEPP; - false, indicates that the IE is not allowed to be modified by an IPX on the side of the SEPP; - default is false. When the IE is not included, the default value shall be applied.

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.	
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.

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.
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.

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 1 29.500 [4], Annex B.

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
"TLS"	TLS security.
"PRINS"	PRotocol for N32 INterconnect Security.

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"	IE of type UE identity (e.g SUPI).
"LOCATION"	IE carrying location information.
"KEY_MATERIAL"	IE carrying keying material.
"AUTHENTICATION_MATERIAL"	IE carrying authentication material like authentication vectors and EAP payload.
"AUTHORIZATION_TOKEN"	IE carrying authorization Token
"OTHER"	IE carrying other data requiring encryption.
"NONSENSITIVE"	IE carrying information that are not sensitive.

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 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.

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.
"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.

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.

6.1.5.4 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

There are no application specific errors defined for the N32-c Handshake API in this release of this specification.

6.2 JOSE Protected Message Forwarding API on N32

6.2.1 API URI

URIs of this API shall have the following root:

{apiRoot}/{apiName}/{apiVersion}/

where "apiRoot" is defined in clause 4.4.1 of 3GPP TS 29.501 [5]. 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" shall be set to "n32f-forward" and the "apiVersion" shall be set to "v1" for the current version of this specification. 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 7540 [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 [15] 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 JSON format shall be supported. The use of the JSON format (see IETF RFC 8259 [8]) shall be signalled by the content type "application/json" or "application/problem+json". See also clause 5.3.4.

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

Custom operation URI	Mapped HTTP method	Description
{apiRoot}/n32f-forward/v1/n32f-process	POST	This is the N32f forwarding API used to forward a reformatted and JOSE protected message to a receiving SEPP.

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/v1/n32f-process**

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

Table 6.1.3.2.1-1: URI variables for this Operation

Name	Definition
apiRoot	See clause 6.1.1.

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.
ProblemDetails	M	1	403 Forbidden	This represents the case where the receiving SEPP fails to process the reconstructed message due to PLMN ID verification failure. The "cause" attribute shall be set to "PLMNID_MISMATCH".
ProblemDetails	M	1	4xx / 5xx	All the mandatory to support 4xx and 5xx status codes as specified in clause 5.2.7.1 and their corresponding application errors specified in clause 5.2.7.2 of 3GPP TS 29.500 [4] shall be supported.

6.2.5 Data Model

6.2.5.1 General

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

Table 6.3.5.1-1 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	
N32fReformattedRspMsg	6.2.5.2.3	
AuthenticatedBlock	6.2.5.2.4	
ClearTextBlock	6.2.5.2.5	
RequestLine	6.2.5.2.6	
HttpHeader	6.2.5.2.7	
HttpPayload	6.2.5.2.8	
MetaData	6.2.5.2.9	
Modifications	6.2.5.2.10	
FlatJweJson	6.2.5.2.11	
FlatJwsJson	6.2.5.2.12	
IndexToEncryptedValue	6.2.5.2.13	
EncodedHttpHeaderValue	6.2.5.2.14	

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 6.2.5.1-2: N32 re-used Data Types

Data type	Reference	Comments
HttpMethod	6.1.5.3.5	
PatchItem	3GPP 29.571 [12]	
UriScheme	3GPP 29.571 [12]	
Fqdn	3GPP 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 payload 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 payload 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 on path are allowed to apply modification policies and if they have any specific modification to be applied on the message contained in the authenticatedBlock.

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 payload 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 payload 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 on path are allowed to apply modification policies and if they have any specific modification to be applied on the message contained in the authenticatedBlock.

6.2.5.2.4 Type: DataToIntegrityProtectAndCipherBlock

Table 6.2.5.2.4-1: Definition of type DataToIntegrityProtectBlock

Attribute name	Data type	P	Cardinality	Description
dataToEncrypt	array(object)	M	1..N	This IE shall contain the input for ciphering as a JSON object block containing an array of free form objects with each entry of the array containing 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 ClearTextBlock

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.
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 payload that needs to be sent in clear text. When present this IE shall contain the encoding of JSON payload 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 API invoked by the NF service consumer / producer behind the SEPP towards its peer NF service in the other PLMN.
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.
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 fragment part of the API, if available.

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 RefToBinary type IE that is referring to the multipart binary parth>/contenttype
				  "ieValueLocation": "MULTIPART_BINARY"
				  "value": <value of the content type of multipart binary>
				},
				{
				  "iePath": <JSON Pointer of RefToBinary type IE that is referring to the multipart binary parth>/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 RefToBinary type IE that is referring to the multipart binary parth>/contenttype
				  "ieValueLocation": "MULTIPART_BINARY"
				  "value": <value of the content type of multipart binary>
				},
				{
				  "iePath": <JSON path of RefToBinary type IE that is referring to the multipart binary parth>/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 provided by the initiating SEPP to the responding SEPP during the parameter exchange procedure (see clause 5.2.3).
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 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".

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 "authenticatedBlock" 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.
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.

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).
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).

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	

6.2.5.3 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.

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 1: 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 the public 3GPP file server in the following locations (see clause 5B of the 3GPP TR 21.900 [20] for further information):

- https://www.3gpp.org/ftp/Specs/archive/OpenAPI/<Release>/, and
- https://www.3gpp.org/ftp/Specs/<Plenary>/<Release>/OpenAPI/.

NOTE 2: To fetch the OpenAPI specification file after CT#83 plenary meeting for Release 15 in the above links <Plenary> must be replaced with the date the CT Plenary occurs, in the form of year-month (yyyy-mm), e.g. for CT#83 meeting <Plenary> must be replaced with value "2019-03" and <Release> must be replaced with value "Rel-15".

A.2 N32 Handshake API

```
openapi: 3.0.0

info:
  version: '1.0.1'
  title: 'N32 Handshake API'
  description: |
  N32-c Handshake Service.
    © 2019, 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 V15.2.0; 5G System; Public Land Mobile Network (PLMN) Interconnection;
Stage 3
  url: http://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'
      '400':
        $ref: 'TS29571_CommonData.yaml#/components/responses/400'
      '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'
      '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'
      '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'
      '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'
        '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'
        '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'
        '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'
        '503':
          $ref: 'TS29571_CommonData.yaml#/components/responses/503'
        default:
          description: Unexpected error
components:
  schemas:
    SecurityCapability:
      anyOf:
        - type: string
          enum:
            - TLS
            - PRINS
        - type: string
    ApiSignature:
      oneOf:
        - $ref: 'TS29571_CommonData.yaml#/components/schemas/Uri'
        - $ref: '#/components/schemas/CallbackName'
    HttpMethod:
      anyOf:
        - type: string
          enum:
            - GET
            - PUT
            - POST
            - DELETE
            - PATCH
            - HEAD
            - OPTIONS
            - CONNECT
            - TRACE
        - type: string

    IeType:
      anyOf:
        - type: string
```

```
        enum:
          - UEID
          - LOCATION
          - KEY_MATERIAL
          - AUTHENTICATION_MATERIAL
          - AUTHORIZATION_TOKEN
          - OTHER
          - NONSENSITIVE
      - type: string

    IeLocation:
      anyOf:
        - type: string
          enum:
            - URI_PARAM
            - HEADER
            - BODY
            - MULTIPART_BINARY
        - type: string

    IeInfo:
      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

    ApiIeMapping:
      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

    ProtectionPolicy:
      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:
      type: object
      required:
        - sender
        - supportedSecCapabilityList
      properties:
        sender:
          $ref: 'TS29510_Nnrf_NFManagement.yaml#/components/schemas/Fqdn'
```

```
      supportedSecCapabilityList:
        type: array
        items:
          $ref: '#/components/schemas/SecurityCapability'
        minItems: 1

  SecNegotiateRspData:
    type: object
    required:
      - sender
      - selectedSecCapability
    properties:
      sender:
        $ref: 'TS29510_Nnrf_NFManagement.yaml#/components/schemas/Fqdn'
      selectedSecCapability:
        $ref: '#/components/schemas/SecurityCapability'

  SecParamExchReqData:
    type: object
    required:
      - n32fContextId
    properties:
      n32fContextId:
        type: string
      jweCipherSuiteList:
        type: array
        items:
          type: string
        minItems: 1
      jwsCipherSuiteList:
        type: array
        items:
          type: string
        minItems: 1
      protectionPolicyInfo:
        $ref: '#/components/schemas/ProtectionPolicy'

  SecParamExchRspData:
    type: object
    required:
      - n32fContextId
    properties:
      n32fContextId:
        type: string
      selectedJweCipherSuite:
        type: string
      selectedJwsCipherSuite:
        type: string
      selProtectionPolicyInfo:
        $ref: '#/components/schemas/ProtectionPolicy'

  N32fContextInfo:
    type: object
    required:
      - n32fContextId
    properties:
      n32fContextId:
        type: string
  CallbackName:
    type: object
    required:
      - callbackType
    properties:
      callbackType:
        type: string
  N32fErrorInfo:
    type: object
    required:
      - n32fMessageId
      - n32fErrorType
    properties:
      n32fMessageId:
        type: string
      n32fErrorType:
        $ref: '#/components/schemas/N32fErrorType'
      failedModificationList:
        type: array
        items:
```

```
            $ref: '#/components/schemas/FailedModificationInfo'
          minItems: 1
        errorDetailsList:
          type: array
          items:
            $ref: '#/components/schemas/N32fErrorDetail'
          minItems: 1
    FailedModificationInfo:
      type: object
      required:
        - ipxId
        - n32fErrorType
      properties:
        ipxId:
          $ref: 'TS29510_Nnrf_NFManagement.yaml#/components/schemas/Fqdn'
        n32fErrorType:
          $ref: '#/components/schemas/N32fErrorType'
    N32fErrorDetail:
      type: object
      required:
        - attribute
        - msgReconstructFailReason
      properties:
        attribute:
          type: string
        msgReconstructFailReason:
          $ref: '#/components/schemas/FailureReason'
    N32fErrorType:
      anyOf:
        - type: string
          enum:
            - INTEGRITY_CHECK_FAILED
            - INTEGRITY_CHECK_ON_MODIFICATIONS_FAILED
            - MODIFICATIONS_INSTRUCTIONS_FAILED
            - DECIPHERING_FAILED
            - MESSAGE_RECONSTRUCTION_FAILED
        - type: string
    FailureReason:
      anyOf:
        - type: string
          enum:
            - INVALID_JSON_POINTER
            - INVALID_INDEX_TO_ENCRYPTED_BLOCK
            - INVALID_HTTP_HEADER
        - type: string
```

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

```
openapi: 3.0.0

info:
  version: '1.0.1'
  title: 'JOSE Protected Message Forwarding API'
  description: |
    N32-f Message Forwarding Service.
    © 2019, 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 V15.2.0; 5G System; Public Land Mobile Network (PLMN) Interconnection;
Stage 3
  url: http://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
```

```
      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'
        '400':
          $ref: 'TS29571_CommonData.yaml#/components/responses/400'
        '403':
          $ref: 'TS29571_CommonData.yaml#/components/responses/403'
        '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'
        '503':
          $ref: 'TS29571_CommonData.yaml#/components/responses/503'
        default:
          description: Unexpected error
components:
  schemas:
    FlatJweJson:
      type: object
      required:
        - ciphertext
      properties:
        protected:
          type: string
        unprotected:
          type: object
        header:
          type: object
        encrypted_key:
          type: string
        aad:
          type: string
        iv:
          type: string
        ciphertext:
          type: string
        tag:
          type: string

    FlatJwsJson:
      type: object
      required:
        - payload
        - signature
      properties:
        payload:
          type: string
        protected:
          type: string
        header:
          type: object
        signature:
          type: string

    N32fReformattedReqMsg:
      type: object
      required:
        - reformattedData
      properties:
        reformattedData:
          $ref: '#/components/schemas/FlatJweJson'
```

```
      modificationsBlock:
        type: array
        items:
          $ref: '#/components/schemas/FlatJwsJson'
        minItems: 1

  N32fReformattedRspMsg:
    type: object
    required:
      - reformattedData
    properties:
      reformattedData:
        $ref: '#/components/schemas/FlatJweJson'
      modificationsBlock:
        type: array
        items:
          $ref: '#/components/schemas/FlatJwsJson'
        minItems: 1

  DataToIntegrityProtectAndCipherBlock:
    type: object
    required:
      - dataToEncrypt
    properties:
      dataToEncrypt:
        type: array
        items:
          type: object
        minItems: 1
  DataToIntegrityProtectBlock:
    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:
    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
  HttpHeader:
    type: object
    required:
      - header
      - value
    properties:
      header:
        type: string
      value:
```

```
      $ref: '#/components/schemas/EncodedHttpHeaderValue'
    HttpPayload:
      type: object
      required:
        - iePath
        - ieValueLocation
        - value
      properties:
        iePath:
          type: string
        ieValueLocation:
          $ref: 'TS29573_N32_Handshake.yaml#/components/schemas/IeLocation'
        value:
          type: object
    MetaData:
      type: object
      required:
        - n32fContextId
        - messageId
        - authorizedIpxId
      properties:
        n32fContextId:
          type: string
        messageId:
          type: string
        authorizedIpxId:
          type: string
    Modifications:
      type: object
      required:
        - identity
      properties:
        identity:
          $ref: 'TS29510_Nnrf_NFManagement.yaml#/components/schemas/Fqdn'
        operations:
          type: array
          items:
            $ref: 'TS29571_CommonData.yaml#/components/schemas/PatchItem'
          minItems: 1
    IndexToEncryptedValue:
      type: object
      required:
        - encBlockIndex
      properties:
        encBlockIndex:
          $ref: 'TS29571_CommonData.yaml#/components/schemas/Uinteger'
    EncodedHttpHeaderValue:
      oneOf:
        - type: string
        - $ref: '#/components/schemas/IndexToEncryptedValue'
```

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 payload 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 payload 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>, "nextHopId": <FQDN of 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 payload 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 payload 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>, "nextHopId": <FQDN of 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 a RefToBinary type IE>,
        "ieValueLocation": "BODY",
        "value": <value of the Content ID>
      },
      {
        "iePath": <JSON Pointer of IE 2 - which is a RefToBinary type IE>/contenttype,
        "ieValueLocation": "MULTIPART_BINARY",
        "value": <value of the Content Type>
      },
      {
        "iePath": <JSON Pointer of IE 2 - which is a RefToBinary type IE>/data,
        "ieValueLocation": "MULTIPART_BINARY",
        "value": <BASE 64 encoded byte array of the binary part>
      }
      {
        "iePath": <JSON Pointer of IE 3 - which is a RefToBinary type IE>,
        "ieValueLocation": "BODY",
        "value": <value of the Content ID>
      },
      {
        "iePath": <JSON Pointer of IE 2 - which is a RefToBinary type IE>/contenttype,
        "ieValueLocation": "MULTIPART_BINARY",
        "value": <value of the Content Type>
      },
      {
        "iePath": <JSON Pointer of IE 3 - which is a RefToBinary type IE>/data,
        "ieValueLocation": "MULTIPART_BINARY",
        "value": {"encBlockIndex": 2}
      }
    ]
}
```

The DataToIntegrityProtectAndCipherBlock for this example looks like

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

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 PLMN 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.

C.2 TLS security between SEPPs

C.2.1 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 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.

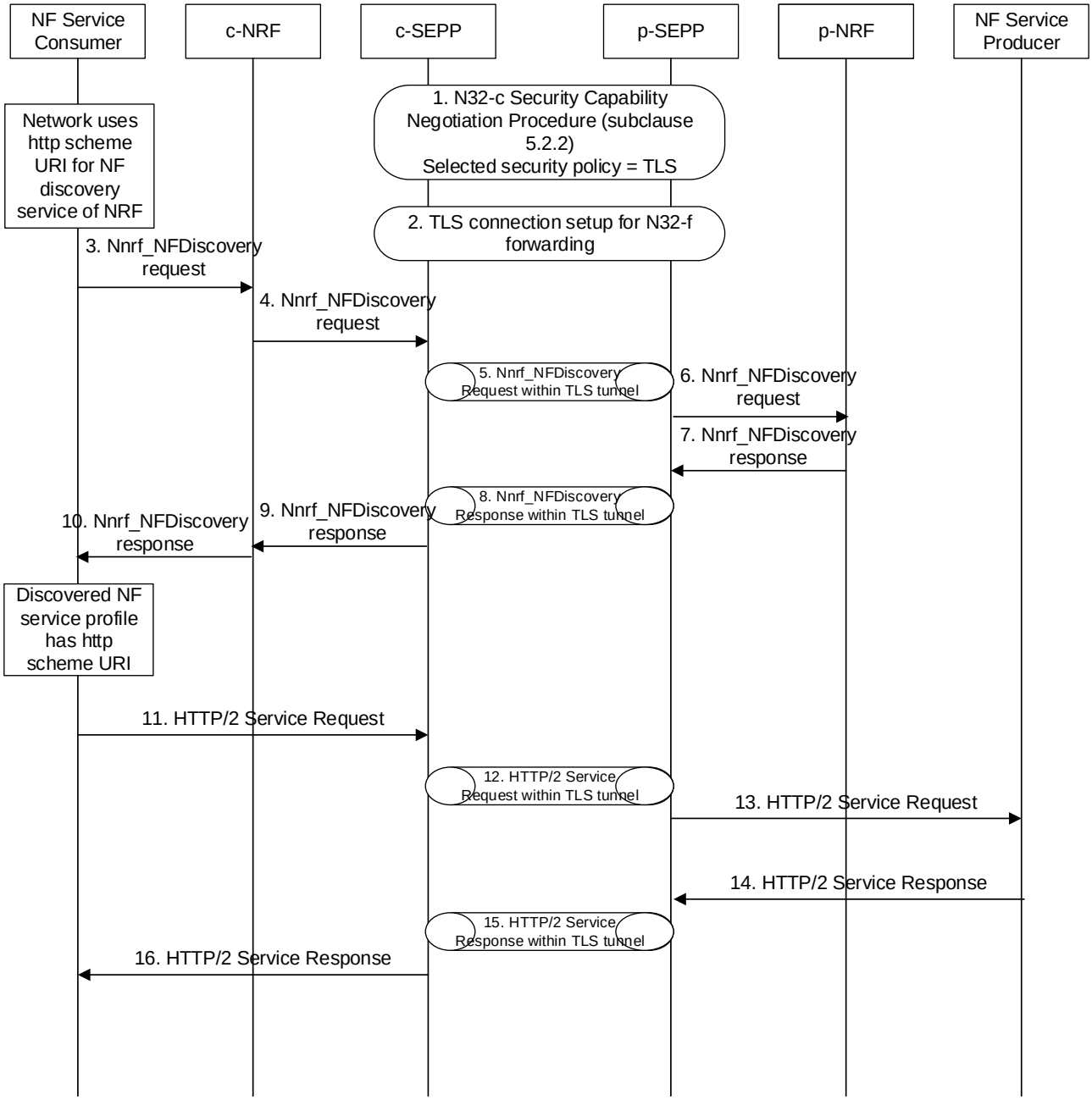

Figure C.2.1-1 End to end call flow when http scheme URI is used and TLS security is used between SEPPs

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 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.2.2 When https 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 TLS;
- "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.

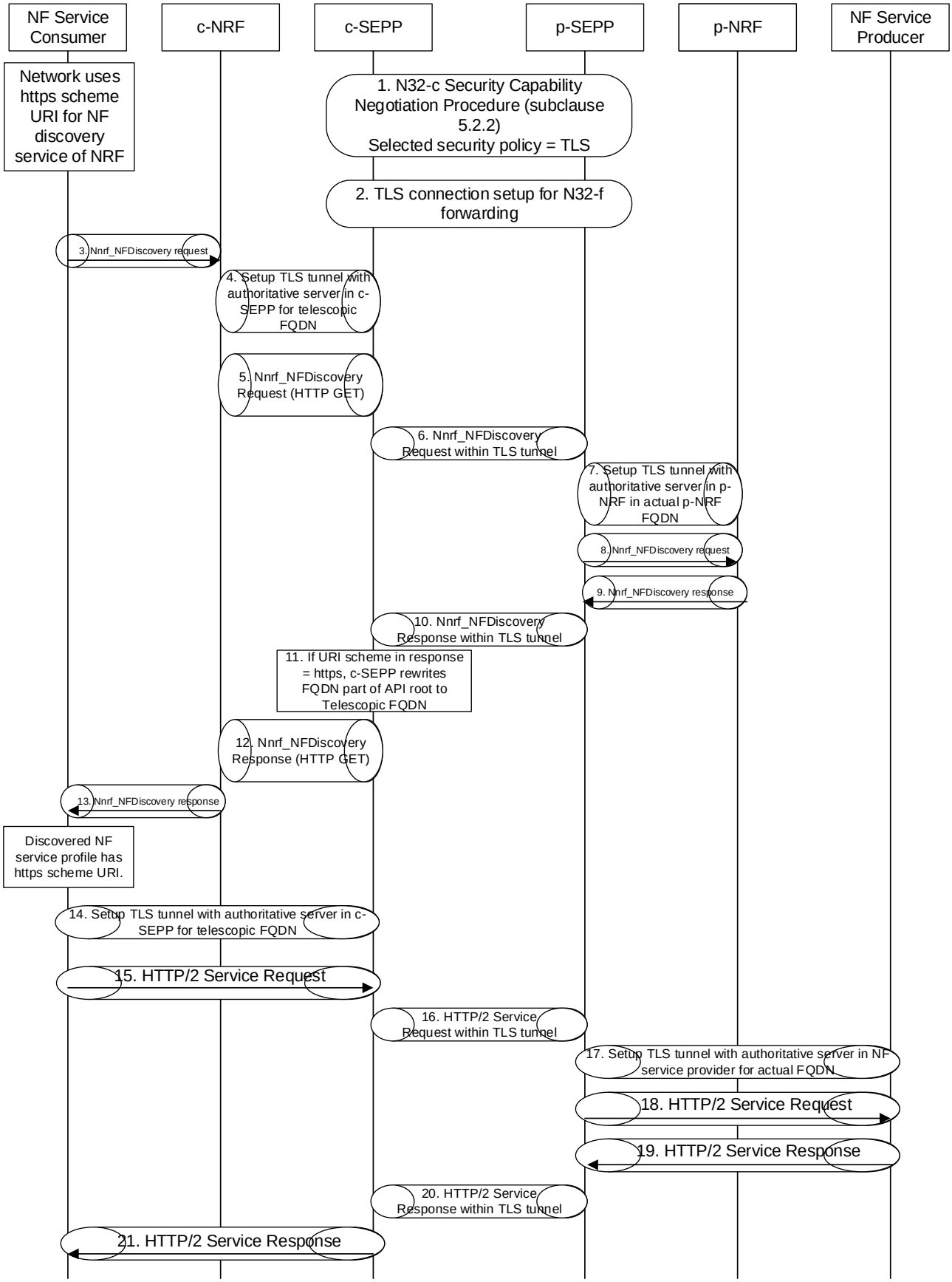
NF Service Consumer
c-NRF
c-SEPP
p-SEPP
p-NRF
NF Service Producer
Network uses https scheme URI for NF discovery service of NRF
1. N32-c Security Capability Negotiation Procedure (subclause 5.2.2) Selected security policy = TLS
2. TLS connection setup for N32-f forwarding
3. Nnrf_NFDiscovery request
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.2.2-1 End to end call flow when https scheme URI is used and TLS 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 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 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 label part of the telescopic FQDN in the request URI with a corresponding label of the p-SEPP (if the label part of the telescopic FQDN contains a label of c-SEPP's local significance) 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, replaces the label part of the the telescopic FQDN in the request URI to the URI of the p-NRF's NF discovery service 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. 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 label part of the telescopic FQDN in the request URI with a corresponding label of the p-SEPP 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, replaces the label part of the the telescopic FQDN in the request URI to the URI of the NF service producer 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. 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.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.

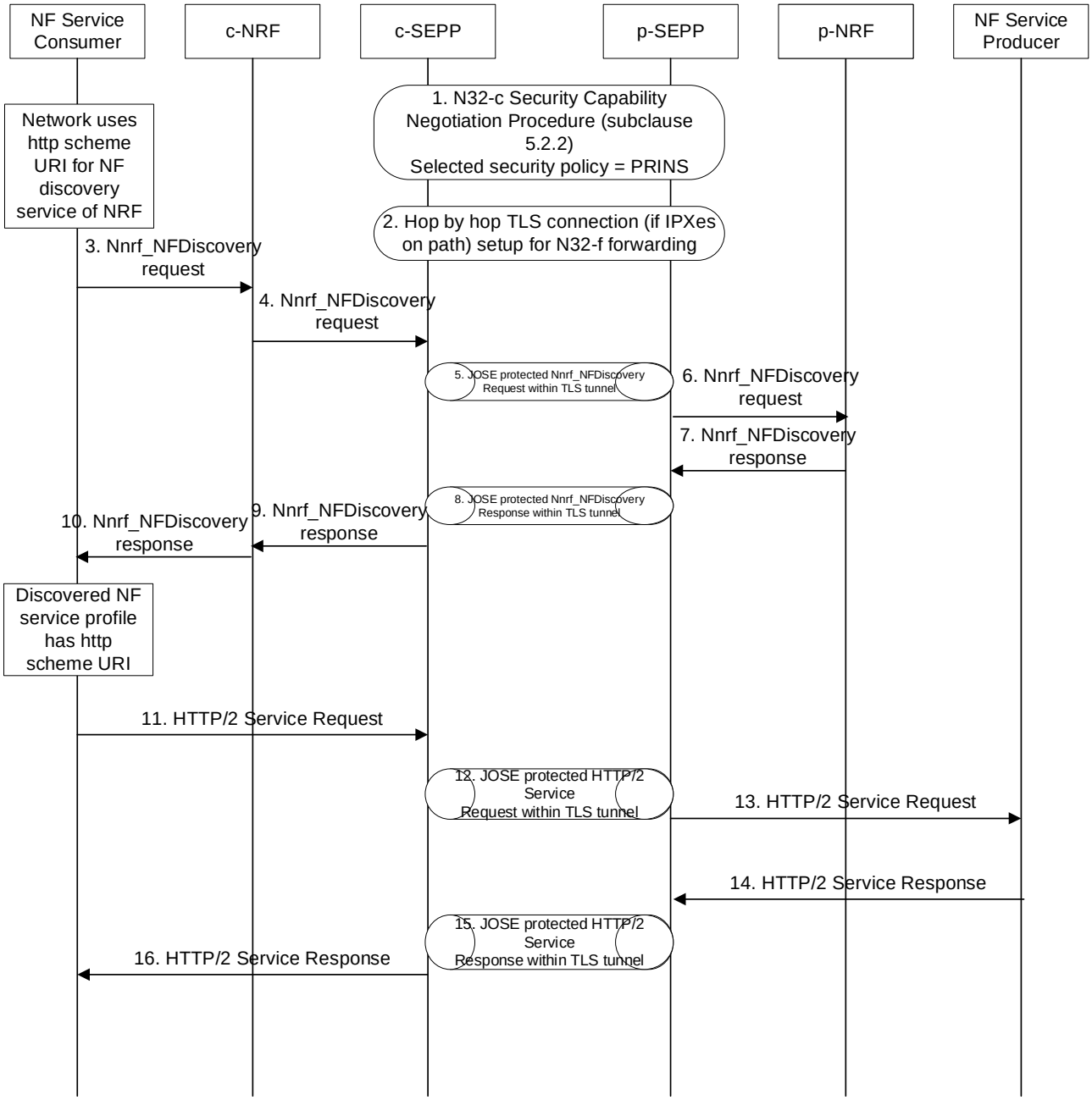

Figure C.3.1-1 End to end call flow when http scheme URI is used and "PRINS" security is used between SEPPs

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 hop by 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.
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.
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

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.

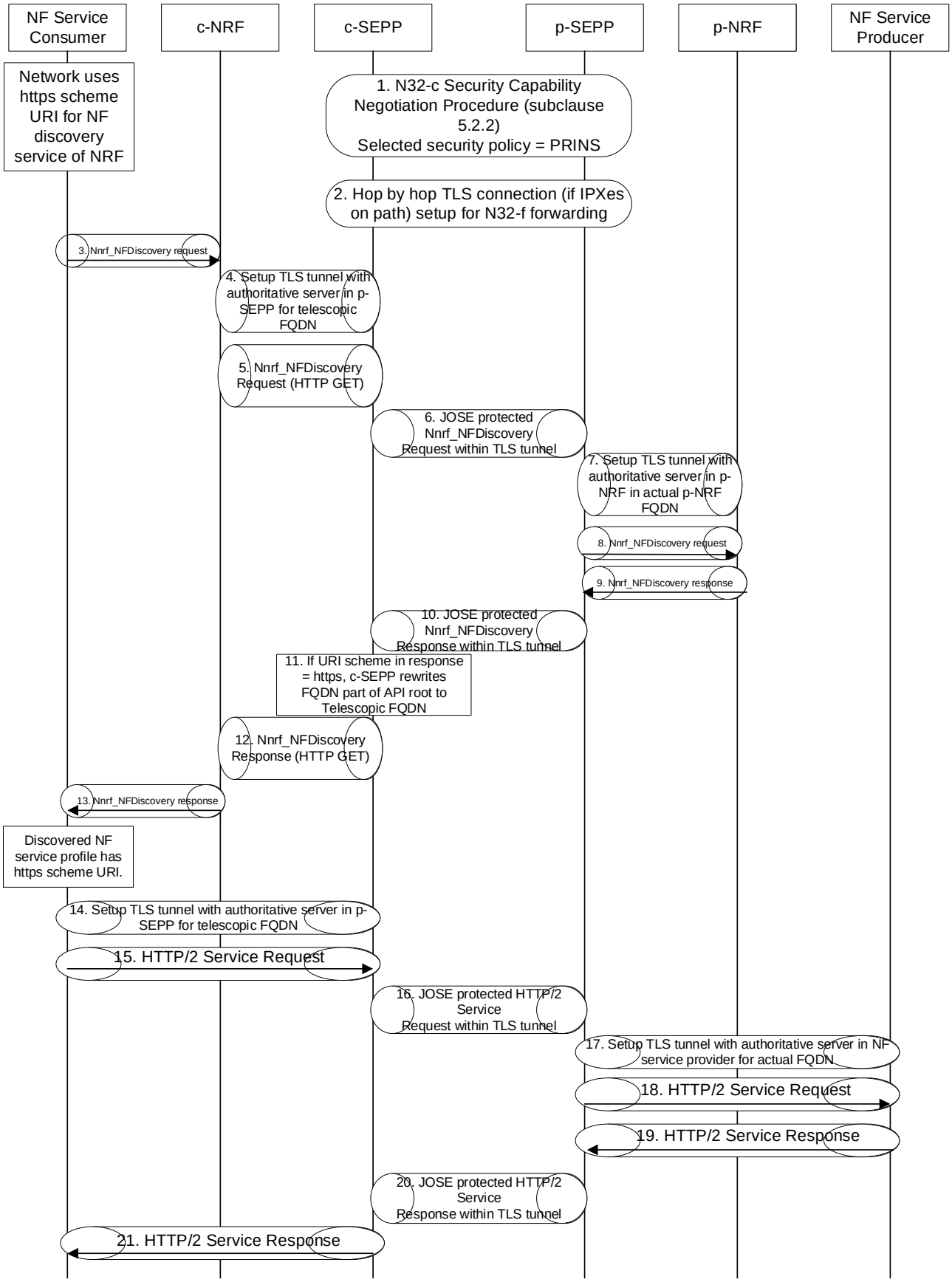
NF Service Consumer
c-NRF
c-SEPP
p-SEPP
p-NRF
NF Service Producer
Network uses https scheme URI for NF discovery service of NRF
1. N32-c Security Capability Negotiation Procedure (subclause 5.2.2) Selected security policy = PRINS
2. Hop by hop TLS connection (if IPXes on path) setup for N32-f forwarding
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.3.2-1 End to end call flow when https 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 TLS 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 hop by 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 label part of the telescopic FQDN in the request URI with a corresponding label of the p-SEPP (if the label part of the telescopic FQDN contains a label of c-SEPP's local significance) 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 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, replaces the label part of the the telescopic FQDN in the request URI to the URI of the p-NRF's NF discovery service 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. 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 label part of the telescopic FQDN in the request URI with a corresponding label of the p-SEPP 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.

17. The p-SEPP extracts the HTTP message received on the TLS connection, replaces the label part of the the telescopic FQDN in the request URI to the URI of the NF service producer 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. 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].

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 (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

History

Document history		
V15.1.0	April 2019	Publication
V15.2.0	July 2019	Publication
V15.3.0	October 2019	Publication