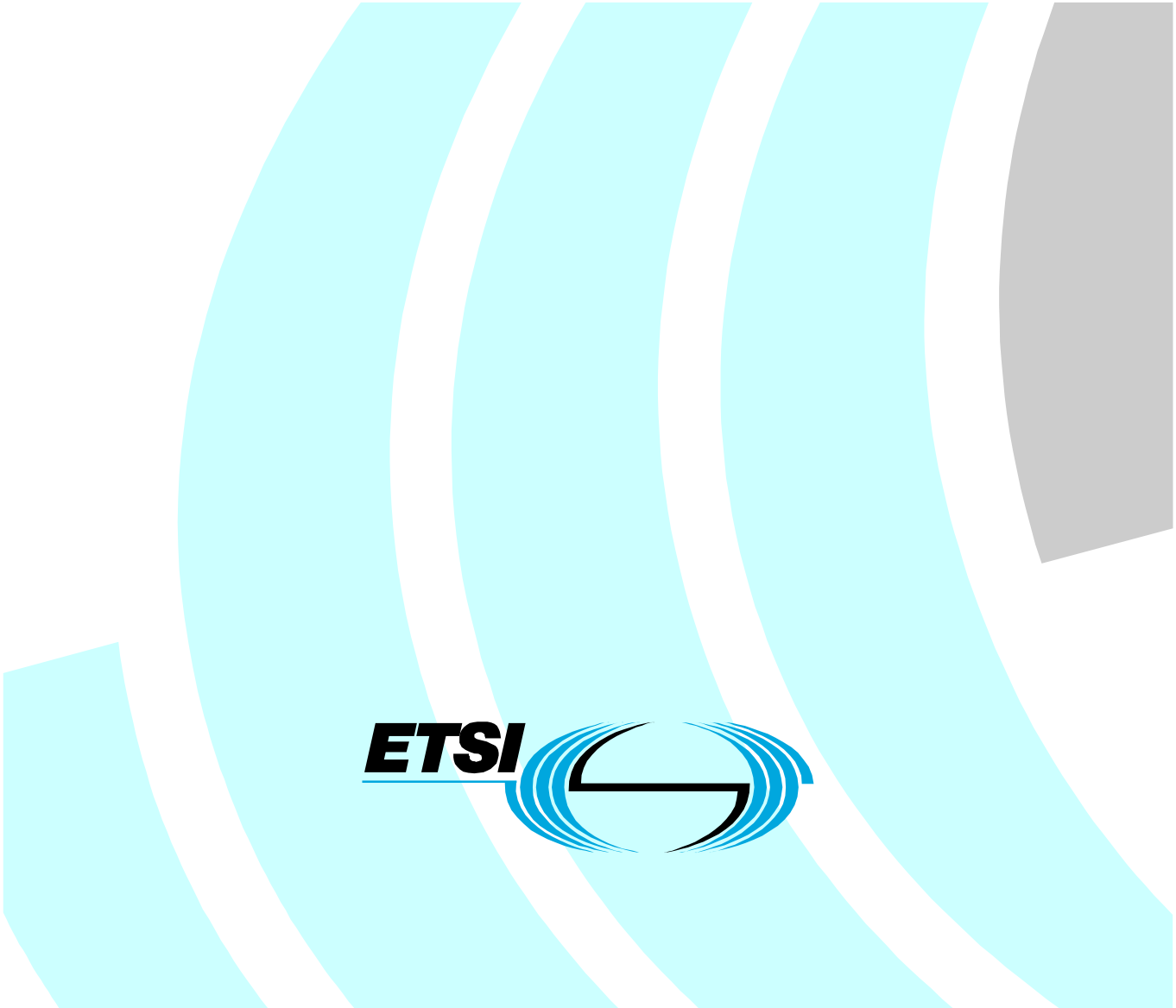
ETSI TS 101 882 V1.1.1 (2002-05)

Technical Specification

Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON) Release 3; Protocol Framework Definition; General (meta-protocol)



Reference
DTS/TIPHON-03016

Keywords
interface, IP, protocol, VoIP

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

Contents

Intellectual Property Rights

IPRs essential or potentially essential to the present document 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 (http://webapp.etsi.org/IPR/home.asp).

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.

Foreword

This Technical Specification (TS) has been produced by ETSI Project Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON).

Introduction

The present document is a product in TIPHON release 3 (see TR 101 301 [6]) of step D of the TIPHON development process described in TR 101 835 [7].

The approach being taken to standardization in TIPHON represents a departure from that used in the past for PSTN, ISDN and GSM. Its aim is to allow much greater scope for competition through innovation in the design of equipment and services. Its aim is also to provide adequate standardization to facilitate the operation of services across interconnected networks, even networks that use different technologies. The present document presents the initial core set of service capabilities envisaged to be required to enable service providers to offer services on TIPHON networks that may safely inter-work with existing PSTN services while enabling more advanced services to be subsequently developed.

Figure 1 shows the relationship of the present document with other TIPHON Release 3 deliverables.

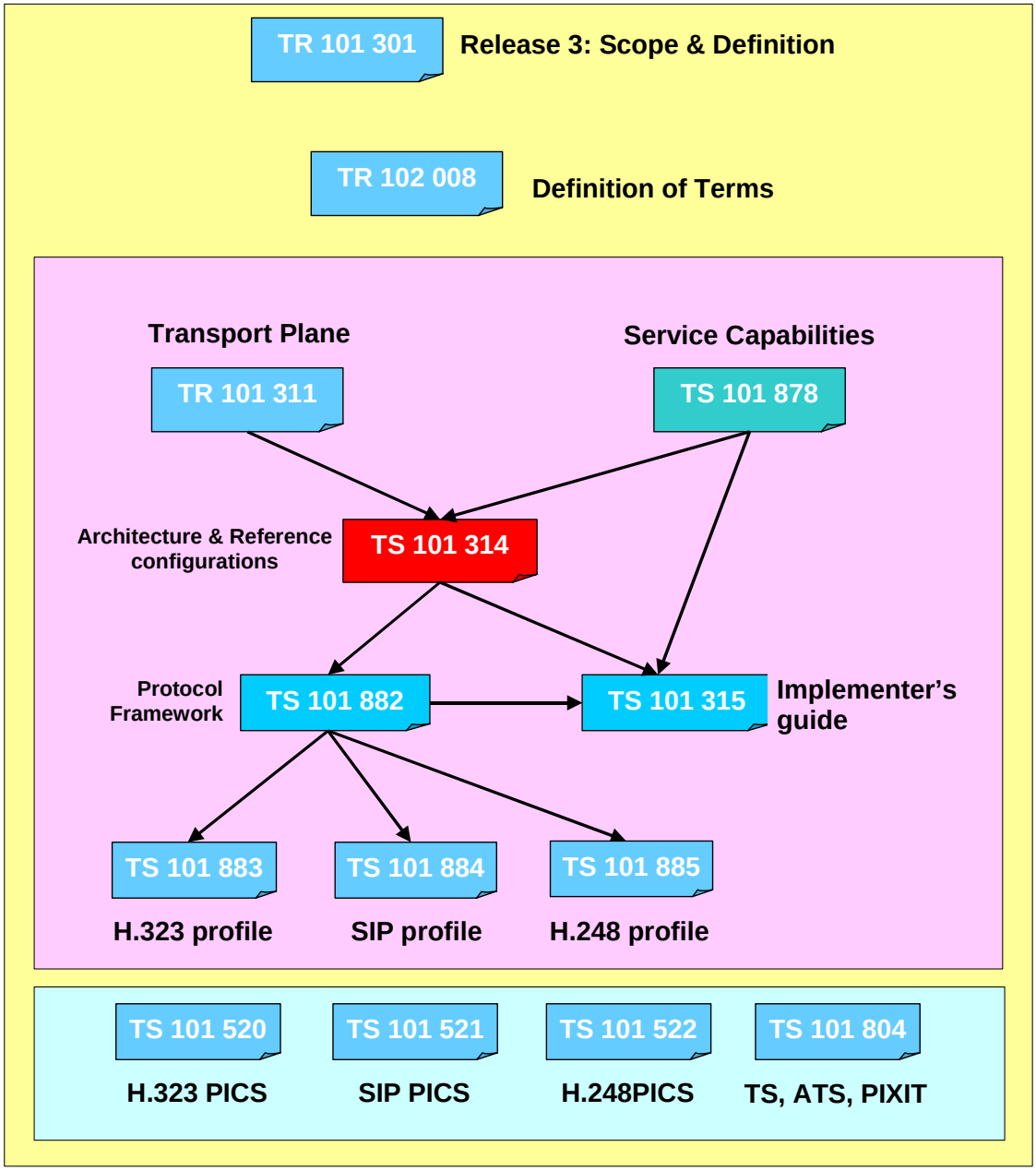


Figure 1: Relationship with other TIPHON Release 3 documents

- TR 101 311 [9] provides the requirements on the transport plane;
- TS 101 878 [3] defines service capabilities that are used in the TIPHON Release 3 for a simple call;
- TS 101 882 (the present document) provides the Protocol Framework based on the TIPHON Release 3 architecture to implement the simple call service capabilities as defined in the present document;
- TS 101 315 [5] is an implementer's guide that shows how to use of the meta-protocol to realize the capabilities as defined in TS 101 878 [3];
- TS 101 883 [10] provides the protocol mappings for the ITU-T H-323 profile;
- TS 101 884 provides the protocol mappings for the SIP profile;
- TS 101 885 [11] provides the protocol mappings for the ITU-T H-248 profile;
- TS 101 314 [1] provides the architecture and reference configurations for TIPHON Release 3.

1 Scope

The present document defines protocol frameworks for reference points defined in the TIPHON architecture TS 101 314 [1] that are required to implement the capabilities described in TS 101 878 [3] such that implementations compliant to the framework using candidate protocols interoperate.

The protocol framework takes the form of a set of meta-protocols described both in syntax (using Abstract Syntax Notation 1 (ASN.1 (see bibliography)) and in behaviour (using Message Sequence Charts (MSCs (see ITU-T Recommendation Z.120 in bibliography) and simple Specification and Description Language (SDL (see ITU-T Recommendation Z.100 in bibliography)) diagrams in addition to text). The meta-protocols show both the service primitives used by higher or lower layers to invoke, control and report on the progress of the meta-protocol, and the Meta Protocol Data Units (M-PDUs) used to communicate with peer entities.

Meta-protocols are described in this edition of the present document for the following reference points defined in TS 101 314 [1]: R; C; N and T.

The present document is applicable to the protocols that are necessary to support TIPHON Release 3.

Where the text indicates the status of a requirement (i.e. as strict command or prohibition, as authorizations leaving freedom or as a capability or possibility), this may modify the nature of a requirement within a candidate protocol used to provide the capability.

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 and/or edition number or version number) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies.

[1] ETSI TS 101 314: "Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON) Release 3; Abstract Architecture and Reference Points Definition; Network Architecture and Reference Points".

[2] ETSI TR 101 877: "Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON); Requirements Definition Study; Scope and Requirements for a Simple call".

[3] ETSI TS 101 878: "Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON) Release 3; Service Capability Definition; Service Capabilities for a simple call".

[4] ISO/IEC 9646-7: "Information technology - Open Systems Interconnection - Conformance testing methodology and framework - Part 7: Implementation Conformance Statements".

[5] ETSI TS 101 315: "Telecommunications and Internet protocol Harmonization Over Networks (TIPHON) Release 3; Functional Entities, Information Flow and Reference Point Definitions; Guidelines for application of TIPHON functional architecture to inter-domain services".

[6] ETSI TR 101 301: "Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON) Release 3; Release Definition; TIPHON Release 3 Definition".

[7] ETSI TR 101 835: "Telecommunications and Internet Protocol Harmonization over Networks (TIPHON); Project method definition".

[8] ETSI TR 102 008: "Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON) Release 3; Terms and Definitions".

[9] ETSI TR 101 311: "Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON) Release 3; Service Independent requirements definition; Transport Plane".

[10] ETSI TS 101 883: "Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON) Release 3; Technology Mapping; Implementation of TIPHON architecture using H.323".

[11] ETSI TS 101 885: "Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON) Release 3; Technology Mapping; Technology Mapping of TIPHON reference point N to H.248/MEGACO protocol".

[12] ETSI TS 101 520: "Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON); Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON); Support of ITU-T Recommendation H.323".

[13] ETSI TS 101 521: "Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON); Protocol Implementation Conformance Statement (PICS) proforma for the support of call signalling protocols and media stream packetization for packet-based multimedia communication systems; Support of ITU-T Recommendation H.225.0".

[14] ETSI TS 101 522: "Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON); Protocol Implementation Conformance Statement (PICS) proforma for the support of control protocol for multimedia communication; Support of ITU-T Recommendation H.245".

[15] ETSI TS 101 804: "Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON) Release 3; Technology compliance specifications".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in TR 101 877 [2] and TS 101 878 [3] apply.

3.2 Abbreviations

For the purposes of the present document, the abbreviations defined in TR 101 877 [2], TS 101 878 [3] and the following apply:

API	Application Programming Interface
ASN.1	Abstract Syntax Notation One
BC	Bearer Control
CC	Call Control
CCUA	Call Control User Agent
FE	Functional Entity
FG	Functional Grouping
GoS	Grade of Service
IP	Internet Protocol
IPTN	IP Telephony Network
ISDN	Integrated Services Digital Network
MC	Media Control
M-PDU	Meta Protocol Data Unit
MSC	Message Sequence Chart
PCM	Pulse Code Modulation
PDU	Protocol Data Unit
PSTN	Public Switched Telephone Network
QoS	Quality of Service
SAP	Service Access Point
SC	Service Control
SCN	Switched Circuit Networks
SDL	Specification and Description Language
SL	Service Layer
SNCC	Serving Network Call Control

TCC-SAP	TIPHON Call Control SAP
TLL-SAP	TIPHON Lower Layer SAP
TNCC	Transit Network Call Control
TRL	TIPHON Resource Location
TR-SAP	TIPHON Registration SAP
TT-SAP	TIPHON Transport SAP
URI	Uniform Resource Identifier

4 Introduction

The annexes in the present document contain normative descriptions of the meta-protocols that apply to the reference points defined in TS 101 314 [1]. Each annex is complete and the interactions between the meta-protocols are further described in TS 101 315 [5].

Annex A (normative): Meta-protocol at reference point R

TIPHON networks shall offer a Registration point of Attachment (RpoA). If the RpoA is not offered then the network cannot be considered as TIPHON enabled.

EXAMPLE: If a TIPHON network is IP based with configuration of terminals (hosts) offered by DHCP then the DHCP procedure will give the terminal its IP address parameters, DNS parameters, and the Registration point of Attachment (RpoA) as well as "anonymous" services that the user of that transport domain can invoke without prior authorization (e.g. emergency services and information services).

A.1 Overview

The registration meta-protocol operations enable a user (the registrant) to seek and gain authority to invoke service in some domain for which ingress/egress is strictly controlled. The service applications to be offered shall be determined, in part, by data held in the user profile.

Figure 2 shows the relationship of the core elements in registration.

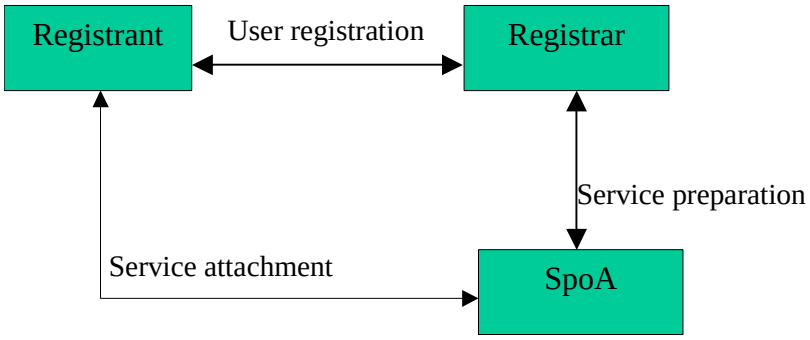


Figure 2: Relationship of registration entities

The registration service shall enable a user to receive service in both home and visited domains. The registrar to which a registrant registers shall maintain a profile of service applications for the registrant. The registrar may authorize access to service applications in the domain in which the registrant is situated or in some other domain in which the entity controlling the service application is located.

Registration may be an implicit service offered at subscription to terminals, in such cases the meta-protocol described in this clause shall not apply.

NOTE: This can be compared with attachment of a telephone set to a PSTN switch or PABX.

The registration meta-protocol is in two stages:

- the registrant registers with the registrar and if successful gains access to the profile of service applications; and
- if successful, the registrar shall supply the credentials for each service application in the form of a ticket to be used by the registrant when requesting service from an application server.

Each service application available to the registrant may be offered by different service providers. These service providers may be in different domains and offer different SpoAs for each of them. The ticket shall indicate that the registrant is authorized to use the service application via the appropriate SpoAs. The ticket shall also identify that registration is valid for a fixed period.

When first performed within a session the registration mode shall be identified by setting the RegistrationMode element (see clause A.5) to "InitialRegistration". Periodic re-registration may occur at any time and shall follow the protocol described in this clause by setting the RegistrationMode element (see clause A.5) to "LocationUpdate".

The terminal has to be registered and authorized before being able to make or receive service, however this "Register before service invocation" may be overridden for certain call types (e.g. emergency calls).

Each registrant has a unique registration identity that shall permit the domain of the home registrar to be identified.

A.1.1 Reachable indication

By successful completion of the registration service the registrant (user) indicates to the registrar (home) that the registrant is reachable and attached.

If the registrant no longer wishes to be reachable for any registered service application in the user profile the registrant shall explicitly detach from that service application by communicating with the SpoA.

If the registrar decides that the user should no longer be reachable for a service application or set of service applications, e.g. if a pre-paid account has been depleted, the registrar shall explicitly initiate the de-registration protocol in order to inform the registrant.

A.1.2 Registration service definition

The registration service shall be offered across a network or set of networks (parts of which may be under different administrative control). The registration service shall be invoked by the use of primitives visible at the TIPHON Registration Service Access Point (TR-SAP) and shown in table 1.

Table 1: Registration primitives visible at TR-SAP

Primitive	Short description	Capability (see note)
TR_RegistrationRequest_req	Allows the user plane to initiate a registration.	
TR_RegistrationRequest_conf	Gives the result of a user initiated registration to the user plane.	
TR_DeRegistrationRequest_req	Allows the user plane to initiate a de-registration.	
TR_DeRegistrationRequest_conf	Gives the result of a user initiated de-registration to the user plane.	
TR_RegistrationStatus_ind	Informs the user plane of any notification from the registrar.	
NOTE: The capability cross references to the capability definition in TS 101 878 [3].		

Table 2: Parameters in registration primitives

Primitive	Parameters		
	Request	Confirm	Indication
TR_RegistrationRequest	UserID, [TerminalID] [List of ServiceApplications]	RegistrationResult	-
TR_DeRegistrationRequest	UserID, [List of ServiceApplications]	DeRegistrationResult	-
TR_RegistrationStatus	-	-	TBD

The parameters in the registration primitives shall take the following values:

RegistrationResult =

Success {list of ServiceApplications};

FailureReason {list of ServiceApplications};

No response from Registrar;

Protocol timer expired;

…

DeRegistrationResult =

Success;

No Active Registrations;

…

ServiceApplication =

Service#1;

Service#2;

…

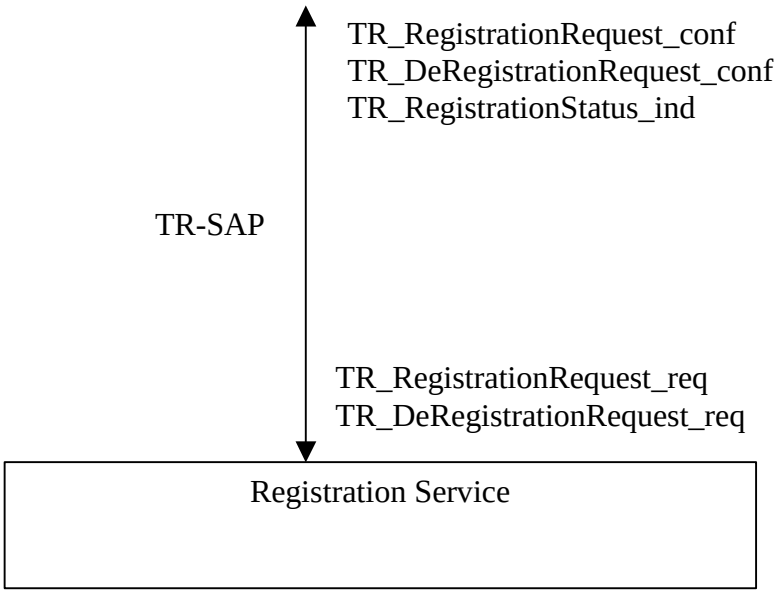


Figure 3: Services offered by the registration service identified by primitives

A.1.3 Ticket content and processing

A service authorization ticket shall:

- identify the user (as registrant);
- identify the registrar (as the issuer of the ticket);
- identify the service application(s);
- identify the service provider for each of the service application(s);
- show the time period during which service application(s) is (are) to be provided to the user; and
- optionally provide a means to cryptographically authenticate the user to the service provider(s).

The user identity in the ticket shall be the identity required by the service application. A single user may have many different identities depending upon the service application(s) chosen. If a user has more than one identity a ticket shall be provided for each identity and this shall contain the list of service applications for the identity.

The ticket shall be defined in ASN.1 as follows:

```
TicketType ::= SEQUENCE
{
    registrantId        VisibleString,
    registrarId         VisibleString,
    serviceCredential   SET OF ServiceCredentialType,
    cryptoDigest        DigestType OPTIONAL
}

ServiceCredentialType ::= SEQUENCE
{
    serviceAppId        ServiceApplicationType,
    spoA                SpoAType, -- shall be in the user's terminal addressing realm
    startTime           GeneralizedTime,
    stopTime            GeneralizedTime, -- Shall be greater than StartTime
    cryptoDigest        DigestType OPTIONAL
}
```

A.1.3.1 Authentication

The registration service may include a strong authentication embedded (or implicit) service. When authentication is offered the registration request shall contain the registrant's credentials in the form of an authentication token.

The following entities should be authenticated:

- terminals with dynamically provisioned parameters;
- functional elements within the network with dynamically provisioned parameters;
- terminals with variable point of transport attachment;
- terminals with variable point of service attachment;
- functional elements within the network with variable point of transport attachment; and
- functional elements within the network with variable point of service attachment.

If strong authentication is not used the resulting authorization, even if successful, shall be deemed to be not authenticated. The service applications to be offered shall be determined, in part, by data in the user profile. On successful authentication the result given to the user shall indicate how to reach these service applications securely (i.e. such that the application server knows that the request comes from an authenticated source).

A.2 Registration for service applications

Successful registration shall allow the registrant access to service applications. A local copy of the service application entitlement (user service profile) may be maintained at the registrant terminal. The capabilities of the terminal to support each of the service applications in the service entitlement shall be maintained locally to the terminal. On single function terminals the service profile may be treated as implicit.

When requesting registration the registrant shall indicate those service applications from the service entitlement that the registrant wishes to receive and that the registrant's terminal is able to support.

A.3 Registrar discovery

The identity and address of the registrar may be provided to the user (or user's terminal) at the time of provisioning, or by using a configuration protocol (e.g. DHCP), by the registrar and maintained within the *service profile* as a populated registration identity element.

The registration identity shall be of the following form:

```
RegIdType ::= SEQUENCE
{
    registrarId         VisibleString,
    registrarLoc        TRL,
    registrantId        VisibleString
}
```

Where the TRL element type shall indicate a TIPHON Resource Location and be of the type below:

```
TRL ::= SEQUENCE
{
    protocolID          VisibleString,
    nameorAddress       VisibleString,
    port                INTEGER OPTIONAL
}
```

A.4 Constituent functional elements

Service applications shall always be executed in the home environment. The home environment may be real or virtual.

Table 3: Registration functional elements

Identity	Name
FE1	Registrant
FE2 (see note 2)	Registrar (see note 1) pointed to by RpoA
FE3 (see note 2)	Application Server pointed to by Service point of Attachment (SpoA)
NOTE 1: Contains a register of currently registered terminals (keyed to Registration identity (regID)). NOTE 2: FE2 and FE3 may be implemented as distributed entities.	

A.4.1 Registrant

The registrant is the logical entity being registered and may take the form of a terminal (e.g. telephone transceiver) or be more directly related to the user (e.g. a human offering credentials).

When requesting registration the registrant shall provide to the registrar a list of service applications for which it is to be registered.

A.4.2 Registrar

The registrar shall maintain the user profile of the registrant.

The registrar shall also maintain a record of SpoAs for each RpoA in order to optimize the binding to local application servers for each registration. The registrar shall validate a registration request against the user profile and identify a service provider local to the registrant's RpoA. If a SpoA is identified locally to the RpoA this SpoA shall act as a home service environment. If no SpoA can be identified locally to the RpoA the SpoA shall be offered from an appropriate but more distant server.

Having identified an appropriate service registrar the registrar shall contact it to request authorization for the registrant. The resulting authorization shall be sent to the registrant in the form of a ticket and shall identify the SpoA.

A.4.3 SpoA

In the context of registration the SpoA shall maintain the attachment of users to service.

A.5 Information flows

The information flows described in table 4 and in the succeeding clauses comprise the registration service M-PDUs. The formal ASN.1 declaration of these M-PDUs is given in clause A.8.

Table 4: Registration M-PDUs

M-PDU name (see notes 1, 2)	Direction	Elements	M/O/C
U_RegistrationRequest	FE1 to FE2	RegId RegistrationMode RpoA ServiceApplicationId AuthenticationToken	M M M M O
D_RegistrationResponse	FE2 to FE1	RegId ServiceCredentials	M M
D_RegistrationReject	FE2 to FE1	RegId ServiceRejectReason	M M
D_RegistrationPending	FE2 to FE1	RegId	M
U_DeRegistrationRequest	FE1 to FE2	RegId	M
D_DeRegistrationResponse	FE2 to FE1	RegId RegistrationRemovedFlag	M M
U_SpoAServiceAttachRequest	FE1 to FE3	RegId ServiceRequestTicket	M M
D_SpoAServiceAttachResponse	FE3 to FE1	RegId ServiceOfferTicket	M M
D_SpoAServiceAttachReject	FE3 to FE1	RegId ServiceRejectReason	M M
U_SpoAServiceDetachRequest	FE1 to FE3	RegId ServiceOfferTicket	M M
D_SpoAServiceDetachResponse	FE3 to FE1	RegId ServiceDetachedFlag	M M
D_SpoAClientAttachNotify	FE2 to FE3	RegistrarID RegistantID ServiceAppId RegistrantAuthTicket	M M M M
U_SpoAClientNotifyResponse	FE3 to FE2	RegistrarID ClientAcceptedFlag SpoAAuthTicket	M M C
D_SpoAClientDetachNotify	FE2 to FE3	RegistrarID RegistantID ServiceAppId RegistrantAuthTicket	M M M M
U_SpoAClientDetachConfirm	FE3 to FE2	RegistrarID ClientDetachedFlag	M M
NOTE 1: M-PDUs prefixed by "U_" indicate M-PDUs generated by a terminal and have direction only towards the network (i.e. Up to the network). NOTE 2: M-PDUs prefixed by "D_" indicate M-PDUs generated by the network in the direction only of the terminal (i.e. Down from the network).			

A.5.1 U_RegistrationRequest

The U_RegistrationRequest message shall be used by a terminal device (FE1) to request the registrar (FE2) to open the user's profile, to identify a SpoA (FE3) for each service being registered against, and to indicate availability.

Direction: FE1 to FE2;

Response to: none;

Response expected: D_RegistrationPending or D_RegistrationResponse or D_RegistrationReject.

The parameters of the U_RegistrationRequest shall be of type M-PDURegistrationRequestType, which is formally declared in ASN.1 as below.

```
M-PDURegistrationRequestType ::= SEQUENCE
{
    regID               RegIdType,
    registrationMode    RegistrationModeType DEFAULT initialRegistration,
    rpoA                RpoAType,
    serviceAppId        SET OF ServiceApplicationType,
    authToken           AuthenticationTokenType OPTIONAL
}

AuthenticationTokenType ::= TokenType

RegistrationModeType ::= ENUMERATED
{
    initialRegistration (0),
    locationUpdate (1)
}
```

A.5.2 D_RegistrationConfirm

The D_RegistrationConfirm message shall be used by FE2 to explicitly accept a registration request from FE1.

Direction: FE2 to FE1;

Response to: U_RegistrationRequest;

Response expected: none.

The parameters of the D_RegistrationConfirm shall be of type M-PDURegistrationResponseType, which is formally declared in ASN.1 as below.

```
M-PDURegistrationResponseType ::= SEQUENCE
{
    regID               RegIdType,
    serviceCredential   SET OF TicketType
}
```

A.5.3 D_RegistrationReject

The D_RegistationReject message shall be used by FE2 to explicitly reject a registration request from FE1.

Direction: FE2 to FE1;

Response to: U_RegistrationRequest;

Response expected: none.

The parameters of the D_RegistrationReject shall be of type M-PDURegistrationRejectType, which is formally declared in ASN.1 as below.

```
M-PDURegistrationRejectType ::= SEQUENCE
{
    regID               RegIdType,
    serviceRejectReason SET OF ServiceRejectReasonType
```

```
}

ServiceRejectReasonType ::= SEQUENCE
{
    serviceAppId        ServiceApplicationType, -- Enumerated list defined by operator
    rejectReason        TIPHONErrorType     -- Enumerated list defined by the present document
}

TIPHONErrorType ::= SEQUENCE
{
    reason              ENUMERATED {badThingWrong (0)},
    diagnostic          ENUMERATED {somethingWrong(0)} OPTIONAL,
    freeText            VisibleString OPTIONAL -- This should inform the user behaviour
}
```

A.5.4 D_RegistrationPending

The D_RegistrationPending message shall be used by FE2 to indicate that the registration process is continuing.

Direction: FE2 to FE1;

Response to: U_RegistrationRequest;

Response expected: none.

The parameters of the D_RegistrationPending shall be of type M-PDURegistrationPendingType, which is formally declared in ASN.1 as below.

```
M-PDURegistrationPendingType ::= SEQUENCE
{
    regID           RegIdType
}
```

A.5.5 U_DeRegistrationRequest

The U_DeRegistrationRequest message shall be used by FE1 to request clearance of all service attachments.

Direction: FE1 to FE2;

Response to: none;

Response expected: D_DeRegistrationResponse.

The parameters of the U_DeRegistrationRequest shall be of type M-PDUDeRegistrationRequestType, which is formally declared in ASN.1 as below.

```
M-PDUDeRegistrationRequestType ::= SEQUENCE
{
    regID           RegIdType
}
```

A.5.6 D_DeRegistrationResponse

The D_DeRegistationResponse message shall be used by FE2 to explicitly indicate acceptance of the deregistration and to confirm that all service attachments maintained by FE2 have been cleared.

Direction: FE2 to FE1;

Response to: U_DeRegistrationRequest;

Response expected: none.

The parameters of the D_DeRegistrationResponse shall be of type M-PDUDeRegistrationResponseType, which is formally declared in ASN.1 as below.

```
M-PDUDeRegistrationResponseType ::= SEQUENCE
{
    regID                   RegIdType,
    registrationRemovedFlag BOOLEAN
}
```

A.5.7 U_SpoAServiceAttachRequest

The U_SpoAServiceAttachRequest message shall be used by FE1 to request attachment to FE3 (SpoA) for a specific service offering credentials received from FE2 in the D_RegistrationResponse.

Direction: FE1 to FE3;

Response to: D_RegistrationResponse (from FE2);

Response expected: D_SpoAServiceAttachResponse.

The parameters of the U_SpoAServiceAttachRequest shall be of type M-PDUSpoAServiceAttachRequestType, which is formally declared in ASN.1 as below.

```
M-PDUSpoAServiceAttachRequestType ::= SEQUENCE
{
    regID                   RegIdType,
    serviceRequestTicket    TicketType
}
```

A.5.8 D_SpoAServiceAttachResponse

The D_SpoAServiceAttachResponse message shall be used by FE3 to indicate acceptance of the service request.

Direction: FE3 to FE1;

Response to: U_SpoAServiceAttachRequest;

Response expected: none.

The parameters of the D_SpoAServiceAttachResponse shall be of type M-PDUSpoAServiceAttachResponseType, which is formally declared in ASN.1 as below. Note that the SpoA for each service requested is to be found in the returned `TicketType`.

```
M-PDUSpoAServiceAttachResponseType ::= SEQUENCE
{
    regID               RegIdType,
    serviceOfferTicket  TicketType
}
```

A.5.9 D_SpoAServiceAttachReject

The D_SpoAServiceAttachReject message shall be used by FE3 to explicitly reject a service attach request.

Direction: FE3 to FE1;

Response to: U_SpoAServiceAttachRequest;

Response expected: none.

The parameters of the D_SpoAServiceAttachReject shall be of type M-PDUSpoAServiceAttachRejectType, which is formally declared in ASN.1 as below.

```
M-PDUSpoAServiceAttachRejectType ::= SEQUENCE
{
    regID               RegIdType,
    serviceRejectReason ServiceRejectReasonType -- Enumerated list defined by the present document
}
```

A.5.10 U_SpoAServiceDetachRequest

The U_SpoAServiceDetachRequest message shall be used by FE1 to explicitly detach from a SpoA.

Direction: FE1 to FE3;

Response to: none;

Response expected: D_SpoAServiceDetachResponse.

The parameters of the U_SpoAServiceDetachRequest shall be of type M-PDUSpoAServiceDetachRequestType, which is formally declared in ASN.1 as below.

```
M-PDUSpoAServiceDetachRequestType ::= SEQUENCE
{
    regID               RegIdType,
    serviceOfferTicket  TicketType
}
```

A.5.11 D_SpoAServiceDetachResponse

The D_SpoAServiceDetachResponse message shall be used by FE3 to indicate acceptance of a service detach request.

Direction: FE3 to FE1;

Response to: U_SpoAServiceAttachRequest;

Response expected: none.

The parameters of the D_SpoAServiceDetachResponse shall be of type M-PDUSpoAServiceDetachResponseType, which is formally declared in ASN.1 as below.

```
M-PDUSpoAServiceDetachResponseType ::= SEQUENCE
{
    regID               RegIdType,
    serviceDetachedFlag BOOLEAN
}
```

A.5.12 D_SpoAClientAttachNotify

The D_SpoAClientAttachNotify message shall be used by FE2 to inform FE3 that a client is intending to attach for service and to indicate that this attachment is approved by FE3.

Direction: FE2 to FE3;

Response to: U_RegistrationRequest (from FE1);

Response expected: U_SpoAClientNotifyResponse.

The parameters of the D_SpoAClientAttachNotify shall be of type M-PDUSpoAClientAttachNotifyType, which is formally declared in ASN.1 as below.

```
M-PDUSpoAClientAttachNotifyType ::= SEQUENCE
{
    registrarID             RegIdType,
    registantID             RegIdType,
    serviceAppId            ServiceApplicationType,
    registrantAuthTicket    TicketType
}
```

A.5.13 U_SpoAClientNotifyResponse

The U_SpoAClientNotifyResponse message shall be used by FE3 to explicitly accept or reject a request by FE2 to offer services to a client (FE1).

Direction: FE3 to FE2;

Response to: D_SpoAClientAttachNotify;

Response expected: none.

The parameters of the U_SpoAClientNotifyResponse shall be of type M-PDUSpoAClientNotifyResponseType, which is formally declared in ASN.1 as below.

```
M-PDUSpoAClientNotifyResponseType ::= SEQUENCE
{
    registrarID         RegIdType,
    clientAcceptedFlag  BOOLEAN,
    spoAAuthTicket      TicketType OPTIONAL -- Conditional on value of flag
}
```

A.5.14 D_SpoAClientDetachNotify

The D_SpoAClientDetachNotify message shall be used by FE2 to inform FE3 that a client has requested deregistration from all services through the registrar and that this is approved requiring the SpoA to remove authorization for the client to use the services of the SpoA.

Direction: FE2 to FE3;

Response to: U_DeRegistrationRequest (from FE1);

Response expected: U_SpoAClientDetachResponse.

The parameters of the D_SpoAClientDetachNotify shall be of type M-PDUSpoAClientDetachNotifyType, which is formally declared in ASN.1 as below.

```
M-PDUSpoAClientDetachNotifyType ::= SEQUENCE
{
    registrarID             RegIdType,
    registantID             RegIdType,
    serviceAppId            ServiceApplicationType,
    registrantAuthTicket    TicketType
}
```

A.5.15 U_SpoAClientDetachResponse

The U_SpoAClientDetachResponse message shall be used by FE3 to indicate that service has been withdrawn from the client (FE1).

Direction: FE3 to FE2;

Response to: D_SpoAClientDetachNotify;

Response expected: none.

The parameters of the U_SpoAClientDetachResponse shall be of type M-PDUSpoAClientDetachResponseType, which is formally declared in ASN.1 as below.

```
M-PDUSpoAClientDetachResponseType ::= SEQUENCE
{
    registrarID             RegIdType,
    clientDetachedFlag      BOOLEAN
}
```

A.6 Behavioural description

A.6.1 Registration

A.6.1.1 Registrant

Registration shall be invoked by the registrant by sending a U_RegistrationRequest M-PDU to the registrar indicating for which service applications the request shall apply. At the same time, the registrant shall start the timer TR001. Invocation may be the result of one of the following conditions:

- a request from the registering user;
- the end of a predefined period to ensure that the registration is regularly refreshed;
- a change of TpoA.

The mode of registration is identified by appropriate setting of the RegistrationMode element of the U_RegistrationRequest M-PDU.

If timer TR001 expires FE1 shall inform the user application of the failure of the registration with the TR_RegistrationRequest_conf primitive with failure reason set to "No response from Registrar".

FE1 shall take reasonable steps to combine requests from user applications in order to operate the network in an efficient manner. If a user application makes a request at a time after other registrations have already been completed FE1 shall re-register for all the services now requested by user applications so that new calls are requested using data that can be held in a single ticket.

On receipt of the D_RegistrationPending M-PDU from FE2, FE1 shall reset and restart timer TR001.

On receipt of the D_RegistrationResponse M-PDU from FE2, FE1 shall prepare for attachment to each service it has received credentials for. For each service FE1 shall prepare an U_SpoAServiceAttachRequest M-PDU based upon the content of the ServiceCredentials. FE1 shall cancel Timer TR001.

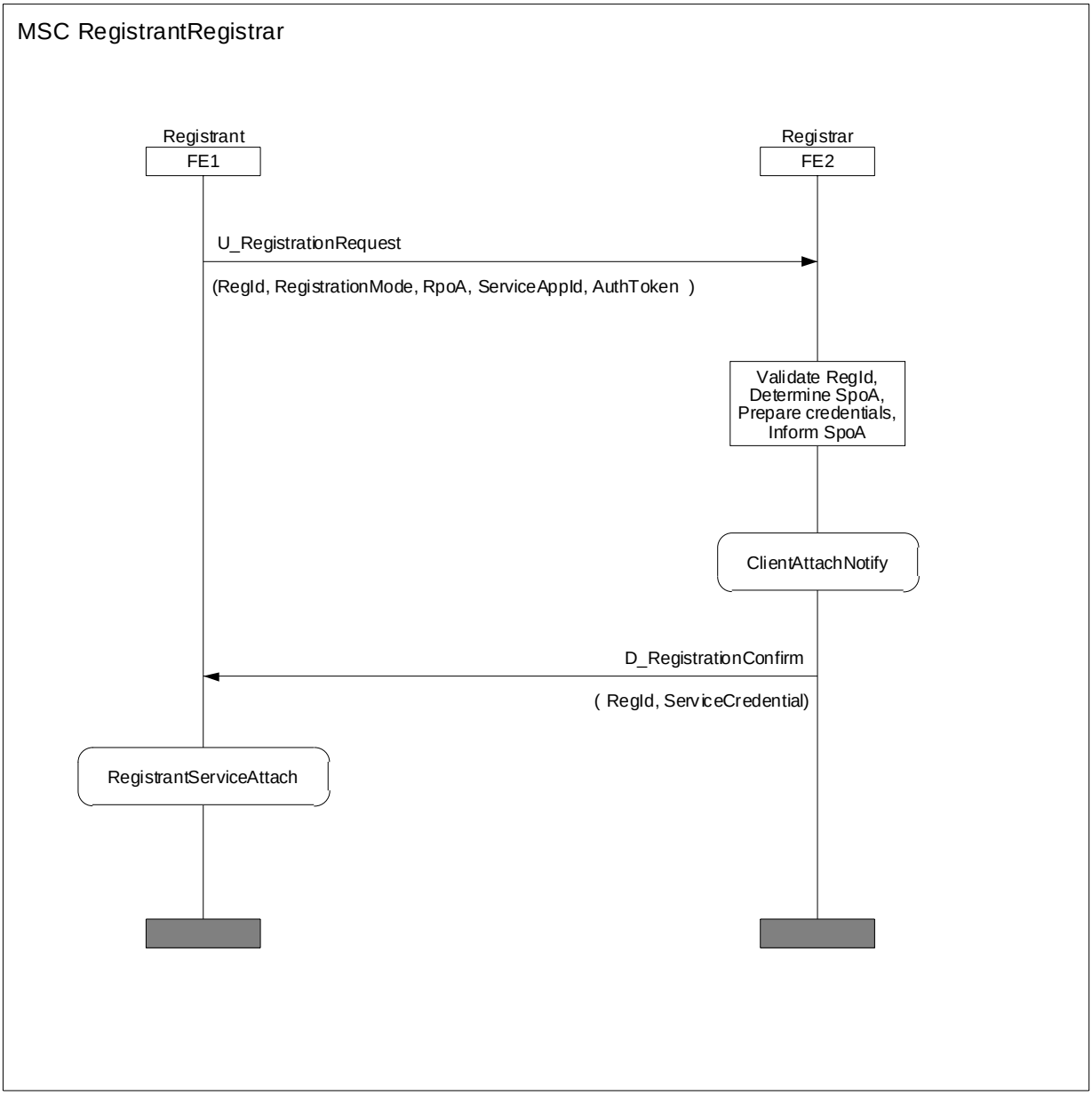


Figure 4: Registration meta-protocol MSC (registrant to registrar)

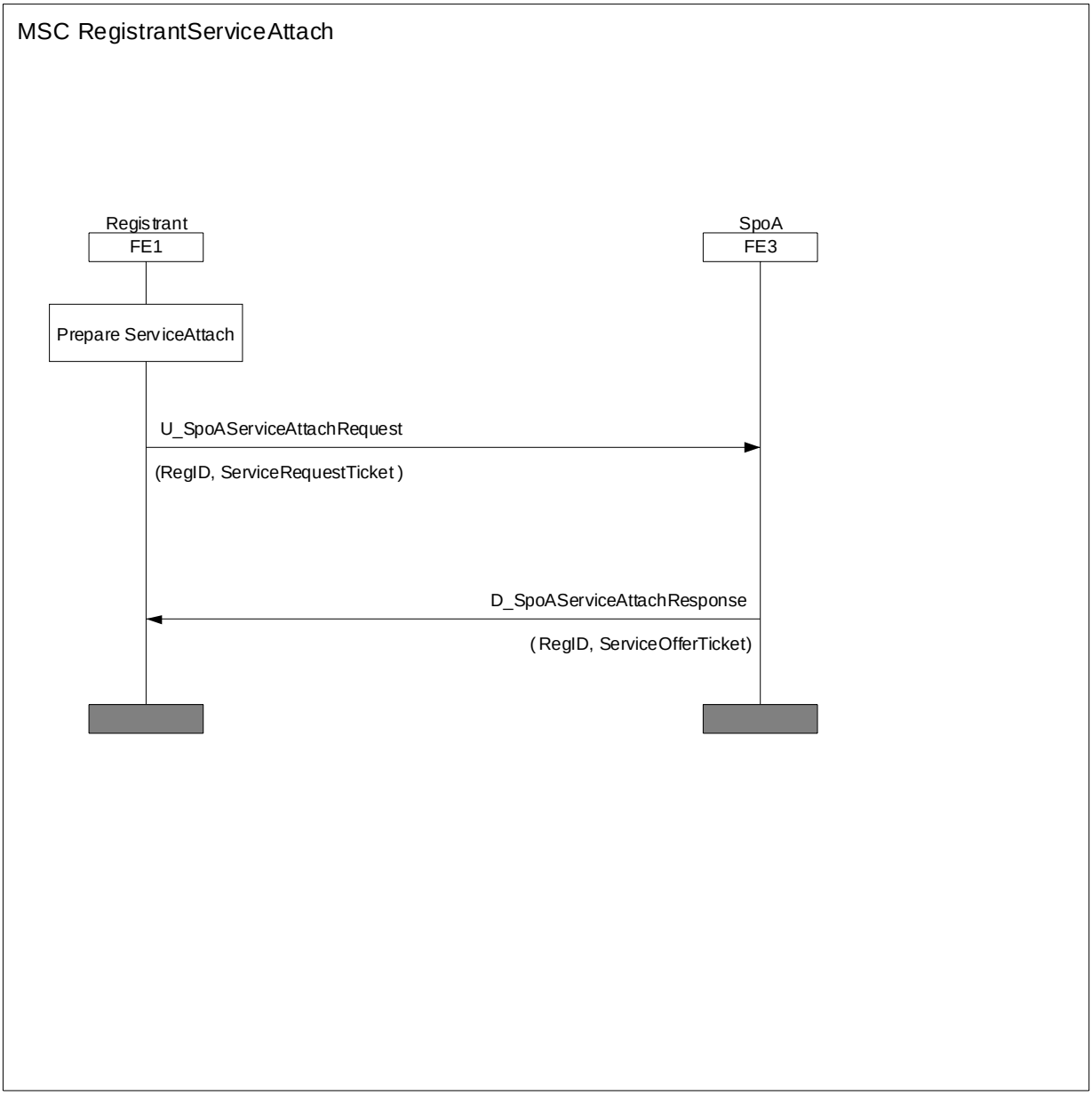


Figure 5: Registration meta-protocol MSC (registrant to SpoA)

A.6.1.2 Registrar

On receipt of U_RegistrationRequest from FE1 the registrar (FE2) shall perform the following tasks:

Verification of FE1 (Registrant)

- The offered RegId shall be checked to ensure that it appears in the locally maintained list of User-Profiles. If no User-Profile exists for the offered RegId then the registrar shall reject the registration attempt by sending D_RegistrationReject to FE1 with a single ServiceRejectReason element set to "Identity not known", and the ServiceApplicationId element set to NULL. If the profile is found it shall be recovered and the next task shall be performed.

For each service requested FE2 (the registrar) shall perform the following tasks:

Verification of requested service

- FE2 shall determine if the requested service exists in the User-Profile. If the requested service does not exist in the User-Profile and if explicit authorization is required FE2 shall reject the registration attempt with the reject reason element set to "Invalid or Unknown service" in D_RegistrationReject. If the service is found the next task shall be performed.

Identification of SpoA

- FE2 shall attempt to find a SpoA at a location appropriate to the offered TpoA. This requires FE2 to maintain a record of the TpoA of each SpoA for each service. On identifying a SpoA FE2 shall perform the next task.

Notify SpoA of Intent to Attach

- FE2 shall request FE3 (the SpoA) to offer service to FE1 by sending a D_SpoAClientAttachNotify M-PDU to FE3 containing data indicating the Registrant, the Registrar, the Service, and a ticket to allow validation of the request.

Wait for FE3 to respond

On receipt of U_SpoAClientNotifyResponse FE2 shall perform the following tasks:

On failure

- If a fail response is received from FE3 and alternatives are available FE2 shall send a D_SpoAClientAttachNotify M-PDU to the new FE3 containing data indicating the Registrant, the Registrar, the Service, and a ticket to allow validation of the request.

On success

- When successful and final failures have been received for all SpoAs the registrar shall compose the service credentials (ticket) to be sent to the registrant and send a D_RegistrationResponse M-PDU.

NOTE: The number of FE3 retries should be informed by the value of Timer TR001 established by FE2 for its FE1s.

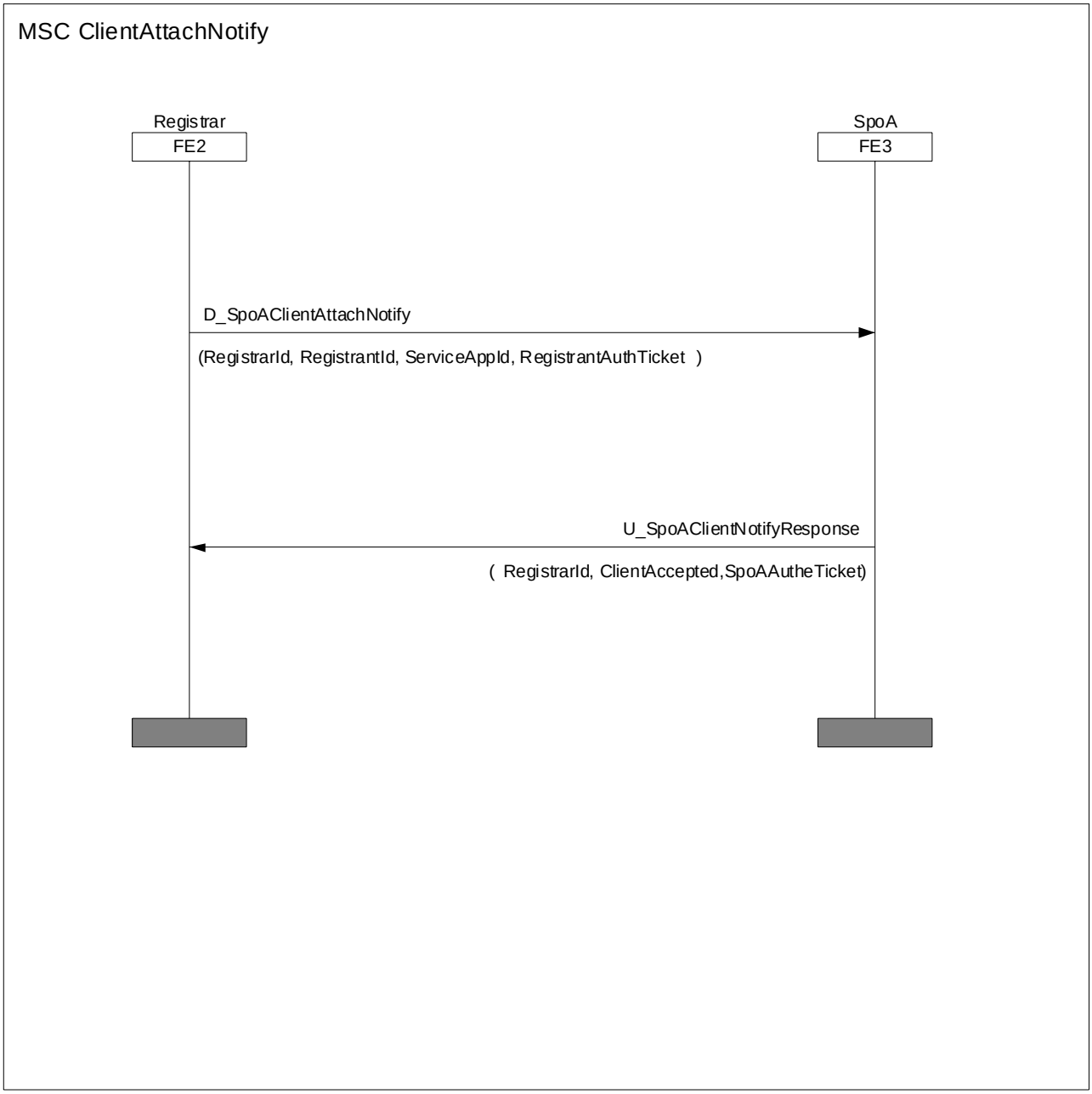


Figure 6: MSC of registrar to SpoA protocol registration

A.6.1.3 SpoA

On receipt of D_SpoAClientAttachNotify from FE2 (registrar) FE3 (the SpoA) shall verify that it has sufficient resource to fulfil any requests from the client against the service invoked. If sufficient resource is available FE3 shall return U_SpoAClientNotifyConfirm to FE2 with an authorization ticket valid for the service.

On receipt of U_SpoAServiceAttachRequest M-PDU from the registrant SpoA shall validate the ticket and if correct shall offer service, confirming this with the D_SpoAServiceAttachConfirm M-PDU. If the validation fails SpoA shall notify the client using the D_SpoAServiceAttachReject M-PDU.

A.6.2 Deregistration

Deregistration may take two forms:

1) Centralized deregistration from all services through registrar.

2) Localized deregistration from a single service through SpoA.

The forms are described in this and succeeding clauses.

A.6.2.1 Registrant

To initiate deregistration the user application shall transfer a TR_DeRegistrationRequest_req primitive across TR_SAP to the registration entity in the terminal (FE1). The TR_DeRegistrationRequest_req shall indicate the service(s) to be deregistered.

If FE1 has no active service attachments it shall respond to the user application with the TR_DeRegistrationRequest_conf primitive indicating error of type "No-Active-Service-Attachments".

If all services are to be deregistered FE1 shall send U_DeRegistrationRequest M-PDU to the registrar entity (FE2), and start timer TR002. On receipt of D_DeRegistrationResponse FE1 shall clear and stop timer TR002.

If one or more, but not all, services are to be deregistered FE1 shall send U_SpoAServiceDetachRequest M-PDU to the appropriate SpoA (FE3), and start timer TR002. If more than one SpoA is involved FE1 shall send U_SpoAServiceDetachRequest M-PDUs as required by the service to be deregistered. On receipt of D_SpoAServiceDetachConfirm from FE3 FE1 shall clear and stop timer TR002.

A.6.2.2 Registrar

On receipt of U_DeRegistrationRequest from FE1 FE2 shall send a D_SpoAClientDetachNotify M-PDU to each FE3 for which a ticket was issued.

On receipt of all U_SpoAClientDetachConfirm M-PDUs from each FE3 FE2 shall send D_DeRegistrationConfirm back to FE1.

A.6.2.3 SpoA

On receipt of U_SpoAServiceDetachRequest FE3 shall validate the content of the received ServiceOfferTicket against the offered regID. If valid FE3 shall delete all resources reserved against the received regID and return D_SpoAServiceDetachResponse M-PDU to FE1 with ServiceDetached element set to TRUE.

On receipt of D_SpoAClientDetachNotify from FE2 FE3 shall validate the content of the RegistrantAuthTicket against the triplet of RegistrantId, RegistrarId and ServiceAppId. If valid FE3 shall delete all resources reserved against the received RegistrantId and return U_SpoAClientDetachResponse M-PDU with element ClientDetached set to TRUE.

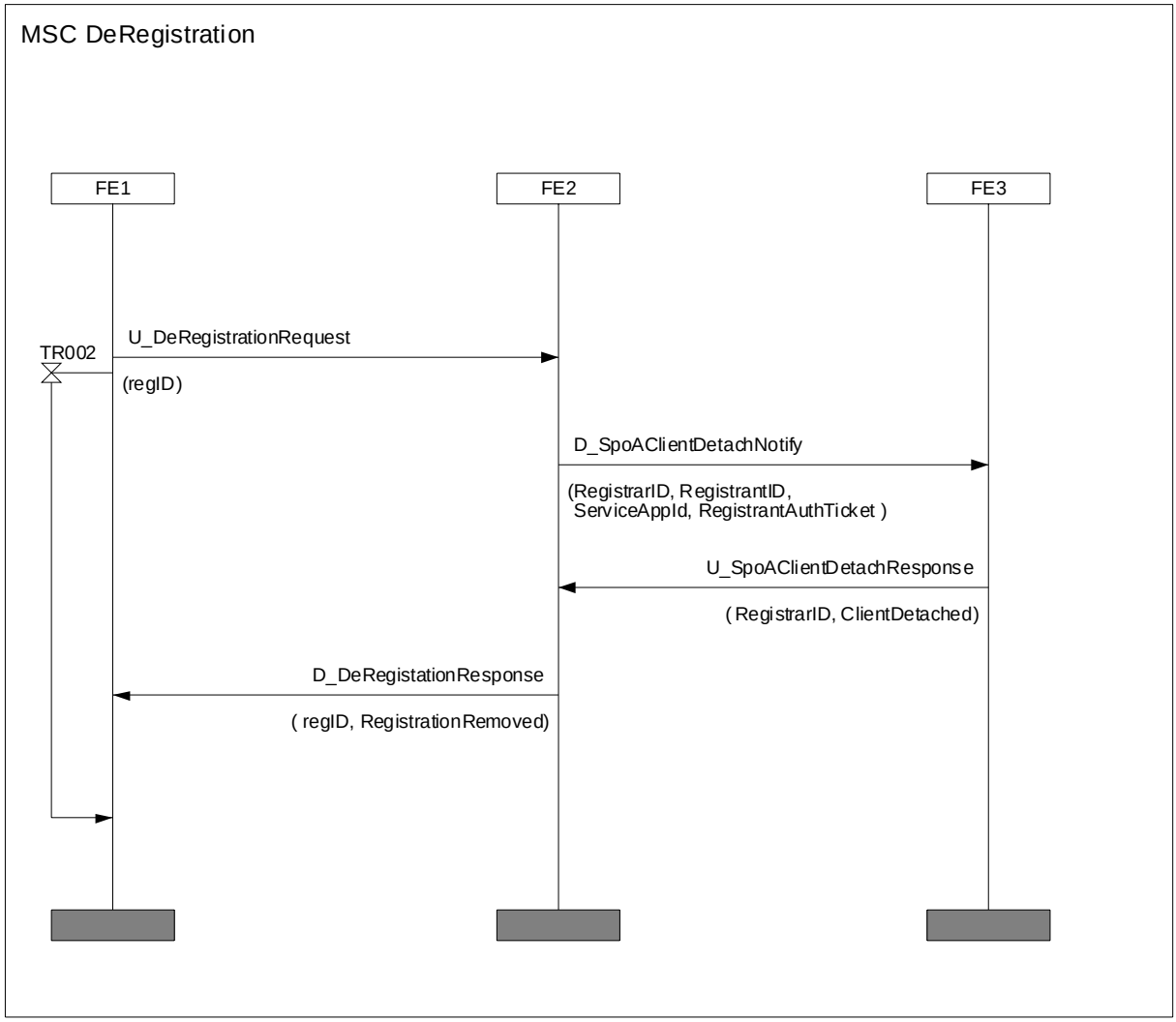


Figure 7: User initiated deregistration of all services

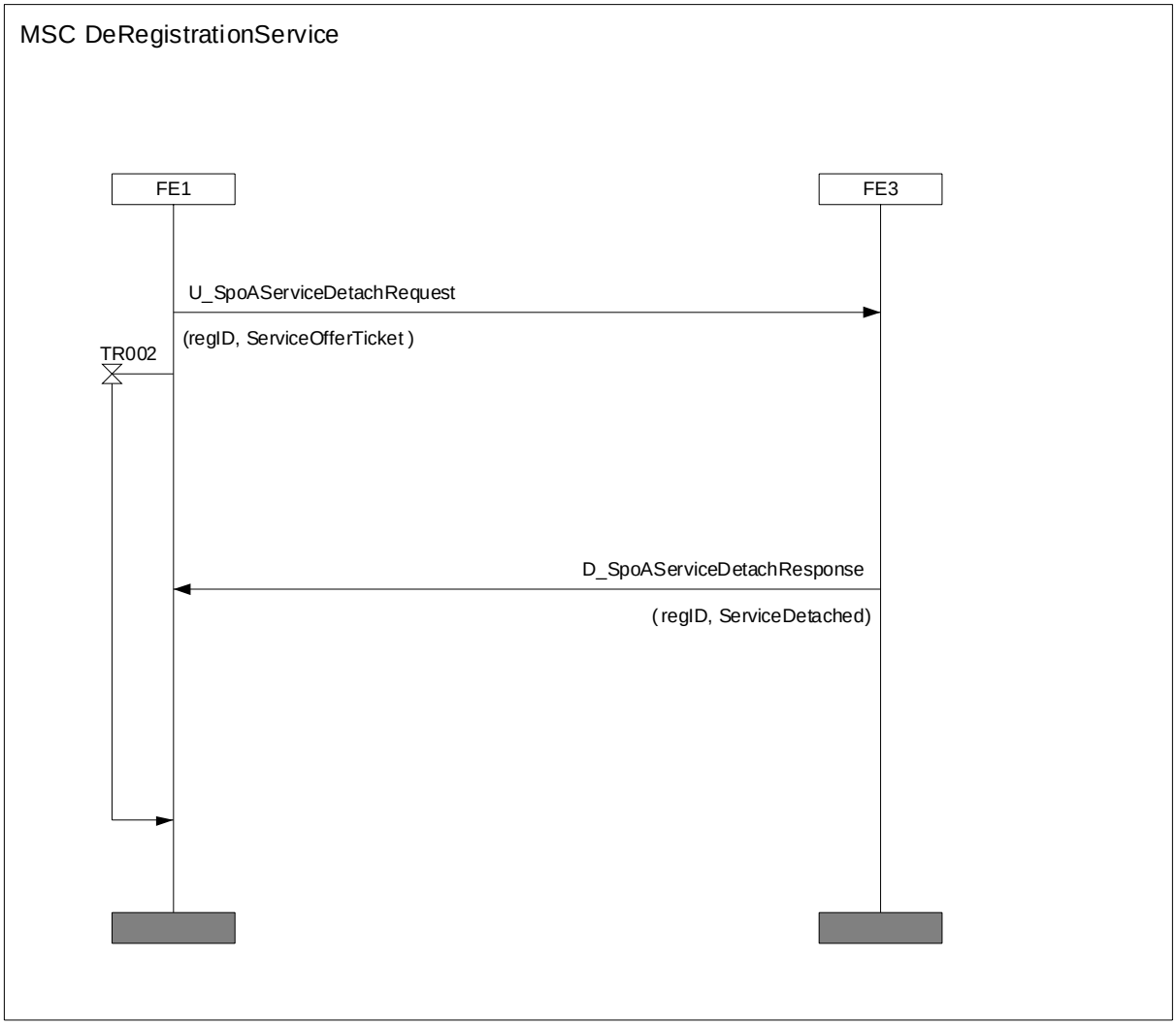


Figure 8: User initiated deregistration of individual service

A.7 Timers

A.7.1 TR001, Registration timer

The default value of this timer shall be 30 s.

A.7.2 TR002, Deregistration timer

The default value of this timer shall be 30 s.

A.8 Data definitions (ASN.1)

This clause contains the data definitions for information flows visible at reference point R.

NOTE: When checked using the OSS Syntax checker this ASN.1 contains no errors but warns that some M-PDUs encode to the same form of structure (an example being U_SpoAServiceAttachRequest and D_SpoAServiceAttachResponse which each contain a sequence of RegIDType and a TicketType). This is not treated in this clause as an error but as information.

```
TIPHONr3-RefPtR DEFINITIONS ::=

BEGIN

-- Start of M-PDU definitions

M-PDURegistrationPendingType ::= SEQUENCE
{
    regID           RegIdType
}

M-PDUDeRegistrationRequestType ::= SEQUENCE
{
    regID           RegIdType
}

M-PDUDeRegistrationResponseType ::= SEQUENCE
{
    regID                   RegIdType,
    registrationRemovedFlag BOOLEAN
}

M-PDURegistrationRequestType ::= SEQUENCE
{
    regID               RegIdType,
    registrationMode    RegistrationModeType DEFAULT initialRegistration,
    rpoA                RpoAType,
    serviceAppId        SET OF ServiceApplicationType,
    authToken           AuthenticationTokenType OPTIONAL
}

M-PDURegistrationResponseType ::= SEQUENCE
{
    regID               RegIdType,
    serviceCredential   SET OF TicketType
}

M-PDURegistrationRejectType ::= SEQUENCE
{
    regID               RegIdType,
    serviceRejectReason SET OF ServiceRejectReasonType
}

M-PDUSpoAServiceAttachRequestType ::= SEQUENCE
{
    regID                   RegIdType,
    serviceRequestTicket    TicketType
}

M-PDUSpoAServiceAttachResponseType ::= SEQUENCE
{
    regID               RegIdType,
    serviceOfferTicket  TicketType
}

M-PDUSpoAServiceAttachRejectType ::= SEQUENCE
{
    regID               RegIdType,
    serviceRejectReason ServiceRejectReasonType -- Enumerated list defined by the present document
}

M-PDUSpoAServiceDetachRequestType ::= SEQUENCE
{
    regID               RegIdType,
    serviceOfferTicket  TicketType
```

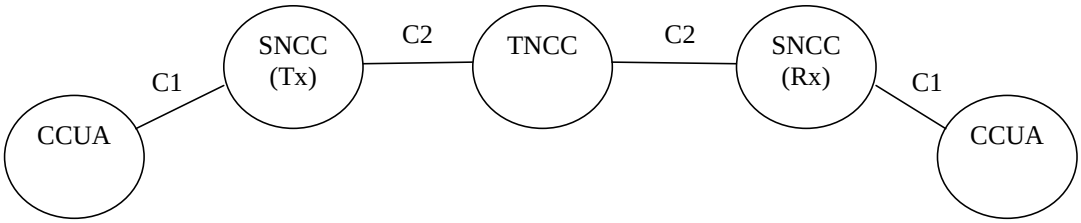
```
}

M-PDUSpoAServiceDetachResponseType ::= SEQUENCE
{
    regID               RegIdType,
    serviceDetachedFlag BOOLEAN
}

M-PDUSpoAClientAttachNotifyType ::= SEQUENCE
{
    registrarID             RegIdType,
    registantID             RegIdType,
    serviceAppId            ServiceApplicationType,
    registrantAuthTicket    TicketType
}

M-PDUSpoAClientNotifyResponseType ::= SEQUENCE
{
    registrarID         RegIdType,
    clientAcceptedFlag  BOOLEAN,
    spoAAuthTicket      TicketType OPTIONAL -- Conditional on value of flag
}

M-PDUSpoAClientDetachNotifyType ::= SEQUENCE
{
    registrarID             RegIdType,
    registantID             RegIdType,
    serviceAppId            ServiceApplicationType,
    registrantAuthTicket    TicketType
}

M-PDUSpoAClientDetachResponseType ::= SEQUENCE
{
    registrarID         RegIdType,
    clientDetachedFlag  BOOLEAN
}

-- End of M-PDU definitions

-- Start of element definitions

TicketType ::= SEQUENCE
{
    registrantId        VisibleString,
    registrarId         VisibleString,
    serviceCredential   SET OF ServiceCredentialType,
    cryptoDigest        DigestType OPTIONAL
}

ServiceCredentialType ::= SEQUENCE
{
    serviceAppId    ServiceApplicationType,
    spoA            SpoAType, -- shall be in the user's terminal addressing realm
    startTime       GeneralizedTime,
    stopTime        GeneralizedTime, -- Shall be greater than StartTime
    cryptoDigest    DigestType OPTIONAL
}

RegIdType ::= SEQUENCE
{
    registrarId     VisibleString,
    registrarLoc    TRL,
    registrantId    VisibleString
}

TRL ::= SEQUENCE
{
    protocolID          VisibleString,
    nameorAddress       VisibleString,
    port                INTEGER OPTIONAL
}

AuthenticationTokenType ::= TokenType

RegistrationModeType ::= ENUMERATED
{
    initialRegistration (0),
    locationUpdate (1)
```

```
}

ServiceRejectReasonType ::= SEQUENCE
{
    serviceAppId        ServiceApplicationType, -- Enumerated list defined by operator
    rejectReason        TIPHONErrorType     -- Enumerated list defined by the present document
}

TIPHONErrorType ::= SEQUENCE
{
    source      ENUMERATED
                {
                    callControl,
                    bearerControl,
                    mediaControl,
                    transportControl,
                    …
                },
    severity    ENUMERATED
                {
                    fatalError,
                    warning,
                    information
                },
    reason      ENUMERATED
                {
                    invalid,
                    not_Supported,
                    unavailable,
                    does_not_exist,
                    insufficient_resources,
                    …
                },
    diagnostic  TIPHONErrorDiagnosticType OPTIONAL,
    freeText    VisibleString OPTIONAL,
    embeddedError   TIPHONErrorType OPTIONAL
}

SpoAType ::= VisibleString

ServiceApplicationType ::= VisibleString

DigestType ::= VisibleString

TokenType ::= VisibleString

RpoAType ::= VisibleString

-- End of element definitions

END
```

Annex B (normative): Meta-protocol at reference point C

B.1 Overview

Reference point C as defined in TS 101 314 [1] lies between peer call control entities and between peer bearer control entities. The meta-protocols visible at this reference point are therefore associated with call control and bearer control only.



CCUA = Call Control User Agent
SNCC = Serving Network Call Control
TNCC = Transit Network Call Control

Figure 9: Reference model for Call Control

Call control in TIPHON is a predicated protocol in which the bearer capabilities to support a call in each functional group in the end to end path have to be established such that a known end-to-end bearer service is agreed between the communicating parties before committing to call setup.

NOTE: The end-to-end bearer service may be realized over across multiple technologies. This infers the presence of Media Control (see annex C) or inter connect functional entities between the media legs.

Prior to the invocation of any protocol at reference point C the identity and location of the relevant service provider may be provided to the originating terminal either through the registration meta-protocol (and thus be contained in the service ticket received (see annex A)), or by pre-arrangement.

The call control service is a peering service offered across a network. The call control service shall be invoked in the terminal by the use of primitives visible at the TIPHON Call Control Service Access Point (TCC-SAP) as shown in table 5.

Table 5: Call control primitives visible at TCC-SAP

Primitive	Short description	Capability (see note)
TCC_CallSetup_req	Used to invoke a call setup	Basic Call Control (see [3], clause 8.1), Carrier Selection (see [3], clause 8.11).
TCC_CallSetup_conf	Used by the CC service to inform the user plane of the success of a call setup attempt	Basic Call Control (see [3], clause 8.1).
TCC_CallSetup_ind	Used by the CC service to inform the user plane of an incoming call setup attempt	Basic Call Control (see [3], clause 8.1).
TCC_CallSetup_resp	Used by the user plane to direct the CC service how to deal with the incoming call setup attempt	Basic Call Control (see [3], clause 8.1).
TCC_CallClear_req/conf	Used by the user plane to invoke a call cleardown	Basic Call Control (see [3], clause 8.1).
TCC_CallClear_ind/resp	Used to indicate to the user plane that the other party has invoked a call cleardown	Basic Call Control (see [3], clause 8.1).
TCC_CallModify_req/conf	Used by the user plane to modify data relating to call in progress, or to add data to a signalling session in progress	Overlap sending of called party identifier information (see [3], clause 8.24).
TCC_CallModify_ind/resp	Used to indicate to the user plane that the call in progress is to be modified, or to request data from the user plane to a signalling session in progress	Overlap sending of called party identifier information (see [3], clause 8.24).
TCC_SetLocalProfile_req/conf	Allows the user plane to set parameters for the **local** call processing. This includes the ability to set Anonymous Call rejection behaviour, and to set calling party identification restrictions	Anonymous Call Rejection (see [3], clause 8.7), Calling User Identity Generation (see [3], clause 8.4).
TCC_CallReport_ind	Used to indicate to the user plane that reports relating to the processing of the call have been received. These may be used to trigger tones in a user's headset (e.g. ringing tone)	Basic Call Control (see [3], clause 8.1).
NOTE: The capability cross references to the capability definition in TS 101 878 [3].		

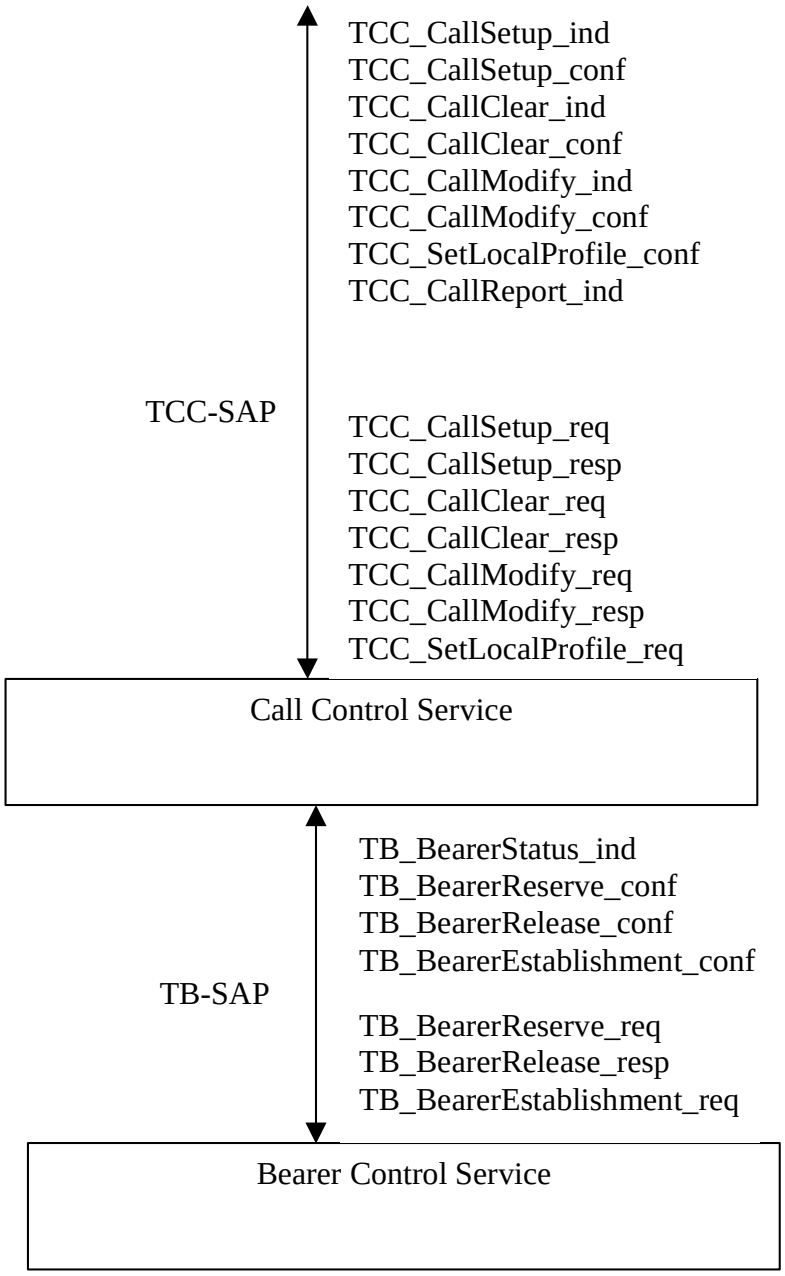
To support the function of bearer control and establishment the primitives shown in table 6 are visible at the TIPHON Bearer Service Access Point (TB-SAP).

NOTE 1: TB-SAP is not visible from the application plane in this edition of the present document but accessed only by the call control entities.

NOTE 2: TB-SAP is not mapped in TS 101 314 [1].

Table 6: Bearer control primitives visible at TB-SAP

Primitive	Short description	Capability (see note)
TB_BearerReserve_req	Allows the upper layer to reserve bearer capability	
TB_BearerReserve_conf	Indicates to the upper layer the result of a bearer reservation request	
TB_BearerRelease_req	Allows the upper layer to delete a bearer reservation	
TB_BearerRelease_conf	Indicates to the upper layer the result of a bearer release request	
TB_BearerStatus_ind	Indicates to the upper layer the current status of the bearer	
TB_BearerEstablishment_req	Used by the upper layer applications to request establishment of previously reserved bearer resources	
TB_BearerEstablishment_conf	Confirms the establishment	
NOTE: The capability cross references to the capability definition in TS 101 878 [3].		



NOTE 1: TCC-SAP encompasses the SC1 reference point defined in [1].
NOTE 2: TB-SAP is not defined in [1].

Figure 10: Primitives offered to enable call and bearer control services

The call control meta-protocol exists at reference point C1 (see [1]) for interactions between the terminal and the network resident call control, and at reference point C2 for interactions between network resident call control entities.

The bearer control meta-protocol exists at reference point C1 (see [1]) for interactions between the terminal and the network resident bearer control, and at reference point C2 for interactions between network resident bearer control entities.

B.2 Constituent functional elements

The Call Control functional elements shall maintain the call state and use the supporting meta-protocol to setup and release calls as well as allocate and release resources for the media phase of the call using the Bearer Control functions.

Table 7: Call Control functional elements

Identity	Name
CCUA-O	Call control agent at originating terminal
SNCC-O	Call control agent at SpoA of originating terminal
TNCC	Call control agent in an intermediate network
SNCC-T	Call control agent at SpoA of terminating terminal
CCUA-T	Call control agent at terminating terminal

The Bearer Control functional elements shall maintain the bearer state and use the supporting meta-protocol service to setup and release bearers.

Table 8: Bearer Control functional elements

Identity	Name
BCUA-O	Bearer control agent at originating terminal
SNBC-O	Bearer control agent at SpoA of originating terminal
TNBC	Bearer control agent in an intermediate network
SNBC-T	Bearer control agent at SpoA of terminating terminal
BCUA-T	Bearer control agent at terminating terminal

B.3 Information flows

The information flows described in table 9 comprise the call control M-PDUs. A formal ASN.1 declaration of these M-PDUs is given. The Call Control M-PDUs exist at reference points C1 [1] (from endpoint to network) and C2 [1] (between networks, which may be under different administrative control).

Table 9: Call Control M-PDUs

M-PDU name (see notes 1 and 2)	Parameters	M/O/C
U_CallRequest	CallID	M
	CallingPartyRestriction	M
	CallingPartyID	C
	CalledPartyID	M
	CallPriority	M
	operatorSelection (see note 3)	O
	ServiceOfferTicket	O
D_CallReject	CallId	M
	CallRejectReason	M
D_CallReport	CallId	M
	ReportReason	M
	ReportParameters	C
D_CallConnect	CallId	M
U_CCAdditionalDigits	CallID	M
	AdditionalDigits	M
D_CallRequest	CallID	M
	CallingPartyID	C
	CallingPartyRestriction	M
	CalledPartyID	M
	CallPriority	M
U_CallAlert	CallId	M
U_CallConnect	CallId	M

M-PDU name (see notes 1 and 2)	Parameters	M/O/C
NW_CallRequest	CallID	M
	CallingPartyID	C
	CallingPartyRestriction	M
	CalledPartyID	M
	CallPriority	M
NW_CallReport	CallId	M
	ReportReason	M
	ReportParameters	C
NW_CallConnect	CallId	M
NW_CallReject	CallId	M
	CallRejectReason	M
NOTE 1: M-PDUs prefixed by "U_" indicate M-PDUs generated by a terminal and have direction only towards the network (i.e. Up to the network). NOTE 2: M-PDUs prefixed by "D_" indicate M-PDUs generated by the network in the direction only of the terminal (i.e. Down from the network). NOTE 3: operatorSelection may be used to select a preferred carrier or service provider. NOTE 4: The bearer descriptor set contains an unordered description of all bearers able to maintain the call service.		
Encoding of parameter presence in M-PDU: M = Mandatory; C = Conditional; O = Optional		

B.3.1 U_CallRequest

The U_CallRequest message shall be used by a terminal device (FE5) to request the SpoA for call control (FE6) to establish a call.

Direction: FE5 to FE6;

Response to: TCC_CallSetup_req;

Response expected: D_CallConnect or D_CallReject or D_CallReport.

NOTE 1: For this edition of the document U_CallRequest is normally sent concurrently with U_BearerRequest.

The parameters of the U_CallRequest shall be of type M-PDUCallRequestType, which is formally declared in ASN.1 as below.

```
M-PDUCallRequestType ::= SEQUENCE
{
    callId                  CallIdType,
    callingPartyRestriction IdentityRestrictionType,
    callingPartyId          TIPHONPartyIdType OPTIONAL,
    calledPartyId           TIPHONPartyIdType,
    callpriority            TIPHONCallPriorityType DEFAULT normal,
    operatorSelection       OperatorSelectionType OPTIONAL,
    nWRoutingNumber         SET OF RoutingNumberType OPTIONAL,
    nWLocationData          SET OF LocationData OPTIONAL,
    serviceOfferTicket      TicketType OPTIONAL
}
```

NOTE 2: The nWRoutingNumber and nWLocationData elements are used only by the network functional entities and have no meaning in U_CallRequest.

Where:

```
TIPHONPartyIdType ::= CHOICE
{   e164        E164Type,
    url         VisibleString,
    displayName VisibleString
}

IdentityRestrictionType ::= ENUMERATED
{
    identityAvailable (0),
    identityUnavailable (1)
}

TIPHONCallPriorityType ::= ENUMERATED
{
    normal (0),
    emergency (1),
    authorizedETS (2)
}
```

B.3.2 D_CallReject

The D_CallReject message shall be used by the originating SpoA (FE6) to reject a call attempt by FE5.

Direction: FE6 to FE5;

Response to: U_CallRequest;

Response expected: none.

The parameters of the D_CallReject shall be of type M-PDUCallRejectType.

```
M-PDUCallRejectType ::= SEQUENCE
{
    callId              CallIdType,
    callRejectReason    TIPHONErrorType
}
```

B.3.3 D_CallReport

The D_CallReport message shall be used by the originating SpoA (FE6) to report the progress of a call attempt to the terminal device (FE5).

Direction: FE6 to FE5;

Response to: U_CallRequest or U_CCAdditionalDigits;

Response expected: none.

The parameters of the D_CallReport shall be of type M-PDUCallReportType.

```
M-PDUCallReportType ::= SEQUENCE
{
    callId          CallIdType,
    report          TIPHONReportType,
    reportParams    VisbleString OPTIONAL
}
```

Where:

```
TIPHONReportType ::= ENUMERATED
{
    addressComplete (0),
    addressIncomplete (1),
    callProceeding (2),
    callAlerting (3)
}
```

B.3.4 D_CallConnect

The D_CallConnect message shall be used by the originating SpoA (FE6) to indicate that the call setup phase has been completed and that the media phase can be entered.

Direction: FE6 to FE5;

Response to: U_CallRequest (indirect response);

Response expected: none.

The parameters of the D_CallConnect shall be of type M-PDUCallConnectType.

```
M-PDUCallConnectType ::= SEQUENCE
{
    callId      CallIdType
}
```

NOTE: For this edition of the present document D_CallConnect is normally sent concurrently with U_BearerConnect.

B.3.5 U_CCAdditionalDigits

The U_CCAdditionalDigits message shall be used by a terminal device (FE5) to offer additional address digits to FE6 to finalize the initial phase of call establishment.

Direction: FE5 to FE6;

Response to: D_CallReport (Incomplete address);

Response expected: D_CallReport.

The parameters of the U_CCAdditionalDigits shall be of type M-PDUCCAdditionalDigitsType, where the data-type is formally declared in ASN.1 as below.

```
M-PDUCCAdditionalDigitsType ::= SEQUENCE
{
    callId              CallIdType,
    additionalDigits    SEQUENCE OF DialledDigitType
}
```

Where

```
DialledDigitType ::= VisibleString (SIZE (1)) (FROM ("1234567890#*"))
```

B.3.6 D_CallRequest

The D_CallRequest message shall be used by the terminating SpoA (FE8) to initiate a call to a terminal device (FE9).

Direction: FE8 to FE9;

Response to: none;

Response expected: U_CallAlert.

The parameters of the D_CallRequest shall be of type M-PDUCallRequestType.

NOTE: For this edition of the present document D_CallRequest is normally sent concurrently with D_BearerRequest.

B.3.7 U_CallAlert

The U_CallAlert message shall be used by a terminal device (FE9) to indicate to FE8 an intent to accept the incoming call.

Direction: FE9 to FE8;

Response to: D_CallRequest;

Response expected: none.

The parameters of the U_CallAlert shall be of type M-PDUCallAlertType, which is formally declared in ASN.1 as below.

```
M-PDUCallAlertType ::= SEQUENCE
{
    CallId                  CallIdType
}
```

B.3.8 U_CallConnect

The U_CallConnect message shall be used by a terminal device (FE9) to indicate to FE8 that bearer and media reservations have been made for the call.

Direction: FE9 to FE8;

Response to: D_CallRequest;

Response expected: D_CallConnectAck.

The parameters of the U_CallConnect shall be of type M-PDUCallConnectType, which is formally declared in ASN.1 as below.

```
M-PDUCallConnectType ::= SEQUENCE
{
    CallId                  CallIdType
}
```

B.3.9 Void

B.3.10 NW_CallRequest

The NW_CallRequest message shall be used by a network device (FE6) to request FE7 (or FE8) to establish a call.

Direction: FEx to FEx;

Response to: U_CallRequest (from FE5);

Response expected: NW_CallReport or NW_CallConnect.

The parameters of the NW_CallRequest shall be of type M-PDUCallRequestType.

NOTE: For this edition of the present document NW_CallRequest is normally sent concurrently with NW_BearerRequest.

B.3.11 NW_CallReport

The NW_CallReport message shall be used between network devices to report on the progress of a call.

Direction: FEx to FEx;

Response to: NW_CallRequest;

Response expected: NW_CallConnect or none (see note).

The parameters of the NW_CallReport shall be of type M-PDUCallReportType.

NOTE: If the report is to indicate call leg denial in favour of re-routing the sending entity does not expect a response.

B.3.12 NW_CallConnect

The NW_CallConnect message shall be used by a network device (FE6) to request FE7 (or FE8) to establish a call.

Direction: FEx to FEx;

Response to: NW_CallRequest;

Response expected: none.

The parameters of the NW_CallConnect shall be of type M-PDUCallConnectType.

NOTE: For this edition of the present document NW_CallConnect is normally sent concurrently with NW_BearerConnect.

B.3.13 U_BearerRequest

The U_BearerRequest message shall be used by a terminal device (FE15) to request FE16 to establish a bearer for the associated call.

Direction: FE15 to FE16;

Response to: none;

Response expected: D_BearerConnect.

The parameters of the U_BearerRequest shall be of type M-PDUBearerRequestType, which is formally declared in ASN.1 as below.

```
M-PDUBearerRequestType ::= SEQUENCE
{
    bearerId        BearerIdType,
    uplinkBearer    SET OF BearerDescriptorType,
    downlinkBearer  SET OF BearerDescriptorType
}
```

Where the **BearerDescriptor** combines a **QoSServiceClass** (TIPHON Service class) and a list of **FlowDescriptors** offering multiple ways to achieve that **QoSServiceClass**.

```
BearerDescriptorType ::= SEQUENCE
{
    serviceClass    TIPHONServiceClassType,
    flowDescriptor  SET OF FlowDescriptorType
}
```

The **ServiceClass** represents the TIPHON QoS service class.

```
TIPHONServiceClassType ::= ENUMERATED
{
    bestEffort,             -- R value greater than 50 (not guaranteed)
    narrowbandAcceptable,   -- R value greater than 50 (guaranteed)
    narrowbandMedium,       -- R value greater than 70 (guaranteed)
    narrowbandHigh,         -- R value greater than 80 (guaranteed)
    wideband                -- R value not defined (guaranteed)
}
```

The **FlowDescriptor** describes a media flow and contains information about the codec, terminal delay and a transport descriptor.

```
FlowDescriptorType ::= SEQUENCE
{
    codecDescriptor     CodecDescriptorType,
    delayBudget         INTEGER, -- In milliseconds
    framesPerPacket     INTEGER,
    transportDescriptor TransportDescriptorType
}
```

The **TransportDescriptor** provides all relevant information for the transport of a flow. It contains the following information:

- **maximum gross bit rate:** the maximum bit rate of the CODEC including packetization and framing overhead;
- **delay budget:** the allowed (remaining) delay for the flow;
- **packet rate:** this parameter gives a hint to the transport to arrange the appropriate number of buffers;
- **packet delay variation:** depending on network option either the allowed or achieved variation in the delay (jitter);
- **packet loss:** depending on network option either the allowed or achieved loss of packets in transport;
- **originator transport address:** the sending IP address and port; and
- **destination transport address:** the receiving IP address and port.

```
TransportDescriptorType ::= SEQUENCE
{
    maxCodecGrossBitRate    INTEGER, -- In bits per second
    remainderDelayBudget    INTEGER, -- In milliseconds
    packetRate              INTEGER, -- In packets per second
    packetDelayVariation    INTEGER, -- In milliseconds
    packetLoss              INTEGER, -- In percent
    originatorMpoA          MpoAType,
    destinationMpoA         MpoAType
}
```

NOTE: For this edition of the present document U_BearerRequest is normally sent concurrently with U_CallRequest.

B.3.14 D_BearerConnect

The D_BearerConnect message shall be used by the network (FE16) to indicate to FE15 the bearers for the associated call.

Direction: FE16 to FE15;

Response to: U_BearerRequest;

Response expected: none.

The parameters of the D_BearerConnect shall be of type M-PDUBearerConnectType where only one flow descriptor shall appear for each direction.

```
M-PDUBearerConnectType ::= SEQUENCE
{
    bearerId        BearerIdType,
    uplinkBearer    BearerSelectionType,
    downlinkBearer  BearerSelectionType
}
```

Where:

```
BearerSelectionType ::= SEQUENCE
{
    serviceClass    TIPHONServiceClassType,
    flowDescriptor  FlowDescriptorType
}
```

B.3.15 NW_BearerRequest

The NW_BearerRequest message shall be used between network functional elements (FE16, FE17, FE18) to establish a bearer for the associated call.

Direction: FE16 [to FE17] to FE18;

Response to: U_BearerRequest (from FE15);

Response expected: NW_BearerConnect.

The parameters of the NW_BearerRequest shall be of type M-PDUBearerRequestType.

B.3.16 NW_BearerConnect

The NW_BearerConnect message shall be used between network functional elements (FE16, FE17, FE18) to establish a bearer for the associated call.

Direction: FE18 [to FE17] to FE16;

Response to: NW_BearerRequest;

Response expected: none.

The parameters of the NW_BearerConnect shall be of type M-PDUBearerConnectType where only one flow descriptor shall appear for each direction.

B.3.17 NW_CallReject

The NW_CallReject message shall be used by a network device to reject the establishment of a call.

Direction: FE7 to FE6;

Response to: NW_CallRequest;

Response expected: none.

The parameters of the NW_CallReject shall be of type M-PDUCallRejectType.

B.4 Behaviour

B.4.1 Call setup

The Call Control behaviour is determined in terms of a finite state machine and the allowed transitions between states. The TCC states are defined in table 10.

NOTE: The states defined here are for the meta-protocol only and may not be visible in a compliant mapping by a candidate protocol. The compliant mapping may be stateful or stateless.

Table 10: Terminal Call Control states

State	Summary description
IDLE	The resting state for the call control entity. This state is reached after successful registration and on opening of the protocol.
CALLACTIVE	The media state of a call.
TO_CALLSETUP	The state during which terminal originated calls are established.
TT_CALLSETUP	The state during which terminal terminated calls are established.
CALLDISCONNECT	The intermediate state whilst awaiting confirmation of call clearance before returning to IDLE state.

B.4.1.1 Outgoing call (terminal originated, terminal behaviour)

The text, Message Sequence Charts (MSCs), and SDL given in this clause indicate the normative behaviour of the terminal to serving call control entity (FE5 to FE6).

To initiate the call setup the user application shall transfer a TCC_CallSetup_req primitive across TCC_SAP to the Call Control (CC) entity (FE5). The TCC_CallSetup_req shall be handled by a CC entity that is in state IDLE.

FE5 shall convert the content of the TCC_CallSetup_req into a corresponding U_CallRequest M-PDU and send it using the TT_TransportFrame_req primitive (see annex D) to the service specific SpoA (identified at registration). FE5 shall then enter the TO_CALLSETUP state and shall start timer TC001.

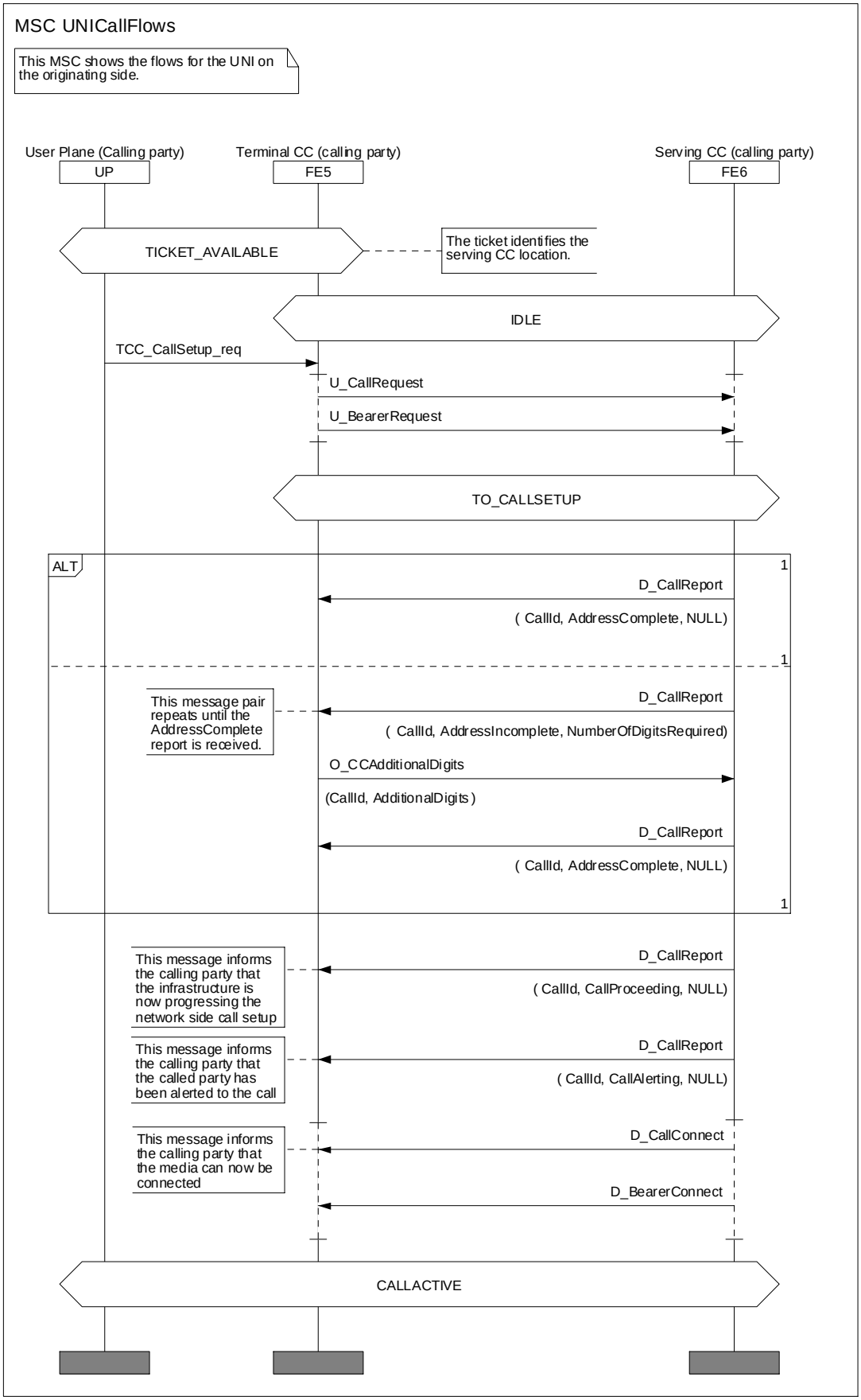
The server call control entity (FE6) shall validate the ticket. If any element of the ticket is invalid FE6 shall signal to FE5 using the D_CallReject M-PDU and FE5 shall return to state IDLE.

FE6 shall validate the called party identifier. If the called party identifier is complete FE6 shall signal to FE5 using the D_CallReport M-PDU with report reason set to AddressComplete and the additional report parameters field set to NULL. If however the called party identifier is incomplete FE6 shall notify FE5 using the D_CallReport M-PDU with report reason set to AddressIncomplete and the additional report parameter field set to indicate the number of digits missing. FE5 shall respond with the U_CCAdditionalDigits M-PDU. This exchange shall continue until such time as FE6 determines that the called party identifier is complete and signalled to FE5 using the D_CallReport M-PDU with report reason set to AddressComplete and the additional report parameter field set to NULL.

FE6 shall indicate to FE5 its intention to further process the call request by signalling using the D_CallReport M-PDU with report reason set to CallProceeding and the additional report parameter field set to NULL. The CallId field shall at this time be fixed by FE6 for the call and all further M-PDUs for this call shall use this value of CallId.

On receipt of signalling from the terminating side (FE8/FE9) FE6 shall signal to FE5 using the D_CallReport M-PDU with report reason set to CallAlerting and the additional report parameter field set to NULL. At this point FE6 is able to indicate to FE5 the bearer characteristics that shall be used in the call.

On receipt of signalling from the terminating side (FE8/FE9) FE6 shall signal to FE5 using the D_CallConnect M-PDU. FE5 shall then establish the bearers indicated by FE6, FE5 shall clear timer TC001, and move to state CALLACTIVE.



NOTE: D_CallConnect is only sent after bearer and media have been established.

Figure 11: Initial flows of terminal originated call setup at UNI (C1)

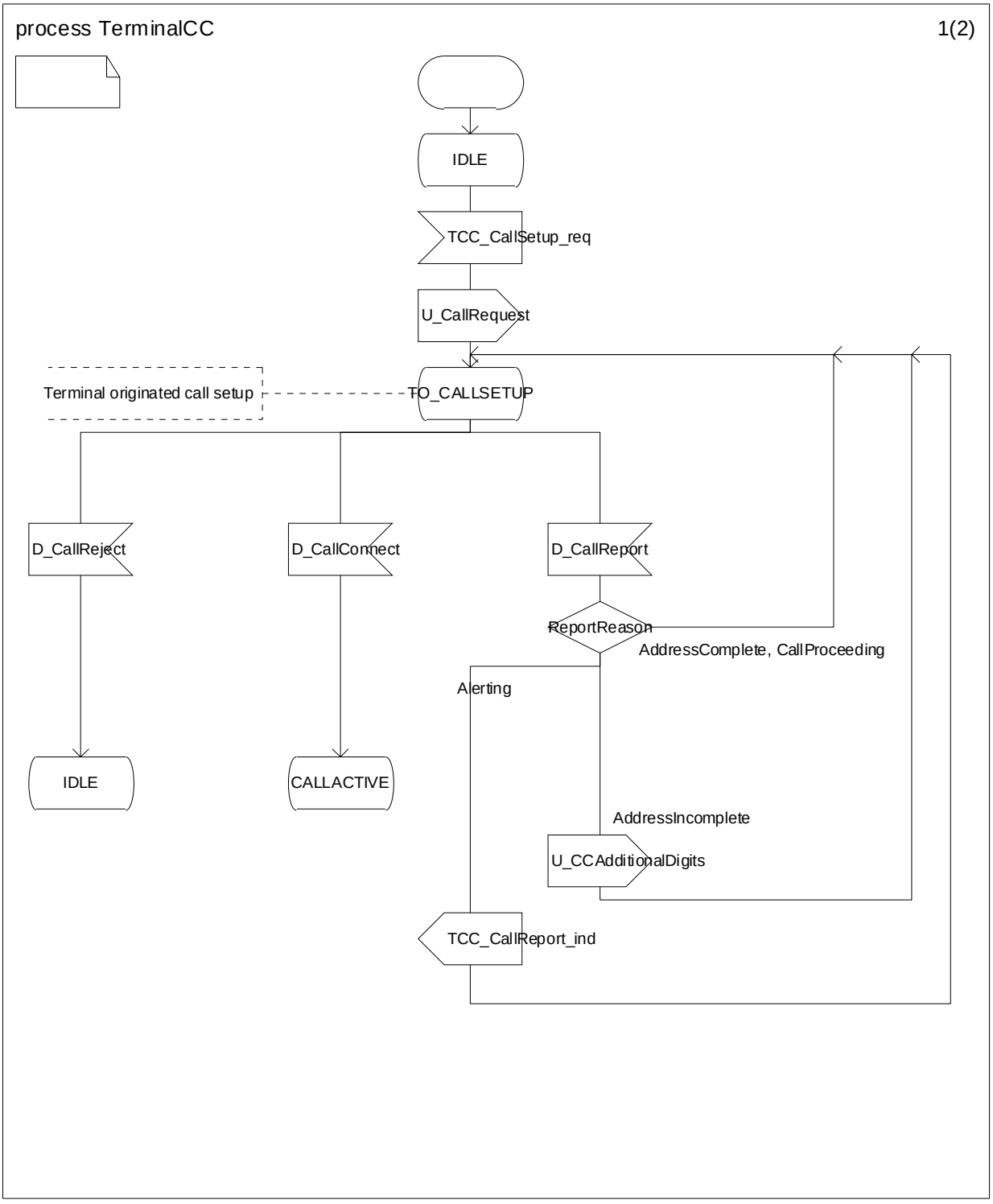


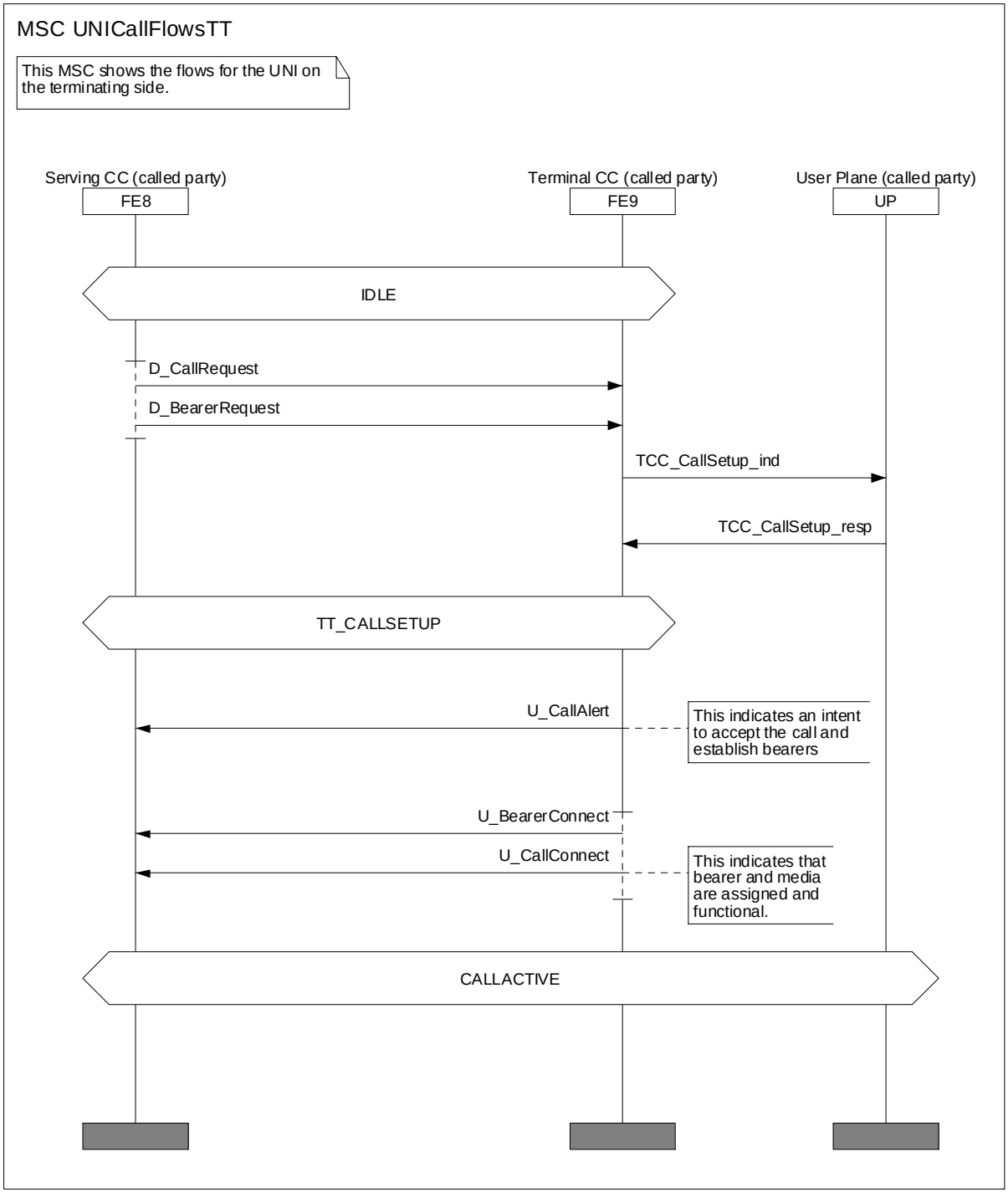
Figure 12: UNI state transition model for terminal originated call setup

B.4.1.2 Incoming call (Terminal Terminated, terminal behaviour)

Notification of the arrival of an incoming call attempt to the terminal call control entity (FE9) shall be made by reception of the D_CallRequest M-PDU. This notification shall be delivered to the application plane using the TCC_CallSetup_ind primitive to which the application shall respond using the TCC_CallSetup_resp primitive indicating to FE9 to accept or reject the call. On receipt of D_CallRequest FE9 shall enter state TT_CALLSETUP and shall start timer TC002.

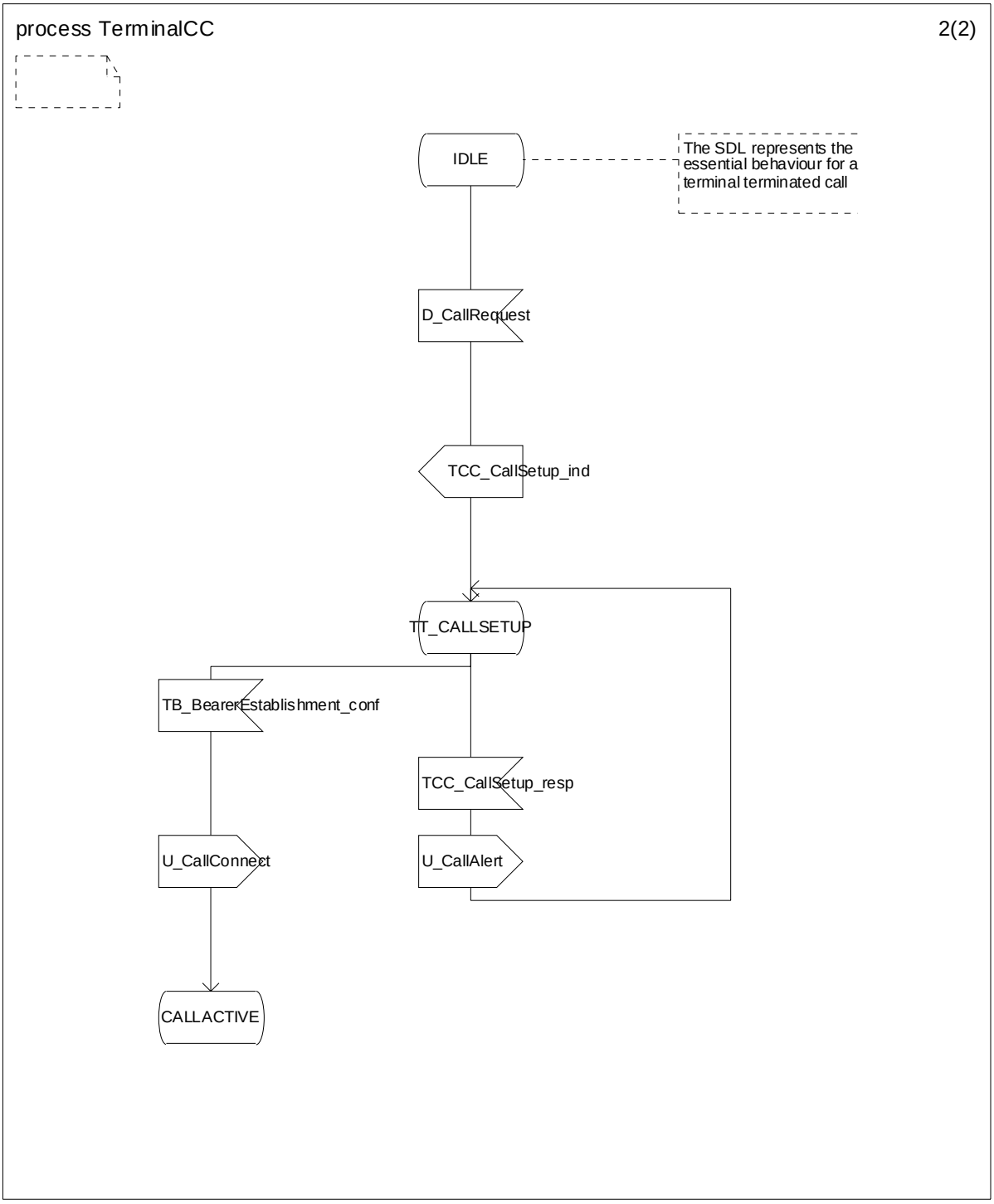
The D_CallRequest M-PDU shall contain the value of CallId to be used in all subsequent signalling related to the call. The D_CallRequest M-PDU shall contain the bearer characteristics assigned by FE8 to the call and pass these to the bearer control entity using the TB_BearerReserve_req primitive.

On receipt of the TB_BearerReserve_conf primitive indicating that the bearer and media resources for the requested call have been made available FE9 shall send U_CallConnect M-PDU to FE8, shall clear timer TC002, and shall enter state CALLACTIVE.



NOTE: U_CallConnect is only sent after bearer and media have been established.

Figure 13: UNI flows for terminal terminated call setup



NOTE: The TB_BearerReserve_req is shown to indicate that the bearer and media have been established and thus to allow the active phase of the call to be entered.

Figure 14: UNI state transition model for terminal terminated call setup

B.4.1.3 Network behaviour

FE6 shall receive U_CallRequest M-PDU from FE5 and shall enter state TO_CALLSETUP.

The ServiceApplication element in U_CallRequest shall indicate to FE6 the QoS and service class requested. The authorization of the calling party to make a call attempt against the requested service application shall be verified. If the service application is authorized the call setup processing shall continue. If the service application is not authorized the call setup shall be rejected and FE6 shall return D_CallReject with CallRejectReason set to "Service not authorized" and FE6 shall enter state IDLE.

FE6 shall determine the route to the next call control server by validating the CalledParty identity and determining a routing address (the identity of the next SpoA in the chain). If the called party identity cannot be validated with reason "Incomplete Called Party Identity" FE6 shall request additional digits by sending D_CallReport with ReportReason set to "Address Incomplete". If the called party identity cannot be validated with reason "Unknown Called Party Identity" FE6 shall return D_CallReject with CallRejectReason set to "Called Party not recognized" and FE6 shall enter state IDLE.

If the called party address is valid FE6 shall determine the route to the next signalling element (FE7 or FE8) and send NW_CallRequest to the next SpoA (i.e. FE7 or FE8).

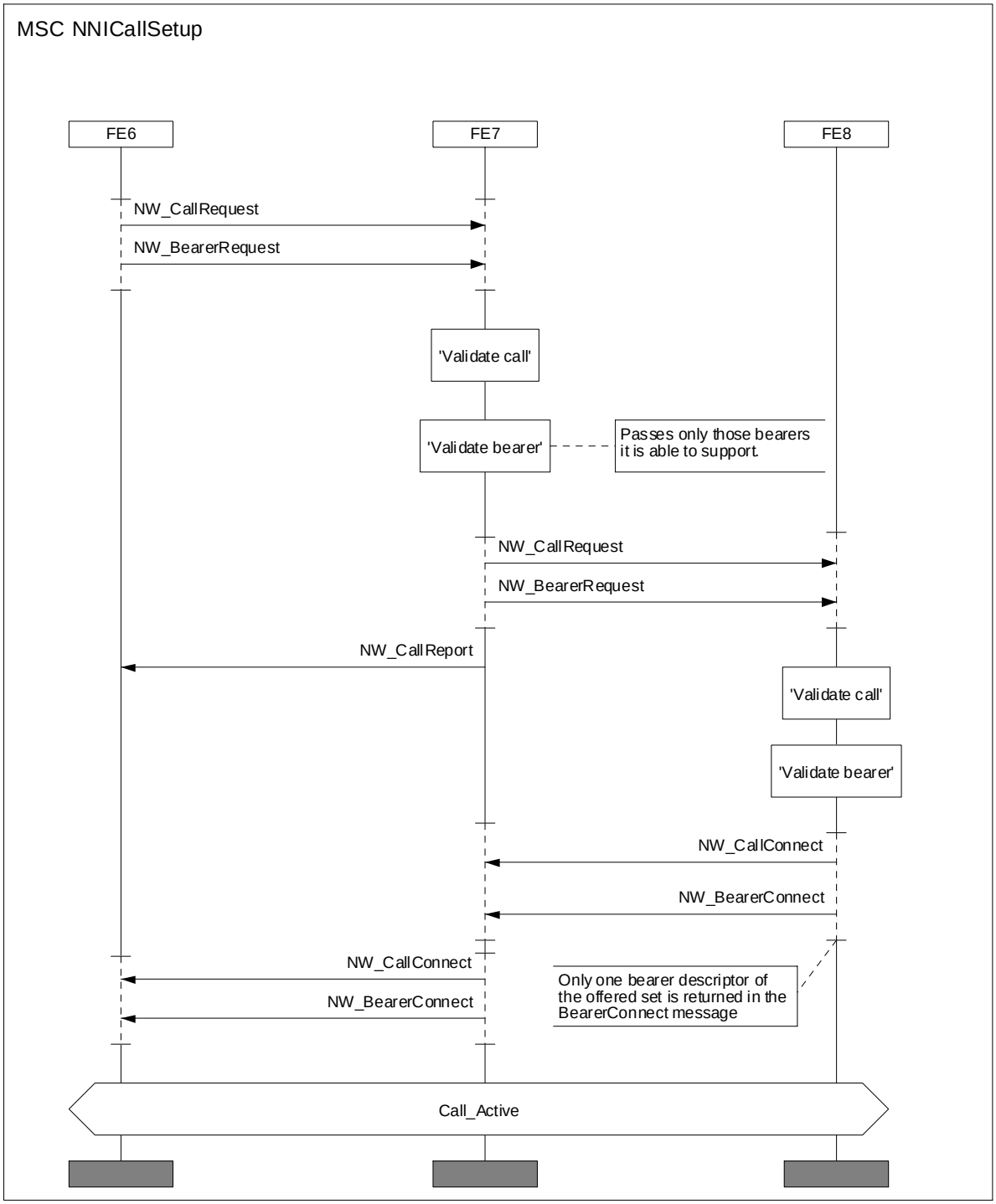


Figure 15: NNI Information flows for call setup

B.4.2 Call Cleardown

B.4.2.1 Terminal behaviour

Calls can be cleared by either the calling or the called party. Each entity in the call path (media and signalling) shall return to the IDLE state as a result of cleardown.

NOTE 1: For purposes of clarity the remainder of this description will refer to FE5 as the clearing CC entity.

NOTE 2: The behaviour in this clause applies to non-emergency calls only.

The clearing party shall initiate the call cleardown phase by issuing a TCC_CallClear_req primitive to FE5 across TCC_SAP. FE5 shall enter state CALLDISCONNECT and send an U_CallClear M-PDU to FE6 and enter sub-state WAITRELEASE.

On receipt of M-PDU D_RELEASE from FE6 FE5 shall send TCC_CallClear_conf primitive across TCC-SAP to the application, shall clear CallId, and shall initiate release of the bearer by using the TB_BearerRelease_req primitive across TB-SAP. On receipt of TB_BearerRelease_conf primitive across TB-SAP FE5 shall enter state IDLE.

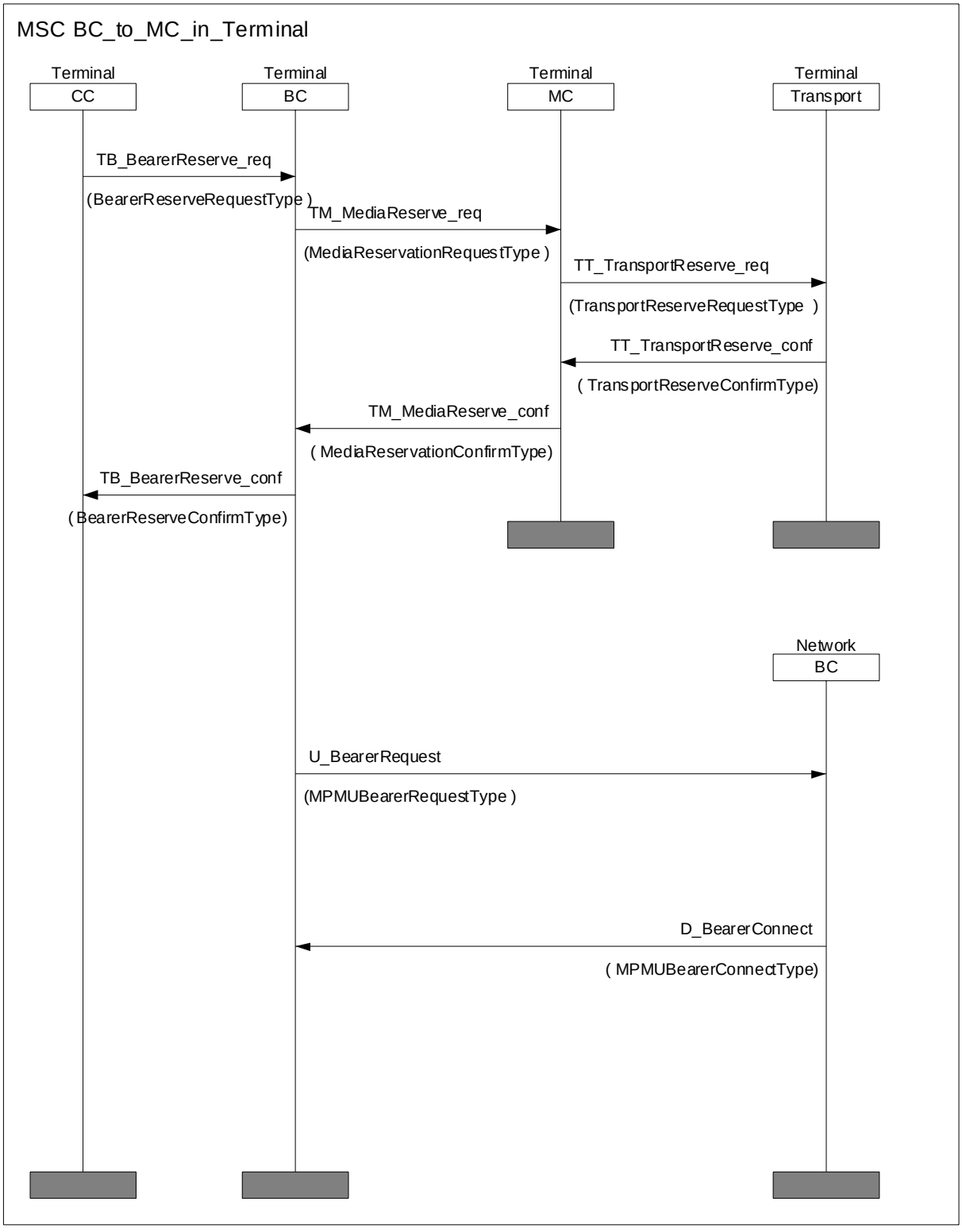


Figure 16: Bearer pre-selection

B.5 Timers

B.5.1 TC001, originating call control, call setup timer

The default value of this timer shall be 30 s.

B.5.2 TC002, terminating call control, call setup timer

The default value of this timer shall be 30 s.

B.6 Data definitions (ASN.1)

This clause contains the data definitions for information flows visible at reference point C.

NOTE: When checked using the OSS Syntax checker this ASN.1 contains no errors but warns that some M-PDUs encode to the same form of structure or that duplicate tags appear in structures. This is not treated in this clause as an error but as information.

```
TIPHONr3-RefPtC DEFINITIONS ::=

BEGIN

--
-- Start of M-PDU data definitions
--

M-PDUCallRequestType ::= SEQUENCE
{
    callId                  CallIdType,
    callingPartyRestriction IdentityRestrictionType,    callingPartyId          TIPHONPartyIdType
OPTIONAL,
    calledPartyId           TIPHONPartyIdType,
    callpriority            TIPHONCallPriorityType DEFAULT normal,
    nwData                  SET OF VisibleString OPTIONAL,
    operatorSelection       OperatorSelectionType OPTIONAL,
    nWLocationData          SET OF LocationData OPTIONAL,
    nWRoutingNumber         SET OF RoutingNumberType OPTIONAL,
    serviceOfferTicket      TicketType OPTIONAL
}

M-PDUCallRejectType ::= SEQUENCE
{
    callId              CallIdType,
    callRejectReason    TIPHONErrorType
}

M-PDUCallReportType ::= SEQUENCE
{
    callId          CallIdType,
    report          TIPHONReportType,
    reportParams    VisibleString OPTIONAL
}

M-PDUCallConnectType ::= SEQUENCE
{
    callId      CallIdType
}

M-PDUCCAdditionalDigitsType ::= SEQUENCE
{
    callId              CallIdType,
    additionalDigits    SEQUENCE OF DialledDigitType
}

M-PDUCallAlertType ::= SEQUENCE
{
    callId      CallIdType
```

```
}

M-PDUBearerRequestType ::= SEQUENCE
{
    bearerId        BearerIdType,
    uplinkBearer    SET OF BearerDescriptorType,
    downlinkBearer  SET OF BearerDescriptorType
}

M-PDUBearerConnectType ::= SEQUENCE
{
    bearerId        BearerIdType,
    uplinkBearer    BearerSelectionType,
    downlinkBearer  BearerSelectionType
}

--
-- End of M-PDU data definitions
--

--
-- Start of data element definitions
--

Addr32 ::= SEQUENCE (SIZE(4)) OF Byte

Addr128 ::= SEQUENCE (SIZE(16)) OF Byte

BearerDescriptorType ::= SEQUENCE
{
    serviceClass    TIPHONServiceClassType,
    flowDescriptor  SET OF FlowDescriptorType
}

BearerIdType ::= INTEGER

BearerSelectionType ::= SEQUENCE
{
    serviceClass    TIPHONServiceClassType,
    flowDescriptor  FlowDescriptorType
}

Byte ::= INTEGER (0..255)

CallIdType ::= INTEGER

CodecDescriptorType ::= VisibleString

DialledDigitType ::= VisibleString (SIZE (1)) (FROM ("1234567890#*"))

DigestType ::= VisibleString

E164Type ::= SEQUENCE
{
    natureOfAddress     NatureOfAddressType,
    screeningIndicator  ENUMERATED
                        {
                            userProvided,
                            notVerifiedUserProvided,
                            verifiedAndPassedNetworkProvided
                        } OPTIONAL,
    digits              SEQUENCE OF TelephoneDigitType
}

NatureOfAddressType ::= ENUMERATED
{
    nationalSubscriberNumber,
    nationalUnknown,
    nationalSignificantNumber,
    internationalNumber,
    nationalNetworkSpecificNumber,
    nationalSignificantRoutingNumber,
    …
}
```

```
FlowDescriptorType ::= SEQUENCE
{
    codecDescriptor     CodecDescriptorType,
    delayBudget         INTEGER, -- In milliseconds
    framesPerPacket     INTEGER,
    transportDescriptor SET OF TransportDescriptorType
}

IdentityRestrictionType ::= ENUMERATED
{
    identityAvailable,
    identityUnavailable
}

IPv4Type ::= SEQUENCE
{
    iPv4address     Addr32,
    ipv4protocol    MpoAProtocolType,
    ipv4port        INTEGER (0..65535)
}

IPv6Type ::= SEQUENCE
{
    iPv6address     Addr128,
    ipv6protocol    MpoAProtocolType,
    ipv6port        INTEGER (0..65535)
}

LocationData ::= VisibleString

MpoAProtocolType ::= ENUMERATED
{
    uDP,
    rTP,
    rTCP_plus_RTCP,
    rTCP
}

MpoAType ::= CHOICE
{
    iPv4address     IPv4Type,
    iPv6address     IPv6Type,
    scn             INTEGER  -- SCN
}

NetworkFGIdType ::= VisibleString

NetworkIdType ::= VisibleString

OperatorSelectionType ::= SEQUENCE OF TelephoneDigitType

RoutingNumberType ::= CHOICE
{
    toSCN   RoutingNumberToSCNType,
    toIP    RoutingNumberToIPType
}

RoutingNumberToSCNType ::= CHOICE
{
    recipientNetworkID      NetworkIDType,
    pointOfInterconnection  MpoAType,
    recipientExchange       SomeTypeC
}

RoutingNumberToIPType ::= CHOICE
{
    serviceProviderIdentity SpoAType,
    pointOfInterconnection  MpoAType,
    recipientNetworkFG      NetworkFGIdType
}

ServiceApplicationType ::= VisibleString
```

```
ServiceCredentialType ::= SEQUENCE
{
    serviceAppId    ServiceApplicationType,
    spoA            SpoAType, -- shall be in the user's terminal addressing realm
    startTime       GeneralizedTime,
    stopTime        GeneralizedTime, -- Shall be greater than StartTime
    cryptoDigest    DigestType OPTIONAL
}

SpoAType ::= VisibleString

TelephoneDigitType ::= NumericString (SIZE (1)) (FROM ("1234567890"))

TicketType ::= SEQUENCE
{
    registrantId        VisibleString,
    registrarId         VisibleString,
    serviceCredential   SET OF ServiceCredentialType,
    cryptoDigest        DigestType OPTIONAL
}

TIPHONCallPriorityType ::= ENUMERATED
{
    normal (0),
    emergency (1),
    authorizedETS (2)
}

TIPHONErrorType ::= SEQUENCE
{
    source      ENUMERATED
                {
                    callControl,
                    bearerControl,
                    mediaControl,
                    transportControl,
                    …
                },
    severity    ENUMERATED
                {
                    fatalError,
                    warning,
                    information
                },
    reason      ENUMERATED
                {
                    invalid,
                    not_Supported,
                    unavailable,
                    does_not_exist,
                    insufficient_resources,
                    …
                },
    dataElementInError  ASN1ElementId OPTIONAL,
    diagnostic  TIPHONErrorDiagnosticType OPTIONAL,
    freeText    VisibleString OPTIONAL,
    embeddedError   TIPHONErrorType OPTIONAL
}

TIPHONPartyIdType ::= CHOICE
{   e164        E164Type,
    url         VisibleString,
    displayName VisibleString
}

TIPHONReportType ::= ENUMERATED
{
    addressComplete (0),
    addressIncomplete (1),
    callProceeding (2),
    callAlerting (3),
    nwCallLegDenialNoData,
    nwCallLegDenialReroutingData, -- Add Rerouting data in reportParams element
}
```

```
TIPHONServiceClassType ::= ENUMERATED
{
    bestEffort ,            -- R value greater than 50 (not guaranteed)
    narrowbandAcceptable,   -- R value greater than 50 (guaranteed)
    narrowbandMedium,       -- R value greater than 70 (guaranteed)
    narrowbandHigh,         -- R value greater than 80 (guaranteed)
    wideband                -- R value not defined (guaranteed)
}

UserInputType ::= CHOICE
{
    userDialledDigits   SEQUENCE OF DialledDigitType,
    userEnteredString   VisibleString
}

TransportDescriptorType ::= SEQUENCE
{
    maxCodecGrossBitRate    INTEGER, -- In bits per second
    remainderDelayBudget    INTEGER, -- In milliseconds
    packetRate              INTEGER, -- In packets per second
    packetDelayVariation    INTEGER, -- In milliseconds
    packetLoss              INTEGER, -- In percent
    originatorMpoA          MpoAType,
    destinationMpoA         MpoAType
}

--
-- End of data element definitions
--

END
```

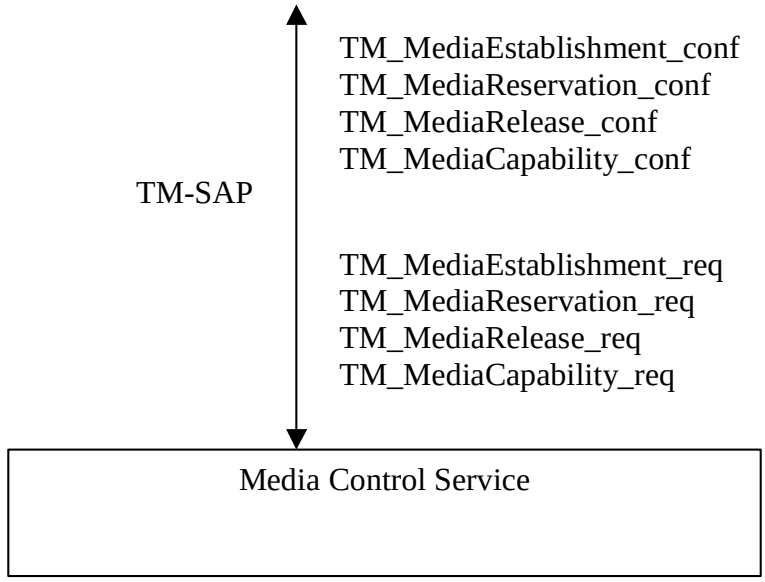
Annex C (normative): Meta-protocol at reference point N

C.1 Media control service

Media control allows reservation and allocation of resources for formatting of the media stream (e.g. to reserve processing capability for soft codecs, or to switch into the path hard codecs).

Table 11: Media control primitives visible at TM-SAP

Primitive	Short description	Capability (see note)
TM_MediaEstablishment_req	Used by the application to request establishment of previously reserved media resources.	
TM_MediaEstablishment_conf	Confirms the establishment.	
TM_MediaReservation_req	Requests the media controller to reserve one of its currently available media options.	
TM_MediaReservation_conf	Confirms the reservation.	
TM_MediaCapability_req	Requests the media controller to report on the currently available media options	
TM_MediaCapability_conf	Provides the upper layers with the currently available media options.	
TM_MediaRelease_req	Requests the media controller to release currently established media flows.	
TM_MediaRelease_conf	Confirms to the upper layers that previously established media flows have been released.	
NOTE: The capability cross references to the capability definition in TS 101 878 [3].		



NOTE: TM-SAP encompasses reference point N defined in [1].

Figure 17: Services offered by the Media Control service identified by primitives

Media Control (MC) shall establish the media elements required to support both call and bearer. MC shall be used to establish a QoS controlled transport capability in accordance with the QoS class identified by the call control meta-protocol.

MC shall do the following:

- Maintain the media state.
- Setup and release media elements.

MC is described by a finite state machine as shown in figures 18 and 19. The behaviour of MC at each state transition is described.

NOTE: Each reservation applies to one media resource and associated traffic resource only. If multiple media reservations are required then multiple invocations of this interface are also required.

Table 12: Parameters in media control primitives

Primitive	Parameters	
	Request	Confirm
TM_MediaReservation	RequestedMediaHandle RxFlowDescriptor TxFlowDescriptor	RequestedMediaHandle Status ReservedMediaHandle ReservedRxFlowDescriptor ReservedTxFlowDescriptor
TM_MediaEstablishment	ReservedMediaHandle	ReservedMediaHandle Status EstablishedMediaHandle
TM_MediaRelease	EstablishedMediaHandle	EstablishedMediaHandle Status
TM_MediaCapability	TBD	TBD

Where:

- xxMediaHandle shall be of type MediaHandleType (see clause C.2).
- xxFlowDescriptor shall be of type FlowDescriptorType (see clause C.2).
- Status shall be of type MediaStatusType (see clause C.2).

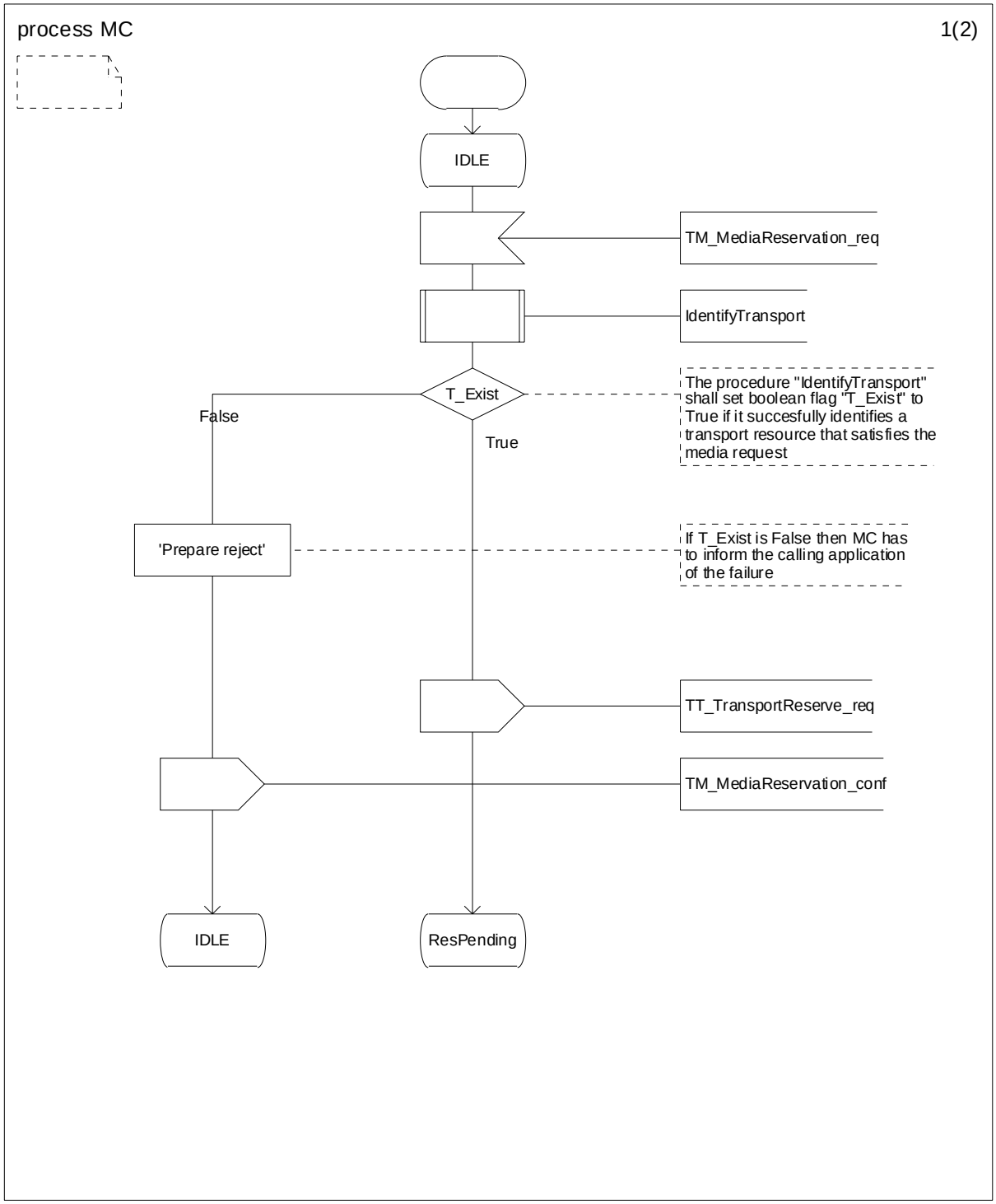


Figure 18: State transition diagram of MC process (page 1 of 2)

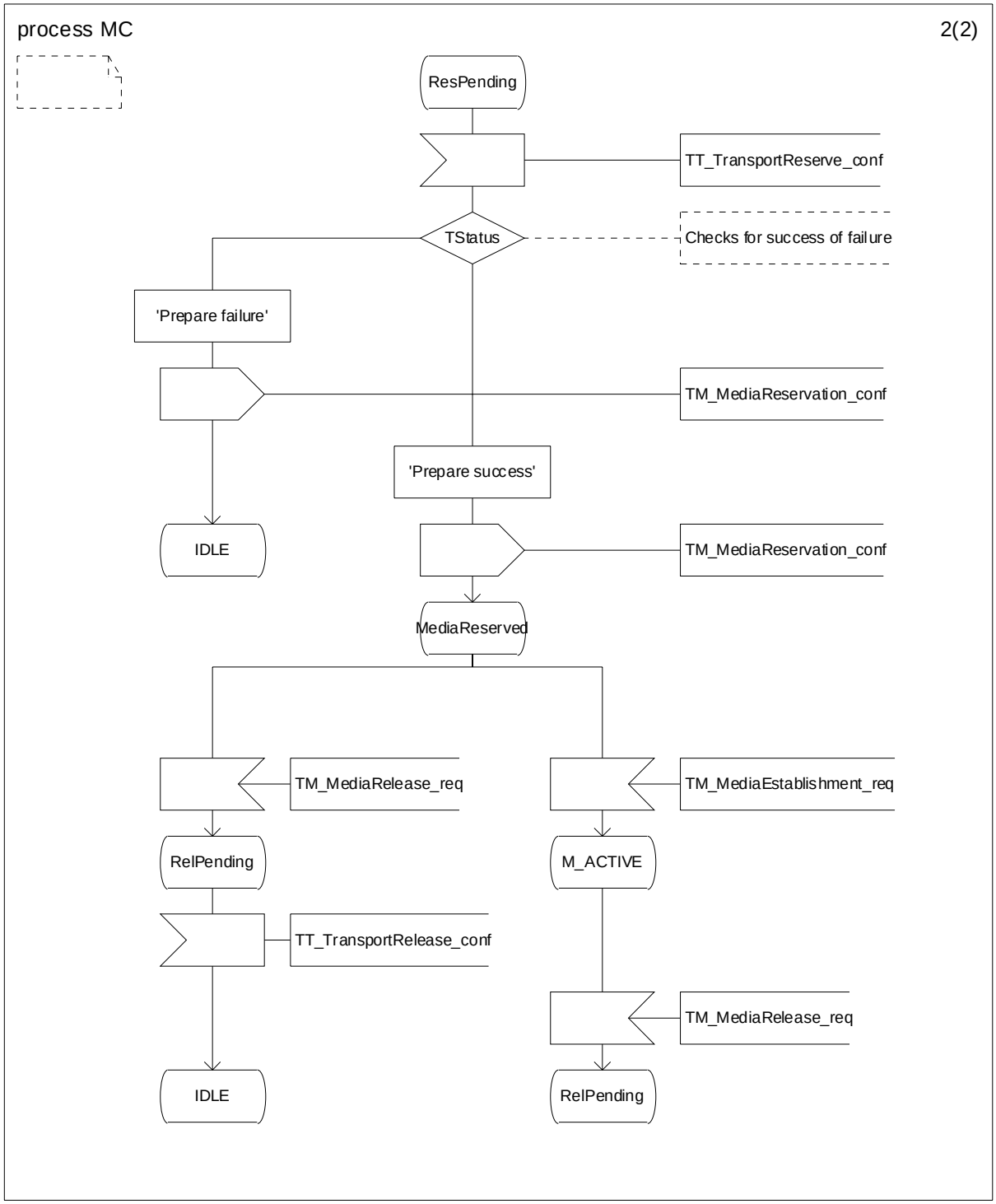


Figure 19: State transition diagram of MC process (page 2 of 2)

Table 13: States of MC

State	Transition message	Next state	Description of behaviour
IDLE	TM_MediaReservation_req	ResPending	Clause C.1.1
ResPending	TT_TransportReservation_conf (success) TT_TransportReservation_conf (failure)	MediaReserved IDLE	Clause C.1.2
MediaReserved	TM_MediaRelease_req TM_MediaEstablishment_req	RelPending M_ACTIVE	Clause C.1.3
RelPending	TT_TransportRelease_conf	IDLE	Clause C.1.4
M_ACTIVE	TM_MediaRelease_req	RelPending	Clause C.1.5
NOTE: Messages which do not result in a change of state are not indicated in this table.			

C.1.1 Behaviour in IDLE state

When in IDLE state no reservations of media, and no activations of media, shall be in force.

On receipt of message TM_MediaReservation_req MC shall invoke the "IdentifyTransport" procedure that identifies the transport requirements for the codec identified in the TM_MediaReservation_req and take the result into TT_TransportReserve_req to be sent across TT_SAP to the transport plane and move to state ResPending. If procedure "IdentifyTransport" fails to identify a set of transport resources MC shall prepare TM_MediaReservation_conf with status set to "Failure, no transport resource identified" and return to state IDLE.

C.1.2 Behaviour in ResPending state

In ResPending state MC is waiting for TT_TransportReserve_conf. On receipt of TT_TransportReserve_conf MC shall check the value of the status element. If the received status value indicates failure MC shall prepare TM_MediaReservation_conf with status set to "Failure, transport resource unavailable" and return to state IDLE. If the received status value indicates success MC shall prepare TM_MediaReservation_conf with status set to "Success" and shall return the media descriptors and enter state MediaReserved.

C.1.3 Behaviour in MediaReserved state

In MediaReserved state media can either be confirmed and put into use, or can be released. On receipt of TM_MediaRelease_req MC shall send TT_TransportRelease_req to make the Transport controller release all reservations made prior to entering this state. On receipt of TM_MediaEstablishment_req MC shall enter state M_ACTIVE.

C.1.4 Behaviour in RelPending state

In state RelPending MC shall respond only to a received TT_TransportRelease_conf primitive indicating that the resource has been released in the transport plane and shall move to state IDLE.

C.1.5 Behaviour in M_ACTIVE state

In M_ACTIVE state media transported media packets are passed through the assigned media resources. On receipt of TM_MediaRelease_req MC shall issue a TT_TransportRelease_req command to the transport controller and move to state RelPending.

C.2 Data definitions (ASN.1)

This clause contains the data definitions for information flows visible at reference point N.

```
TIPHONr3-RefPtN DEFINITIONS ::=

BEGIN

-- Start of primitive definitions

MediaReservationRequestType ::= SEQUENCE
{
    invokingControllerReference MediaHandleType,
    rxFlowDescriptor        SET OF FlowDescriptorType,
    txFlowDescriptor        SET OF FlowDescriptorType
}

MediaReservationConfirmType ::= SEQUENCE
{
    invokingControllerReference MediaHandleType,
    status                  MediaStatusType,
    mediaDescriptor         SET OF MediaDescriptorWithHandle
}
```

```
MediaEstablishementRequestType ::= SEQUENCE
{
    mcReservedMediaReference        MediaHandleType
}

MediaEstablishmentConfirmType ::= SEQUENCE
{
    mcReservedMediaReference        MediaHandleType,
    status                  MediaStatusType,
    mcEstablishedMediaReference MediaHandleType OPTIONAL
}

MediaReleaseRequestType ::= SEQUENCE
{
    mcEstablishedMediaReference MediaHandleType
}

MediaReleaseConfirmType ::= SEQUENCE
{
    mcEstablishedMediaReference MediaHandleType,
    status                  MediaStatusType
}

-- End of primitive definitions

-- Start of element definitions

MediaDescriptorWithHandle ::= SEQUENCE
{
    mcReservedMediaReference            MediaHandleType,
    reservedrxFlowDescriptor    FlowDescriptorType,
    reservedtxFlowDescriptor    FlowDescriptorType
}

MediaStatusType ::= TIPHONErrorType

MediaHandleType ::= VisibleString

FlowDescriptorType ::= SEQUENCE
{
    codecDescriptor         CodecDescriptorType,
    framesPerPacket         INTEGER,
    frameRate               INTEGER,
    transportDescriptor     SET OF TransportDescriptorType
}

TransportDescriptorType ::= SEQUENCE
{
    genericQoSDescriptor    GenericQoSDescriptorType,
    specificQoSDescriptor   SpecificQoSDescriptorType OPTIONAL,
    originatorMpoA          MpoAType,
    destinationMpoA         MpoAType
}

SpecificQoSDescriptorType ::= VisibleString

GenericQoSDescriptorType ::= SEQUENCE
{
    delayBudget             INTEGER, -- In milliseconds
    maxPacketSize           INTEGER, -- In bits
    packetRate              INTEGER, -- In packets per second
    packetDelayVariation    INTEGER, -- In milliseconds
    packetLoss              INTEGER - In percent
}

CodecDescriptorType ::= SEQUENCE
{
    codecID                     ENUMERATED {g711, ...},
    silenceSuppressionEnabled   BOOLEAN,
    codecSpecificParameters     VisibleString
}
```

```
TIPHONErrorType ::= SEQUENCE
{
    source      ENUMERATED
                {
                    callControl,
                    bearerControl,
                    mediaControl,
                    transportControl,
                    …
                },
    severity    ENUMERATED
                {
                    fatalError,
                    warning,
                    information
                },
    reason      ENUMERATED
                {
                    invalid,
                    not_Supported,
                    unavailable,
                    does_not_exist,
                    insufficient_resources,
                    …
                },
    diagnostic  TIPHONErrorDiagnosticType OPTIONAL,
    freeText    VisibleString OPTIONAL,
    embeddedError   TIPHONErrorType OPTIONAL
}

Byte ::= INTEGER (0..255)

IPv4Type ::= SEQUENCE
{
    iPv4address     Addr32,
    ipv4protocol    MpoAProtocolType,
    ipv4port        INTEGER (0..65535)
}

IPv6Type ::= SEQUENCE
{
    iPv6address     Addr128,
    ipv6protocol    MpoAProtocolType,
    ipv6port        INTEGER (0..65535)
}

MpoAType ::= CHOICE
{
    iPv4address     IPv4Type,
    iPv6address     IPv6Type,
    scn             INTEGER  -- SCN
}

MpoAProtocolType ::= ENUMERATED
{
    uDP,
    rTP,
    rTCP_plus_RTCP,
    rTCP
}

Addr32 ::= SEQUENCE (SIZE(4)) OF Byte

Addr128 ::= SEQUENCE (SIZE(16)) OF Byte

-- End of element definitions

END
```

Annex D (normative): Meta-protocol at reference point T

D.0 Introduction

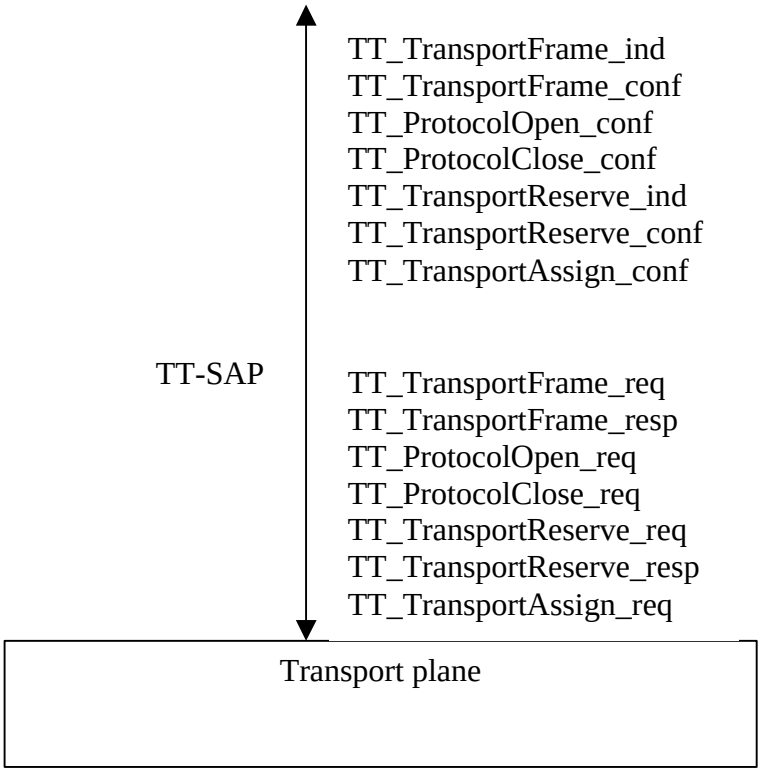
The transport plane offers the following services to the application plane (visible at TIPHON Transport Service Access Point (TT_SAP)):

- Transport of a frame (usually an M-PDU from CC or Registration).
- Reservation of a specific transport capability.

These services are established through a set of primitives as shown in table 14.

Table 14: Transport primitives visible at TT-SAP

Primitive	Short description
TT_TransportFrame_req/conf	Requests the transport plane to transfer a frame of data.
TT_TransportFrame_ind/resp	Indication from the transport plane to the application plane that a frame of data has been received.
TT_TransportReserve_req/conf	Request to the transport plane to reserve a transport capability. To be used in QoS transport establishment.
TT_TransportReserve_ind/resp	Indication from the transport plane to the application plane that a transport capability has been reserved.
TT_TransportAssign_req/conf	Moves a transport reservation into a transport assignment.
TT_TransportRelease_req/conf	Request to the transport plane to release a previously reserved transport capability.
TT_ProtocolOpen_req/conf	Request to open a path from the transport plane to the application plane for a specific protocol.
TT_ProtocolClose_req/conf	Request to close a path from the transport plane to the application plane for a specific protocol.
TT_TransportCapability_req/conf	Allows the application plane to request the current capability of the transport plane.
TT_TransportCapability_ind/resp	Allows the transport plane to indicate to the application plane its current capabilities.



NOTE: TT-SAP encompasses reference point T defined in [1].

Figure 20: Services offered by the Transport plane identified by primitives

Table 15: Parameters in transport primitives (req/conf)

Primitive	Parameters	
	Request	**Confirm**
TT_TransportReserve	TargetTransportDescriptor EndPointId	ReservedTransportHandle ReservedTransportDescriptor
TT_TransportAssign	ReservedTransportHandle	AssignedTransportHandle
TT_TransportFrame	TransportHandle EndPointId DataField	ResponseDataField
TT_TransportRelease	AssignedTransportHandle	-
TT_ProtocolOpen	ProtocolId	ProtocolOpenResult
TT_ProtocolClose	ProtocolId	ProtocolCloseResult
NOTE: The TransportHandle is returned on a successful reservation. Predefined values of the TransportHandle are used for non-QoS transport.		

Table 16: Parameters in transport primitives (ind/resp)

Primitive	Parameters	
	Indication	**Response**
TT_TransportFrame	DataField ConfirmationFlag ResponseToEndPointId	ResponseDataField

The primitives are described completely using ASN.1 in clause D.2.

D.1 Transport control state machine

Transport Control (TC) for the allocation of QoS controlled transport resources is described by a finite state machine as shown in figure 21. The behaviour of TC at each state transition is described.

For transfer of signalling packets not requiring a QoS controlled transport this state machine does not need to be invoked.

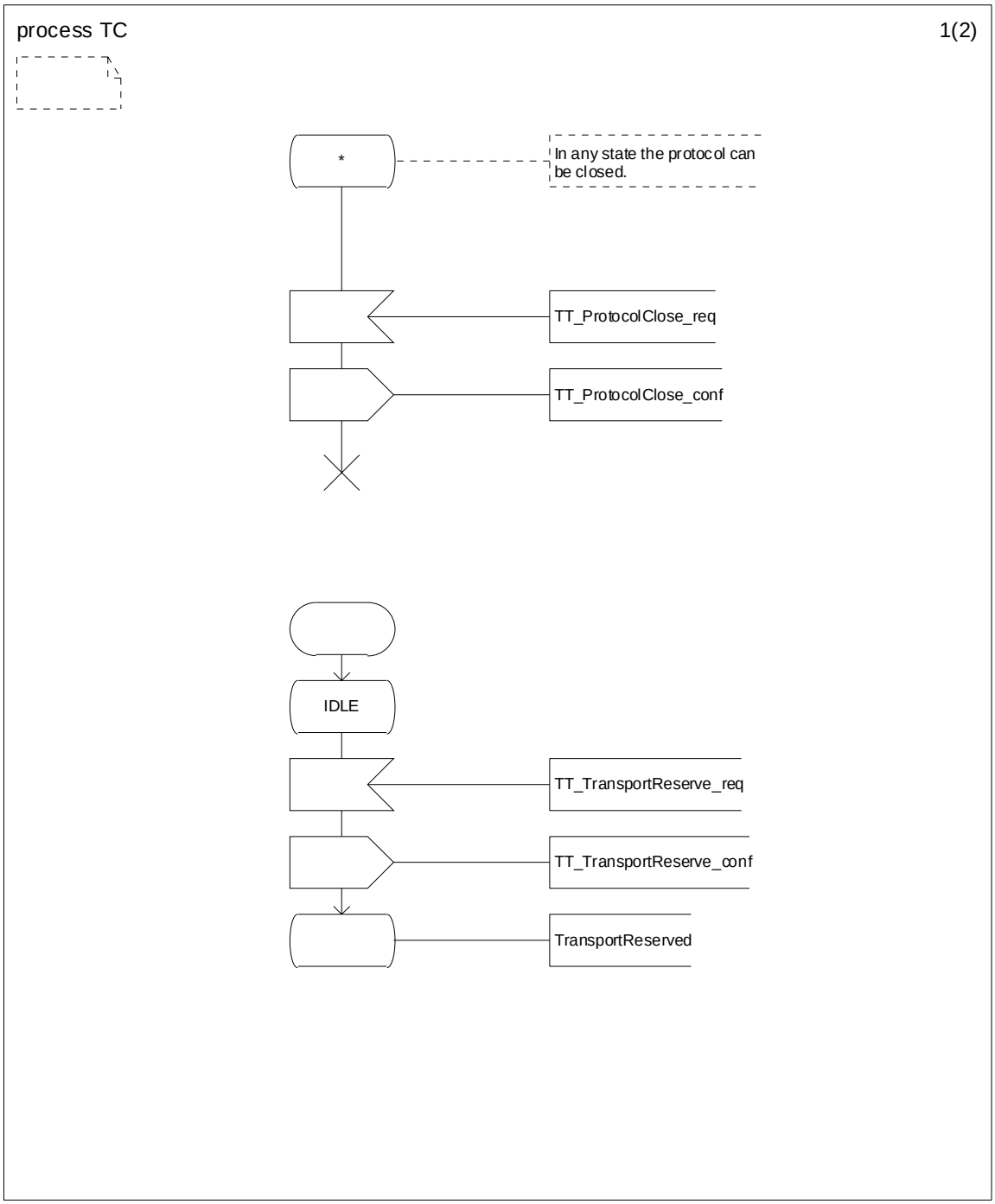


Figure 21: State transitions for TransportControl

Table 17: States of TC

State	Transition message	Next state	Description of behaviour
IDLE	TT_TransportReserve_req	TransportReserved	Clause D.1.1
TransportReserved	TT_TransportAssign_req TT_TransportRelease_req	TransportActive IDLE	Clause D.1.2
TransportActive	TT_TransportRelease_req	IDLE	Clause D.1.3
NOTE: Messages which do not result in a change of state are not indicated in this table.			

D.1.1 Behaviour in IDLE state

IDLE state is entered when the protocol has been opened using the TT_ProtocolOpen_req/conf exchange. IDLE state applies to each protocol. When in IDLE state no reservations of transport resource for QoS enabled traffic, and no activations of transport resources for QoS enabled traffic, shall be in force.

On receipt of TT_TransportReserve_req the TC process shall reserve resources appropriate to the request, confirm the reservation using the TT_TransportReserve_conf primitive and move to state TransportReserved.

The state machine shall be closed when the protocol is closed using the TT_ProtocolClose_req/conf exchange.

D.1.2 Behaviour in TransportReserved State

On receipt of TT_TransportAssign_req the TC process shall confirm the reservation of resources identified and confirm the assignment using the TT_TransportAssign_conf primitive and move to state TransportActive.

D.1.3 Behaviour in TransportActive State

When in TransportActive state data packets for QoS can be transported.

In order to send a frame of data from A (in the application plane) to B (in the application plane) the message sequence chart in figure 22 shows the interaction of primitives.

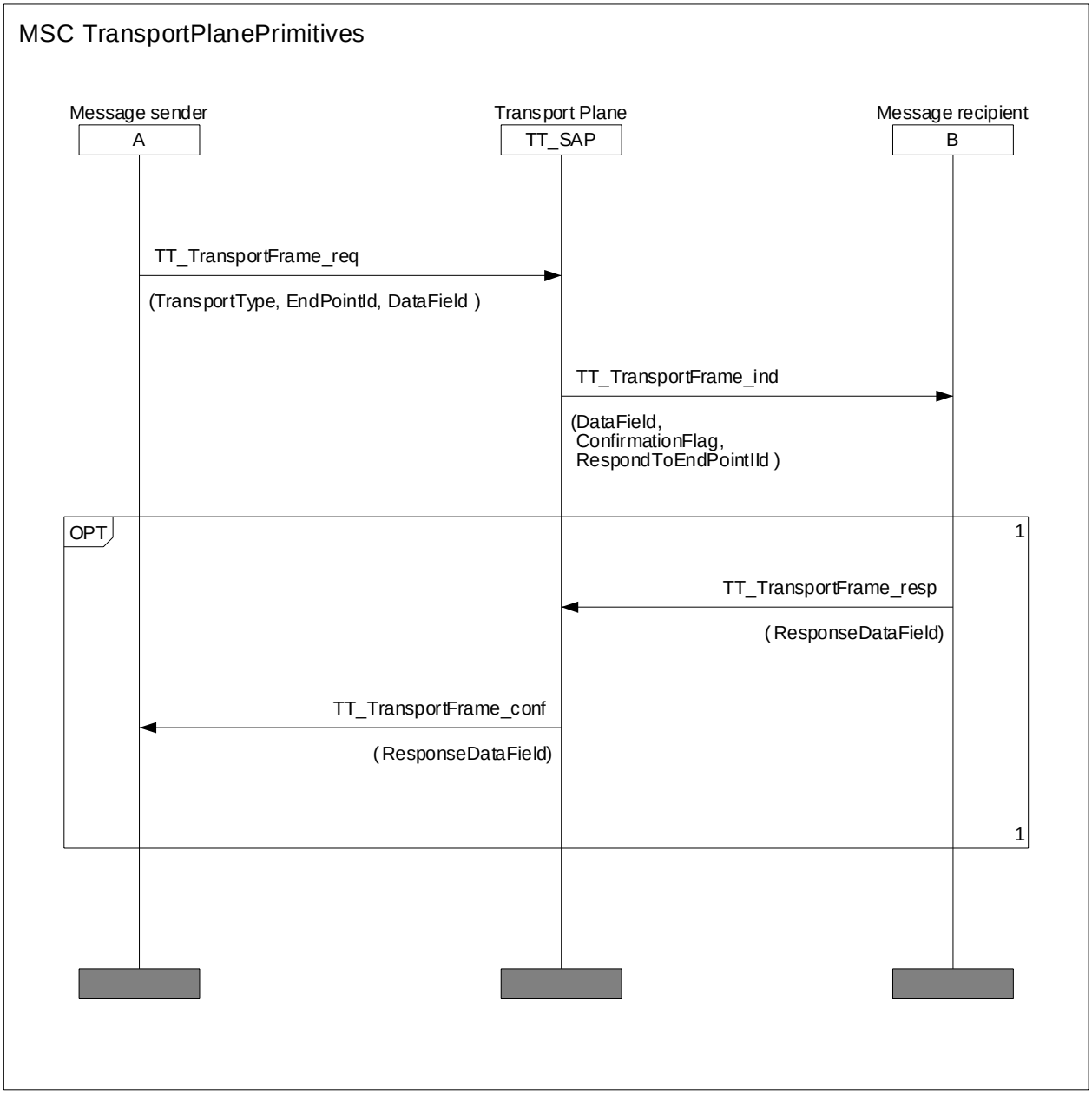


Figure 22: MSC of frame transfer from application plane to transport plane

In figure 22 the response and confirmation flows are shown as optional. The option is set true for a confirmed data transfer, false otherwise.

In figure 24 a similar MSC is shown for the transport reservation request. Typically this will be used during a call setup to prepare a QoS controlled channel for the speech to use during the active phase of the call. In IP media cases the path will be uni-directional and the B-party will need to make a similar reservation.

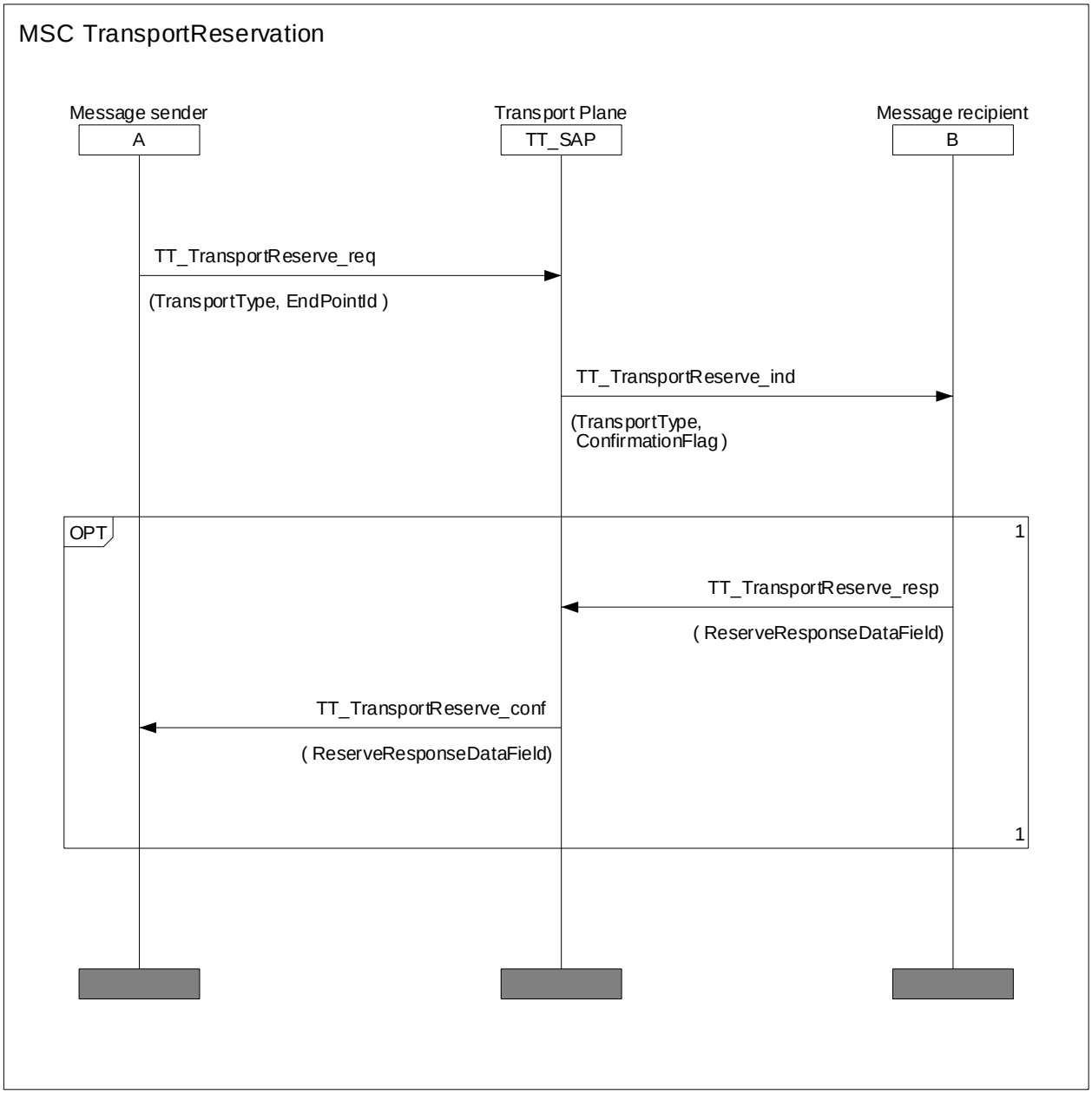


Figure 23: MSC of transport reservation from application plane to transport plane

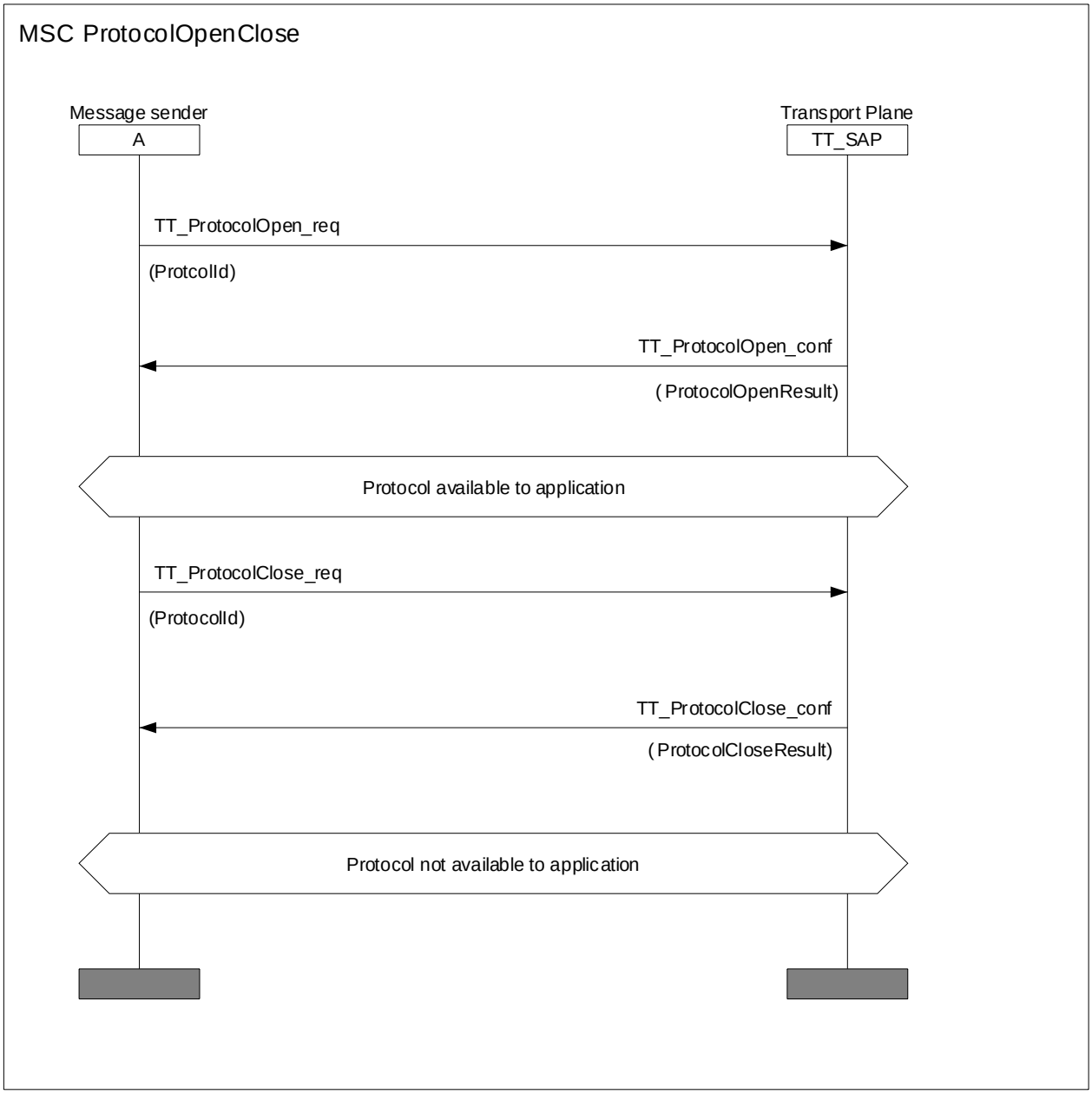


Figure 24: MSC of application plane requesting transport plane to open or close a protocol

D.2 Data definitions (ASN.1)

This clause contains the data definitions for information flows visible at reference point T.

```
TIPHONr3-RefPtT DEFINITIONS ::=

BEGIN

TransportReserveRequestType ::= SET OF TransportDescriptorType

TransportReserveConfirmType ::= SET OF TDTWithHandleType

TDTWithHandleType ::= SEQUENCE
{
    handle      TransportHandleType,
    transport   TransportDescriptorType
}

TransportHandleType ::= VisibleString

TransportAssignType ::= SET OF TransportHandleType

TIPHONErrorType ::= SEQUENCE
{
    source      ENUMERATED
                {
                    callControl,
                    bearerControl,
                    mediaControl,
                    transportControl,
                    …
                },
    severity    ENUMERATED
                {
                    fatalError,
                    warning,
                    information
                },
    reason      ENUMERATED
                {
                    invalid,
                    not_Supported,
                    unavailable,
                    does_not_exist,
                    insufficient_resources,
                    …
                },
    diagnostic  TIPHONErrorDiagnosticType OPTIONAL,
    freeText    VisibleString OPTIONAL,
    embeddedError   TIPHONErrorType OPTIONAL
}

END
```

Annex E (normative): Implementation Conformance Statement Proforma Cover

E.1 Guidance for completing the PICS proforma

E.1.1 Purposes and structure

The PICS proforma cover material is subdivided into clauses for the following categories of information:

- Guidance for completing the PICS proforma;
- Identification of the implementation;
- Identification of the protocol; and
- Global statement of conformance.

E.1.2 Abbreviations and conventions

The PICS proforma contained in this annex is comprised of information in tabular form.

Item column

The item column contains a number, which identifies the item in the table.

Item description column

The item description column describes in free text each respective item (for example parameters, timers, etc.). It implicitly means "is < item description > supported by the implementation?"

Status column

The following notations, defined in ISO/IEC 9646-7, are used for the status column:

M or m	mandatory - the capability is required to be supported;
O or o	optional - the capability may be supported or not;
N/A or n/a	not applicable - in the given context, it is impossible to use the capability;
X or x	prohibited (excluded) - there is a requirement not to use this capability in the given context;
O.i or o.i	qualified optional - for mutually exclusive or selectable options from a set. "i" is an integer which identifies a unique group of related optional items and the logic of their selection which is defined immediately following the table;
Ci.j or ci.j	conditional - the requirement on the capability ("m", "o", "x" or "n/a") depends on the support of other optional or conditional items. "i" is an integer identifying the table and "j" is an integer sequentially allocated inside the table. Ci.j forms a unique conditional status expression which is defined immediately following the table;
I or i	irrelevant (out-of-scope) - the requirement on the capability is outside the scope of the reference specification. No answer is requested from the supplier.

Reference column

The reference column makes reference to the referring clause in the reference document except where explicitly stated otherwise.

Support column

The support column shall be filled in by the supplier of the implementation. The following common notations, defined in ISO/IEC 9646-7, are used for the support column:

Y or y	supported by the implementation;
N or n	not supported by the implementation;
N/A or n/a	no answer required (allowed only if the status is n/a, directly or after evaluation of a conditional status).

If this PICS proforma is completed in order to describe a multiple-profile support in a system, it is necessary to be able to answer that a capability is supported for one profile and not supported for another. In that case, the supplier shall enter the unique reference to a conditional expression, preceded by "?" (For example ?3). This expression shall be given in the space for comments provided at the bottom of the table. It uses predicates defined in the SCS, each of which refers to a single profile and which takes the value TRUE if and only if that profile is to be used.

EXAMPLE: ?3: IF prof1 THEN Y ELSE N.

It is also possible to provide a comment to an answer in the space provided at the bottom of the table.

NOTE: As stated in ISO/IEC 9646-7, support for a received PDU requires the ability to parse all valid parameters of that PDU. Supporting a PDU while having no ability to parse a valid parameter is non-conformant. Support for a parameter on a PDU means that the semantics of that parameter are supported.

Values allowed column

The values allowed column contains the type, the list, the range, or the length of values allowed. The following notations are used:

Range of values:	< min value > .. < max value >:	Example: 5 .. 20.
List of values:	< value1 >, < value2 >, ..., < valueN >:	Example: 2, 4, 6, 8, 9; Example: '1101'B, '1011'B, '1111'B; Example: '0A'H, '34'H, '2F'H.
List of named values:	< name1 >(< val1 >), < name2 >(< val2 >), ..., < nameN >(< valN >:	Example: reject(1), accept(2).
Length:	size (< min size > .. < max size >):	Example: size (1 .. 8).

Values supported column

The values supported column shall be filled in by the supplier of the implementation. In this column, the values or the ranges of values supported by the implementation shall be indicated.

References to items

For each possible item answer (answer in the support column) within the PICS proforma a unique reference exists, used, for example, in the conditional expressions. It is defined as the table identifier, followed by a solidus character "/", followed by the item number in the table. If there is more than one support column in a table, the columns are discriminated by letters (a, b, etc.), respectively.

EXAMPLE 1: A.5/4 is the reference to the answer of item 4 in table 5 of annex A.

EXAMPLE 2: A.6/3b is the reference to the second answer (i.e. in the second support column) of item 3 in table 6 of annex A.

Prerequisite line

A prerequisite line takes the form: Prerequisite: < predicate >.

A prerequisite line after a clause or before a table header indicates that the whole clause or the whole table is not required to be completed if the predicate is FALSE.

E.1.3 Instructions for completing the PICS proforma

The supplier of the implementation shall complete the PICS proforma in each of the spaces provided. In particular, an explicit answer shall be entered, in each of the support or supported column boxes provided, using the notation described in clause E.1.2.

If necessary, the supplier may provide additional comments in space at the bottom of the tables, or separately on sheets of paper.

More detailed instructions are given at the beginning of the different clauses of the PICS proforma.

E.2 Identification of the implementation

Identification of the Implementation Under Test (IUT) and the system in which it resides (the System Under Test (SUT)) should be filled in so as to provide as much detail as possible regarding version numbers and configuration options.

The product supplier information and client information should both be filled in if they are different.

A person who can answer queries regarding information supplied in the PICS should be named as the contact person.

E.2.1 Date of the statement

.....

E.2.2 Implementation Under Test (IUT) identification

IUT name:

.....

.....

IUT version:

.....

E.2.3 System Under Test (SUT) identification

SUT name:

.....

.....

Hardware configuration:

.....

.....

.....

Operating system:

.....

E.2.4 Product supplier

Name:

.....

Address:

.....

.....

.....

Telephone number:

.....

Facsimile number:

.....

E-mail address:

.....

Additional information:

.....

.....

.....

E.2.5 Client (if different from product supplier)

Name:

.....

Address:

.....

.....

.....

Telephone number:

.....

Facsimile number:

.....

E-mail address:

.....

Additional information:

.....

.....

E.2.6 PICS contact person

(A person to contact if there are any queries concerning the content of the PICS)

Name:

.....

Telephone number:

.....

Facsimile number:

.....

E-mail address:

.....

Additional information:

.....

.....

Annex F (informative): Bibliography

ITU-T Recommendation X.680: "Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation".

ITU-T Recommendation Z.100: "Specification and description language (SDL)".

ITU-T Recommendation Z.120: "Message Sequence Chart (MSC)".

ETSI TS 101 884: "Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON) Release 3; Technology Mapping; Implementation of TIPHON architecture using SIP".

History

Document history		
V1.1.1	May 2002	Publication