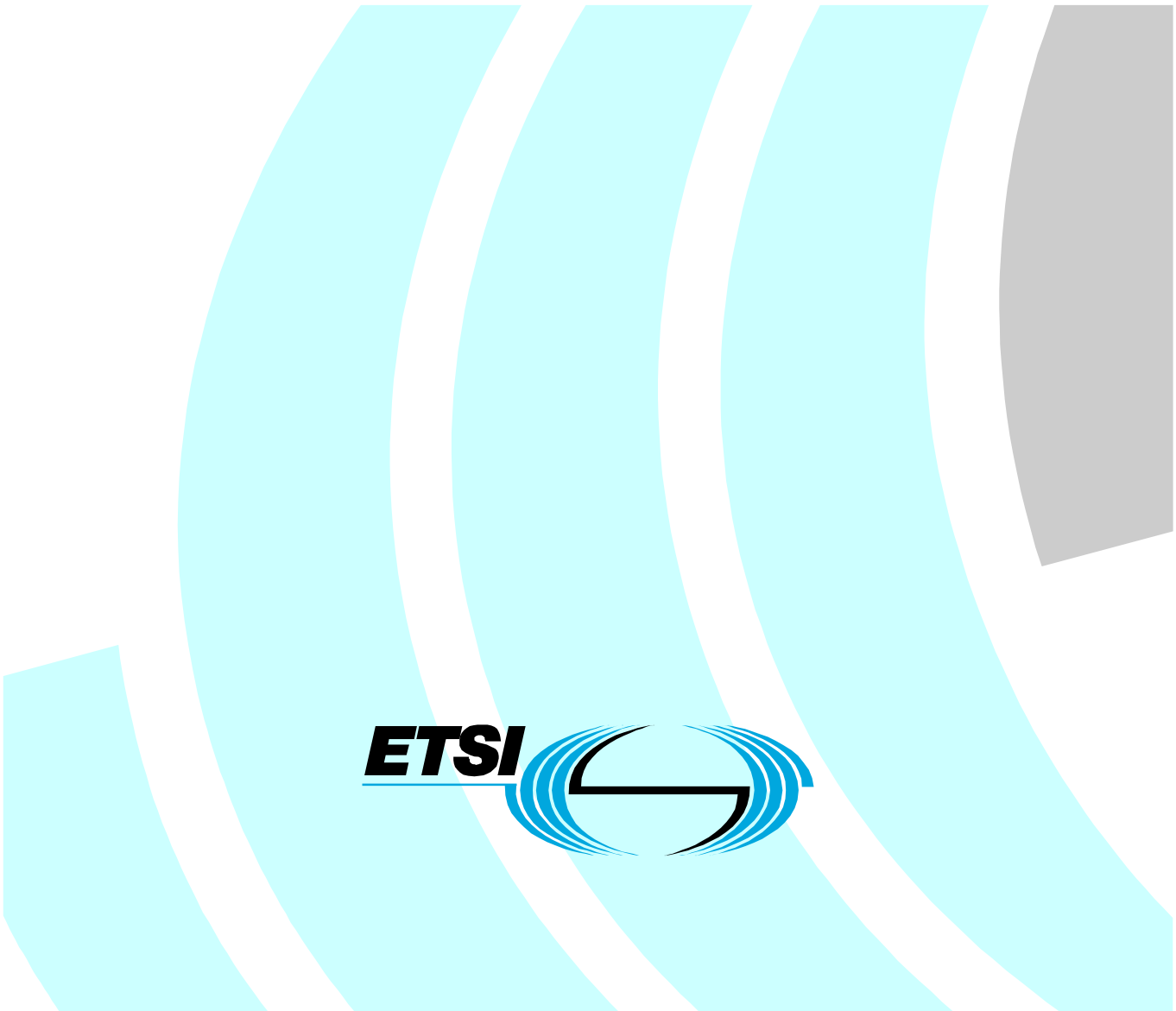
ETSI TS 101 882-3 V4.1.1 (2003-11)

Technical Specification

Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON) Release 4; Protocol Framework Definition; Part 3: TIPHON Simple Call service meta-protocol definition



Reference
RTS/TIPHON-03016-3R4

Keywords
interface, IP, meta-protocol, protocol, service

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).

The present document is part 3 of a multi-part deliverable. Full details of the entire series can be found in TS 101 882-1 [1].

1 Scope

The present document defines the stage 1 and stage 2 requirements (as defined by ITU-T Recommendation I.130 [4]) for the simple call service required by TIPHON.

The simple call service defined, refers to a single media call only.

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.

Referenced documents which are not found to be publicly available in the expected location might be found at http://docbox.etsi.org/Reference.

[1] ETSI TS 101 882-1: "Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON) Release 4; Protocol Framework Definition; Part 1: Meta-protocol design rules, development method, and mapping guideline".

[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 4; Service Capability Definition; Service Capabilities for TIPHON Release 4".

[4] ITU-T Recommendation I.130: "Method for the characterization of telecommunications services supported by an ISDN and network capabilities of an ISDN".

[5] ITU-T Recommendation Z.100 (1996): "Specification and description language (SDL) with corrigendum 1".

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

[7] ETSI TS 101 882-4: "Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON) Release 4; Protocol Framework Definition; part 4: Media control meta protocol".

3 Definitions and abbreviations

3.1 Definitions

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

TIPHON system: collection of inter-working communication networks in which there exists at least one TIPHON gateway connecting two of the networks together

user: entity that uses telecommunication services offered by a TIPHON system

3.2 Abbreviations

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

CFE	simple Call Functional Entity
FE	Functional Entity
QoS	Quality of Service
SDL	Specification and Description Language
SpoA	Service point of Attachment

4 TIPHON simple call service

4.1 Description

A TIPHON simple call is the means by which a temporary logical association is established (and subsequently cleared) between two or more users within a TIPHON system for the purpose of conveying information. The meta-protocol carries sufficient information to allow a bearer to be established if required.

4.2 Procedures

4.2.1 Provision/withdrawal

TIPHON Simple Call is available on a per-name basis to all subscribers to a TIPHON system.

4.2.2 Normal procedures

4.2.2.1 Activation/deactivation

TIPHON simple call shall be permanently activated.

4.2.2.2 Invocation and operation

TIPHON simple call shall be invoked by the calling user (party A) requesting a connection of defined parameters to a second party (party B).

Party A may indicate a preferred carrier of the connection at invocation.

Party A may indicate its own name at invocation to allow the called party to identify the calling party.

Either party to the TIPHON simple call or the serving network shall be able to remove the temporary logical association formed on invocation of the service.

During call set-up transcoding between different codecs may be performed.

4.2.3 Exceptional procedures

If the network cannot complete the connection to party B then party A shall be notified.

If any of the following failure conditions are detected within a domain, the association shall be removed in each domain involved and any associated resource released. The domain that detects the failure shall supply a reason for failure to each other domain. Failure conditions:

- No route found to called party;

- Requested policy not supported;
- Required transport resources not available;
- Reserved media and transport resources released due to time out;
- No compatible codec supported by called party;
- Called user busy (number of simultaneous calls exceeded);
- Call release before call set-up completed.

4.3 Interaction with other services or service capabilities

The following list of service capability interactions include only the service capabilities which are not used for the definition of the simple call service.

4.3.1 Call class service capabilities

4.3.1.1 Redirect

No interaction.

4.3.1.2 SetPriority

No interaction.

4.3.1.3 Join

No interaction.

4.3.1.4 Interrogate

No interaction.

4.3.1.5 SetCondition

No interaction.

4.3.1.6 ClearCondition

No interaction.

4.3.1.7 Park

No interaction.

4.3.1.8 Retrieve

No interaction.

4.3.1.9 LocationDelivery

No interaction.

4.3.2 Bearer class service capabilities

4.3.2.1 Join

No interaction.

4.3.2.2 Modify

No interaction.

4.3.2.3 SetCondition

No interaction.

4.3.2.4 ClearCondition

No interaction.

4.3.3 Profile class service capabilities

4.3.3.1 Register

No interaction.

NOTE: The QOS to be used for subsequent calls by the registered user may form part of the information supplied at registration.

4.3.3.2 Attach

No interaction.

4.3.3.3 Deregister

No interaction.

4.3.3.4 Detach

No interaction.

4.3.3.5 Authenticate

No interaction.

4.3.3.6 Authorize

No interaction.

4.3.3.7 Transfer

No interaction.

4.3.3.8 SetStatus

No interaction.

4.3.3.9 SetCondition

No interaction.

4.3.3.10 ClearCondition

No interaction.

4.4 Service capabilities used in service definition

Although not explicitly identified, aspects of the following service capabilities are used in definition of the simple call service:

Call class service capabilities:

- Setup;
- IdentityDelivery;
- Cleardown;
- Route.

Bearer class service capabilities:

- Create;
- Delete.

Profile class service capabilities:

- GetStatus.

The TIPHON Release 4 service capabilities are defined in TS 101 878 [3].

4.5 Overall behaviour

The UML activity diagram in figure 1 shows the dynamic simple call signalling for a TIPHON system providing simple call service.

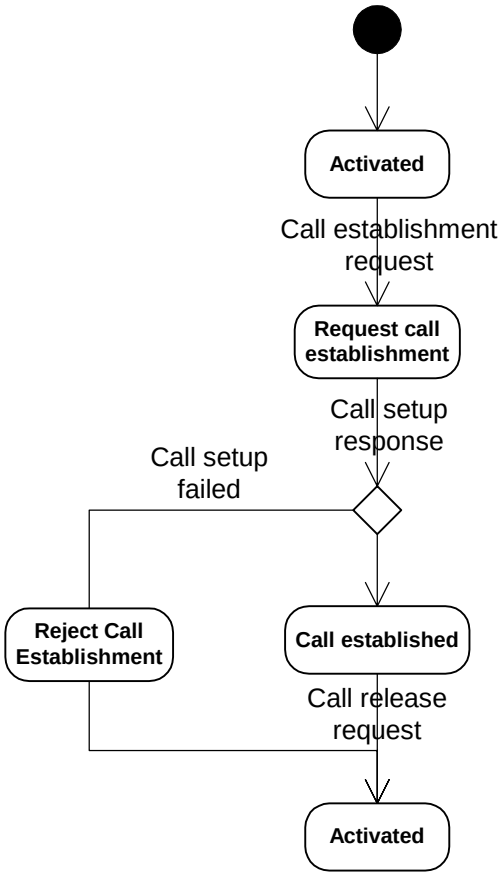


Figure 1: Overall behaviour of simple call service signalling

NOTE: The service behaviour model in figure 1 defines global states of the simple call service, hence the same state names are not re-used in the behaviour descriptions of the individual functional entities.

5 Functional entity model and information flows

5.1 Functional entity model

5.1.1 Description of model

This functional entity model covers the signalling aspects of call control with call routing but does not include the flow of data in the media path.

The functional entity model uses the following Call Functional Entities (CFE):

- Calling User — The application at the calling user's terminal which instigates the service request;
- $CFE1_{OTC}$ — The originating user service agent in the calling user's terminal that instigates the service request and processes call and bearer functions (see note 3);
- $CFE2_{PE}$ — The serving network policy control function associated with the calling user's service provider;
- $CFE3_{ONC}$ — The originating call and bearer coordination function that is responsible for establishing the call on behalf of the calling user (see note 3);
- $CFE4_{OR}$ — The originating call routing function, providing routing information and number/address translations (see note 1);

- $CFE5_{ONM}$ The originating media coordination function serving the calling user;
- $CFE6_{INC}$ An intervening call and bearer control coordination function (see note 3). This CFE is responsible for establishing the call via the intervening domain;
- $CFE7_{IR}$ An intervening routing function (see note 1);
- $CFE8_{INM}$ An intervening media coordination function;
- $CFE9_{TNC}$ The destination call and bearer coordination function (see note 3) that is responsible for establishing the requested call on behalf of the called user;
- $CFE10_{TNM}$ The destination media coordination function serving the called user;
- $CFE11_{TTC}$ The service agent that processes an incoming call to the called user, processing call and bearer functionality (see note 3);
- $CFE12_{OTM}$ The media control function in the calling user's terminal. (see note 2);
- $CFE13_{TTM}$ The media control function in the called user's terminal. (see note 2);
- Called User The application in the called user's terminal at which the service request is terminated.

NOTE 1: Routing is based upon both the called user's identity and the requested service which shall include QoS parameters.

NOTE 2: The media control functional entities in the originating and terminating terminal are not part of the modelled behaviour in this release. The media control service meta-protocol is defined in TS 101 882-4 [7].

NOTE 3: In this release the call and bearer functionality in the different TIPHON domains is modelled as a single entity. In future releases the call and bearer functionality may be modelled as separate entities.

The following functional relationships also is part of the functional entity model for simple call:

- ra between the Calling User and the Calling User's service agent ($CFE1_{OTC}$);
- rb between the Calling User's service agent ($CFE1_{OTC}$) and the Policy function ($CFE2_{PE}$);
- rc between the Calling User's service agent ($CFE1_{OTC}$) and the originating call and bearer coordination function ($CFE3_{ONC}$);
- rd between the originating call coordination function ($CFE3_{ONC}$) and the originating call routing function ($CFE4_{OR}$);
- re between the originating call coordination function ($CFE3_{ONC}$) and originating media coordination function ($CFE5_{ONM}$);
- rf between the originating call coordination function ($CFE3_{ONC}$) and an intervening call and bearer coordination function ($CFE6_{INC}$);
- rg between an intervening call coordination function ($CFE6_{INC}$) and an intervening routing function ($CFE7_{IR}$);
- rh between an intervening call coordination function ($CFE6_{INC}$) and an intervening media coordination function ($CFE8_{INM}$);
- ri between an intervening call coordination function ($CFE6_{INC}$) and the destination call and bearer coordination function ($CFE9_{TNC}$);

- rj between the destination call coordination function ($CFE9_{TNC}$) and the destination media coordination function ($CFE10_{TNM}$);
- rk between the destination call coordination function ($CFE9_{TNC}$) and the service agent that processes an incoming call to the called user ($CFE11_{TTC}$);
- rl between the called user's service agent function ($CFE11_{TTC}$) and the Called User;
- rm between the Calling User's service agent ($CFE1_{OTC}$) and the media control function in the originating user's terminal (see note);
- rn between the called user's service agent function ($CFE11_{TTC}$) and the media control function in the called user's terminal (see note).

NOTE: The relationship between the call control FE and the media control FE in the originating and terminating terminal is not further defined in this release.

The TIPHON simple call functional entity model is shown in figure 2.

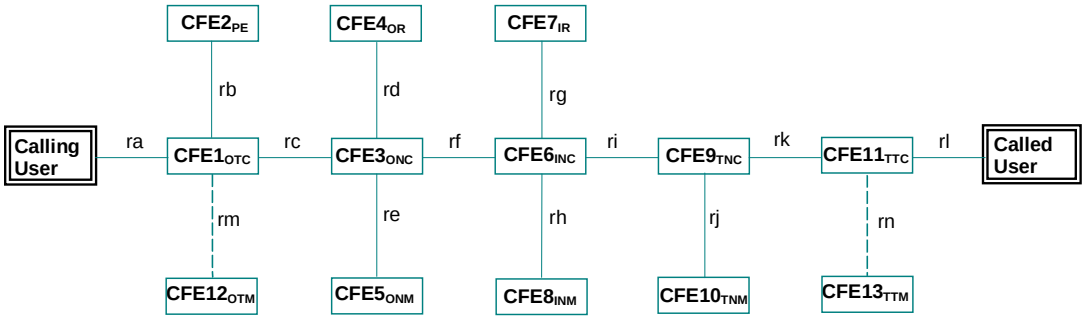


Figure 2: Functional Entity Model for TIPHON simple Call

5.1.2 Description of functional entities

5.1.2.1 Calling User

The Calling User functional entity acts on behalf of an end user to request the establishment of a simple call.

5.1.2.2 Calling User's service agent, $CFE1_{OTC}$

On receipt of a call set-up request from the Calling User, $CFE1_{OTC}$ requests $CFE2_{PE}$ if the ticket is valid and the current policy permits the requested QoS to be used in the call. If the requested QoS is permitted $CFE1_{OTC}$ forwards the call set-up request in a Orig`CallSetup` to the simple call service SpoA (identified at Registration).

5.1.2.3 The policy control function associated with the calling user's service provider, $CFE2_{PE}$

This FE checks if the ticket is valid and checks its service database to determine if requested service parameters are permitted in the requested call, e.g. QoS parameters.

5.1.2.4 Originating network call coordination function, $CFE3_{ONC}$

This FE requests the originating routing function for destination address information and requests the provision of appropriate media resource from the originating media control function $CFE5_{ONM}$ and then passes the call set-up request to the next call coordination function, $CFE6_{INC}$ or $CFE9_{TNC}$.

5.1.2.5 Originating network routing function, $CFE4_{OR}$

$CFE4_{OR}$ checks the provided destination address upon request from the originating call coordination function and returns information on the completeness of the called address and next hop information.

5.1.2.6 Originating network media coordination function, $CFE5_{ONM}$

This FE attempts to establish a media connection with the requested capabilities between the indicated incoming user access point and an outgoing network access point providing a media connection between the calling and called user, possible via intervening media control functions.

5.1.2.7 Intervening network call coordination function, $CFE6_{INC}$

If present, this CFE request an intervening routing function for destination address information and requests the provision of appropriate media resource from an intervening media control function and then passes the call set-up request to the next call coordination function, $CFE6_{INC}$ or $CFE9_{TNC}$.

5.1.2.8 Intervening network routing function, $CFE7_{IR}$

If present, this CFE checks a provided destination address upon request from an intervening call coordination function and returns information on the completeness of the called address and next hop information.

5.1.2.9 Intervening network media coordination function, $CFE8_{INM}$

If present, this CFE attempts to establish a media connection between the indicated incoming user access point and an outgoing network access point providing an appropriate connection between the calling and called user, possible via intervening media control functions.

5.1.2.10 Destination network call coordination function, $CFE9_{TNC}$

This CFE requests the destination routing function for called user address information and requests the provision of appropriate media resource from the destination network media control function and then passes the call set-up request to the called user service agent.

5.1.2.11 Destination network media coordination function, $CFE10_{TNM}$

The CFE attempts to establish a media connection between the indicated network access point and the called user access point providing an appropriate connection between the calling and called user, possible via intervening media control functions.

5.1.2.12 Called user's service agent, $CFE11_{TTC}$

This CFE informs the end user that a call set-up request is being made and conveys the Called User response to the destination network call control function, $CFE9_{TNC}$.

5.1.2.13 Called User

The Called User functional entity acts on behalf of the called end user to respond to a simple call set-up.

5.2 Information flows

5.2.1 Definition of information flows

5.2.1.1 Relationship ra

5.2.1.1.1 TCC_OrigCallSetup

`TCC_OrigCallSetup` is a confirmed information flow that shall be sent across relationship ra from the calling user to $CFE1_{OTC}$ to indicate a request for a simple call establishment. Table 1 lists the elements within the `TCC_CallSetup` information flow.

Table 1: Contents of TCC_OrigCallSetup

TCC_OrigCallSetup			
Information element	**Value**	**Request**	**Response**
Call Identifier	Alphanumeric handle		M
Called user ID (see note 1)	TIPHON user name - E.164 number - URL	M	
Calling User ID restriction	- Calling User ID presentation available - Available Unavailable	M	
Calling User ID	- Calling User ID - E.164 number - URL - DisplayName	O (see note 2)	
Operator selection	- Prefix dial string - Operator Identifier	O (see note 3)	
QoS Service Class	- Predefined - TIPHON QoS class - 3 Best - 2H High - 2M Medium - 2A Acceptable - 1 Best effort - Non-standardized QoS class	M	
Traffic descriptor	- Media peak rate - Maximum media frame size	M	
Service Offer Ticket	Service Ticket	M	
Codec	- List of possible codecs - Codec type	M	O (see note 4, see note 5)
Transcode count	Number of codec transcodings	M	O (see note 6)
Previous Domain Egress point	Network specific address	M	
Next Domain Egress point	Network specific address		O (see note 7)
Result	- Call Established - with requested QoS - Rejection cause - Transport not available - Requested QoS not available - Called user unknown - No compatible codec available - Policy Rejection - Called user busy		M

NOTE 1: Shall be either an E.164 number or a URL.
NOTE 2: Shall be present if "Calling User ID restriction" information element is set to value "available".
NOTE 3: If present, the call shall be routed via the network of the specified operator, otherwise the call shall be established via the calling user's default operator.
NOTE 4: The list of codecs shall be limited to a single entry in the response.
NOTE 5: This element shall be included if the Result element is set to "Call Established".
NOTE 6: This element shall be included if the Result element is set to "No compatible codec available".
NOTE 7: This element shall be included only if required to establish a dynamic address relationship between network functional groups.

5.2.1.1.2 TCC_CallRelease

`TCC_CallRelease` is a confirmed information flow that shall be sent across relationship rb between calling user and $CFE1_{OTC}$ when a call is terminated.

Table 2: Contents of TCC_CallRelease

TCC_CallRelease			
Information element	**Value**	**Request**	**Response**
Call Identifier	Alphanumeric handle	O (see note 1)	M
CauseCode	- userInitated - networkInitiated	M	
Result	- successful - failed		M
NOTE 1: May be omitted when call release is invoked before call setup is complete.			

5.2.1.1.3 TCC_CallAlerting

`TCC_CallAlerting` is an unconfirmed information flow that shall be sent across relationship rb from $CFE1_{OTC}$ to the calling user when the called party is being alerted to the call.

Table 3: Contents of TCC_CallAlerting

TCC_CallAlerting		
Information element	**Value**	**Request**
Call Identifier	Alphanumeric handle	M

5.2.1.2 Relationship rb

5.2.1.2.1 ServingNWPolicy

`ServingNWPolicy` is a confirmed information flow that shall be sent across relationship rb from $CFE1_{OTC}$ to $CFE2_{PE}$ to request permission for a new call establishment with a specific end-to-end QoS. Table 4 lists the elements within the `ServingNWPolicy` information flow.

NOTE: The "Transport QoS parameters" information element values may be derived from the specified QoS service class in the setup request or the default QoS class used, if the QoS service class is omitted in the setup request.

Table 4: Contents of ServingNWPolicy

ServingNWPolicy			
Information element	**Value**	**Request**	**Response**
Calling user ID	TIPHON user name	M	
Called user ID	TIPHON user name	M	
Transport QoS parameters	- Maximum delay - Maximum packet delay variation - Maximum mean packet loss	O (see note 1)	
QoS Service Class	- Predefined - TIPHON QoS class - 3 Best - 2H High - 2M Medium - 2A Acceptable - 1 Best effort - Non-standardized QoS class	O (see note 1)	
Service Offer Ticket	Service Ticket	M	
Result	- Call permitted - Rejection cause - Invalid ticket - Service not subscribed to - Service currently not available		M
NOTE 1: One of these information elements shall be provided.			

5.2.1.3 Relationship rc

5.2.1.3.1 CallSetup

`CallSetup` is a confirmed information flow that shall be sent across relationship rc from the calling user service agent to the Originating call coordinating CFE. Table 5 lists the elements within the `CallSetup` information flow.

NOTE: A bearer for media flow may be of three types: unidirectional, bi-directional symmetric (default for simple call), and bi-directional asymmetric.

Table 5: Contents of CallSetup

CallSetup			
Information element	**Value**	**Request**	**Response**
Call Identifier	Alphanumeric handle	M (see note 1)	M
Calling User ID restriction	- Calling User ID presentation available - Available - Unavailable	M	
Calling User ID	- Calling User ID - E.164 number - URL - DisplayName	O (see note 2)	
Called user ID	TIPHON user name	M	
Transport QoS parameters	- Maximum delay - Maximum packet delay variation - Maximum mean packet loss	M	
Transport parameters qualifier	- Transport QoS parameters indicate total remaining budget - Transport QoS parameters indicate budget available per domain	M	
Traffic descriptor	- Media peak rate - Maximum media frame size	M	
Codec	- List of possible codecs - Codec type	M	O (see note 3, see note 4)
Transcode count	Number of codec transcodings	M	O (see note 7)
Calling User Access Point	Network specific address	O (see note 5)	
Destination service domain	Domain address	O (see note 6)	
Routing number	Domain address	O (see note 6)	
Previous Domain Egress point	Network specific address	M	
Next Domain Egress point	Network specific address		O (see note 8)
Result	- Call established - Rejection cause - Transport not available - Requested QoS not available - Called user unknown - No compatible codec available - Called user busy		M

NOTE 1: A temporary Call Identifier value may be used in the call setup request.
NOTE 2: Shall be present if "Calling User ID restriction" information element is set to value "available".
NOTE 3: The list of codecs shall be limited to a single entry in the response.
NOTE 4: This element shall be included if the result of the request is "Call established".
NOTE 5: This information element may provided to support the routing decision.
NOTE 6: This element is available only if by some means the next or destination network address can be determined initially. If so, this information may simplify route calculations in other functional groups.
NOTE 7: This element shall be included if the result of the request is "No compatible codec available".
NOTE 8: This element shall be included only if required to establish a dynamic address relationship between network functional groups.

5.2.1.3.2 CallRelease

`CallRelease` is a confirmed information flow that shall be sent across relationship rc between the calling user service agent to the Originating call coordinating CFE.

Table 6: Contents of CallRelease

CallRelease			
Information element	**Value**	**Request**	**Response**
Call Identifier	Alphanumeric handle	M	M
CauseCode	- userInitated - networkInitiated	M	
Result	- successful - failed		M

5.2.1.3.3 CallAlerting

`CallAlerting` is an unconfirmed information flow that shall be sent across relationship rc from the Originating call coordinating CFE to calling user service agent $CFE1_{OTC}$ when the called party is being alerted to the call.

Table 7: Contents of CallAlerting

CallAlerting		
Information element	**Value**	**Request**
Call Identifier	Alphanumeric handle	M

5.2.1.4 Relationship rd, rg

5.2.1.4.1 CallRoute

`CallRoute` is a confirmed information flow that shall be sent across relationship rd and rg from a call coordination to a routing function to indicate a request for address and routing information. Table 8 lists the elements within the `CallRoute` information flow.

Table 8: Contents of CallRoute

CallRoute			
Information element	Value	Request	Response
Call Identifier	Alphanumeric handle	M	M
Called user ID	TIPHON user name	M	
Calling User ID restriction	- Calling User ID presentation available Available Unavailable	M	
Calling user ID	TIPHON user name	O (see note 1)	
Routing number information	Domain address	O (see note 2)	O (see note 4)
Calling User Access Point	Network specific address	O (see note 3)	
Destination service domain	Domain address	O (see note 2)	
Transport QoS parameters	- Maximum delay - Maximum packet delay variation - Maximum mean packet loss	M	
Transport parameters qualifier	- Transport QoS parameters indicate total remaining budget - Transport QoS parameters indicate budget available per domain	M	
Traffic descriptor	- Media peak rate - Maximum media frame size	M	
Codec	- List of possible codecs - Codec type	M	
Next network node list	- Domain Address list		O (see note 5)
Result	- Routing information - No compliant routing information found - Next node		M

NOTE 1: Shall be present if Calling user Id presentation is available.
NOTE 2: Any routing and destination domain information available shall be passed to the routing functional to support the routing process.
NOTE 3: The "Calling User Access Point" may be provided to support the routing decision.
NOTE 4: Updated routing information may result from the routing process.
NOTE 5: Shall be present if Result is "Next node" and shall contain a non-empty ordered list of possible next node domain addresses.

5.2.1.5 Relationship re, rh, rj

5.2.1.5.1 MediaReservation

`MediaReservation` is a confirmed information flow that shall be sent across relationships re, rh, and rj to request the reservation of a media flow towards the called user's address. Table 9 lists the elements within the `MediaReservation` information flow.

Table 9: Contents of MediaReservation

MediaReservation			
Information element	**Value**	**Request**	**Response**
BearerId	Alphanumeric "handle"	M	M
QoS Parameters Qualifier	- QoS parameters indicate total remaining budget - QoS parameters indicate budget available per domain	O (see note 4)	O (see note 1)
Media descriptor	CodecDescr { CodecType, CodecParameters, SilenceSuppression, EchoCancelling, MediaPeakRate, MaxMediaFrameSize }, [CodecDescr {…}] Priority	M (see note 2)	
MediaId	Alphanumeric "handle"		O (see note 3)
QoS Parameters	- PacketTransmissionRate - PacketLossRate - Jitter - Integrity - TransitDelay	O (see note 4)	O (see note 1, see note 3)
PreviousDomainEgressAddress (forward path)	Network specific address	M	
NextDomainAddress	Network domain address	O (see note 5)	
UserDomainAddress	Network specific address	O (see note 5)	
Egress Point (forward path)	Network specific address		O (see note 3)

Result	- Resource reserved
	- Rejection cause
	  - Media resource not available
	  - Media resource not supported
			M

NOTE 1: This information element shall be included if the value of the transport parameters qualifier in the request is "QoS parameters indicate total remaining budget". NOTE 2: The media descriptor specifies the stronger requirements from the list of proposed codecs. Selection of the codec is done by the called user, so the actual media resources needed can be determined when media establishment is performed. The optional CodecDescr is present only when transcoding is performed. NOTE 3: Shall be included if information element is "Resource reserved". NOTE 4: Mandatory if QoS is required. NOTE 5: Exactly one of these information elements shall be present.			

In the MediaReservation request, the address parameters, "Previous domain Ingress address" and "Next domain address" shall be specified as inter-network addresses. "User domain address" may be specified as either an intra- or inter-network address. In the MediaReservation response, the address parameter "Egress point" shall be specified as an inter-network-address.

The information element "Next domain address" is derived from the "CallRoute response". The information element "User domain address" is derived from the called user information available in the destination network domain.

5.2.1.5.2 MediaEstablishment

`MediaEstablishment` is a confirmed information flow that shall be sent across relationships re, rh, and rj to request the establishment of a previously reserved bearer and media transport path towards the called user's address. Table 10 lists the elements within the `MediaEstablishment` information flow.

Table 10: Contents of MediaEstablishment

MediaEstablishment			
Information element	**Value**	**Request**	**Response**
BearerId	Alphanumeric "handle"	M	M
MediaId	Alphanumeric "handle"	M	M
Next Domain Egress point (reverse path)	Network specific address	M	
Egress point (reverse path)	Network specific address		O (see note)

Result	- Media allocated
	- Rejection cause
	  - Unable to allocate resource
	  - Resource no longer available
			M

NOTE: Shall be present if Result is "Media allocated".			

5.2.1.5.3 MediaRelease

`MediaRelease` is an unconfirmed information flow that shall be sent across relationships re, rh, and rj to request that a previously reserved bearer and media transport path is released as it is no longer required. Table 11 lists the elements within the `MediaRelease` information flow.

Table 11: Contents of MediaRelease

MediaRelease		
Information element	**Value**	**Request**
Bearer identifier	Alphanumeric "handle"	M
MediaId	Alphanumeric "handle"	O (see note)
NOTE: If the MediaId is not present all media resources associated to the specified BearerId are released.		

5.2.1.6 Relationship rf, ri

5.2.1.6.1 NwCallSetup

`NwCallSetup` is a confirmed information flow that shall be sent across relationship rf and ri from the originating call coordination function to the destination call coordination function or an intervening call coordination function. Table 12 lists the elements within the Nw`CallSetup` information flow.

Table 12: Contents of NwCallSetup

NwCallSetup			
Information element	**Value**	**Request**	**Response**
Call Identifier	Alphanumeric handle	M	M
Calling User ID restriction	- Calling User ID presentation available - Available - Unavailable	M	
Calling User ID	- TIPHON user name	O (see note 1)	
Called user ID	TIPHON user name	M	
Previous Domain Egress point	Network specific address	M	
Next Domain Egress point	Network specific address		O (see note 7)
Bearer identifier	Alphanumeric "handle"	M	
Transport QoS parameters	- Maximum delay - Maximum packet delay variation - Maximum mean packet loss	M	
Transport parameters qualifier	- Transport QoS parameters indicate total remaining budget - Transport QoS parameters indicate budget available per domain	M	
Traffic descriptor	- Media peak rate - Maximum media frame size	M	
Codec	- List of possible codecs - Codec type	M	O (see note 2, see note 3)
Transcode count	Number of codec transcodings	M	O (see note 6)
Destination service domain	Domain address	O (see note 4)	
Calling User Access Point	Network specific address	O (see note 5)	
Routing number	Domain address	O (see note 4)	
Result	- Call established - Rejection cause - Transport not available - Requested QoS not available - Called user unknown - No compatible codec available - Called user busy		M

NOTE 1: Shall be present if Calling user Id presentation is available.
NOTE 2: The list of codecs shall be limited to a single entry in the response.
NOTE 3: This element shall be included if the result of the request is "Call established".
NOTE 4: This element is available only if by some means routing information or destination network domain can be determined. If so, this information may simplify route calculations in other functional groups.
NOTE 5: The "Calling User Access Point" may be provided to support the routing decision.
NOTE 6: This element shall be included if the result of the request is "No compatible codec available".
NOTE 7: This element shall be included only if required to establish a dynamic address relationship between network functional groups.

5.2.1.6.2 NwCallRelease

`NwCallRelease` is a confirmed information flow that shall be sent across relationship rf and ri sent between originating, intervening or destination call coordination FEs.

Table 13: Contents of NWCallRelease

NWCallRelease			
Information element	**Value**	**Request**	**Response**
Call Identifier	Alphanumeric handle	M	M
CauseCode	- userInitated - networkInitiated	M	
Result	- successful - failed		M

5.2.1.6.3 NwCallAlerting

`NwCallAlerting` is an unconfirmed information flow that shall be sent across relationships rf and ri from the destination or an intervening call coordinating CFE to the originating or an intervening call coordination CFE when the called party is being alerted to the call.

Table 14: Contents of NWCallAlerting

NWCallAlerting		
Information element	**Value**	**Request**
Call Identifier	Alphanumeric handle	M

5.2.1.7 Relationship rk

5.2.1.7.1 DestCallSetup

`DestCallSetup` is a confirmed information flow that shall be sent across relationship rk from the destination call coordination function to the called user's service agent. Table 15 lists the elements within the Dest`CallSetup` information flow.

Table 15: Contents of DestCallSetup

DestCallSetup			
Information element	**Value**	**Request**	**Response**
Call Identifier	Alphanumeric handle	M	M
Called user ID	TIPHON user name	M	
Calling User ID	TIPHON user name	O (see note 1)	
Transport QoS parameters	- Maximum delay - Maximum packet delay variation - Maximum mean packet loss	M	
Codec	- List of possible codecs - Codec type	M	O (see note 2, see note 3)
Transcode count	Number of codec transcodings	M	O (see note 4)
Previous Domain Egress point	Network specific address	M	
Next Domain Egress point	Network specific address		O (see note 5)
Result	- Call established - Rejection cause - No compatible codec available - Called user busy		M

NOTE 1: This information element shall be present if the Calling UserId information has been passed to the terminating call control functional entity, and the restriction parameter has value "available".
NOTE 2: The list of codecs shall be limited to a single entry in the response.
NOTE 3: This element shall be included if the result of the request is "Call established".
NOTE 4: This element shall be included if the result of the request is "No compatible codec available".
NOTE 5: This element shall be included only if required to establish a dynamic address relationship between network functional groups.

5.2.1.7.2 CallRelease

`CallRelease` is a confirmed information flow that shall be sent across relationship rk between the destination call coordination function and the called user service agent.

Table 16: Contents of CallRelease

CallRelease			
Information element	**Value**	**Request**	**Response**
Call Identifier	Alphanumeric handle	M	M
CauseCode	- userInitated - networkInitiated	M	
Result	- successful - failed		M

5.2.1.8 Relationship rl

5.2.1.8.1 TCC_DestCallSetup

`TCC_DestCallSetup` is a confirmed information flow that shall be sent across relationship rm from the called user's service agent $CFE11_{TTC}$ to the Called User to indicate an incoming call. Table 17 lists the elements within the `TCC_DestCallSetup` information flow.

Table 17: Content of TCC_DestCallSetup

TCC_DestCallSetup			
Information element	**Value**	**Request**	**Response**
Call Identifier	Alphanumeric handle	M	M
Calling User ID	TIPHON user name	O (see note 1)	
Called user ID	TIPHON user name	M	
Transport QoS parameters	- Maximum delay - Maximum packet delay variation - Maximum mean packet loss	M	
Codec	- List of possible codecs - Codec type	M	O(see note 2)
Transcode count	Number of codec transcodings	M	O (see note 3)
Previous Domain Egress point	Network specific address	M	
Next Domain Egress point	Network specific address		O (see note 4)
Call accept	- Call accepted - Call rejected cause - No compatible codec - Busy		M
NOTE 1: This information element shall be present if the Calling UserId information has been passed to the terminating call user agent. NOTE 2: The Codec element shall be included if the Call accept element is "Call accepted". NOTE 3: This element shall be included if the result of the request is "No compatible codec". NOTE 4: This element shall be included only if required to establish a dynamic address relationship between network functional groups.			

5.2.1.8.2 TCC_CallRelease

`TCC_CallRelease` is a confirmed information flow that shall be sent across relationship rl between called user and $CFE11_{TTC}$ when a call is terminated.

Table 18: Contents of TCC_CallRelease

TCC_CallRelease			
Information element	**Value**	**Request**	**Response**
Call Identifier	Alphanumeric handle	M	M
CauseCode	- userInitated - networkInitiated	M	
Result	- successful - failed		M

5.2.2 Timers

"Reservation Hold Timer": A reservation hold timer is used in the call coordination functional entity to make sure that resource is not held indefinitely.

5.2.3 Information flow sequences

A standard specifying TIPHON meta-protocols for simple call signalling shall provide signalling procedures in support of the information flow sequences specified below. In addition, signalling procedures should be provided to cover other sequences arising from error situations and interactions with other service capabilities.

In the figures, simple call signalling information flows are represented by solid arrows. Within a column representing a simple call signalling functional entity, the numbers refer to functional entity actions listed in clause 5.3.

The following abbreviations are used:

- req request;
- resp response.

5.2.3.1 Normal operation

5.2.3.1.1 Call set-up

Figure 3 illustrates the information flows for successful call set-up.

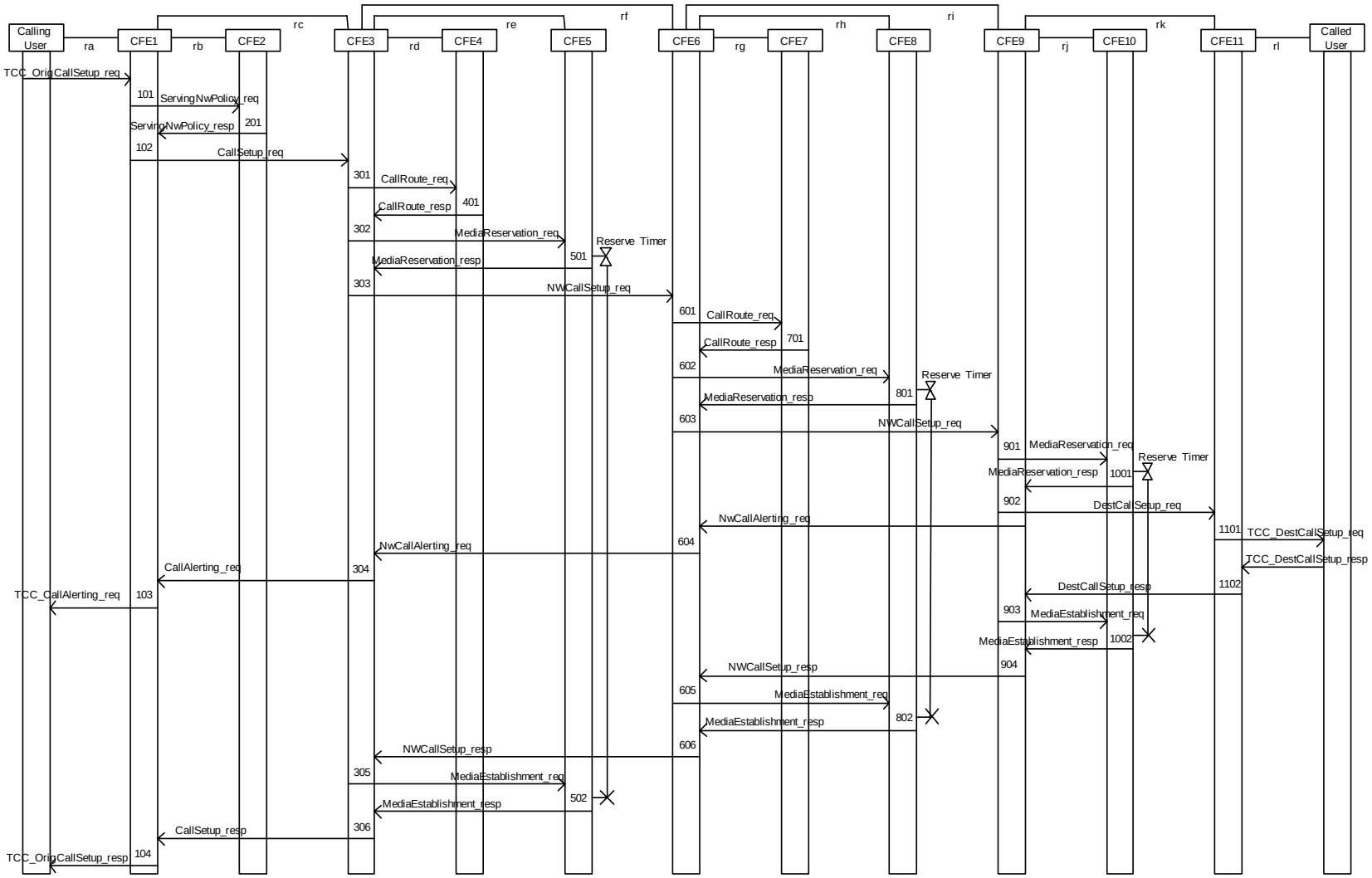


Figure 3: Information flows for successful call set-up

Figure 4 illustrates successful call set-up with transcoding. Other successful call set-up scenarios involving transcoding in the originating or destination service domain are possible. These are not shown.

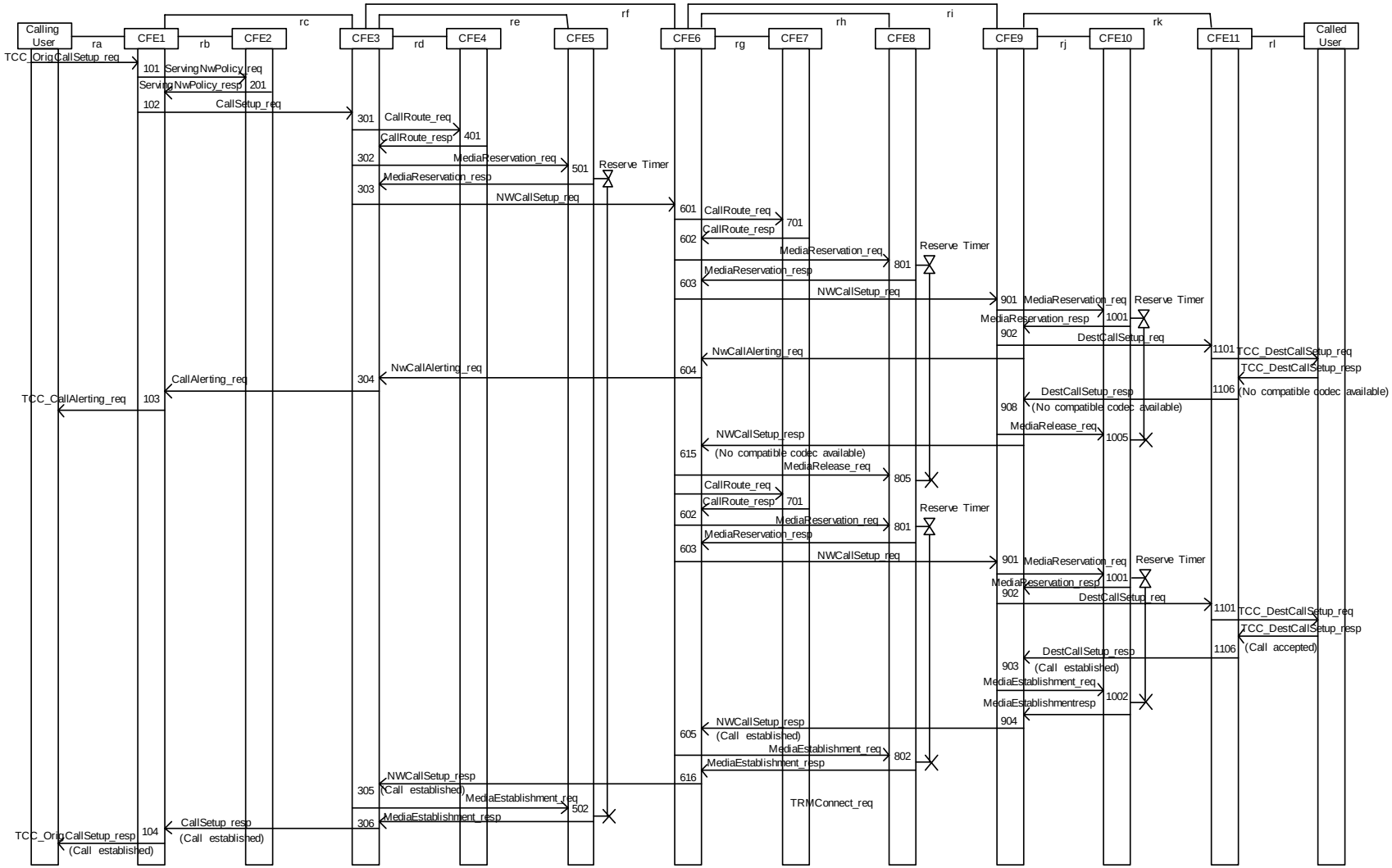


Figure 4: Information flows for successful call set-up with transcoding

5.2.3.1.2 Call clear-down

Figure 5 illustrates call clear-down initiated by the called user. For simple call both parties may initiate call clear-down at any time, hence also simultaneously call release requests are possible.

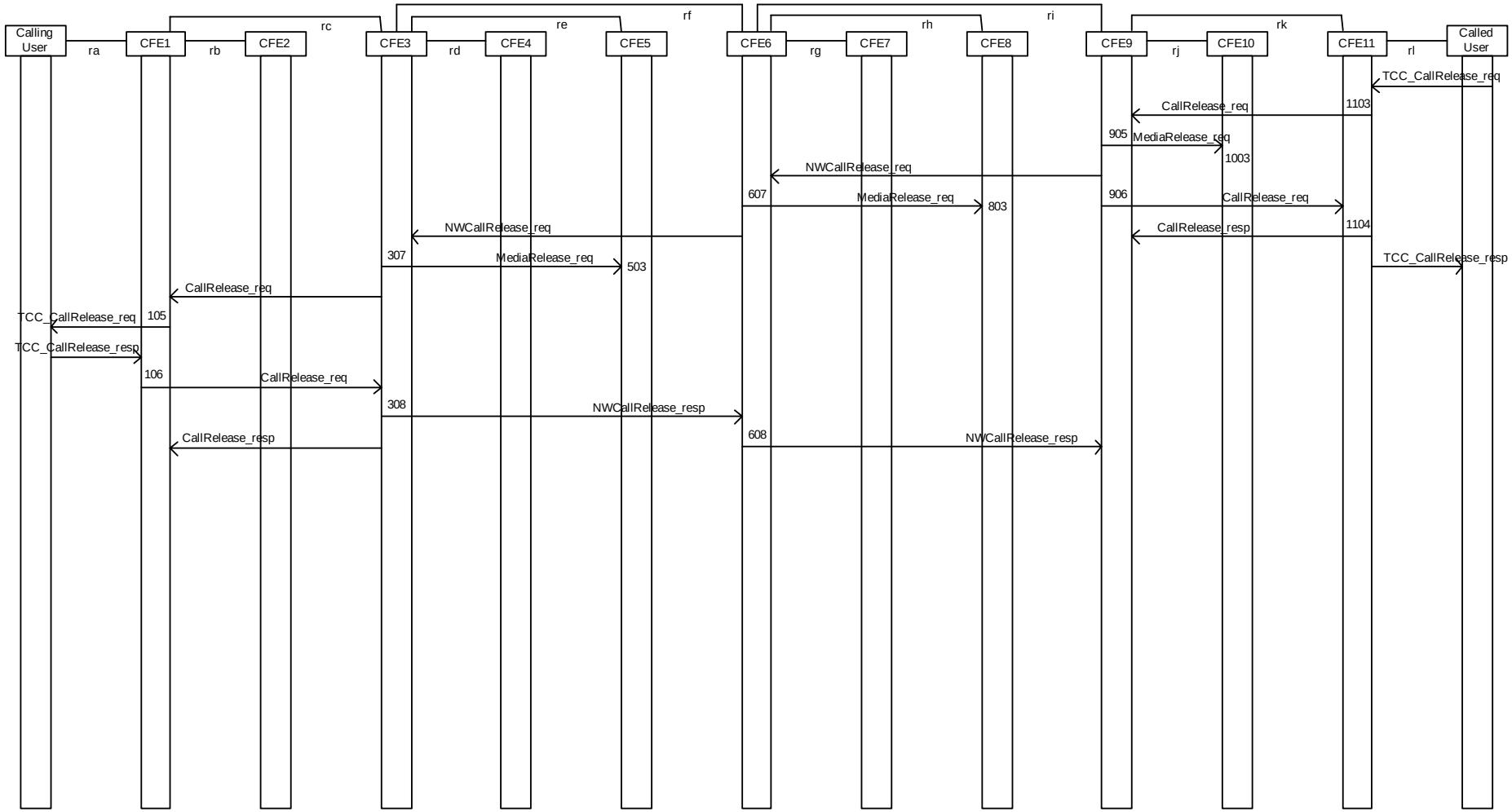


Figure 5: Information flows for successful call release initiated by called user

5.2.3.2 Exceptional behaviour

Figure 6 shows the information flow sequence for rejection of call establishment due to no route to called party found.

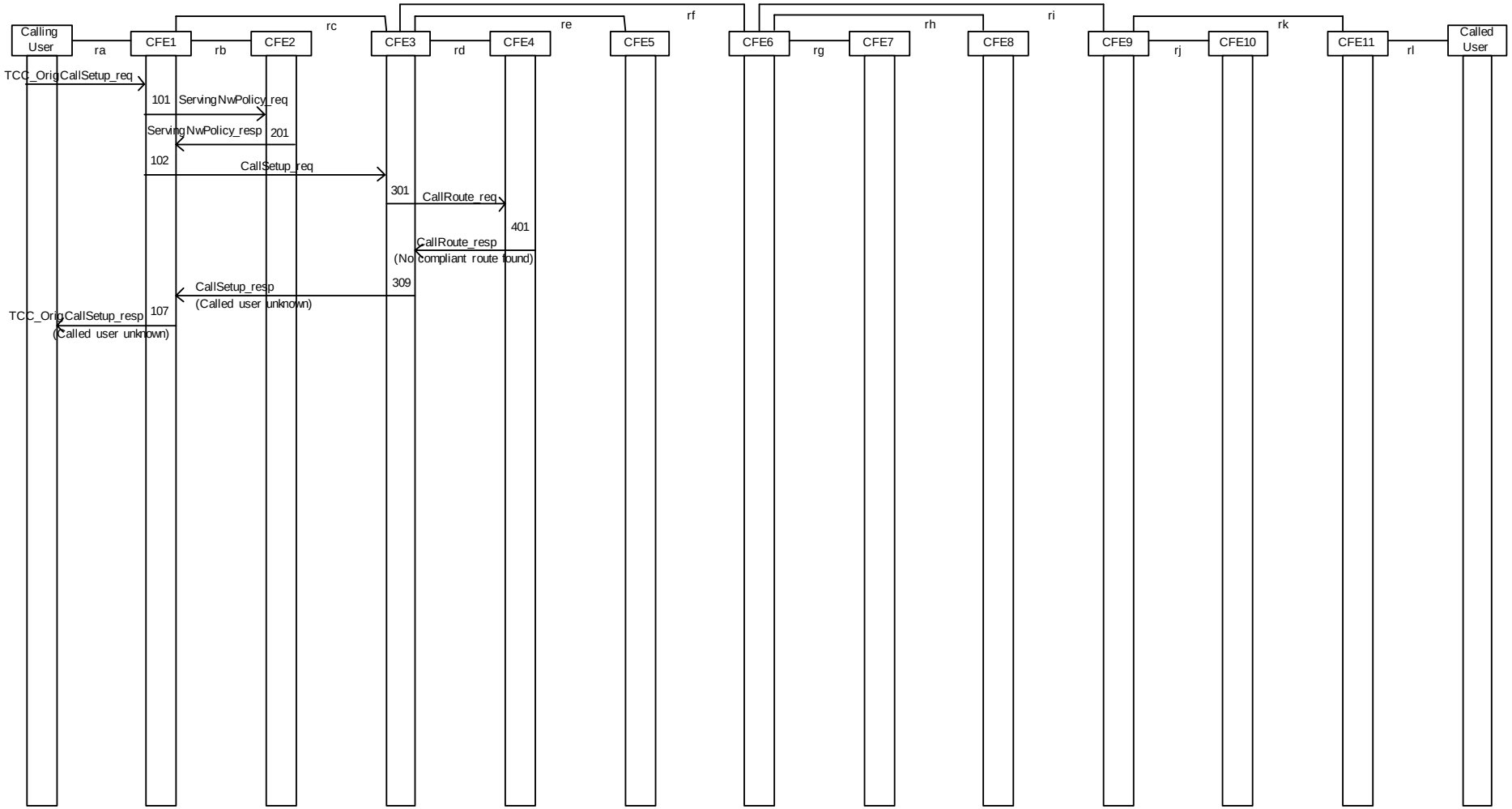


Figure 6: Information flows for unsuccessful call set-up due to called party unknown

Figure 7 shows call establishment due to requested QoS not supported for calling party.

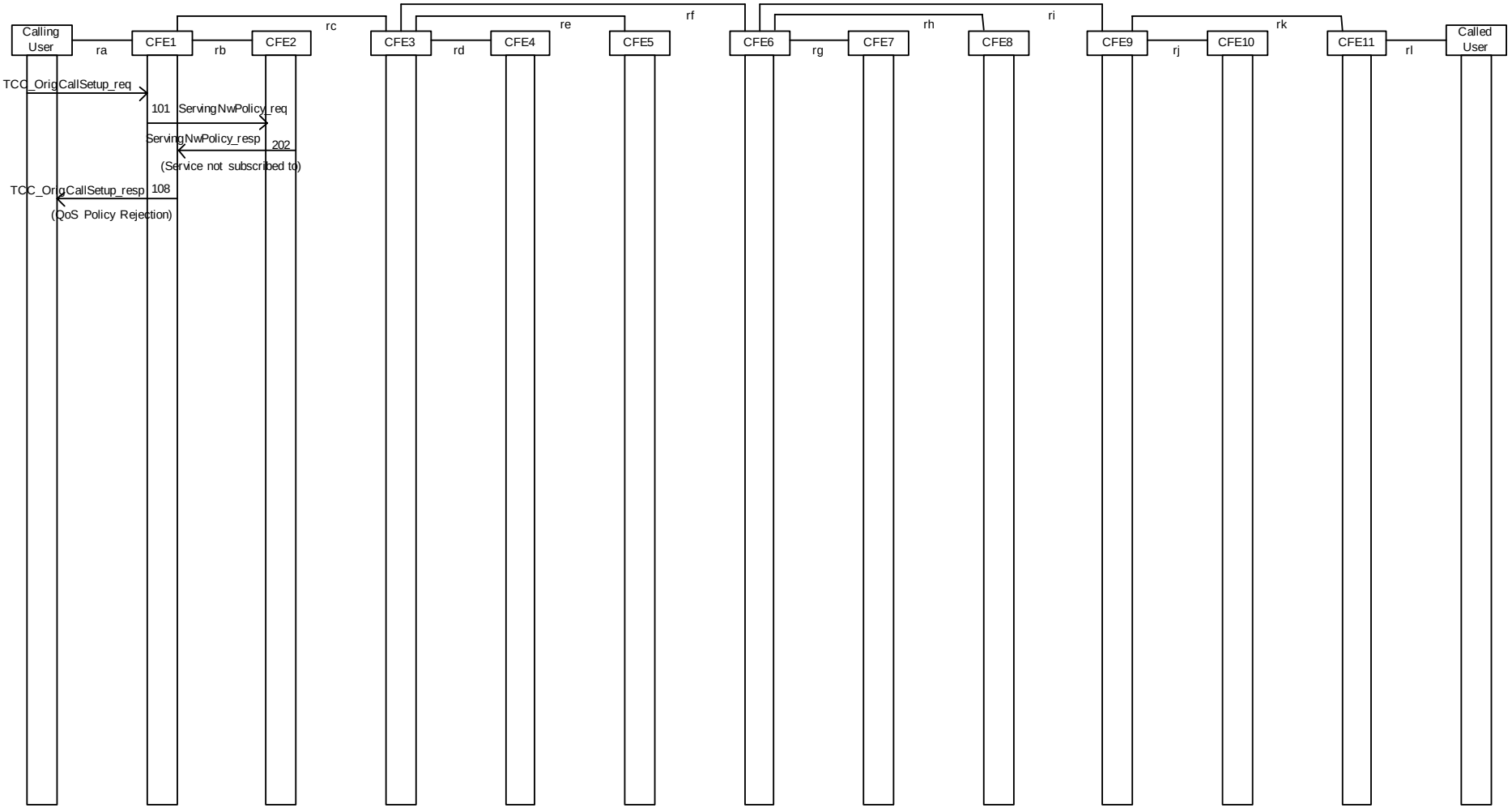


Figure 7: Information flows for unsuccessful call set-up due to QoS policy decision

Figure 8 shows rejection of call establishment due to required transport resources not being available.

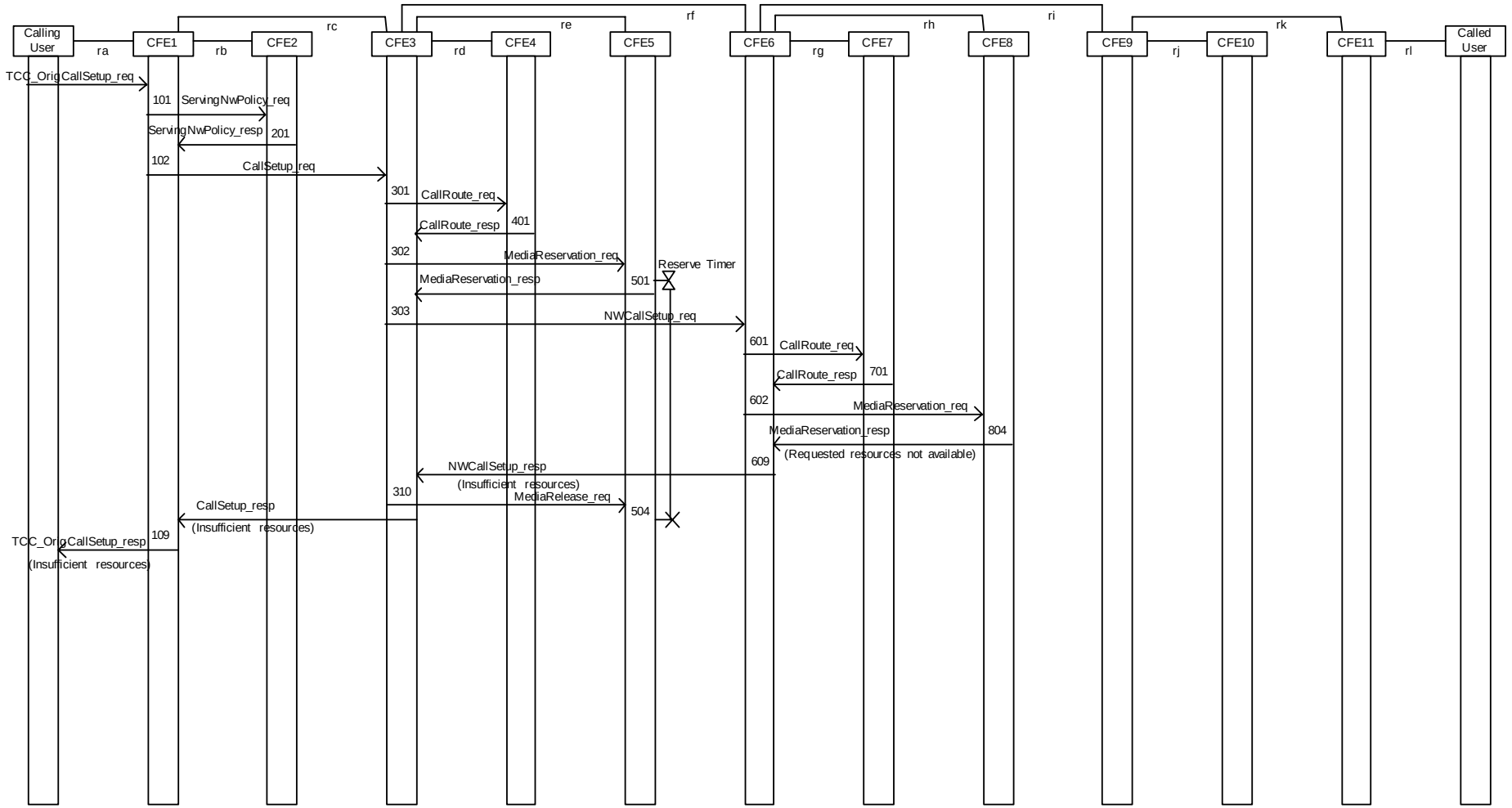


Figure 8: Information flows for unsuccessful call set-up due to transport resources unavailable

Figure 9 shows rejection of call establishment due to transport reservation time out.

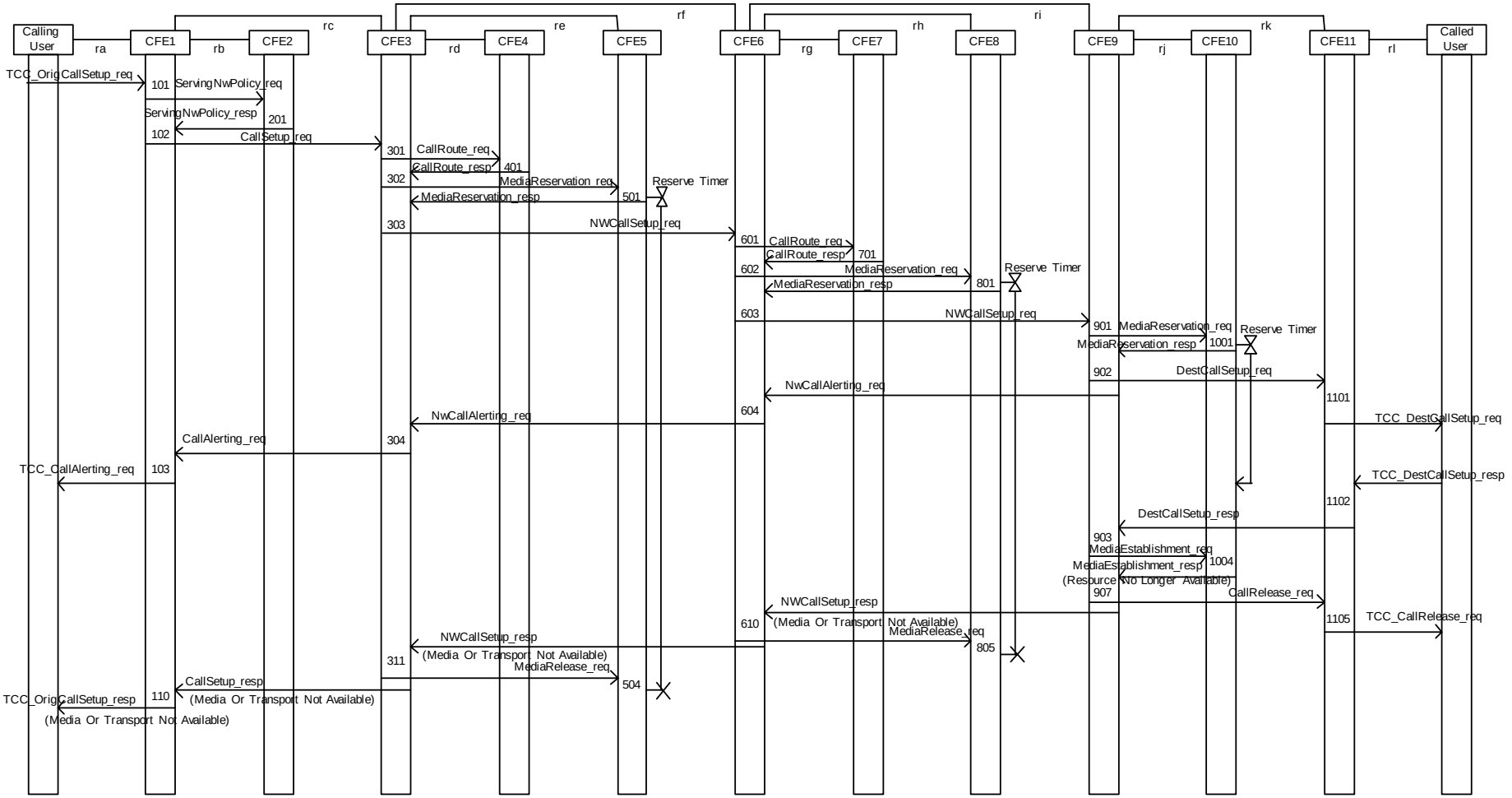


Figure 9: Information flows for unsuccessful call set-up due to transport reservation time out

Figure 10 shows rejection of call establishment due to no compatible codec being available at the called party.

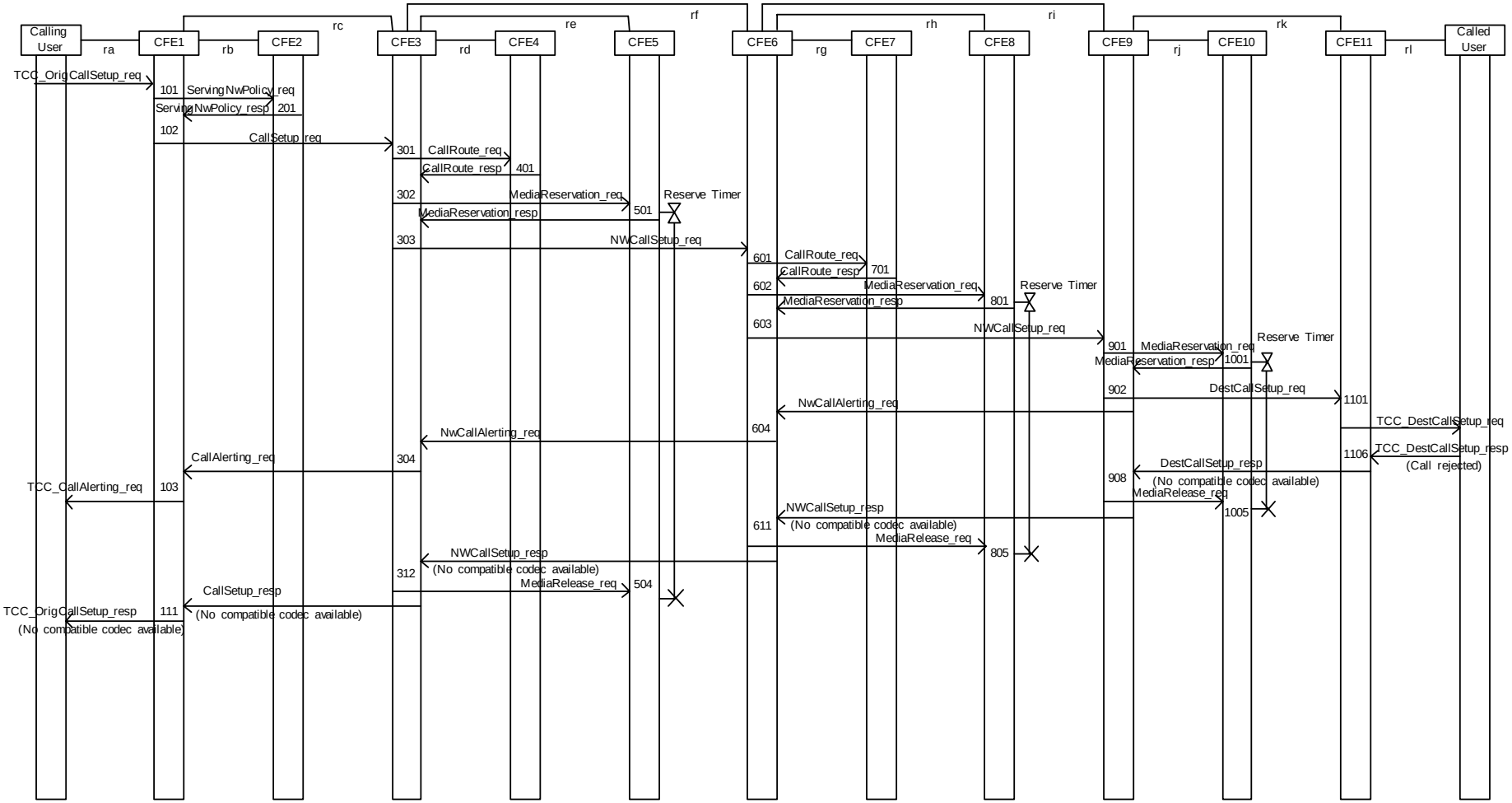


Figure 10: Information flows for unsuccessful call set-up due to called party offering no compatible codec

Figure 11 shows unsuccessful call establishment due to called user being busy.

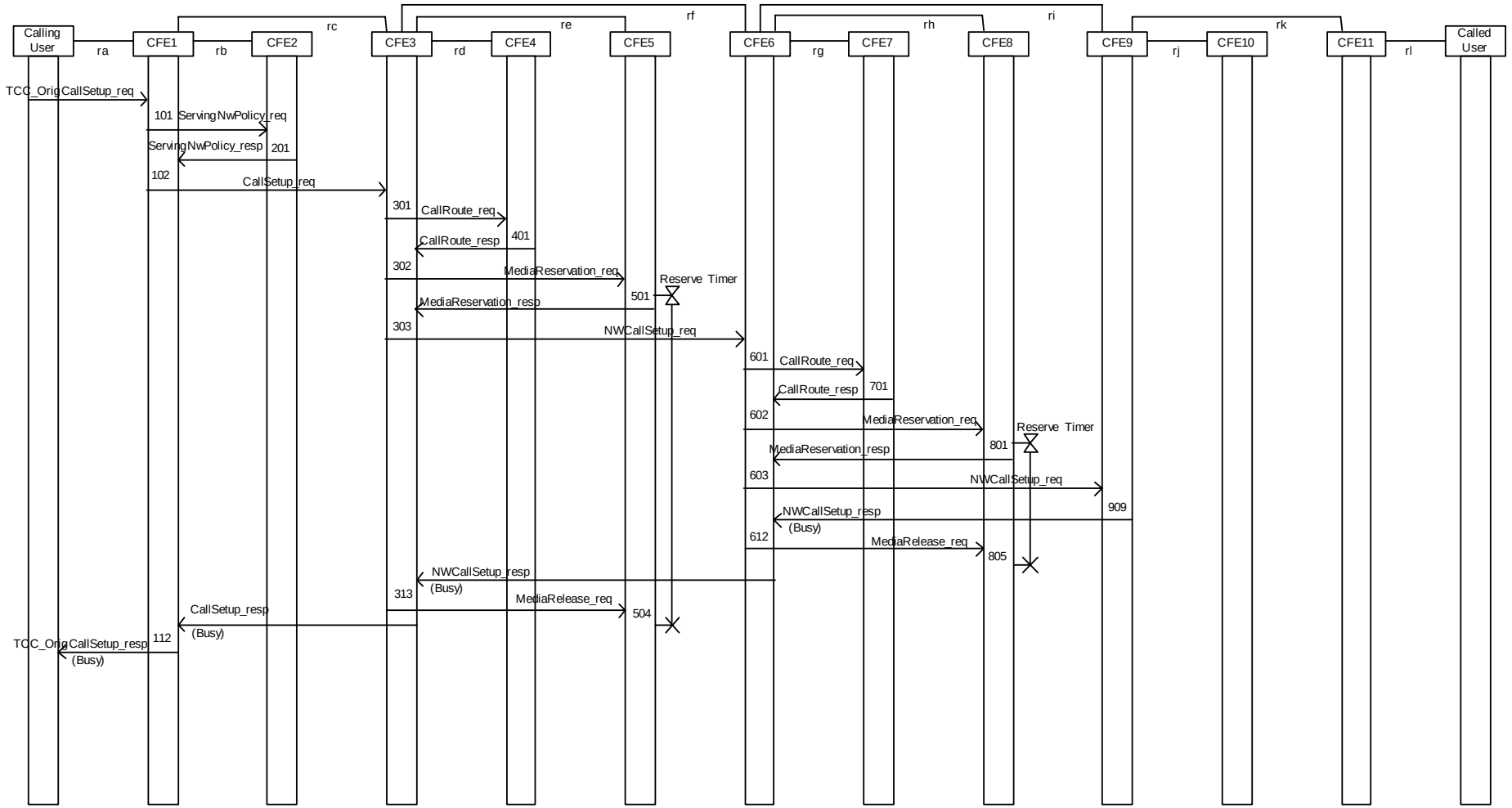


Figure 11: Information flows for unsuccessful call set-up due to called user busy

Figure 12 illustrates call clear-down initiated before call set-up has been completed.

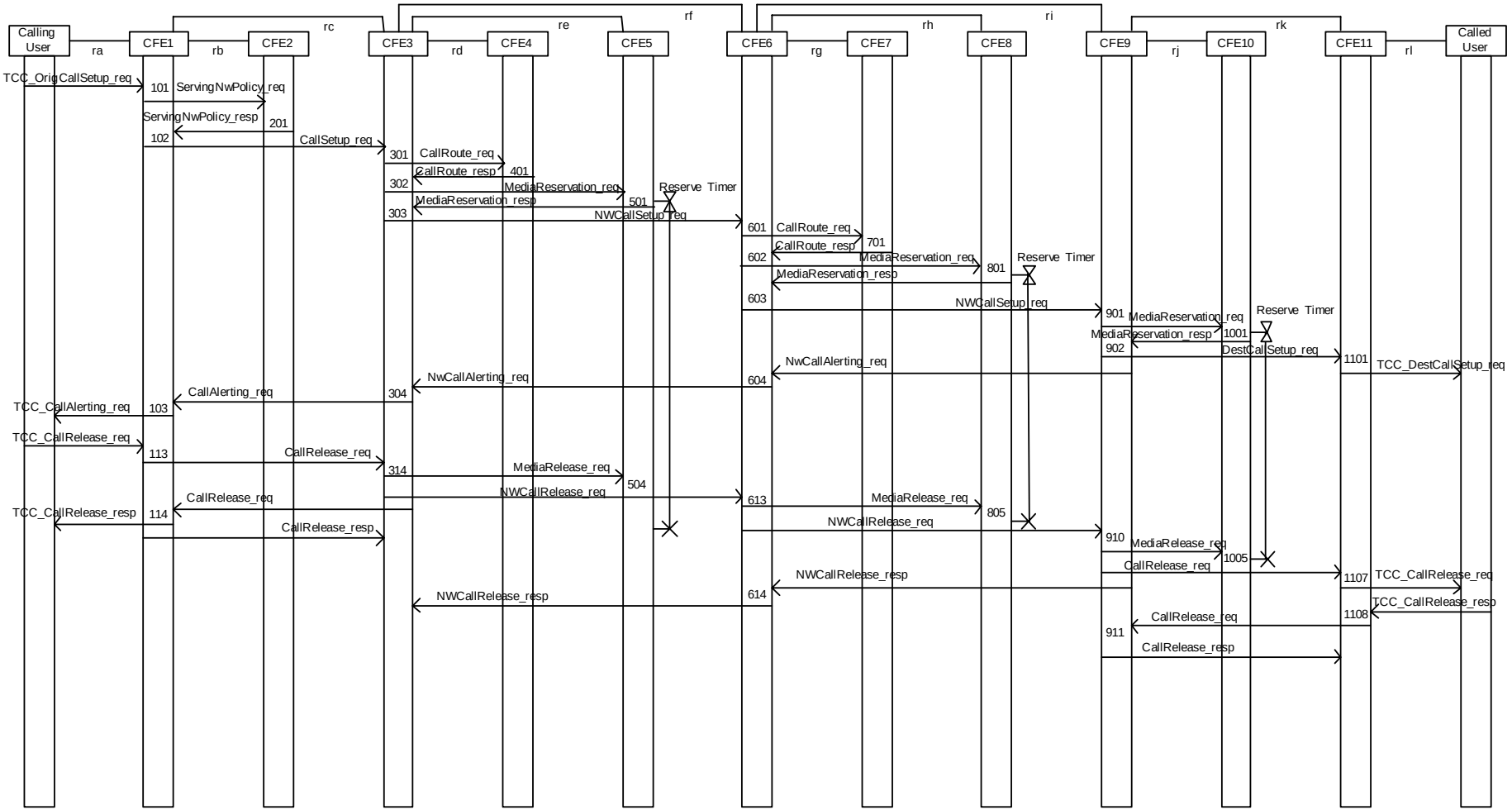


Figure 12: Information flows for call clear down before call set-up complete

5.3 Simple call functional entity actions

Throughout the descriptions of CFE actions, the following conventions are used to identify information flows:

- An information flow is referred to as a "request" at the CFE that sends it and as an "indication" at the CFE that receives it.
- The corresponding confirmation is referred to as a "response" at the CFE that sends it and as a "confirmation" at the CFE that receives it.

The following CFE actions shall occur at the points indicated in the figures of clause 5.2.3.

5.3.1 Actions of $CFE1_{OTC}$

101: On request from the Calling User (TCC_`OrigCallSetup` indication), determine transport parameters to be used, formulate a `ServingNWPolicy` request and send it to $CFE2_{PE}$.

102: When a positive ServingNWPolicy confirmation is received, derive the QoS transport parameters from the requested QoS class and construct the `CallSetup` request. The request is sent to the originating call control FE, $CFE3_{ONC}$, based on pre-defined or explicit operator selection information.

103: When a CallAlerting indication is received from the originating call coordination FE, $CFE3_{ONC}$, send a TCC_CallAlerting indication to the Calling User to indicate the called user is being alerted.

104: When a CallSetup confirmation is received from $CFE3_{ONC}$ with a positive call setup status construct a TCC_`OrigCallSetup` confirm and send it to the Calling User.

105: When a CallRelease indication is received from $CFE3_{ONC}$ and no call release has previously been initiated from the calling party, construct a call release request (TCC_CallRelease indication) to the Calling User.

106: When a TCC_CallRelease confirmation is received from the calling user, send a CallRelease indication to $CFE3_{ONC}$ and wait for a final CallRelease confirm.

107: When a CallSetup confirmation is received with call setup status "Called user unknown", construct a TCC_`OrigCallSetup` response with result "Called user unknown" and send it to the Calling User.

108: When a ServingNWPolicy confirmation is received with result "Invalid ticket", "Service not subscribed to" or "Service not currently available", construct a TCC_`OrigCallSetup` response with result "Policy rejection" and send it to the Calling User.

109: When a CallSetup confirmation is received with result "Insufficient resources", construct a TTC_OrigCallSetup response with result "Insufficient resources" and send it to the Calling User.

110: When a CallSetup confirmation is received with result "Transport not available", construct a TTC_OrigCallSetup response with result "Transport not available" and send it to the Calling User.

111: When a CallSetup confirmation is received with result "No compatible codec available", construct a TTC_OrigCallSetup response with result "No compatible codec available" and send it to the Calling User.

112: When a CallSetup confirmation is received with result "Busy", construct a TTC_OrigCallSetup response with result "Busy" and send it to the Calling User.

113: When a TCC_CallRelease indication is received from the calling user before the call setup procedure is complete, send a CallRelease to the originating call coordination FE, $CFE3_{ONC}$.

114: When a CallRelease indication is received from $CFE3_{ONC}$ and a call release has previously been initiated from the calling party, construct a call release response (TCC_CallRelease response) to the Calling User, and prepare a CallRelease response and send it to $CFE3_{ONC.}$

5.3.2 Actions of $CFE2_{PE}$

201: When a ServingNWPolicy indication is received check if the requested policy permits the call to proceed and if so, create a positive ServingNWPolicy confirmation and send it to $CFE1_{OTC}$.

202: When a ServingNWPolicy indication is received check if the requested policy permits the call to proceed and if not, create a ServingNWPolicy response with result "Service not subscribed to" or "Service not currently available" and send it to $CFE1_{OTC}$.

5.3.3 Actions of $CFE3_{ONC}$

301: Receive a CallSetup indication from $CFE1_{OTC}$, construct a call route request and send it to the routing FE, $CFE4_{OR}$.

302: If a positive call route confirmation is received, a next hop address is found, prepare a resource reservation request fulfilling the QoS requirements. Send the MediaReservation request to the originating call transport FE, $CFE5_{ONM}$.

303: When a positive MediaReservation confirmation is received, construct a NWCallSetup indication and send it to the identified next hop address, the intermediate call coordination function, $CFE6_{INC}$.

304: When a NwCallAlerting indication is received from the subsequent call control FE, $CFE6_{INC}$, send a CallAlerting indication to the calling user's service agent, $CFE1_{OTC}$.

305: Receive a NWCallSetup confirmation from the intermediate call coordination FE with a positive call setup result, send a MediaEstablishment indication to $CFE5_{ONM}$ to allocate the previously reserved resources.

306: When a positive MediaEstablishment confirmation is received, send a CallSetup response with Result "Call established".

307: When in an established call a NWCallRelease indication is received, construct a MediaRelease request to $CFE5_{ONM}$ to release allocated resources, and send a CallRelease indication to the calling user agent, $CFE1_{OTC}$.

308: When a CallRelease indication is received from the calling user agent, send a NWCallRelease confirmation to the intermediate coordination FE, $CFE6_{INC}$, and a CallRelease confirmation to the calling user agent, $CFE1_{OTC}$.

309: When a call route confirmation is received with result "No compliant route found", send a CallSetup response with result "Called user unknown" to $CFE1_{OTC}$.

310: When NWCallSetup confirmation is received from $CFE6_{INC}$ with result "Insufficient resources", send a MediaRelease request to $CFE5_{ONM}$, prepare a CallSetup response with result "Insufficient resources" and send it to $CFE1_{OTC}$.

311: When NWCallSetup confirmation is received from $CFE6_{INC}$ with result "Transport not available", send a MediaRelease request to $CFE5_{ONM}$, prepare a CallSetup response with result "Transport not available" and send it to $CFE1_{OTC}$.

312: When NWCallSetup confirmation is received from $CFE6_{INC}$ with result "No compatible codec available" and no transcoding is possible, send a MediaRelease request to $CFE5_{ONM}$, prepare a CallSetup response with result "No compatible codec available" and send it to $CFE1_{OTC}$.

313: When NWCallSetup confirmation is received from $CFE6_{INC}$ with result "Busy", send a MediaRelease request to $CFE5_{ONM}$, prepare a CallSetup response with result "Busy" and send it to $CFE1_{OTC}$.

314: When CallRelease indication is received from $CFE1_{OTC}$, if transport resources have been reserved send a MediaRelease request to $CFE5_{ONM}$ to release these resources, prepare a NWCallRelease and send it to $CFE6_{INC}$, and prepare a CallRelease request and send it to $CFE1_{OTC}$.

5.3.4 Actions of $CFE4_{OR}$

401: When a CallRoute indication is received, a next hop address list shall be identified based on the called user and the QoS parameters. A CallRoute confirmation containing the identified next hop address list shall be sent to the originating call coordination FE, $CFE3_{ONC}$.

5.3.5 Actions of $CFE5_{ONM}$

501: When a MediaReservation indication is received it shall be checked if resource is available to meet the QoS requirements and if so it shall be reserved. A MediaReservation confirmation shall be sent to $CFE3_{ONC}$ and the Reservation Hold Timer shall be started.

502: When a MediaEstablishment indication is received and the reserved resource is available the Reservation Hold Timer shall be stopped, the resource allocated, and a MediaEstablishment confirmation shall be sent to $CFE3_{ONC}$.

503: When a MediaRelease indication is received the allocated resource shall be released.

504: When a MediaRelease indication is received allocated resources shall be released and the Reservation hold timer stopped.

5.3.6 Actions of $CFE6_{INC}$

601: On request from the originating call coordination FE (NW`OrigCallSetup` request), construct a CallRoute request based on the Called User and QoS parameters. Send the CallRoute indication to the intermediate call route FE, $CFE7_{IR}$.

602: If a next hop address list is returned in the CallRoute confirmation from $CFE7_{IR}$, prepare a resource reservation request fulfilling the QoS requirements using the first next hop address in the list. Send the MediaReservation request to the intermediate call transport FE, $CFE8_{INM}$.

603: When a positive MediaReservation confirmation is received, construct a NWCallSetup request and send it to the destination call coordination function, $CFE9_{TNC}$.

604: When a NwCallAlerting indication is received from the destination call control coordination FE (or a subsequent intermediate call control FE), send a NwCallAlerting indication to the originating call control coordination FE, $CFE3_{ONC}$, ((or a previous intermediate call control FE).

605: Receive a NWCallSetup confirmation from the destination call coordination FE with a positive call setup result, send a MediaEstablishment request to $CFE8_{INM}$ to allocate the previously reserved resources.

606: When a positive MediaEstablishment confirmation is received, send a NWCallSetup confirmation with Result "Call established" to $CFE3_{ONC}$.

607: Receive a NWCallRelease indication from the destination call coordination FE, send a MediaRelease request to $CFE8_{INM}$ to release allocated resources and send a NWCallRelease indication to the originating call coordination FE, $CFE3_{ONC}$.

608: When a NWCallRelease confirmation is received from $CFE3_{ONC}$, send a NWCallRelease confirmation to the destination coordination FE, $CFE9_{TNC}$.

609: When a MediaReservation confirmation is received with result "Requested resources not available", prepare a NWCallSetup response with result "Insufficient resources" and send it to $CFE3_{ONC}$.

610: When a NWCallSetup confirmation is received with result "Transport not available", send a MediaRelease request to $CFE8_{INM}$, prepare a NWCallSetup response with result "Transport not available" and send it to $CFE3_{ONC}$.

611: When a NWCallSetup confirmation is received with result "No compatible codec available" and no transcoding is possible, send a MediaRelease request to $CFE8_{INM}$, prepare a NWCallSetup response with result "No compatible codec available" and send it to $CFE3_{ONC}$.

612: When a NWCallSetup confirmation is received with result "Busy", send a MediaRelease request to $CFE8_{INM}$, prepare a NWCallSetup response with result "Busy" and send it to $CFE3_{ONC}$.

613: When a NWCallRelease indication is received from the originating call control FE, $CFE3_{ONC}$, send a MediaRelease request to $CFE8_{INM}$, and send a NWCallRelease request to the destination call control FE, $CFE9_{TNC}$.

614: When a NWCallRelease confirmation is received from $CFE9_{TNC}$, send a NWCallRelease confirmation to the originating coordination FE, $CFE3_{ONC}$.

615: When a NWCallSetup confirmation is received from $CFE9_{TNC}$ with result "No compatible codec available" and transcoding is possible, send a MediaRelease request to $CFE8_{INM,}$ prepare a call route request based on the transcoding and send a CallRoute request including the new codec.

616: When a positive MediaEstablishment confirmation is received and transcoding is performed, send a NWCallSetup response to $CFE3_{ONC}$ with Result "Call established" and Codec set to the codec chosen to transcode from.

5.3.7 Actions of $CFE7_{IR}$

701: When a CallRoute indication is received, a next hop address list shall be identified based on the called user and the QoS parameters. A CallRoute confirmation containing the identified next hop address list shall be sent to the intermediate call coordination FE, $CFE6_{INC}$.

5.3.8 Actions of $CFE8_{INM}$

801: When a MediaReservation indication is received it shall be checked if resource is available to meet the QoS requirements and if so it shall be reserved. A MediaReservation confirmation shall be sent to $CFE6_{INC}$ and the Reservation Hold Timer shall be started.

802: When a MediaEstablishment indication is received and the reserved resource is available the Reservation Hold Timer shall be stopped, the resource allocated, and a MediaEstablishment response shall be sent to $CFE6_{INC}$.

803: When a MediaRelease indication is received the allocated resource shall be released.

804: When a MediaReservation indication is received with transport resource requirements that cannot be met, prepare a MediaReservation response with result "Requested resources not available" and send it to $CFE6_{INC}$.

805: When a MediaRelease indication is received and the Reservation timer has not expired, the timer shall be stopped and the reserved transport resources released.

5.3.9 Actions of $CFE9_{TNC}$

901: When a Nw`CallSetup` indication is received from $CFE6_{INC}$, check if the called user terminal capabilities allow for an additional call (or if terminal capabilities are not available) construct a resource reservation request fulfilling the QoS requirements, and send the MediaReservation request to the destination call transport FE, $CFE10_{TNM}$.

902: When a positive MediaReservation confirmation is received, construct a DestCallSetup indication and send it to the called user agent, $CFE11_{TTC}$ and send a NwCallAlerting indication to the previous call control FE, $CFE6_{INC}$, to indicate that the called user is being alerted.

903: Receive a DestCallSetup confirmation from the user agent FE with a positive call setup result, send a MediaEstablishment request to $CFE10_{TNM}$ to allocate the previously reserved resources.

904: Receive a positive MediaEstablishment confirmation, send a NWCallSetup confirmation with Result "Call established" to $CFE6_{INC}$.

905: Receive a CallRelease indication from the called user agent, send a MediaRelease request to $CFE10_{TNM}$ to release allocated resources, send a NWCallRelease indication to the intermediate call coordination FE, $CFE6_{INC}$. Receive a NWCallRelease confirmation when $CFE6_{INC}$ has cleared down the call.

906: Send a CallRelease indication to the called user agent, CFE11and receive a CallRelease confirmation when the user agent has cleared down the call.

907: When a MediaEstablishment confirmation is received with result "Unable to connect", send a CallRelease_Req to $CFE11_{TTC}$, prepare a NWCallSetup response with result "Transport not available" and send it to $CFE6_{INC}$.

908: When a DestCallSetup confirmation is received from $CFE11_{TTC}$ with result "No compatible codec available" and no transcoding is possible, send a MediaRelease request to $CFE10_{TNM}$, prepare a NWCallSetup response with result "No compatible codec available" and send it to $CFE6_{INC}$.

909: When a NwCallSetup indication is received from $CFE6_{INC}$, and from the called user terminal capabilities information it is determined that no additional simultaneous call can be established, prepare a NWCallSetup response with result "Busy" and send it to $CFE6_{INC}$.

910: When a NWCallRelease indication is received from $CFE6_{INC}$, send a MediaRelease request to $CFE10_{TNM}$, prepare and send a CallRelease request to the called user agent, CFE11, and reply back to $CFE6_{INC}$ by sending a NWCallRelease response.

911: Receive a CallRelease indication from the called user agent, CFE11, send a CallRelease response the called user agent.

5.3.10 Actions of $CFE10_{TNM}$

1001: When a MediaReservation indication is received it shall be checked if resource is available to meet the QoS requirements and if so it shall be reserved. A MediaReservation confirmation shall be sent to $CFE9_{TNC}$ and the Reservation Hold Timer shall be started.

1002: When a MediaEstablishment indication is received and the reserved resource is available the Reservation Hold Timer shall be stopped, the resource allocated, and a MediaEstablishment response shall be sent to $CFE9_{TNC}$.

1003: When a MediaRelease indication is received the allocated resource shall be released.

1004: When a MediaEstablishment indication is received after a reserved resources has been released due to Reservation Hold timer expiration, prepare a MediaEstablishment response with result "Unable to connect" and send it to $CFE9_{TNC}$.

1005: When a MediaRelease indication is received the transport resource reservation timer shall be stopped and allocated resources released.

5.3.11 Actions of $CFE11_{TTC}$

1101: Receive a DestCallSetup indication from the destination call coordination FE, $CFE9_{TNC}$, send a TCC_DestCallSetup indication to the Called User.

1102: When a positive TCC_DestCallSetup confirmation is received from the Called User, the DestCallSetup confirmation is sent to $CFE9_{TNC}$.

1103: If in an active call a TCC_CallRelease indication is received from the Called User a CallRelease indication shall be sent to the called user coordination FE, $CFE9_{TNC}$.

1104: When after a call release has been initiated by the called user, a CallRelease indication is received from $CFE9_{TNC}$, a CallRelease confirmation shall be sent to $CFE9_{TNC}$ and a TCC_CallRelease confirmation shall be sent to the called user.

1105: When a CallRelease indication is received from $CFE9_{TNC}$, prepare a TCC_CallRelease request and send it to the Called User.

1106: When a TCC_DestCallSetup confirmation is received from the Called User with result "No compatible codec", prepare a DestCallSetup response with result "No compatible codec available" and send it to $CFE9_{TNC}$. (If the received reject cause is "Busy", a DestCallSetup response with result "Busy" shall be sent. This scenario is not shown.)

1107: When a CallRelease indication is received from $CFE9_{TNC}$ and no call release has been initiated by the called user, send a TCC_CallRelease request to the called user.

1108: When from the called user a TCC_CallRelease confirmation is received, send a CallRelease request to the destination call control FE, $CFE9_{TNC.}$

5.4 Functional entity behaviour

The behaviour specified in this subclause is intended to illustrate typical CFE behaviour in terms of information flows sent and received.

The behaviour of each CFE is shown using the Specification and Description Language (SDL) defined in ITU-T Recommendation Z.100 [5].

NOTE: The complete SDL model which was used for validation purposes (and from which the following process diagrams were extracted) is available as separate document in the ITU defined Common Interchange Format.

5.4.1 Behaviour of $CFE1_{OTC}$

The behaviour of $CFE1_{OTC}$ is shown in the SDL process diagram in figure 13.

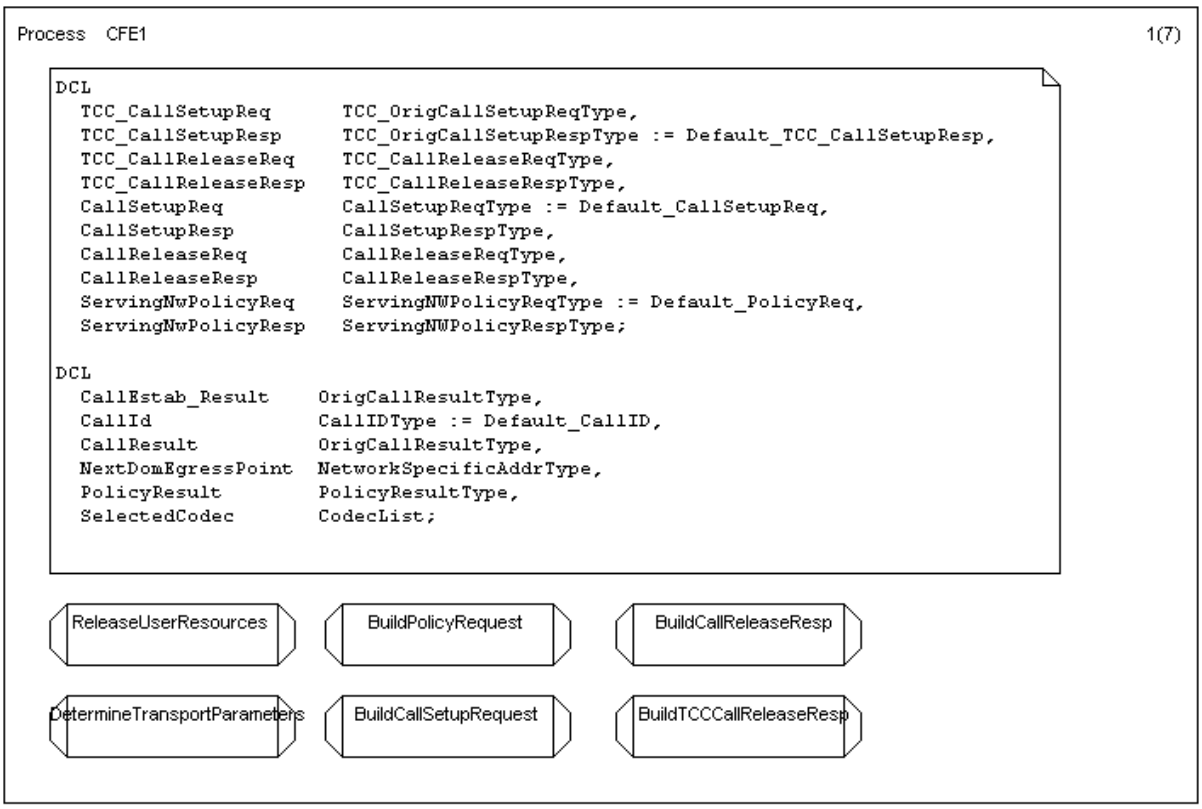


Figure 13 (sheet 1 of 7): SDL process diagram for functional entity $CFE1_{OTC}$

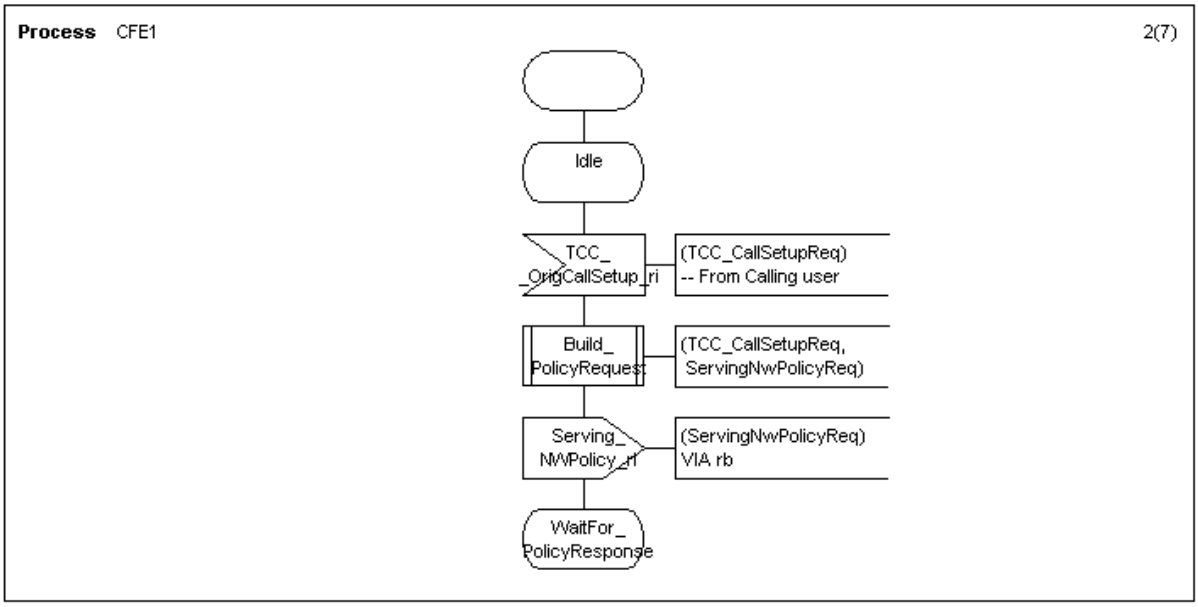


Figure 13 (sheet 2 of 7): SDL process diagram for functional entity $CFE1_{OTC}$

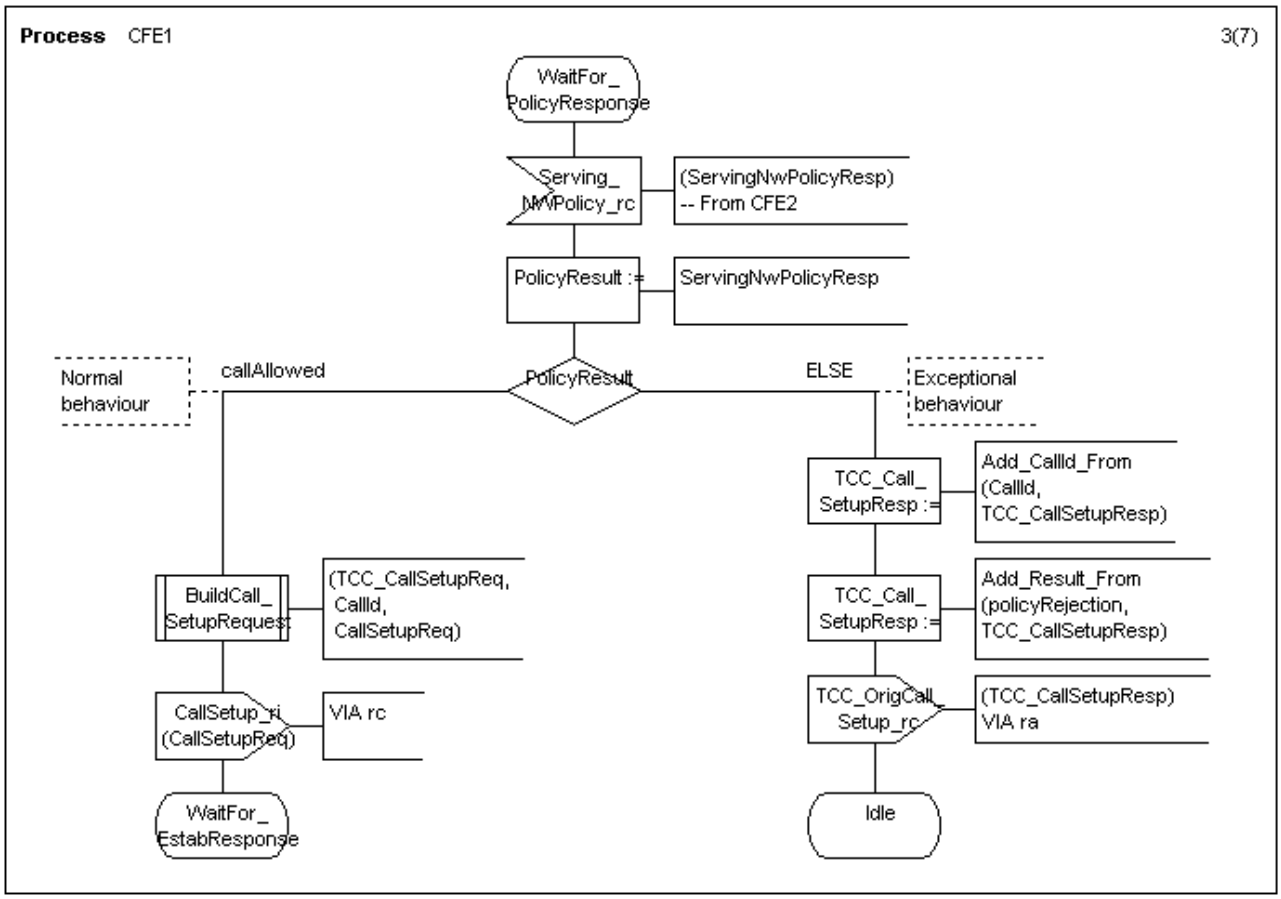


Figure 13 (sheet 3 of 7): SDL process diagram for functional entity $CFE1_{OTC}$

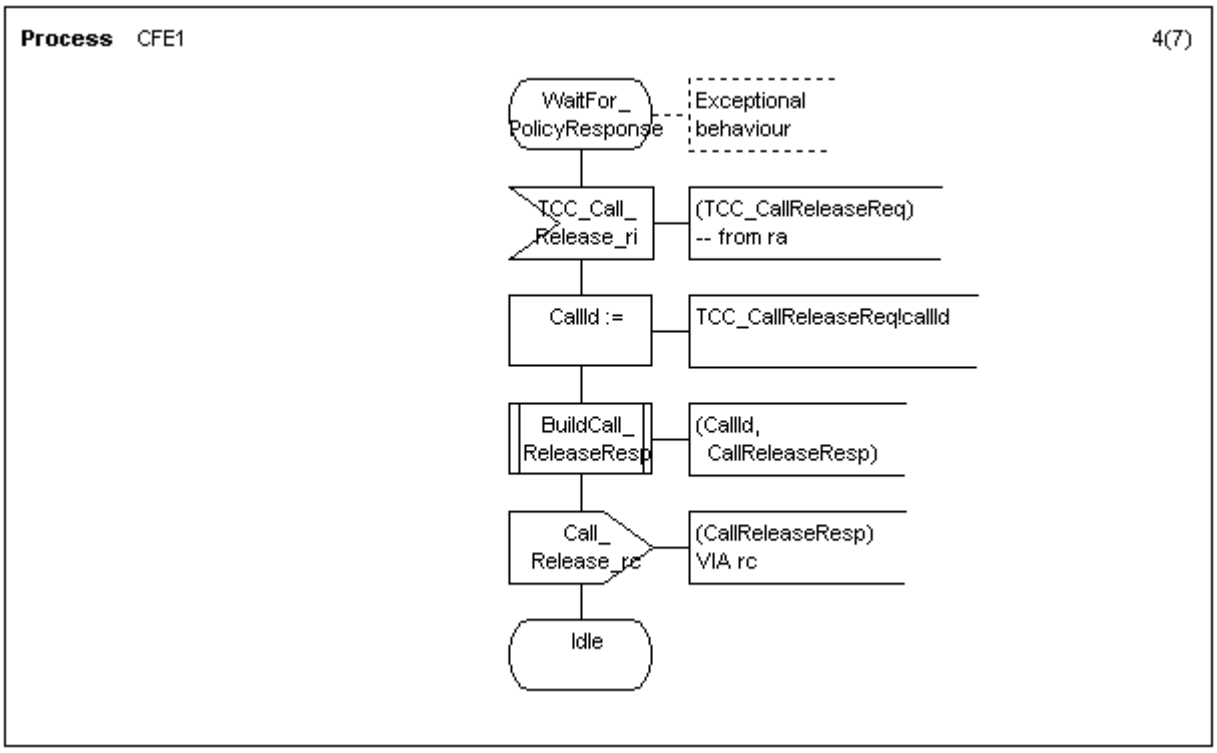


Figure 13 (sheet 4 of 7): SDL process diagram for functional entity $CFE1_{OTC}$

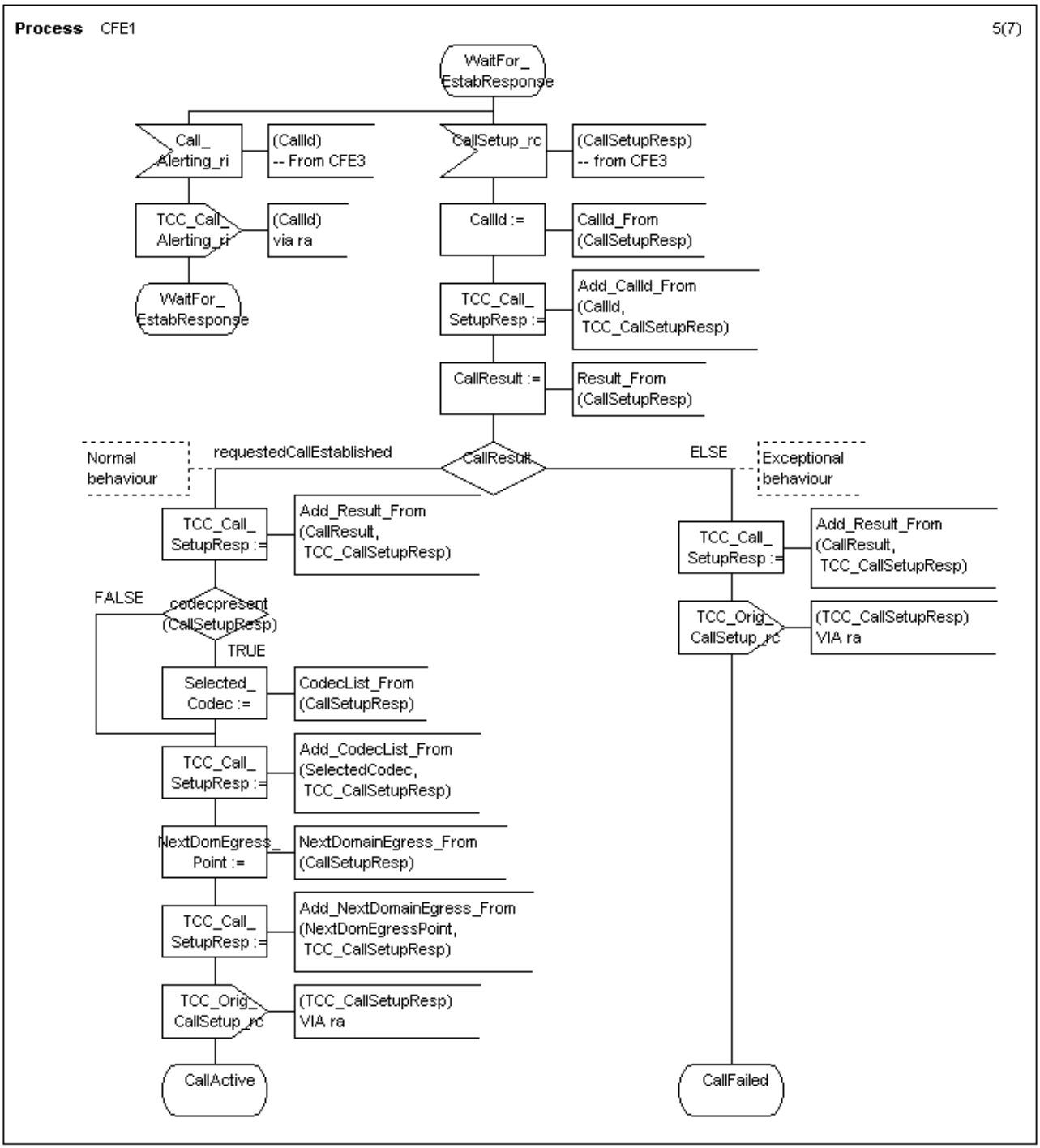


Figure 13 (sheet 5 of 7): SDL process diagram for functional entity $CFE1_{OTC}$

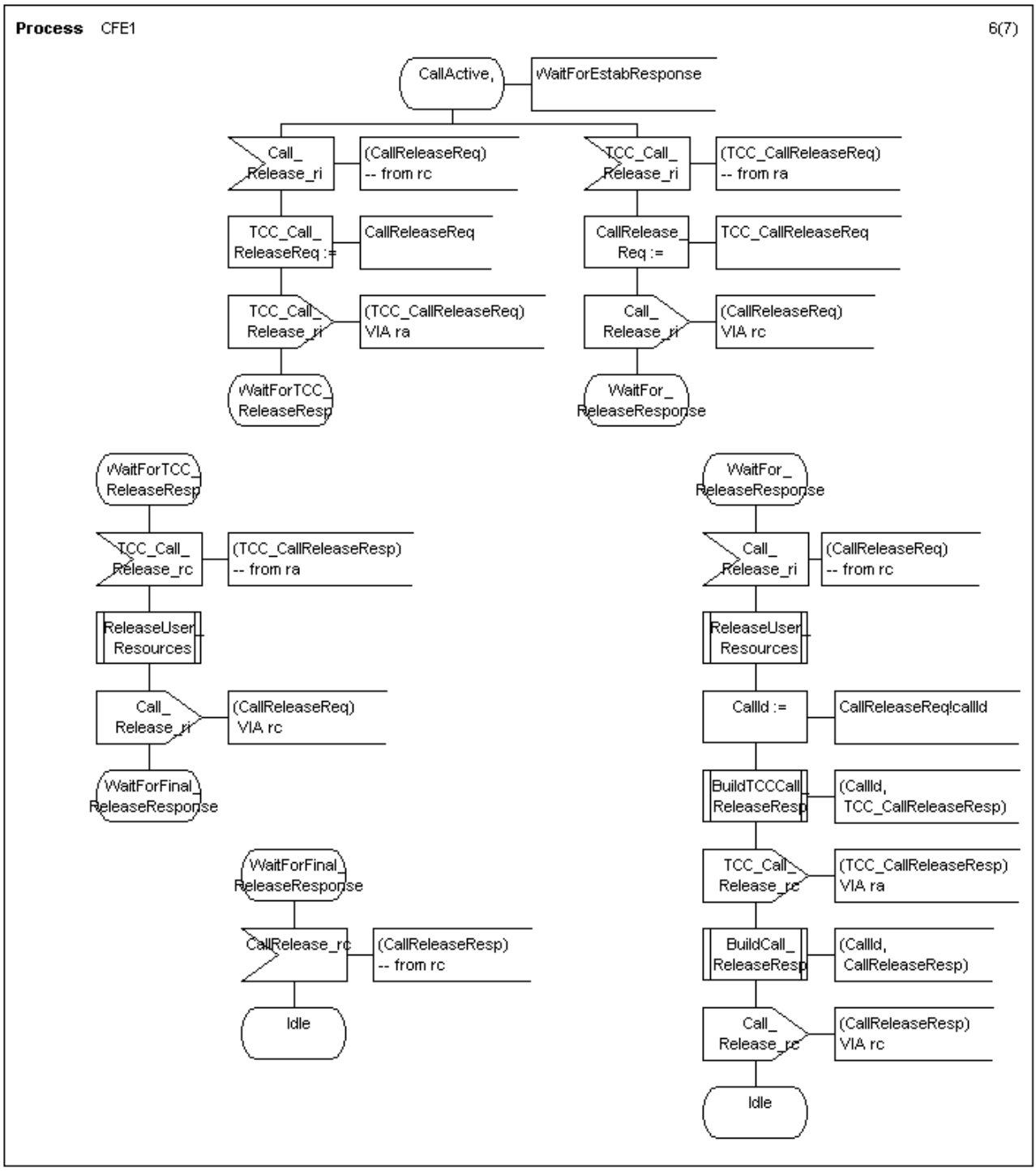


Figure 13 (sheet 6 of 7): SDL process diagram for functional entity $CFE1_{OTC}$

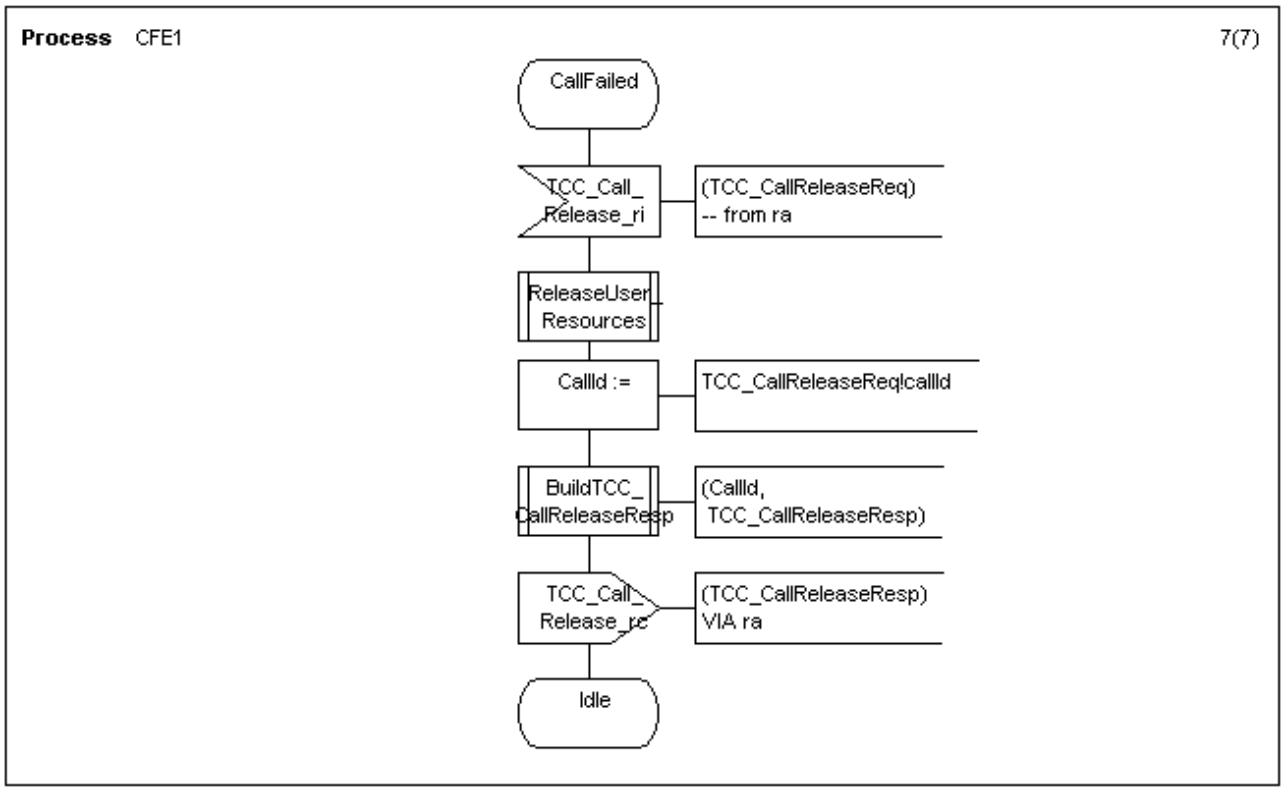


Figure 13 (sheet 7 of 7): SDL process diagram for functional entity $CFE1_{OTC}$

5.4.2 Behaviour of $CFE2_{PE}$

The behaviour of $CFE2_{PE}$ is shown in the SDL process diagram in figure 14.

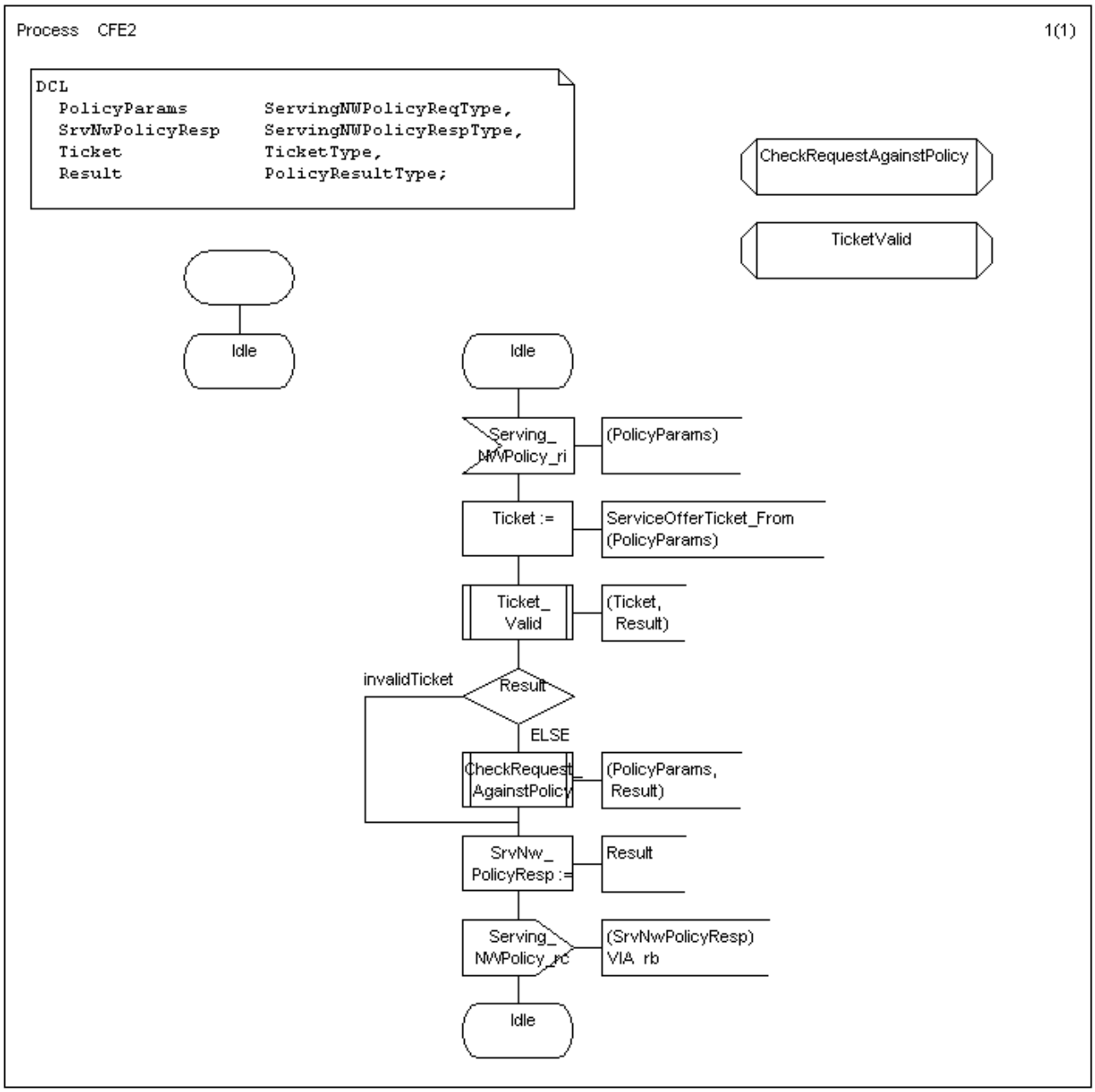


Figure 14: SDL process diagram for functional entity $CFE2_{PE}$

5.4.3 Behaviour of $CFE3_{ONC}$

The behaviour of $CFE3_{ONC}$ is shown in the SDL process diagram in figure 15.

Process CFE3 1(11)

```
DCL
  CallSetupReq           CallSetupReqType := Default_CallSetupReq,
  CallSetupResp          CallSetupRespType := Default_CallSetupResp,
  CallRouteReq           CallRouteReqType := Default_CallRouteReq,
  CallRouteResp          CallRouteRespType,
  CallReleaseReq         CallReleaseReqType,
  CallReleaseResp        CallReleaseRespType,
  NWcallSetupReq         NW_CallSetupReqType := Default_NW_CallSetupReq,
  NWcallSetupResp        NW_CallSetupRespType,
  NwCallReleaseReq       NW_CallReleaseReqType,
  NwCallReleaseResp      NW_CallReleaseRespType,
  MediaReservationReq    MediaReservationReq_Type := Default_MediaReserveReq,
  MediaReservationResp   MediaReservationResp_Type,
  MediaEstablishmentReq  MediaEstablishmentReq_Type := Default_MediaEstablishmentReq,
  MediaEstablishmentResp MediaEstablishmentResp_Type,
  MediaReleaseReq        MediaReleaseReq_Type;

DCL
  AppliedCodecList    CodecList,
  AttemptTranscoding  Boolean,
  BearerId            BearerIdType := Default_BearerId,
  Callee              TiphonUserName,
  Caller              TiphonUserName,
  CallId              CallIDType := Default_CallID,
  CallResult          NWCallResultType,
  ConnectResult       MediaEtabResultType,
  DestCodecSelected   CodecType := Default_Codec,
  EgressPoint         NetworkSpecificAddrType,
  NextHopAddr         NetworkDomainAddrType,
  NextHopAddrList     DomainAddrListType,
  OrigCodecList       CodecList,
  OrigCodecSelected   CodecType := Default_Codec,
  ReserveResult       MediaResResultType,
  RouteResult         RouteResultType,
  Transcoded          Boolean := FALSE,
  TranscodeCount      TranscodeCountType;
```

```
/* The maximum allowed number of codec transcodings in the call path */

SYNONYM MAXTRANSCODINGS Integer = EXTERNAL;
```

```
SYNONYM Consumed Boolean = FALSE;
SYNONYM Available Boolean = TRUE;
```

```
SYNONYM NullBearerId BearerIdType = '';
```

Figure 15 (sheet 1 of 11): SDL process diagram for functional entity $CFE3_{ONC}$

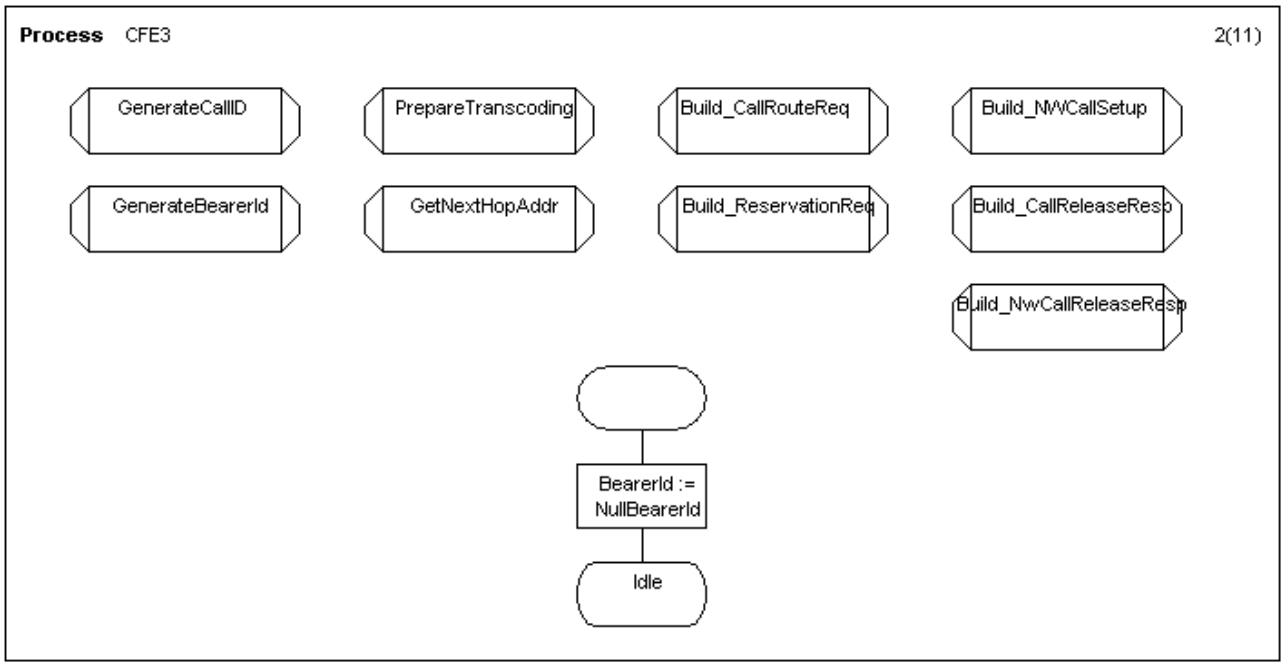


Figure 15 (sheet 2 of 11): SDL process diagram for functional entity $CFE3_{ONC}$

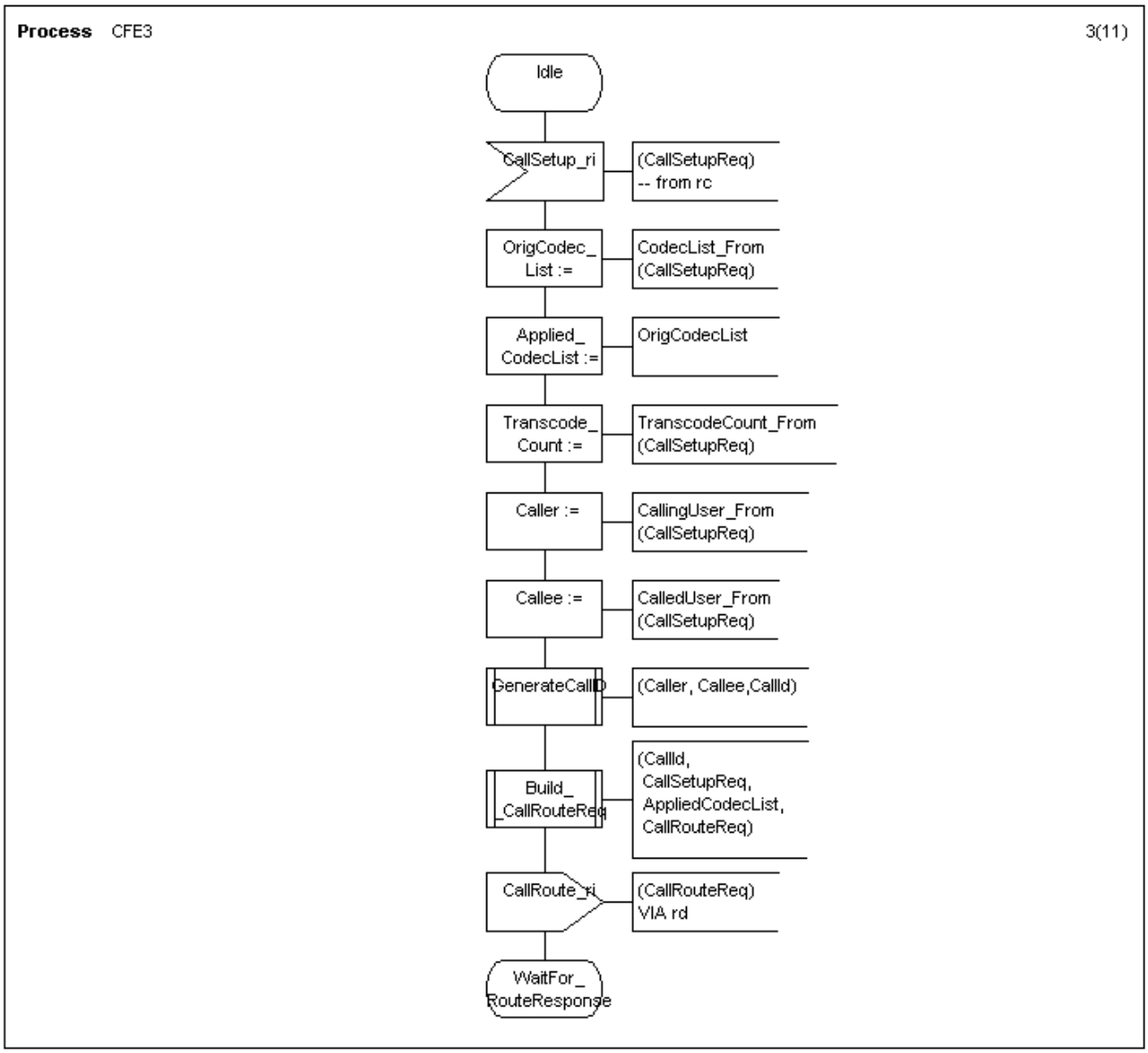


Figure 15 (sheet 3 of 11): SDL process diagram for functional entity $CFE3_{ONC}$

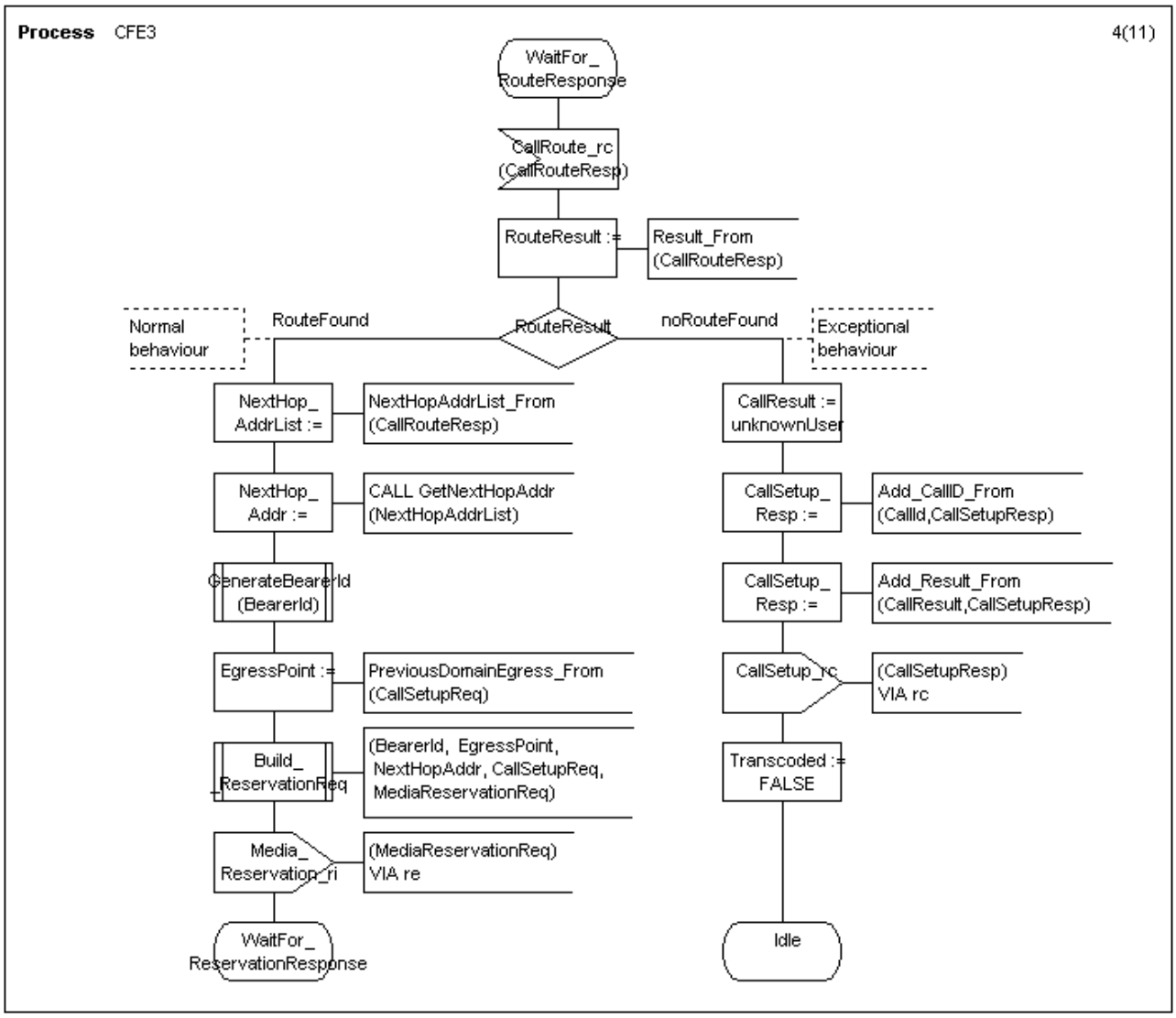


Figure 15 (sheet 4 of 11): SDL process diagram for functional entity $CFE3_{ONC}$

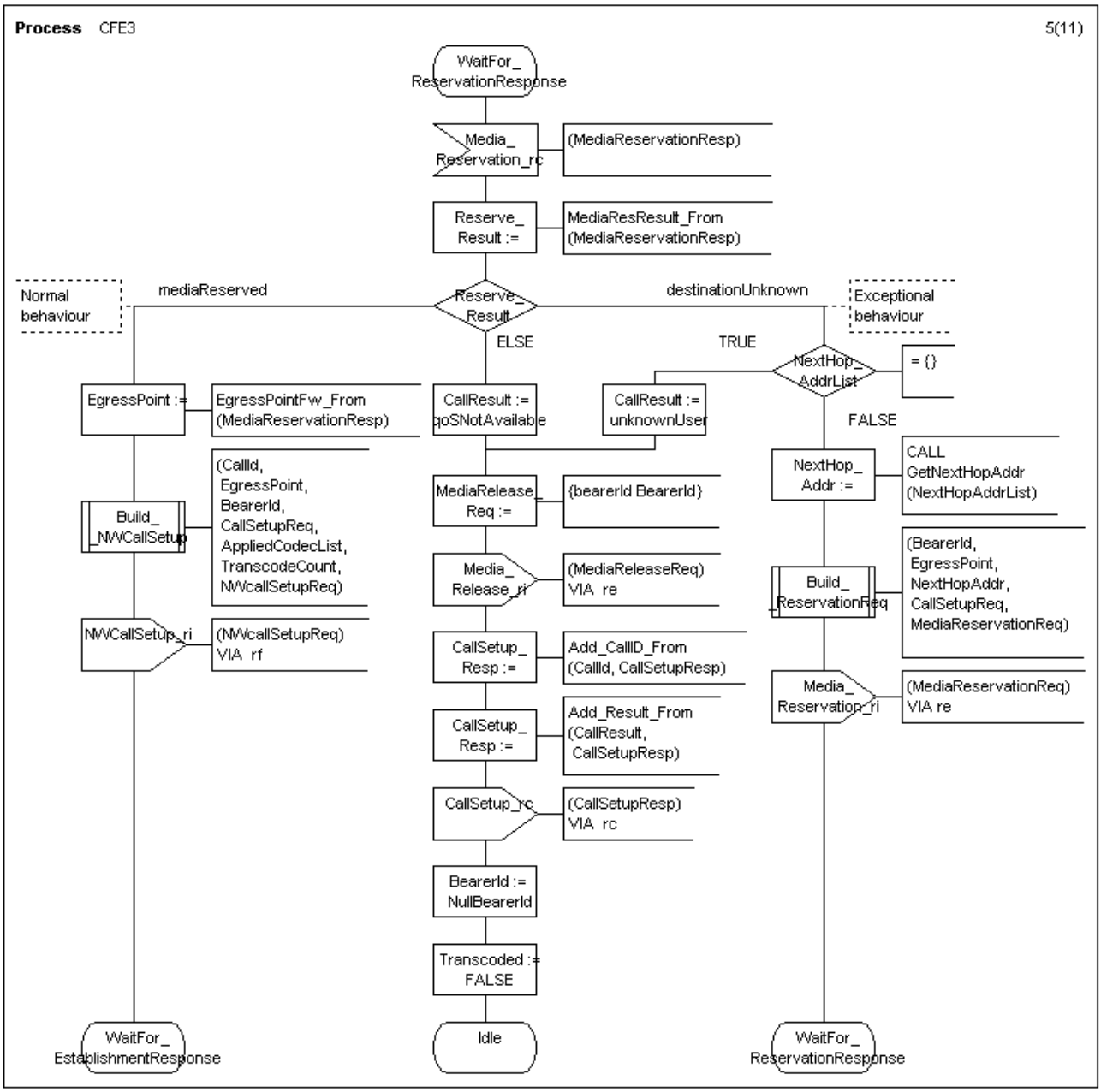


Figure 15 (sheet 5 of 11): SDL process diagram for functional entity $CFE3_{ONC}$

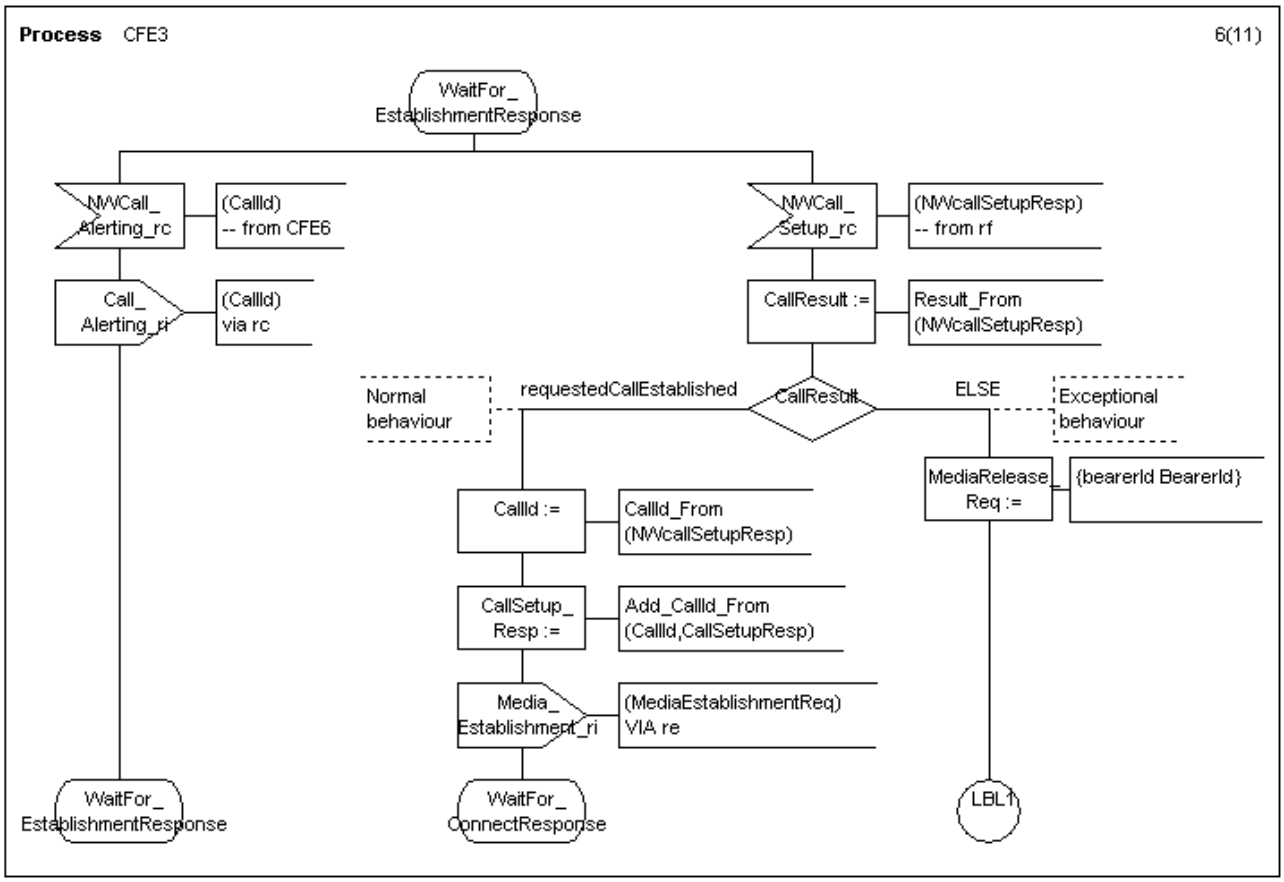


Figure 15 (sheet 6 of 11): SDL process diagram for functional entity $CFE3_{ONC}$

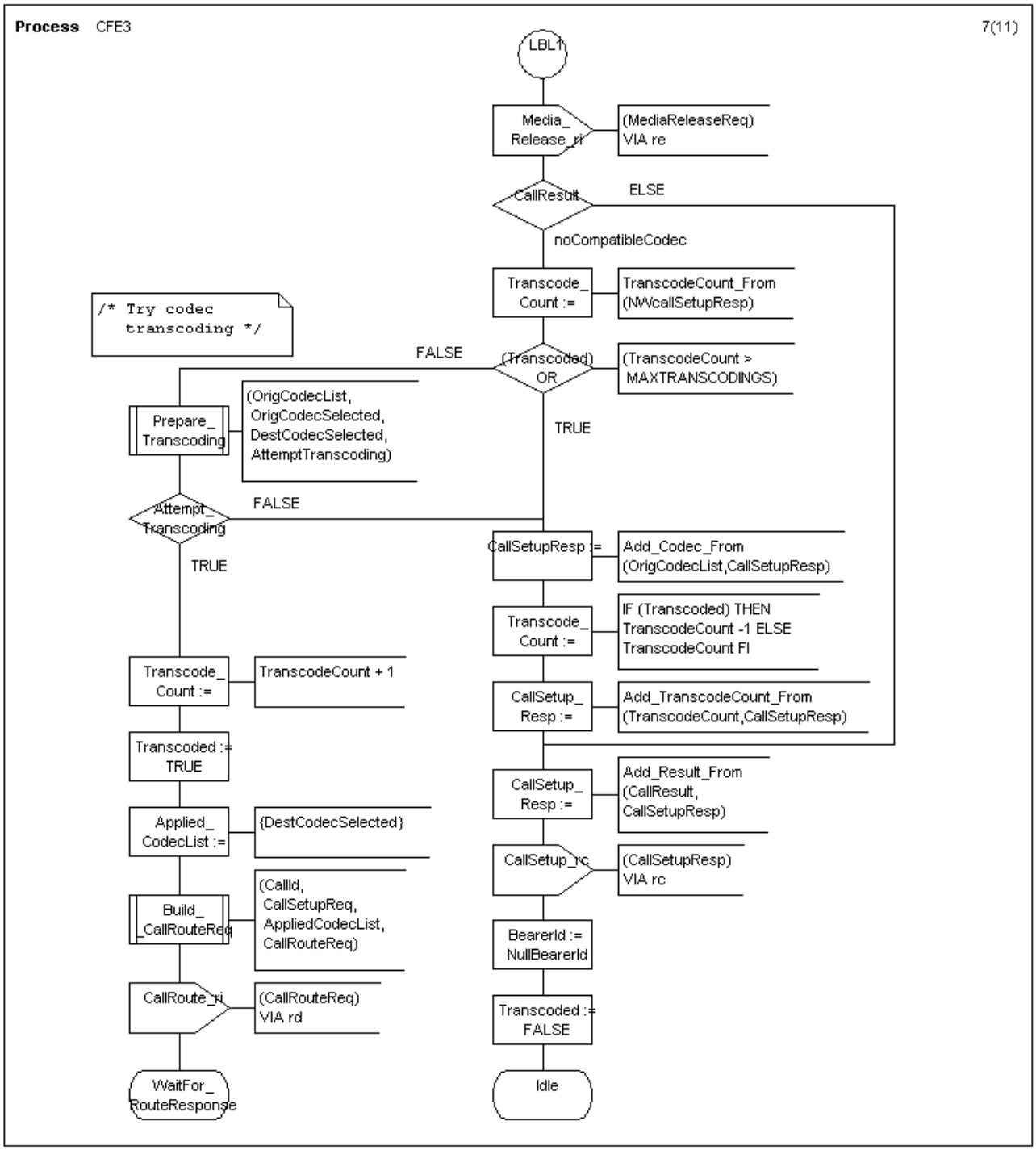


Figure 15 (sheet 7 of 11): SDL process diagram for functional entity $CFE3_{ONC}$

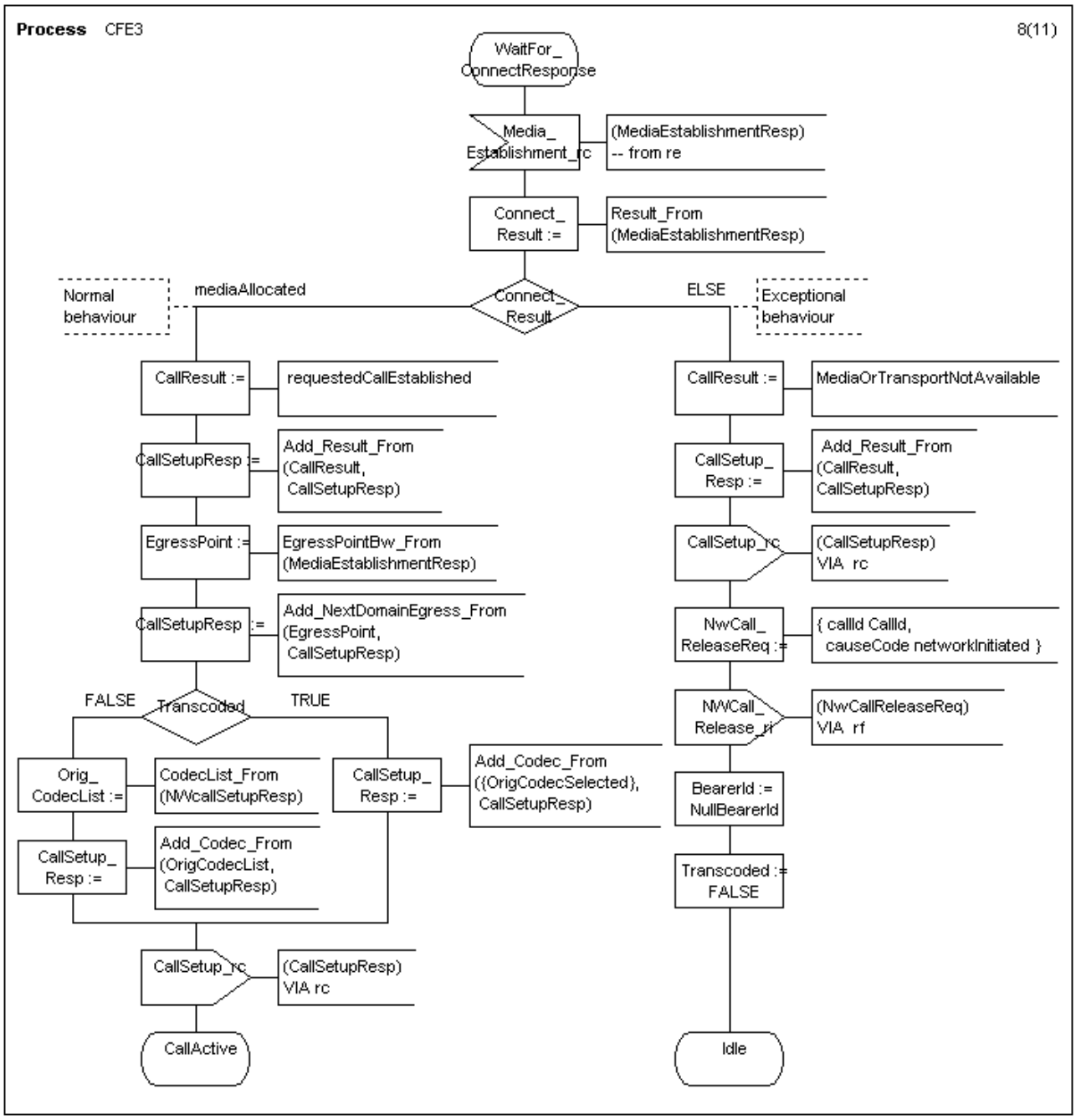


Figure 15 (sheet 8 of 11): SDL process diagram for functional entity $CFE3_{ONC}$

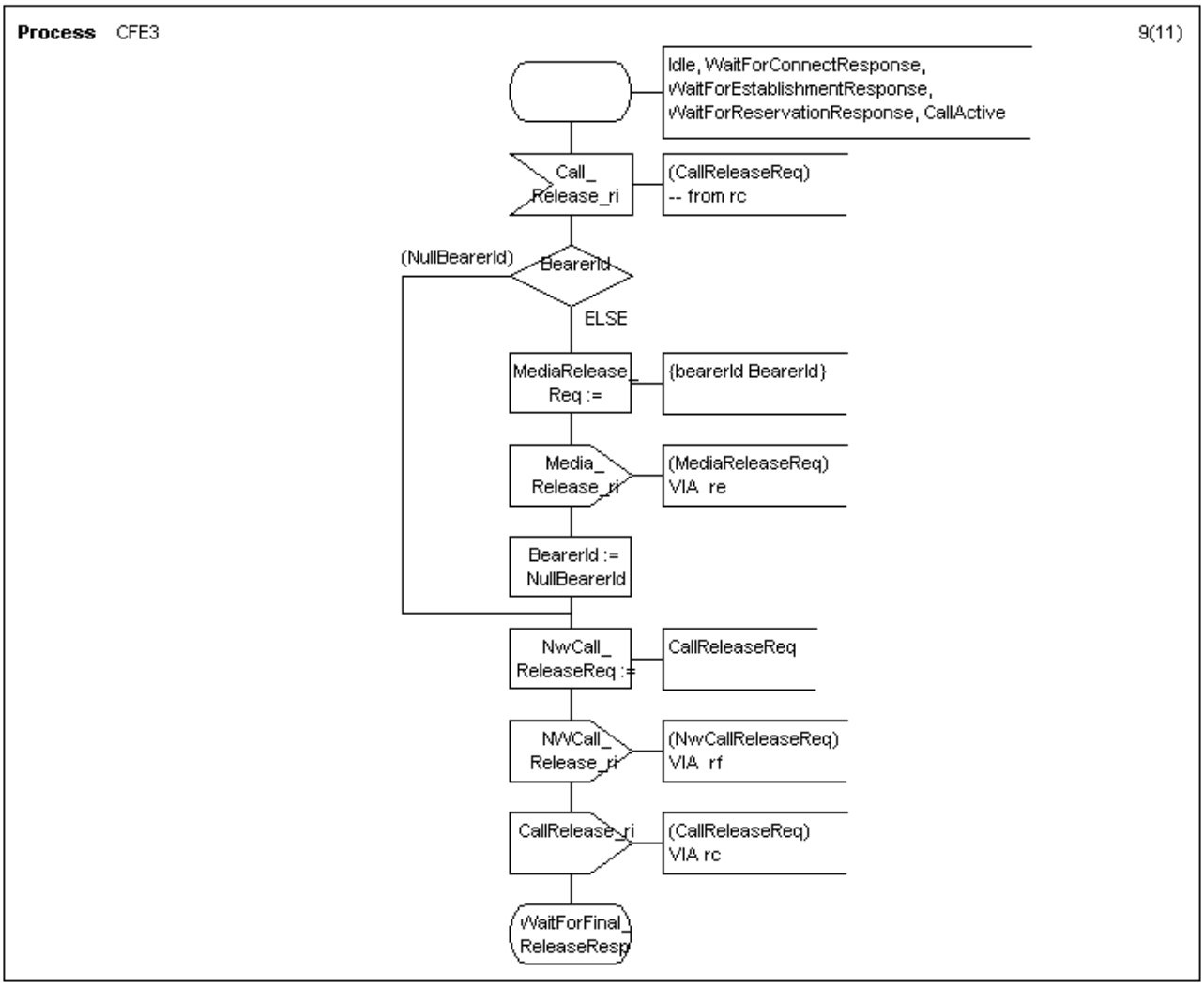


Figure 15 (sheet 9 of 11): SDL process diagram for functional entity $CFE3_{ONC}$

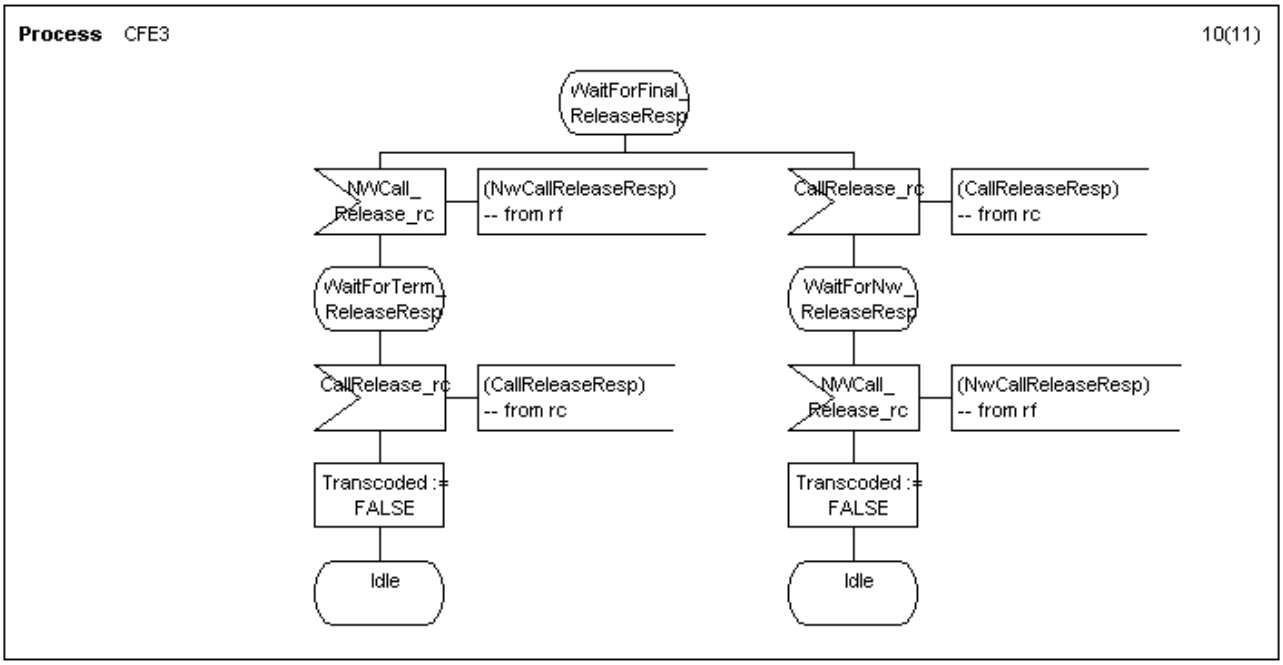


Figure 15 (sheet 10 of 11): SDL process diagram for functional entity $CFE3_{ONC}$

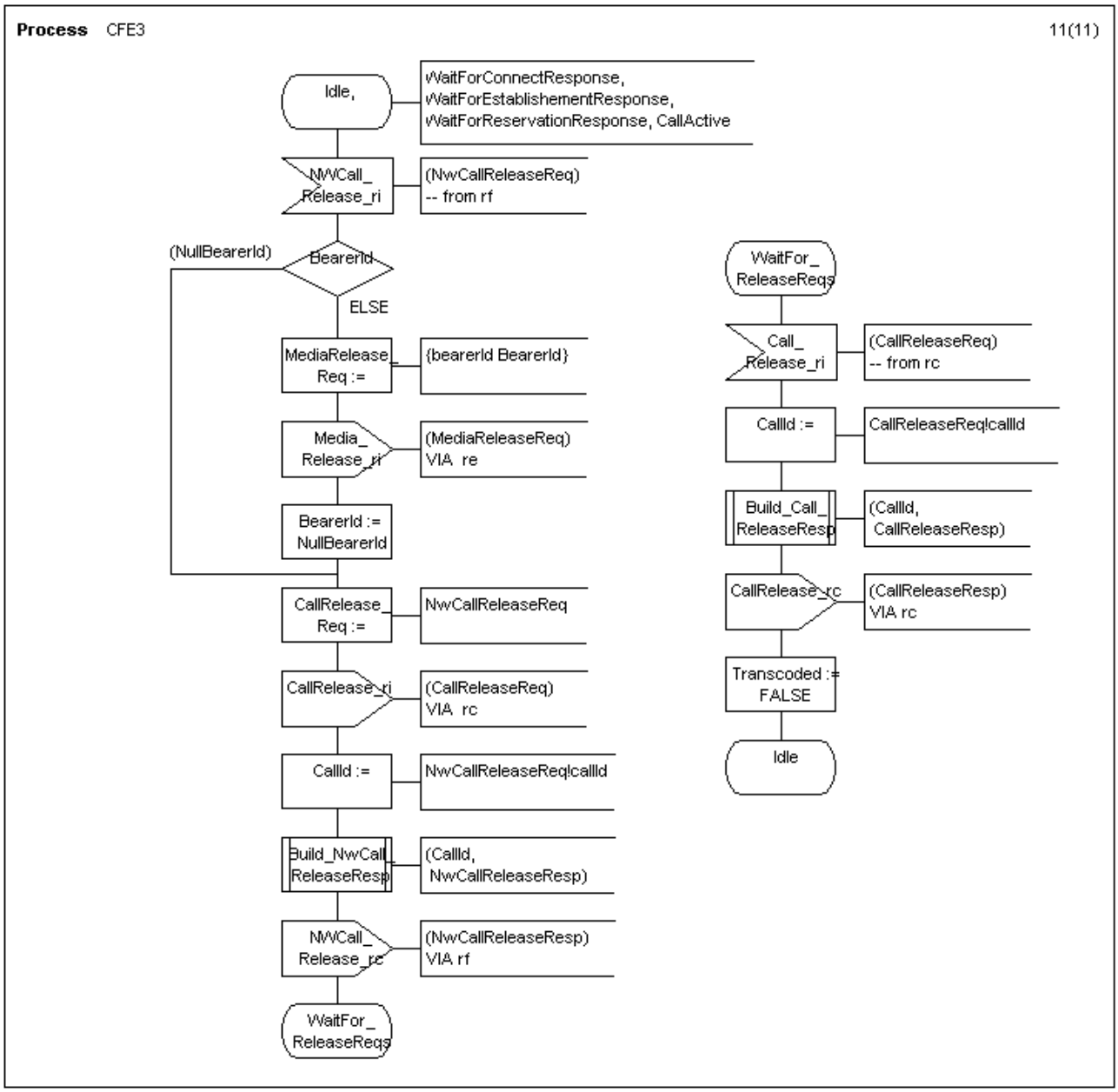


Figure 15 (sheet 11 of 11): SDL process diagram for functional entity $CFE3_{ONC}$

5.4.4 Behaviour of $CFE4_{OR}$ and $CFE7_{IR}$

The behaviour specifications of functional entities $CFE4_{OR}$ and $CFE7_{IR}$ are, for the purposes of the present document, identical. This behaviour is shown in the SDL process type diagram in figure 16.

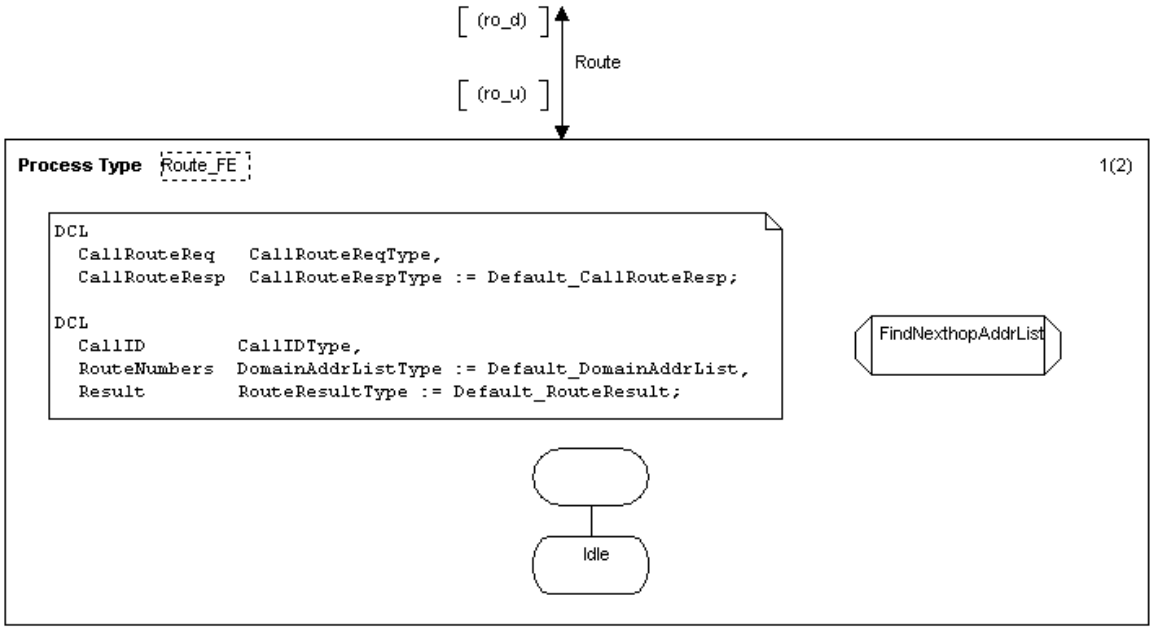


Figure 16 (sheet 1 of 2): SDL process type diagram for functional entities $CFE4_{OR}$ and $CFE7_{IR}$

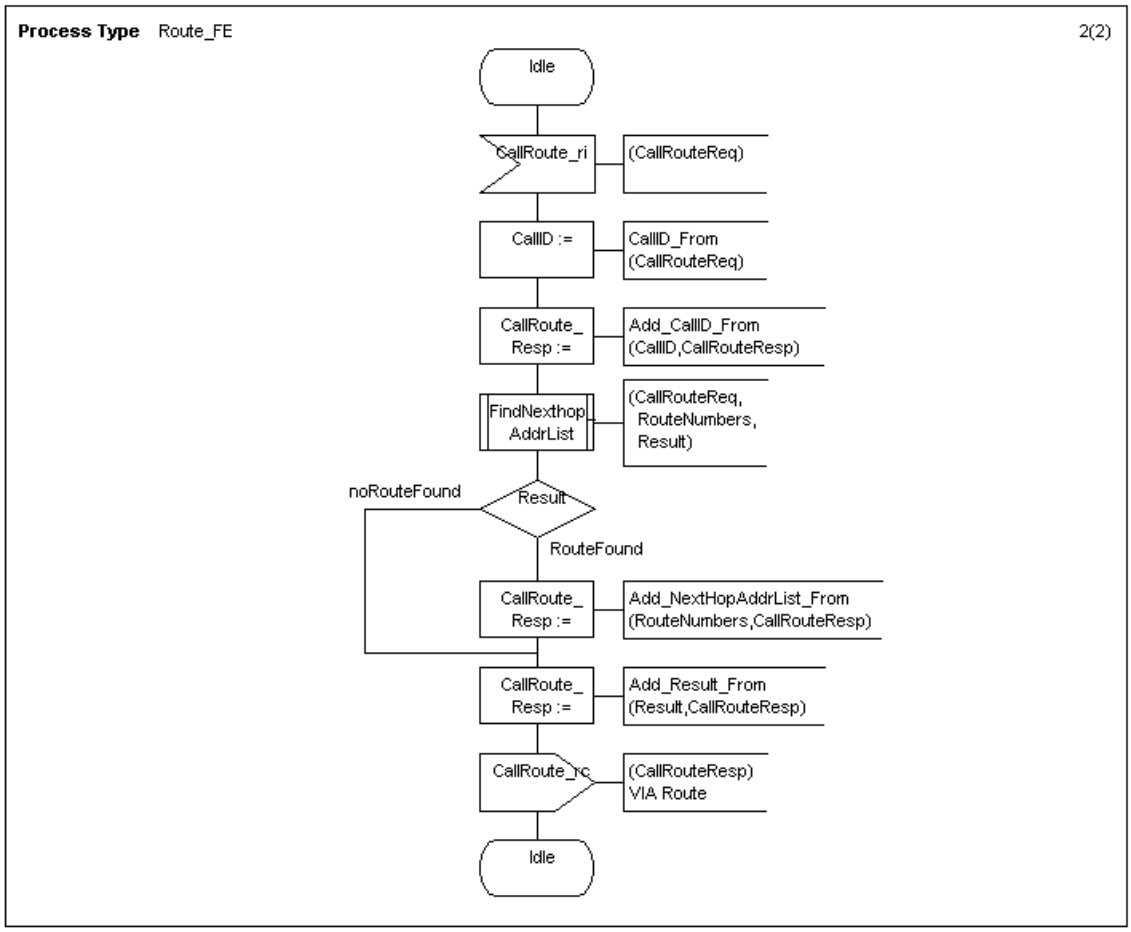


Figure 16 (sheet 1 of 2): SDL process type diagram for functional entities $CFE4_{OR}$ and $CFE7_{IR}$

5.4.5 Behaviour of $CFE5_{ONM}$, $CFE8_{INM}$, and $CFE10_{TNM}$

The behaviour specifications of functional entities $CFE5_{ONM}$, $CFE8_{INM}$, and $CFE10_{TNM}$ are, for the purposes of the present document, identical. This behaviour is shown in the SDL process type diagram figure 17.

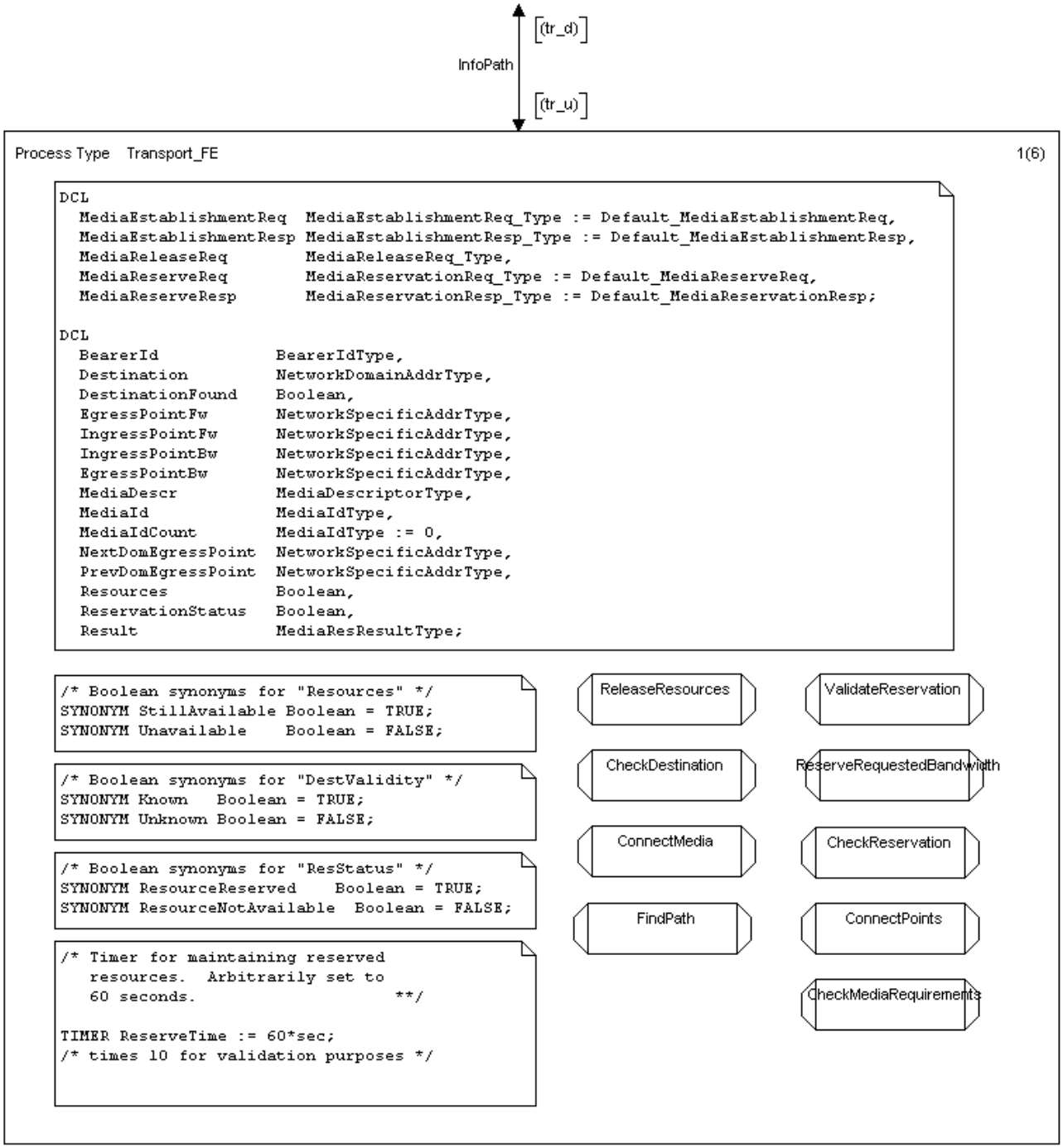


Figure 17 (sheet 1 of 6): SDL process type diagram for functional entities $CFE5_{ONM}$, $CFE8_{INM}$, and $CFE10_{TNM}$

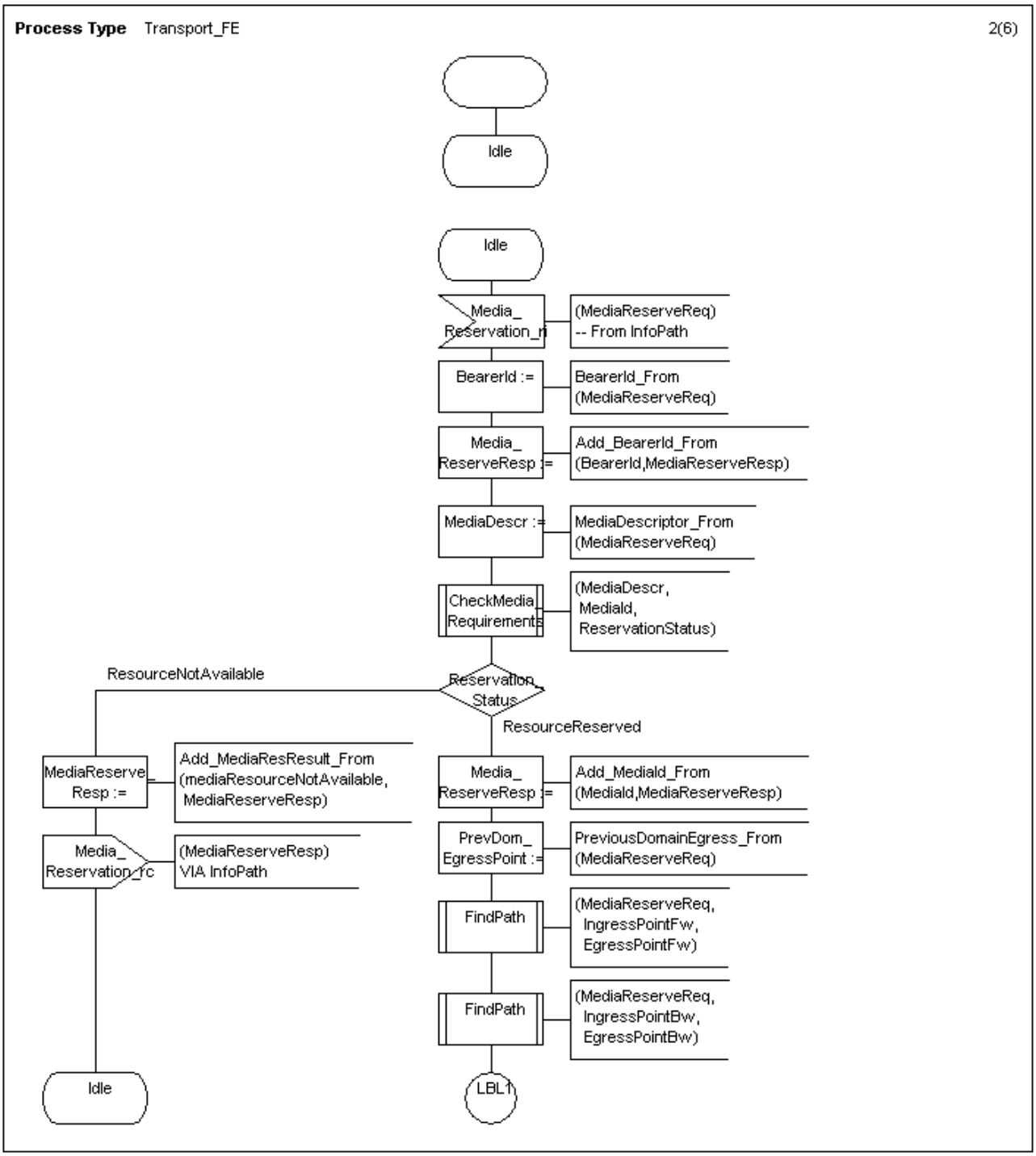


Figure 17 (sheet 2 of 6): SDL process type diagram for functional entities $CFE5_{ONM}$, $CFE8_{INM}$, and $CFE10_{TNM}$

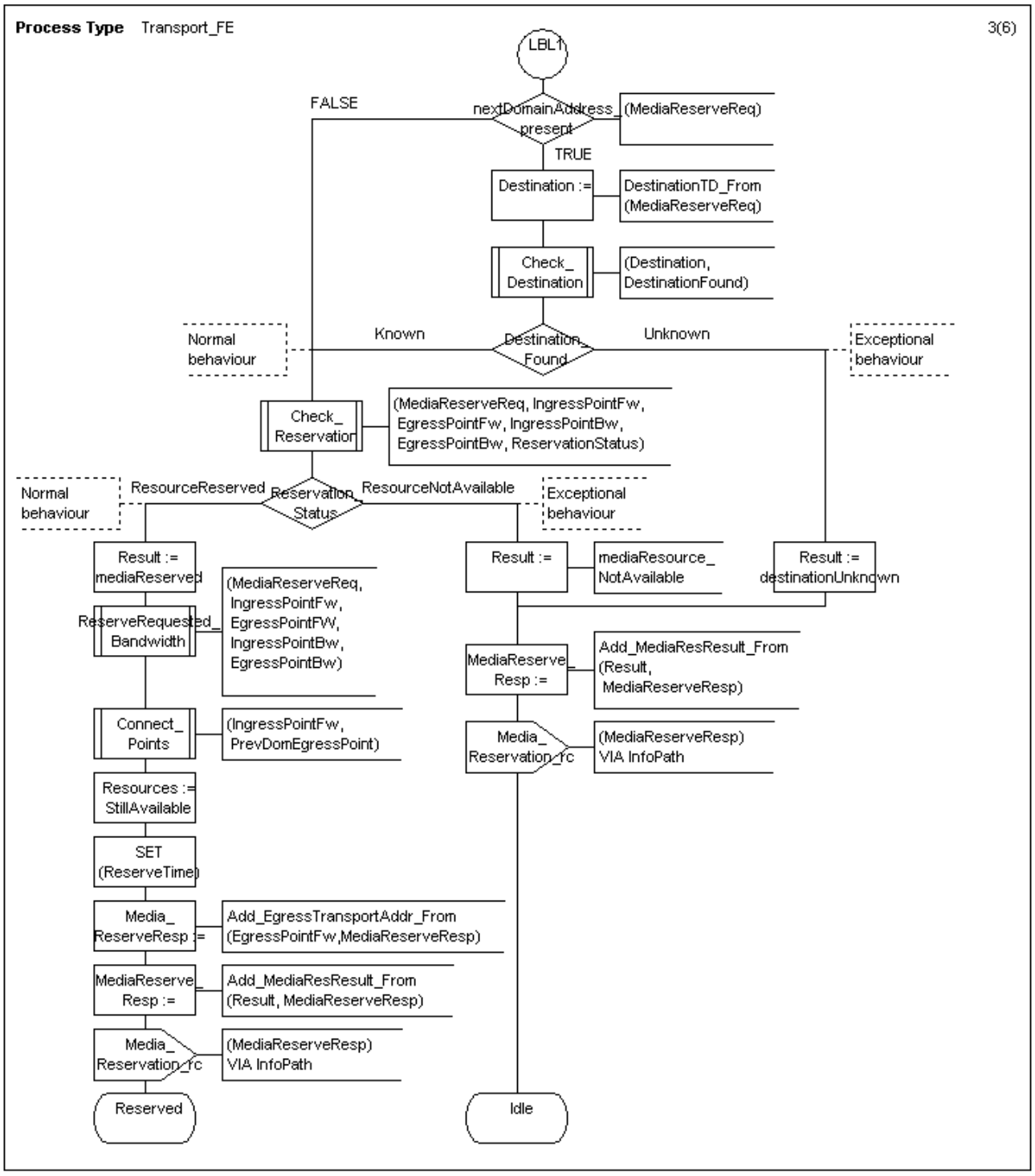


Figure 17 (sheet 3 of 6): SDL process type diagram for functional entities $CFE5_{ONM}$, $CFE8_{INM}$, and $CFE10_{TNM}$

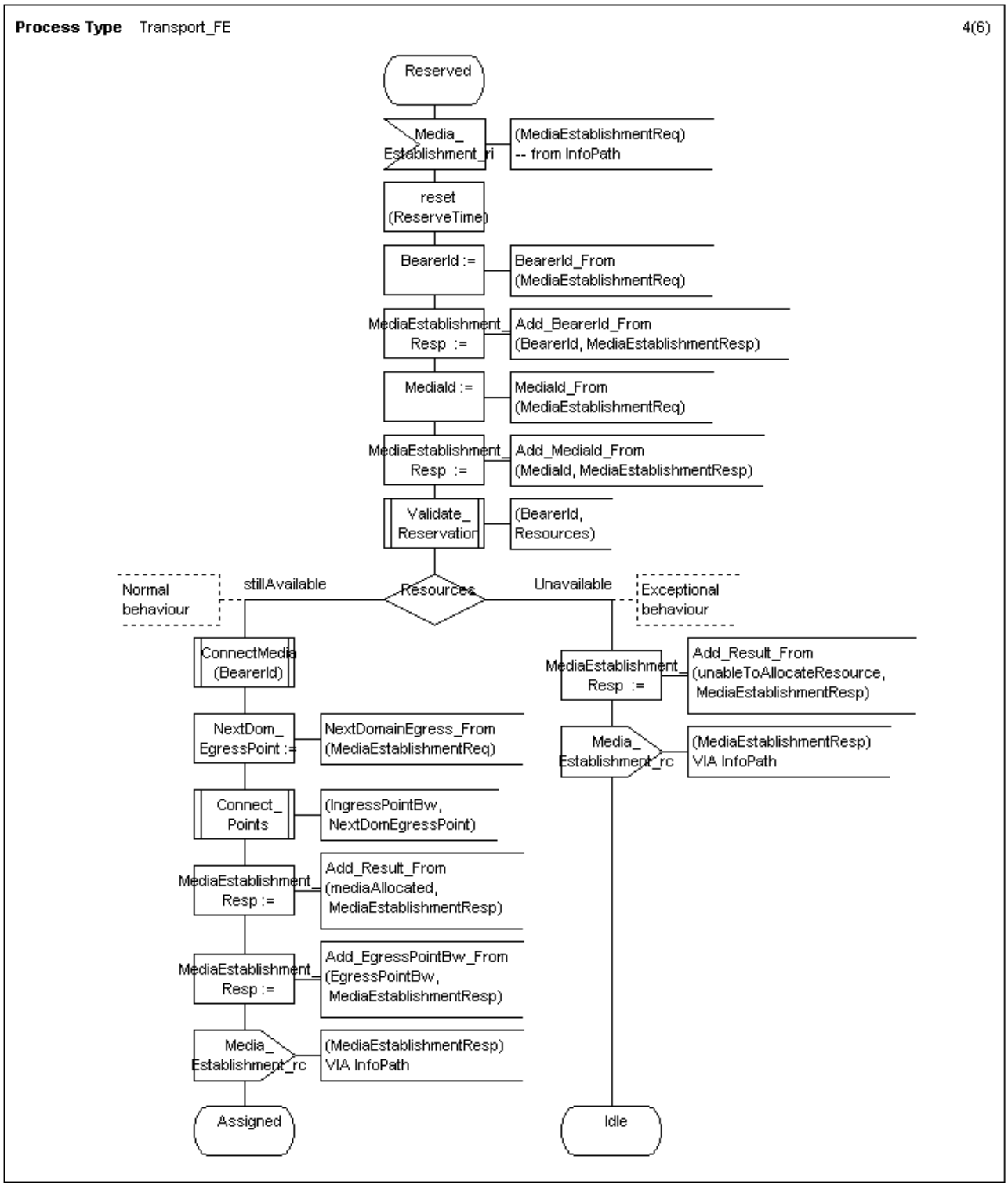


Figure 17 (sheet 4 of 6): SDL process type diagram for functional entities $CFE5_{ONM}$, $CFE8_{INM}$, and $CFE10_{TNM}$

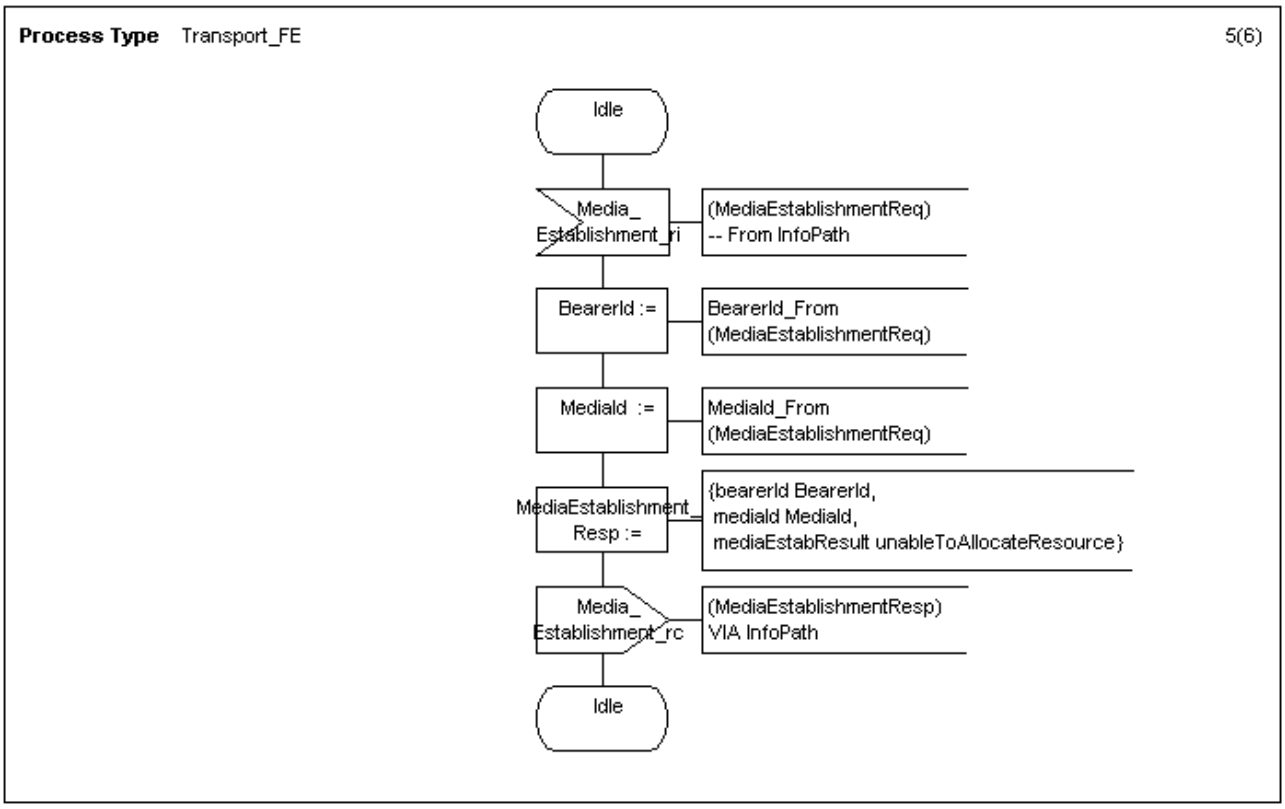


Figure 17 (sheet 5 of 6): SDL process type diagram for functional entities $CFE5_{ONM}$, $CFE8_{INM}$, and $CFE10_{TNM}$

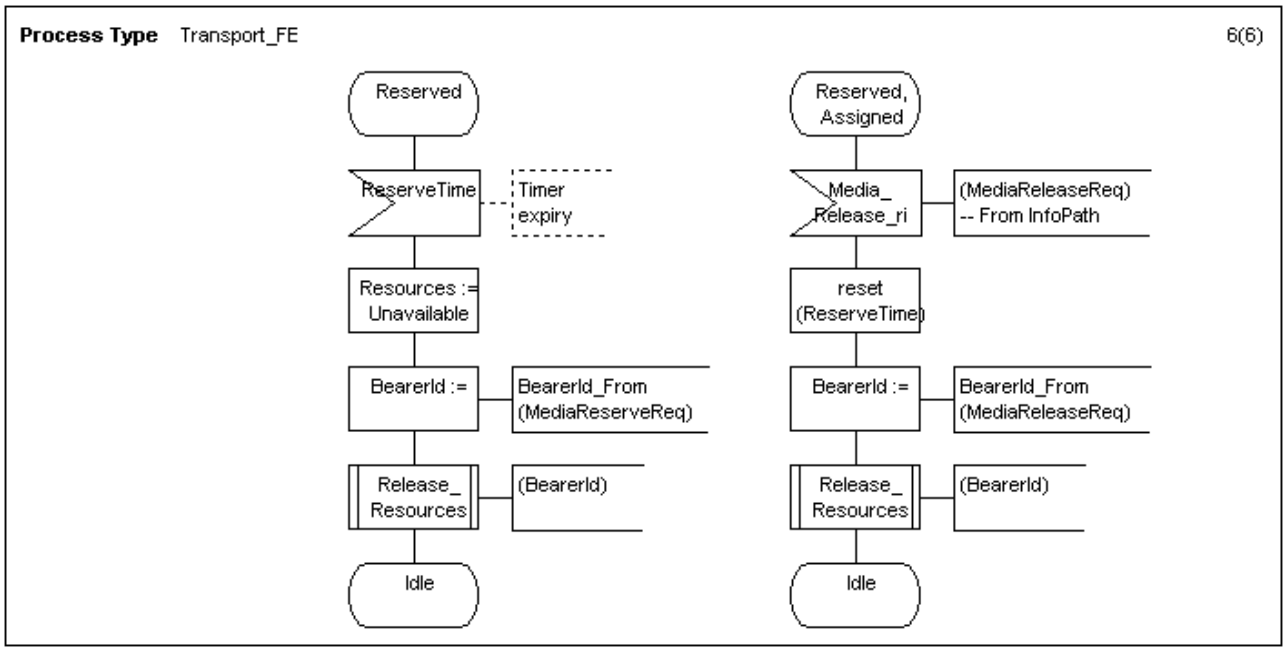


Figure 17 (sheet 6 of 6): SDL process type diagram for functional entities $CFE5_{ONM}$, $CFE8_{INM}$, and $CFE10_{TNM}$

5.4.6 Behaviour of $CFE6_{INC}$

The behaviour of $CFE6_{INC}$ is shown in the SDL process diagram in figure 18.

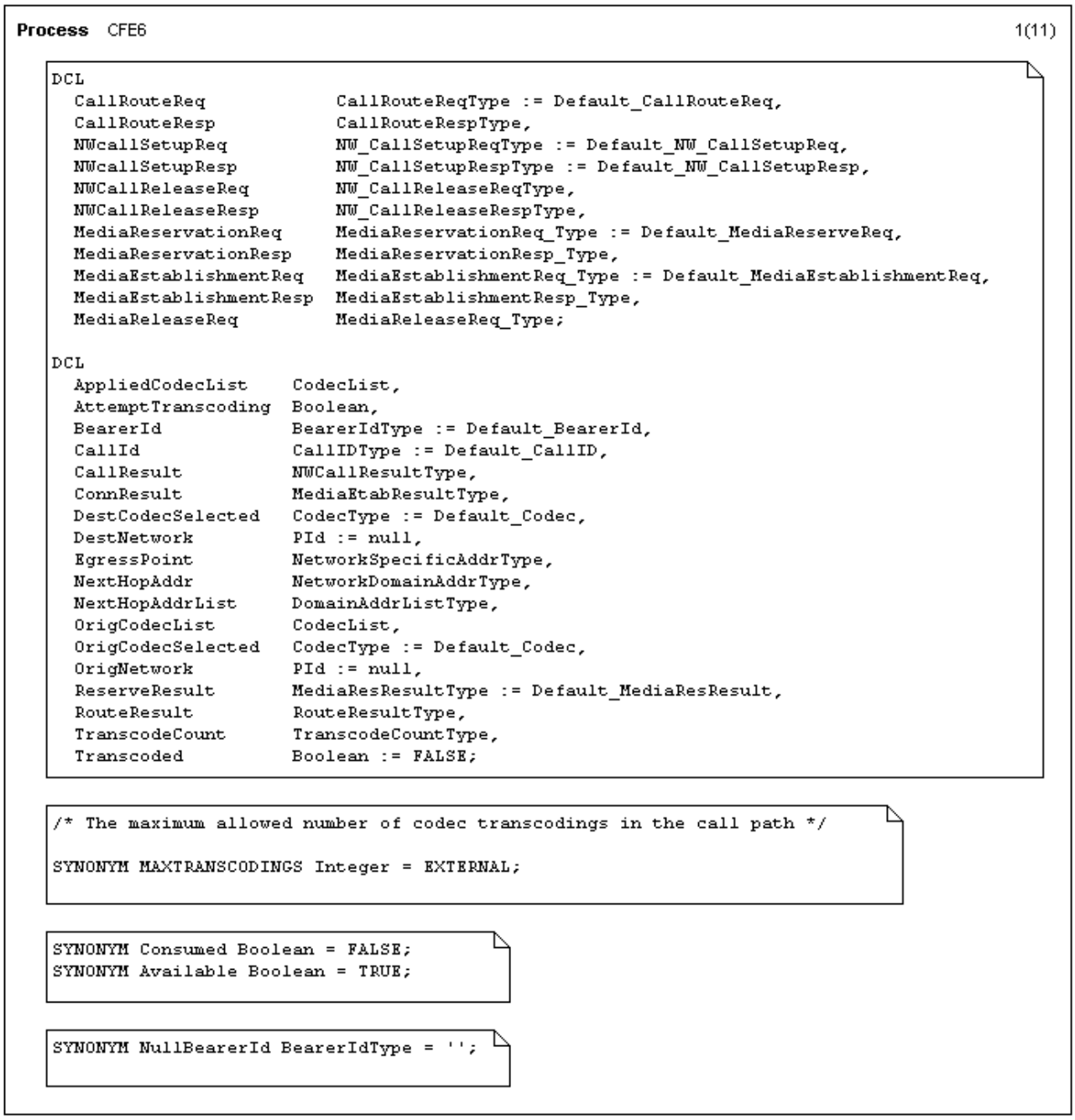


Figure 18 (sheet 1 of 11): SDL process diagram for functional entity $CFE6_{INC}$

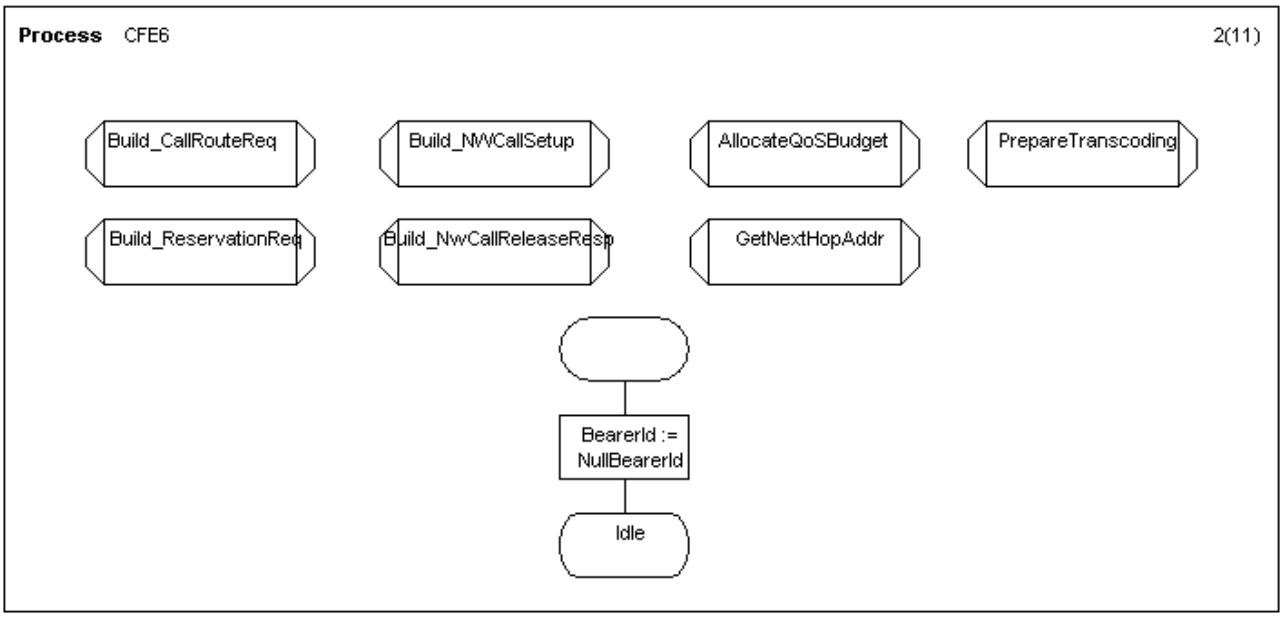


Figure 18 (sheet 2 of 11): SDL process diagram for functional entity CFE6$_{INC}$

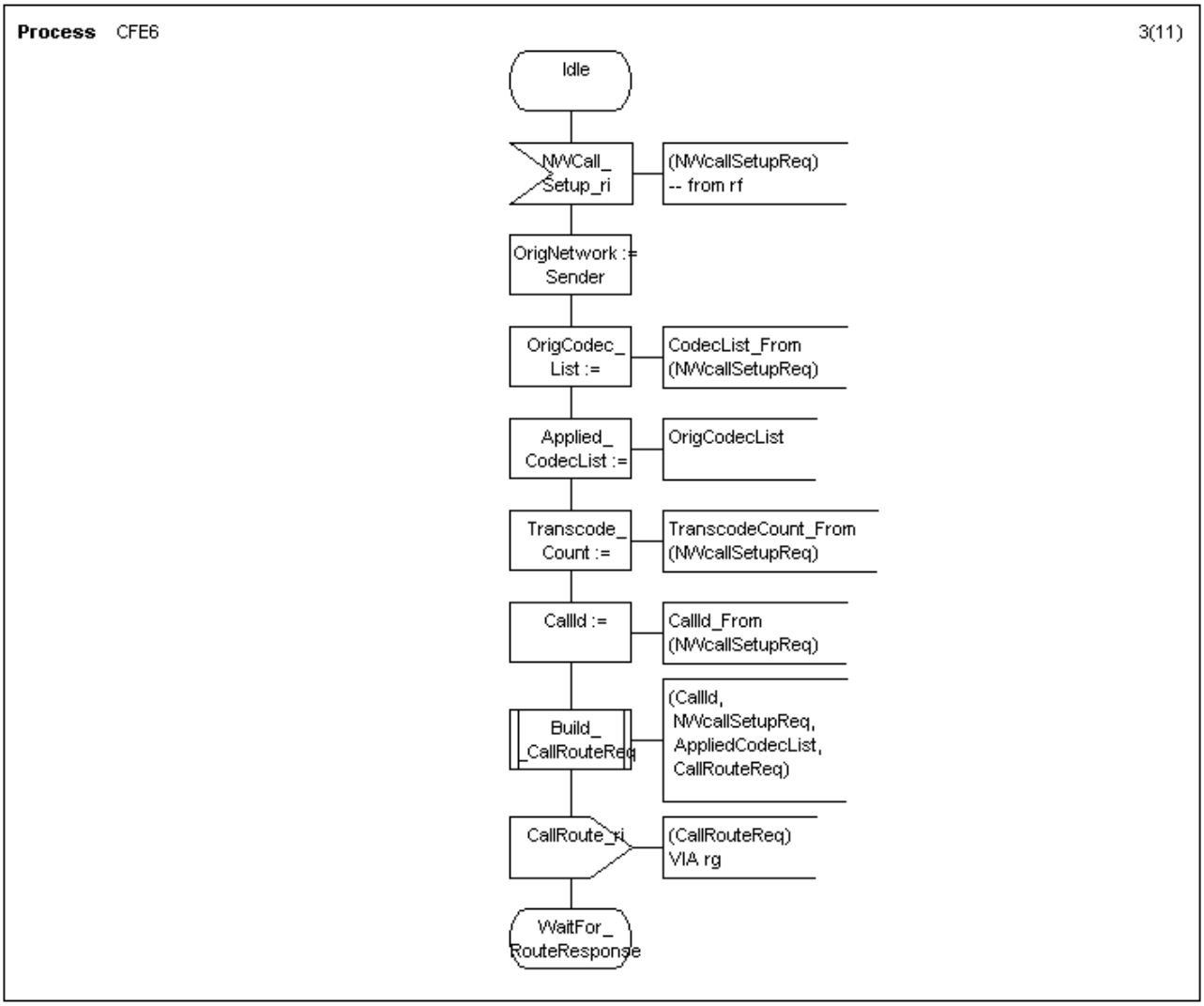


Figure 18 (sheet 3 of 11): SDL process diagram for functional entity CFE6$_{INC}$

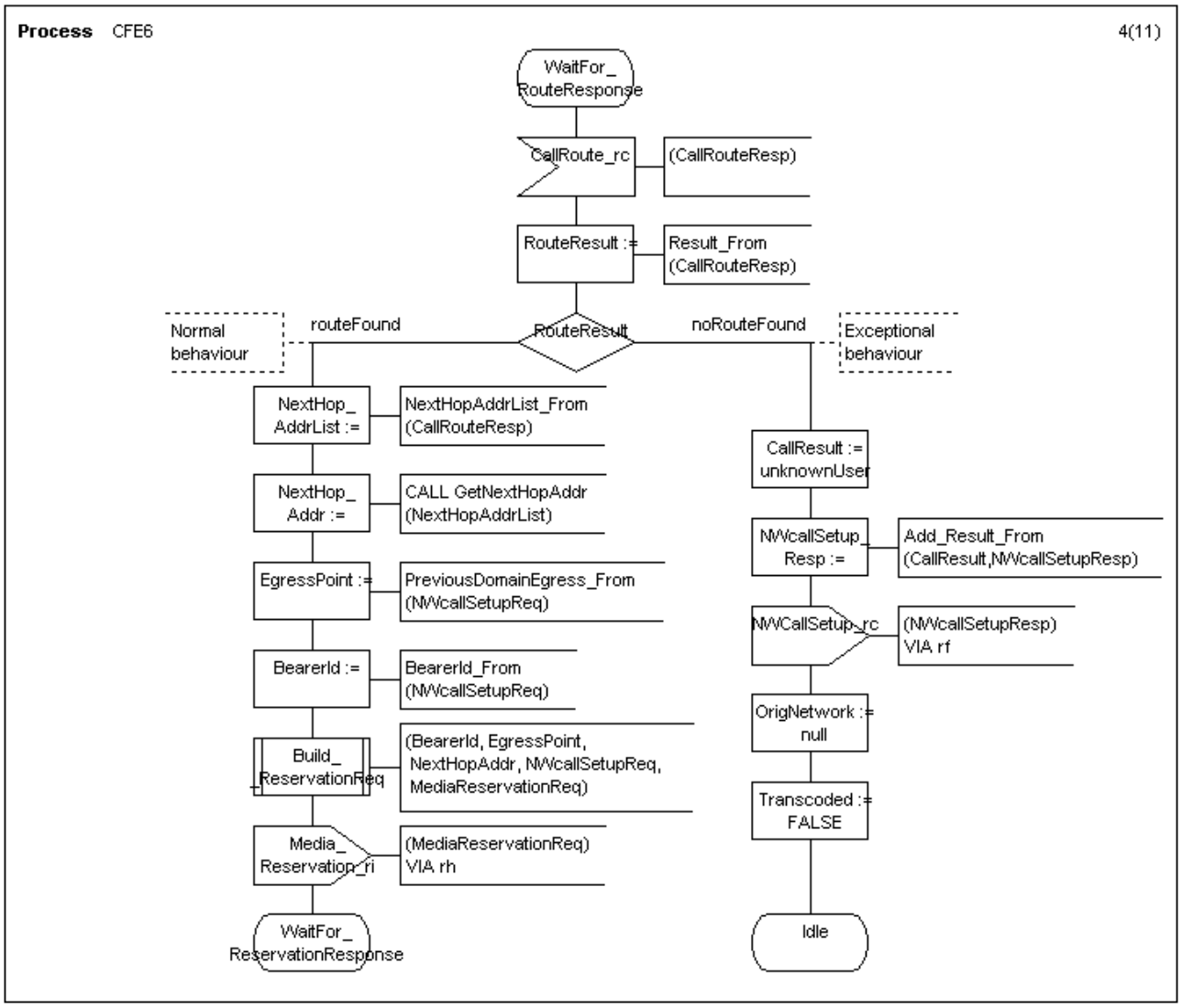


Figure 18 (sheet 4 of 11): SDL process diagram for functional entity $CFE6_{INC}$

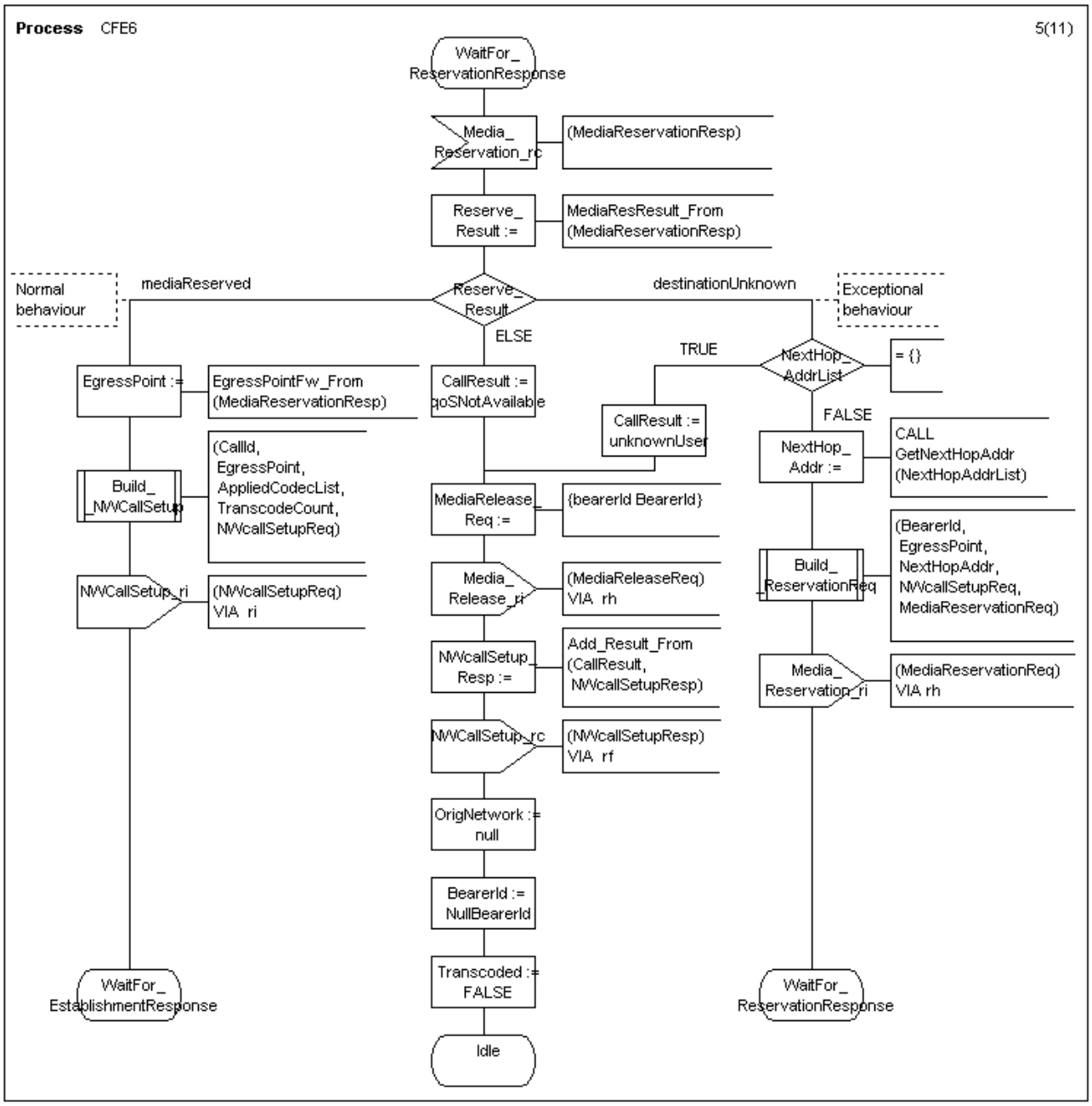


Figure 18 (sheet 5 of 11): SDL process diagram for functional entity CFE6$_{INC}$

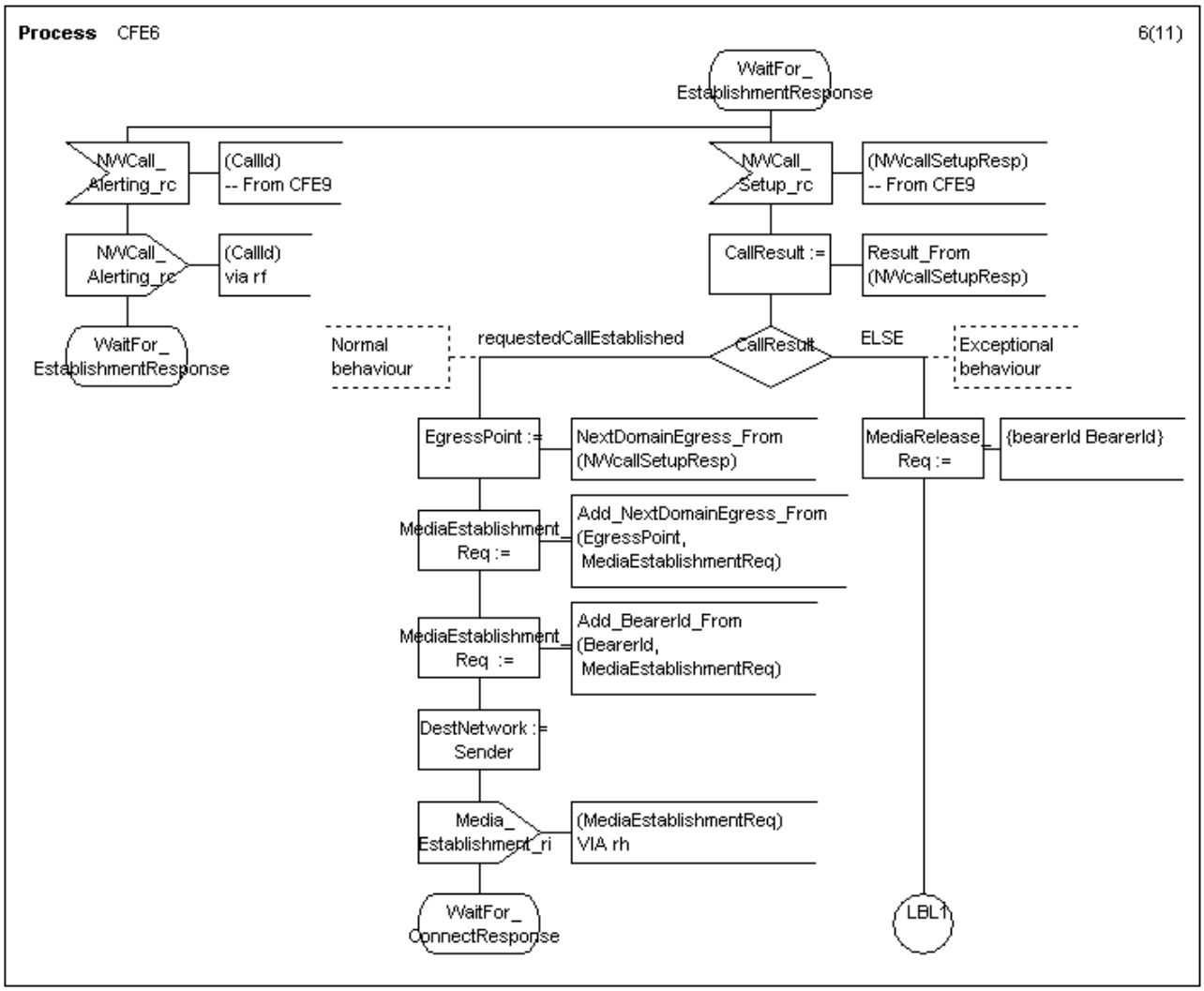


Figure 18 (sheet 6 of 11): SDL process diagram for functional entity $CFE6_{INC}$

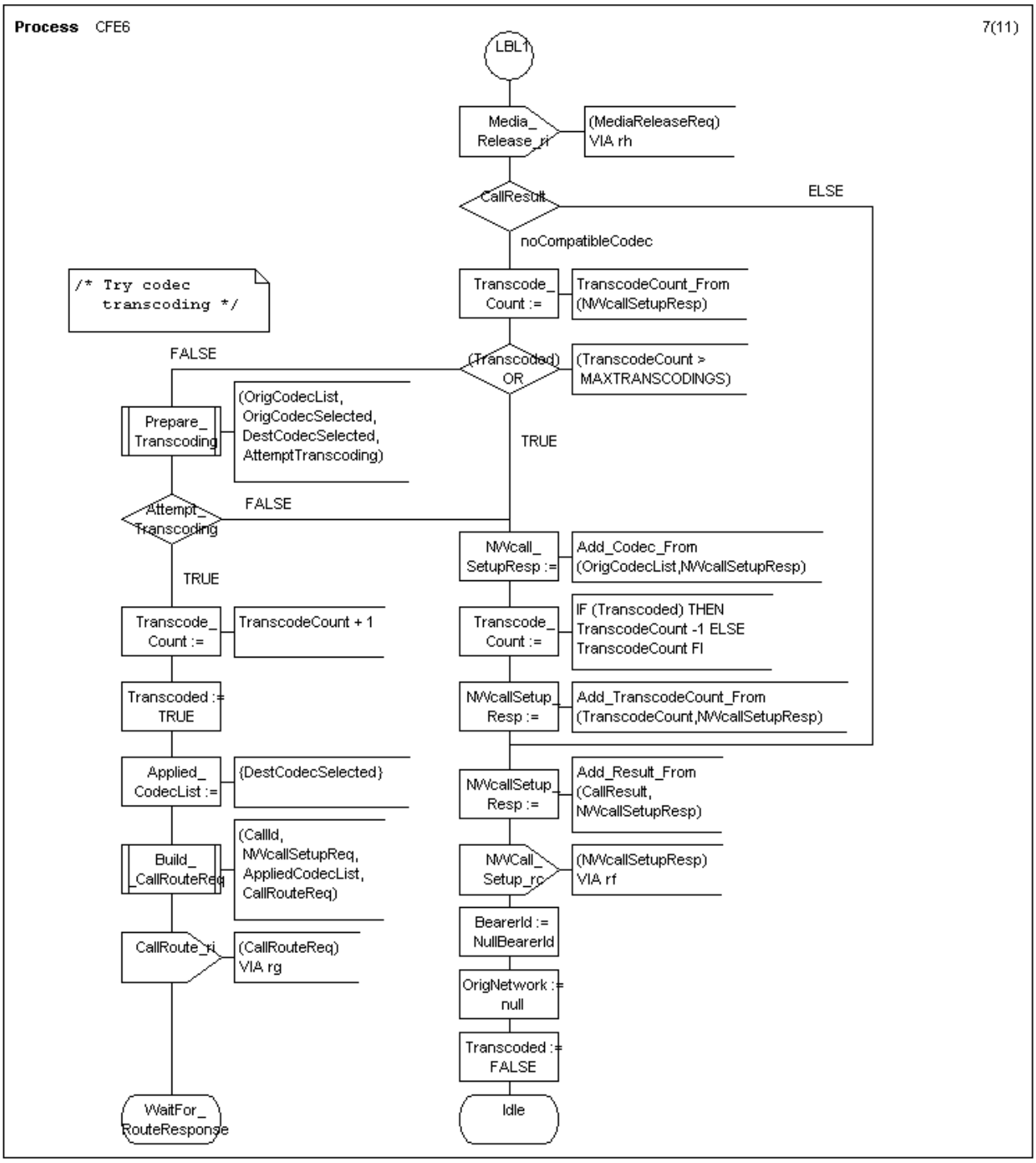


Figure 18 (sheet 7 of 11): SDL process diagram for functional entity $CFE6_{INC}$

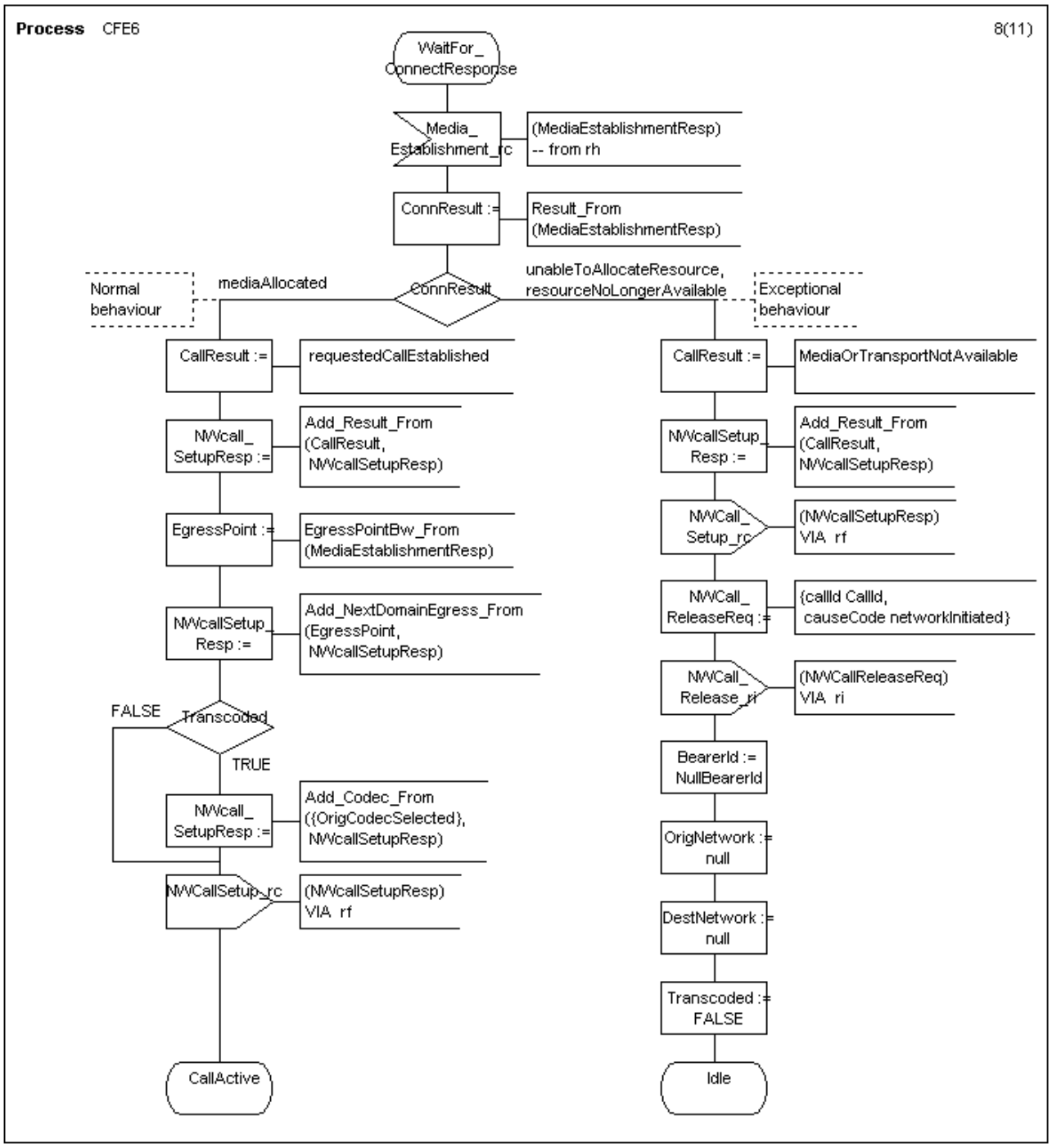


Figure 18 (sheet 8 of 11): SDL process diagram for functional entity $CFE6_{INC}$

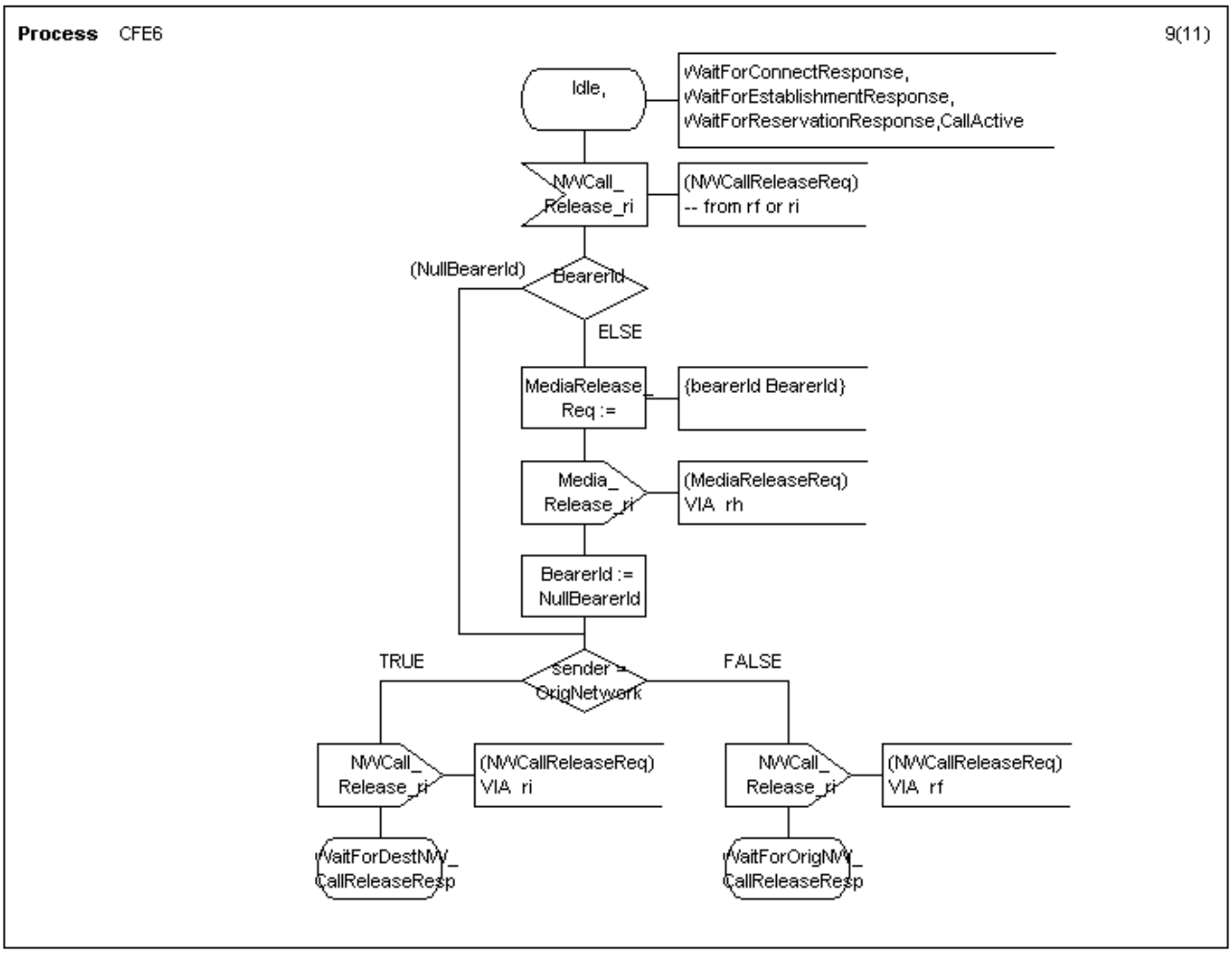


Figure 18 (sheet 9 of 11): SDL process diagram for functional entity $CFE6_{INC}$

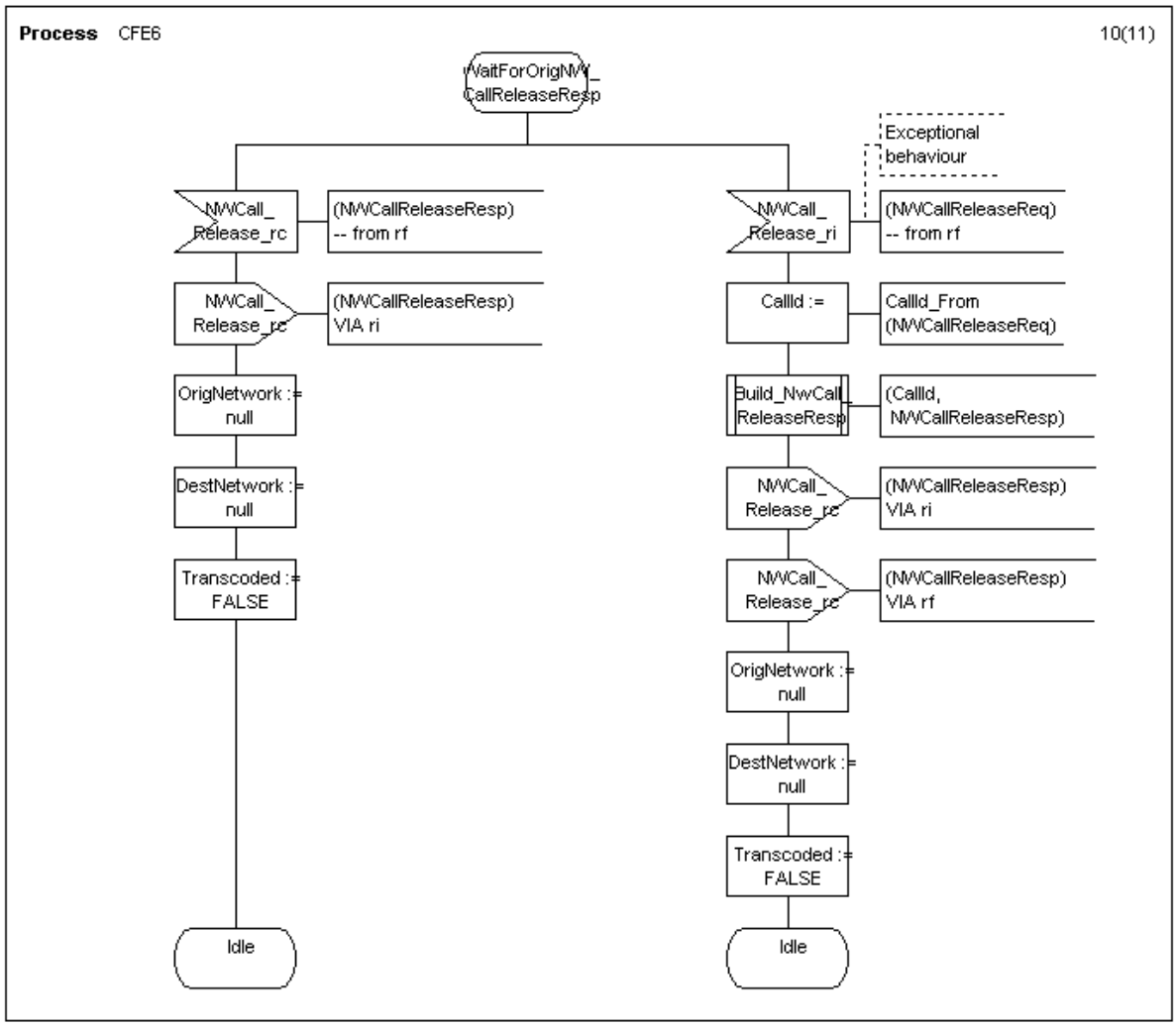


Figure 18 (sheet 10 of 11): SDL process diagram for functional entity $CFE6_{INC}$

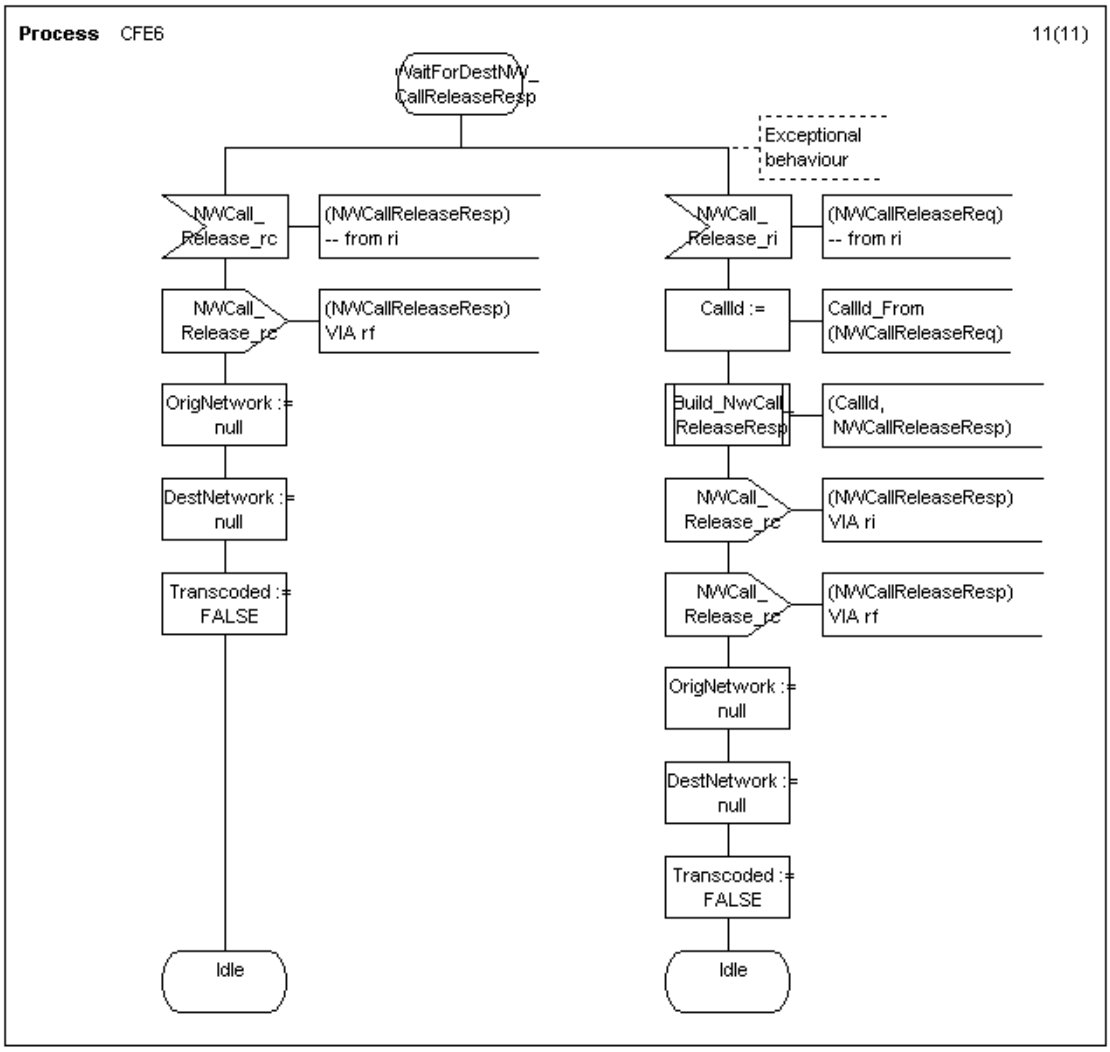


Figure 18 (sheet 11 of 11): SDL process diagram for functional entity $CFE6_{INC}$

5.4.7 Behaviour of $CFE9_{TNC}$

The behaviour of $CFE9_{TNC}$ is shown in the SDL process diagram in figure 19.

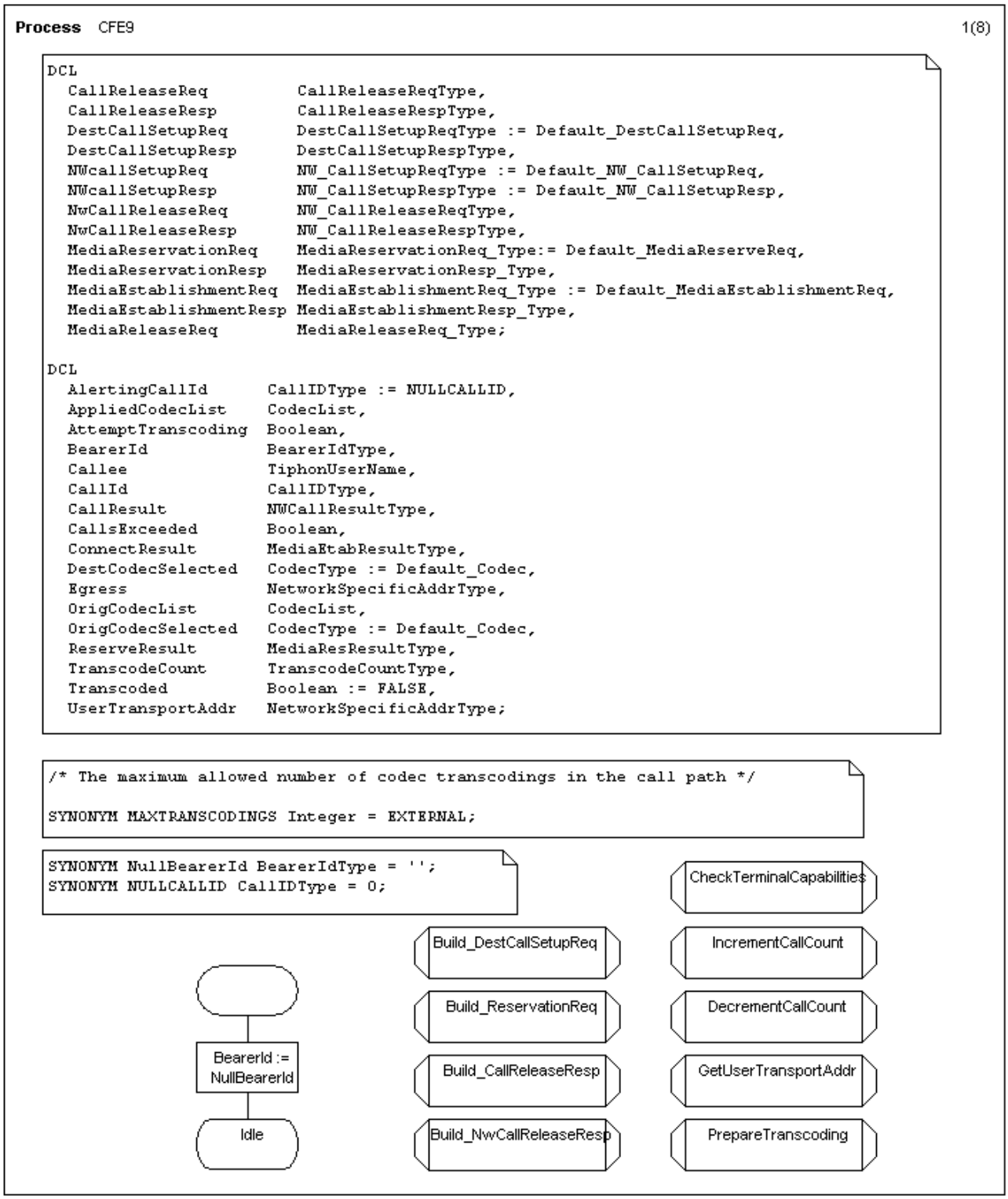


Figure 19 (sheet 1 of 8): SDL process diagram for functional entity $CFE9_{TNC}$

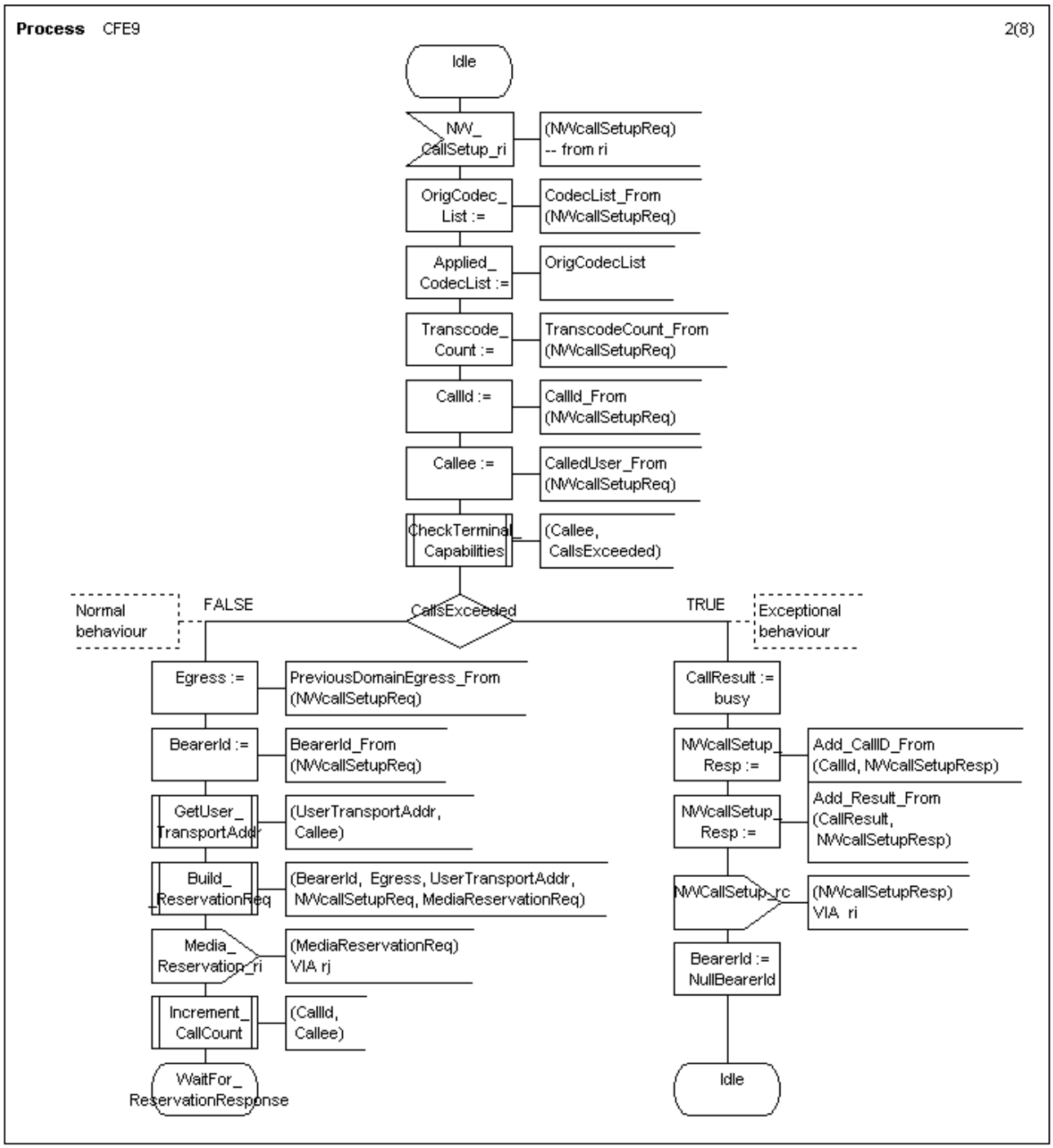


Figure 19 (sheet 2 of 8): SDL process diagram for functional entity $CFE9_{TNC}$

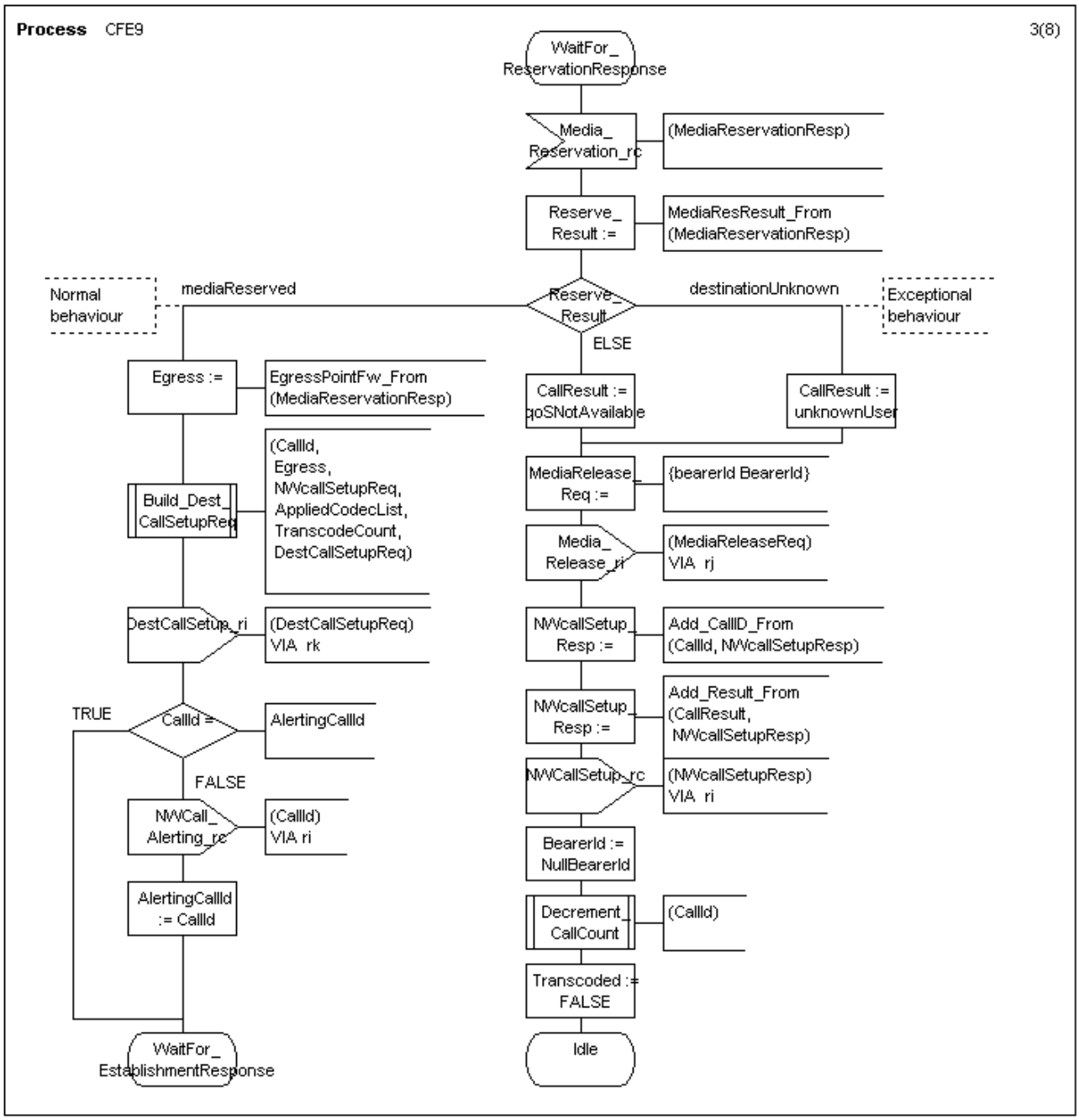


Figure 19 (sheet 3 of 8): SDL process diagram for functional entity $CFE9_{TNC}$

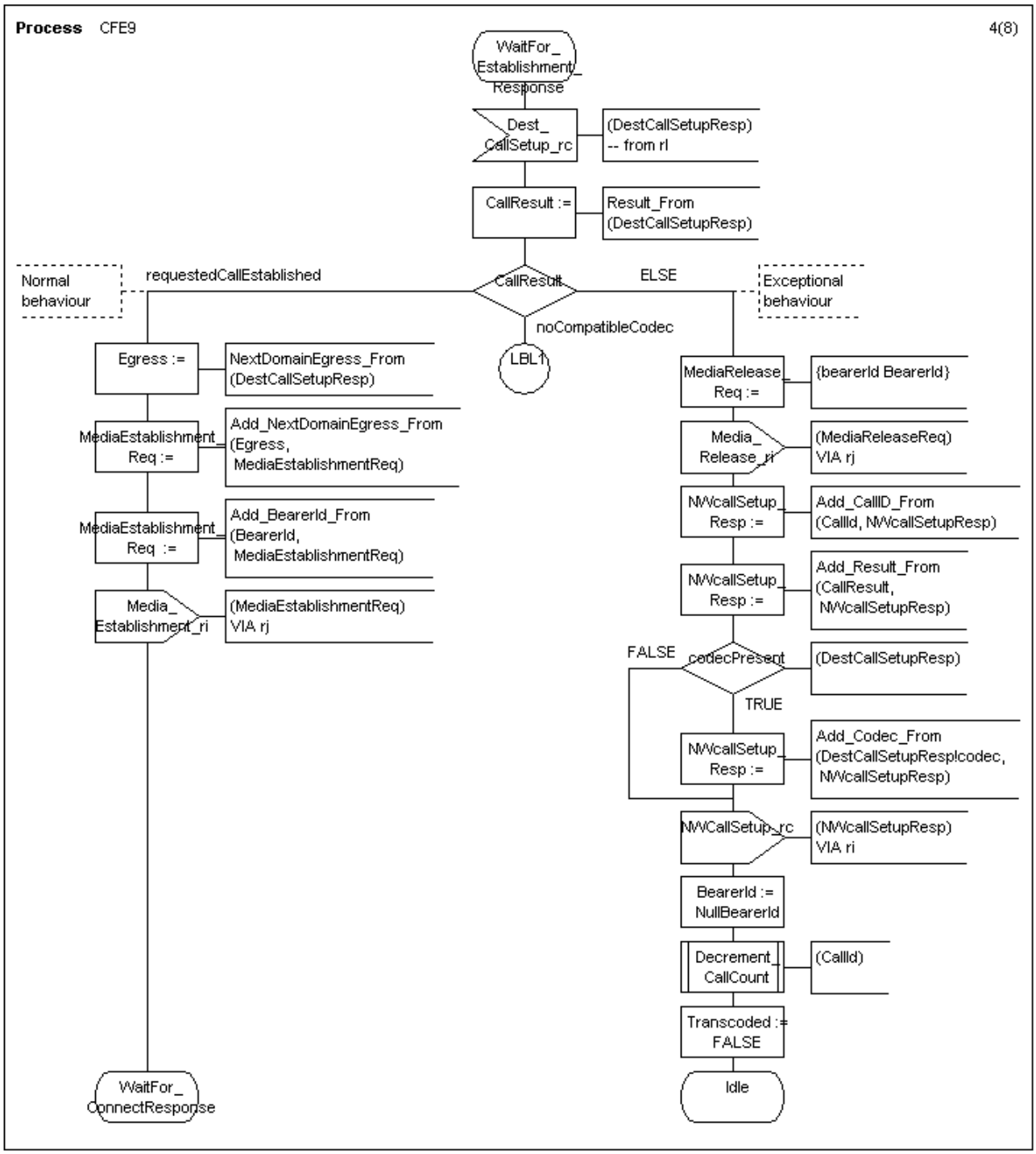


Figure 19 (sheet 4 of 8): SDL process diagram for functional entity CFE9$_{TNC}$

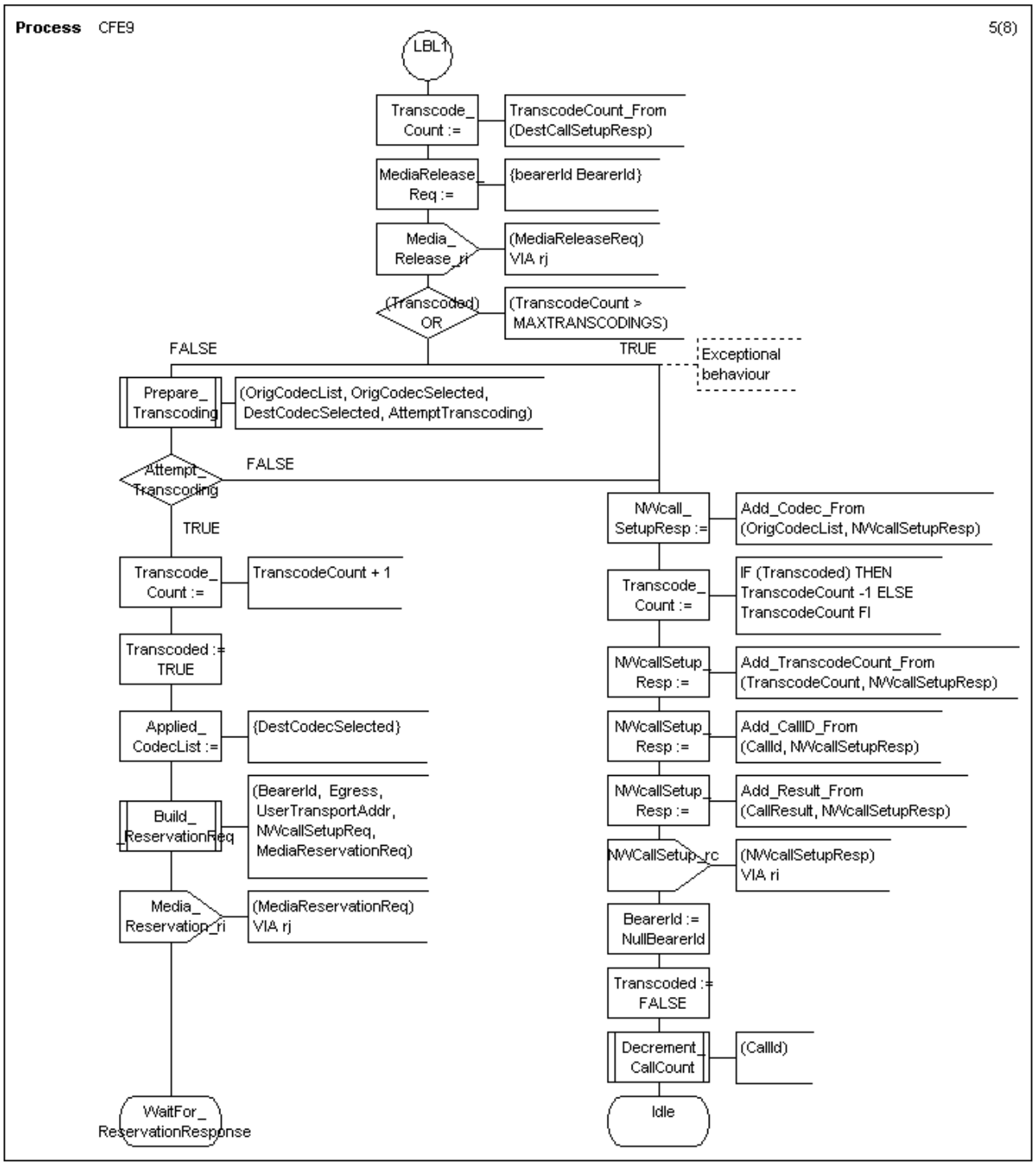


Figure 19 (sheet 5 of 8): SDL process diagram for functional entity $CFE9_{TNC}$

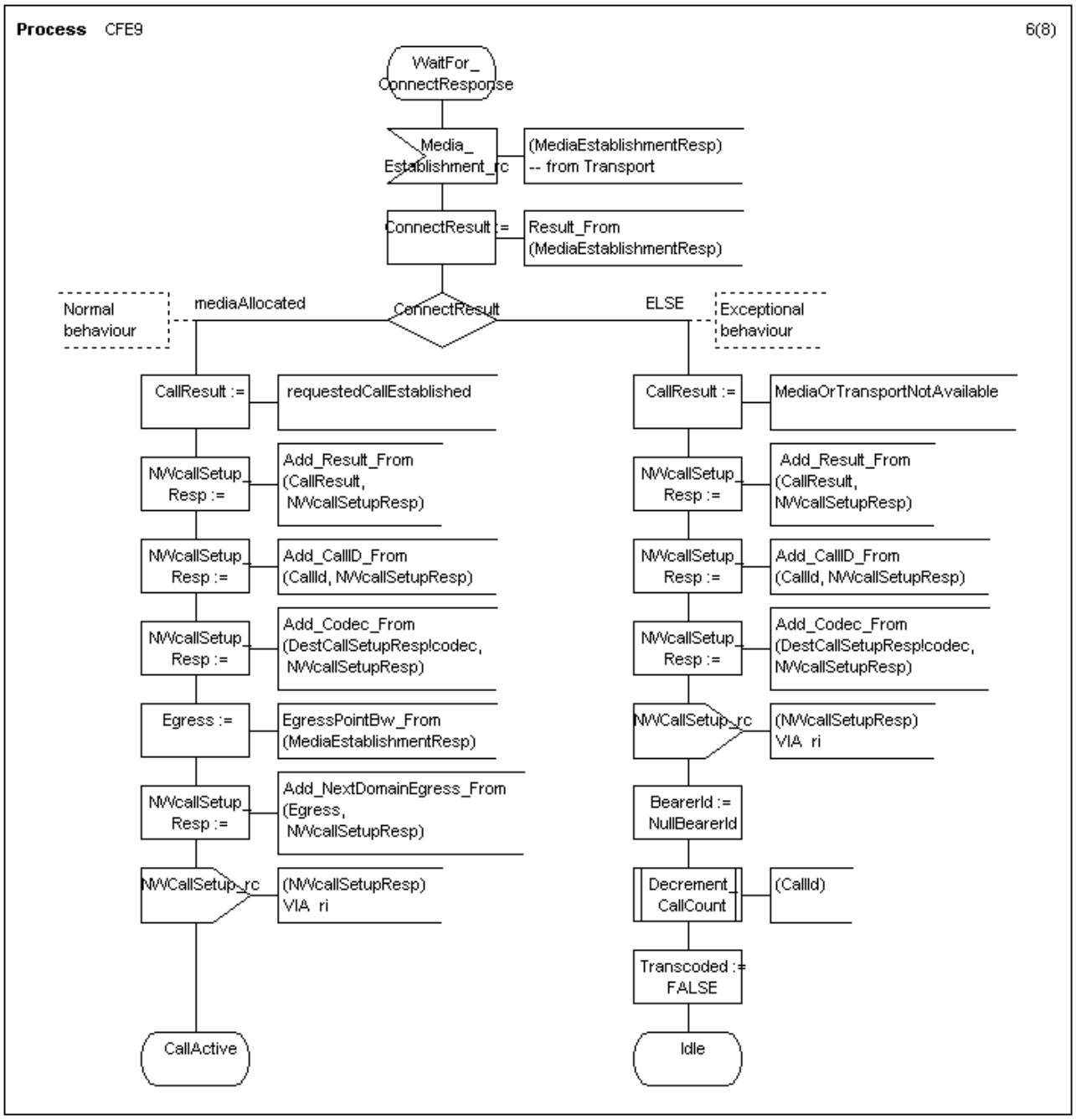


Figure 19 (sheet 6 of 8): SDL process diagram for functional entity $CFE9_{TNC}$

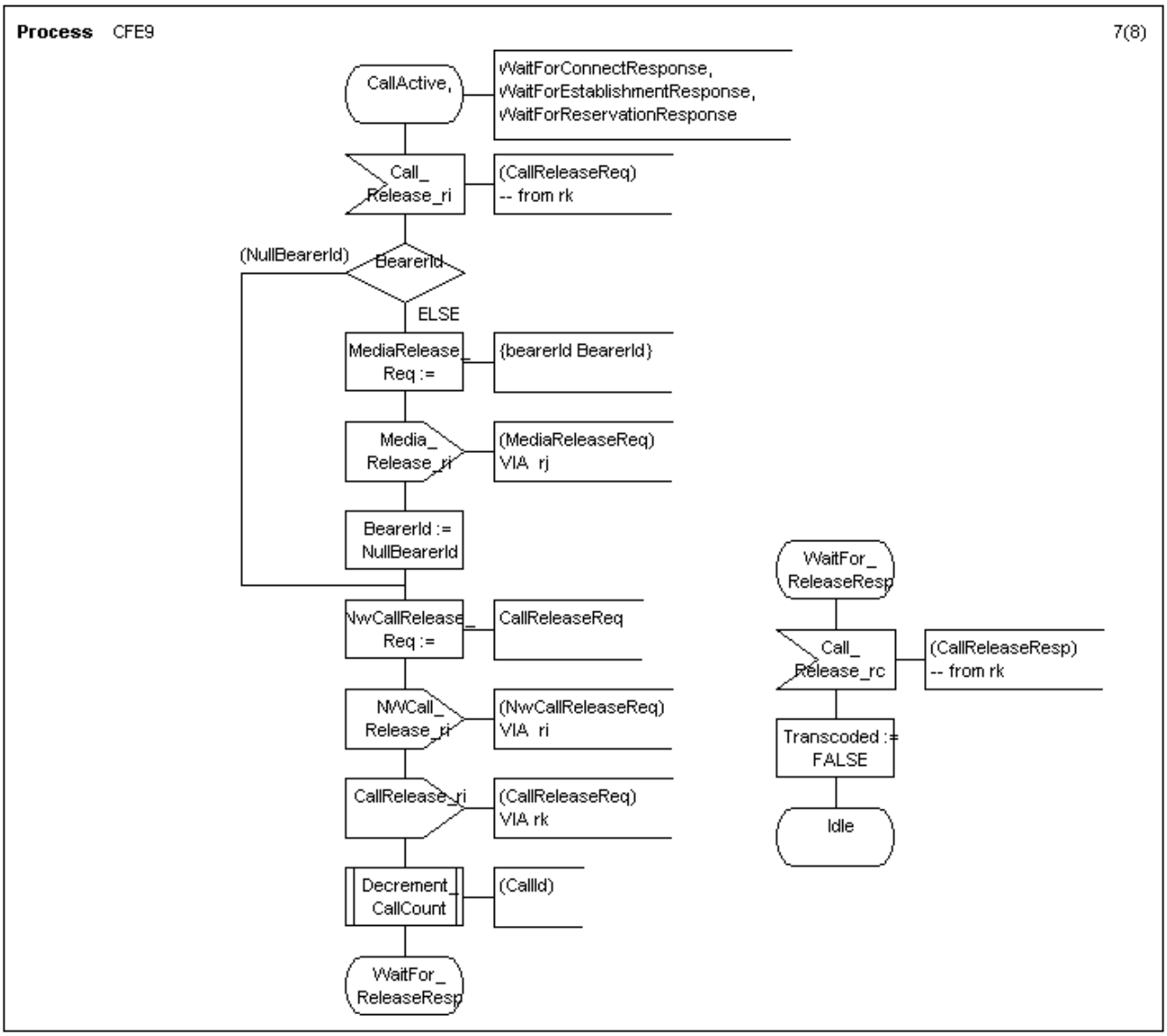


Figure 19 (sheet 7 of 8): SDL process diagram for functional entity $CFE9_{TNC}$

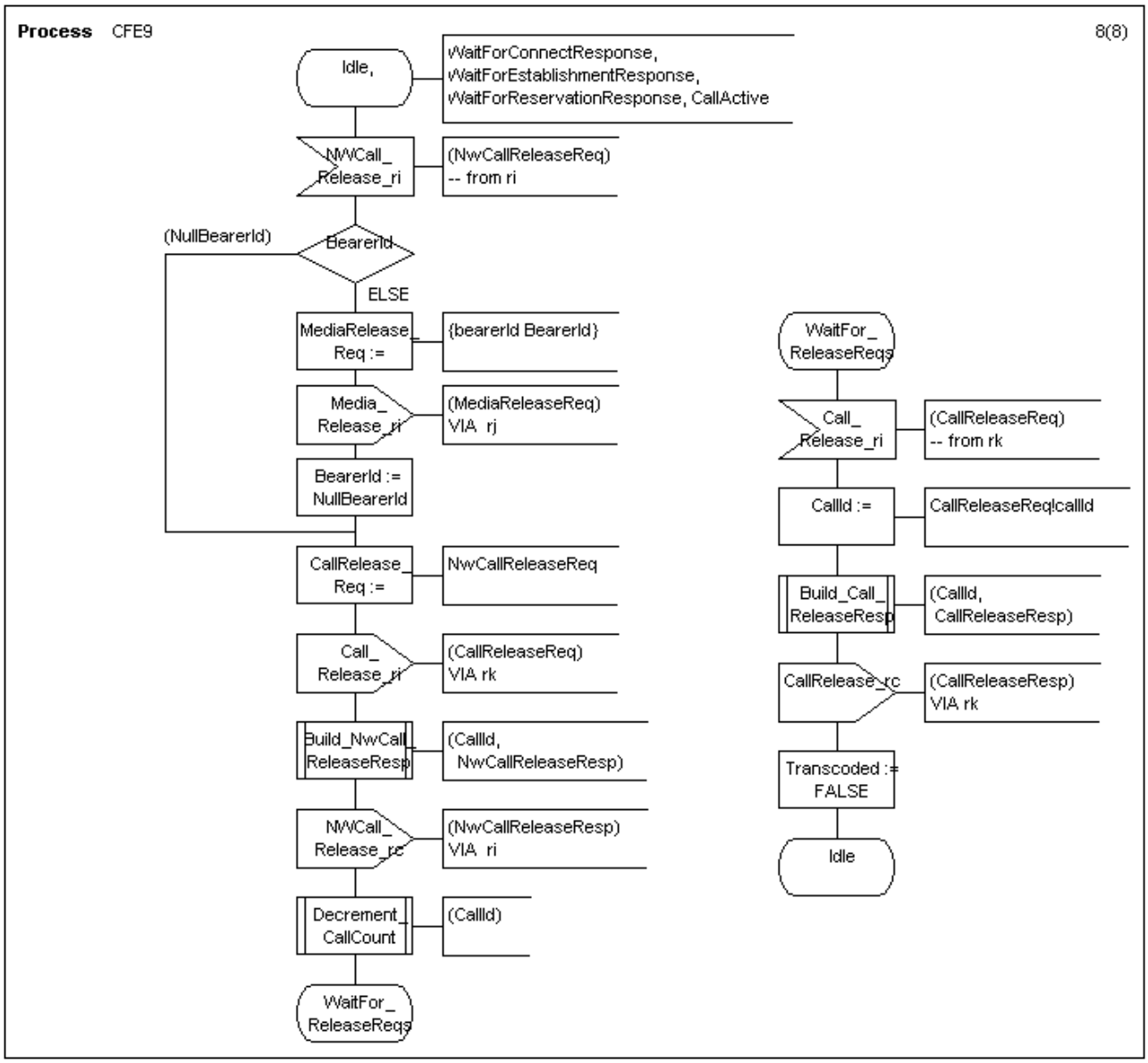


Figure 19 (sheet 8 of 8): SDL process diagram for functional entity $CFE9_{TNC}$

5.4.8 Behaviour of $CFE11_{TTC}$

The behaviour of $CFE11_{TTC}$ is shown in the SDL process diagram in figure 20.

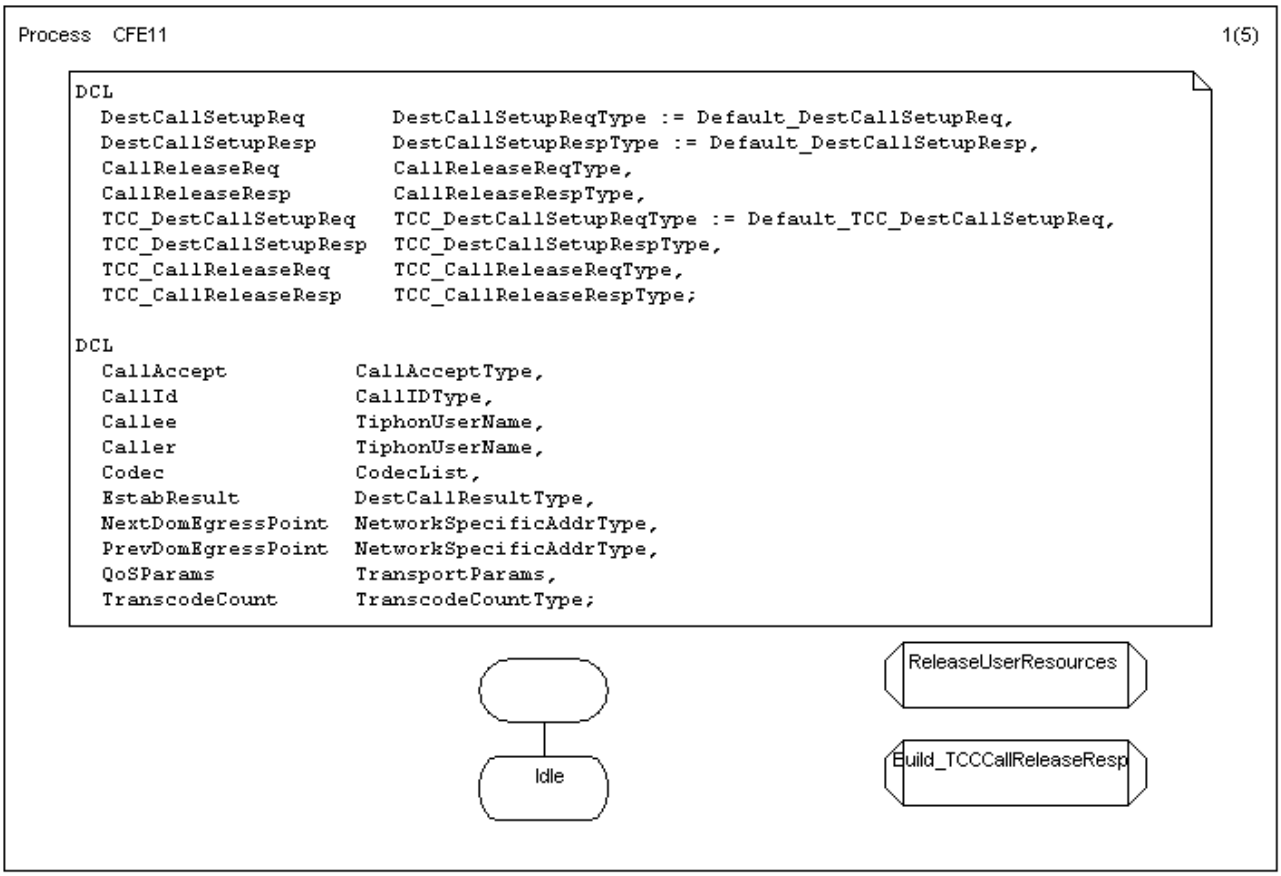


Figure 20 (sheet 1 of 5): SDL process diagram for functional entity $CFE11_{TTC}$

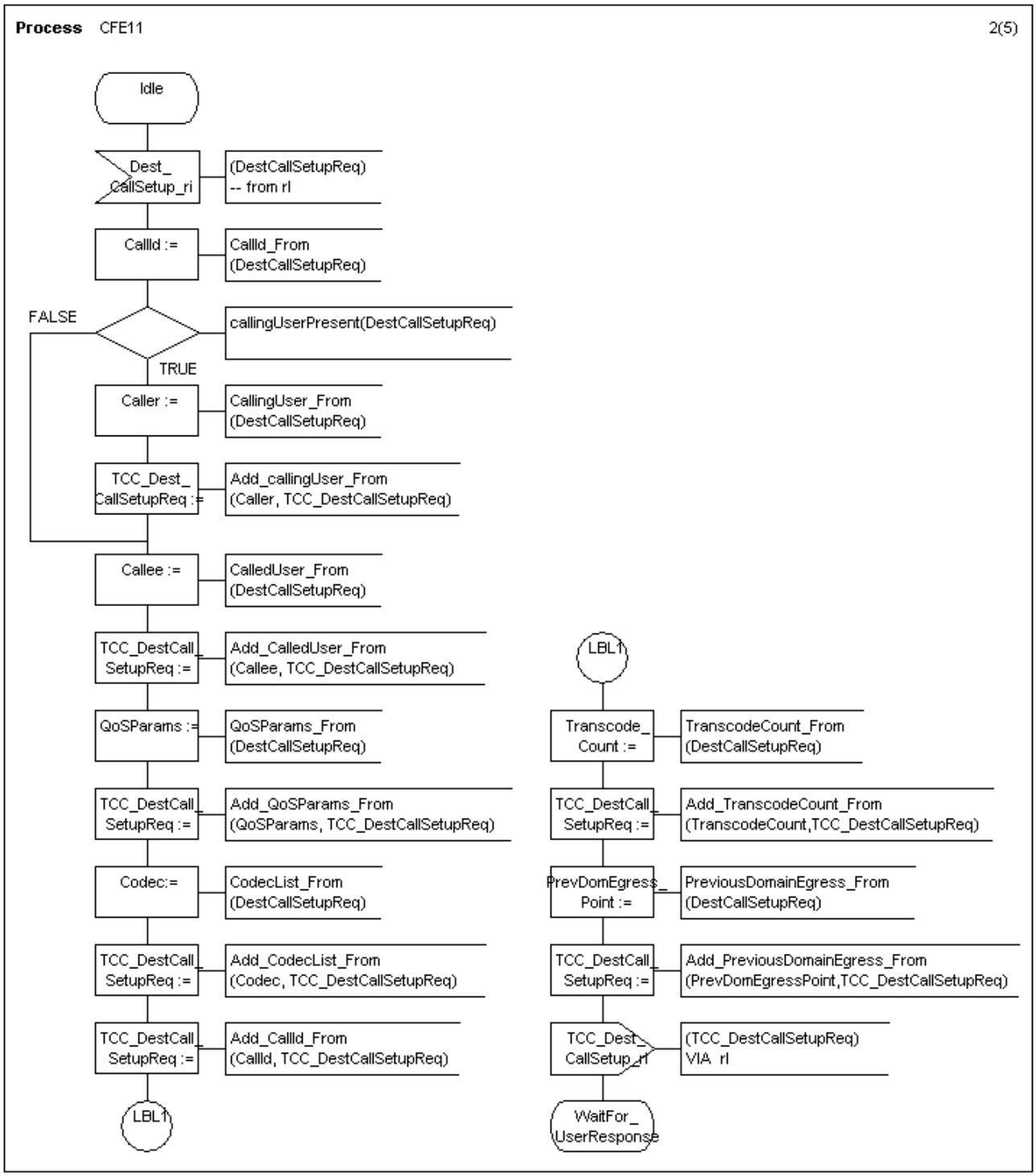


Figure 20 (sheet 2 of 5): SDL process diagram for functional entity $CFE11_{TTC}$

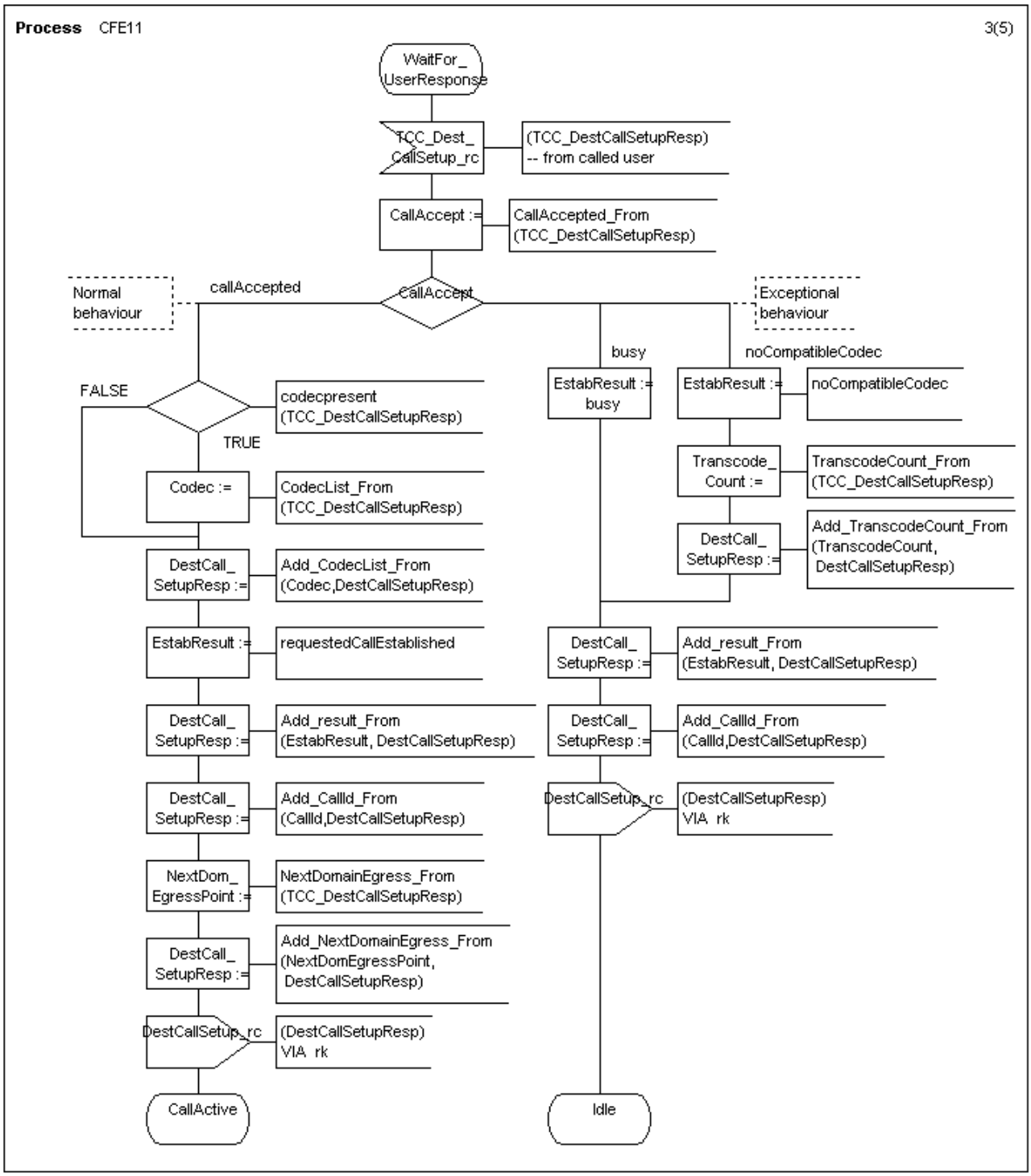


Figure 20 (sheet 3 of 5): SDL process diagram for functional entity $CFE11_{TTC}$

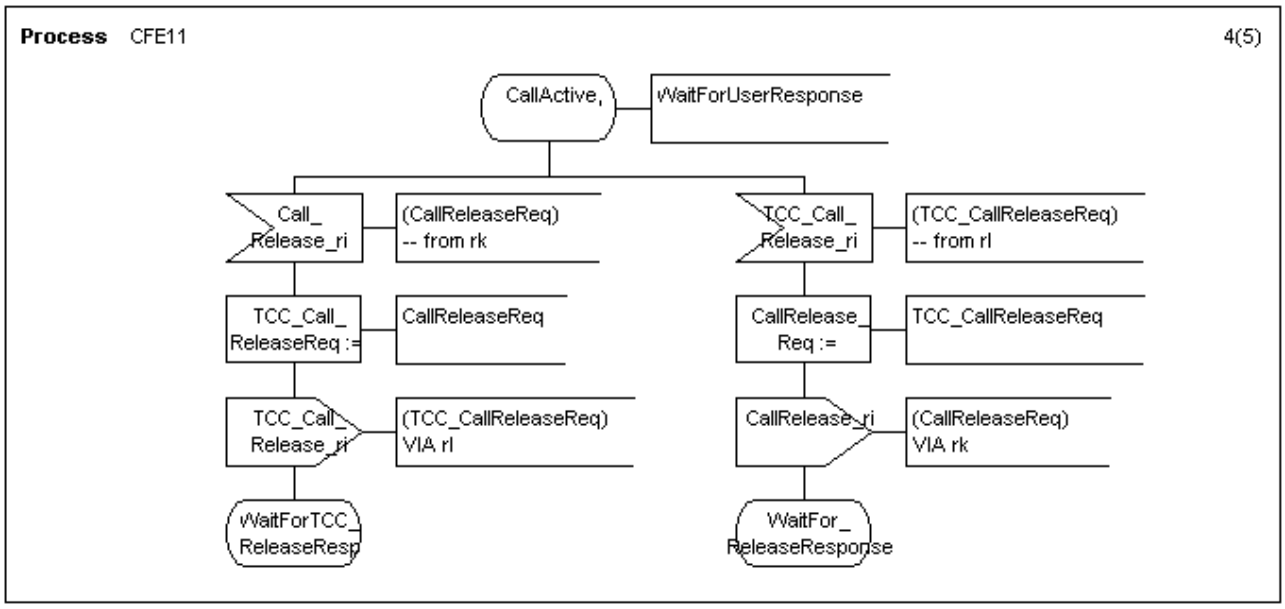


Figure 20 (sheet 4 of 5): SDL process diagram for functional entity CFE11$_{TTC}$

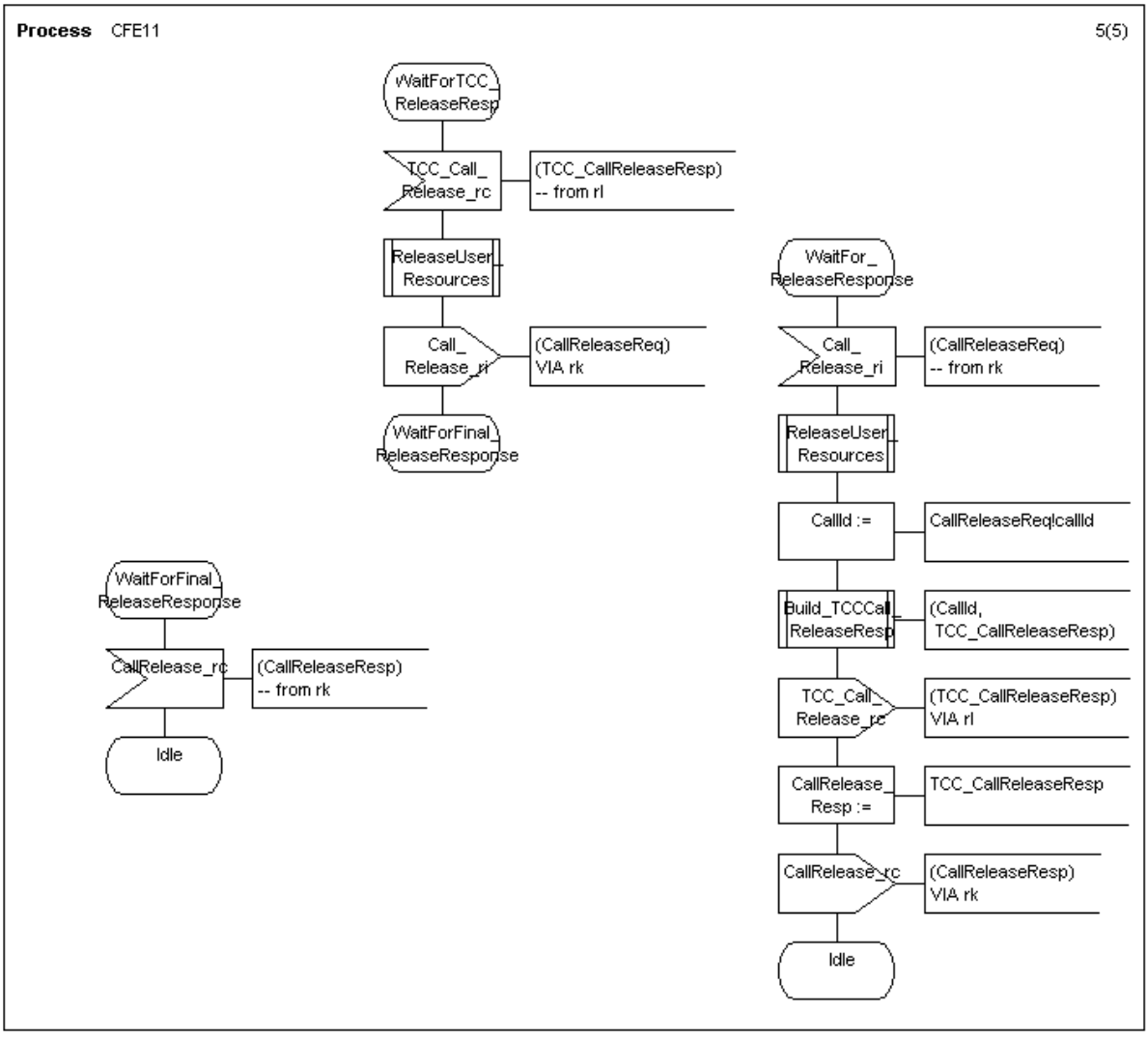


Figure 20 (sheet 5 of 5): SDL process diagram for functional entity CFE11$_{TTC}$

The ASN.1 data type and default values definitions can be found in Annex B.

5.5 Allocation of functional entities to domains

The possible allocation of CFEs to TIPHON domains is shown in table 19.

Table 19: Allocation of CFEs to TIPHON domains

Scenario	CFE1	CFE2	CFE3	CFE4	CFE5	CFE6	CFE7	CFE8	CFE9	CFE10	CFE11
Figure 3	UD	SD	SD	SD	SD/TD	SD	SD	SD/TD	SD	SD/TD	UD
Figure 5	UD	SD	SD	SD	SD/TD	SD	SD	SD/TD	SD	SD/TD	UD
Figure 6	UD	SD	SD	SD	SD/TD	SD	SD	SD/TD	SD	SD/TD	UD
Figure 7	UD	SD	SD	SD	SD/TD	SD	SD	SD/TD	SD	SD/TD	UD
Figure 8	UD	SD	SD	SD	SD/TD	SD	SD	SD/TD	SD	SD/TD	UD
Figure 9	UD	SD	SD	SD	SD/TD	SD	SD	SD/TD	SD	SD/TD	UD
Figure 10	UD	SD	SD	SD	SD/TD	SD	SD	SD/TD	SD	SD/TD	UD

NOTE: The following abbreviations are used in this table:
UD: end-User Domain
SD: Service Domain
TD: Transport Domain

The quality policy FE, CFE2, is allocated to the service domain in the originating serving network functional group, and CFE1 resolves the location of this entity based on the calling user identity and the optional operator pre-selection information elements.

Annex A (normative): Complete SDL model

The complete SDL model used for simulation and validation is provided in separate files. The SDL model is included in file "TS101882-3SDL.cbf" and the ASN.1 definition is included in file "TIPHON_SimpleCallTypes.pr". The SDL model is also included in PDF format in file "TS101882-3SDL.pdf". All These files are contained in archive ts_10188203v040101p0.zip which accompanies the present document.

Annex B (normative): ASN.1 definitions and default values

For the purposes of modelling the simple call service in SDL, the information flows and default values have been specified using the Abstract Syntax Notation 1 (ASN.1) defined in ITU-T Recommendation X.680 [6].

B.1 ASN.1 information flows definition

```
TIPHON_SimpleCallTypes DEFINITIONS ::=
BEGIN

-- ------------------------------------------------
-- TIPHON Call Control information flow data types --
-- ------------------------------------------------

TCC_OrigCallSetupReqType ::= SEQUENCE
{ calledUserID              TiphonUserName,
  callingUserIDRestriction  IdentityRestrictionType ,
  callingUser               TiphonUserName OPTIONAL,
  operatorSelect            OperatorSelection OPTIONAL,
  qoSServiceClass           QoSClass,
  trafficDescriptor         TrafficDesc,
  serviceOfferTicket        TicketType,
  codec                     CodecList,
  previousDomainEgress      NetworkSpecificAddrType,
  transcodeCount            TranscodeCountType
}

TCC_OrigCallSetupRespType ::= SEQUENCE
{ callID             CallIDType,
  codec              CodecList OPTIONAL,
  transcodeCount     TranscodeCountType OPTIONAL,
  nextDomainEgress   NetworkSpecificAddrType OPTIONAL,
  result             OrigCallResultType
}

TCC_CallReleaseReqType ::= CallReleaseReqCommonType

TCC_CallReleaseRespType ::= CallReleaseRespCommonType

TCC_CallAlertingReqType ::= CallIDType

CallSetupReqType ::= SEQUENCE
{ callID                    CallIDType,
  calledUserID              TiphonUserName,
  callingUserIDRestriction  IdentityRestrictionType,
  callingUser               TiphonUserName OPTIONAL,
  transportQoSParams        TransportParams,
  transportParmQualifier    TransportParmQualifierType,
  trafficDescriptor         TrafficDesc,
  codec                     CodecList,
  transcodeCount            TranscodeCountType,
  callingUserAccessPoint    NetworkSpecificAddrType OPTIONAL,
  routingNumber             DomainAddr  OPTIONAL,
  destServiceDomain         DomainAddr OPTIONAL,
  previousDomainEgress      NetworkSpecificAddrType
}

CallSetupRespType ::= SEQUENCE
{ callID             CallIDType,
  codec              CodecList  OPTIONAL,
  transcodeCount     TranscodeCountType OPTIONAL,
  nextDomainEgress   NetworkSpecificAddrType OPTIONAL,
  result             NWCallResultType
}

CallReleaseReqType ::= CallReleaseReqCommonType

CallReleaseRespType ::= CallReleaseRespCommonType
```

```
CallAlertingReqType ::= CallIDType

DestCallSetupReqType ::= SEQUENCE
{ callID                    CallIDType,
  calledUserID              TiphonUserName,
  callingUser               TiphonUserName OPTIONAL,
  transportQoSParams        TransportParams,
  codec                     CodecList,
  previousDomainEgress      NetworkSpecificAddrType,
  transcodeCount            TranscodeCountType
}

DestCallSetupRespType ::= SEQUENCE
{ callID             CallIDType,
  codec              CodecList OPTIONAL,
  transcodeCount     TranscodeCountType OPTIONAL,
  nextDomainEgress   NetworkSpecificAddrType OPTIONAL,
  result             DestCallResultType
}

NW_CallSetupReqType ::= SEQUENCE
{ callID                    CallIDType,
  calledUserID              TiphonUserName,
  callingUserIDRestriction  IdentityRestrictionType ,
  callingUser               TiphonUserName OPTIONAL,
  previousDomainEgress      NetworkSpecificAddrType,
  bearerId                  BearerIdType,
  transportQoSParams        TransportParams,
  transportParmQualifier    TransportParmQualifierType,
  trafficDescriptor         TrafficDesc,
  codec                     CodecList,
  transcodeCount            TranscodeCountType,
  callingUserAccessPoint    NetworkSpecificAddrType OPTIONAL,
  routingNumber             DomainAddr  OPTIONAL,
  destServiceDomain         DomainAddr   OPTIONAL
}

NW_CallSetupRespType ::= SEQUENCE
{ callID             CallIDType,
  codec              CodecList  OPTIONAL,
  transcodeCount     TranscodeCountType OPTIONAL,
  nextDomainEgress   NetworkSpecificAddrType OPTIONAL,
  result             NWCallResultType
}

NW_CallReleaseReqType ::= CallReleaseReqCommonType

NW_CallReleaseRespType ::= CallReleaseRespCommonType

NW_CallAlertingReqType ::= CallIDType


TCC_DestCallSetupReqType ::= SEQUENCE
{ callID                CallIDType,
  callingUserID         TiphonUserName OPTIONAL,
  calledUserID          TiphonUserName,
  transportQoSParams    TransportParams,
  codec                 CodecList,
  previousDomainEgress  NetworkSpecificAddrType,
  transcodeCount        TranscodeCountType
}

TCC_DestCallSetupRespType ::= SEQUENCE
{ callID             CallIDType,
  codec              CodecList OPTIONAL,
  transcodeCount     TranscodeCountType OPTIONAL,
  nextDomainEgress   NetworkSpecificAddrType OPTIONAL,
  callAccept         CallAcceptType
}


-- Data structures for the Route information signals --

CallRouteReqType ::= SEQUENCE
{ callID                    CallIDType,
  calledUserID              TiphonUserName,
  callingUserIDRestriction  IdentityRestrictionType ,
  callingUser               TiphonUserName OPTIONAL,
```

```
  routingNumber             DomainAddr  OPTIONAL,
  transportQoSParams        TransportParams,
  transportParmQualifier    TransportParmQualifierType,
  trafficDescriptor         TrafficDesc,
  codec                     CodecList,
  callingUserAccessPoint    NetworkSpecificAddrType OPTIONAL,
  destServiceDomain         DomainAddr OPTIONAL
}

CallRouteRespType ::= SEQUENCE
{ callID           CallIDType,
  routingNumber    NetworkSpecificAddrType OPTIONAL,
  nextHopAddrList  DomainAddrListType OPTIONAL,
  result           RouteResultType
}


-- Data structures for the Policy information signals --

ServingNWPolicyReqType ::= SEQUENCE
{ callingUserID  TiphonUserName,
  calledUserID  TiphonUserName,
  requestedQoS  RequestedQoSType,
  serviceOfferTicket  TicketType
}

ServingNWPolicyRespType ::= policyResultType

-- Data structures for the media and transport control signals --

MediaReservationReq_Type  ::= SEQUENCE
{ bearerId             BearerIdType,
  qosParmQualifier     TransportParmQualifierType OPTIONAL,
  mediaDescriptor      MediaDescriptorType,
  qosParms             QoSParametersType OPTIONAL,
  previousDomEgressFw  NetworkSpecificAddrType,
  nextDomainAddress    NetworkDomainAddrType OPTIONAL,
  userDomainAddress    NetworkSpecificAddrType OPTIONAL
}

MediaReservationResp_Type  ::= SEQUENCE
{ bearerId          BearerIdType,
  qosParmQualifier  TransportParmQualifierType OPTIONAL,
  mediaId           MediaIdType OPTIONAL,
  qosParameters     QoSParametersType OPTIONAL,
  egressPointFw     NetworkSpecificAddrType OPTIONAL,
  mediaResResult    MediaResResultType
}

MediaEstablishmentReq_Type  ::= SEQUENCE
{ bearerId             BearerIdType,
  mediaId              MediaIdType,
  nextDomainEgressRev  NetworkSpecificAddrType
}

MediaEstablishmentResp_Type ::= SEQUENCE
{ bearerId          BearerIdType,
  mediaId           MediaIdType,
  egressPointRev    NetworkSpecificAddrType OPTIONAL,
  mediaEstabResult  MediaEtabResultType
}

MediaReleaseReq_Type ::= SEQUENCE
{ bearerId   BearerIdType,
  mediaId    MediaIdType OPTIONAL
}

-- ------------------------------------------------------------
-- TIPHON Call Control service information element data types  --
-- ------------------------------------------------------------

BearerIdType  ::= Visiblestring (SIZE (0..128))

BearerIntegrityType ::= ENUMERATED
{ timeSlotSequenceIntegrety,
  serviceDataUnitIntegrety,
  unstructured,
  dataSequenceIntegrety,
```

```
  integrety8kHz
}

CallAcceptType ::= ENUMERATED
{ callAccepted,
  noCompatibleCodec,
  busy
}

CallIDType ::= Natural

CallReleaseReqCommonType ::= SEQUENCE
{
  callId  CallIDType OPTIONAL,
  causeCode CauseCodeType
}

CallReleaseRespCommonType ::=
SEQUENCE
{ callId CallIDType,
  result ReleaseResultType
}

CauseCodeType ::= ENUMERATED
{ userInitiated,
  networkInitiated
}

CodecDescrType ::= SEQUENCE
{ codecId                     CodecID,
  codecParms                  CodecParametersType,
  silenceSuppressionEnabled   Boolean,
  echoCancelling              Boolean,
  mediaPeakRate               FrameRateType,
  maxMediaFrameSize           FrameCountType
}

CodecID      ::= Visiblestring (SIZE (1..15))

CodecList    ::= SEQUENCE (SIZE (1..8)) OF CodecType

CodecParametersType ::= SEQUENCE
{ framesPerPacket             FrameCountType,
  maxCodecFrameSize           FrameSizeType,
  codecSpecificParameters     Visiblestring
}

CodecType    ::= SEQUENCE
{ codecID            CodecID,
  framesperPacket    FrameCountType
}

DestCallResultType ::= OrigCallResultType (<= busy)

DigestType ::= Visiblestring

DomainAddr      ::= CHOICE
{ ipv4DomainAddr  [0] FourOctetsType,
  ipv6DomainAddr  [1] SixteenOctetsType
}

DomainAddrListType ::= SEQUENCE OF NetworkDomainAddrType

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

E212Number  ::= NumericString (SIZE (15))

FourOctetsType  ::= Octet String (SIZE (4))

FrameCountType   ::= Integer (0..maxFrameCount)

FrameLength    ::= Integer (1..65535)

FrameRateType  ::= Integer (1..255)

FrameSizeType ::= Integer(0..255)

IdentityRestrictionType ::= ENUMERATED
```

```
{ identityAvailable,
  identityUnavailable
}

IPAddressType ::= CHOICE
{ ipv4Address  IPv4AddressType,
  ipv6Address  IPv6AddressType
}

IPv4AddressType  ::= SEQUENCE
{  addr  FourOctetsType,
   port  TwoOctets
}

IPv6AddressType     ::= SEQUENCE
{  addr  SixteenOctetsType,
   port  TwoOctets
}

maxFrameCount Integer ::= 255     -- For non-framing codecs maxFrameCount contains
                                  -- the maximum number of samples per packet.

maxQoSClass Integer ::= 255

MediaEtabResultType ::= ENUMERATED
{ mediaAllocated,
  unableToAllocateResource,
  resourceNoLongerAvailable
}

MediaDescriptorType ::= SEQUENCE
{ mediaIdHandle        MediaIdType OPTIONAL,
  codecDescr           CodecDescrType,
  codecDescrOptional   CodecDescrType OPTIONAL, -- present if transcoding in use
  connectionPriority   PriorityType
}

MediaIdType ::= Integer

MediaResResultType ::= ENUMERATED
{ mediaReserved,
  mediaResourceNotAvailable,
  mediaResourceNotSupported,
  destinationUnknown
}

MicroSeconds  ::= Integer(0..10000000)  -- Allows up to 10s to be expressed in micro-sec --

NetworkDomainAddrType ::= CHOICE
{ ipv4Domain  FourOctetsType,
  ipv6Domain  SixteenOctetsType
}

NetworkSpecificAddrType ::= CHOICE
{
  slotNumber SlotNumberType,
  ipAddress  IPAddressType
}

NonStandardQoSClass ::= QoSClass(16..maxQoSClass)

NWCallResultType ::= OrigCallResultType (<= unknownUser)

OneOctet        ::= Octet String (SIZE (1))

OperatorSelection ::= CHOICE
{ prefixdial   SEQUENCE OF TelephoneDigitType,
  operatorID   Visiblestring
}

OrigCallResultType   ::= ENUMERATED
{ requestedCallEstablished (0),
  noCompatibleCodec,
  busy,
  MediaOrTransportNotAvailable,
  qoSNotAvailable,
  unknownUser,
  policyRejection
```

```
}

PercentX1000  ::= Integer(0..100000)  -- Allows up to 100% to be expressed in   --
                                      -- increments of 0.001%                   --

PolicyResultType  ::= ENUMERATED
{ callAllowed,
  invalidTicket,
  serviceNotSubscribedTo,
  serviceCurrentlyNotAvailable
}

predefinedQoS           QoSClass ::= 0

PriorityType ::= ENUMERATED
{ normal,
  emergency
}

QoSClass  ::= Integer(0..maxQoSClass)

QoSParametersType ::= SEQUENCE
{ packetTxRate       TrafficDesc,
  packetLossRate     PercentX1000,
  maxDelayVariation  MicroSeconds,
  bearerIntegrity    BearerIntegrityType,
  transitDelay       MicroSeconds
}

ReleaseResultType ::= ENUMERATED
{ successful,
  failed
}

RequestedQoSType  ::= CHOICE
{ transportQoSParams TransportParams,
  qoSServiceClass    QoSClass
}

RouteResultType ::= ENUMERATED
{ noRouteFound,
  RouteFound
}

ServiceCredentialsType ::= SEQUENCE OF ServiceCredentialType

ServiceCredentialType ::= SEQUENCE
{ serviceAppId  ServiceApplicationType,
  spoA          SpoAType,
  startTime     GeneralizedTime,
  stopTime      GeneralizedTime, -- Shall be greater than StartTime
  cryptoDigest  DigestType OPTIONAL
}

ServiceApplicationType ::= Visiblestring

SixteenOctetsType   ::= Octet String (SIZE (16))

SpoAType ::= Visiblestring

SlotNumberType ::= Integer

TelephoneDigitType ::= NumericString  (FROM ("0":"9"))

TicketType ::= SEQUENCE
{ registrantId       Visiblestring,
  registrarId        Visiblestring,
  serviceCredential  ServiceCredentialsType,
  cryptoDigest       DigestType OPTIONAL
}

tiphonQoSClass-1        QoSClass ::= 1
tiphonQoSClass-2A       QoSClass ::= 2
tiphonQoSClass-2M       QoSClass ::= 3
tiphonQoSClass-2H       QoSClass ::= 4
tiphonQoSClass-3        QoSClass ::= 5

TiphonUserName ::= CHOICE
```

```
{ e164         E164Number,
  url          Visiblestring,
  displayName  Visiblestring
}

TrafficDesc  ::= SEQUENCE
{  peakFrameRate   FrameRateType,
   maxFrameLength  FrameLength
}

TranscodeCountType ::= Integer (0..255)

TransportParams ::= SEQUENCE
{ maximumDelay  MicroSeconds,
  maxDelayVariation MicroSeconds,
  maxMeanPacketLoss PercentX1000
    -- Packet loss is specified as % x 1000 to avoid --
    -- the need for REAL numbers when loss is less   --
    -- than one percent                              --
}

TransportParmQualifierType ::= ENUMERATED
{ totalRemainingBudget,
  budgetAvailableForDomain
}

TwoOctets       ::= Octet String (SIZE (2))

END
```

B.2 ASN.1 Default values

The Simple call service model has been initialized with the following values for validation.

```
DEFINITONS DefaultDefinitions ::=
BEGIN
/**  Default value definitions for simple call data types **/

Default_TiphonUserName TiphonUserName ::= e164 : Default_E164Number

Default_E164Number E164Number ::= "0"

Default_TrafficDesc TrafficDesc ::=
{ peakFrameRate     Default_FrameRate,
  maxFrameLength    Default_FrameLength  }

Default_CodecList CodecList  ::= { Default_Codec }

Default_Codec CodecType ::=
{ codecID            Default_CodecID,
  framesperPacket    Default_FrameCount  }

Default_CodecID CodecID ::= "CodecID"

Default_FrameCount FrameCountType ::= 0

Default_RequestedQoS RequestedQoSType ::= transportQoSParams : Default_TransportParams

Default_BearerId BearerIdType ::= "Handle"

Default_FrameRate  FrameRateType ::= 1

Default_FrameLength FrameLength ::= 1

Default_TransportParams TransportParams ::=
{  maximumDelay      Default_MicroSeconds,
   maxDelayVariation Default_MicroSeconds,
   maxMeanPacketLoss Default_PercentX1000 }

Default_MicroSeconds  MicroSeconds ::= 0

Default_PercentX1000  PercentX1000  ::= 0

Default_FourOctets   FourOctetsType   ::= '11112222'H
```

```
Default_DomainAddr  DomainAddr ::= ipv4DomainAddr : Default_FourOctets

Default_DomainAddrList  DomainAddrListType ::= {}

Default_IdentityRestriction  IdentityRestrictionType ::= identityUnavailable

Default_CallID  CallIDType ::= 0

Default_Ticket  TicketType ::=
{ registrantId       "",
  registrarId        "",
  serviceCredential  Default_ServiceCredentials }

Default_ServiceCredentials  ServiceCredentialsType ::= {}

Default_TransportParmQualifier  TransportParmQualifierType ::= totalRemainingBudget

Default_TranscodeCount TranscodeCountType ::= 0

Default_CallResult  DestCallResultType ::= requestedCallEstablished

Default_TCC_CallSetupResp  TCC_OrigCallSetupRespType ::=
{ callID  Default_CallID,
  result  Default_CallResult }

Default_CallSetupReq  CallSetupReqType ::=
{ callID                    Default_CallID,
  calledUserID              Default_TiphonUserName,
  callingUserIDRestriction  Default_IdentityRestriction,
  transportQoSParams        Default_TransportParams,
  transportParmQualifier    Default_TransportParmQualifier,
  trafficDescriptor         Default_TrafficDesc,
  codec                     Default_CodecList,
  transcodeCount            Default_TranscodeCount,
  previousDomainEgress      Default_NetworkSpecificAddr }

Default_CallSetupResp  CallSetupRespType  ::=
{ callID Default_CallID,
  result  Default_CallResult }

Default_DestCallSetupReq  DestCallSetupReqType ::=
{
  callID                    Default_CallID,
  calledUserID              Default_TiphonUserName,
  transportQoSParams        Default_TransportParams,
  codec                     Default_CodecList,
  previousDomainEgress      Default_NetworkSpecificAddr,
  transcodeCount            Default_TranscodeCount }

Default_DestCallSetupResp  DestCallSetupRespType ::=
{ callID  Default_CallID,
  codec   Default_CodecList,
  result  Default_CallResult }

Default_NW_CallSetupReq  NW_CallSetupReqType ::=
{ callID                    Default_CallID,
  calledUserID              Default_TiphonUserName,
  callingUserIDRestriction  Default_IdentityRestriction,
  previousDomainEgress      Default_NetworkSpecificAddr,
  bearerId                  Default_BearerId,
  transportQoSParams        Default_TransportParams,
  transportParmQualifier    Default_TransportParmQualifier,
  trafficDescriptor         Default_TrafficDesc,
  codec                     Default_CodecList,
  transcodeCount            Default_TranscodeCount }

Default_NW_CallSetupResp  NW_CallSetupRespType ::=
{ callID  Default_CallID,
  result  Default_CallResult }

Default_CallRouteReq  CallRouteReqType ::=
{ callID                    Default_CallID,
  calledUserID              Default_TiphonUserName,
```

```
  callingUserIDRestriction  Default_IdentityRestriction,
  transportQoSParams        Default_TransportParams,
  transportParmQualifier    Default_TransportParmQualifier,
  trafficDescriptor         Default_TrafficDesc,
  codec                     Default_CodecList }

Default_RouteResult  RouteResultType ::= noRouteFound

Default_CallRouteResp  CallRouteRespType ::=
{ callID           Default_CallID,
  nextHopAddrList  Default_DomainAddrList,
  result           Default_RouteResult }

Default_MediaReserveReq  MediaReservationReq_Type::=
{  bearerId             Default_BearerId,
   mediaDescriptor      Default_MediaDescriptor,
   previousDomEgressFw  Default_NetworkSpecificAddr }

Default_MediaResResult MediaResResultType ::= mediaReserved

Default_MediaReservationResp MediaReservationResp_Type::=
{ bearerId          Default_BearerId,
  mediaResResult    Default_MediaResResult  }

Default_MediaEstablishmentReq MediaEstablishmentReq_Type  ::=
{ bearerId             Default_BearerId,
  mediaId              Default_MediaId,
  nextDomainEgressRev  Default_NetworkSpecificAddr }

Default_MediaEstablishmentResp MediaEstablishmentResp_Type ::=
{ bearerId          Default_BearerId,
  mediaId           Default_MediaId,
  mediaEstabResult  Default_MediaEtabResult }

Default_PolicyReq  ServingNWPolicyReqType ::=
{ callingUserID      Default_TiphonUserName,
  calledUserID       Default_TiphonUserName,
  requestedQoS       Default_RequestedQoS,
  serviceOfferTicket Default_Ticket }

Default_TCC_DestCallSetupReq  TCC_DestCallSetupReqType ::=
{ callID                Default_CallID,
  calledUserID          Default_TiphonUserName,
  transportQoSParams    Default_TransportParams,
  codec                 Default_CodecList,
  previousDomainEgress  Default_NetworkSpecificAddr,
  transcodeCount        Default_TranscodeCount }

Default_MediaDescriptor MediaDescriptorType ::=
{  codecDescr           Default_CodecDescr,
   connectionPriority   Default_Priority }

Default_CodecDescr CodecDescrType ::=
{ codecId                    Default_CodecId,
  codecParms                 Default_CodecParameters,
  silenceSuppressionEnabled  FALSE,
  echoCancelling             FALSE,
  mediaPeakRate              Default_FrameRate,
  maxMediaFrameSize          Default_FrameCount  }

Default_Priority PriorityType ::= normal

Default_CodecParameters  CodecParametersType ::=
{ framesPerPacket           Default_FrameCount,
  maxCodecFrameSize         Default_FrameSize,
  codecSpecificParameters   '' }

Default_FrameSize FrameSizeType ::= 1

Default_NetworkSpecificAddr NetworkSpecificAddrType ::=
  ipaddress:(ipv4address:({addr '00000001'H,port '12'H}))

Default_MediaId MediaIdType ::= 0
```

```
Default_MediaEtabResult MediaEtabResultType ::= mediaAllocated

END
```

History

Document history		
V1.1.1	May 2002	Publication as TS 101 882
V4.1.1	November 2003	Publication