EUROPEAN TELECOMMUNICATION STANDARD

ETS 300 325

March 1994

Source: ETSI TC-TE

Reference: DE/TE-02029

ICS: 33.080

Key words: ISDN, PCI

Integrated Services Digital Network (ISDN); Programming Communication Interface (PCI) for Euro-ISDN

ETSI

European Telecommunications Standards Institute

ETSI Secretariat

Postal address: F-06921 Sophia Antipolis CEDEX - FRANCE
Office address: 650 Route des Lucioles - Sophia Antipolis - Valbonne - FRANCE
X.400: c=fr, a=atlas, p=etsi, s=secretariat - **Internet:** secretariat@etsi.fr

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

Contents

Foreword

This European Telecommunication Standard (ETS) has been produced by the Terminal Equipment (TE) Technical Committee of the European Telecommunications Standards Institute (ETSI).

Annexes B, C, D, F, G and H to this ETS are normative while annexes A, E, J and K are informative.

Transposition dates	
Date of latest announcement of this ETS (doa):	30th June 1994
Date of latest publication of new National Standard or endorsement of this ETS (dop/e):	31st December 1994
Date of withdrawal of any conflicting National Standard (dow):	31st December 1994

Introduction

The number of different Integrated Services Digital Network (ISDN) Programming Interfaces used by terminal equipment has hindered the development of applications using ISDN which, in turn, has proved a constraint to the usage of ISDN on modern terminal equipment.

This ETS defines the ETSI ISDN Application Programming Interface (API), called ISDN Programming Communication Interface (PCI). The ISDN PCI is an application interface for accessing and administering ISDN services.

It has been defined in order to provide a standard that terminal equipment providers should implement instead of providing their own programming interface. Thus allowing the portability of applications that use the ISDN PCI across a range of terminal equipment based on different operating systems.

The ISDN PCI has been defined with the Application Developer in mind and, where possible, eliminates the need for a detailed knowledge of ISDN. It has also been defined in such a manner that extensions provided to take advantage of future ISDN developments do not effect the operation of existing applications.

Blank page

1 Scope

This ETS specifies the Integrated Services Digital Network Programming Communication Interface (ISDN PCI) for the accessing and administering of the following ISDN services:

- Bearer Services (as defined in ETS 300 102-1 [2]);
- Supplementary Services (as defined in ETS 300 196);
- Virtual Circuit (VC) or Permanent Virtual Circuit (PVC) Bearer Services on the B- and D-channels.

The PCI defined in this ETS:

- covers both Basic and Primary rate ISDN access;
- is independent of operating system, hardware and programming languages. It provides language and operating system binding for common operating system environments;
- supports concurrent applications;
- supports concurrent protocol stacks related to data exchange;
- supports application access to multiple channels on multiple ISDN accesses;
- provides the Open Systems Interconnection (OSI) connection-mode network service as defined by CCITT Recommendation X.213 [7] using the method defined in ISO 9574;
- provides an interface for applications requiring direct control of ISDN services;

- shows the impact of security issues on the interface;
- has been defined to allow future extension of functionality.

Further standards specify the method of testing and detailed application specific requirements to determine conformance based on this ETS.

2 Normative references

This ETS incorporates by dated and undated references, provisions from other publications. These normative references are cited at the appropriate places in the text and the publications are listed hereafter. For dated references, subsequent amendments to or revisions of any of these publications apply to these ETS only when incorporated in it by amendment or revision. For undated references the latest edition of the publication referred to applies.

[1] ETS 300 080 (1992): "Integrated Services Digital Network (ISDN); ISDN lower layer protocols for telematic terminals".

[2] ETS 300 102-1 (1990): "Integrated Services Digital Network (ISDN); User-network interface layer 3, Specifications for basic call control".

[3] ISO/IEC 8208 (1990): "Information technology; Data communications; X.25 Packet Layer Protocol for Data Terminal Equipment".

[4] ISO 7776 (1986): " Information Processing systems; Data communications; High-level data link control procedures; Description of the X.25 LAPB-compatible DTE data link procedures".

[5] ISO/IEC 9646 (1991): "Information technology - Open Systems Interconnection - Conformance testing methodology and framework".

[6] ISO/IEC ISP 10609 (1992): "Information technology; International Standardized Profiles TB, TC, TD and TE; Connection-mode Transport Service over connection-mode Network Service".

[7] CCITT Recommendation X.213 (1988): "Network Service Definition for Open Systems Interconnection for CCITT Applications".

[8] ISO CEI/9899 (1990): "Programming Language-C".

[9] ETR 018: "Integrated Services Digital Network (ISDN); Application of the BC-, HLC-, LLC- information elements by terminals supporting ISDN services".

NOTE: For further references to publications, which are of interest when reading this ETS, refer to the bibliography contained in Annex A.

3 Definitions and abbreviations

3.1 Definitions

For the purposes of this ETS, the following definitions apply:

Address Set: A set of parameters containing remote and local user layer or signalling addresses.

Administration Plane: The logical grouping of functionality for management of PCI User Facility-Network Access Facility(PUF-NAF) dialogue as well as for access to local or network related Network Access Facility (NAF) resources.

Attribute Set: A set of parameters driving user protocols and ISDN signalling.

B-Channel: The logical ISDN channel for the use of data transfer.

Control Plane: The logical grouping of functionality for access of ISDN signalling.

D-Channel: The logical ISDN channel used for signalling and, in some cases, for data transfer.

Euro-ISDN: ISDN offering services and interoperability in Europe as agreed upon in the "Memorandum of Understanding on the Implementation of European ISDN Service by 1992".

Exchange Function: PUF functionality realising the Exchange Mechanism.

Exchange Mechanism: The means provided for the PUF to interchange messages with the NAF.

ISDN access: A set of ISDN channels provided by a single NAF to access ISDN services.

ISDN Programming Communication Interface (ISDN PCI): A network (ISDN) oriented software interface providing access provisions for programming network signalling and user data exchange.

Message: A unit of information transferred through the interface between the NAF and the PCI User Facility (PUF).

Network Access Facility (NAF): A functional unit located between the ISDN PCI and the network related layers.

Network Connection Object (NCO): An abstract object within the NAF created by the PUF to gain access to network signalling or data.

Network layer Message Access (NMA): A logical message access to ISDN network layer user protocols.

NULL Layer: This describes an empty layer of the OSI reference model. Such a layer does not contain any functionality and passes requests and responses transparently to adjourning layers.

PCI User Facility (PUF): The functional unit using the ISDN PCI to access a NAF. In fact, the local application using the interface.

Signalling Message Access (SMA): A logical message access to signalling part of ISDN.

Transparent Message Access (TMA): A logical message access to ISDN physical layer.

Type-Length-Value coding (TLV coding): The coding scheme used for binary presentation of Messages.

User Connection: A connection accessible through User Plane functionality.

User Plane: A logical grouping of functionality for access of user protocols and data.

User Protocol: The protocol running and conforming to User Plane functionality.

3.2 Abbreviations

For the purposes of this ETS, the following abbreviations apply:

AOC-D	Advice of Charge during call, AOC-E Advice of Charge at end of call
API	Application Programming Interface
ASP	Abstract Service Primitive (i.e. OSI service primitive exchanged at a Service Access Point (SAP))
Bcug	Bilateral closed user group
CLIP	Calling Line Identification Presentation
CLIR	Calling Line Identification Restriction
CONS	Connection Oriented Network Service
CW	Call Waiting
COLP	Connected Line Identification Presentation
COLR	Connected Line Identification Restriction
DDI	Direct Dialling In
DTMF	Dual Tone Multi Frequency
ETS	European Telecommunication Standard
ETSI	European Telecommunications Standards Institute
EXid	exchange identifier
HLC	High Layer Compatibility
ICS	Implementation Conformance Statement
ISDN	Integrated Services Digital Network
ISPBX	ISDN Private Branch eXchange
IUT	Implementation Under Test (i.e. protocol layer which is subject to test)
LAP B	Link Access Procedure Balanced
LAP D	Link Access Procedure for D-channel
LLC	Low Layer Compatibility
LT	Lower Tester (i.e. tester interfaced at SAP at lower boundary of IUT)
MOU	Memorandum Of Understanding of the European Community, 1989
MSB	Most Significant Bit
MSN	Multiple Subscriber Number
N-SAP	Network layer - Service Access Point
NAF	Network Access Facility
NCO	Network Connection Object
NCOID	NCO Identifier
NMA	Network layer Message Access
PCI	Programming Communication Interface
PCI-NAF	PCI User Facility - Network Access Facility
PciMPB	Pci Message Parameter Block
PCO	Point of Control and Observation
PDU	Protocol Data Unit
Ph-SAP	Physical layer - Service Access Point
PUF	Programming Communication Interface User Facility
SAP	Service Access Point
SMA	Signalling Message Access
SUB	Subaddressing
SUT	System Under Test (i.e. complete ISDN terminal equipment)
TCP	Test Co-ordination Procedure (between Upper Tester (UT) and LT)
TLV coding	Type-Length-Value coding (used for presentation of ISDN PCI messages)
TM PDU	Test Management Protocol Data Unit
TMA	Transparent Message Access
UT	Upper Tester (i.e. tester interfaced at SAP at upper boundary of IUT)
X.25 PLP	X.25 Packet Layer Protocol

4 Reader's guidance and overview

4.1 Reader's guide

This ETS is intended for:

- software developers and implementors of applications by providing them with the definition of a simple, standardized and portable interface giving access to the Euro-ISDN;

- manufacturers and developers of ISDN adapters and system software with the aim of providing a standardized programming interface to Euro-ISDN communications;

- users of ISDN based software and management personnel by providing them with background information and selection criteria for choosing ISDN products and applications.

4.2 How to use this ETS

Readers who:

- need a quick overview over the ISDN PCI features and capabilities should read the overview provided in subclause 4.3. More detailed information about the architecture and functional description is provided in Clause 5. Clause 3 provides useful information on definitions of terms and abbreviations used;

- intend to implement an application using this ISDN PCI interface ETS should first read Clauses 4, 5 and 7. Clause 3 provides useful information on the definitions of terms and abbreviations used. For more detailed information Clause 6 and Annex F should also be inspected. Coding examples are provided in Annex J;

- intend to build an ISDN adapter card or equipment should also first inspect Clauses 4, 5 and 7. Clause 3 provides useful information on the definitions of terms and abbreviations used. For more detailed information Clause 6 and Annex F should be inspected. Specific NAF information is contained in Annex E.

Table 1 gives a descriptive list showing the full contents of this ETS.

Table 1: List of ETS contents

Clause Annex	contains ...
Clause 1	... the scope of this ETS. This describes what this ETS covers.
Clause 2	... normative references.
Clause 3	... definitions of the terms and abbreviations used throughout this ETS.
Clause 4	... gives an overview and reader's guidance.
Clause 5	... the architecture, the functional description of the interface and an overview describing the interactions between the component elements of the interface.
Clause 6	... the conventions used to describe the messages, the set of messages used to communicate through the interface and the parameters carried by these messages.
Clause 7	... the exchange mechanism, describing how messages are exchanged between the PUF and the NAF.
Clause 8	... a brief description of security aspects and functionality provided by this ETS.
Annex A	... the Bibliography. Informative references useful for the understanding of this ETS.
Annex B	... the mapping between the ISDN PCI messages and the underlying protocols.
Annex C	... details for NAFs providing external equipment support (telephony).
Annex D	... rules for use of the X.25 protocol.
Annex E	... guidelines for NAF developers and manufacturers giving guidance for implementation and extension of this ETS.
Annex F	... operating system dependencies and implementation rules for various operating systems.
Annex G	... the PCI ICS proforma: a template for conformance implementation conformance statements of the NAF developers.
Annex H	... definition of standard profile. This Annex provides the content of static attributes.
Annex J	... sample coding in C language illustrating operating system specific implementation of the exchange mechanism.
Annex K	... C language illustrating TLV encoding/decoding example.

4.3 Overview

4.3.1 Background

With countries of the European Community committed by their Memorandum of Understanding to implement one standard for ISDN throughout Europe, it is a logical step forward to define an API that provides access to this Euro-ISDN. This API is called the ISDN PCI. The goals of the ISDN PCI can be summarised as follows:

- provide access to Euro-ISDN, while not prohibiting access to existing ISDN implementations;
- allow for standalone and distributed operation;
- provide an interface that is capable of supporting multiple applications;
- provide an interface capable of providing support for multiple ISDN accesses;
- provide an interface that supports both Basic and Primary access;
- focus on an access to several protocols for data transfer based on CCITT Recommendation X.213 [7], while allowing the use of other protocols;
- provide co-ordinated operation of ISDN and the user plane protocols to provide the ISO Network Service interface (ISO 9574);
- provide an interface that is, as far as possible, operating system independent;
- define the ISDN PCI in sufficient detail to ensure binary compatibility between different implementations within the same operating system on the same platform;
- allow access to supplementary services provided by the ISDN;
- provide support for physical devices such as telephones;
- good performance.

The ISDN PCI can be seen as satisfying these goals and providing an interface that is highly suitable to:

- ISDN adapter manufacturers;
- ISDN application writers;
- ISDN users as a procurement requirement for selecting ISDN products and applications.

4.3.2 Functional overview

The basic model of the ISDN PCI consists of two entities: a service user called the PCI User Facility (PUF) and a service provider called the Network Access Facility (NAF). The PUF and the NAF interact by means of messages. Using these messages, the PUF requests the NAF to perform actions and to return the results to the PUF.

The ISDN PCI interface to the NAF, depending on the underlying protocols supported, is divided into 3 planes:

- a Control Plane which provides access to the services offered by ISDN;
- a User Plane which provides access to the protocols used to transfer the data over connections established through the ISDN;
- an Administration Plane which provides the mechanisms that support the objects and identifiers required by the other two planes.

Each plane groups a distinctive set of functionality which is exchanged through the ISDN PCI. The method for exchanging the information is called an Exchange Mechanism and is defined separately. This definition is generic in nature and may be applied to several operating systems. The adaptation of the Exchange Mechanism to a choice of specific, popular operating systems is covered in separate Clauses.

Apart from the use of non-connection related facilities, e.g. security features and support of external equipment, the use of the ISDN PCI is based upon the establishment of ISDN and user protocol connections. Within the PCI the concept of a Network Connection Object (NCO) is used to control these connections. NCOs are defined by the use of Administration Plane messages and used in both Control and User Plane messages to establish, use and remove connections.

4.3.3 Connection management

Performing ISDN connections leads firstly to set up the protocol related information required for this connection and then to unambiguously identify all messages attached to this connection. The specification provides a mechanism to link the attributes - static or dynamic - to the connection based on NCO management.

4.3.4 The planes

The Control Plane supports messages that allow the PUF to establish, control and remove connections, and to access the services provided by the ISDN. Five classes of message are defined. The first class is associated with the basic call set-up and shall be mandatory for the NAF to provide while the other 4 classes associated with telephony supplementary services, user-to-user signalling and adjournment of calls are optional.

The Administration Plane is responsible for managing attribute sets, addresses and NCOs. It offers messages that provide information concerning the state of any external equipment that the NAF controls, such as a telephone. It also provides messages to manage the security features used on a particular connection. Four classes of message are defined. The first, associated with the basic operation of the NAF, is defined as mandatory for the NAF to provide while the other 3 classes, associated with security and external equipment support and manufacturer specific features, are optional.

The User Plane provides messages that allow the use of underlying protocols. At present two sets of messages are defined. One set allows access to User Plane protocols providing the OSI Network-layer service interface. The other set provides a transparent interface where the application selects the protocol to be run over the connection.

4.3.5 Properties

The ISDN PCI defines that each NAF provide the PUF with a list of it's static properties. These properties define the capabilities of the NAF generally and the resources to which it has access to in a particular configuration. It is through these sets of properties that the PUF is informed of the variety of messages the NAF supports within each plane. As an example, these properties may refer to the types of external equipment available or the type and number of ISDN channel(s) available.

4.3.6 External equipment (i.e. telephony)

The ISDN PCI provides support of external equipment controlled by the NAF. The types of external equipment supported relate to various types of telephony equipment such as headsets and others. This support is achieved by treating the external equipment as a special type of transparent access so that when an ISDN connection is established (using the transparent protocol) the relevant ISDN channel for that external equipment is attached - this method is preferred rather than providing the PUF with User Plane messages. Administration Plane messages are provided to monitor the state of external equipment.

4.3.7 ISDN accesses and the multi-applications environment

The definition of the ISDN PCI supports various NAF - PUF configurations. It puts no constraints on the NAF implementation and it allows:

- a NAF to provide either only one or more than one ISDN accesses;
- a NAF to allow access to only one or more than one PUF concurrently;
- any number of NAFs to coexist within the same terminal equipment.

4.3.8 Exchange Mechanism

The realisation of an interface, like the ISDN PCI, across several distinct platforms (operating system) is often difficult to reconcile. Each operating system has a particular way of doing this and very often the same interface cannot be ported from one system to another. One way of going about solving this problem is by means of defining an Exchange Mechanism. The Exchange Mechanism abstracts the functionality between the interacting elements by means of functions. The ISDN PCI interface defines seven functions that allow the registration, de-registration and conversation of exchanges. During the registration phase, a function is usable to provide a list of the available NAFs within the system.

Messages passed to the NAF are copied before control is returned to the PUF. Once the PUF regains control, it is free to re-use any memory associated with the message. The memory allocation for data being transferred or received is performed by pointers inside the messages, this mechanism avoids making unnecessary copies. It is the responsibility of the PUF to provide memory for the NAF to place any messages. In order to assist the PUF in the provision of memory space, a signalling mechanism defined within each operating system is described; this mechanism consists of notifying the PUF when a message is available.

5 Functional model

5.1 Introduction

This Clause describes the functional model for the ISDN PCI. It introduces the architecture of the ISDN PCI and its location with respect to the OSI reference model. This Clause also describes the functionality of the ISDN PCI, the interactions between the entities located around the ISDN PCI. Furthermore, it describes sequencing of messages, to indicate in which way the entities may exchange information.

There is also a description of the identifiers involved in the ISDN PCI and the error mechanism it provides.

5.2 Architecture

The ISDN PCI is the specification of the communication interface inside terminal equipment which wishes to access an ISDN. Using this interface a higher layer entity may access the services of an ISDN network in a standardized way.

The ISDN PCI is a software interface between a service user and a service provider. As a software interface, the ISDN PCI consists of the specification of the interface and a description of the functionality which lies directly below the interface.

The ISDN PCI is an interface specification which is implemented in a real computer environment. This environment imposes problems, e.g. associating the entities and exchanging information between the entities. As a result, the ISDN PCI contains some functionality to deal with the problems of implementing it within a computer environment.

Two entities can be distinguished around the ISDN PCI. These are the service user and the service provider. These entities, along with the ISDN PCI and their information interchange is described in subclause 5.2.1.

5.2.1 ISDN PCI and its components

PUF

Throughout this ETS, the term PCI User Facility (PUF) is used to refer to the service user. It refers to all the functional layers which use the interface to access the services of the ISDN.

NAF

The term Network Access Facility (NAF) is used to refer to the ISDN PCI service provider. This service provider refers to all elements which are necessary to provide access to the services of ISDN. These elements can be both software and hardware. No distinction is made to this point. The NAF behaves as representing the services of one ISDN access.

ISDN PCI

The ISDN PCI defines the interface located at the top of the NAF(s). The ISDN PCI defines a number of functions. First, the ISDN PCI allows for the association between the PUF and the NAF. After the PUF and NAF are associated all the operations are performed by an Information Exchange Mechanism. The Exchange Mechanism is another part of the functionality of the ISDN PCI. Figure 1 describes how the ISDN PCI relates to the surrounding components. The arrow indicates the information flow.

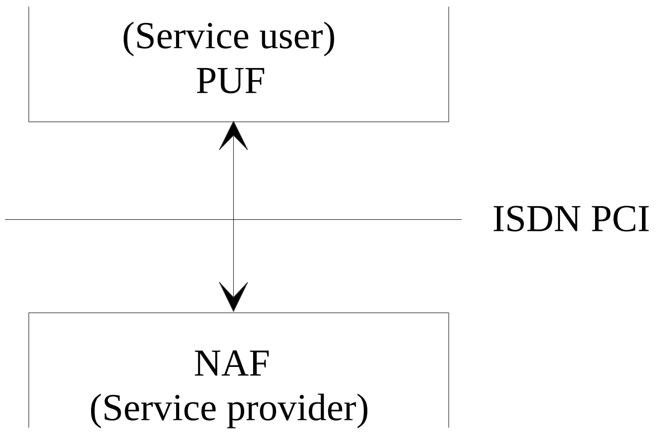


Figure 1: Functional picture of ISDN PCI with surrounding components

Query Entity

The PUF is bound to the NAF in order to be able to communicate. The binding mechanism describes how the PUF can get contact to a particular NAF within the local computer environment. The Query Entity is defined as an instance, that provides the relevant information for binding.

Messages

Accessing the functionality described by the ISDN PCI is achieved by means of messages. The PUF and NAF use the functionality of the Information Exchange Mechanism to exchange messages. The messages inform the entities of the operations to perform, or the results of performed operations.

5.2.2 ISDN PCI architecture

The ISDN PCI has its own structure. This structure consists of three planes, which form the functional separation of functionality. Each plane has its own set of messages. The ISDN PCI distinguishes the following planes:

- **Control Plane**
  the Control Plane is related to the signalling part of a connection, which is, via the NAF, associated with the signalling in the ISDN D-channel. It covers the functionality provided by the service in the D-channel, such as connection control, control of service characteristics, supplementary services;
- **User Plane**
  the User Plane is related to the user connection, which may either be associated with a connection on the B-channel or a data connection on the D-channel. It is associated, via the NAF, with the functionality provided by the data services in the D- and B-channels, which consists of services for end-to-end data exchange;
- **Administration Plane**
  the Administration Plane does not relate to ISDN. It covers the required functionality for control and configuration of the Control Plane and User Plane. Furthermore, the Administration Plane is responsible for managing special equipment accessible through the ISDN PCI.

Figure 2 gives a representation of the three planes.

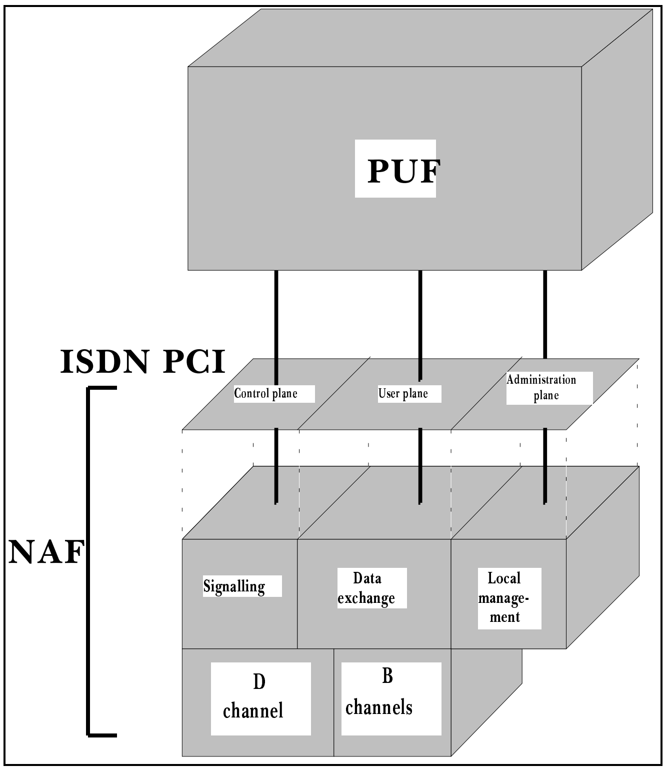


Figure 2: Relation between planes and ISDN

5.2.3 OSI location

The ISDN PCI is located at the boundary between layers 3 and 4 of the OSI reference model. The signalling part of ISDN is defined up until layer 3. As an interface specification to ISDN, the ISDN PCI is also located at this boundary. When locating the ISDN PCI at exactly this boundary, all the services of the signalling part (ISDN call control) are accessible on this point.

The Control Plane provides the services as defined in ETS 300 102-1 [2] and ETS 300 196 and is, therefore, located at the upper boundary of the D-channel protocol. ETS 300 102-1 [2] is the ETS which defines basic call control in Euro-ISDN. ETS 300 196 is the ETS which defines the support of supplementary services in Euro-ISDN.

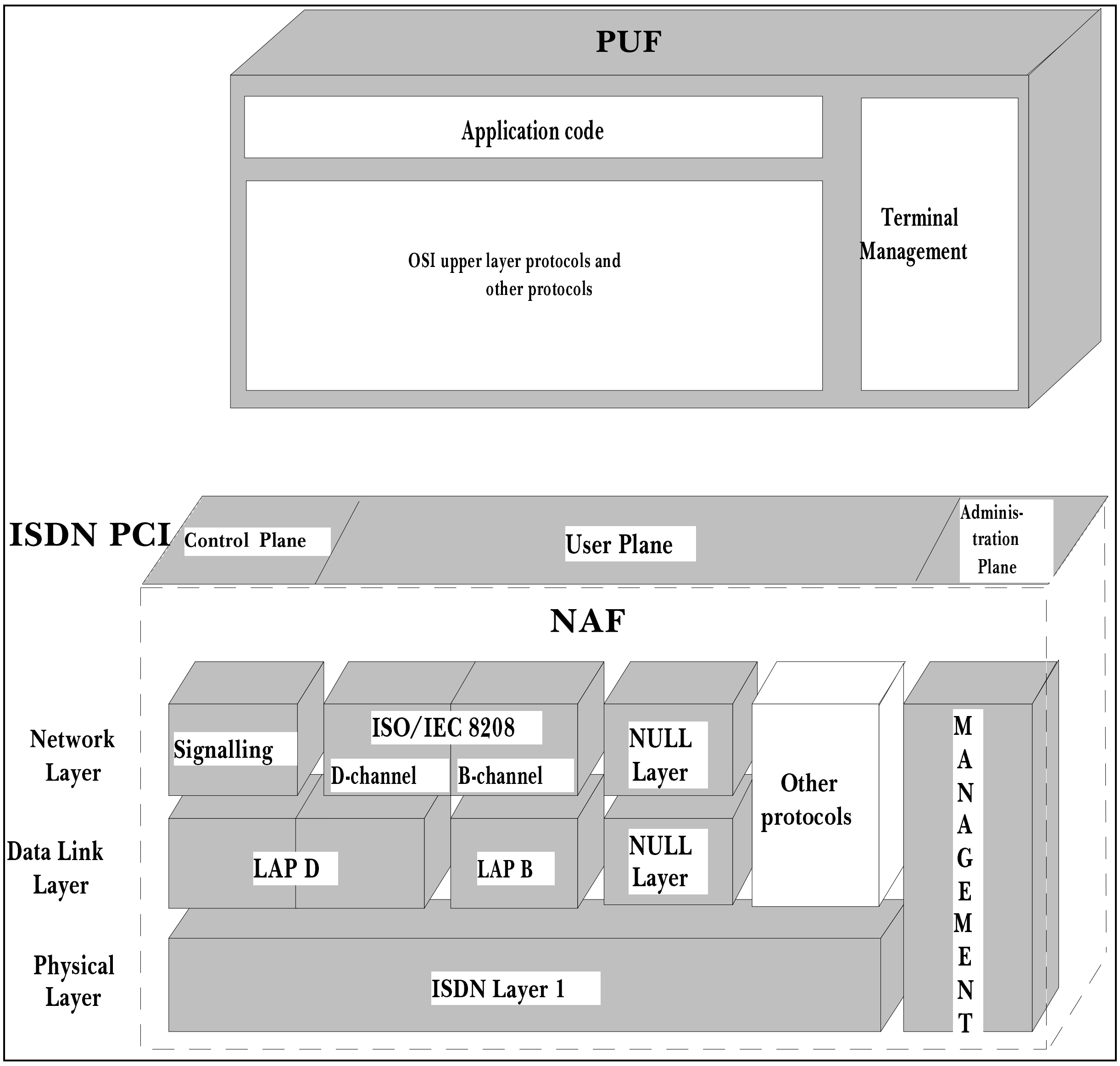
The User Plane provides the services for a variety of protocols.

The User Plane provides the services defined in CCITT Recommendation X. 213 [7] and is, therefore, located at the Network layer Service Access Point (N-SAP). The User Plane provides this access using the User Plane protocols on a connection in the B-channel or on a data connection in the D-channel dependent on the protocol. The User Plane also allows for access of an other Service Access Point (SAP). For the support of transparent access to the ISDN B-channel, the User Plane provides access to the Physical Service Access Point (Ph-SAP).

In the case of transparent access, the NAF considers layers 2 and 3 as Null layers, as shown in figure 3.

The Administration Plane is outside the scope of the OSI reference model. The Administration Plane only has a local impact. It takes care of the management functionality required by the User Plane, Control Plane and external equipment accessible through the ISDN PCI.

Figure 3 gives a view on the direction of the ISDN PCI relating to the OSI reference model.



NOTE: The reference to ISO/IEC 8208 [3] in the figure 3 is only an example of a supported CCITT Recommendation X.213 [7] User Plane service.

Figure 3: OSI location of the ISDN PCI

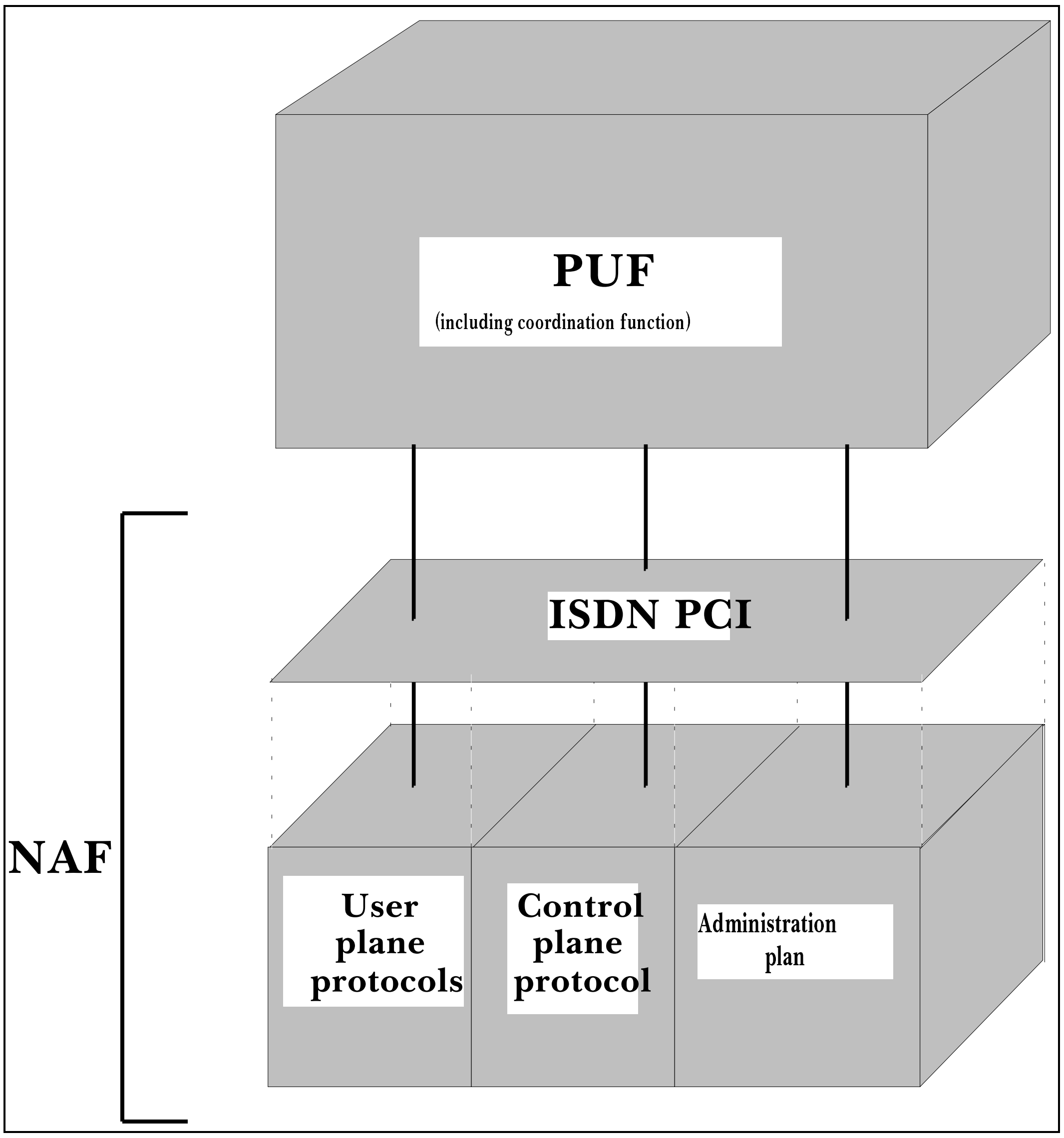
5.2.4 Co-ordination cases

This ETS covers two ways to access the data exchange functionality provided by ISDN on layer 3 of the OSI reference model. Both ways shall be provided by the NAF and may run simultaneously.

The other data exchange functionality which the ISDN PCI is providing is a transparent access on a connection by connection, as described in subclause 5.5.4

Case 1: PUF co-ordination

The ISDN PCI provides direct access to signalling and to the user connection, associated with the D- and B-channels of ISDN. A PUF which uses this method shall handle the establishment of a user connection by using the basic call control provided by the Control Plane. The co-ordination between signalling and user connection shall be handled exclusively by the PUF. Figure 4 shows the PUF provided co-ordination function. As a result of controlling the signalling connection, the PUF can use the supplementary services.



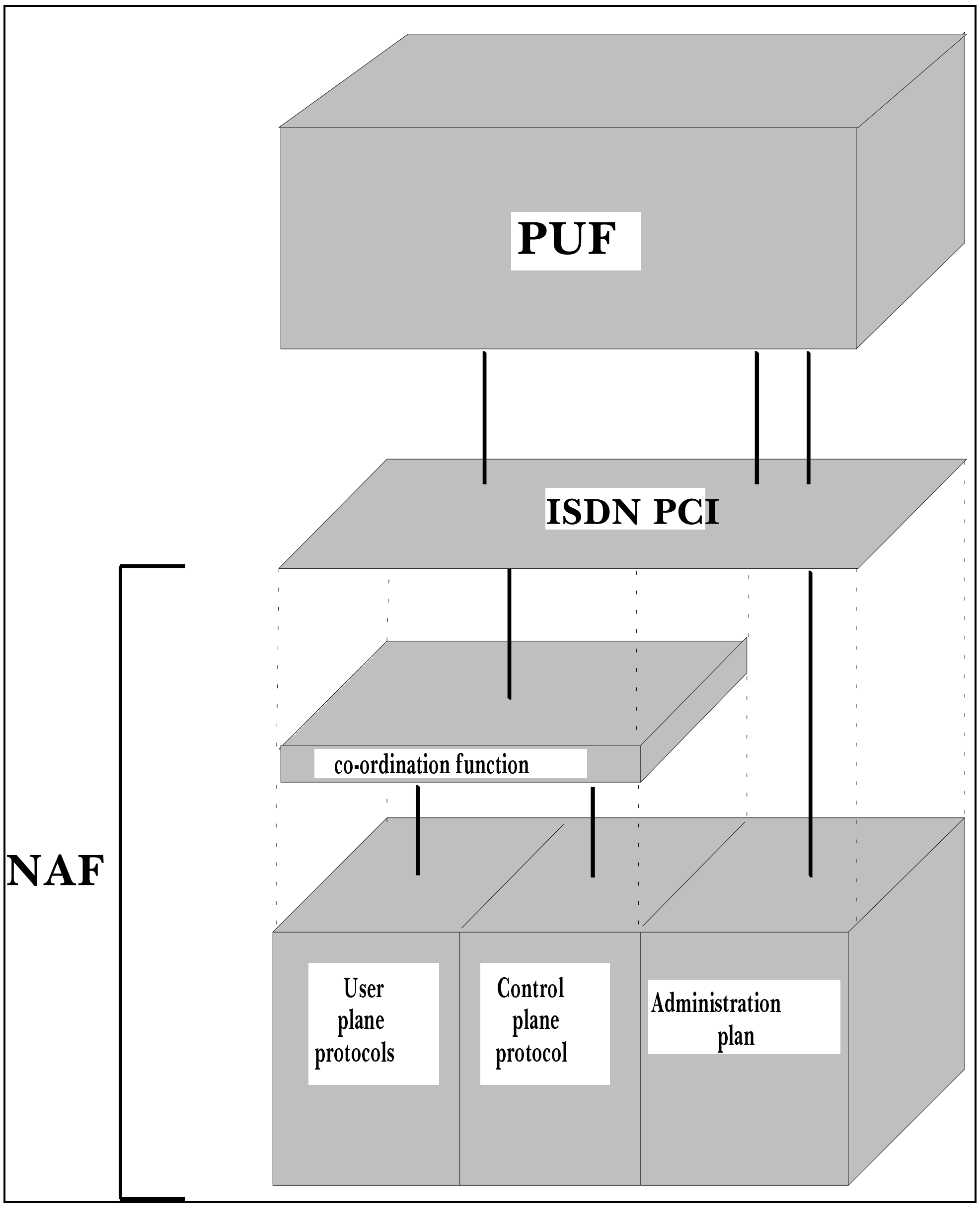
NOTE: The co-ordination is performed either implicitly or explicitly inside the PUF. The existence of a co-ordination function inside the PUF is outside the scope of this ETS.

Figure 4: Case 1 PUF co-ordination

Case 2: NAF co-ordination

In case 2, the ISDN PCI abstracts from the separation between signalling and user connection, as provided in ISDN via the D- and B-channels. The PUF is offered an ISO Connection-mode Network Service (CONS) as defined in CCITT Recommendation X.213 [7]. This abstraction is provided by a co-ordination function, which maps the primitives of CONS in the User Plane according to the primitives of the Control Plane and User Plane protocols. The co-ordination function can only be used with the User Plane protocols relating to CCITT Recommendation X.213 [7]. The use of the co-ordination function with transparent access is for further study.

The co-ordination function is provided as part of the NAF. Since the NAF manages the co-ordination between signalling and user connection, the PUF shall not access the Control Plane. Figure 5 shows the NAF provided co-ordination function.



NOTE: The co-ordination function is only defined for User Plane protocols related to CCITT Recommendation X.213 [7]. The use of the co-ordination function for transparent access is for further study.

Figure 5: Case 2-NAF co-ordination

5.3 Functionality

5.3.1 Introduction

As described in subclause 5.2.2, the ISDN PCI functionality is provided by the three planes, with associated message sets to access the functionality. How the exchange of messages between PUF and NAF takes place is described in subclause 5.5.

In order to access ISDN signalling or data, the PUF shall create a NCO. The creation and destruction of network connection is the main part of the functionality of the resource management which is described in subclause 5.3.2.

After having performed this successfully, the PUF is in an "idle" state and may subsequently access ISDN signalling (except in case 2, NAF co-ordination) or transfer data. Subclauses 5.3.3 and 5.3.4 respectively, describe the functionality for connection management and data management.

5.3.2 Resource management

The first functionality which is described is resource management. This functionality is needed to be able to use the ISDN PCI for communication. The resource management contains functionality for local management. The functionality covers the management of:

- Network Connection Objects (NCOs);
- external equipment.

The Administration Plane of the ISDN PCI provides the functionality defined by the resource management.

The resource management revolves around the NCO, which is the object needed for subsequent communication. An NCO refers to an abstract object containing all relevant configuration information for one user connection. The configuration information for an NCO shall be assigned by the PUF using one of two principle methods:

a) referencing a standardized attribute set;
b) specifying all configuration information during NCO creation.

Method a) provides a simple way for the PUF to select appropriate configuration information by referencing a standardized attribute set identifier. However, this method is available at the cost of flexibility, since attribute sets are standardized and may only be used in the provided manner. Annex H gives the list of standardized attribute sets.

Method b) gives the PUF the opportunity to specify configuration information for any special needs on its own. However, this involves a lot of details concerning D-channel and B-channel parameters and shall, therefore, be left for the sophisticated PUF-implementor.

5.3.2.1 Attribute sets

Attribute sets are used to keep together important parameters for driving user protocols, for driving the ISDN signalling protocol and for collecting some management information relevant to the NCOs (statistics, cost, ...). User protocols and ISDN signalling is accessed through the functionality of the User Plane and the Control Plane. A collection of attribute sets exists for both planes. These sets are:

- signalling attribute set (related to the Control Plane);
- user protocol attribute set (related to the User Plane);
- administration attribute set (related to the Administration Plane).

The Administration attribute set is not involved in the NCO creation but is only updated during the life of the NCO and can be accessed at any time through the resource management. It can be seen as king of the dashboard of the NCO.

The resource management offers functionality to reference specific attribute sets when creating an NCO.

5.3.2.2 Network connection objects

The resource management functionality covers:

- the creation of an NCO;
- the grouping of NCOs.

An NCO is an abstract object created by the NAF in response to requests by the PUF prior to the establishment of a connection.

As a rule there is one NCO per connection, independent of which type of connection the NCO is related to. This can be a signalling connection or a connection for data transfer.

After the successful creation of a NCO a unique identifier, the NCO Identifier (NCOID), becomes available. This NCOID shall be supplied in subsequent operations regarding connection establishment and data transfer.

At the creation time of an NCO, the PUF can indicate that the newly created NCO should be grouped to another already existing NCO.

The purpose of the grouping is to provide the ability to share a channel when using a network layer protocol, which allows sharing of several logical connections on one physical channel. The sharing is reserved to one PUF.

The grouping of the NCOs is done by using the Group-ID. A unique Group-ID shall be returned on the successful creation of an NCO. This Group-ID can subsequently be supplied at the creation of an additional NCO, which shall then be grouped to the first NCO. If no Group-ID is supplied, the NCO shall not be grouped. The Group-ID is only guaranteed to be unique for the interaction between the PUF and the NAF.

As the GroupID is only unique for the PUF NAF relation, multiple PUFs which access the same NAF cannot share the same connection.

For an incoming call, the NAF selects the appropriate NCOs and is then helped by the PUF to choose the unique one. This is done using the SelectorID, supplied at the creation of the NCO. This gives the PUF the opportunity to handle a list of NCOs that the NAF will exclusively deal with.

In case of a non co-ordinated NCO (C/U3), the User and Control Planes may have different directions. For example, the User Plane may be listening, while the Control Plane is calling.

5.3.2.3 Support of external equipment

Access to external equipment, such as telephones, is provided to the PUF through the functionality of the three planes.

As long as an NCO that specifies an external equipment in its configuration information exists, the NAF shall generate the appropriate Control Plane messages if the state of that external equipment changes.

The connection management (see subclause 5.3.3) and data management (see subclause 5.3.4) provide functionality to manage connections with these NCO.

Five types of external equipments are defined:

1) external equipment without telephony hook mechanism. This type of external equipment only contains the transceivers. In this case, the PUF is responsible for the management of the ISDN connection;

2) external equipment with telephony hook mechanism. In this case, all telephony hook mechanism events are available at the PCI level and the PUF is responsible for the management of the ISDN connection;

3) external equipment with telephony hook mechanism and which is able to manage the ISDN connection. In this case all telephony hook mechanism events are available at the PCI;

4) external equipment with keypad and with or without telephony hook mechanism. In this case all dialling events, all telephony hook mechanism events are available at the PCI and the PUF is responsible for the management of the ISDN connection;

5) external equipment with keypad and with or without telephony hook mechanism which is able to manage the ISDN connection. In this case, all dialling events, all telephony hook mechanism events and informations about the status of the communication are available at the PCI.

All these types of external equipments are connected to the NAF by the means of a proprietary connection which is outside the scope of this ETS and provide to the PUF the availability or not of the external equipment.

In the case of type 4 and 5 external equipments, two types of dialling are possible:

- blocksending: one Control Plane message containing the complete destination address is provided to the PUF;

- overlap sending: one Control Plane message per key pressed is provided to the PUF. During a communication, Dual Tone Multi Frequency (DTMF) codes can be sent via the keypad.

Type 3 external equipments are able to deal with incoming calls alone when the computer is off.

Type 5 external equipments are able to deal with incoming and outgoing calls when the computer is off.

Each action to the handset generates a Control Plane message to the PUF. Depending on the type of external equipment, different level of messages are sent to the PUF:

- for type 1 external equipment:
  - availability/unavailability;
- for type 2 and 3 external equipment:
  - availability/unavailability;
  - on-hook;
  - off-hook.
- for type 4 and 5 external equipment:
  - availability/unavailability;
  - on hook;
  - off hook;
  - a code representing the key pressed on the keypad in the case of an overlap sending;
  - a table of codes representing the complete destination address in the case of a block sending.

In the case of type 2 and 3 external equipments and if the PUF has created an NCO that specifies an external equipment in its signalling attribute set, a connection which involves this external equipment may be established and breakdown in different ways:

- case of the outgoing calls:
  - the user off-hooks the handset and the PUF issues the overlap or block dialling;
  - the PUF issues the overlap or block dialling and the user off-hooks the handset;

- case of the incoming calls:
  - the user off-hooks the handset and the PUF receives a Control Plane message to inform it;
  - the PUF answers to the incoming call and the user off-hooks the handset;

- case of local breakdown:
  - the user on-hooks the handset and the PUF receives a Control Plane message to inform it;
  - the PUF releases the call and the user on-hooks the handset;

- case of remote breakdown:
  - the PUF receives a Control Plane message and the user on-hooks the handset.

In the case of type 4 and 5 external equipments and if the PUF created an NCO that specifies an external equipment in its signalling attribute set, a connection which involves this external equipment may be established and breakdown in different ways:

- case of the outgoing calls:
  - the user off-hooks the handset and the PUF issues the overlap or block dialling;
  - the user off-hooks the handset which generates a Control Plane message to the PUF and uses the keypad of the external equipment to issue the overlap dialling. Each key pressed generates a Control Plane message to the PUF;
  - the user off-hooks the handset which generates a Control Plane message to the PUF and uses the keypad of the external equipment to issue the block dialling. The end of the destination address is detected by the means of a special key. A Control Plane message is generated to the PUF;

  - the PUF issues the overlap or block dialling and the user off-hooks the handset;

- case of the incoming calls:
  - the user off-hooks the handset and the PUF receives a Control Plane message to inform it;
  - the PUF answers to the incoming call and the user off-hooks the handset;

- case of local breakdown:
  - the user on-hooks the handset and the PUF receives a Control Plane message to inform it;
  - the PUF releases the call and the user on-hooks the handset;

- case of remote breakdown:
  - the PUF receives a Control Plane message and the user on-hooks the handset.

5.3.2.4 Support of security features

Access to security features below the ISDN PCI is provided to the PUF through the functionality of the Administration Plane.

The security features provided through the ISDN PCI cover the use of security algorithms on connections.

The PUF can activate and deactivate security features on a specific connection by supplying the NCO of the connection in Administration Plane messages.

5.3.2.5 Support of manufacturer specific features

Access to manufacturer specific features is provided to the PUF through the functionality of the Administration Plane.

The PUF can access manufacturer specific features by using this functionality. It is a way to handle extra functionality not provided by the ISDN PCI.

The information exchanged between PUF and NAF is dependant of the implementation of the feature and is, therefore, not covered in this ETS.

5.3.3 Connection management

The second functionality is the connection management. It covers:

- connection set-up and removal;
- supplementary services.

The connection set-up and breakdown covers the basic functionality of the connection management. The supplementary services provide additional functionality related to the connection management.

The Control Plane of the ISDN PCI provides the functionality defined by the connection management.

5.3.3.1 Connection set-up and removal

The only way for a PUF to achieve a connection is to enter the "idle" state by the creation of an NCO. Subsequently, it can perform a connection request or wait for a connection indication. After the connection is removed, the PUF returns to the "idle" state and can subsequently re-use the NCO for a new connection. The NCO becomes invalid if it is destroyed or if the PUF deregisters from the NAF.

At the creation of an NCO, the PUF shall decide which type of connection is to be achieved. The ISDN PCI provides for access to:

- signalling connection, running the Euro-ISDN signalling protocol;
- connection for information transfer, optionally running communication protocols.

For signalling connections the ISDN PCI provides functionality to set-up and breakdown connections. The functionality is covered by one message access at the top of the layer 3 of the signalling protocol. This message access is called Signalling Message Access (SMA).

If the PUF has created an NCO associated with external equipment, the ISDN PCI provides functionality to set-up and breakdown connections and all user actions with the external equipment (on-hook, off-hook, dialling) are taken into account for the signalling part. Furthermore, some external equipments are able to manage ISDN signalling when the host is off.

In the case of telephony, additional functionality can be available, which allows the temporary breakdown (suspend) and subsequent re-establishment of connections (resume).

As an NCO is coupled to a single PUF, connection passing between PUFs cannot be accommodated.

5.3.3.2 Support of supplementary services

The supplementary services provide additional functionality related to the connection management.

Supplementary services, as provided by the connection management of ISDN, are available to the PUF when Case 1 of subclause 5.2.4 applies. The PUF is responsible for the handling of the connection management and can, therefore, control the supplementary services provided via the signalling.

NOTE: The use of supplementary services, when using the co-ordination function as described in Case 2 of subclause 5.2.4, is for further study.

In this ETS the following supplementary services are identified and described:

- Multiple Subscriber Number (MSN) (ETSs 300 050, 300 051 and 300 052);
- Direct Dialling In (DDI) (ETSs 300 062, 300 063 and 300 064);
- Calling Line Identification Presentation (CLIP) (ETSs 300 089, 300 091 and 300 092);
- Calling Line Identification Restriction (CLIR) (ETSs 300 090, 300 091 and 300 093);
- Subaddressing (SUB) (ETSs 300 059, 300 060, 300 061);
- Advice of Charge during call (AOC-D) (ETS 300 179);
- Advice of Charge at end of call (AOC-E) (ETS 300 180);
- Call Waiting (CW) (ETSs 300 056, 300 057 and 300 058);
- Connected Line Identification Presentation (COLP) (ETSs 300 094, 300 096 and 300 097);
- Connected Line Identification Restriction (COLR)(ETSs 300 095, 300 096 and 300 098).

The first four supplementary services are the supplementary services identified in the ISDN MOU as being of priority one. This means these services shall be available in every ISDN, and are, therefore, included in the ISDN PCI. Only Terminal Portability (TP), as a priority one service, is not included in the ISDN PCI.

The latter six supplementary services were identified as adding useful functionality to the ISDN PCI.

For all these supplementary services a special way of coding has been introduced in the ISDN PCI, to facilitate the use of these services.

Any other supplementary services may be provided by the transparent coding of the supplementary services. This is described in ETS 300 196 and the ETSs for supplementary services. The transparent coding shall be provided as an optional feature.

5.3.4 Data management

The data management covers the functionality to:

- establish data connections on already established physical connections;
- exchange data.

The User Plane of the ISDN PCI provides the functionality defined by the data management.

For user data transfer the ISDN PCI provides two message accesses at the top of the ISDN network layer:

- Network layer Message Access (NMA);
- Transparent Message Access (TMA).

The NMA provides access to the User Plane protocols running in the ISDN network layer. Thus, it provides access to a network layer connection.

The TMA provides access to the transparent network and link layers (NULL Layers) and thus provides direct access to the physical layer of ISDN, providing a byte synchronized control over a B-channel. The byte synchronized control is not restricted to the digital bearer service.

For both types of connection it is important that there exists a signalling connection before any data access can be done. In general, establishment of that signalling connection is achieved by use of Control Plane functionality, whereas the establishment of the data access is achieved by use of User Plane functionality.

When using a connection on the TMA with an NCO which is associated with external equipment, the data generated on the connection shall be sent to the external equipment rather than used to generate TMA messages.

5.3.4.1 Connection via the Network layer Message Access (NMA)

The ISDN PCI provides co-ordination functionality for the NMA which removes the need for the PUF to use Control Plane functionality. This co-ordination functionality, which is available to the PUF on demand, implicitly builds a signalling connection when a user connection is requested. However, the PUF may decide not to use the co-ordination function and to establish the signalling connection by itself through the use of Control Plane functionality via the Signalling Management Access (SMA).

5.3.4.2 Connection via the Transparent Message Access (TMA)

Due to the nature of the TMA, which transparently accesses the physical ISDN layer, no user protocol is running. Thus, by establishment of a signalling connection the transparent data access becomes available. Unlike the access via the NMA, only one data connection is accessible per signalling connection.

5.4 Relating functionality to planes

When relating the functionality as described in subclause5.3, the following relations apply:

- the Administration Plane of the ISDN PCI provides the functionality defined by the resource management;
- the Control Plane of the ISDN PCI provides the functionality defined by the connection management;
- the User Plane of the ISDN PCI provides the functionality defined by the data management.

Inside the planes the functionality is described using operations or operational groups.

5.4.1 Optional features

When relating the functionality to the planes, there shall be some operations or operational groups which shall not be mandatory for a NAF to supply.

In the description of the planes there are to be indications of which operations or operation groups are mandatory or optional.

The fact that the description allows optional features for the NAF does not mean that the ISDN PCI contains any optional features. The ISDN PCI, as an interface specification, shall allow the exchange of any message. Optional here refers to the availability of these features to the PUF, supplied by the NAF. If the PUF requests a feature which is not provided by the NAF, the PUF shall be informed of this.

5.4.2 Administration Plane

The Administration Plane provides access to operations which facilitate management of connections e.g. definition and management of attribute and address sets as well as management of network connection objects. Furthermore, the following miscellaneous operations are provided via this plane:

- error report operation;
- security operation;
- manufacturer specific operation.

Table 2 provides an overview on Administration Plane operations.

Table 2: Administration Plane operations

Operation name	**Purpose of operation**
Create NCO	Create a network connection object
Destroy NCO	Destroy a network connection object
GetInfo NCO	Obtain information about a network connection object
Error	Report non connection related error condition
Security (NOTE)	Manipulate security
Manufacturer Specific (NOTE)	Request manufacturer specific functionality
NOTE: These operational groups are optional for the NAF.	

5.4.3 Control Plane

The Control Plane provides access to operations which handle the basic call control of the ISDN signalling.

In the Control Plane, no clear separation of operations exists c.f. in the Administration Plane. It shall be possible to distinguish between a number of operational groups in the Control Plane. Table 3 provides an overview on Control Plane operational groups.

Table 3: Control Plane operations

Operational group name	**Purpose of operational group**
Connection establishment	Handling incoming and outgoing calls
Connection breakdown	Handling of removal of connections or refusal of calls
User-to-user information transfer (NOTE)	Exchanging user-to-user information and providing control for this exchange
Adjournment of calls (NOTE)	Provision of suspending and resuming calls
Facility invocation (NOTE)	Handling the invocation of facilities
External equipment (NOTE)	Indicate status or change of state of external equipment.
NOTE: These operational groups are optional for the NAF.	

5.4.4 User Plane

The User Plane provides operations which facilitate establishment, data exchange and release of logical communication channels. It uses standardized services and procedures as defined for the selected user message access.

In the following subclauses, the different methods for message access which are supported by the ISDN PCI are explained. Subclause 5.4.4.2 gives information on how the extendibility of the ISDN PCI should be seen. It indicates how a new message access can be added to the ISDN PCI.

5.4.4.1 The transparent access

As stated in subclause 5.3.4, the ISDN PCI supports a Transparent Message Access (TMA).

As with any message access, this message access offers it own set of operations. Table 4 provides an overview on User Plane operations for this message access.

Due to the nature of the TMA, only operations allowing direct byte stream access are provided for this message access.

Table 4: User Plane operations for transparent access

Operation name	Purpose of operation
Data	Data transfer (byte synchronised)
Error	Indicates an error has occurred

5.4.4.2 The network layer access

As stated in subclause 5.3.4, the ISDN PCI supports a Network layer Message Access (NMA).

There are different user layer protocols which can be accessed through the NMA. One of these User Plane protocols is selectable at the creation of the NCO. The available user protocols are listed in table 6.

Independent of the user protocols, the message access offers it own set of operations. Table 5 provides an overview on User Plane operations for this message access.

The operational set of the NMA is based on CCITT Recommendation X.213 [7]. It offers the possibility to access ISDN data without the need for establishing a signalling connection explicitly. This can be done by using the co-ordination function (Case 2 of subclause 5.2.4). When the co-ordination function is not used (Case 1 of subclause 5.2.4) the functionality of the User Plane can be accessed in the same way along with controlling the signalling connection.

Table 5: User Plane operations for network layer access

Operation name	**Purpose of operation**
Connect	Establish a peer-to-peer user connection
Data	Exchange data over an established user connection, hereby relying on flow control provided by underlying protocol
Expedited data	Exchange data over an established user connection without relying on flow control provided by underlying protocol
Data acknowledge	Acknowledgement of data reception over an established user connection
Reset	Clearing of data transfer
Disconnect	Disconnect connection
ReadyToReceive (NOTE)	Control the normal data flow
NOTE: This operation is not based on CCITT Recommendation X.213 [7].	

Table 6 lists the protocols which are supported by the NMA. The table lists whether it is mandatory (M) or optional (O) for each protocol supported by the NAF. The two Null layers presented in table 6 are represented in Clause 6 with only one NULL box in the protocol statements with the User Plane messages and the User Plane parameters.

Table 6: Supported User Plane protocols

Protocol	Supported by the NAF
Network layer according to ETS 300 080 [1]	M
ISO/IEC 8208 [3]	M
Network layer of CCITT Recommendation T.70 (NOTE)	O
Null layer 3 with access to CCITT Recommendation X.75 on layer 2	O
Null layer 3 with transparent access to HDLC framing	O
NOTE: In this ETS, whenever CCITT Recommendation T.70 is referenced, T.70NL is implied.	

5.5 PUF NAF interactions

This subclause describes the type of functions which are available to the PUF in its interactions with a NAF and in which order they can be used.

For all functions the following properties apply:

- initiated by the PUF, which means that only the PUF can initiate the association the PUF to the NAF;
- requested by using function calls from the PUF to the NAF;
- performed in a synchronous manner.

A PUF which requests a NAF to perform a function shall regain control of the CPU from the NAF after completion of the function call.

In the interaction between PUF and NAF the following phases can be distinguished:

- Registration Phase

  Before a PUF and a NAF can interchange information, the PUF associates with the NAF. As it is possible that within a system more than one NAF may be available, and, additionally, these may be from different NAF manufacturers, a method is defined which allows the PUF to discover which NAFs are accessible within a system. This phase is called the Registration Phase. This phase allows access to a list of accessible NAFs via the PCI-Handles. Then the PUF may discover properties of the NAF that have been selected by the PCI-Handle and establish an association to the NAF.

- Conversation Phase

  At this point PUF and NAF can exchange messages. This phase is called the Conversation Phase. The PUF controls the exchange of messages between the NAF and itself. This means that the PUF fills the message with relevant parameters and gives it to the NAF for processing, or the PUF asks the NAF to receive a message by providing resources to the NAF.

  There are two methods for a PUF to discover that the NAF has a message for it. The simplest way for the PUF to get available messages is to poll the NAF. The second method provides a mechanism to give the NAF a fast way to notify a PUF that a message is available. With this method the PUF explicitly allows the NAF to notify it on the availability of a message. This method has the advantage of introducing a efficient way of operating, for PUFs which are concerned with performance.

  For example, this method shall help PUFs which have bound to multiple NAFs. However, PUFs who use this method shall be more complex in design than those who do not.

- De-registration Phase

  When a PUF does not need to exchange messages with a NAF, it disassociates from the NAF. This phase is called the De registration Phase. This phase is important in terms of resource management in the NAF, especially for memory resources. The PUF shall disassociate to guarantee an efficient use of the global system.

Table 7 gives the list of functions grouped into their respective phases. The following subclauses give more details on the functionality of the different phases.

Table 7: ISDN PCI functions grouped into phases

Phase	Function	Purpose of function
Registration	PciGetHandles	Provide a list of accessible NAFs and obtain their PCI-Handles
	PciGetProperty	Provide detailed information on a NAF
	PciRegister	Associate the PUF to the NAF
Conversation	PciPutMessage	Transfer a message from the PUF to the NAF
	PciGetMessage	Ask the NAF to receive a message, by providing resources
	PciSetSignal	Establish mechanism to allow the NAF to notify the PUF when a message is available
De registration	PciDeregister	Disassociate the PUF from the NAF

The functions shall be used in a certain order. Figure 6 presents a state diagram for the ISDN PCI function calls.

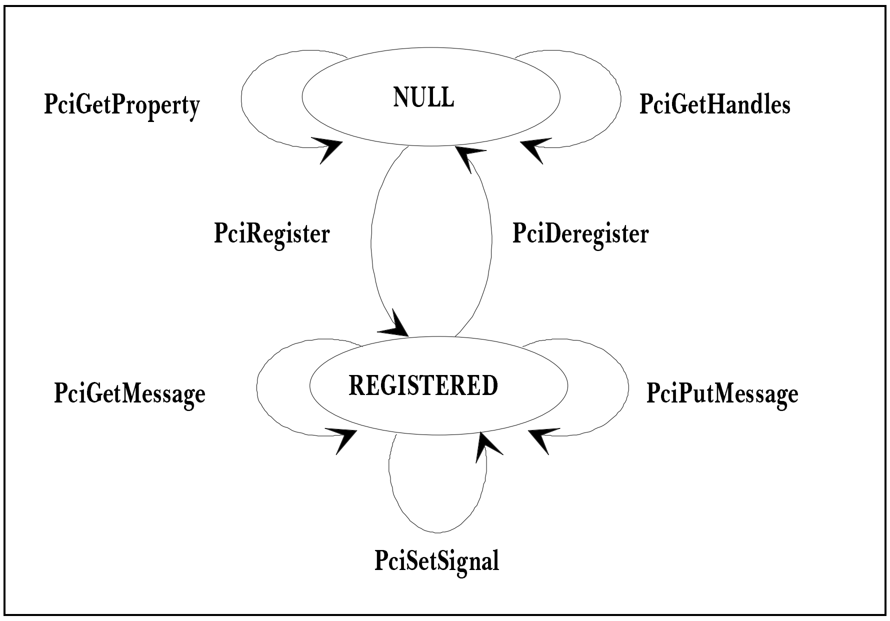


Figure 6: ISDN PCI function calls order

The messages are transferred between the PUF and the NAF by use of the PciPutMessage and PciGetMessage functions.

5.6 Total interaction overview

As an example of the sequencing of operations, figures 7 and 8 present a chronological interaction overview of the actions the PUF shall to perform to get a connection.

In these figures, the following conventions are used:

- for the complete figure, dashed lines mean optional;
- in the part of the figure on Conversation Phase, arrows from the PUF to the NAF mean usage of PciPutMessage and arrows from the NAF to the PUF mean usage of PciGetMessage;
- text written in small letters refers to messages as described in Clause 6 of this ETS.

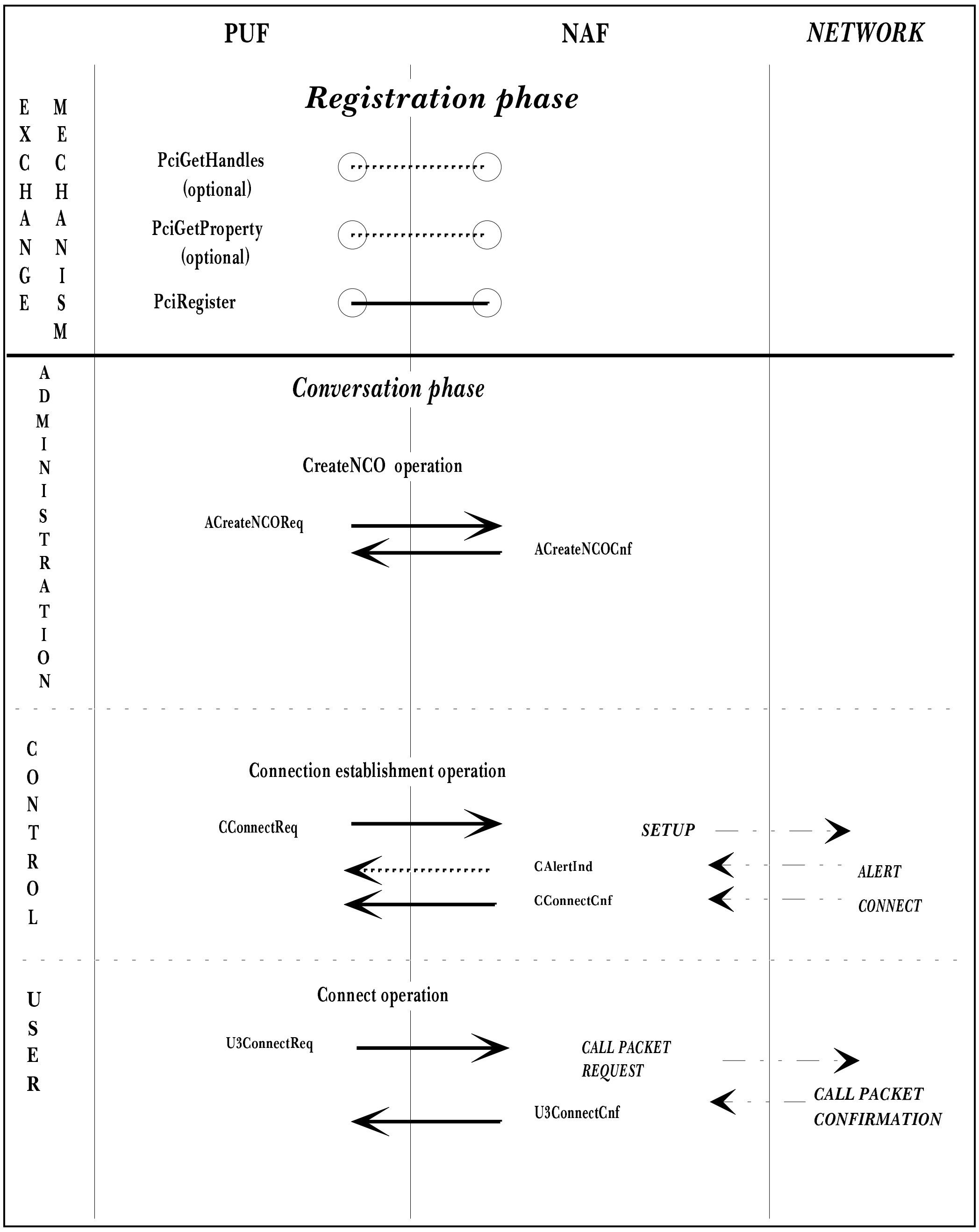


Figure 7: Sample of sequencing operations - non co-ordination case

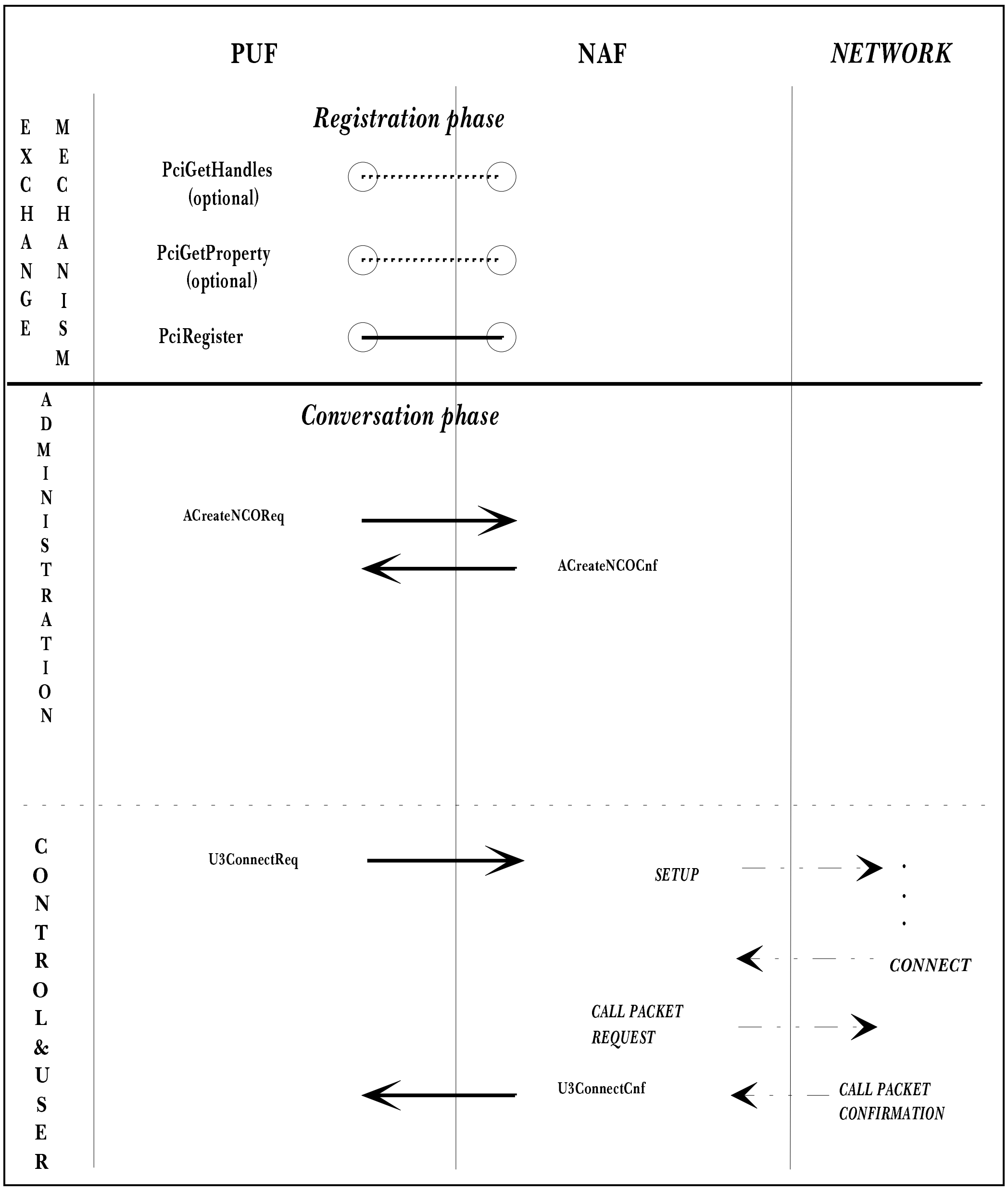


Figure 8: Sample of sequencing operations - co-ordination case

5.7 Identifiers

For its operations, the ISDN PCI defines identifiers. These identifiers shall be used by the PUF to identify concrete objects or connections in an abstract manner.

This subclause summarizes the identifiers used in the ISDN PCI specification. Only the functional description of these identifiers are given. The details are introduced in later Clauses.

For details of identifiers, see Clause 7 related to the Exchange Mechanism.

PCI-Handle	This identifier is an abstract reference to a NAF. The Handle shall be used to optionally find out information on the NAF and to register to the NAF from the exchange mechanism function **PciRegister**.
ExID	This identifier is the representation of the associating between a PUF and a NAF. It is provided by the exchange mechanism function **PciRegister**. It is needed in every ISDN PCI function call in relation with this associating.

Identifiers related to the Conversation Phase of the communication between PUF and NAF. For details of these identifiers refer to Clause 6.

CAttributeName	This identifier relates to a static attribute set of Control Plane parameters. It is represented as a name. This identifier shall subsequently be used in creating an NCO. The complete list of static attribute sets is defined in Annex H.
UAttributeName	This identifier relates to an attribute set of User Plane parameters. It is represented as a name. This identifier shall subsequently be used in creating an NCO. The complete list of static attribute sets is defined in the Annex H.
ExtEquipName	This identifier relates to external equipment. It is represented as a name. This name may be obtained either implicitly or by use of the PciGetProperty function.
NCOID	This identifier relates to the connection which is referred to by the PUF. It is the way for the PUF to indicate to the NAF which connection is referred to Since a connection always corresponds with an NCO, this identifier is a reference to this NCO. The NCOID is provided in response to the Create NCO message.
GroupID	Abstract identifier for grouping. It is only useful for the level 3 user connections.
RequestID	This reference identifies the message which is exchanged between the PUF and the NAF in the Administration Plane. Subsequent responses to this message shall contain the same RequestID to identify the original message. It is allowed multiple asynchronous transmissions to this plane.
SelectorID	This reference identifies the NCO related to the message in case of multiple NCOs matching, on an incoming call. The NAF shall select only one NCO in this abstract PUF set, indicated by the same SelectorID value. This is a way for the PUF to limit the amount of NCO selected by a NAF and then to limit the number of messages generated by the NAF in case of an incoming call.

5.8 Error handling

5.8.1 Overview

Error information is returned to the PUF by the means of function return codes and information present within messages. Generally, the function return codes provide error information generated by the passing of parameters to the NAF from the PUF and the checking of those parameters data. Messages contain error information reflecting the checking of the data referenced by the parameters, the processing of earlier messages or events from the protocols in use.

5.8.2 Function error handling

For each function the supplied parameter values are checked, if any of them are found to be in error then the fact shall be reported as a function return code and the action requested by the function shall not take place.

The parameter examination that takes place (and order of checking) when a function is invoked by a PUF depends on the function being invoked.

5.8.3 Message error handling

Error detection takes place at 2 stages during the processing of a PCI message:

1) when the message is initially examined by the NAF, to ensure that it is suitable for further processing. This checking is administrative in nature, so any errors encountered are returned in Administration Plane messages;

2) when the message is processed by the NAF, the way in which error information is passed to the PUF depends on the plane that the message belongs to and the protocol underlying that plane.

The initial examination that takes place (and order of checking) when the message is first received from a PUF is as follows:

a) NAF availability is checked;
b) Message identifier is checked;
   - unknown message, not defined by PCI;
   - unsupported message, defined by PCI but not supported by NAF.

In the case of Administration Plane messages, any error information is returned on the corresponding confirm message. In the case of Control and User Plane messages, error information is returned in the Administration Plane AErrorInd message.

The error detection that takes place (and order of checking) when the message is processed by the NAF is protocol dependant. Error information is returned by a mechanism particular to the protocol in use. These mechanisms are described in subclause 6.8.

6 Description of ISDN PCI messages

As described in subclause 5.5, "PUF NAF Interactions", the exchange of messages is realised through 2 functions, PciPutMessage and PciGetMessage, which may be called as soon as the PUF is bound to the NAF. Due to the nature of these functions, which may be used independent from each other, correlation of "got messages" to "put messages" shall be performed by the PUF. For this reason, the messages of each plane are containing identifiers allowing correlation between messages.

The following subclauses describe the messages provided by each plane of the ISDN PCI, as well as the parameters used in conjunction with each message. The actual information presentation and coding for the operations and parameters is described in subclause 6.5 "Information presentation".

6.1 Conventions

The description of the messages, their parameters and fields is independent of hardware and operating systems.

6.1.1 Address conventions

When using any address in this ETS, the following conventions shall apply:

- the called address refers to the remote address if the address is used in the direction PUF to NAF and if the address is used in the direction NAF to PUF;

- the calling address refers to the local address if the address is used in the direction PUF to NAF and if the address is used in the direction NAF to PUF.

6.1.2 Provision of information

The provision of, or requirement of, items in the message can vary. The following conventions and abbreviations are used:

M = (Mandatory): this item shall be supplied.

C = (Conditional): a condition determines if this item is supplied. The condition is explained as a comment to the item.

O = (Optional): this item may or may not be supplied. For the exchange from PUF to NAF this implies that the PUF is free to provide the item or not. For the exchange from NAF to PUF this implies that the NAF shall only supply the item if it is available.

Information coming from the NAF reflects information provided by the Network.

6.1.3 Message conventions

This subclause presents conventions used in the tables for describing the messages.

Each message belongs to a class. With each message the class is indicated. Not all classes are available for a NAF to support. A NAF provider may chose to implement only certain classes. Each plane contains its own classes.

For each plane, a PUF can only rely on the availability of messages from class 1 (basic class). The other messages belong to additional classes. If a NAF implements an additional class, all messages in this class shall be provided.

The message indicates its direction of transfer in the suffix part of its name. Messages with the suffix Req or Rsp are transferred from the PUF to the NAF. Messages with the suffix Ind and Cnf are transferred from the NAF to the PUF. Message identifiers are provided in decimal.

6.1.4 Parameter conventions

With the description of parameters the following conventions are used:

- the name of the field shall be given;
- the type of the field shall be given in decimal;
- the entity in charge to provide the content of the field - the Direction column. The following abbreviations are used:

P	charge of the PUF;
N	charge of the NAF;
B	both PUF and NAF can provide the content;

- the length of the field may be given. This indicates the number of octets this field shall occupy. The term octet does not refer to any hardware or operating system dependant implementation. It refers to the basic information unit in all systems.

6.1.4.1 Parameter ordering

No ordering between parameters in messages is needed. The ordering of the parameters is not described by the ordering in the tables.

The ordering of the fields within the parameters is defined in the subclause 6.5.

6.1.4.2 Parameter repetition

Parameters in a message can be repeated. In the following subclauses the parameters which can be repeated shall be identified by a *. The star indicates any number of repetitions that can be supplied.

If a parameter is repeated in the exchange from PUF to NAF more than the allowed number of repetitions, only the allowed number of repetitions shall be interpreted by the NAF. Any additional repetition shall be ignored.

In the exchange from NAF to PUF only the allowed number of repetitions shall be provided.

6.1.4.3 Parameter checking

No particular checking process should be performed by the NAF for parameters coming from the Network.

6.1.5 Default philosophy

For the values of parameters and fields in parameters a default philosophy applies. This means that, if appropriate, the value "default" is shown in the description. After this value the value implied by the default is given.

The default value shall only be used in the message exchange from PUF to NAF. If a parameter is not provided in a message, the value provided during the NCO creation operation takes place.

In the exchange from NAF to PUF only the real value shall be given.

6.2 Administration Plane messages

The Administration Plane messages may be divided into the following groups:

- management of network connection objects;
- management of connection security;
- error report message;
- NAF manufacturer messages.

For management of NCOs there are messages available for creating and destroying a connection object. During creation of an NCO static or dynamic attribute and address sets are linked to the created NCO. On conclusion of the creation of an NCO, an NCOID becomes available, which shall be used in subsequent User or Control Plane operations related to the created NCO. A collection of predefined attribute sets is presented in Annex H.

For security to be used on connections there are messages available to request security be used or stopped on a connection. These messages are optional and may not be provided by all NAFs. Their availability shall be indicated in the properties definition provided by the NAF.

For the reporting of error information a single message is provided by the NAF. This is used to report general error conditions.

All request messages of the Administration Plane may contain a request identifier (RequestID). This identifier, if assigned by a PUF on a request message, shall be returned by the NAF on the related confirm message.

Table 8 gives an overview of Administration Plane messages. The messages themselves are described in detail in the following subclauses.

Table 8: Administration Plane messages

Mess. Identif	**Class**	**Message Name**	**Purpose of Message**
101	1	ACreateNCOReq	Request to create a network connection object.
102	1	ACreateNCOCnf	Confirmation of the "CreateNCO" operation.
103	1	ADestroyNCOReq	Request to destroy a network connection object.
104	1	ADestroyNCOCnf	Confirmation of "DestroyNCO" operation.
105	1	AGetNCOInfoReq	Request information concerning a specific NCO.
106	1	AGetNCOInfoCnf	Confirmation reporting information for the relevant NCO.
108	1	AErrorInd	Indicate that a non-protocol related error has occurred.
109	2	ASecurityReq	Request to engage/stop security algorithm.
110	2	ASecurityCnf	Confirmation to engage/stop security algorithm.
111	3	AManufacturerReq	Request for a specific manufacturer functionality.
112	3	AManufacturerInd	Provide the PUF with information linked to the requested functionality.

6.2.1 ACreateNCOReq

Class: 1 (Basic Class).

Description: Request message for creating a network connection object (NCO).

The PUF shall provide an NCOType which identifies the type of NCO which is to be created. Depending on this type, there are more parameters needed (conditional parameters). For details refer to tables 9 and 10.

The PUF can supply a unique request identifier (RequestID) which can be used to identify the corresponding confirmation message of this operation.

Parameters:

Name	**Required**	**Comment**
RequestID	O	Request identifier, generated by the PUF.
NCOType	M	Specification of NCO type.
CDirection	C	Determines how NCO shall be used, for the Control Plane.
UDirection	C	Determines how NCO shall be used, for the User Plane.
CAttributeName	C	Name of static control plane attribute.
CAttribute parameters	C	Control Plane attribute parameters. Exclusive with CattributeName; see table 20 for more details.
UAttributeName	C	Name of static User Plane attribute.
UAttribute parameters	C	User Plane attribute parameters. Exclusive with UAttributeName; see tables 21 and 22 for more details.
CAddress parameters	O	Control Plane address; see table 24 for more details.
UAddress parameters	O	User Plane address; see table 25 for more details.
GroupID	C	Required if NCO is to be grouped.
SelectorID	O	Helps the NAF to select the right NCO.

Remark: See also subclause 6.7 on usage of the NCO.

Related: ACreateNCOCnf.

6.2.2 NCOType and conditional parameter specification

Currently there are 5 types of NCOs defined. These types are shown in table 9.

Table 9: NCOTypes

NCOType	**NCO allows PUF ...**
C	... signalling access only.
C/U1	... signalling and transparent user access via TMA.
C/U3	... signalling and network layer user access via NMA.
U3	... network user access via NMA with NAF signalling co-ordination (NAF co-ordination functionality).
U3G	...network user access via NMA to additional virtual circuits. This NCO shall be grouped to an already created U3 or C/U3 type NCO.

Table 10 shows which conditional parameters shall be specified in the ACreateNCOReq message in relation to the selected NCOType.

Table 10: Specification of conditional ACreateNCOReq message parameters

NCOType	SigAttribute Type	UsrAttribute Type	SigAddress Type	UsrAddress Type	GroupID
C	C Attribute		C Address		
C/U1	C Attribute	U1 Attribute	C Address		
C/U3	C Attribute	U3 Attribute	C Address	U3 Address	
U3	C Attribute	U3 Attribute	C Address	U3 Address	
U3G		U3 Attribute		U3 Address	Reference to NCO See NOTE
NOTE:	If an NCO is to be grouped - which can only be done in case of the U3G NCOType - a reference by GroupID to an already created NCO of type U3 or C/U3 shall be provided. Therefore, the creation of such an NCO type shall have been carried out successfully in order to have the GroupID available.				

6.2.3 ACreateNCOCnf

Class: 1 (Basic Class).

Description: Confirmation message of the CreateNCO operation requested previously. The confirmation message can be correlated to the correct ACreateNCOReq message by use of the returned RequestID.

The confirmation message may contain the NCO identifier (NCOID) which shall be used on further requests through the User or Control Plane related to the created NCO as well as the GroupID which shall be used for subsequent ACreateNCOReq messages, if grouping to the created NCO is intended.

Parameters:

Name	**Provided**	**Comment**
RequestID	C	Provided if supplied on request message.
CompletionStatus	M	Completion status of the CreateNCO operation of the NAF.
NCOID	C	NCO identifier if CompletionStatus Success else absent.
GroupID	C	Group identifier, provided if NCO created was of type C/U3 or U3 and if CompletionStatus Success.

Related: ACreateNCOReq.

6.2.4 ADestroyNCOReq

Class: 1 (Basic Class).

Description: Destroys an existing NCO created by the same PUF. The PUF can supply a request identifier (RequestID) which can be used to identify the corresponding confirmation message of this operation.

Parameters:

Name	**Required**	**Comment**
RequestID	O	Request identifier, generated by the PUF.
NCOID	M	Identifier of NCO to be destroyed.

NOTE: NCO may not be destroyed if it is in use for an established connection or a connection that is attempting to be established. When a non-grouped NCO is destroyed, any NCOs grouped to it become unusable except when the grouped NCO relates to an established connection or a connection that is attempting to be established. In this case, the NCO remains usable until the related connection is removed. An unusable NCO can only be destroyed using the ADestroyNCOReq message.

Related: ADestroyNCOCnf.

6.2.5 ADestroyNCOCnf

Class: 1 (Basic Class).

Description: Confirmation message of the DestroyNCO operation previously requested. The confirmation message can be correlated to the correct ADestroyNCOReq message by use of the RequestID.

Parameters:

Name	**Provided**	**Comment**
RequestID	C	Provided if supplied on request message.
NCOID	M	Identify the NCO on which the Destroy operation was requested.
CompletionStatus	M	Completion status of the DestroyNCO operation of the NAF.

Related: ADestroyNCOReq.

6.2.6 AErrorInd

Class: 1 (Basic Class).

Description: This message is related to administrative (i.e. non-protocol related) checking of messages.

Parameters:

Name	**Provided**	**Comment**
RequestID	C	Provided if supplied on request message.
CompletionStatus	M	Value indicating the error that has occurred.

Related: None.

6.2.7 AGetNCOInfoReq

Class: 1 (Basic Class).

Description: Request message for getting information about an NCO. Each NCO is characterized by some attributes (see Administration Attribute set parameters, subclause 6.6.73) which are accessible from the PUF thanks to this request and its confirmation.

Parameters:

Name	**Required**	**Comment**
NCOID	M	Identifier of NCO requested on.

Related: AGetNCOInfoCnf.

6.2.8 AGetNCOInfoCnf

Class: 1 (Basic Class).

Description: Confirmation message sent by the NAF to the PUF for answering an AGetNCOInfoReq. It contains the information (see Administration Attribute set parameters, subclause 6.6.73) relevant for the requested NCO.

Parameters:

Name	**Provided**	**Comment**
NCOID	M	Identifier of NCO requested on.
CompletionStatus	M	Completion status of the GetNCOInfo operation of the NAF.
AAttribute	C	Administration plane attribute set parameters if CompletionStatus Success else absent.

Related: AGetNCOInfoReq.

6.2.9 ASecurityReq

Class: 2 (Additional class).

Description: This message allows the PUF to engage a security algorithm provided by the NAF. The PUF shall provide the NCOID the connection it wants to have the security algorithm applied to. The PUF can indicate any connection for security to be applied to. The PUF shall be informed by the NAF with a ASecurityCnf message if it is possible to use security on the indicated connection.

The ASecurityReq message does not state how the connection is secured, or which type of information inside the connection shall be affected by the security algorithm. It is up to the security algorithm to handle the securing of the connection.

The Algorithm parameter indicates to the NAF which security algorithm shall be used to secure the connection. The security algorithm is identified by its name. The names of the available algorithms can be obtained using the Property information. By using the name "nosecurity" for this parameter, the PUF can indicate that security is no longer needed for the connection.

The optional Key parameter is used by the PUF to give relevant information for the security algorithm to the NAF. The parameter is optional because the security algorithm may or may not need specific information to be activated. The kind of information to be used for the Key parameter is dependant of the security algorithm activated.

Parameters:

Name	**Required**	**Comment**
RequestID	O	Request identifier, generated by the PUF.
NCOID	M	Identify the connection for which security shall be activated.
Algorithm	M	The name of the security algorithm to use.
Key	O	Key to use for the security Algorithm.

Related: ASecurityCnf.

6.2.10 ASecurityCnf

Class: 2 (Additional class).

Description: Confirmation message sent to the PUF by the NAF upon completion of the ASecurityReq. The RequestID correlates this confirmation message to the corresponding ASecurityReq.

The CompletionStatus Success indicates that the required security algorithm has been activated or stopped for the requested connection, otherwise the reason for non-activation of the security algorithm is returned. The reason is algorithm specific.

Parameters:

Name	**Provided**	**Comment**
RequestID	C	Provided if supplied on request message.
CompletionStatus	M	Completion status of the ASecurity operation of the NAF.

Related: ASecurityReq.

6.2.11 AManufacturerReq

Class: 3 (Additional class).

Description: This message allows the PUF to request the NAF to provide a private manufacturer functionality.

This is the way to handle private functionality not provided by the ISDN PCI.

Parameters:

Name	**Required**	**Comment**
RequestID	M	Request Identifier.
ManufacturerCode	M	Identifies the manufacturer code (provided by the manufacturer).

Remark: Information about the functionality shall be mandatory. It is not provided as a parameter of the message but shall be contained in the data buffer.

Related: None.

6.2.12 AManufacturerInd

Class: 3 (Additional class).

Description: This message gives to a PUF specific information dealing with the requested functionality. The NAF is only allowed to issue manufacturer indications, when the PUF had earlier issued at least one manufacturer private request.

Parameters:

Name	Provided	Comment
RequestID	M	Request Identifier.
ManufacturerCode	M	Identifies the manufacturer code (provided by the manufacturer).
CompletionStatus	O	Identifies the result which is manufacturer specific.

Remark: If information is provided, it shall not be done as a parameter of the message but in the data buffer.

Related: AManufacturerReq.

6.3 Control Plane messages

The Control Plane messages may be divided into six classes:

1) connection establishment and connection breakdown;
2) overlap sending specific messages;
3) user-to-user information transfer;
4) adjournment of calls;
5) facility invocation;
6) external equipment.

As described in subclause 6.1.3, not all these classes may be accessible through the ISDN PCI. A NAF may choose to implement only a number of categories from the above list. The error mechanism to indicate to the PUF that a message is not available is described in the subclause 5.8.

A PUF can only rely on the availability of the class 1 messages. The availability of other classes of message is NAF dependent.

Table 11 gives an overview of Control Plane messages, the class they belong to and their message identifier.

Table 11: Control Plane messages

Mess. Identif.	Class	Message Name	Purpose of Message
201	1	CAlertReq	State the compatibility with the incoming call.
202	1	CAlertInd	The called terminal states that it may handle a call.
203	1	CConnectReq	Initiate an outgoing call.
204	1	CConnectInd	Present an incoming call.
205	1	CConnectRsp	Accept an incoming call.
206	1	CConnectCnf	Indicate acceptance of an outgoing call by the called terminal.
207	1	CDisconnectReq	Remove a connection or refuse an incoming call.
208	1	CDisconnectInd	Indicate the connection has been removed or the outgoing call has been refused.
209	1	CDisconnectRsp	Confirm the end of a connection.
210	1	CDisconnectCnf	Indicate the other terminal has ended the connection.
212	1	CProgressInd	Indicate a B channel is connected.
214	1	CStatusInd	Indicate a protocol error.
216	2	CSetupAckInd	Indicate more information is required to proceed the outgoing call.
217	2	CConnectInfoReq	Send more information to process the call.
218	2	CProceedingInd	Indicate no more establishment information shall be accepted for this call.
219	3	CUserInformationReq	Send user-to-user information.
220	3	CUserInformationInd	Present received user-to-user information.

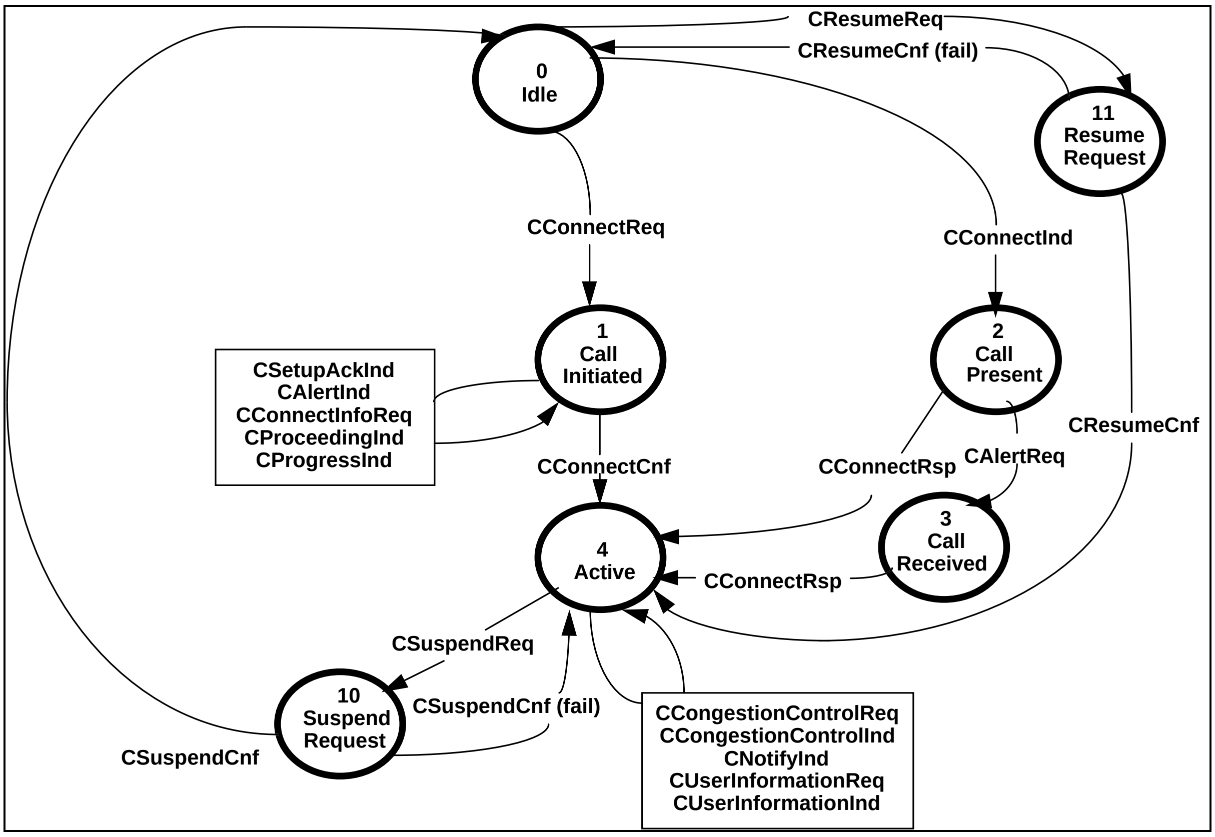
(continued)

Table 11: Control Plane messages (concluded)

Mess. Identif.	Class	Message Name	Purpose of Message
221	3	CCongestionControlReq	Apply flow control operations to user-to-user information exchange.
222	3	CCongestionControlInd	Indicate flow control operation to be applied to user-to-user information exchange.
223	4	CSuspendReq	Suspend a connection.
224	4	CSuspendCnf	Response to the demand for suspending a connection.
225	4	CResumeReq	Resume a suspended connection.
226	4	CResumeCnf	Response to the demand for resuming a connection.
228	4	CNotifyInd	Indicate a new state for a connection.
229	5	CFacilityReq	Request a facility from the network.
230	5	CFacilityInd	Indicate a facility coming from the network.
232	6	CExtEquipAvailabilityInd	Indicate that the external equipment is or not connected to the NAF.
234	6	CExtEquipBlockDiallingInd	Indicate that the call is completely initiated by the external equipment (block dialling).
236	6	CExtEquipKeyPressedInd	Provide to the PUF the code of a pressed key.
238	6	CExtEquipOffHookInd	Indicate that the handset is off-hooked.
240	6	CEXtEquipOnHookInd	Indicate that the handset is on-hooked.

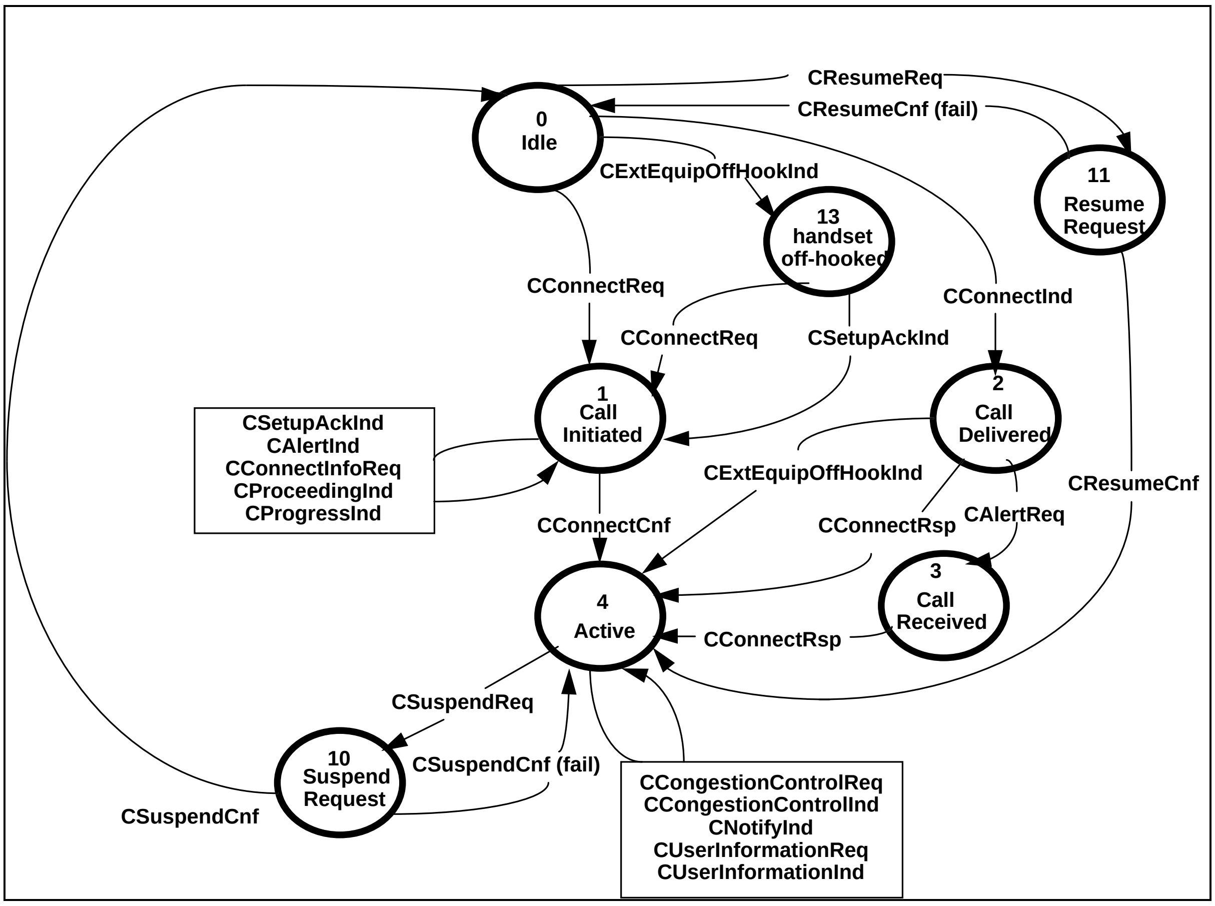
6.3.1 Sequencing of Control Plane messages

Figures 9, 10, 11 and 12 present the state diagrams affecting the state of a PUF connection.



NOTE: CExtEquipavailabalityInd can be used in all states. It causes a transition to state 0 if the external equipment is unavailable.

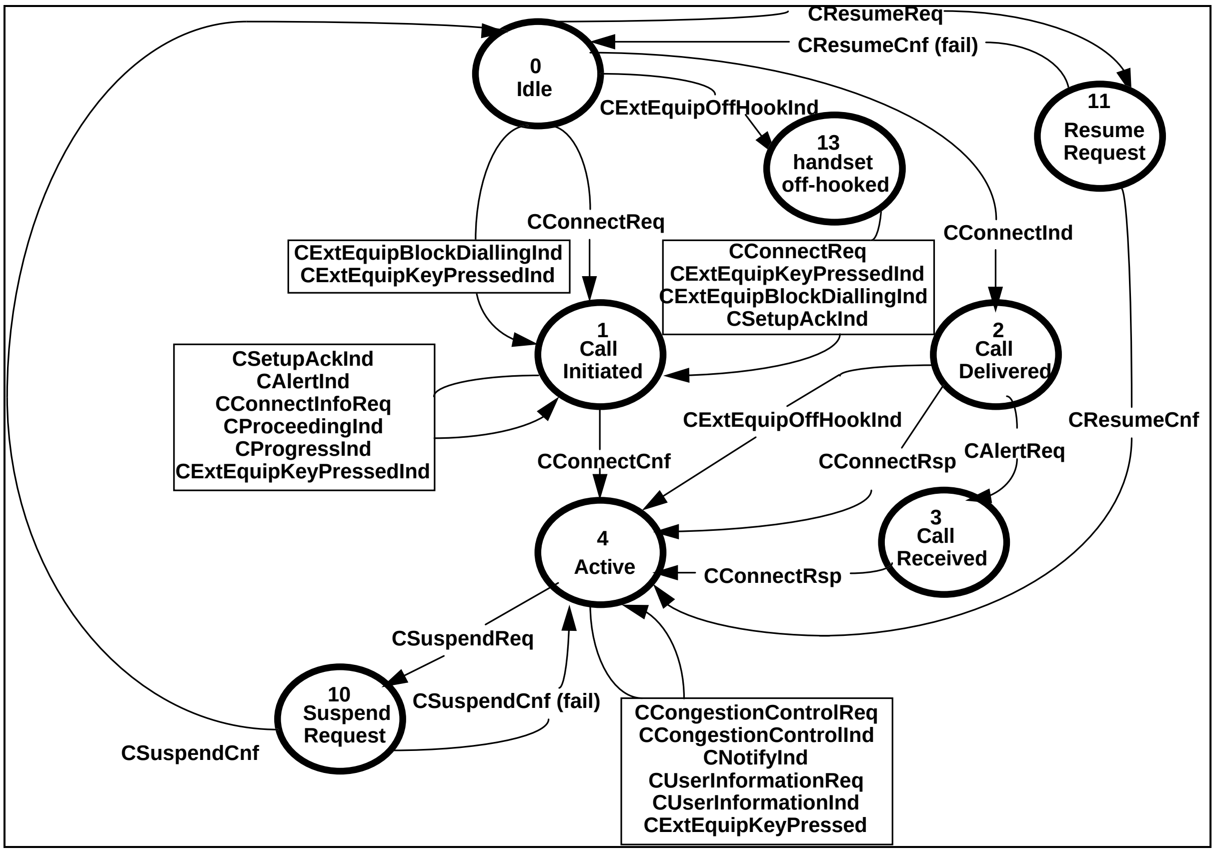
Figure 9: State diagram of a Control Plane no external equipment or external equipment type 1



NOTE 1: CExtEquipOnHookInd can be used in all states except 0. It causes a transition to state 0.

NOTE 2: CExtEquipavailabalityInd can be used in all states. It causes a transition to state 0 if the external equipment is unavailable.

Figure 10: State diagram of a Control Plane external equipment type 2 or 3

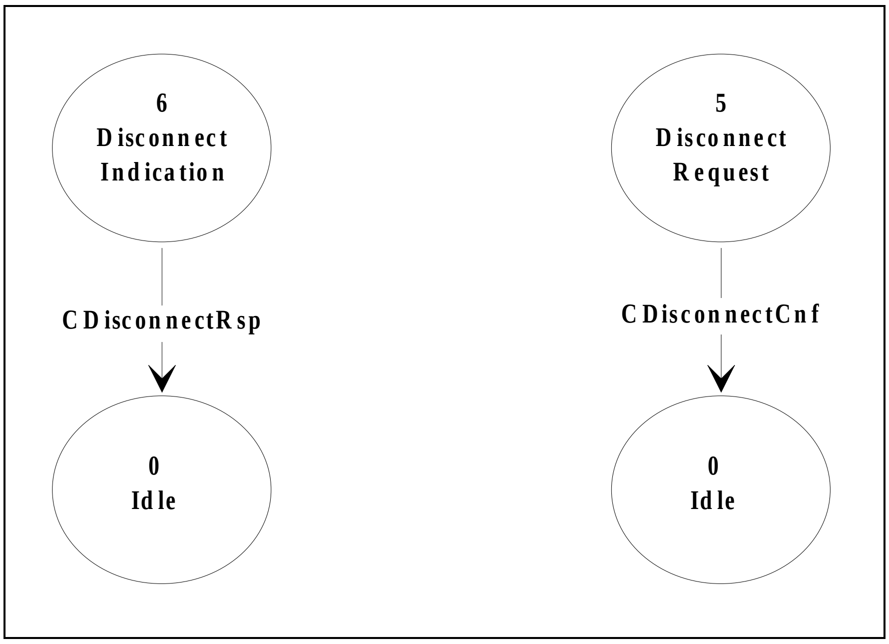


NOTE 1: CExtEquipOnHookInd can be used in all states except 0. It causes a transition to state 0.

NOTE 2: CExtEquipavailabalityInd can be used in all states. It causes a transition to state 0 if the external equipment is unavailable.

NOTE 3: In figures 9, 10 and 11 if the PUF reaches the Idle state (state 0) by receiving a CSuspendCnf the connection is suspended and may be reused. The NCO however still describes the interaction between PUF and NAF for this connection and cannot be reused. The PUF shall use the NCO again when the connection is resumed or disconnected.

Figure 11: State diagram of a Control Plane external equipment type 4 or 5



Remarks: CDisconnectInd can be used in all states except 0, 5 and 6. It causes a transition to state 6.

CDisconnectReq can be used in all states except 0, 5 and 6. It causes a transition to state 5.

Related network messages and complementary intermediate states can be found in Annex E, Clause E.1.

Any CStatusInd message causes a transition to state 0.

NOTE 1: Figures 9, 10, 11 and 12 do not provide any information on the user-to-user information transfer. These messages, depending on the user-to-user service level, do not affect the state of a call from the PUF point of view.

NOTE 2: In order to simplify the interface, a filtering functionality might be added; using this functionality, the PUF may select the subset of messages handled. A detailed description of this functionality is for further study.

Figure 12: State diagram of a Control Plane connection: disconnection

6.3.2 CAlertReq

Class: 1 (Basic class).

Description: This message allows a PUF to indicate its compatibility with an incoming call.

Parameters:

Name	Required	Comment
NCOID	M	Identifies the call.
Facility	O *	Supplementary services operation or information.
UserToUserInfo	O	Information to be exchanged between ISDN users.

Remarks: The availability of UserToUserInfo depends on the user-to-user service level. See subclause 6.3. for details on user-to-user information.

Related: CConnectReq, CConnectInd, CConnectRsp, CConnectCnf, CAlertInd.

6.3.3 CAlertInd

Class: 1 (Basic class).

Description: The PUF receives this message when the called terminal has indicated its compatibility.

Parameters:

Name	Provided	Comment
NCOID	M	Identifies the call.
ChannelIdentification	O	Identification of the channel used.
Facility	O *	Supplementary services operation or information.
Display	O	Information provided by the Network to be displayed.
UserToUserInfo	O	Information to be exchanged between ISDN users.

Remarks: The availability of UserToUserInfo depends on the user-to-user service level. See subclause 6.3. for details on user-to-user information.

Related: CConnectReq, CConnectInd, CConnectRsp, CConnectCnf, CAlertInd.

6.3.4 CConnectReq

Class: 1 (Basic class).

Description: This message is sent by the PUF to initiate an outgoing call. The call shall be initiated to the remote address. This address may be specified in the message, or have been specified in the address parameters used either to create the referenced NCO.

The PUF shall specify the BearerCap parameter to indicate which type of bearer channel is needed. This parameter shall be specified in the message, or have been specified in the attribute parameters used to create the referenced NCO.

The PUF may specify the LLC and HLC parameters, to indicate what type of lower layer and higher layer protocols shall be used for this call.

Parameters:

Name	**Required**	**Comment**
NCOID	M	Identifies the call. This information is provided by the PUF.
CallingNumber	O	Local address (NOTE 1).
CallingSubaddress	O	Local subaddress (NOTE 1).
CalledNumber	O	Remote address (NOTES 1 and 3).
CalledSubaddress	O	Remote subaddress (NOTES 1 and 3).
ChannelIdentification	O	Used by PUF to indicate type of requested Channel. See ChannelIdentification parameter in subclause 6.6.18 for details of supported values. If not provided default is any channel (NOTE 2).
BearerCap	O	Transmission capability required from channel (NOTE 2).
LLC	O	Lower Layer Compatibility information element (NOTE 2).
HLC	O	High Layer Compatibility information element (NOTE 2).
Keypad	O	Keypad facility information element.
Facility	O *	Supplementary services operation or information.
UserToUserInfo	O	Information to be exchanged between ISDN users.

NOTE 1: Can be supplied during the creation of NCO. If specified on both message and within the NCO, then parameter specified on message is used and NCO parameter is ignored.

NOTE 2: Can be supplied during the NCO creation. If specified on both message and within the NCO, then parameter specified on message is used and NCO parameter is ignored.

NOTE 3: Either a CalledNumber or a CalledSubaddress parameter - in the message or in during the NCO creation - shall be supplied.

Remarks: The availability of UserToUserInfo depends on the user-to-user service level. See subclause 6.3. for details on user-to-user information.

Related: CConnectCnf, CAlertReq, CAlertInd, CConnectInfoReq.

6.3.5 CConnectInd

Class: 1 (Basic class).

Description: This message offers an incoming call to all appropriate PUFs (see subclause 6.7.1). At this point the call is in the establishment phase, no connection has been established yet.

The number of the calling user may be available to the PUF. If so, it shall be represented in the parameters CallingNumber and CallingSubaddress.

The PUF may receive the parameters BearerCap, LLC, HLC which shall indicate:

- what type of bearer channel shall be used;
- what type of lower layer protocols shall be used for this call;
- what type of higher layer protocols shall be used for this call.

Parameters:

Name	**Provided**	**Comment**
NCOID	M	Identifies the call. This information element is provided by the NAF.
ChannelIdentification	O	Identification of the channel used.
CallingNumber	O *	Remote address.
CallingSubaddress	O	Remote subaddress.
CalledNumber	O	Local address.
CalledSubaddress	O	Local subaddress.
BearerCap	O	Network physical resource provided.
LLC	O	Lower Layer Compatibility information element.
HLC	O	High Layer Compatibility information element.
DateTime	O	Date and Time.
Facility	O *	Supplementary services operation or information.
Display	O	Information provided by the Network to be displayed.
UserToUserInfo	O	Information to be exchanged between ISDN users.

Remark: When a PUF receives the No Channel Available information, it can clear or suspend a call to provide a free channel if it wishes to establish a connection.

The availability of UserToUserInfo depends on the user-to-user service level. See subclause 6.3. for details on user-to-user information.

Related: CConnectReq, CConnectRsp, CConnectCnf, CAlertReq, CAlertInd.

6.3.6 CConnectRsp

Class: 1 (Basic class).

Description: This message allows a PUF to accept an incoming call. After sending this message, the channel is considered to be established.

The PUF can supply a new value for the LLC, if it is negotiating LLC values.

Parameters:

Name	**Required**	**Comment**
NCOID	M	Identifies the call.
ChannelIdentification	O	Used by PUF to indicate type of requested Channel. See ChannelIdentification parameter in subclause 6.6.17 for details of supported values. A value can be provided if the B-channel chosen by the PUF is not the same as those the NAF presents.
LLC	O	Negotiated value.
Facility	O *	Supplementary services operation or information.
UserToUserInfo	O	Information to be exchanged between ISDN users.

Remarks: The availability of UserToUserInfo depends on the user-to-user service level. See subclause 6.3. for details on user-to-user information.

Related: CConnectReq, CConnectInd, CConnectCnf, CAlertReq, CAlertInd.

6.3.7 CConnectCnf

Class: 1 (Basic class).

Description: This message is the response from the called party, indicating it accepts the call. When the PUF receives this message, a channel is considered to be established.

If values for LLC are being negotiated, a new value for the LLC parameter may be supplied in this message.

Parameters:

Name	**Provided**	**Comment**
NCOID	M	Identifies the call.
ChannelIdentification	O	Identification of the channel used.
LLC	O	Negotiated value.
DateTime	O	Date and Time.
Facility	O *	Supplementary services operation or information.
Display	O	Information provided by the Network to be displayed.
UserToUserInfo	O	Information to be exchanged between ISDN users.
ConnectedNumber	O	Part of the remote address.
ConnectedSubaddress	O	Part of the remote address.

Remarks: The availability of UserToUserInfo depends on the user-to-user service level. See subclause 6.3. for details on user-to-user information.

Related: CConnectReq, CConnectInd, CConnectRsp, CAlertReq, CAlertInd.

6.3.8 CDisconnectReq

Class: 1 (Basic class).

Description: This message allows the PUF to initiate the disconnection of a connection or refuse a call.

This message shall be acknowledged by a CDisconnectCnf.

The PUF may indicate the reason to disconnect a connection or refuse a call by supplying the CauseToNAF parameter.

Parameters:

Name	**Required**	**Comment**
NCOID	M	Identifies the call.
CauseToNAF	O *	PUF reason to disconnect the call. If not provided by the PUF, the #16 'Normal Call Clearing' cause shall be provided by the NAF.
Facility	O *	Supplementary services operation or information.
UserToUserInfo	O	Information to be exchanged between ISDN users.

Remarks: The availability of UserToUserInfo depends on the user-to-user service level. See subclause 6.3. for details on user-to-user information.

Related: CDisconnectInd, CDisconnectRsp, CDisconnectCnf.

6.3.9 CDisconnectInd

Class: 1 (Basic class).

Description: This message informs the PUF that the remote user has initiated the disconnection of the connection or has refused the call.

The PUF shall acknowledge this message with a CDisconnectRsp.

The CauseToPUF parameter shall indicate the reason for disconnecting or refusing.

Parameters:

Name	**Provided**	**Comment**
NCOID	M	Identifies the call.
CauseToPUF	M	Reason why the call is being disconnected. If not provided by the Network the NAF shall introduce the #16 'Normal Call Clearing' cause. See also the remark.
Facility	O *	Supplementary services operation or information.
Display	O	Information provided by the Network to be displayed.
UserToUserInfo	O	Information to be exchanged between ISDN users.

Remarks: The network shall only transfer one cause to the NAF, so the PUF shall only receive one cause.

The availability of UserToUserInfo depends on the user-to-user service level. See subclause 6.3. for details on user-to-user information.

Related: CDisconnectReq, CDisconnectRsp, CDisconnectCnf.

6.3.10 CDisconnectRsp

Class: 1 (Basic class).

Description: With this message, the PUF acknowledges that a connection has ended or a call has been refused. From the point of view of the PUF the channel is now cleared, and the NCOID may be re-used by the NAF.

This message is sent by the PUF to acknowledge the CDisconnectInd.

Parameters:

Name	**Required**	**Comment**
NCOID	M	Identifies the call.
Facility	O *	Supplementary services operation or information.

Related: CDisconnectReq, CDisconnectInd, CDisconnectCnf.

6.3.11 CDisconnectCnf

Class: 1 (Basic class).

Description: With this message, the PUF is informed that the connection has ended or a call been refused, and the channel has been cleared down. The NCOID may now be reused by the NAF.

This message is the acknowledgement by the remote user or by the network of the CDisconnectReq.

Parameters:

Name	**Provided**	**Comment**
NCOID	M	Identifies the call.
CauseToPUF	O	Reason why a supplementary service request has been rejected by the Network.
Facility	O *	Supplementary services operation or information.
Display	O	Information provided by the Network to be displayed.

Related: CDisconnectReq, CDisconnectInd, CDisconnectRsp.

6.3.12 CProgressInd

Class: 1 (Basic class).

Description: The PUF receives this message when information is available in the B-channel. The channel shall be connected.

Parameters:

Name	Provided	Comment
NCOID	M	Identifies the call.
ChannelIdentification	O	Identification of the channel used.
Display	O	Information provided by the network to be displayed.
ProgressIndicator	M	Details concerning call progress.

Related: CConnectReq, CConnectInd, CConnectRsp, CConnectCnf, CAlertInd.

6.3.13 CStatusInd

Class: 1 (Basic class).

Description: With this message, the PUF shall be informed that a signalling protocol error, as defined in subclause 6.8.8, has occurred.

Parameters:

Name	Provided	Comment
NCOID	M	Identifies the call.
CauseToPUF	M	Identifies the protocol error that has occurred.

Related: None.

6.3.14 CSetupAckInd

Class: 2 (Additional class).

Description: The PUF receives this message when more establishment information is needed to perform the call, in the overlap sending case.

Parameters:

Name	Provided	Comment
NCOID	M	Identifies the call.
ChannelIdentification	O	Identification of the channel used.
Display	O	Information provided by the network to be displayed.
ProgressIndicator	O	Details concerning call progress.

Related: CConnectInfoReq, CConnectReq.

6.3.15 CConnectInfoReq

Class: 2 (Additional Class).

Description: This message allows a PUF to use the overlap sending technique for connection establishment. Overlap sending means that the PUF supplies the address information in more than one step: a CConnectReq message with incomplete address information may be followed by several CConnectInfoReq messages until the address is complete. This mechanism is the natural reflection of dialling on a keypad.

Parameters:

Name	**Required**	**Comment**
NCOID	M	Identifies the call.
CalledNumber	M	Part of the remote address (NOTE).
NOTE: With each CConnectInfoReq message the NAF accumulates the address information. The PUF does not indicate that the address information is complete; this can implicitly be concluded from the receipt of a CProceedingInd message. A Subaddress can only be specified in the CConnectReq message. This is due to the restrictions imposed by the D-channel protocol (SETUP network message).		

Remarks: The PUF shall have sent a CConnectReq message with the first part of the called number information prior to this message.

Related: CConnectReq, CProceedingInd, CSetupAckInd.

6.3.16 CProceedingInd

Class: 2 (Additional class).

Description: The PUF receives this message when no more establishment information can be accepted, in the overlap sending case. As the Network may not provide this message, the PUF can not rely on its reception.

Parameters:

Name	**Provided**	**Comment**
NCOID	M	Identifies the call.
ChannelIdentification	O	Identification of the channel used.
Display	O	Information provided by the network to be displayed.
ProgressIndicator	O	Details concerning call progress.

Related: CConnectReq, CConnectInfoReq, CSetupAckInd.

6.3.17 CUserInformationReq

Class: 3 (Additional class).

Description: This message allows a PUF to request the sending of user-to-user information on an established connection.

The call state that allows the PUF to send user-to-user Information is dependent on the user-to-user service level provided by the Network or the subscription. See subclause 6.3. for details on user-to-user information.

Parameters:

Name	**Required**	**Comment**
NCOID	M	Identifies the call.
MoreData	O	Indicates to the peer entity that another user-to-user message shall follow.
UserToUserInfo	M	Information to be exchanged between ISDN users.

Remarks: This message is available only if a user-to-user service level 2 and above has been subscribed to. See subclause 6.3. for details on user-to-user information.

Related: CUserInformationInd, CCongestionControlReq, CCongestionControlInd.

6.3.18 CUserInformationInd

Class: 3 (Additional class).

Description: This message allows a NAF to present to the PUF user-to-user information received on an established connection.

The call state that allows the reception of user-to-user Information is dependent on the user-to-user service level provided by the Network or the subscription. See subclause 6.3. for details on user-to-user information.

Parameters:

Name	**Provided**	**Comment**
NCOID	M	Identifies the call.
MoreData	O	If present, the peer entity indicates that another User to User message shall follow.
UserToUserInfo	M	Information exchanged between ISDN users.

Remarks: This message is available only if a user-to-user service level 2 and above has been subscribed to. See subclause 6.3. for details on user-to-user information.

Related: CUserInformationReq, CCongestionControlReq, CCongestionControlInd.

6.3.19 CCongestionControlReq

Class: 3 (Additional class).

Description: This message allows a PUF to apply flow control operations on the user-to-user information provided via the CUserInformationInd message.

The flow control operation is only defined to operate on the local side of the connection. The flow control operates using the ready/not ready mechanism. The initial condition for user-to-user information exchange shall be ready. To set the condition for flow control the PUF shall set the parameter CongestionLevel to the appropriate value.

Parameters:

Name	**Required**	**Comment**
NCOID	M	Identifies the call.
CongestionLevel	M	Flow control value.
CauseToNAF	O	Include if information lost.
NOTE: This message in available only if a user-to-user service level 2 and above has been subscribed to. See subclause 6.3. for details on user-to-user information.		

Remarks: For the flow control provided by this message, ready is assumed as the initial status. The flow control for each direction shall be operated independently.

Related: CUserInformationReq, CUserInformationInd, CCongestionControlInd.

6.3.20 CCongestionControlInd

Class: 3 (Additional class).

Description: This message allows a NAF to indicate to a PUF that a flow control operations has been applied to the user-to-user information provided via the CUserInformationReq message.

The flow control operation is only defined to operate on the local side of the connection. The flow control operates using the ready/not ready mechanism. The initial condition for user-to-user information exchange shall be ready. The parameter CongestionLevel shall give the new value for the flow control on the user-to-user information exchange to the PUF.

Parameters:

Name	**Provided**	**Comment**
NCOID	M	Identifies the call.
CongestionLevel	M	Flow control value.
CauseToPUF	O (NOTE)	Include if information lost.
Display	O	Information provided by the Network to be displayed.
NOTE: The network shall only transfer one cause to the NAF, so the PUF shall only receive one cause.		

Remarks: This message is available only if a user-to-user service level 2 and above has been subscribed to. See subclause 6.3. for details on user-to-user information.

For the flow control provided by this message, ready is assumed as being the initial status.

The flow control for each direction shall be operated independently.

Related: CUserInformationReq, CUserInformationInd, CCongestionControlReq.

6.3.21 CSuspendReq

Class: 4 (Additional class).

Description: This message allows a PUF to suspend, but not to disconnect, a connection.

After sending this message, the PUF shall be informed if the connection is suspended.

Parameters:

Name	Required	Comment
NCOID	M	Identifies the call.

Remarks: Using this message in conjunction with running a protocol on the connection is the responsibility of the PUF.

When suspending a connection, it is not guaranteed that the connection can subsequently be resumed.

Related: CSuspendCnf, CResumeReq, CResumeCnf, CNotifyInd.

6.3.22 CSuspendCnf

Class: 4 (Additional class).

Description: This message is the answer to a CSuspendReq message. The NAF provides the PUF with the result of its suspend request.

The parameter Response shall indicate if the connection is suspended.

If the PUF receives a CSuspendCnf the connection is suspended and may be re-used. The NCO, however, still describes the interaction between PUF and NAF for this connection and cannot be reused. The PUF shall have to use the NCO again when the connection is resumed or disconnected.

Parameters:

Name	Provided	Comment
NCOID	M	Identifies the call.
CompletionStatus	M	Indicates the state of the suspension: - success: if the suspension is accepted; - operationfailed: if the suspension is refused.
CauseToPUF	C (NOTE)	Mandatory if case of suspension refused, it indicates the reason why the suspension was refused. Absent in case of success.
Display	O	Information provided by the Network to be displayed.
NOTE: The network shall only transfer one cause to the NAF, so the PUF shall only receive one cause.		

Related: CSuspendReq, CResumeReq, CResumeCnf, CNotifyInd.

6.3.23 CResumeReq

Class: 4 (Additional class).

Description: This message allows a PUF to resume, i.e. a suspended connection is reconnected.

After sending this message, the PUF shall be informed if the suspended connection is reconnected.

Parameters:

Name	Required	Comment
NCOID	M	Identifies the call.

Related: CSuspendReq, CSuspendCnf, CResumeCnf, CNotifyInd.

6.3.24 CResumeCnf

Class: 4 (Additional class).

Description: This message is the answer to an CResumeReq message. The NAF provides the PUF with the result of its resume request.

The response parameter shall indicate if the connection is resumed.

Parameters:

Name	Provided	Comment
NCOID	M	Identifies the call.
CompletionStatus	M	Indicates the state of the resume operation: - Success: if the operation succeed; - OperationFailed: if the operation failed.
CauseToPUF	C (NOTE)	Mandatory if case of operation failure, it indicates the reason why the operation was refused. Absent in case of success.
Display	O	Information provided by the Network to be displayed.
NOTE: The network shall only transfer one cause to the NAF, so the PUF shall only receive one cause.		

Remarks: The result for resuming a connection may be negative (OperationFailed) if the NAF or the network does not have resources available, i.e. channels, to reconnect the connection.

Related: CSuspendReq, CSuspendCnf, CResumeReq, CNotifyInd.

6.3.25 CNotifyInd

Class: 4 (Additional class).

Description: This message is provided by the NAF to indicate to the PUF a new state for the connection.

As example, this message may be issued if the remote user suspends or resumes a connection.

Parameters:

Name	**Provided**	**Comment**
NCOID	M	Identifies the call.
NotificationIndicator	M	New state.
Display	O	Information provided by the Network to be displayed.

Related: CSuspendReq, CSuspendCnf, CResumeReq, CResumeCnf.

6.3.26 CFacilityReq

Class: 5 (Additional class).

Description: This message is intended to provide to the PUF the ability to request a facility from the Network. This facility may or may not be related to an established connection.

For details on the use of facility messages and parameters and the coding for the facility parameter refer to subclauses 6.3. and 6.6.30.

Parameters:

Name	**Required**	**Comment**
NCOID	O	Provided by the PUF if the facility is related to an established connection.
Facility	M	Supplementary services operation or information.
NOTE: If the PUF supplies transparent coding of the facility information element, all information following this transparent coding shall be handed back transparently.		

Related: CFacilityInd.

6.3.27 CFacilityInd

Class: 5 (Additional class).

Description: This message provides to the PUF the facility coming from to the Network. This facility may or may not be related to an established connection.

For details on the use of facility messages and parameters and the coding for the facility parameter refer to subclauses 6.3. and 6.6.30.

Parameters:

Name	**Provided**	**Comment**
NCOID	O	Provided by the NAF if the facility is related to an established connection.
Facility	M (NOTE)	Supplementary services operation or information.
Display	O	Information provided by the Network to be displayed.
NOTE: If the PUF has supplied transparent coding of the facility information element, all information following this transparent coding shall be handed back transparently.		

Related: CFacilityReq.

6.3.28 CExtEquipAvailabalityInd

Class: 6 (Additional class).

Description: With this message, the PUF is informed about the availability of the external equipment. When a connection is active, if the external equipment becomes unavailable the NAF is in charge to break down the communication.

Parameters:

Name	**Provided**	**Comment**
NCOID	M	Identifies the call. This information element is provided by the NAF.
ExtEquipAvailability	M	Indicates the external equipment availability.

Related: None.

6.3.29 CExtEquipBlockDiallingInd

Class: 6 (Additional class).

Description : With this message, the PUF get the dialling information made by the user with the keypad of the external equipment in the case of a block sending. This message contains the complete remote address and/or the remote subaddress.

Parameters:

Name	**Provided**	**Comment**
NCOID	M	Identifies the call. This information element is provided by the NAF.
ExtEquipBlockDialling	M	Provides to the PUF the remote address and/or subaddress in the case where the external equipment allows the block sending.

Related: None.

6.3.30 CExtEquipKeyPressedInd

Class: 6 (Additional class).

Description: With this message, the PUF get the dialling information made by the user with the keypad of the external equipment in the case of an overlap sending. One message is provided to the PUF for each key pressed.

Parameters:

Name	**Provided**	**Comment**
NCOID	M	Identifies the call. This information element is provided by the NAF.
ExtEquipKeyPressed	M	Provides to the PUF the code of the pressed key if the external equipment dials in the overlap sending mode.

Related: None.

6.3.31 CExtEquipOffHookInd

Class: 6 (Additional class).

Description: With this message, the PUF is informed that the handset of the external equipment is off-hooked. Depending on the type of external equipment and on the current state of the connection, this message can be interpreted according to different ways (see figures 9, 10 and 11).

Parameters:

Name	Provided	Comment
NCOID	M	Identifies the call. This information element is provided by the NAF.

Related: CExtEquipOnHookInd.

6.3.32 CExtEquipOnHookInd

Class: 6 (Additional class).

Description: With this message, the PUF is informed that the handset of the external equipment is on-hooked. Following the type of external equipment and following the current state of the connection, this message can be interpreted according to different ways (see figures 9, 10 and 11).

Parameters:

Name	Provided	Comment
NCOID	M	Identifies the call. This information element is provided by the NAF.

Related: CExtEquipOffHookInd.

6.3.33 User to User information exchange

The use of user-to-user information exchange is dependent on the user-to-user service level provided by the Network or the subscription.

In ETS 300 102-1 [2] three user-to-user service levels are defined:

- service 1:
  user-to-user information exchange during the set-up and clearing phase of a call;
- service 2:
  user-to-user information exchange during call establishment;
- service 3:
  user-to-user information exchange while a call is in the Active state.

For the PUF, the use of UserToUserInfo parameter inside messages and the UserInformation messages is depending on the service level.

The following usage of UserToUserInfo parameter and UserInformation messages is defined, relating to the service level:

- service 1:
  using the UserToUserInfo parameter in:
  - CAlertReq;
  - CAlertInd;
  - CConnectReq;
  - CConnectInd;
  - CConnectRsp;
  - CConnectCnf;
  - CDisconnectReq;
  - CDisconnectInd;

- service 2:
  using the CUserInformation messages between the sending/receiving of CAlertReq/Ind and CConnectRsp/Cnf messages.

- service 3:
  using CUserInformation messages in the active state of a call.

All three services may be used separately or in any combination in association with a single call.

NOTE: Services 2 and 3 are currently provided using the method described in ETS 300 102-1 [2]. An ETS describing the user-to-user Information Exchange using supplementary services is being developed. The use of this new method is for further study.

6.3.34 Implementation of supplementary services

This subclause gives an explanation of how the supplementary services which were described in subclause 5.3.3.2 function in relation to the ISDN PCI and which definition is provided in the ISDN PCI description.

6.3.34.1 Multiple Subscriber Number (MSN)

Description: MSN provides the possibility to assign multiple ISDN numbers to a single ISDN interface. The MSN can be used as a selection criteria for accepting incoming calls. A PUF can create several NCOs. By doing this it is sensitive to several MSN at the same time.

Operation: The destination number shall be presented to the called PUF if it has subscribed to MSN. The PUF shall react as follows:

if the MSN digits are applicable to its identity it shall react in the predetermined way;

if there are no MSN digits it shall react as if it is a normal call.

Implementation: The MSN of the called user shall be inserted in the CalledNumber field, the MSN of the calling user shall be inserted in the CallingNumber field.

Relevant fields:	CallingNumber	(subclause 6.6.11).
	CalledNumber	(subclause 6.6.7).

6.3.34.2 Direct Dialling In (DDI)

Description: DDI enables a user to directly call another user on an Integrated Services Private Branch eXchange (ISPBX) or other private system without intervention.

The part of the ISDN number, which is significant to the called user, is passed to that user during the process of the call. This supplementary service is based on the use of the ISDN number and does not include sub-addressing.

Operation: A PUF which wants to use this supplementary service shall include the correct number in the CalledNumber field.

Relevant fields: CallingNumber (subclause 6.6.11).

6.3.34.3 Calling Line Identification Presentation (CLIP)

Description: CLIP provides the called user with the calling user's ISDN number, possibly with sub-address information.

Operation: The PUF shall be supplied with the number of the calling user, possibly including sub-address, at the start of the call on all incoming calls.

Implementation: The PUF shall receive the ISDN number of the calling user in the CallingNumber field and the sub-address, if available, of the calling user in the CallingSubaddress field.

The PUF shall get an indication that the CallingNumber is not available, if the calling user has activated the Calling Line Identification Restriction (CLIR) supplementary service (see subclause 6.3.34.4).

Relevant fields: CallingNumber (subclause 6.6.11);

CallingSubaddress (subclause 6.6.12).

6.3.34.4 Calling Line Identification Restriction (CLIR)

Description: CLIR allows the calling user to restrict the presentation of its ISDN number and sub-address to the called user.

Operation: The calling PUF may indicate, either by subscription or by indication or by default, that his ISDN number may not be supplied to the called user, if this called user has CLIP. The PUF which has the CLIP supplementary service shall get an indication that the ISDN number of the calling user is not available.

If subscribed to in permanent mode, CLIR is always active.

If subscribed to in per call basis mode, CLIR is activated or deactivated upon request, depending on the default selected at subscription.

Implementation: To override the default, in a per call subscription, the calling PUF shall include the CallingNumber field, with the subfield Presentation set to "restricted" or "allowed".

Relevant fields: CallingNumber (subclause 6.6.11);

subfield Presentation in CallingNumber (subclause 6.6.11).

6.3.34.5 Subaddressing (SUB)

Description: SUB allows the called user to expand his addressing capacity beyond the normal ISDN number capacity. To one ISDN number, several subaddresses can be allocated to identify terminals directly. Only the called user shall define the significance of the subaddress.

Operation: The provided subaddress shall be transported transparently by the network. A receiving terminal can use the information for selection purposes.

Implementation: The calling PUF shall include the CalledSubaddress field in the connect message to activate this supplementary service.

Relevant fields: CalledSubaddress (subclause 6.6.8).

6.3.34.6 Advice of Charge during call (AOC-D)

Description: This supplementary service allows the PUF to obtain charging information during a connection.

Operation: Either at the establishment of the connection the PUF can activate the AOC-D supplementary service, or this service is available for every connection.

During the connection the PUF shall get the subtotal for this connection, either as currency or as charging units. At the end of the connection, the PUF shall obtain the total charging information for the connection.

Implementation: The PUF shall activate the AOC-D supplementary service, by including the corresponding facility field in the connect message.

After activation, the charging information shall be presented by the PUF using the facility field. For the coding of this field see subclause 6.6.30.

Errors shall be reported by using the facility field. The PUF shall not take any protocol action upon receiving an error.

Relevant fields: Facility (subclause 6.6.30).

6.3.34.7 Advice of Charge at end of call (AOC-E)

Description: This supplementary service allows the PUF to obtain charging information at the end of a connection.

Operation: Either at the establishment of the connection the PUF can activate the AOC-E supplementary service, or this service is available for every connection.

At the end of the connection, the PUF shall obtain the total charging information for the connection either as currency or as charging units.

Implementation: The PUF shall activate the AOC-E supplementary service, by including the corresponding facility field in the connect message.

After activation, the charging information shall be presented by the PUF using the facility field. For the coding of this field see subclause 6.6.30.

Errors shall be reported by using the facility field. The PUF shall not take any protocol action upon receiving an error.

Relevant fields: Facility (subclause 6.6.30).

6.3.34.8 Call Waiting (CW)

Description: Call Waiting is a supplementary service applicable to circuit-switched bearer services which permits a PUF to be informed of an incoming call when the terminal equipment is busy. The PUF has then the choice of accepting, rejecting or ignoring the waiting call. The CW supplementary service is considered meaningful when applied to the telephony teleservice and the speech and 3,1 kHz audio bearer services. Furthermore, it can be applied to other circuit-switched services.

Operation: In the case of an incoming call to the busy terminal equipment, if the PUF has control of a (circuit) information channel and is compatible with this incoming call, it can proceed with the call, by sending an ALERTING message to the network.

As a network option, a subscriber option may be offered to the calling user that it shall receive notification that its call is waiting.

Implementation: In the case where no channel is available, the called PUF shall receive a CConnectInd message with the channelIdentification field with the value "nochannel".

Whether a channel is available or not, the PUF shall indicate that it proceeds with the waiting call by sending a CAlertReq.

When making a call, the PUF might receive a CNotifyInd with a Notify field with value "CallWaiting" to indicate the call is waiting call.

Relevant field:	ChannelIdentification	(subclause 6.6.17);
	NotificationIndicator	(subclause 6.6.51).

6.3.34.9 Connected Line Identification Presentation (COLP)

Description: This supplementary service is a supplementary service offered to the calling PUF. It provides the connected PUF's ISDN number, possibly with subaddress information to the calling PUF.

The COLP supplementary service is not a dialling check but an indication to the calling PUF of the connected address. In a full ISDN environment, the connected line identity shall include all the information necessary to unambiguously identify the connected line.

The connected line identify may include the ConnectedSubaddress information element generated by the connected PUF which shall be transparently transported by the network. The network cannot be responsible for the content of this ConnectedSubaddress.

Operation: If the calling PUF is provided with the COLP supplementary service, the calling PUF shall receive a CConnectCnf message which contains the ConnectedNumber field and optionally the ConnectedSubaddress field of the connected PUF.

Implementation: The network invokes automatically the COLP supplementary service on each outgoing call made by the calling PUF.

The connected line identity supplied to the calling PUF is made up of a number of information units:

- the national ISDN number;
- the country code and possible other indications only for international calls;
- optionally subaddress information, if provided by the connected PUF.

Relevant fields:	ConnectedNumber	(subclause 6.6.21);
	ConnectedSubaddress	(subclause 6.6.22).

6.3.34.10 Connected Line Identification Restriction (COLR)

Description: The COLR supplementary service is a supplementary service offered to the connected PUF to prevent presentation of the connected PUF's ISDN number and subaddress information, (if any), to the calling PUF.

Operation: If the Calling PUF subscribes to the COLP supplementary service and the COLR supplementary service has been invoked by the called PUF, then the calling PUF shall receive a CConnectCnf message without the ConnectedNumber field and the ConnectedSubaddress field due to restriction.

Implementation: Two subscription options are possible:

- permanent mode (active for all calls);
- temporary mode (specified py the called PUF per call).

Relevant fields:	ConnectedNumber	(subclause 6.6.21);
	ConnectedSubaddress	(subclause 6.6.22).

6.4 User Plane messages

The User Plane messages provide a CCITT Recommendation X.213-access to different protocol stacks. Following is a list and short description of all User Plane messages. Table 12 gives an overview of NMA messages.

Table 12: Overview of NMA messages

Mess. Identif.	Class	Message Name	Purpose of Message
301	1	U3ConnectReq	Request establishment of a user connection.
302	1	U3ConnectInd	Indicate establishment of a user connection has been requested.
303	1	U3ConnectRsp	Indicate acceptance of user connection establishment.
304	1	U3ConnectCnf	Confirm user connection has been established.
305	1	U3DisconnectReq	Request removal of user connection.
306	1	U3DisconnectInd	Indicate removal of user connection.
307	1	U3DataReq	Request data transfer on an established user connection.
308	1	U3DataInd	Indicate arrival of transferred data on an established user connection.
309	1	U3ExpeditedDataReq	Request expedited data transfer on an established user connection.
310	1	U3ExpeditedDataInd	Indicate presence of transferred expedited data on an established user connection.
311	1	U3ResetReq	Request reset to initial state of an established user connection.
312	1	U3ResetInd	Indicate reset to initial state of an established user connection.
313	1	U3ResetRsp	Indicate acceptance of reset to initial state of an established user connection.
314	1	U3ResetCnf	Confirm acceptance of reset to initial state of an established user connection.
315	1	U3DataAcknowledgeReq	Request acknowledgement of data received on an established user connection.
316	1	U3DataAcknowledgeInd	Indicate acknowledgement of data transferred on an established user connection.
317	1	U3ReadyToReceiveReq	Used to perform flow control for a user connection.
318	1	U3ReadyToReceiveInd	Used to indicate flow control status on a user connection.
319	1	U3ErrorInd	Indicate an error.

Not all of the above messages are used in every cases. The following table identifies the messages used in the specific User Plane protocols. A cross placed in the protocol column indicates application of the relevant message to the protocol.

Table 13 gives an overview of NMA messages including the protocol dependencies. The NULL column refers to the TMA messages presented in table 14.

Table 13: Overview of NMA messages

Mess. Identif.	**Message Name**	**ETS 300 080 [1]**	**ISO/IEC 8208 [3]**	**CCITT Rec. T.70**	**NULL (NOTE 1)**
301	U3ConnectReq	X	X	X	
302	U3ConnectInd	X	X	X	
303	U3ConnectRsp	X	X	X	
304	U3ConnectCnf	X	X	X	
305	U3DisconnectReq	X	X	X	
306	U3DisconnectInd	X	X	X	
307	U3DataReq	X	X	X	X
308	U3DataInd	X	X	X	X
309	U3ExpeditedDataReq		X		
310	U3ExpeditedDataInd		X		
311	U3ResetReq	X	X		
312	U3ResetInd	X	X		
313	U3ResetRsp	X	X		
314	U3ResetCnf	X	X		
315	U3DataAcknowledgeReq		X		
316	U3DataAcknowledgeInd		X		
317	U3ReadyToReceiveReq	X	X		
318	U3ReadyToReceiveInd	X	X		
319	U3ErrorInd				see NOTE 2
NOTE 1: This NULL layer 3 represents several protocols on layer 2 as described in subclause 5.4.4.2. This representation is also copied to the protocol statements with the User Plane messages and the User Plane parameters. NOTE 2: Used if layer 2 does not support flow control.					

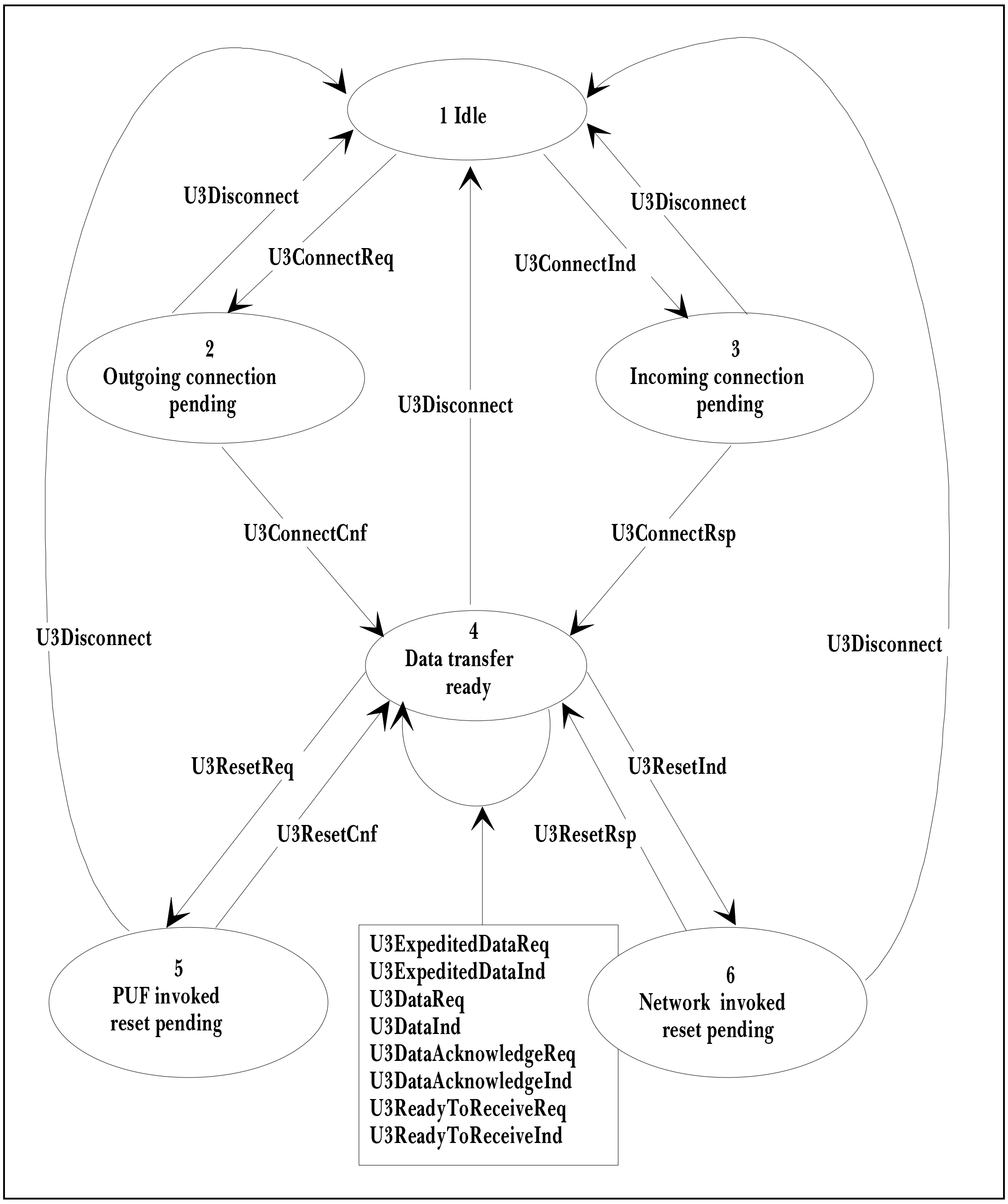
Table 14 gives an overview of TMA messages.

Table 14: Overview of TMA messages

Mess. Identif.	**Class**	**Message Name**	**Purpose of Message**
307	1	U1DataReq	Request transfer of data.
308	1	U1DataInd	Indicate arrival of transferred data.
319	1	U1ErrorInd	Indicate an error.
NOTE: The same message identifiers are used in NMA and TMA, so the messages only differ in kind of abstraction layer.			

6.4.1 Sequencing of User Plane messages

Figure 13 shows the different states a user connection using the NMA messages can get and in which order these messages shall be used.



NOTE: Where U3Disconnect appears it can be either U3DisconnectReq or U3DisconnectInd.

Figure 13: Overview of the User Plane messages

6.4.2 Co-ordination function

This subclause gives an explanation of how the co-ordination function, which was described in subclause 5.2.4, can be used by a PUF. The co-ordination function provides to the PUF a Connection Oriented mode Network Service (CONS) according to CCITT Recommendation X.213 [7]. Therefore, if the co-ordination function is used, the layer 2 and layer 3 protocols used are the selected protocols from the NMA.

If a PUF wants to use the co-ordination function, all messages from the Administration Plane can still be used by the PUF, since the co-ordination function does not affect this plane. For achieving a connection which is to be NAF co-ordinated, the PUF has exchanges the following message:

ACreateNCOReq, with NCOType U3 and the relevant information.

A connection can then be requested using the U3ConnectReq. All other messages in the User Plane can still be used by the PUF. No Control Plane messages can be used in combination with an NCO of type U3. For interaction overview see subclause 5.6. For the State diagram, see Annex E, Clause E.3.

6.4.3 U3ConnectReq

Class: 1 (Basic class).

Description: This message allows a PUF to initiate the establishment of a user connection.

Parameters:

Name	Required	Comment
NCOID	M	Identifies the user connection.
CalledDTEAddress	O	If provided, this value supersedes the NCO value.
CalledDTEAddressExt	O	If provided, this value supersedes the NCO value.
CallingDTEAddress	O	If provided, this value supersedes the NCO value.
CallingDTEAddressExt	O	If provided, this value supersedes the NCO value.
ReceiptConfirm	O	Used to request confirmation of data receipt for this user connection.
ExpeditedData	O	Used to request use of expedited data for the user connection.
QOSParameters	O	Quality of Service.
UserData	O	Maximum length is 16, or 128 if FastSelect parameter is used.
Bcug	O	Used to specify the ISO 8208 [3] Bilateral Closed User Group facility. If specified then Called address parameters are not allowed.
FastSelect	O	If used this parameter invokes the use of the Fast Select Facility in the ISO 8208 [3] Call Request.
PacketSize	O	Requested value, overrides any value specified as part of NCO creation.
WindowSize	O	Requested value, overrides any value specified as part of NCO creation.
FacilityData	O	Used to supply ISO 8208 [3] facilities. If present, the following facilities shall be overridden by information found elsewhere in this message: - BCUG; - FastSelect; - Called Address Extension; - Calling Address Extension.

Protocols: This message is used in following User Plane protocols:

ETS 300 080 [1]	ISO 8208 [3]	T.70	NULL
X	X	X	

Related: U3ConnectCnf.

6.4.4 U3ConnectInd

Class: 1 (Basic class).

Description: This message informs a PUF of an incoming demand to establish a user connection.

Parameters:

Name	**Provided**	**Comment**
NCOID	M	Identifies the user connection.
CalledDTEAddress	O	Called address.
CalledDTEAddressExt	O	Called Address extension.
CallingDTEAddress	O	Calling address.
CallingDTEAddressExt	O	Calling address extension.
ReceiptConfirm	O	Indicates if confirmation of data receipt is required on this user connection
ExpeditedData	O	Indicates if use of expedited data is allowed on this user connection.
QOSParameters	O	Quality of Service.
UserData	O	Maximum length is 16, or 128 if FastSelect parameter is present.
Bcug	O	Used to pass ISO 8208 [3] Bilateral Closed User Group facility information. If present then addressing information shall not be present.
FastSelect	O	Authorisation type to transmit UserData.
PacketSize	M	Value to be used for this user connection.
WindowSize	M	Value to be used for this user connection.
FacilityData	O	Used to supply ISO 8208 [3] facilities. The following facilities, if present, are presented by the use of specific parameters: - BCUG; - FastSelect; - Called Address Extension; - Calling Address Extension.

Protocols: This message is used in following User Plane protocols:

ETS 300 080 [1]	**ISO 8208 [3]**	**T.70**	**NULL**
X	X	X	

Related: U3ConnectRsp.

6.4.5 U3ConnectRsp

Class: 1 (Basic class).

Description: This message allows a PUF to accept the establishment of a user connection.

Parameters:

Name	**Required**	**Comment**
NCOID	M	Identifies the user connection.
CalledDTEAddress	O	Called address.
CalledDTEAddressExt	O	Called Address extension.
CallingDTEAddress	O	Calling address.
CallingDTEAddressExt	O	Calling address extension.
RespondingDTEAddress	O	The address used to accept the user connection. This may be different from the original called address.
RespondingDTEAddressExt	O	The address extension used to accept the user connection. This may be different from the original called address extension.
ReceiptConfirm	O	Used to accept or not accept use of receipt confirmation for data on this user connection.
ExpeditedData	O	Used to accept or not accept use of expedited data on this user connection.
QOSParameters	O	Quality of Service.
UserData	O	Maximum length is 16, or 128 if FastSelect parameter was present on U3ConnectInd.
PacketSize	O	Used to indicate agreed value.
WindowSize	O	Used to indicate agreed value.
FacilityData	O	Used to supply ISO 8208 [3] facilities.

Protocols: This message is used in following User Plane protocols:

ETS 300 080 [1]	**ISO 8208 [3]**	**T.70**	**NULL**
X	X	X	

Related: U3ConnectInd.

6.4.6 U3ConnectCnf

Class: 1 (Basic class).

Description: This message informs the PUF on the establishment of a user connection.

Parameters:

Name	Provided	Comment
NCOID	M	Identifies the user connection.
CalledDTEAddress	O	Called address.
CalledDTEAddressExt	O	Called Address extension.
CallingDTEAddress	O	Calling address.
CallingDTEAddressExt	O	Calling address extension.
RespondingDTEAddress	O	The address used to accept the user connection. This may be different from the original called address.
RespondingDTEAddressExt	O	The address extension used to accept the user connection. This may be different from the original called address extension.
ReceiptConfirm	O	Indicates if receipt confirmation of data can be used on this user connection.
ExpeditedData	O	Indicates if expedited data can be used on this user connection.
QOSParameters	O	Quality of Service.
UserData	O	Maximum length is 16, or 128 if FastSelect parameter was present on U3ConnectReq.
PacketSize	M	Value to be used for this user connection.
WindowSize	M	Value to be used for this user connection.
FacilityData	O	Used to supply ISO 8208 [3] facilities.

Protocols: This message is used in following User Plane protocols:

ETS 300 080 [1]	ISO 8208 [3]	T.70	NULL
X	X	X	

Related: U3ConnectReq.

6.4.7 U3DisconnectReq

Class: 1 (Basic class).

Description: This message allows a PUF to remove a user connection.

Parameters:

Name	**Required**	**Comment**
NCOID	M	Identifies the user connection.
X213Cause	O	X.213 reason to remove the user connection. Both X213Cause and X25Cause cannot be used on the same message. If neither X213Cause and X25Cause are supplied the X213Cause parameter with the value of disconnection-normal condition shall be used.
RespondingDTEAddress	O	The address used to accept the user connection. This may be different from the original called address.
RespondingDTEAddressExt	O	The address extension used to accept the user connection. This may be different from the original called address extension.
UserData	O	Only allowed if FastSelect parameter was specified during the user connection establishment. Maximum size of 128 octets.
X25Cause	O	ISO 8208 [3] reason to remove the user connection. Both X213Cause and X25Cause cannot be used on the same message.
X25Diagnostic	C	Complementary information for ISO 8208 [3] reason. Optional if X25Cause parameter supplied else not allowed.
FacilityData	O	Used to supply ISO 8208 [3] facilities.
NOTE CCITT Recommendation X.213 [7] cause is exclusive with CCITT Recommendation X.25 information. If CCITT Recommendation X.25 cause, optionally associated with the X.25 diagnostic, is used the X.213 [7] cause shall not appear.		

Protocols: This message is used in following User Plane protocols:

ETS 300 080 [1]	**ISO 8208 [3]**	**T.70**	**NULL**
X	X	X	

Related: None.

6.4.8 U3DisconnectInd

Class: 1 (Basic class).

Description: This message informs a PUF that a user connection has been removed.

Parameters:

Name	**Provided**	**Comment**
NCOID	M	Identifies the user connection.
X213Origin	M	Identifies the initiator of the user connection removal.
X213Cause	O	X.213 Reason to remove the user connection.
UserData	O	Only allowed if FastSelect parameter was specified during the user connection establishment. Maximum size of 128 octets.
RespondingDTEAddress	O	The address used to accept the user connection. This may be different from the original called address extension.
RespondingDTEAddressExt	O	The address extension used to accept the user connection. This may be different from the original called address extension.
X25Cause	O	ISO 8208 [3] Reason to remove the user connection. Both X213Cause and X25Cause cannot be used on the same message.
X25Diagnostic	C	Complementary information for ISO 8208 [3] Reason. Optional if X25Cause parameter supplied else not allowed.
FacilityData	O	Used to supply ISO 8208 [3] facilities.
NOTE: CCITT Recommendation X.213 [7] cause is exclusive with CCITT Recommendation X.25 information. If CCITT Recommendation X.25 cause, optionally associated with the CCITT Recommendation X.25 diagnostic, is used the CCITT Recommendation X.213 [7] cause shall not appear.		

Protocols: This message is used in following User Plane protocols:

ETS 300 080 [1]	**ISO 8208 [3]**	**T.70**	**NULL**
X	X	X	

Related: None.

6.4.9 U3DataReq

Class: 1 (Basic class).

Description: This message allows a PUF to send a data packet. The size of a data packet is restricted to the data packet size negotiated during the user connection establishment.

Parameters:

Name	Required	Comment
NCOID	M	Identifies the user connection.
Bit_DQM	O	Used to set the ISO 8208 Qualifier bit, the ISO 8208 [3] More Bit and to request confirmation of receipt of data or to set the T70 More Bit.

Remark: Data to send are mandatory. They are not provided as a parameter of the message.

If the NCO has been created with the U3Protocol parameter set to L3Protocol set to NULL (4) and L2Protocol set to 'Frame Oriented Transparent' (1), then HDLC framing is issued. The FCS is inserted at the end of each data block with the flag.

Mandatory data shall be provided in the data buffer.

Protocols: This message is used in following User Plane protocols:

ETS 300 080 [1]	ISO 8208 [3]	T.70	NULL
X	X	X	X

Related: U3ReadyToReceiveInd.

6.4.10 U3DataInd

Class: 1 (Basic class).

Description: This message indicates the presence of received data to a PUF. The size of a data packet is restricted to the data packet size negotiated during the user connection establishment.

Parameters:

Name	Provided	Comment
NCOID	M	Identifies the user connection.
Bit_DQM	O	Used to indicate the ISO 8208 [3] Qualifier bit value, the ISO 8208 [3] More Bit value and the need of confirmation of data reception or to indicate the T70 More Bit reception.

Remark: Data received are always provided, but not as a parameter of the message.

If the NCO has been created with the U3Protocol parameter set to L3Protocol set to NULL (4) and L2Protocol set to 'Frame Oriented Transparent' (1), then HDLC framing is issued. The FCS is extracted from the data block by the NAF before the data block is provided to the PUF.

Data is provided in the data buffer. The buffer, in this case, shall be mandatory.

Protocols: This message is used in following User Plane protocols:

ETS 300 080 [1]	ISO 8208 [3]	T.70	NULL
X	X	X	X

Related: U3ReadyToReceiveReq.

6.4.11 U3ExpeditedDataReq

Class: 1 (Basic class).

Description: This message allows a PUF to send expedited data. This data is not constrained by the flow control mechanism used to control U3DataReq messages.

Parameters:

Name	Required	Comment
NCOID	M	Identifies the user connection.
UserData	M	Expedited data to transfer.

Protocols: This message is used in following User Plane protocols:

ETS 300 080 [1]	ISO 8208 [3]	T.70	NULL
	X		

Related: None.

6.4.12 U3ExpeditedDataInd

Class: 1 (Basic class).

Description: This message indicates to a PUF the reception of expedited data. This data was not constrained by the flow control mechanisms used to control U3DataInd messages.

Parameters:

Name	**Provided**	**Comment**
NCOID	M	Identifies the user connection.
UserData	M	Expedited data received.

Protocols: This message is used in following User Plane protocols:

ETS 300 080 [1]	**ISO 8208 [3]**	**T.70**	**NULL**
	X		

Related: None.

6.4.13 U3ResetReq

Class: 1 (Basic class).

Description: This message allows the PUF to reset a user connection.

Parameters:

Name	**Provided**	**Comment**
NCOID	M	Identifies the use connection.
X213Cause	O	X.213 Reason to reset the user connection. If neither X213Cause and X25Cause are supplied the X213Cause parameter with the value of disconnection-normal condition will be used.
X25Cause	O	ISO 8208 [3] Reason to reset the user connection.
X25Diagnostic	C	ISO 8208 [3] Complementary information. Optional only if X25Cause supplied, else not allowed.
NOTE: X.213 cause is exclusive with X.25 information. If X.25 cause, optionally associated with the X.25 diagnostic, is used the X.213 cause shall not appear.		

Protocols: This message is used in following User Plane protocols:

ETS 300 080 [1]	**ISO 8208 [3]**	**T.70**	**NULL**
X	X		

Related: U3ResetCnf.

6.4.14 U3ResetInd

Class: 1 (Basic class).

Description: This message informs the PUF of the reset of a user connection.

Parameters:

Name	**Provided**	**Comment**
NCOID	M	Identifies the user connection.
X213Origin	M	Identifies the initiator of the reset user connection.
X213Cause	O	X213 Reason to reset the user connection.
X25Cause	O	ISO 8208 [3] Reason to reset the user connection.
X25Diagnostic	O	ISO 8208 [3] Complementary information. Optional only if X25Cause supplied, else not allowed.
NOTE: X.213 cause is exclusive with X.25 information. If X.25 cause, optionally associated with the X.25 diagnostic, is used the X.213 cause shall not appear.		

Protocols: This message is used in following User Plane protocols:

ETS 300 080 [1]	**ISO 8208 [3]**	**T.70**	**NULL**
X	X		

Related: U3ResetRsp.

6.4.15 U3ResetRsp

Class: 1 (Basic class).

Description: This message allows the PUF to respond to a user connection reset, indicating that it has dealt with the reset and is ready to proceed.

Parameters:

Name	**Required**	**Comment**
NCOID	M	Identifies the user connection.

Protocols: This message is used in following User Plane protocols:

ETS 300 080 [1]	**ISO 8208 [3]**	**T.70**	**NULL**
X	X		

Related: U3ResetInd.

6.4.16 U3ResetCnf

Class: 1 (Basic class).

Description: This message completes the reset operation of a user connection. The PUF is now able to transfer data once again.

Parameters:

Name	**Provided**	**Comment**
NCOID	M	Identifies the user connection.

Protocols: This message is used in following User Plane protocols:

ETS 300 080 [1]	**ISO 8208 [3]**	**T.70**	**NULL**
X	X		

Related: U3ResetReq.

6.4.17 U3DataAcknowledgeReq

Class: 1 (Basic class).

Description: This message allows the PUF to acknowledge received Data. It should be used when a U3DataInd message is received with the ReceiptConfirm parameter set indicating receipt of confirmation is required.

Parameters:

Name	**Required**	**Comment**
NCOID	M	Identifies the user connection.

Protocols: This message is used in following User Plane protocols:

ETS 300 080 [1]	**ISO 8208 [3]**	**T.70**	**NULL**
	X		

Related: U3DataInd.

6.4.18 U3DataAcknowledgeInd

Class: 1 (Basic class).

Description: This message informs the PUF of the reception of an acknowledgement for transferred data. It acknowledges a U3DataReq message that was sent with the ReceiptConfirm parameter requesting confirmation of data reception.

Parameters:

Name	**Provided**	**Comment**
NCOID	M	Identifies the user connection.

Protocols: This message is used in following User Plane protocols:

ETS 300 080 [1]	**ISO 8208 [3]**	**T.70**	**NULL**
	X		

Related: U3DataReq.

6.4.19 U3ReadyToReceiveReq

Class: 1 (Basic class).

Description: This message allows the PUF to indicate to the NAF if it can accept incoming data (U3DataInd message). This message can only apply to an already established user connection. Setting the ReadyFlag parameter to TRUE allows the NAF to transfer incoming data to the PUF. Setting the ReadyFlag to FALSE inhibits the transfer.

This flow control mechanism does not imply an end-to-end flow control.

Parameters:

Name	Required	**Comment**
NCOID	M	Identifies the user connection.
ReadyFlag	M	This flag indicates whether or not the PUF is ready to accept incoming data.

Remarks: For a given connection, if more than one message with the same flag value is sent, it shall be ignored by the NAF.

Protocols: This message is used in following User Plane protocols:

ETS 300 080 [1]	**ISO 8208 [3]**	**T.70**	**NULL**
X	X		

Related: U3DataInd.

6.4.20 U3ReadyToReceiveInd

Class: 1 (Basic class).

Description: This message allows the NAF to indicate to the PUF if the user connection permits the sending of data (U3DataReq messages). This message can only apply to an already established user connection. If the ReadyFlag parameter value is FALSE, the NAF cannot send data. If the value is TRUE the NAF indicates that data transfer is allowed.

This flow control mechanism does not imply an end-to-end flow control.

Parameters:

Name	Provided	**Comment**
NCOID	M	Identifies the user connection.
ReadyFlag	M	This flag indicates whether or not the NAF is ready to receive data for transmission on a user connection.

Protocols: This message is used in following User Plane protocols:

ETS 300 080 [1]	**ISO 8208 [3]**	**T.70**	**NULL**
X	X		

Related: U3DataReq.

6.4.21 U3ErrorInd

Class: 1 (Basic class).

Description: This message indicates to a PUF that an error has occurred.

Parameters:

Name	**Provided**	**Comment**
NCOID	M	Identifies the User Plane connection.
CompletionStatus	M	Identifies type of error.

Protocols: This message is used in following User Plane protocols:

ETS 300 080 [1]	**ISO 8208 [3]**	**T.70**	**NULL**
			X
NOTE: Used if Layer 2 and Layer 3 does not support flow control.			

Related: None.

6.4.22 U1DataReq

Class: 1 (Basic class).

Description: This message allows a PUF to send *transparent* data on the B channel. By default, the data is sent without any protocol as byte stream. The synchronisation used on the B-channel is character oriented. When no more data is available, the NAF sends the default octet provided in the Attribute Set used for this connection.

Parameters:

Name	**Required**	**Comment**
NCOID	M	Identifies the Control Plane connection.

Remark: Data to send shall be mandatory. They are not provided as a parameter of the message.

Mandatory data shall be provided in the data buffer.

Related: None.

6.4.23 U1DataInd

Class: 1 (Basic class).

Description: This message indicates to a PUF received *transparent* data on the B-channel. The data is received without any protocol or control as byte stream. The byte provided as the default padding character in the Attribute Set is not extracted from the data received.

Parameters:

Name	**Provided**	**Comment**
NCOID	M	Identifies the Control Plane connection.

Remark: Data received is always provided, but not as a parameter of the message.

Data is provided in the data buffer. This buffer, in this case, shall be mandatory.

Related: U1ErrorInd.

6.4.24 U1ErrorInd

Class: 1 (Basic class).

Description: This message indicates to a PUF that an error has occurred.

Parameters:

Name	**Provided**	**Comment**
NCOID	M	Identifies the Control Plane connection.
CompletionStatus	M	Identifies type of error.

Related: None.

6.5 Information presentation

In subclause 6.6, the types used shall be understood as:

-	Octet	referred to a byte (8 bits);
-	Boolean	referred to an octet with limited set of values (0 = FALSE, else = TRUE);
-	Octet-string	referred to an array of octets with a variable or fixed size;
-	IA5-string	referred to an Octet-string composed with octets in the IA5 alphabet.

Every parameter is encoded using TLV coding as following:

-	type	= 1 Octet;
-	length	= 1 Octet;
-	value	with octet boundary.

Fields included into the parameter are coded as structured information. The order inside this structured information is defined by the order provided into the presentation of the parameter itself in subclause 6.6. Omitted fields reduce the size of the parameter.

Values in parenthesis are decimal coded.

EXAMPLE: This example describes parameters of a CDisconnectInd message coming from the NAF.

The *CDisconnectInd* message identifier is provided by the NAF in the *MessageID* field of the PCIMPB structure as describe in the subclause 7.3. Parameters are placed into fill the *Message* parameter of the PciGetMessage function (see subclause 7.3.8). The *MessageActualUsedSize* field of the PCIMPB structure is filled with the 12 value.

The cause content is the "Normal call clearing" #16 cause, the diagnostic optional parameter is not provided. The NCOID value is 3. Values are given using the decimal format.

Total parameter length 12

NCOID type	49	
NCOID length	4	
NCOID value	3	(03 00 00 00)
Cause type	15	(CauseToPUF)
Cause length	4	
Cause value	16	
Cause Standard	1	(CCITT)
Cause location	1	(user)
Cause Recommendation	1	(Q.931)

In a byte oriented representation (in decimal), the content of the *buffer* results as follow:

```
49, 04, 03, 00, 00, 00,              (NCOID)
15, 4, 16, 01, 01, 01, 01            (CauseToPUF)
```

6.6 Message parameters

This subclause describes parameters for each plane presented. They are alphabetically ordered. The following table only gives an overview of message parameters being User Plane protocol dependent. The NULL column refers to the TMA set of messages.

Table 15: User Plane protocol dependent message parameters

Sub clause	**Message Parameter Name**	**ETS 300 080 [1]**	**ISO 8208 [3]**	**CCITT T.70**	**NULL**
6.6.1	Algorithm		X		
6.6.2	Bcug	X	X		
6.6.4	Bit_DQM	see NOTE	X	X	
6.6.5	CalledDTEAddress	X	X		
6.6.6	CalledDTEAddressExt	X	X		
6.6.9	CallingDTEAddress	X	X		
6.6.10	CallingDTEAddressExt	X	X		
6.6.29	ExpeditedData		X		
6.6.31	FacilityData	see NOTE	X		
6.6.32	FastSelect	see NOTE	X		
6.6.33	GroupID	X	X		
6.6.42	L3ConnectionMode		X		
6.6.43	L3InComingVCCount		X		
6.6.44	L3OutgoingVCCount		X		
6.6.45	L3TwoWayVCCount	X	X		
6.6.52	PacketSize	see NOTE	X		
6.6.54	QOSParameter	see NOTE	X		
6.6.55	ReadyFlag	X	X		
6.6.58	RespondingDTEAddress	X	X		
6.6.59	RespondingDTEAddressExt	X	X		
6.6.67	WindowSize	see NOTE	X		
6.6.70	X25Cause	X	X		
6.6.71	X25Diagnostics	X	X		
NOTE: This parameter shall be used in accordance with the rules of ETS 300 080 [1].					

6.6.1 Algorithm

Description: This parameter is used to pass the name of the security algorithm to be used to the NAF.

Type: 1.

Fields	Field type	Direction	Required	Comment
Algorithm	IA5-string	P	M	The security algorithm is identified by its name. The names of the available algorithms can be obtained using the Property information. "nosecurity": this value for this parameter indicates that security is no longer needed for the connection. 16 is maximum length.

Protocols: This parameter is used in following User Plane protocols:

ETS 300 080 [1]	ISO 8208 [3]	T.70	NULL
	X		

6.6.2 Bilateral closed user group (Bcug)

Description: This parameter is used to pass ISO 8208 [3] bilateral closed user group information to/from the PUF.

Type: 2.

Fields	Field type	Direction	Required	Comment
Bcug	Octet-string	B	M	Index to bilateral closed user group selected for user connection. 4 is the fixed length.

Protocols: This parameter is used in following User Plane protocols:

ETS 300 080 [1]	ISO 8208 [3]	T.70	NULL
X	X		

6.6.3 BearerCap

Description: This parameter is used to pass bearer capability to/from the PUF.

Type: 3.

Fields	Field type	Direction	Required	Comment
BearerCap	Octet-string	B	M	Bearer capability information element. Maximum length is 12.
NOTE: Values for this field are defined in the ETS 300 102-1 [2].				

6.6.4 Bit_DQM

Description: This parameter is used to pass to/from the PUF:

- need for receipt of data (bit 1). This bit is equivalent to the X.25 D bit;
- ISO/IEC 8208 [3] Qualifier bit value (bit 2);
- ISO/IEC 8208 [3] More Data bit value (bit 3).

Each information use a binary position. The Most Significant Bit (MSB) if the bit 8 and the Least Significant Bit is the bit 1. Bit 1 is for value 1, bit 2 for value 2 and bit 3 for value 4. The result value applying to this parameter is a the sum of the value for each bit (logical OR).

Type: 4.

Fields	Field type	Direction	Required	Comment
DQM	Octet	B	M	**Bit 1:** 1 - Confirmation of data reception is allowed or required. 0 - Confirmation of data reception is not allowed or not required. **Bit 2:** 1 - Set ISO 8208 [3] Qualifier bit. 0 - Reset ISO 8208 [3] Qualifier bit **Bit 3:** 1 - Set ISO 8208 [3] More bit. 0 - Reset ISO 8208 [3] More bit

Protocols: This parameter is used in following User Plane protocols:

ETS 300 080 [1]	ISO 8208 [3]	T.70	NULL
see NOTE	X	X	
NOTE: This parameters shall be used in accordance with ETS 300 080 [1].			

Remarks: Invalid use of the More bit with the Qualifier bit shall result in the user connection being reset.

6.6.5 CalledDTEAddress

Description: This parameter is used to pass remote DTE address information to/from the PUF.

Type: 5.

Fields	Field type	Direction	Required	Comment
Address	IA5-string	B	M	15 octets is the maximum length

Protocols: This parameter is used in following User Plane protocols:

ETS 300 080 [1]	ISO 8208 [3]	T.70	NULL
X	X		

Remark: The BCD translation is provided by the NAF.

In the message exchange from PUF to NAF this parameter shall either be supplied in the NCO or in the appropriate message.

6.6.6 CalledDTEAddressExt

Description: This parameter is used to pass remote DTE address extension information to/from the PUF.

Type: 6.

Fields	Field type	Direction	Required	Comment
AddressExt	IA5-string	B	M	40 octets is the maximum length.

Protocols: This parameter is used in following User Plane protocols:

ETS 300 080 [1]	ISO 8208 [3]	T.70	NULL
X	X		

Remark: The BCD translation is provided by the NAF.

6.6.7 CalledNumber

Description: This parameter is used to pass details concerning the called address to/from the PUF.

Type: 7.

Fields	Field type	Direction	Required	Comment
NumberType	Octet	B	M	Default (255) - default is unknown
				unknown (0)
				international (1)
				national specific (2)
				network (3)
				subscriber (4)
				abbreviated (6)
NumberPlan	Octet	N	M	Default (255) - default is unknown
				unknown (0)
				isdn (1)
				data (3)
				telex (4)
				national (8)
				private (9)
Number	IA5-string	B	M	20 is maximum length
NOTE: In the message exchange from PUF to NAF this parameter shall either be supplied in the NCO or in the appropriate message.				

6.6.8 CalledSubaddress

Description: This parameter is used to pass the Called Subaddress to/from the PUF.

Type: 8.

Fields	Field type	Direction	Required	Comment
NumberType	Octet	B	M	nsap (0)
				user (2)
Indicator	Octet	B	M	even (0)
				odd (1)
				This field is only meaningful if NumberType is set to user. It indicates if the number contains an odd or even number of BCD digits.
Number	IA5-string	B	M	20 is maximum length.

6.6.9 CallingDTEAddress

Description: This parameter is used to pass local DTE address information to/from the PUF.

Type: 9.

Fields	Field type	Direction	Required	Comment
Address	IA5-string	B	M	15 octets is the maximum length.

Protocols: This parameter is used in following User Plane protocols:

ETS 300 080 [1]	ISO 8208 [3]	T.70	NULL
X	X		

Remark: The BCD translation is provided by the NAF.

6.6.10 CallingDTEAddressExt

Description: This parameter is used to pass local DTE address extension information to/from the PUF.

Type: 10.

Fields	Field type	Direction	Required	Comment
AddressExt	IA5-string	B	M	40 octets is the maximum length

Protocols: This parameter is used in following User Plane protocols:

ETS 300 080 [1]	ISO 8208 [3]	T.70	NULL
X	X		

Remark: The BCD translation is provided by the NAF.

6.6.11 CallingNumber

Description: This parameter is used to pass details concerning the calling address to/from the PUF.

Type: 11.

Fields	Field type	Direction	Required	Comment
NumberType	Octet	B	M	Default (255) - default is unknown
				unknown (0)
				international (1)
				national specific (2)
				network (3)
				subscriber (4)
				abbreviated (6)
NumberPlan	Octet	B	M	Default (255) - default is unknown
				unknown (0)
				isdn (1)
				data (3)
				telex (4)
				national (8)
				private (9)
Presentation	Octet	B	M	Default (255) - default is allowed
				allowed (0)
				restricted (1)
				not available (2)
				Indicates whether the Number should be provided to the called user
Screening	Octet	B	M	Default (255) - default is usernotscreened
				usernotscreened (0)
				userverified (1)
				networkprovided (3)
				Indicates any checking that has been applied to the Number
Number	IA5-string	B	M	20 is maximum length

NOTE: Only "ISDN/telephony numbering plan" and "unknown" shall be allowed for the PUF as number plan identifier within the calling party number information element, when using CLIP.
Only "subscriber number", "national number" and "international number" shall be allowed for the PUF as type of number within the calling party number information element, when using CLIP and specifying a complete number.
Only "unknown" shall be allowed for the PUF as type of number within the calling party number information element, when using CLIP and specifying an incomplete number for DDI.

6.6.12 CallingSubaddress

Description: This parameter is used to pass Calling Subaddress details to/from the PUF.

Type: 12.

Fields	Field type	Direction	Required	Comment
NumberType	Octet	B	M	nsap (0)
				user (2)
Indicator	Octet	B	M	even (0)
				odd (1)
				This field is only meaningful if NumberType is set to user. It indicates if the number contains an odd or even number of BCD digits.
Number	IA5-string	B	M	20 is maximum length

6.6.13 CAttributeName

Description: This parameter is used to pass the name of a static set of Control Plane attributes from the PUF.

Type: 13.

Fields	Field type	Direction	Required	Comment
AttributeName	IA5-string	P	M	16 is maximum length.

6.6.14 CauseToNAF

Description: This parameter is used to pass Cause Information from the PUF to the NAF.

Type: 14.

Fields	Field type	Direction	Required	Comment
Cause	Octet	P	M	Cause value.

6.6.15 CauseToPUF

Description: This parameter is used to pass Cause Information from the NAF to the PUF.

Type: 15.

Fields	Field type	Direction	Required	Comment
Cause	Octet	N	M	Cause value
Standard	Octet	N	M	Default (255) - default is ccitt
				ccitt (0)
				international (1)
				national (2)
				network (3)
Location	Octet	N	M	Default (255) - default is user
				user (0)
				privatelocal (1)
				publiclocal (2)
				transit (3)
				publicremote (4)
				privateremote (5)
				international (7)
				networkbeyond (10)
Recommendation	Octet	N	M	Default (255) - default is Q931
				Q931 (0)
				X21 (3)
				X25 (4)
Diagnostics	Octet-string	N	C	Depends on the cause value.
				Length is fixed to 2.
				The lower octet contents least significant byte.

6.6.16 CDirection

Description: This parameter is used to pass information concerning the usage of a particular NCO to the NAF, for the Control Plane part.

Type: 16.

Fields	Field type	Direction	Required	Comment
Direction	Octet	P	M	listen (1)
				call (2)
				both (3)

6.6.17 ChannelIdentification

Description: This parameter is used to pass Channel Information from/to the PUF.

Type: 17.

Fields	Field type	Direction	Required	Comment
Selection	Octet	B	M	nochannel (0) - no channel is available Bchannel (1) anychannel (3) - use any available channel Dchannel (4)
Number	Octet	B	O	This optional parameter is used by the PUF to select a particular B channel. A value of 255 means select the first available B channel.

Remarks: For CConnectReq message all values of Selection except nochannel and D-Channel are supported.

The number field can be used on the CAttribute set parameter structure or with CConnectReq message to select a particular permanent connected B-channel, or D-channel in the case where multiple TEI's are supported.

6.6.18 ChargingInfo

Description: This parameter is used to Transmit the charging information, if any, relevant to an NCO, in the Administration Attribut Set Parameter.

Type: 18.

Fields	Field type	Direction	Required	Comment
Tag	Octet	N	M	charginginfo (3) chargingerror (4) (See subclause 6.6.30 : coding of FacilityTag)
Value	Octet-string	N	C	Length and content depend on the Tag. Absent if Tag is chargingerror. (See subclause 6.6.30 : coding of FacilityValue)

6.6.19 CompletionStatus

Description: This parameter is used to pass completion information to the PUF.

Type: 19.

Fields	Field type	Direction	Required	Comment
Status	Octet	N	M	Completion report value.
ErrorSpecific	Octet-string	N	C	Presence depends on value of Status field. See subclause 6.8.7 for more details. Length shall be in the range 0 to 16 octets.

6.6.20 CongestionLevel

Description: This parameter is used to pass congestion level details to/from the PUF.

Type: 20.

Fields	Field type	Direction	Required	Comment
Level	Octet	B	M	ready (1)
				notready (15)

6.6.21 ConnectedNumber

Description: This parameter is used to pass details concerning the connected number to the PUF.

Type: 21.

Fields	Field type	Direction	Required	Comment
NumberType	Octet	N	M	default (255) - default is unknown
				unknown (0)
				international (1)
				national (2)
				network (3)
				subscriber (4)
				abbreviated (6)
NumberPlan	Octet	N	M	default (255) - default is unknown
				unknown (0)
				isdn (1)
				data (3)
				telex (4)
				national (8)
				private (9)
Number	IA5-string	N	M	20 is maximum length

6.6.22 ConnectedSubaddress

Description: This parameter is used to pass the Connected Subaddress to the PUF.

Type: 22.

Fields	Field type	Direction	Required	Comment
NumberType	Octet	N	M	nsap (0)
				user (1)
Indicator	Octet	N	M	even(0)
				odd (1)
				This field is only meaningful if NumberType is set to user. It indicates if the number contains an odd or even number of BCD digits
Number	IA5-string	N	M	20 is maximum length

6.6.23 DateTime

Description: This parameter is used to pass date and time information to the PUF. This information is provided by the Network.

Type: 23.

Fields	Field type	Direction	Required	Comment
Year	Octet	N	M	0 to 99
Month	Octet	N	M	1 to 12
Day	Octet	N	M	1 to 31
Hour	Octet	N	M	0 to 23
Minute	Octet	N	M	0 to 59

6.6.24 Display

Description: This parameter is used to pass display information to the PUF.

Type: 24.

Fields	Field type	Direction	Required	Comment
Information	IA5-string	N	M	32 is maximum length

6.6.25 ExtEquipAvailability

Description: This parameter is used to pass the information related the availability of the external equipment.

Type: 25.

Fields	Field type	Direction	Required	Comment
Availability	Boolean	N	M	State of the external equipment TRUE - equipment available FALSE - equipment unavailable

6.6.26 ExtEquipBlockDialling

Description: This parameter is used to pass the information related the block dialling made with the keypad of the external equipment.

Type: 26.

Fields	Field type	Direction	Required	Comment
BlockDialling	IA5-string	N	M	Remote address and/or subaddress typed on the keypad of the external equipment. A star ('*') separates address and subaddress fields. 41 is the maximum length.

6.6.27 ExtEquipKeypressed

Description: This parameter is used to pass the information related the pressed keys on the keypad of the external equipment.

Type: 27.

Fields	Field type	Direction	Required	Comment
Keypressed	Octet	N	M	Keypad information (0 to 9) Numeric key. (10) '*' key. (11) '#' key (12) 'R.' key (13) 'Bis' key

6.6.28 ExtEquipName

Description: This parameter is used to pass the name that identifies an item of external equipment.

Type: 28.

Fields	Field type	Direction	Required	Comment
Type	Octet	P	M	type1 (1) - external equipment is of type 1. type2 (2) - external equipment is of type 2. type3 (3) - external equipment is of type 3. type4 (4) - external equipment is of type 4. type5 (5) - external equipment is of type 5. External equipment type are described in Annex C.
Name	IA5-string	P	M	maximum length is 16 "DEFAULT" - use first defined external equipment of specified type

6.6.29 ExpeditedData

Description: This parameter is used to pass use of expedited data information to/from the PUF.

Type: 29.

Fields	Field type	Direction	Required	Comment
Usage	Boolean	B	M	TRUE - Use of expedited data is required or supported. FALSE - Use of expedited data is not required or not supported.

Protocols: This parameter is used in following User Plane protocols:

ETS 300 080 [1]	ISO 8208 [3]	T.70	NULL
	X		

6.6.30 Facility

Description: This parameter is used to pass Facility information to/from the PUF. If another facility information than that defined in the FacilityTag 1 to 4 values is expected, the transparent (5) value should be used.

Type: 30.

Fields	Field type	Direction	Required	Comment
FacilityTag	Octet	B	M	chargingduring (1) - This value is used to request charging information during the connection. This tag is used in the direction from the PUF to the NAF. During the connection the NAF shall send subtotals of the charging information to the PUF. At the end of the connection the NAF shall provide the total charging information. chargingend (2) - This value is used to request charging information at the end of a connection. This tag is used in the direction from the PUF to the NAF. At the end of the connection the NAF shall provide the total charging information to the PUF. charginginfo (3) - This value is used to indicate that the Contents field contains charging related information. This value is used in the direction from NAF to PUF. chargingerror (4) - This value is used to indicate that the Contents field contains Error information related to the charging supplementary service. This value is used in the direction from NAF to PUF. transparent (5) - Allows the PUF or the NAF to send/receive facility information coded in format used by the Network.
FacilityValue	Octet-string		C	Length and contents depend on value of FacilityTag field. Contents field is not allowed when FacilityTag field value is *chargingduring* or *chargingend.*

The coding of the FacilityValue field in the case when the FacilityTag field value is charginginfo, is defined in table 16.

NOTE: For transparent coding, the size of the contents is the size of the facility parameter minus 1.

Table16: Coding of the FacilityValue field in the case of ChargingInfo

Subfield	Field type	Value	Comment
TypeOfTotal	Octet	subtotal (1)	Indicates whether the charging information is a total or a subtotal.
		total (2)	
TypeOfCharge	Octet	currencyinfo (2)	The charging information is represented as currency information.
		unitInfo (3)	The charging information is represented as charging units.
		freeofcharge (4)	The connection is free of charge.
		unknown (1)	The type of the charging information cannot be determined.
Value	Octet	value after decimal point. The value represents n x 1/256.	Value of the charging information in fixed point notation. If the octet 2 indicates freeofcharge, all three octets shall contain zero (0) to indicate a value of 00,0.
	Octet	least significant octet before decimal point	
	Octet	most significant octet before decimal point	

The coding of the Contents field in the case when the Tag field value is ChargingError, is defined in the table 17.

Table 17: Coding of the FacilityValue field in the case of ChargingError

Subfield	Field type	Value	Comment
ChargingError-Cause	Octet	notsubscribed (50)	The user has not subscribed to the advice of charge (AOC) supplementary service.
		notavailable (63)	The advice of charge (AOC) supplementary service is not available.
		notimplemented (69)	The advice of charge (AOC) supplementary service is not implemented.
		InvalidCallState (101)	The advice of charge (AOC) supplementary service is invoked in an invalid call state. The supplementary service can be only be invoked in the CConnectReq.
		NoChargingInfoAvailab le (128)	There is no charging information available.

6.6.31 FacilityData

Description: This parameter is used to pass ISO/IEC 8208 [3] Facility information to/from the PUF.

Type: 31.

Fields	Field type	Direction	Required	Comment
FacilityData	Octet string	B	M	Encoded as facility information defined in ISO 8208 [3]. 109 octets is the maximum length.

Protocols: This parameter is used in following User Plane protocols:

ETS 300 080 [1]	ISO 8208 [3]	T.70	NULL
see NOTE	X		
NOTE: These parameters shall be used in accordance with ETS 300 080 [1].			

6.6.32 FastSelect

Description: This parameter is used to pass ISO/IEC 8208 [3] Fast Select Facility information to/from the PUF.

Type: 32.

Fields	Field type	Direction	Required	Comment
FastSelect	Octet	B	M	norestriction (1) - Called DTE is not required to remove the user connection before establishment is complete. restricted (2) - Called DTE is required to remove the user connection before establishment is complete.

Protocols: This parameter is used in following User Plane protocols:

ETS 300 080 [1]	ISO 8208 [3]	T.70	NULL
X	X		

Remarks: When specified on a U3ConnectReq message this parameter allows the UserData parameter to have a maximum length of 128 octets. If the *restricted* option is selected it indicates that the user connection cannot be established and that a U3DisconnectInd should be expected with a maximum UserData parameter of 128 octets. If the *nonrestricted* option is specified then the user connection can be established and the subsequent U3ConnectCnf can have a maximum UserData parameter of 128 octets. Subsequent to this both the U3DisconnectInd and U3DisconnectReq fields may also have maximum UserData parameters of 128 octets.

When received on a U3ConnectInd message this parameter indicates that the UserData parameter with the message can have a maximum length of 128 octets. If the *restricted* option is selected it indicates that the user connection cannot be established and that the PUF must respond with U3DisconnectReq. The UserData parameter with this message can have a maximum length of 128 octets. If the *nonrestricted* option is selected, the PUF can respond with U3ConnectRsp with a maximum UserData parameter of 128 octets Subsequent to this both the U3DisconnectInd and U3DisconnectReq fields may also have maximum UserData parameters of 128 octets.

6.6.33 GroupID

Description: This parameter is used to pass the group identifier to/from the PUF.

Type: 33.

Fields	Field type	Direction	Required	Comment
GroupID	Octet string	B	M	The value is unique for a PUF/NAF relation. 4 is the fixed length.

Protocols: This parameter is used in following User Plane protocols:

ETS 300 080 [1]	ISO 8208 [3]	T.70	NULL
X	X		

6.6.34 High Layer Compatibility (HLC)

Description: This parameter is used to pass High Layer Compatibility (HLC) information to/from the PUF.

Type: 34.

Fields	Field type	Direction	Required	Comment
Standard	Octet	B	M	Default (255) - default is ccitt
				ccitt (0)
				international (1)
				national (2)
				network (3)
Identification	Octet	B	M	telephony (1)
				faxG4C1 (33)
				teletexF184 (36)
				teletexF220 (40)
				teletexF200 (49)
				videotext (50)
				telex (53)
				mhsx400 (56)
				osix200 (65)
				maintenance (94)
				management (95)
ExtIdentification	Octet	B	O	telephony (1)
				faxG4C1 (33)
				teletexF184 (36)
				teletexF220 (40)
				teletexF200 (49)
				videotext (50)
				telex (53)
				mhsx400 (56)
				osix200 (65)

6.6.35 IdleFlag

Description: Flag byte to be sent by the NAF when TMA is idle.

Type: 35.

Fields	Field type	Direction	Required	Comment
IdleFlag	Octet	P	M	Flag byte.

6.6.36 Key

Description: Key to be used for the security algorithm.

Type: 36.

Fields	Field type	Direction	Required	Comment
Key	Octet-string	P	M	The Key parameter is used by the PUF to give relevant information for the security algorithm to the NAF. Maximum length is 255.

6.6.37 Keypad

Description: This parameter is used to pass keypad facility information to the NAF.

Type: 37.

Fields	Field type	Direction	Required	Comment
Keypad	Octet-string	P	M	AI5 characters to convey. Maximum length is 32.

6.6.38 L2ConnectionMode

Description: This parameter is used to pass details of the layer connection mode to the NAF.

Type: 38.

Fields	Field type	Direction	Required	Comment
Value	Octet	P	M	dte (1) - Act as DTE as defined in ISO 7776 [4] dce (2) - Act as DCE as defined in ISO 7776 [4] auto (3) - When calling act as DTE, when called act as DCE

Protocols: This parameter is used in following User Plane protocols:

ETS 300 080 [1]	ISO 8208 [3]	T.70	NULL
	X		

6.6.39 L2FrameSize

Description: This is used to pass details of the layer 2 frame size to the NAF.

Type: 39.

Fields	Field type	Direction	Required	Comment
Value	Octet string	P	M	Frame size (in octets). Length is fixed to 2. The first octet contains the most significant byte.

6.6.40 L2WindowSize

Description: This is used to pass details of the layer 2 window size to the NAF.

Type: 40.

Fields	Field type	Direction	Required	Comment
Value	Octet	P	M	Window size

Protocols: This parameter is used in following User Plane protocols:

ETS 300 080 [1]	ISO 8208 [3]	T.70	NULL
	X		

6.6.41 L2XID

Description: This is used to pass details of the layer 2 XID value and its use.

Type: 41.

Fields	Field type	Direction	Required	Comment
Use	Octet	P	M	send (1) - send XID. match (2) - match XID with XID received. IF XID does not match, connection shall not be established.
Value	Octet-string	P	M	XID value [Identifier and signature]. Maximum length is 64.

Protocols: This parameter is used in following User Plane protocols:

ETS 300 080 [1]	ISO 8208 [3]	T.70	NULL
	X		

6.6.42 L3ConnectionMode

Description: This parameter is used to pass details of the layer connection mode to the NAF.

Type: 42.

Fields	Field type	Direction	Required	Comment
Value	Octet	P	M	dte (1) - act as DTE. dce (2)- act as DCE. auto (3)- act as DTE when calling, act as DCE when called. dxe (4)- use Restart Packet to determine DTE or DCE role as in ISO 8208auto.

Protocols: This parameter is used in following User Plane protocols:

ETS 300 080 [1]	ISO 8208 [3]	T.70	NULL
	X		
NOTE: When using the User Plane protocol **ETS 300 080 [1]**, dxe(4) shall always be used.			

6.6.43 L3IncomingVCCount

Description: This parameter is used to pass the number of connections that may be established at any instant by incoming call establishment requests.

Type: 43.

Fields	Field type	Direction	Required	Comment
Value	Octet-string	P	M	Number of connections. Maximum value is 4095. Length is fixed to 2.

Protocols: This parameter is used in following User Plane protocols:

ETS 300 080 [1]	ISO 8208 [3]	T.70	NULL
	X		

6.6.44 L3OutgoingVCCount

Description: This parameter is used to pass the number of connections that may be established at any instant by outgoing call establishment requests.

Type: 44.

Fields	Field type	Direction	Required	Comment
Value	Octet-string	P	M	Number of connections. Maximum value is 4095. Length is fixed to 2.

Protocols: This parameter is used in following User Plane protocols:

ETS 300 080 [1]	ISO 8208 [3]	T.70	NULL
	X		

6.6.45 L3TwoWayVCCount

Description: This parameter is used to pass the number of connections that may be established at any instant by outgoing or incoming connection establishment requests.

Type: 45.

Fields	Field type	Direction	Required	Comment
Value	Octet-string	P	M	Number of connections. Maximum value is 4095. Length is fixed to 2.

Protocols: This parameter is used in following User Plane protocols:

ETS 300 080 [1]	ISO 8208 [3]	T.70	NULL
X	X		

6.6.46 Low Layer Compatibility (LLC)

Description: This parameter is used to pass Low Layer Compatibility (LLC) information to/from the PUF. Information concerning layer 1 details shall be taken from the BearerCap parameter when an CConnectReq message is issued with an LLC parameter.

Type: 46.

Fields	Field type	Required	Comment
Negotiation	Boolean	M	TRUE - negotiation is allowed
			FALSE - negotiation is not allowed
Layer2protocol	Octet	M	0 - 31
			255 = unspecified
layer2optional	Octet	M	0 - 127
			255 = unspecified
Layer3protocol	Octet	M	0 - 31
			255 = unspecified
layer3optional	Octet	M	0 - 127
			255 = unspecified

6.6.47 ManufacturerCode

Description: This parameter identifies the manufacturer. It is provided by the manufacturer itself.

Type: 47.

Fields	Field type	Direction	Required	Comment
Value	Octet-string	B	M	Manufacturer identification. Maximum length is 255.

6.6.48 MoreData

Description: This parameter is used to indicate that an other user-to-user information message will follow, belonging to the same block.

Type: 48.

Fields	Field type	Direction	Required	Comment
Value	Octet	B	M	Fixed value: TRUE

6.6.49 NCOID

Description: This parameter is used to pass the connection object identifier to/from the PUF.

Type: 49.

Fields	Field type	Direction	Required	Comment
Value	Octet-string		M	This value is unique for a PUF/NAF relation. Length is fixed to 4.

6.6.50 NCOType

Description: This parameter is used to pass the connection object type to the NAF.

Type: 50.

Fields	Field type	Direction	Required	Comment
Identifier	Octet		M	cset (1) - signalling access only. u3set (2) - network user access via NMA with NAF signalling co-ordination (NAF co-ordination functionality). s1u1set (4) - signalling and transparent user access via TMA. s1u3set (5) - signalling and network layer user access via NMA. u3group (6) - network user access via NMA to additional virtual circuits. This NCO must be grouped to an already created U3 or C/U3 type NCO.

6.6.51 NotificationIndicator

Description: This parameter is used to pass notification of network event to the PUF. It may be a suspended or resumed operation.

Type: 51.

Fields	Field type	Direction	Required	Comment
Value	Octet	N	M	suspended (1) resumed (2) callwaiting (3)

6.6.52 PacketSize

Description: This parameter is used to pass ISO/IEC 8208 [3] packet size information to/from the PUF.

Type: 52.

Fields	Field type	Direction	Required	Comment
Negotiation	Boolean	B	M	Used to indicate if negotiation of packet size is possible. TRUE - negotiation possible. FALSE - negotiation not possible.
Invalue	Octet	B	M	Inbound maximum user data length (see table 18). Maximum size of data that can be received with U3DataInd.
Outvalue	Octet	B	M	Outbound maximum user data length (see table 18). Maximum size of data that can be passed with U3DataReq.

Protocols: This parameter is used in following User Plane protocols:

ETS 300 080 [1]	ISO 8208 [3]	T.70	NULL
	X		

Remarks: This parameter is used to determine the maximum size of data buffers that can be passed with the U3DataReq and U3DataInd messages. It is used as follows:

- on U3ConnectReq the PUF may specify the values it wishes to use;
- on U3ConnectCnf the NAF shall always specify the values to be used for the user connection;
- on U3ConnectInd the NAF shall always indicate the values to be used for the user connection. It also indicates if it is possible for the PUF to negotiate these values;
- on U3ConnectRsp the PUF can specify values if the U3ConnectInd indicated that negotiation was possible.

Table 18: Precoded packet size values

Precoded value	Packet size (octet)	Precoded value	Packet size (octet)
4	16	9	512
5	32	10	1 024
6	64	11	2 048
7	128	12	4 096
8	256		

6.6.53 ProgressIndicator

Description: This parameter is used to pass information concerning the progress of a telephony call to the PUF.

Type: 53.

Fields	Field type	Direction	Required	Comment
Standard	Octet	N	M	ccitt (0)
				international (1)
				national (2)
				network (3)
Location	Octet	N	M	user (0)
				privatelocal (1)
				publiclocal (2)
				transit (3)
				publicremote (4)
				privateremote (5)
				international (7)
				networkbeyond (10)
Value	Octet	N	M	notISDN (1) - call is not end to end ISDN, further information may be available in-band.
				destinationnotISDN (2) - Destination address is not ISDN.
				originationnotISDN (3) - Origination address is not ISDN.
				returnedtoISDN (4) - Call has returned to ISDN.
				inbandinformation (8) - In-band information or appropriate pattern now available.

6.6.54 QOSParameters

Description: This parameter is used to pass Quality of Service information to/from the PUF.

Type: 54.

Fields		**Field type**	**Direction**	**Required**	**Comment**
Throughput	Usage	Boolean	B	M	Indicates if following values are included.
	InTarget	Octet	B	C	Values provided in the table 19.
	InLowest	Octet	B	C	Values provided in the table 19.
	InAvailable	Octet	B	C	Values provided in the table 19.
	InSelected	Octet	B	C	Values provided in the table 19.
	OutTarget	Octet	B	C	Values provided in the table 19.
	OutLowest	Octet	B	C	Values provided in the table 19.
	OutAvailable	Octet	B	C	Values provided in the table 19.
	OutSelected	Octet	B	C	Values provided in the table 19.
NCPriority	Usage	Boolean	B	M	Indicates if following values are included.
	Target	Octet	B	C	(NOTE 1).
	Lowest	Octet	B	C	(NOTE 1).
	Available	Octet	B	C	(NOTE 1).
	Selected	Octet	B	C	(NOTE 1).
TransitDelay	Usage	Boolean	B	M	Indicates if following values are included.
	Selected	Octet-string	B	C	(NOTE 2).
	Target	Octet-string	B	C	(NOTE 2).
	Maximum	Octet-string	B	C	(NOTE 2). Conditional if Target - previous one - used else absent.
End to End Transit Delay	Usage	Boolean	B	M	Indicates if following values are included.
	Selected	Octet-string	B	C	(NOTE 2).
	Target	Octet-string	B	C	(NOTE 2).
	Maximum	Octet-string	B	C	(NOTE 2). Conditional if Target - previous one - used else absent.

NOTE 1:	The NCPriority fields can take any value from 1 (highest priority) to 10 (lowest priority). If not used, the field shall be filled with the value 0. If unspecified, the field shall be filled with the value 11.
NOTE 2:	Length is fixed to 2. The lower octet contains least significant byte. 65535 means not used. Delay expressed in milliseconds.

Protocols: This parameter is used in following User Plane protocols:

ETS 300 080 [1]	ISO 8208 [3]	T.70	NULL
X	X		

Table 19: Throughput precoding value

Precoding value	Throughput class	Precoding value	Throughput class
3	75	9	4 800
4	150	10	9 600
5	300	11	19 200
6	600	12	48 000
7	1 200	13	64 000
8	2 400	0	unused

6.6.55 ReadyFlag

Description: This parameter is used to request and indicate flow control status on a user connection.

Type: 55.

Fields	Field type	Direction	Required	Comment
Usage	Boolean	B	M	TRUE - Data transfer is possible. FALSE - Data transfer is not possible.

Protocols: This parameter is used in following User Plane protocols:

ETS 300 080 [1]	ISO 8208 [3]	T.70	NULL
X	X		

6.6.56 RequestID

Description: This parameter is used to pass an identifier to the NAF on a request message. It is returned by the NAF on the associated confirm message.

Type: 56.

Fields	Field type	Direction	Required	Comment
Identifier	Octet-string	B	M	Internal ID provided by the PUF. Length is fixed to 4.

6.6.57 ReceiptConfirm

Description: This parameter is used to request confirmation of data receipt for a User Plane connexion.

Type: 57.

Fields	Field type	Direction	Required	Comment
Value	Boolean	B	M	TRUE - Confirmation requested FALSE - Confirmation not requested

6.6.58 RespondingDTEAddress

Description: This parameter is used to pass responding DTE address information to/from the PUF.

Type: 58.

Fields	Field type	Direction	Required	Comment
Address	IA5-string		M	16 octets is the maximum length.

6.6.59 RespondingDTEAddressExt

Description: This parameter is used to pass responding DTE address extension information from/to the PUF.

Type: 59.

Fields	Field type	Direction	Required	Comment
AddressExt	IA5-string	B	M	40 octets is the maximum length

Protocols: This parameter is used in following User Plane protocols:

ETS 300 080 [1]	ISO 8208 [3]	T.70	NULL
X	X		

6.6.60 SelectorID

Description: This parameter is used by the PUF to select the right NCO on an incoming call (second step of the selection). Also the PUF uses the SelectorID to give the NAF a list of NCOs that should be exclusively dealt with.

Type: 60.

Fields	Field type	Direction	Required	Comment
Identifier	Octet-string	B	M	Internal ID provided by the PUF. Length is fixed to 4.

6.6.61 TEI

Description: This parameter is used to access a permanent link to a data packet switch (packet connection in D-Channel).

Type: 61.

Fields	Field type	Direction	Required	Comment
Value	Octet	B	M	

6.6.62 U3Protocol

Description: This is used to select the User Plane protocol.

Type: 62.

Fields	Field type	Direction	Required	Comment
L3Protocol	Octet	P	M	Default (255) - ETS 300 080 [1]
				ETS 300 080 (1)
				ISO 8208 (2)
				T.70 NL (3)
				NULL (4)
L2Protocol	Octet	P	C	Only allowed if L3Protocol is NULL(4).
				Default (255) - X.75
				X.75 (0)
				frame oriented transparent (1)

6.6.63 UAttributeName

Description: This parameter is used to pass the name of a static set of user plane attributes from the PUF.

Type: 63.

Fields	Field type	Direction	Required	Comment
AttributeName	IA5-string	P	M	16 is maximum length.

6.6.64 UDirection

Description: This parameter is used to pass information concerning the usage of a particular NCO to the NAF, for the User Plane.

Type: 64.

Fields	Field type	Direction	Required	Comment
Direction	Octet	P	M	listen (1)
				call (2)
				both (3)

6.6.65 UserData

Description: This parameter is used to pass Data that is limited in size to/from the PUF.

Type: 65.

Fields	Field type	Direction	Required	Comment
Data	Octet-string	B	M	128 octets is the maximum size. The maximum length allowed varies from message to message and is also different dependent on the use of the FastSelect parameter.

Protocols: This parameter is used in following user plane protocols:

ETS 300 080 [1]	ISO 8208 [3]	T.70	NULL
X	X		

6.6.66 UserToUserInfo

Description: This parameter is used to pass user-to-user information to/from the PUF.

Type: 66.

Fields	Field type	Direction	Required	Comment
Discriminator	Octet	B	M	userspecific (0) - contents of information field is in user specific format. ia5chars (4) - contents of information field is IA5 characters.
Information	Octet-string	B	M	128 is maximum size.

Remarks: The Discriminator field is used to indicate the format of the data in the Information field. Values from 0 to 256 are possible but may be restricted by the ISDN being accessed. The values defined are supported by all NAFs.

6.6.67 WindowSize

Description: This parameter is used to pass ISO/IEC 8208 [3] window size information to/from the PUF.

Type: 67.

Fields	Field type	Direction	Required	Comment
Negotiation	Boolean	B	M	Used to indicate if negotiation of window size is possible. TRUE - negotiation possible. FALSE - negotiation not possible.
Invalue	Octet	B	M	Inbound window size.
Outvalue	Octet	B	M	Outbound window size.

Protocols: This parameter is used in following User Plane protocols:

ETS 300 080 [1]	ISO 8208 [3]	T.70	NULL
X	X		

Remarks: This parameter is used to determine the window sizes to be used for a user connection.

- On U3ConnectReq, the PUF may specify the values it wishes to use.
- On U3ConnectCnf, the NAF shall always specify the values to be used for the user connection.
- On U3ConnectInd the NAF shall always indicate the values to be used for the user connection. It also indicates if it is possible for the PUF to negotiate these values.
- On U3ConnectRsp the PUF can specify values if the U3ConnectInd indicated that negotiation was possible.

6.6.68 X213Cause

Description: This parameter is used to pass X213 Cause information to/from the PUF.

Type: 68.

Fields	Field type	Direction	Required	Comment
Value	Octet	B	M	See User Plane return code values in subclause 6.8.9.

6.6.69 X213Origin

Description: This parameter is used to pass X213 origin information to/from the PUF.

Type: 69.

Fields	Field type	Direction	Required	Comment
Value	Octet	B	M	undefined (1) NAF Provider (2) PUF User (3)

6.6.70 X25Cause

Description: This parameter is used to pass X25 Cause information to/from the PUF.

Type: 70.

Fields	Field type	Direction	Required	Comment
Value	Octet	B	M	See ISO 8208 [3] cause code values.

Protocols: This parameter is used in following User Plane protocols:

ETS 300 080 [1]	ISO 8208 [3]	T.70	NULL
X	X		

6.6.71 X25Diagnostic

Description: This parameter is used to pass X25 Diagnostic information to/from the PUF.

Type: 71.

Fields	Field type	Direction	Required	Comment
Value	Octet	B	M	See ISO 8208 diagnostic values.

Protocols: This parameter is used in following User Plane protocols:

ETS 300 080 [1]	ISO 8208 [3]	T.70	NULL
X	X		

6.6.72 AttributeSet Parameters

AttributeSet parameters depend on the type of parameters to provide with the ACreateNCO request. Tables 20, 21 and 22 show the content of such parameters.

Table 20: Signalling Attribute Set (CAttributeSet) Parameters

Parameters	Required	Comment
ChannelIdentification	O*	See subclause 6.6.17.
HLC	O	See subclause 6.6.34.
LLC	O	See subclause 6.6.46.
BearerCap	O	See subclause 6.6.3.

Table 21: TMA related User Plane Attribute Set (UAttributeSet) Parameters

Parameters	Required	Comment
IdleFlag	C	Flag byte to be sent while idle. See subclause 6.6.35.
ExtEquipName	O	Name of external equipment to be used. If provided connection shall be established to identified external equipment and TMA messages shall not be provided. See subclause 6.6.28.

Remark: IdleFlag parameter is not allowed if ExtEquipName parameter is provided.

Table 22: NMA related User Plane Attribute Set (UAttributeSet) Parameters

Parameters	Required	Comment
WindowSize	O	Layer 3 window size. See subclause 6.6.67.
PacketSize	O	Layer 3 packet size. See subclause 6.6.52.
FastSelect	O	Fast select facility. See subclause 6.6.32.
QOSParameters	O	Quality of service. See subclause 6.6.54.
U3Protocol	O	See remark. See also subclause 6.6.62.
L3ConnectionMode	O	See remark. See also subclause 6.6.42.
L3TwoWayVCCount	O	See remark. See also subclause 6.6.45.
L3IncomingVCCount	O	See remark. See also subclause 6.6.43.
L3OutgoingVCCount	O	See remark. See also subclause 6.6.44.
TEI	O	See remark. See also subclause 6.6.61.
L2ConnectionMode	O	See remark. See also subclause 6.6.38.
L2WindowSize	O	See remark. See also subclause 6.6.40.
L2FrameSize	O	See remark. See also subclause 6.6.39.
L2XID	O	See remark. See also subclause 6.6.41.

Remark: It is only possible to use these parameters during NCO creation containing Control plane information. NCOs that are to be associated by the use of a GroupID may not specify these parameters. Refer to subclause 6.2.1 - ACreateNCO operation - for details.

If parameters are omitted defaults shall be used. The default values are User Plane protocol dependent. The NAF shall supply the correct value depending on the protocol. In the case of ISO/IEC 8208 [3], local configuration as described in Annex E, Clause E.8 may be used.

6.6.73 Administration AttributeSet Parameters

Administration AttributeSet parameters are used to collect some management information about each NCO and are accessible at any time through the GetNCOInfo operation. Table 23 shows the content of this parameter.

Table 23: Administration Attribute Set Parameters

Parameters	Required	Comment
NCOType	O	See subclause 6.6.50.
CDirection	O	See subclause 6.6.16.
CAttributeName	O	See subclause 6.6.13.
CAttribute parameter	O	See table 20.
UDirection	O	See subclause 6.6.64.
UAttributeName	O	See subclause 6.6.63.
UAttribute parameter	O	See table 22.
CAddress parameters	O	See table 24.
UAddress parameters	O	See table 25.
GroupID	O	Providing at the NCO creation time. See subclause 6.6.33.
SelectorID	O	See subclause 6.6.60.
ChargingInfo	O	See subclause 6.6.18.
DateTime	O	Date and Time of the NCO creation. See subclause 6.6.23.
CauseToPUF	O	See subclause 6.6.15.

6.6.74 AddressSet Parameter

Tables 24 and 25 show the structures of the Address.

Table 24: Signalling Address Set (CAddressSet) Parameters

Parameters	Required	Comment
CalledNumber	O	See subclause 6.6.7 for parameter definition.
CalledSubaddress	O	See subclause 6.6.8 for parameter definition.
CallingNumber	O	See subclause 6.6.11 for parameter definition.
CallingSubaddress	O	See subclause 6.6.12 for parameter definition.

Table 25: NMA related User Plane Address Set (UAddressSet) Parameters

Parameters	Required	Comment
CalledDTEAddress	O	See subclause 6.6.5 for parameter definition.
CalledDTEAddressExt	O	See subclause 6.6.5 for parameter definition.
CallingDTEAddress	O	See subclause 6.6.9 for parameter definition.
CallingDTEAddressExt	O	See subclause 6.6.10 for parameter definition.

6.7 Selection criteria

6.7.1 NCO selection

In order to apply the right NCO on an incoming call, the following considerations have to be taken into account by the NAF.

Only the NCOs with UDirection or CDirection set to incoming or both directions are dealt with in this case. The best NCO shall contain an explicit definition for each value used for the checking. The "match" level shall be put on values checked rather than on values assumed. If an information element used as criteria is not provided by the network this criteria shall be ignored by the NAF during the matching operation. Table 26 summarises the matching operation.

Table 26: Matching operation for the NCO selection

Network	NCO	Operation	Result
Provided	Provided	Equal	**Match**
Provided	Provided	Not equal	**No match**
Provided	Not provided	(no operation)	**Match**
Not provided	Provided	(no operation)	**No match**
Not provided	Provided	(no operation)	**No match**
Not provided	Not provided	(no operation)	**Match**

The NAF shall broadcast an incoming call to all PUFs, which have indicated compatibility within an NCO. The incoming call shall then be finally assigned to the PUF which first accepts the call with the appropriate message. All other PUFs shall receive a disconnect indication. Using this procedure also implies that NAF co-ordinated NCOs have a higher priority than PUF co-ordinated NCOs, since the NAF may respond immediately to an incoming call without involving any PUF. In such a case, the call is not seen from non-co-ordinated NCOs.

If CAlertReq message is sent by a PUF, only the first shall be sent to the Network. All others shall be ignored. When a CDisconnectReq message is sent by a PUF, it shall not disconnect the call except if no other NCO has been assigned to this call. This mechanism gives the opportunity to make connection with a delayed NCO.

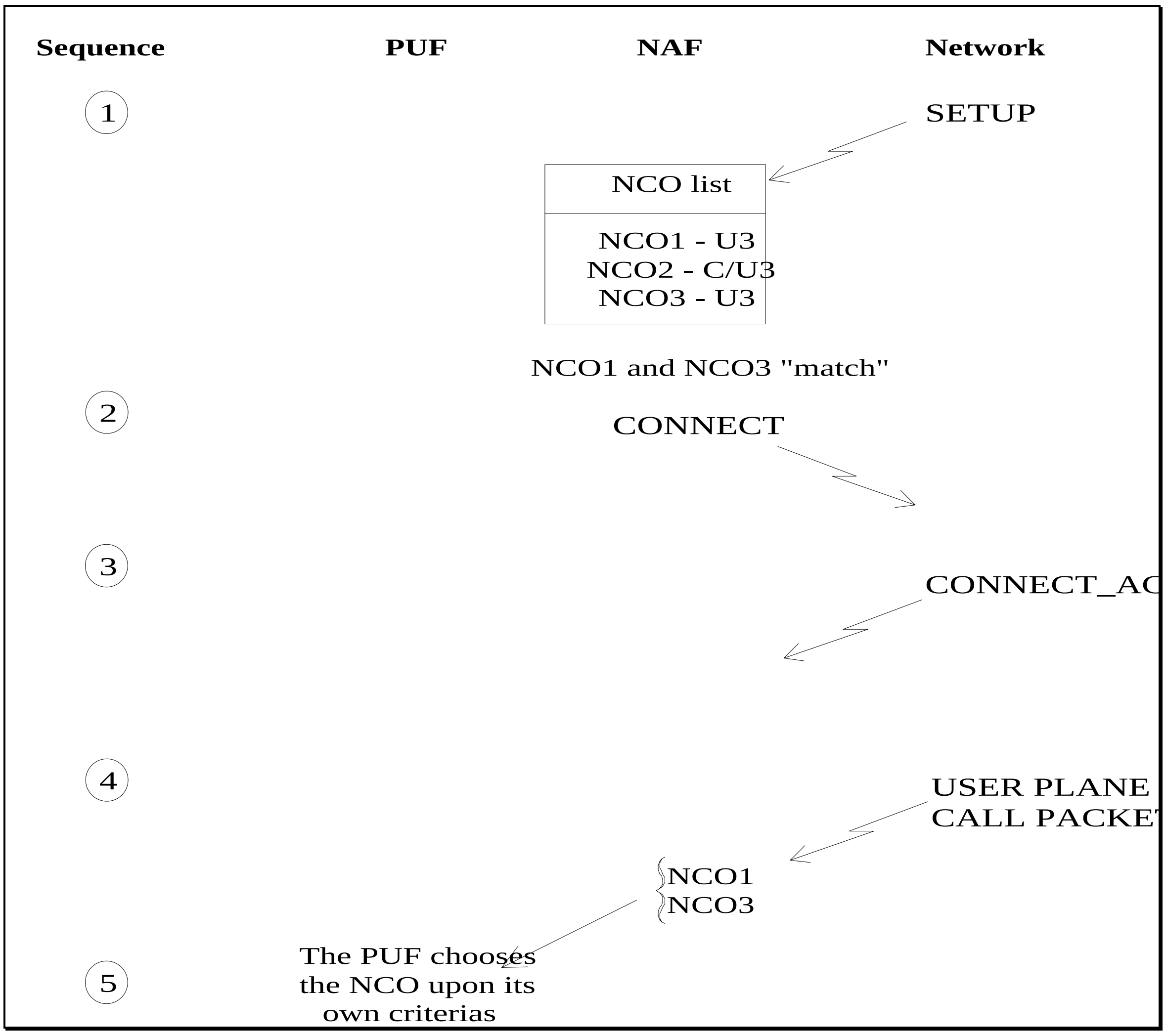


Figure 14: NCO selection procedure for incoming call

For more information on selection, refer to ETR 018 [9].

6.7.1.1 Control Plane information elements

1) Called Address (correct or absent);
2) Called Subaddress (correct or absent);
3) Bearer capabilities (correct);
4) LLC (see note) (correct or absent);
5) HLC (correct or absent).

These five information elements shall match the NCO values to make an NCO eligible. At the end of the selection process, if more than one NCO are eligible, the second step selection shall apply. First the check function, associated with the order of the information elements, shall be used to select an NCO. The latest selection criteria shall be the time. The latest NCO created by the NAF shall be selected first.

EXAMPLE: In the case presented in the table 27, the NCO2 shall be chosen because the NCO2 address information element matches exactly the address information element provided in the incoming call.

Table 27: Matching NCO on an incoming call

Field	Incoming call	NCO1	NCO2
Called address	123456789	not provided	123456789
Called sub address	1002	1002	1002
Bearer capability	speech	speech	speech
LLC	no outband negotiation	not provided	not provided
HLC	telephony	telephony	telephony

6.7.1.2 User Plane information element (layer 3)

To select an NCO, the NAF uses the following parameters:

- packet size negotiation;
- window size negotiation.

6.7.1.2.1 Packet size negotiation

In the INCOMING CALL packet, if the packet size is not provided, the default value, i.e. 128 shall be assumed.

The NCO packet size is correct if one of the following cases is relevant:

- the packet size - provided in the U3AttributeSet - is equal to the packet size provided in the INCOMING CALL packet or assumed;
- if there is no packet size provided in the U3AttributeSet.

6.7.1.2.2 Window size negotiation

In the INCOMING CALL packet, if the Window size is not provided, the default value, i.e. 2 shall be assumed.

The NCO window size is correct if one of the following cases is relevant:

- the window size - provided in the U3AttributeSet - is equal to the window size provided in the INCOMING CALL packet or assumed;
- if there is no window size provided in the U3AttributeSet.

6.7.1.2.3 Effective packet size and window size negotiation

In the U3ConnectRsp, if packet size is not provided, the packet size provided in the incoming call - i.e. U3ConnectInd - shall be accepted by the PUF. The same rules apply to the window size.

In the U3ConnectCnf, if the packet size/window size is not provided, the packet size/window size provided during the outgoing call - i.e. U3ConnectReq - shall be approved for use by the PUF.

6.7.2 Action if no NCO available

6.7.2.1 Control Plane incoming call

A disconnection cause #88 "incompatible destination" shall be issued by the NAF.

6.7.2.2 User Plane incoming call

A disconnect with the X213reason "Connection rejection - reason unspecified transient" shall be issued by the NAF.

6.8 Error checking and codes

This subclause deals with the error checking provided by the ISDN PCI. Initially, the error checking methods employed by each plane are described. Then the function return codes and error return codes for each plane are defined and described.

6.8.1 Administration Plane

For Administration Plane messages, almost all messages operate in Request/Confirm pairs; there are no Indicate/Response messages. Any error detected in a request message shall be notified in the related confirm message.

For Administration Plane messages any error detected shall prevent an operation being performed and hence prevent a change of state.

Within the Administration Plane the AErrorInd message is used to indicate errors which are not covered by the protocols which support the Control Plane and User Plane messages. For example, this message is used to inform the PUF that an invalid NCOID has been specified on a message.

6.8.2 Control Plane

When Mandatory parameters are missing, or a content error occurs in a mandatory parameter, or a parameter is unrecognised, the NAF indicates the error to the PUF as given in subclauses 6.8.2.1 to 6.8.2.3.

6.8.2.1 Invalid state for message

CStatusInd, no change of state for connection.

6.8.2.2 Mandatory parameters

In case of mandatory parameters missing, mandatory parameters content error or unrecognised parameter, the NAF shall indicate this error to the PUF as follows:

- for CConnectReq the PUF is sent an CDisconnectInd;
- for CDisconnectReq the PUF is sent an CDisconnectCnf;
- for any other message the PUF is sent CStatusInd, no operation is performed, and no change of state occurs.

6.8.2.3 Optional Parameter Content Error

The message shall be processed as if the parameter were not present, CStatusInd is sent to the PUF indicating the parameter in error.

6.8.3 Errors in facility requests

Errors related to facility requests depend on the facility being requested. In the case of the Advice of Charge supplementary service, errors are indicated by the use of a CFacilityInd message. The message that generated this error is processed as if there were no facility information present. Specific errors are defined in subclause 6.6.30.

When a PUF uses facilities in the transparent form, it shall be up to the PUF to understand how errors shall be reported and what processing may have occurred within the network.

6.8.4 User Plane (NMA)

Errors are dealt with in the following manner:

6.8.4.1 Invalid Use of Receipt Confirmation Service

- PUF is sent U3DisconnectInd.

6.8.4.2 Invalid Use of Confirmation request on U3DataReq

- PUF is sent U3DisconnectInd.

6.8.4.3 Invalid length of U3DataReq UserData parameter

- PUF is sent U3DisconnectInd.

6.8.4.4 Invalid Use of Expedited Data

- PUF is sent U3DisconnectInd.

6.8.4.5 Invalid Issuing of messages while in Reset state

- PUF is sent U3DisconnectInd.

6.8.4.6 Invalid Use of Bit_DQM (association between More and Qualifier bits) parameters on subsequent U3DataReq messages

- PUF is sent U3ResetInd.

6.8.4.7 Other errors

In case of:

- Mandatory Parameters Missing;
- Mandatory Parameter Content Error;
- Invalid NCO state;
- Unrecognised Parameter;
- Optional Parameter Content Error.

action is:

- for U3ConnectReq, PUF is sent U3DisconnectInd;
- for U3ConnectRsp, PUF is sent U3DisconnectInd;
- for U3DisconnectReq, disconnect virtual circuit, PUF is sent U3DisconnectInd;
- for other Request/Response Messages, PUF is sent U3DisconnectInd.

6.8.5 TMA User Plane

Errors are dealt withas given in subclauses 6.8.5.1 to 6.8.5.4.

6.8.5.1 Mandatory Parameters Missing

- U1ErrorInd with CompletionStatus, MissingParameter.

6.8.5.2 Mandatory Parameter Content Error

- Only mandatory parameter is NCOID, if parameter length is incorrect then U1ErrorInd with cause InvalidParameterLength, if NCOID is invalid then U1ErrorInd with cause InvalidNCOID.

6.8.5.3 Unrecognised Parameter

- U1ErrorInd with CompletionStatus, InvalidParameter.

6.8.5.4 Overflow of Incoming Data

- U1ErrorInd with CompletionStatus, Overflow.

6.8.6 Function Return Codes

Table 28 defines function return codes:

Table 28: Function return codes

Return Code		Meaning
Success	0	Function completed successfully.
QueryEntityNotAvailable	128	The Query entity is not available or an error occurs during dialogue between the PUF and the Query entity.
InvalidSignalNumber	129	The signal number specified is invalid.
InvalidPCIHandle	130	Handle does not identify a NAF.
NAFnotAvailable	255	NAF is no longer available. The NAF has terminated due to error. This is a permanent condition.
NAFBusy	132	NAF is unable, currently, to process this request (lack of resource or other reason). The function may work correctly if re-used at a later time. This is a temporary condition.
MaxPUFsExceeded	133	NAF can support no more PUFs.
InvalidPUFType	134	Invalid or unsupported type of PUF. NAF does not support this type of PUF.
InvalidPCIVersion	135	Invalid or unsupported version of PCI. NAF does not support this version of PCI.
InvalidExID	136	NAF does not recognise Exchange identifier.
InvalidPCIMPB	137	PCI Message Parameter Block address is incorrect.
InvalidMessageBuffer	138	Message Buffer address is invalid.
InvalidDataBuffer	139	Data Buffer address is invalid.
PCIMPBBufferTooSmall	140	PCIMPB Buffer is too small. Provided for operating systems that can check length of available memory.
MessageBufferTooSmall	141	Message Buffer is too small. Message Buffer does not meet message identifier requirements or actual buffer size in PCIMPB is greater than maximum size. On some operating systems this may also indicate that maximum size of data buffer exceeds memory limitations.
DataBufferRequired	142	Data Buffer is required for message.
DataBufferTooSmall	143	Data Buffer provided for message is too small. Data Buffer does not meet message identifier requirements or actual buffer size in PCIMPB is greater than maximum size. On some operating systems this may also indicate that maximum size of data buffer exceeds memory limitations.
PropertyBufferTooSmall	144	The buffer provided with the property information structure(s) is too small.
MessageTooLarge	145	There is no upper bound to the message size because of repetitions of parameters. If the message size exceeds the maximum size possible with an implementation this value is returned.
InvalidHandlesBuffer	146	The PCIHandles buffer address is invalid.

(continued)

Table 28: Function return codes (concluded)

HandlesBufferTooSmall	147	The size of the buffer for PCIHandles is too small to contain all available PCI_HANDLEs.
BufferTooSmall	148	The size of the buffer provided by the PUF is too small to answer the NAF needs (Operating System specific return code).
InvalidRegisterInfoStructure	149	At least one parameter contained in the PCIRegisterInfo structure is invalid (Operating System specific return code).
InvalidOpSysInfoStructure	150	At least one parameter contained in the PCIOpSysInfo structure is invalid (Operating System specific return code).

6.8.7 Administration Plane return code

The following values are returned in the CompletionStatus parameter The ErrorSpecific information column indicates what, if any, information shall be in the ErrorSpecific field:

Table 29: Administration plan return code

Return Code		Meaning	ErrorSpecific Information
Success	0	Operation completed successfully.	Not present
NAFnotAvailable	255	NAF is no longer available. The NAF has terminated due to error. This is a permanent condition.	Not present
RessourceNotAvailable	47	Used with the NCO creation request message to indicate the lack of a ressource (e.g. memory).	Not present
UndefinedMsgType	95	This message identifier is not defined by the ISDN PCI.	Message Identifier
UnsupportedMsgType	97	This message identifier is defined by the ISDN PCI but not supported by this NAF.	Message Identifier
InvalidParameter	99	A parameter is not recognised or is not supported by a message.	Parameter Type

(continued)

Table 29: Administration plan return code (concluded)

Return Code		Meaning	ErrorSpecific Information
MissingParameter	96	A mandatory parameter is missing from a message.	Parameter Type
InvalidParameterLength	182	A parameter's length is outside the allowed range for the parameter.	Parameter Type
InvalidContents	100	A parameter's content is invalid. Used with the NCO creation confirm message to report errors within parameters used to define the NCO.	Parameter Type
InvalidNCOID	81	A message has been passed to the NAF with an invalid NCOID.	NCOID value
NCOIDinUse	183	An NCOID that is in use for an established/establishing connection cannot be used on this message.	NCOID value
InvalidNCOType	184	A message has been passed to the NAF with an invalid NCOType value.	NCOType value
InvalidDirectionType	185	A message has been passed to the NAF with an invalid Direction value.	Not present
AttributeNameError	186	Invalid use of Attribute name. Name is not known, already defined or identifies an attribute set of the wrong type.	Attribute name
ExtraSetError	189	Message contains attribute set name that is not required.	Attribute name
SecurityNotActivated	190	Requested security algorithm has not been activated.	Security algorithm specific value
InvalidCoordValue	191	Invalid value in NAFCoordination parameter.	Not present
InvalidGroupID	192	GroupID value is not recognised by the NAF.	GroupID value
GroupIDError	193	Message is either missing or requires a GroupID.	Not present
InvalidExtEquipName	194	External Equipment name is not known to NAF.	Not present
InvalidExtEquipType	195	Invalid value specified for External Equipment type.	Not present
OperationFailed	196	Requested operation failed.	Not present
ManufacturerCodeError	197	Error in the manufacturer code.	Specific manufacturer complement
FunctionalityNotProvided	198	Functionality not Provided by the NAF.	Not present

6.8.8 Control Plane causes

These values are returned in the CauseToPUF parameter inside the "Cause" field when the parameter is part of a message passed from NAF to PUF.

NOTE: N/A means Not Applicable.

Table 30: Control Plane causes

Value	ETS 300 102-1 [2] Meaning	ISDN PCI Meaning	Generated by	NAF provided Diagnostics
1	Unallocated (unassigned) number		ISDN	N/A
3	No route to destination		ISDN	N/A
7	Call placed on an already established channel		ISDN	N/A
16	Normal call clearing		ISDN	N/A
17	User busy		ISDN	N/A
18	No user responding		ISDN	N/A
19	No answer from user (user alerted)		ISDN	N/A
21	Call Rejected		ISDN	N/A
22	Address changed		ISDN	N/A
26	Non selected user clearing		ISDN	N/A
27	Destination out of order		ISDN	N/A
28	Invalid address format	Parameter has invalid address format.	NAF, ISDN	Not present
29	Facility rejected	Facility is not provided by this NAF.	NAF, ISDN	Not present
30	Response to STATUS ENQUIRY		ISDN	N/A
31	Normal unspecified		ISDN	N/A
34	No circuit/channel available	Temporarily no channel of requested type is available from this NAF.	NAF, ISDN	Not present
42	Switching equipment congestion		ISDN	N/A
43	Access information discarded	Parameter(s) information discarded.	NAF, ISDN	Parameter Types
44	Requested channel/circuit not available	No channel of requested type is available from this NAF.	NAF, ISDN	Not present

(continued)

Table 30: Control Plane causes (continued)

Value	ETS 300 102-1 [2] Meaning	ISDN PCI Meaning	Generated by	NAF provided Diagnostics
47	Resource unavailable, unspecified	Requested external equipment is not available.	NAF, ISDN	Not present
49	Quality of service unavailable		ISDN	N/A
50	Facility requested on Facility parameter is not subscribed		ISDN	N/A
57	Bearer Capability not authorised		ISDN	N/A
58	Bearer Capability not presently available		ISDN	N/A
63	Service or option not available, unspecified		ISDN	N/A
65	Service requested by Bearer Capability is not implemented		ISDN	N/A
66	Channel Type not implemented	NAF does not support this type of channel.	NAF, ISDN	Not present
69	Facility requested is not implemented	NAF does not support this facility.	NAF, ISDN	Not present
79	Service or option not implemented, unspecified		ISDN	N/A
81	Invalid call reference	Invalid NCOID.	NAF	Not present
82	Identified channel does not exist	Identified permanent channel is not defined.	NAF	Not present
85	No call suspended	NCOID does not identify a suspended connection.	NAF	Not present
88	Incompatible destination		ISDN	N/A
96	Mandatory parameter is missing	Mandatory parameter is missing.	NAF	Parameter Type
97	Message Identifier non-existent or not implemented on this network	Message Identifier non-existent or not implemented on this NAF.	NAF	Message Identifier
98	Message not compatible with call state or Message Identifier non-existent or not implemented.	Message not compatible with NCO state or Message Identifier non-existent or not implemented.	NAF	Message Identifier
99	Invalid parameter	Invalid parameter.	NAF	Parameter Type

(continued)

Table 30: Control Plane causes (concluded)

Value	ETS 300 102-1 [2] Meaning	ISDN PCI Meaning	Generated by	NAF provided Diagnostics
100	Invalid parameter contents	Invalid parameter contents.	NAF	Parameter Type
101	Message not compatible with current state	Message not compatible with current state.	NAF	Message Identifier
127	Inter working, unspecified		ISDN	N/A

These values are valid in the CauseToNAF parameter "Cause" field when the parameter is part of a message passed from PUF to NAF. If an invalid value is used it shall be ignored and a value of 16, Normal call clearing, used in its place.

Table 31: Content of the CauseToNAF parameter

Value	Meaning
16	Normal call clearing.
21	Call rejected.
31	Normal unspecified.
88	Incompatible destination.

6.8.9 User Plane causes

These values can be specified and are returned in the X213Cause parameter.

Table 32: X213Cause parameter value

Return Code		Meaning	ErrorSpecific Information
Undefined	220	Undefined error situation.	Not present
DiscPerm	226	Disconnection - permanent condition.	Not present
DiscTrans	225	Disconnection - transient condition.	Not present
NSAPunknown	232	Connection Rejection - NSAP address unknown (permanent condition).	Not present
NSAPunreachableTrans	231	Connection Rejection - NSAP unreachable/transient condition.	Not present
NSAPunreachablePerm	221	Connection Rejection - NSAP unreachable/permanent condition.	Not present
QOSnotavailPerm	230	Connection Rejection - QOS not available/permanent condition.	Not present
QOSnotavailTrans	229	Connection Rejection - QOS not available/transient condition.	Not present
NoReasonPerm	228	Connection Rejection - reason unspecified/permanent condition.	Not present
NoReasonTrans	227	Connection Rejection - reason unspecified/transient condition.	Not present
DiscNorm	241	Disconnection - normal condition.	Not present
DiscAbnorm	242	Disconnection - abnormal condition.	Not present
ConRejectPerm	245	Connection rejection - permanent condition.	Not present
ConRejectTrans	244	Connection rejection - transient condition.	Not present
ConRejectUserData	248	Connection rejection - incompatible information in Userdata parameter.	Not present

These values can be used and are returned in the CompletionStatus parameter.

Table 33: Other NMA UserPlane error values

Return Code		Meaning	ErrorSpecific Information
InvalidNCOID	81	Message contains an invalid NCOID.	Not present
MissingParameter	96	A mandatory parameter is missing from a message.	Parameter Type
InvalidParameter	99	A parameter is not recognised or not supported by a message.	Parameter Type
InvalidParameterLength	182	A parameter's length is outside the allowed range for the parameter.	Parameter Type
Overflow	210	Incoming data has been lost.	Not present

6.8.10 TMA User Plane causes

These values can be used and are returned in the CompletionStatus parameter.

Table 34: TMA UserPlane error values

Return Code		Meaning	ErrorSpecific Information
InvalidNCOID	81	Message contains an invalid NCOID.	Not present
MissingParameter	96	A mandatory parameter is missing from a message.	Parameter Type
InvalidParameter	99	A parameter is not recognised or not supported by a message.	Parameter Type
InvalidParameterLength	182	A parameter's length is outside the allowed range for the parameter.	Parameter Type
Overflow	210	Incoming data has been lost.	Not present

7 Exchange method

This Clause describes the exchange method and the exchange functions which are used to achieve the local exchange of information between a PUF and a NAF. Since the implementation of the exchange functions is operating system dependent, they are described in a generic way.

The rules for the NAF-sided implementation of this generic exchange functions are defined in Annex F on a per operating system basis.

Since the NAF-sided implementation of the exchange functions depends on the underlying operating system, the PUF code calling these functions is operating system dependent as well. To be source code portable between different operating systems, the PUF may want to encapsulate the code calling the NAF by a functional interface, which resembles the generic exchange functions described in this Clause. An example for such an encapsulation is given in Annex J.

The exchange functions are passing and returning parameter values. These values are based on the generic types shown in table 35.

Table 35: Generic types of exchange method

Generic Type	Explanation
PCI_INTEGER	Binary represented signed integer value, covering in the minimum the range of $-2^{15}+1 .. +2^{15}$.
PCI_BYTEARRAY	Array of binary represented byte values, used to present characters. The sign extension on the byte value is undefined. No arithmetic shall be performed on it.
PCI_EXID	Implementation dependent type for presenting the PCI Exchange-ID.
PCI_HANDLE	Operating system dependent type for presenting the PCI-Handle information.
PCI_PROCEDURE	Operating system dependent type for presentation of procedure addresses.

For the implementation (binary representation) of these types refer to Annex F.

Dependent of the operating system, the parameters are passed either by value or by reference. The way parameters are passed is defined in Annex F.

General conventions:

- the function name is prefixed by the letters "Pci".
- each function returns a completion code. Any other value than Success (0) for the completion code indicates an error.

7.1 Registration phase

7.1.1 Overview

Before a PUF and a NAF can interchange information the PUF shall associate with the NAF. For this association, the PUF shall specify the PCI-Handle of the NAF it wants to associate with.

To support many NAF implementations, possibly from different manufacturers, a method is defined which allows the PUF to discover which NAFs are accessible from within a system. For this the optional[1)] function **PciGetHandles** allows the PUF to get a list of accessible PCI-Handles. Subsequently the PUF can extract a PCI-Handle from the list. The presentation of PCI-Handle is described in the operating system specific Annex F.

---

1) The use of this function is optional for the PUF, but it's implementation (provision) is mandatory for the NAF.

If used, the **PciGetHandles** function should be the very first function called by a PUF since it makes all PCI-Handles available. The interworking with other exchange functions is shown in figure 15.

Another optional[1] function available in the registration phase is the **PciGetProperty** function. It allows the PUF to learn the properties of the NAF. On call the PUF gives the PCI-Handle of the NAF of interest. As a result the PUF obtains a list of the static properties of the NAF.

Since the obtained properties contain information about special NAF features, the PUF can use this information to select the NAF(s) it wants to register with. Examples of these special features are a handset or security features.

The only non-optional function of the registration phase is the **PciRegister** function. It allows the PUF to associate with the NAF. The PUF shall provide the PCI-Handle of the NAF it wants to associate with. As a result, an identifier for the association between the PUF and the NAF becomes available. This identifier shall be given in subsequent exchange function calls of this association during the conversation and deregistration phase.

The following terms are used in conjunction with the registration phase:

NAF-Property:	Structured information describing the characteristics (properties) of a NAF. The NAF-Property is implemented system independent using TLV coding (see subclause 6.5). Hence it shall be encoded using the same algorithm as used for encoding of messages. In a multiple NAF environment, a PUF can use this information to select a specific NAF.
PCI-Handle:	NAF access information. This information shall be supplied to the functions of the registration phase in order to find and access a NAF. Implementation of the PCI-Handle is operating system dependent. For example the PCI-Handle may be a name, a file-path or a function address.

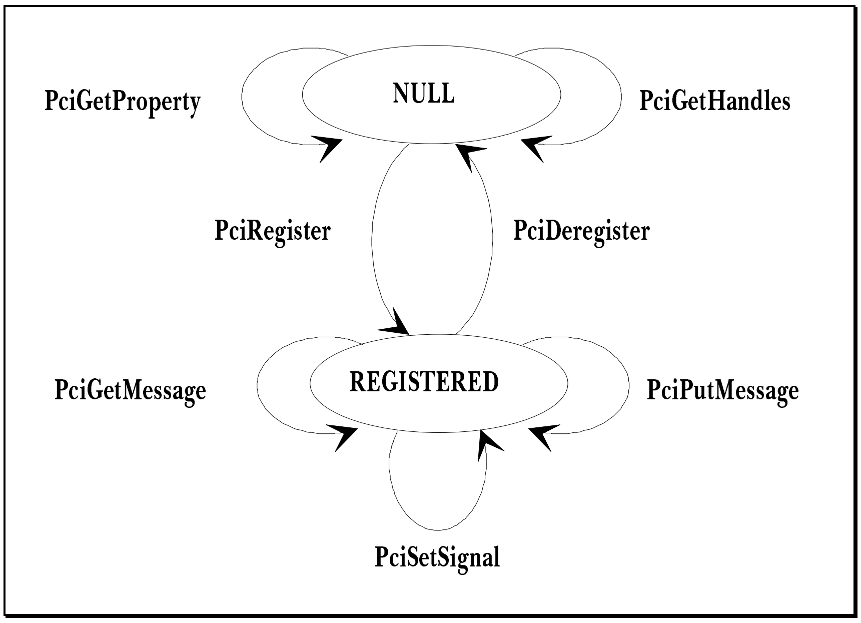


Figure 15: ISDN PCI exchange function calls order including the optional registration phase functions

NOTE: In a many NAF environments a PUF may use the optional functions PciGetHandles and PciGetProperty as described above to select the NAF which suits best to its needs. For more details on the **PciGetProperty** function refer to subclause 7.1.3.

[1] The use of this function is optional for the PUF, but it's implementation (provision) is mandatory for the NAF.

7.1.2 PciGetHandles

This function allows a PUF to ask how many NAFs are accessible and to obtain their PCI-Handles. Using the PCI-Handle, the PUF can subsequently get the NAF-Property or register with this NAF.

Function Name: PciGetHandles

Function Return Value: Errorcode (PCI_INTEGER)

Success
QueryEntityNotAvailable
InvalidHandlesBuffer
HandlesBufferTooSmall

Parameters

Name	Generic Type	Call or Return Value	Comment
MaxHandles	PCI_INTEGER	Call Value	Maximum number of PCI-Handles that can be received.
PCIHandles	Array of PCI_HANDLE	Call Value	a buffer, big enough to receive the requested maximum amount (MaxHandles) of PCI-Handles.
ActualHandles	PCI_INTEGER	Return Value	Actual number of PCI-Handles returned in the given buffer.

The PUF shall give a buffer and it's size to get the list of available PCI-Handles.

The PUF receives the actual number of PCI-Handles copied into the buffer. If this number is greater than the size of the buffer allows, the buffer is not filled and the HandlesBufferTooSmall error is returned. In this case the PUF shall provide another, bigger buffer to get the complete list of PCI-Handles.

7.1.3 PciGetProperty

This function allows a PUF to obtain the NAF-Property. The PUF shall supply a PCI-Handle as call value. A PUF can obtain a PCI-Handle either by the use of the optional **PciGetHandles** function or by use of other means (e.g. local knowledge).

Function Name: PciGetProperty

Function Return Value: Errorcode (PCI_INTEGER)

Success
InvalidPCIHandle
NAFnotAvailable
NAFBusy
PropertyBufferTooSmall

Parameters

Name	Generic Type	Call or Return Value	Comment
PCIHandle	PCI_HANDLE	Call Value	PCI-Handle presentation and values are operating system dependent.
MaximumSize	PCI_INTEGER	Call Value	Maximum size of property allowed on return.
NAFProperty	PCI_BYTEARRAY	Return Value	Property returned. Property is TLV coded, hence not system dependent (see table 35).
ActualSize	PCI_INTEGER	Return Value	Actual size of property returned.

The parameters of NAF-Property are shown in table 36.

Table 36: TLV coded NAF-Property parameter

Parameter	Provided	TLV Coding			Comment and values
		TypeID	Length	Value	
Product	M	1	1..32	Octet	Octet string indicating NAF Product
Manufacturer	M	2	1..32	Octet	Octet string indicating NAF Manufacturer
AccessClass	M	3	1	Octet	Basic Rate (1) or Primary Rate (2)
UserProtocolL3*	M	4	1..4	Octet	Give the supported layer 3 protocols. May be a NAF selection criteria. For defined value see subclause 6.6.62
UserProtocolL2*	M	5	1..2	Octet	Give the supported layer 2 protocols. May be a NAF selection criteria. For defined value see subclause 6.6.62
BChannels	M	6	1	Octet	Number of B-Channels
BPermanent	O	7	1	Octet	Number of Permanent B-Channels
DPermanent	O	8	1	Octet	Number of Permanent D-Channels
APlaneClass*	O	9	1	Octet	Additional Administration Plane functions supported. identified by class. Valid value: 2..3
CPlaneClass*	O	10	1	Octet	Additional Control Plane functions supported identified by class: Valid values are in range 2..6
SuppService*	O	11	1..16	Octet	Specification of supplementary services. For defined values see table 37 below
ExtEquipName*	O	12	2..17	Octet	In order, type and name of external equipment. See subclause 6.6.29
AdditionalUser-Protocol*	O	13	1..16	Octet	Additional User Plane protocols. For further extension
PCIVersion*	O	15	1	Octet	PCI version supported

* Parameter may be repeated.

Table 37: List of supplementary service identifiers (SuppService)

Identifier	Supplementary service
'CLIR'	Calling Line Identification Restriction
'CLIP'	Calling Line Identification Presentation
'AOC-D'	Advice Of Charge During call
'AOC-E'	Advice Of Charge at End of Call
'CW'	Call Waiting
'SUB'	Subaddressing
'MSN'	Multiple Subscriber Number
'COLP'	Connected Line Identification Presentation
'COLR'	Connected Line Identification Restriction
'DDI'	Direct Dialling In
NOTE: The complete reference to this supplementary services is provided in subclause 5.3.3.2.	

7.1.4 PciRegister

This function allows a PUF to be associated to a NAF.
As a calling parameter, the PUF provides the PCI-Handle of the NAF it wants to register with. Furthermore, two structures are passed on the function stack:

- the PCIRegisterInfo structure; and
- the PCIOpSysInfo structure.

The PCIRegisterInfo structure contains PUF and NAF specific parameters which shall be passed between the two entities to ensure proper co-operation. The PCIRegisterInfo structure is shown in table 38 below.

The PCIOpSysInfo structure contains operating system dependent information to be exchanged between PUF and NAF. For details refer to Annex F.

Table 38: Structure of the PCIRegisterInfo structure

Structure Field	Generic Type	Call or Return Value	Explanation
PUFVersion	PCI_INTEGER	Call Value	The version of the ISDN PCI the PUF wants to use. Can be set to 0 in any case (Default).
PUFType	PCI_INTEGER	Call Value	The type of PUF. This parameter is for future extensions (e.g. allow specific type of PUFs like network management PUFs) Currently this value shall be set to 0!
MaxMsgSize	PCI_INTEGER	Return Value	Maximum size of a message the NAF guarantees to deal with: NAF shall neither deliver messages bigger in size nor does it guarantee to accept bigger ones from PUF!
NOTE: The PUVersion number equals the major revision number of this ETS and has currently the value 1. The PUFType value is for further extensions and is currently assigned to 0.			

As a return value the exchange identifier (ExID) becomes available, which identifies the exchange link between PUF and NAF. The ExID shall be provided to other exchange functions interface during the conversation and the deregistration phase.

Function Name:	PciRegister
Function Return Value:	Errorcode (PCI_INTEGER)

Success
InvalidPCIHandle
NAFnotAvailable
NAFBusy
MaxPUFsExceeded
InvalidPUFType
InvalidPUFVersion
InvalidRegisterInfoStructure
InvalidOpSysInfoStructure[1)]

Parameters

Name	Generic Type	Call or Return Value	Comment
PCIHandle	PCI_HANDLE	Call Value	PCI-Handle presentation and values are operating system dependent.
PCIRegisterInfo	PCIRegisterInfo structure	Call Value	Contains PUF-NAF interaction specific information like PUFVersion and PUFType (see table 38)
PCIOpSysInfo	PCIOpSysInfo structure	Call Value	Contains Operating system dependent information. For details refer to Annex F.
ExID	PCI_EXID	Return Value	Exchange-ID.

7.2 Deregistration phase

This phase is the last phase in the information exchange between PUF and NAF. When a PUF wants to disassociate from the NAF it shall invoke the **PciDeregister** function. The use of the **PciDeregister** function shall be mandatory prior to PUF termination.

1) For more (operating system specific) error codes refer to Annex F.

When the PUF disassociates using this function the NAF frees any resources allocated for this PUF, such as clearing already existing connections.

7.2.1 PciDeregister

This function disassociates a PUF from a NAF. The association between the PUF and NAF is identified by the ExID.

Function Name:	PciDeregister
Function Return Value:	Errorcode (PCI_INTEGER)

Success
InvalidExID
NAFnotAvalaible
NAFBusy

Parameters

Name	Generic Type	Call or Return Value	Comment
ExID	PCI_EXID	Call Value	Exchange-ID as received as result of previous PciRegister function.

On return the ExID used becomes invalid, even if the error code returned indicates an error during deregistering. No further access to the NAF shall be possible using this ExID.

7.3 Conversation phase

In the conversation phase, the interaction between the PUF and the NAF consists of message and data exchange. This exchange is carried out by the generic exchange functions **PciPutMessage** and **PciGetMessage** respectively. Messages are provided, in both directions, one by one and entirely. Message and data are associated. The PCI Message Parameter Block (PCIMPB) structure contains information on message and data pointers.

Messages are processed by the NAF in an asynchronous way, but the execution of the exchange functions is synchronous. As the PUF controls the exchange of messages, messages are transmitted or received only when the PUF wishes.

7.3.1 Sending messages

The PciPutMessage function is provided for the PUF to send messages to the NAF. Before using this function the PUF shall fill the Pci Message Parameter Block (PciMPB) with the appropriate values and shall provide the addresses of the message and the data buffer. The latter only in case the PUF sends data associated with the message. The PciMPB contains the Message Identifier and details concerning the usage of the message and data buffers.

7.3.2 Receiving messages

To get a message the PUF simply issues a **PciGetMessage** function call. The PUF can use this function to poll for message availability. On function return it is indicated whether there was a message transfer or not. To avoid polling, the PUF may choose to be informed via a *signal-like* mechanism as soon as a message is available. The NAF shall inform the PUF for each event of message availability. This mode of operation improves the global performance of the system. However, in any case, the PUF shall obtain the message itself via a **PciGetMessage** function call.

7.3.3 Receiving messages using the polling method

To receive a message using this method, the PUF shall poll the NAF repeatedly to check if a message is available. If no message is available, this shall be indicated in a special way.

To be able to receive a message the PUF provides the NAF with a correctly set-up PciMPB, which shall contain the addresses of a message and a data buffer respectively. The size of the message and data buffers shall be big enough to receive the expected message. However, provision of a data buffer is optional, as data is not provided with all messages. It is up to local knowledge (see also Annex E, Clause E.9) in the PUF to determine the necessity of this buffer. The NAF indicates the total length used for each buffer. If no data is available with the message, this shall be indicated by the value zero for the length used. The absence of a message shall be indicated by the NOMESSAGE (0) type in the MessageID field of the PciMPB.

7.3.4 Receiving messages using signal method

To receive a message using this method, first, the PUF shall establish a mechanism for the NAF to notify the PUF when a message is available. This is accomplished using the **PciSetSignal** function.

This method allows a NAF to indicate that a message is available for the PUF without waiting for the PUF to use the **PciGetMessage** function**.** The indication does not involve transfer of the message from the PUF to the NAF.

The NAF notifies the PUF each time a new message is available. It shall do so until the **PciSetSignal** function is used to remove the notification mechanism.

To receive the message from the NAF, the PUF shall use the **PciGetMessage** function as described in the previous subclause.

The function calls the PUF is allowed to invoke while processing the notification may be restricted. Since these restrictions are depending on the operating system, they are defined in Annex F.

7.3.5 PCI Message Parameter Block (PciMPB)

Table 39 shows the structure of the PCI Message Parameter Block (PciMPB).

Table 39: Structure of the PCI Message Parameter Block (PciMPB)

Structure Field	Generic Type	Explanation
MessageID	PCI_INTEGER	Message identifier. Shall be provided by PUF on invocation of PciPutMessage, available for PUF on return of PciGetMessage.
MessageMaximumSize	PCI_INTEGER	Maximum size of message. To be provided on calls to PciGetMessage.
MessageActualUsedSize	PCI_INTEGER	Actual used size of message. Shall be provided by PUF on calls to PciPutMessage, will be available to PUF on return of PciGetMessage.
DataMaximumSize	PCI_INTEGER	Maximum size of data buffer. To be provided on calls to PciGetMessage.
DataActualUsedSize	PCI_INTEGER	Actual used size of data buffer. Shall be provided by PUF on calls to PciPutMessage, will be available to PUF on return of PciGetMessage.

Figure 16 presents how messages are sent or received.

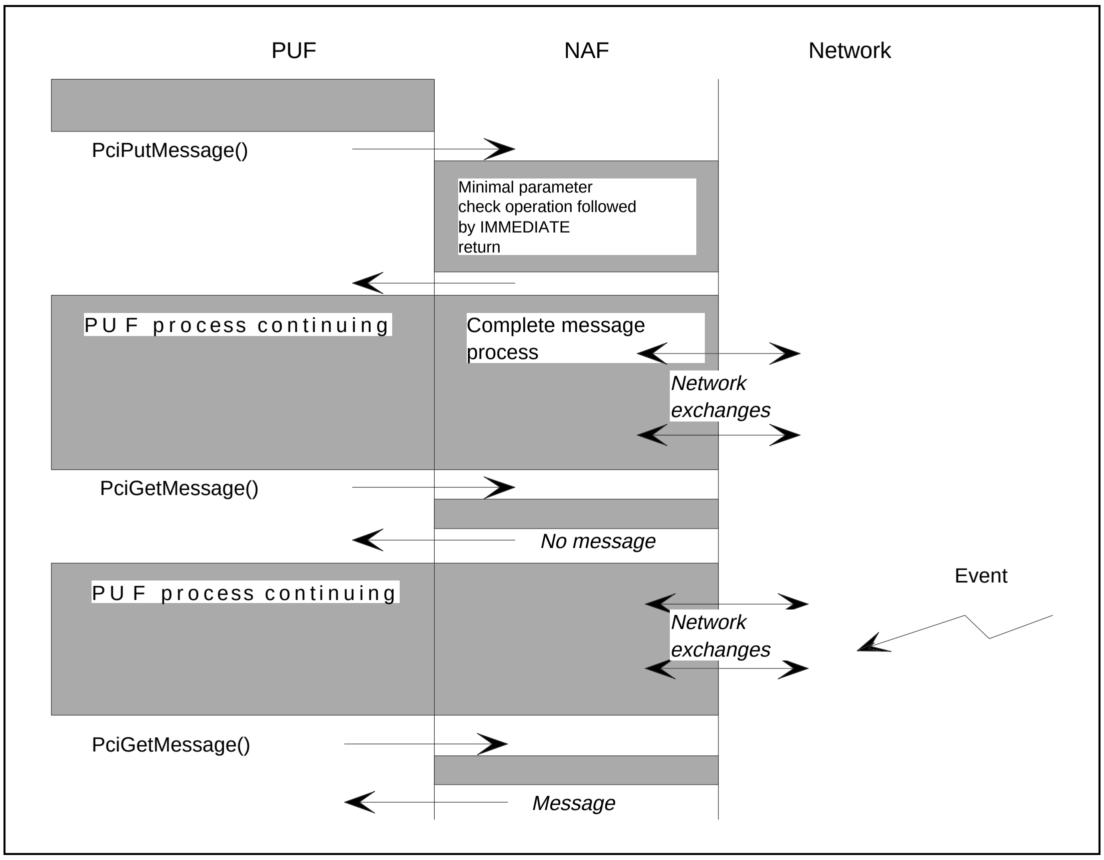


Figure 16: Process to send or to receive messages

7.3.6 PciPutMessage

This function allows a PUF to transmit a message to a NAF.

Function Name:	PciPutMessage
Function Return Value:	Errorcode (PCI_INTEGER)

Success
InvalidExID
NAFnotAvailable
NAFBusy
InvalidPCIMPB
InvalidMessageBuffer
PCIMPBTooSmall
MessageBufferTooSmall
DataBufferTooSmall
MessageTooLarge
DataBufferRequired

Parameters

Name	Generic Type	Call or Return Value	Comment
ExID	PCI_EXID	Call Value	Exchange-ID as received as result of previous PciRegister function.
PCIMPB	PCIMPB structure	Call Value	PCI Message Parameter Block.
Message	PCI_BYTEARRAY	Call Value	Message to be sent to NAF.
Data	PCI_BYTEARRAY	Call Value	Data associated with the message.

The PUF indicates the type of the message in the MessageID field of the PCIMPB.

The PUF indicates the size of the buffer(s) in the ActualUsedSize field of the PCIMPB for the respective buffers, i.e. MessageActualUsedSize for the message buffer and DataActualUsedSize for the data buffer.

It is allowed to provide only a message buffer without a data buffer or a data buffer without message buffer. However the PciMPB structure is always required. To indicate absence of a buffer, the PUF may specify no buffer address instead by supplying a NULL (0) value.

7.3.7 PciGetMessage

This function allows a PUF to get a message from a NAF.

Function Name:	PciGetMessage
Function Return Value:	Error code (PCI_INTEGER)

Success
InvalidExID
NAFnotAvailable
NAFBusy
InvalidPCIMPB
InvalidMessageBuffer
PCIMPBTooSmall
MessageBufferTooSmall
DataBufferTooSmall
MessageTooLarge

Parameters

Name	Generic Type	Call or Return Value	Comment
ExID	PCI_EXID	Call Value	Exchange-ID as received as result of previous PciRegister function.
PCIMPB	PCIMPB structure	Call Value and Return Value	PCI Message Parameter Block.
Message	PCI_BYTEARRAY	Return Value	Message received from NAF.
Data	PCI_BYTEARRAY	Return Value	Data associated with the message.

The PUF is in charge to provide buffers. If a buffer is too small to receive the message or data provided by the NAF, the NAF returns an error.

The PUF indicates the maximum size of the buffer(s) in the MaximumSize fields of the PciMPB for the respective buffers, i.e. MessageMaximumSize for the message buffer and DataMaximumSize for the data buffer.

On return, the NAF indicates the size of the buffer(s) in the ActualUsedSize field of the PciMPB for the respective buffers used.

To indicate no message, the NAF fills the MessageID field in the PCIMPB with NOMESSAGE (0).

7.3.8 PciSetSignal

This function allows a PUF to ask for notification when an event occurs. An event is any incoming message from the Network or from the NAF. The signal mechanism shall stay in effect until the PUF disassociates from the NAF or explicitly shuts-down the notification action (see below).

Function Name: PciSetSignal

Function Return Value: Errorcode (PCI_INTEGER)

Success
InvaldExID
NAFnotAvailable
NAFBusy
InvalidSignalNumber

Parameters

Name	Generic Type	Call or Return Value	Comment
ExID	PCI_EXID	Call Value	Exchange-ID as received as result of previous PciRegister function.
Signal	PCI_INTEGER	Call Value	Value is operating system dependent.
SignalProcedure	PCI_PROCEDURE	Call Value	Value and presentation is operating system dependent.

The real signal mechanism used is operating system dependent. Details, per operating system, can be found in Annex F.

Any new **PciSetSignal** call shall overwrite the previous one.

The signal mechanism can be stopped by supplying a NULL(0) value instead of Signal and SignalProcedure values during call.

8 Security

This Clause addresses communication security using the ISDN PCI.

8.1 General aspects of security in ISDN

ISDN digital nature facilitates adding security, but ISDN has been developed with little thought to security. The deployment of ISDN in the public network is well under way and this constrains how security features may be added. The current specifications for the lower layers of ISDN do not contain specific features for security.

From the point of view of applications, the following needs for security can be seen:

- protecting information confidentiality;
- identifying the parties in communications (authentication);
- assuring the integrity of communicated information;
- controlling access to network services and customer equipment and data;
- being able to prove to a third party the fact that a communication occurred, the contents and the identities of the parties involved (non-repudiation).

From a security perspective, ISDN is more than a lower-layer communication service. Within the ISDN there has to be some concern for the applications, and especially the security requirements of these applications.

A foundation of common security standards for ISDN, particularly for authentication, confidentiality and integrity can provide the needed platform upon which the specific security needed by various ISDN applications can be built. The needed technology exists; it remains only to adopt it to ISDN and incorporate it in standards.

8.2 Security in the ISDN PCI

Although there are no security standards available, the ISDN PCI offers some access to security functionalities which may be available in the NAF. This access provided a first approach for using security on ISDN.

The PUF can access the security functionality of the NAF in the following ways:

- using the supplementary service Calling Line Identification Presentation (CLIP);

  The CLIP supplementary service provides the PUF with the calling user's ISDN number, possibly with sub-address information. The ISDN number and sub-address information shall be provided by the network, and are, therefore, more dependable to be used for identifying the calling user. This supplementary service provides the PUF with a method to identify the other party. See subclause 6.3. for information on the use of this supplementary service.

- Using the security messages in the Administration Plane:

  - ASecurityReq;
  - ASecurityCnf.

  This security access provides the PUF access to encryption and security features which can be provided by the NAF. These messages provide a way to exchange the information needed for using the security features of the NAF. This security access provides a method for protecting information confidentiality. See subclauses 6.2.9 and 6.2.10 for information on the use of these Administration Plane messages.

8.3 Increasing security in the ISDN PCI

As no standards exist for security in ISDN, only limited features for security can be added in the ISDN PCI. These security features are described in subclause 8.2.

Although the standards do not exist, the impact of introducing security in the ISDN PCI can be estimated. There can be seen three levels of introducing security:

1) Security features as supplementary services

   There should be little impact on the ISDN PCI or PUFs. These supplementary services should be handled in the same way as the normal ones.

2) Security as one specific protocol in the NAF

   If a secure protocol is operating on one of the lower layers, the PUF may only have to supply this protocol with the necessary security information. This can be achieved by extending the Administration Plane to allow the transfer of the information. There are several ways for extension:

   - adding a message;
   - extending the attribute sets to contain the security information;
   - NCOs to contain the security information.

3) Definition of a new secure protocol stack for ISDN

   If new secure protocols are to be established, the ISDN PCI shall be altered. New User Plane and Control Plane protocols might have to be established. The extension mechanism provided by the ISDN PCI can be used to supply these new protocols.

Although the impact of introducing security in the ISDN PCI can be estimated, the actual introduction of additional security features in the ISDN PCI is for further study.

Annex A (informative): Bibliography

This bibliography contains references to documents which are of importance to the PUF and NAF developers. The documents can be useful when reading or implementing this ETS.

Directive 86/659/EEC: "Council recommendation of 22 December 1986 on the co-ordinated introduction of the integrated services digital network (ISDN) in the European Community".

ETS 300 102-2 (1990): "Integrated Services Digital Network (ISDN); User-network interface layer 3, Specification for basic call control, Specification Description Language (SDL) diagrams".

ETS 300 196 (1991): "Integrated Services Digital Network (ISDN); Generic functional protocol for the support of supplementary services, Digital Subscriber Signalling System No. one (DSS1) protocol".

ISO/IEC 9574 (1989): "Information technology - Telecommunications and information exchange between systems - Provision of the OSI connection-mode network service by packet mode terminal equipment connected to an integrated services digital network (ISDN)".

ETS 300 050 (1991): "Integrated Services Digital Network (ISDN); Multiple Subscriber Number (MSN) supplementary service, Service description".

ETS 300 051 (1991): "Integrated Services Digital Network (ISDN); Multiple Subscriber Number (MSN) supplementary service, Functional capabilities and information flows".

ETS 300 052 (1991): "Integrated Services Digital Network (ISDN); Multiple Subscriber Number (MSN) supplementary service, Digital Subscriber Signalling System No. one (DSS1) protocol".

ETS 300 059 (1991): "Integrated Services Digital Network (ISDN); Subaddressing (SUB) supplementary service, Service description".

ETS 300 060 (1991): "Integrated Services Digital Network (ISDN); Subaddressing (SUB) supplementary service, Functional capabilities and information flows".

ETS 300 061 (1991): "Integrated Services Digital Network (ISDN); Subaddressing (SUB) supplementary service, Digital Subscriber Signalling System No. one (DSS1) protocol".

ETS 300 062 (1991): "Integrated Services Digital Network (ISDN); Direct Dialling In (DDI) supplementary service, Service description".

ETS 300 063 (1991): "Integrated Services Digital Network (ISDN); Direct Dialling In (DDI) supplementary service, Functional capabilities and information flows".

ETS 300 064 (1991): "Integrated Services Digital Network (ISDN); Direct Dialling In (DDI) supplementary service, Digital Subscriber Signalling System No. one (DSS1) protocol".

ETS 300 053 (1991): "Integrated Services Digital Network (ISDN); Terminal Portability (TP) supplementary service, Service description".

ETS 300 054 (1991): "Integrated Services Digital Network (ISDN); Terminal Portability (TP) supplementary service, Functional capabilities and information flows".

ETS 300 055 (1991): "Integrated Services Digital Network (ISDN); Terminal Portability (TP) supplementary service, Digital Subscriber Signalling System No. one (DSS1) protocol".

ETS 300 089 (1992): "Integrated Services Digital Network (ISDN); Calling Line Identification Presentation (CLIP) supplementary service, Service description".

ETS 300 091 (1992): "Integrated Services Digital Network (ISDN); Calling Line Identification Presentation (CLIP) and Calling Line Identification Restriction (CLIR) supplementary services, Functional capabilities and information flows".

ETS 300 092 (1992): "Integrated Services Digital Network (ISDN); Calling Line Identification Presentation (CLIP) supplementary service, Digital Subscriber Signalling System No. one (DSS1) protocol".

ETS 300 090 (1992): "Integrated Services Digital Network (ISDN); Calling Line Identification Restriction (CLIR), supplementary service Service description".

ETS 300 093 (1992): "Integrated Services Digital Network (ISDN); Calling Line Identification Restriction (CLIR) supplementary service, Digital Subscriber Signalling System No. one (DSS1) protocol".

ETS 300 056 (1991): "Integrated Services Digital Network (ISDN); Call Waiting (CW) supplementary service, Service Description".

ETS 300 057 (1992): "Integrated Services Digital Network (ISDN); Call Waiting (CW) supplementary service, Functional capabilities and information flows".

ETS 300 058 (1991): "Integrated Services Digital Network (ISDN); Call Waiting (CW) supplementary service Digital Subscriber Signalling System No. one (DSS1) protocol".

ETS 300 179 (1992): "Integrated Services Digital Network (ISDN); Advice of Charge: charging information during the call (AOC-D) supplementary service, Service description".

ETS 300 181 (1993): "Integrated Services Digital Network (ISDN); Advice of Charge (AOC) supplementary service, Functional capabilities and information flows".

ETS 300 182 (1993): "Integrated Services Digital Network (ISDN); Advice of Charge (AOC), supplementary service, Digital Subscriber Signalling System No. one (DSS1) protocol".

ETS 300 180 (1992): "Integrated Services Digital Network (ISDN); Advice of Charge: charging information at the end of the call (AOC-E) supplementary service, Service description".

ETS 300 094 (1991): "Integrated Services Digital Network (ISDN); Connected Line Identification Presentation (COLP) supplementary service, Service description".

ETS 300 095 (1991): "Integrated Services Digital Network (ISDN); Connected Line Identification Restriction (COLR) supplementary service, Service description".

ETS 300 096 (1991): "Integrated Services Digital Network (ISDN); Connected Line Identification Presentation (COLP) and Connected Line Identification Restriction (COLR) supplementary services, Functional capabilities and information flows".

ETS 300 097 (1991): "Integrated Services Digital Network (ISDN); Connected Line Identification Presentation (COLP) supplementary service, Digital Subscriber Signalling System No. one (DSS1) protocol".

ETS 300 098 (1991): "Integrated Services Digital Network (ISDN); Connected Line Identification Restriction (COLR) supplementary service, Digital Subscriber Signalling System No. one (DSS1) protocol".

ETS 300 102-2 (1990): "Integrated Services Digital Network (ISDN); User-network interface layer 3, Specifications for basic call control, Specification Description Language (SDL) diagrams".

ISO 8878 (1990): "Information processing systems - Data communications - Use of X.25 to provide the OSI connection-mode network service".

CCITT Recommendation Z.100 (1988): "Functional specification and description language (SDL)".

CCITT Recommendation X.211 (1988): "Physical service definition of open systems interconnection for CCITT applications".

Annex B (normative): Mapping between ISDN PCI messages and Protocols supported

This annex provides the mapping between protocols used and the ISDN PCI messages.

B.1 Control Plane messages

Table B.1 shows the mapping of the Control Plane.

Table B.1: Control Plane message to ETS 300 102-1 [2] mapping

PCI Message	ETS 300 102-1 [2] Message	Direction	NOTES
CAlertReq	Alerting	user to network	
CAlertInd	Alerting	network to user	
CConnectReq	Setup	user to network	
CConnectInd	Setup	network to user	
CConnectRsp	Connect	user to network	
CConnectCnf	Connect	network to user	
CDisconnectReq	Disconnect, Release, Release Complete	user to network	NOTE 1
CDisconnectInd	Disconnect, Release, Release Complete	network to user	NOTE 1
CDisconnectRsp	Release	user to network	NOTE 1
CDisconnectCnf	Release, Release Complete	network to user	NOTE 1
CProgressInd	Progress	network to user	
CStatusInd	Status	network to user	NOTE 2
CProceedingInd	Call proceeding	network to user	
CSetupAckInd	Setup acknowledge	network to user	
CConnectInfoReq	Information	user to network	
CUserInformationReq	User Information	user to network	
CUserInformationInd	User Information	network to user	
CCongestionControlReq	Congestion Control	user to network	

(continued)

Table B.1: Control Plane message to ETS 300 102-1 [2] mapping (concluded)

PCI Message	**ETS 300 102-1 [2] Message**	**Direction**	**NOTES**
CCongestionControlInd	Congestion Control	network to user	
CSuspendReq	Suspend	user to network	
CSuspendCnf	Suspend acknowledge Suspend reject	network to user	
CResumeReq	Resume	user to network	
CResumeCnf	Resume acknowledge Resume reject	network to user	
CNotifyInd	Notify	network to user	
CFacilityReq	Facility	user to network	
CFacilityInd	Facility	network to user	

NOTE 1: In the case of the PCI CDisconnect* messages the specific message received or sent to the ISDN depends upon the state of the call when the CDisconnect* message is received from, or sent to the PUF. Depending on the ISDN message that caused the CDisconnectInd, CDisconnectRsp may or may not cause a message to be sent to the ISDN. CDisconnectCnf shall not be mapped from a message from the ISDN when CDisconnectReq is used to respond to CConnectInd.

NOTE 2: This PCI message may be generated by a protocol error detected by the NAF or by a protocol error indicated by a Status message received from the ISDN.

NOTE 3: External equipment messages are not included in this table.

B.2 Control Plane parameters

The mapping of Control Plane parameters to ETS 300 102-1 [2] information elements is defined in table B.2.

Table B.2: Control Plane parameters

Control Plane parameter	**ETS 300 102-1 [2] Information Element**
BearerCap	Bearer Capability
CalledNumber	Called party number
CalledSubaddress	Called party subaddress
CallingNumber	Calling party number
CallingSubaddress	Calling party subaddress
CauseToPUF	Cause
CauseToNAF	Cause
ChannelIdentification	Channel Identification
CongestionLevel	Congestion level
ConnectedNumber	Called party number
ConnectedSubaddress	Called party subaddress
DateTime	Date/time
Display	Display
Facility	Facility
HLC	High layer compatibility
Keypad	Keypad facility
LLC	Low layer compatibility
NotificationIndicator	Notification Indicator
ProgressIndicator	Progress Indicator
UserToUserInfo	User-user

B.3 User Plane messages

The mapping of User Plane messages to protocol messages depends on whether the NAF is providing the co-ordination function for a particular Control Plane connection.

When the NAF is providing the co-ordination function, the mapping of X.213 service primitives to ETS 300 102-1 [2] messages and X.25 packets is explained in ISO/IEC 9574 and ISO/IEC 8878.

When the NAF is not providing the co-ordination function the mapping of X.213 service primitives to X.25 packets is explained in ISO/IEC 8878.

Table B.3 shows the mapping of User Plane messages to X.213 service primitives.

Table B.3: UserPlane message

PCI Message	X.213 Primitive
U3ConnectReq	N-CONNECT request
U3ConnectInd	N-CONNECT indication
U3ConnectRsp	N-CONNECT response
U3ConnectCnf	N-CONNECT confirm
U3DisconnectReq	N-DISCONNECT request
U3DisconnectInd	N-DISCONNECT indication
U3DataReq	N-DATA request
U3DataInd	N-DATA indication
U3ExpeditedDataReq	N-EXPEDITED-DATA request
U3ExpeditedDataInd	N-EXPEDITED-DATA indication
U3ResetReq	N-RESET request
U3ResetInd	N-RESET indication
U3ResetRsp	N-RESET response
U3ResetCnf	N-RESET confirm
U3DataAcknowledgeReq	N-DATA-ACKNOWLEDGE request
U3DataAcknowledgeInd	N-DATA-ACKNOWLEDGE indication
U3ReadyToReceiveReq	Not equivalent to an X.213 primitive
U3ReadyToReceiveInd	Not equivalent to an X.213 primitive

Annex C (normative): Telephony

This annex presents different types of external equipment handled in this ETS

C.1 Type 1 external equipment

This external equipment is the simplest form of telephony equipment. It does not contain a hook mechanism or a dialling mechanism. It only contains the transceivers and does not manage the ISDN signalling. It is totally under the control of the NAF. A Control Plane message is defined to indicate to the PUF the availability or not of the external equipment (external equipment connected or not connected to the NAF).

It shall be is the responsibility of the NAF to connect a channel to this type of external equipment when the channel becomes active.

If the external equipment is in use, a CConnectReq that attempts to use the external equipment should be rejected with an CDisconnectInd with Cause value 47 (Resource unavailable).

If the external equipment is in use, and an incoming call arrives that attempts to use the external equipment, the NAF should pass a CConnectInd to the relevant PUF. The PUF is then in control to make the external equipment available for use. If it does not, an attempt to connect shall be denied with CDisconnectInd with Cause value 47 (resource unavailable).

C.2 Type 2 external equipment

This external equipment contains a hook mechanism but not a dialling mechanism. This external equipment does not manage ISDN signalling. It can provide some information to the PUF about the state of the handset by the means of two Control Plane messages. Therefore, this external equipment can cause state transition in the PUF automat for the incoming calls and the outgoing calls.

A Control Plane message is defined to indicate to the PUF the availability or non-availability of the external equipment (external equipment connected or not connected to the NAF).

It is the responsibility of the NAF to connect a channel to this type of external equipment when the channel becomes active. It shall be the responsibility of the PUF to ensure that the hook mechanism is in the required state when the channel becomes active.

If the equipment is in use, and an incoming call arrives that attempts to use the equipment, the NAF should pass an CConnectInd to the relevant PUF. The PUF is then in control to make the external equipment available for use. If it does not, an attempt to connect shall be denied with CDisconnectInd with Cause value 47 (Resource unavailable).

C.3 Type 3 external equipment

This external equipment contains a hook mechanism but not a dialling mechanism. This external equipment is connected to the ISDN network; therefore, it is able to manage ISDN signalling when an incoming call arrives in the case where the host is off.

It can provide some information to the PUF about the state of the handset by the means of two Control Plane messages. Therefore, this external equipment can cause state transition in the PUF automat for the incoming calls and the outgoing calls.

A Control Plane message is defined to indicate to the PUF the availability or non-availability of the external equipment (external equipment connected or not connected to the NAF).

If the external equipment is in use, and an incoming call arrives that attempts to use the equipment, the NAF should pass a CConnectInd to the relevant PUF. The PUF is then in control to make the external equipment available for use. If it does not, an attempt to connect shall be denied with CDisconnectInd with Cause value 47 (Resource unavailable).

C.4 Type 4 external equipment

This external equipment contains a dialling mechanism and/or not a hook mechanism. This external equipment does not manage ISDN signalling. This kind of external equipment can allow dialling with block sending or overlap sending. In the case of an overlap sending, a Control Plane message containing the code of the key pressed on the keypad per key pressed is provided to the PUF. In the case of a block sending, a single Control Plane message containing the complete remote address and/or subaddress is provided to the PUF.

If this external equipment contains a hook mechanism, it can provide some information to the PUF about the state of the handset by the means of two Control Plane messages.

A Control Plane message is defined to indicate to the PUF the availability or not of the external equipment (external equipment connected or not connected to the NAF).

All dialling actions and handset actions (if available) realised on this external equipment can cause state transition in the PUF automat for the incoming calls and the outgoing calls.

It is the responsibility of the NAF to connect a channel to this type of external equipment when the channel becomes active.

If the equipment is in use, and an incoming call arrives that attempts to use the equipment, the NAF should pass a CConnectInd to the relevant PUF. The PUF is then in control to make the external equipment available for use. If it does not, an attempt to connect shall be denied with CDisconnectInd with Cause value 47 (Resource unavailable).

C.5 Type 5 external equipment

This external equipment contains a dialling mechanism and/or not a hook mechanism. This external equipment is connected to the ISDN network; therefore, it is able to manage ISDN signalling in the case where the personal computer is off: it allows to make an outgoing call from it and to answer to an incoming call.

This kind of external equipment can allow dialling with block sending or overlap sending. In the case of an overlap sending, a Control Plane message containing the code of the key pressed on the keypad per key pressed is provided to the PUF. In the case of a block sending, a single Control Plane message containing the complete remote address and/or subaddress shall be provided to the PUF.

If this external equipment contains a hook mechanism, it can provide some information to the PUF about the state of the handset by the means of two Control Plane messages.

A Control Plane message is defined to indicate to the PUF the availability or not of the external equipment (external equipment connected or not connected to the NAF).

All dialling actions and handset actions (if available) realised on this external equipment can cause state transition in the PUF automat for the incoming calls and the outgoing calls.

It is the responsibility of the NAF to connect a channel to this type of external equipment when the channel becomes active.

If the equipment is in use, and an incoming call arrives that attempts to use the equipment, the NAF should pass a CConnectInd to the relevant PUF. The PUF is then in control to make the external equipment available for use. If it does not, an attempt to connect shall be denied with CDisconnectInd with Cause value 47 (Resource unavailable).

Annex D (normative): CCITT Recommendation X.25 Usage

D.1 Parameter Values for CCITT Recommendation X.25 Use

Table D.1 shows the required setting of parameters to achieve different types of CCITT Recommendation X.31 operation.

Table D.1: Types of CCITT Recommendation X.31 operation

Type of X.31 Operation	BearerCap	Channel Selection	Channel Number	Called Number
X.31 Case A Switched	64 KHZ	Not Required	Not Required	Required
X.31 Case A Permanent	64 KHZ	B-channel	Required	Not Required
X.31 Case B, B-channel switched	X25	Not Required	Not Required	Not Required
X.31 Case B, B-channel permanent	X25	B-channel	Required	Not Required
X.31 Case B, D-channel	X25	D-channel	Required	Not Required

D.2 Disconnection of ISDN channel with established CCITT Recommendation X.25 Connections

In the co-ordination case, this is covered by ISO 9574.

In the non co-ordination case, the following should be provided to the PUF:

- CDisconnectInd message with cause for channel disconnection;
- For each established CCITT Recommendation X.25 Connection:
    - U3DisconnectInd message with X213Cause and X213Origin as defined by ISO 9574 and X25Cause.
- For each CCITT Recommendation X.25 Connection in the process of being established:
    - U3DisconnectInd message with X213Cause and X213Origin as defined by ISO 9574 and X25Cause.

Annex E (informative): NAF development guidelines

The main body of this ETS contains the description of the ISDN PCI from the PUF point of view. Following this approach, certain points, not directly related to the PUF, which have an impact on the development of the NAF are not described. These points may be of interest for the NAF development and are, therefore, described in this annex. It gives guidelines for the development of the NAF in accordance with the main body of the ETS.

An example of a point which is not covered in the main body is the mapping between the coding for the AOC supplementary service and the special coding used in the ISDN PCI.

There are three basic assumptions for understanding the points described in this annex:

- this annex gives additional points. The NAF should be developed using this ETS. It should implement the ISDN PCI in such a way that the functionality described is offered;

- the main body of this ETS should be given priority if there is anything not clear in this annex or the interpretation between the main body of this ETS and this annex is different;

- this annex does not try to impose any constraints on the implementation of the NAF. The objective is to give guidelines as to how the NAF can be developed to be in line with this ETS.

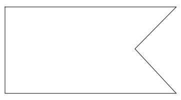
E.1 NAF SDL diagrams

The following SDL diagrams show the internal states of the call control section of the NAF.

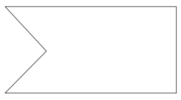
This part of the NAF is treated as a Call Control Block as defined in ETS 300 102-2, subclause 6.2. The primitives shown in upper case are those defined in table 6 of ETS 300 102-2. The complete operation of the NAF is defined by the following SDL diagrams and the SDL diagrams defined in ETS 300 102-2, subclause 7.2.

The following symbols are used within this description. A full description of the symbols and their meaning is given in CCITT Recommendation Z.100.

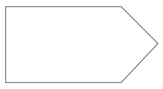
State Symbol



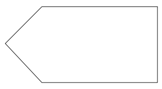
Input (from Network)



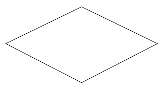
Input (from PUF)



Output (to Network)



Output (to PUF)



Decision Symbol

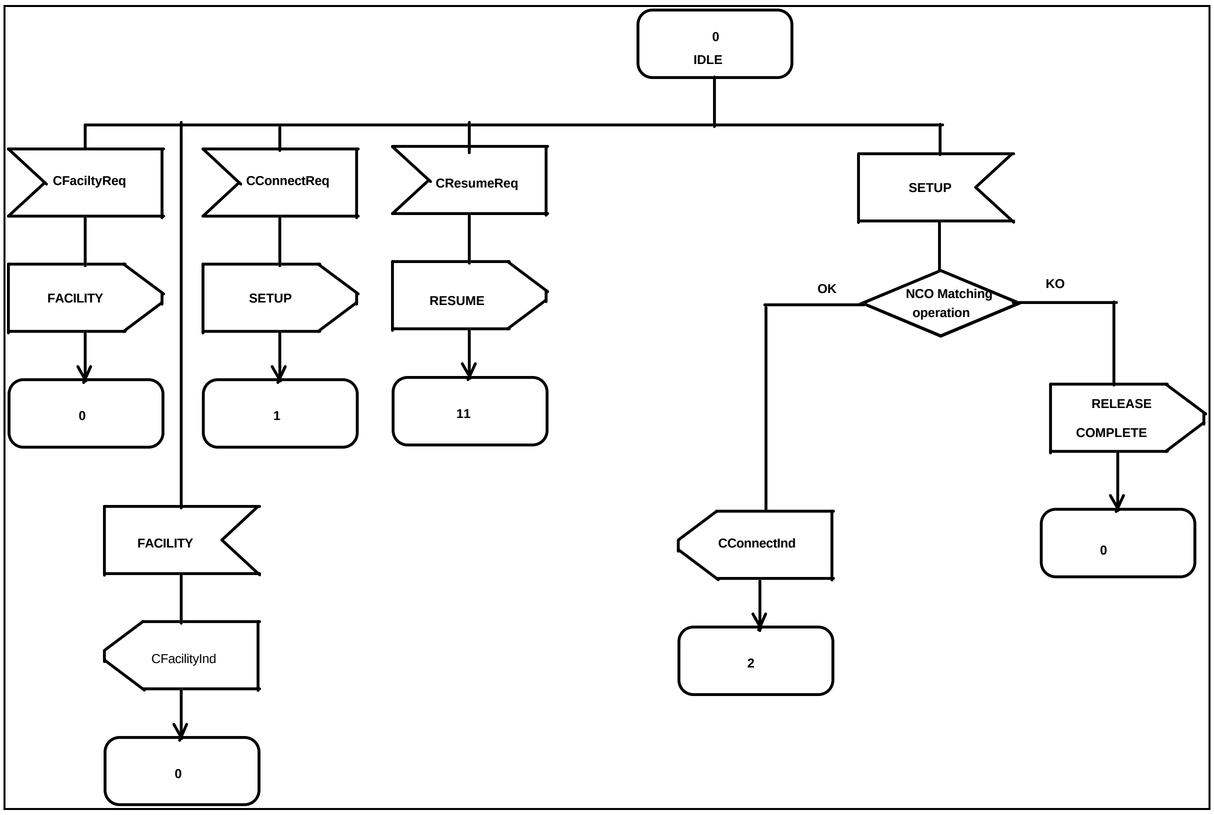


Figure E.1: IDLE

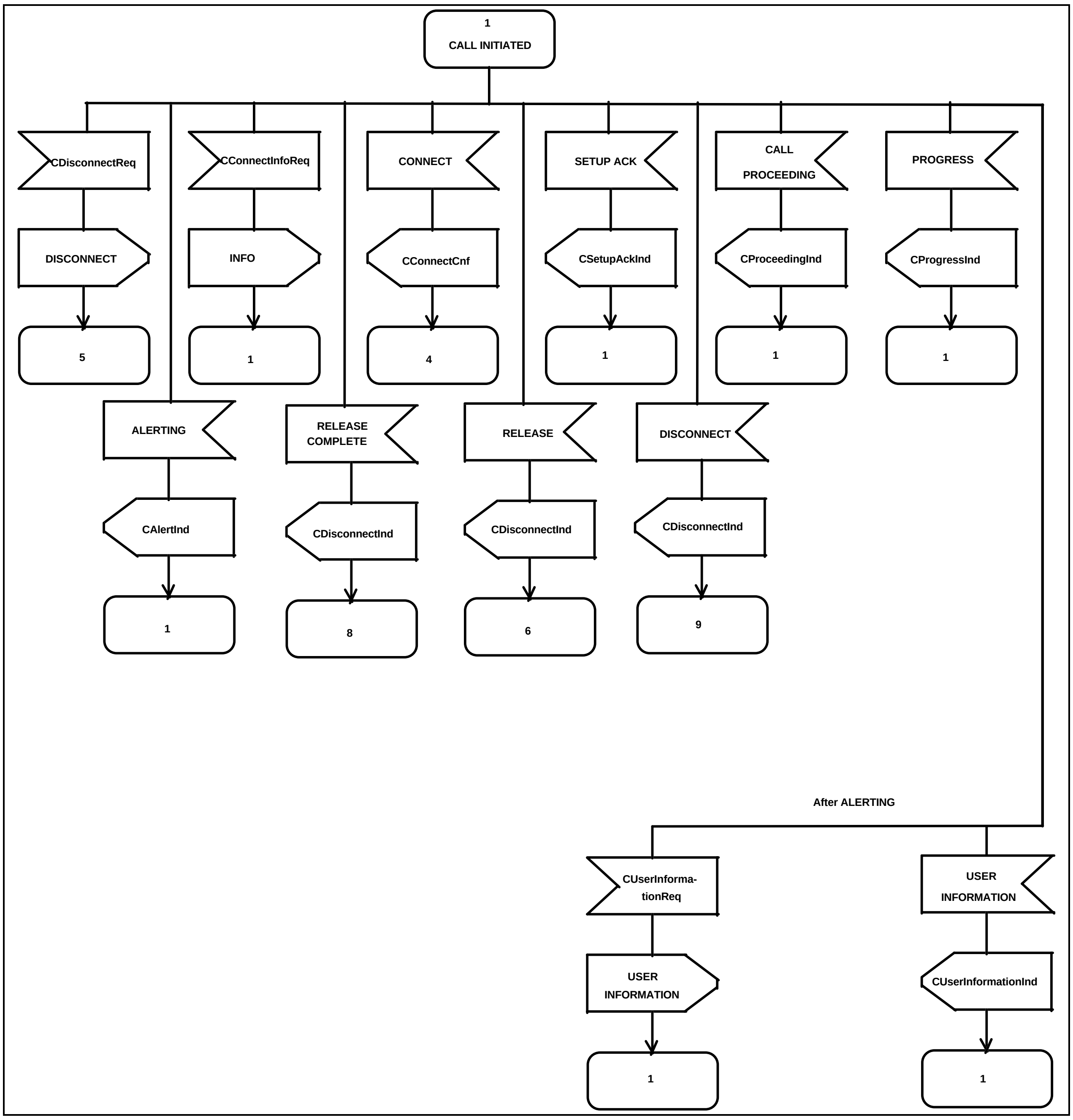


Figure E.2: CALL INITIATED

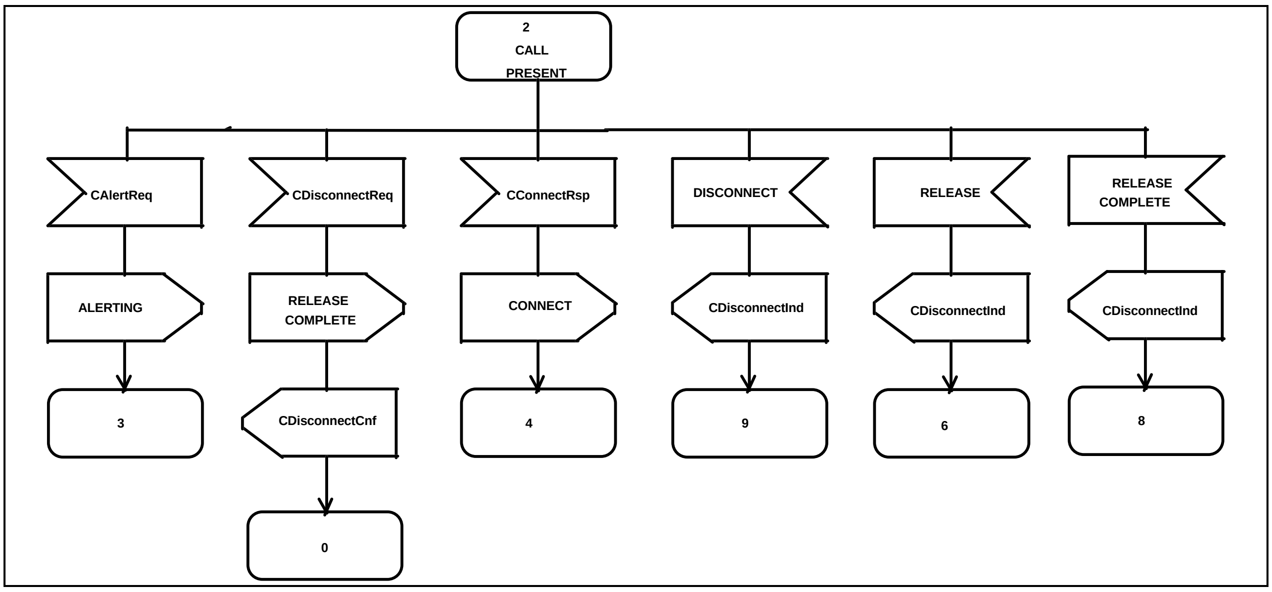


Figure E.3: CALL PRESENT

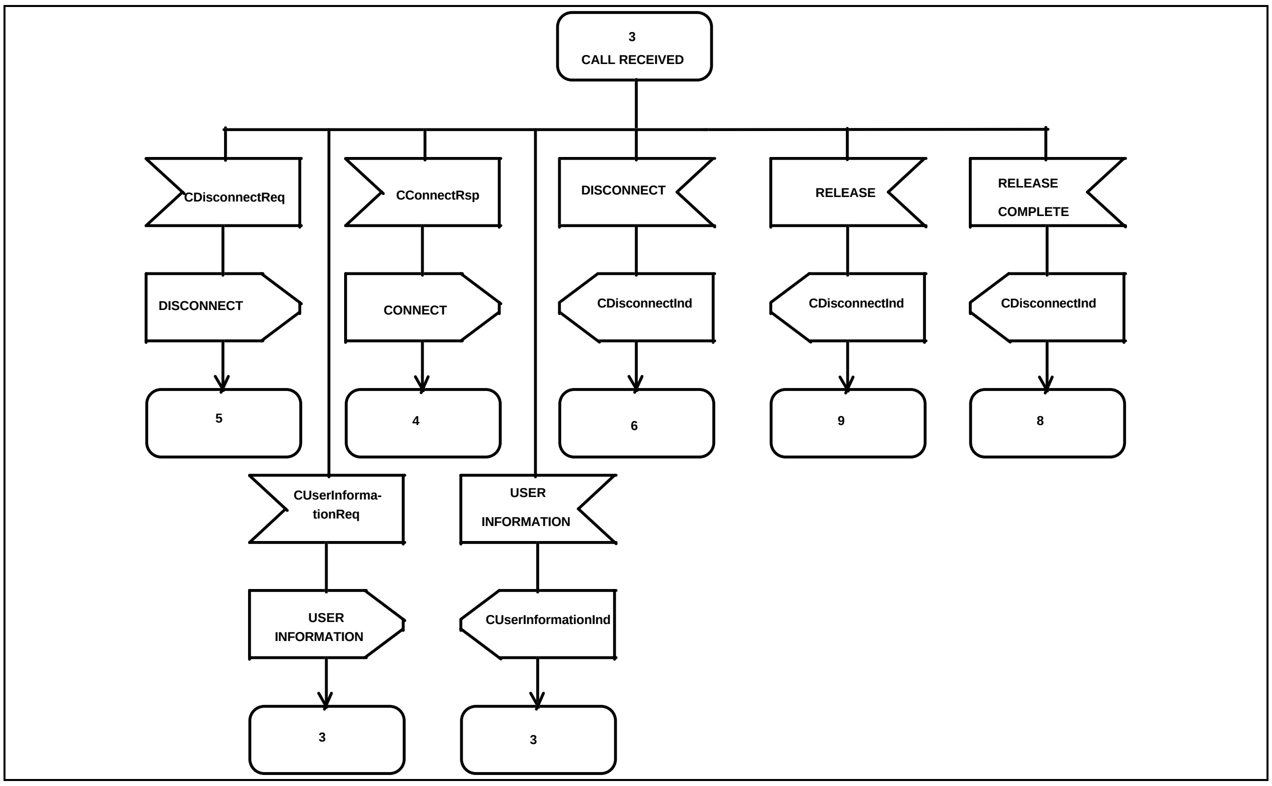
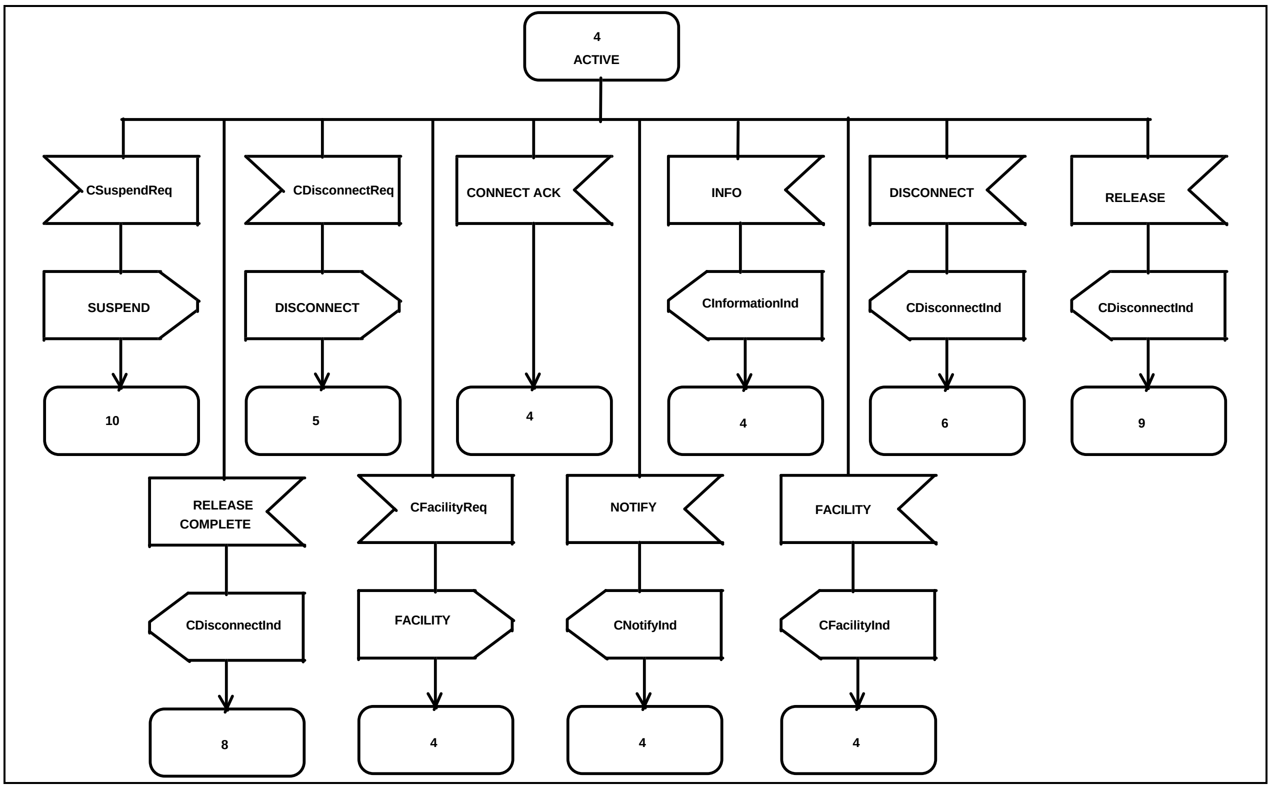


Figure E.4: CALL RECEIVED



NOTE: In order to simplify the state diagrams, the ACTIVE connection state in this ETS, is not strictly equal to the state describes in the ETS 300 102-2.

Figure E.5: ACTIVE connection

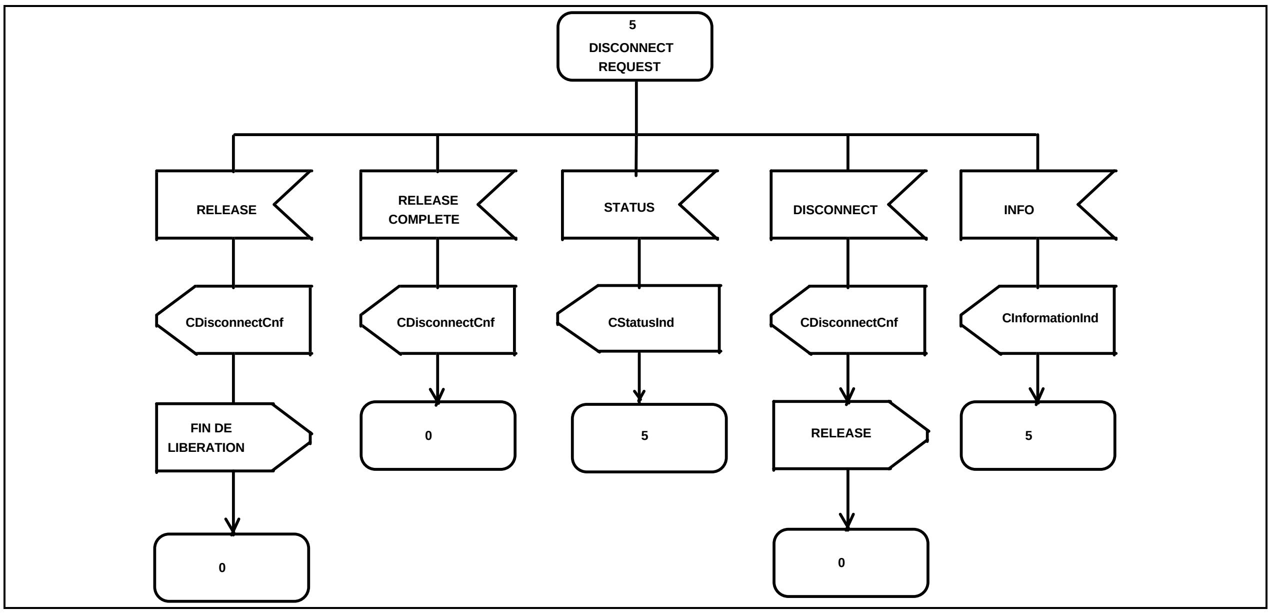


Figure E.6: DISCONNECT request

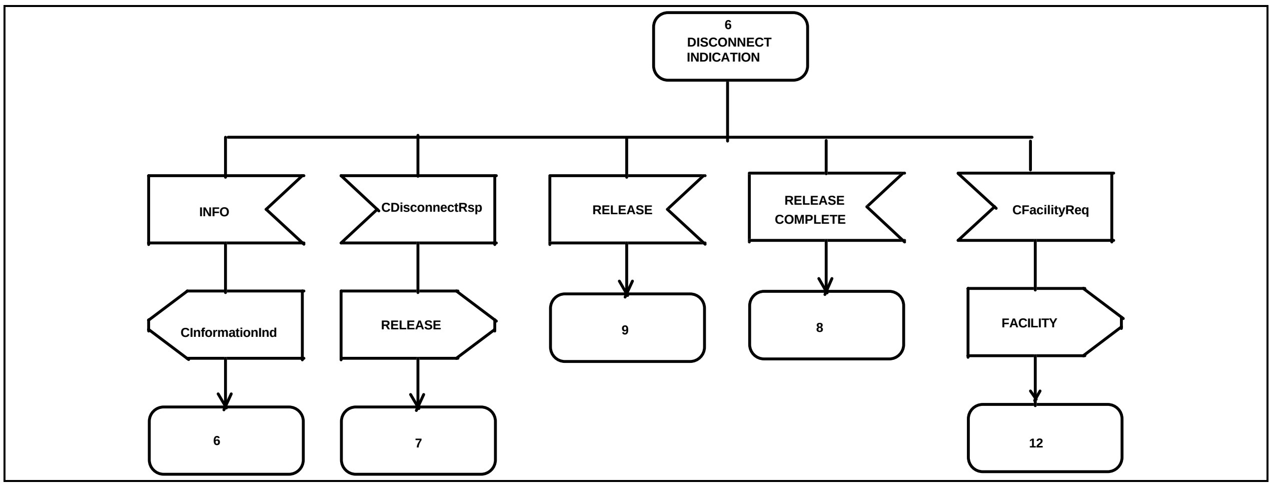


Figure E.7: DISCONNECT indication

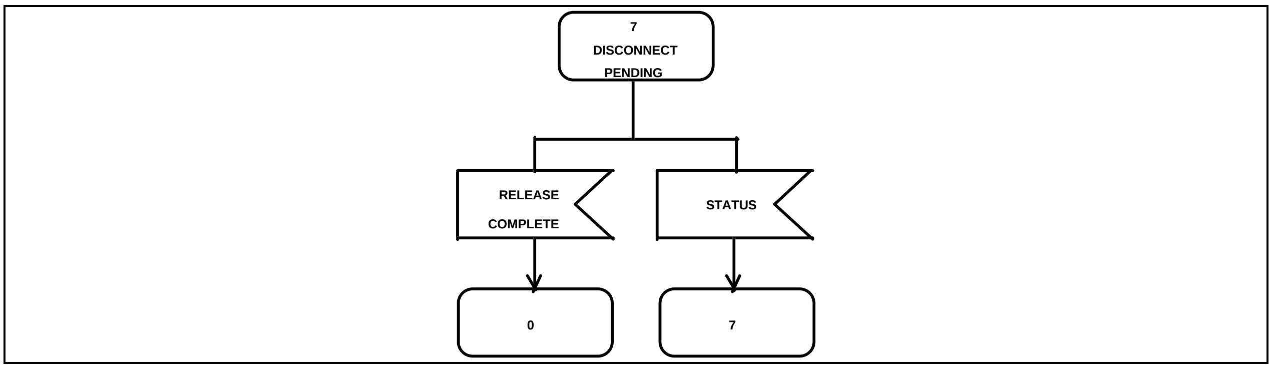


Figure E.8: DISCONNECT pending

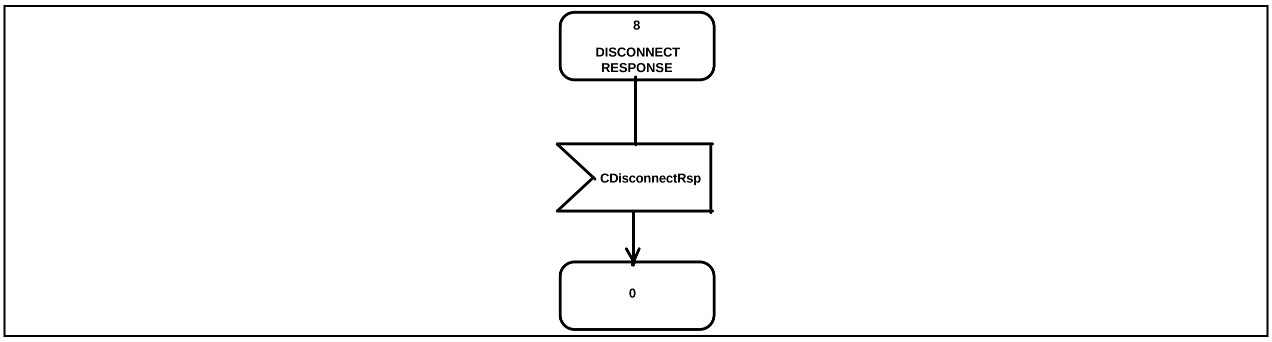


Figure E.9: DISCONNECT response

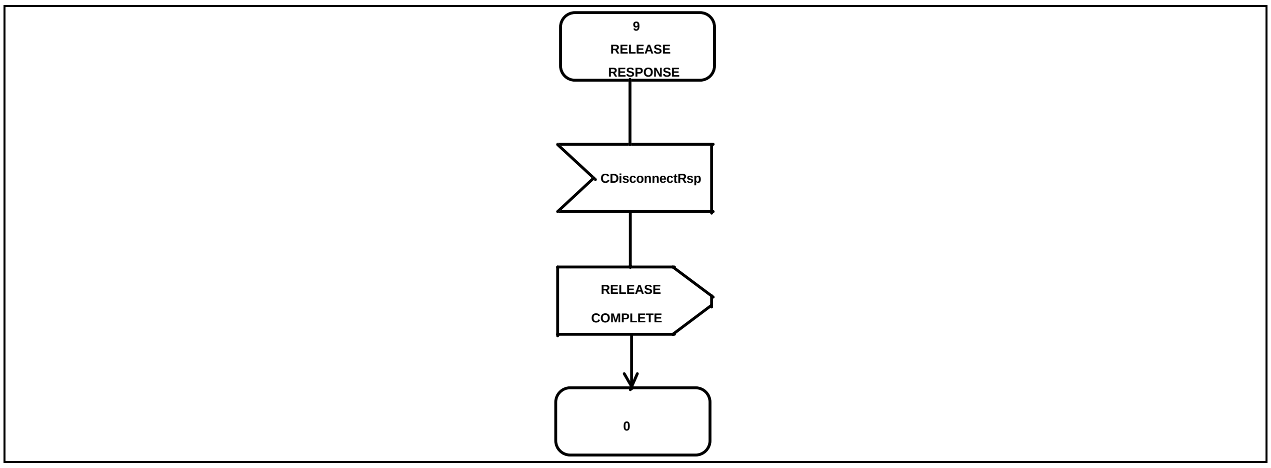


Figure E.10: RELEASE response

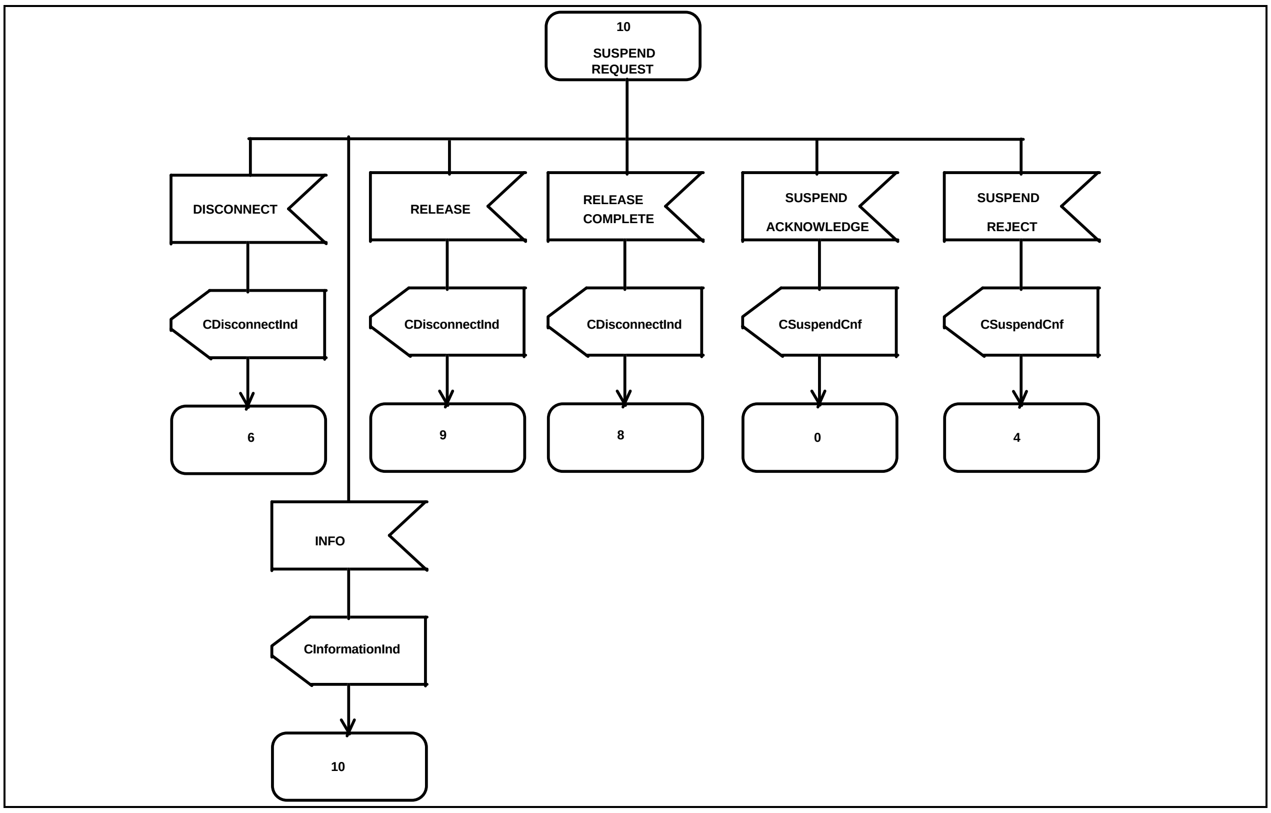


Figure E.11: SUSPEND request

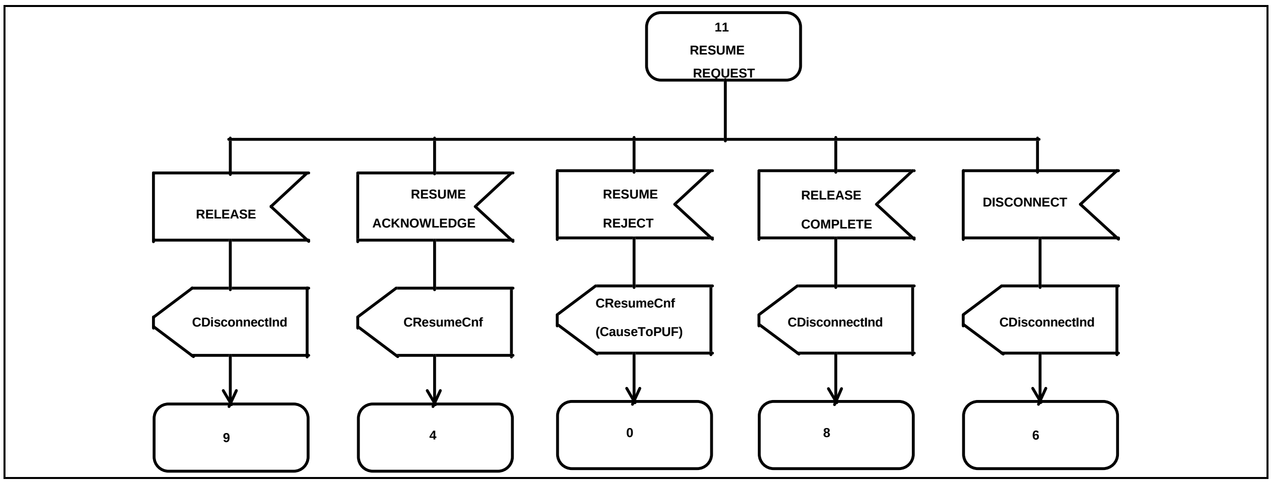


Figure E.12: RESUME request

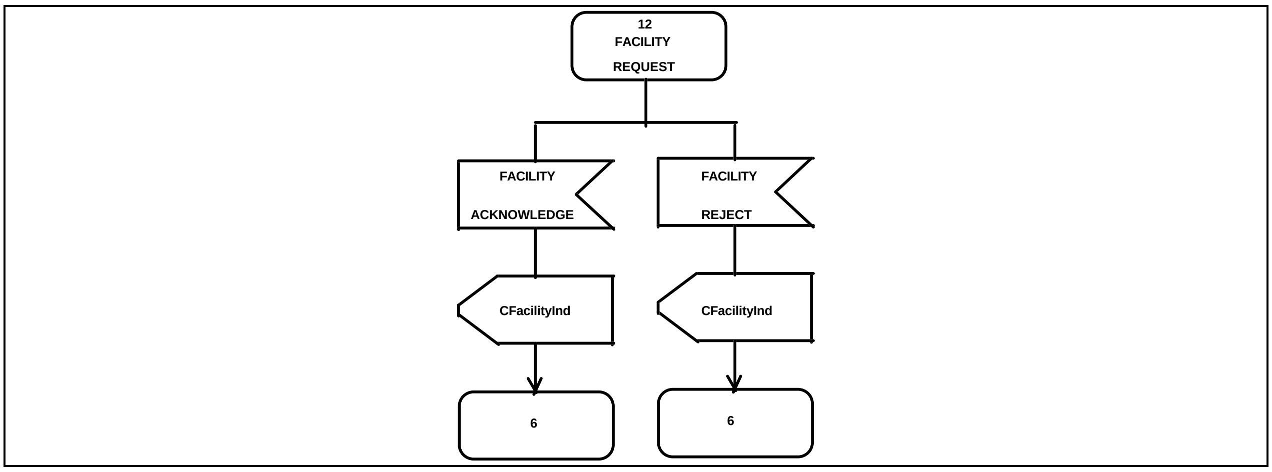


Figure E.13

E.2 Information provided by the NAF

The provision of items in messages can vary. The following conventions apply for the provision of elements by the NAF to the PUF:

- Mandatory parameters
  These items shall be provided.

- Conditional parameters
  The condition determines if they shall be provided.

- Optional parameters
  These items may or may not be provided depending if they are available to the NAF.

E.3 Co-ordination function - outgoing User Plane NMA call

The following state diagram shows the establishment of the Control Plane connection. The states indicated are internal to the NAF.

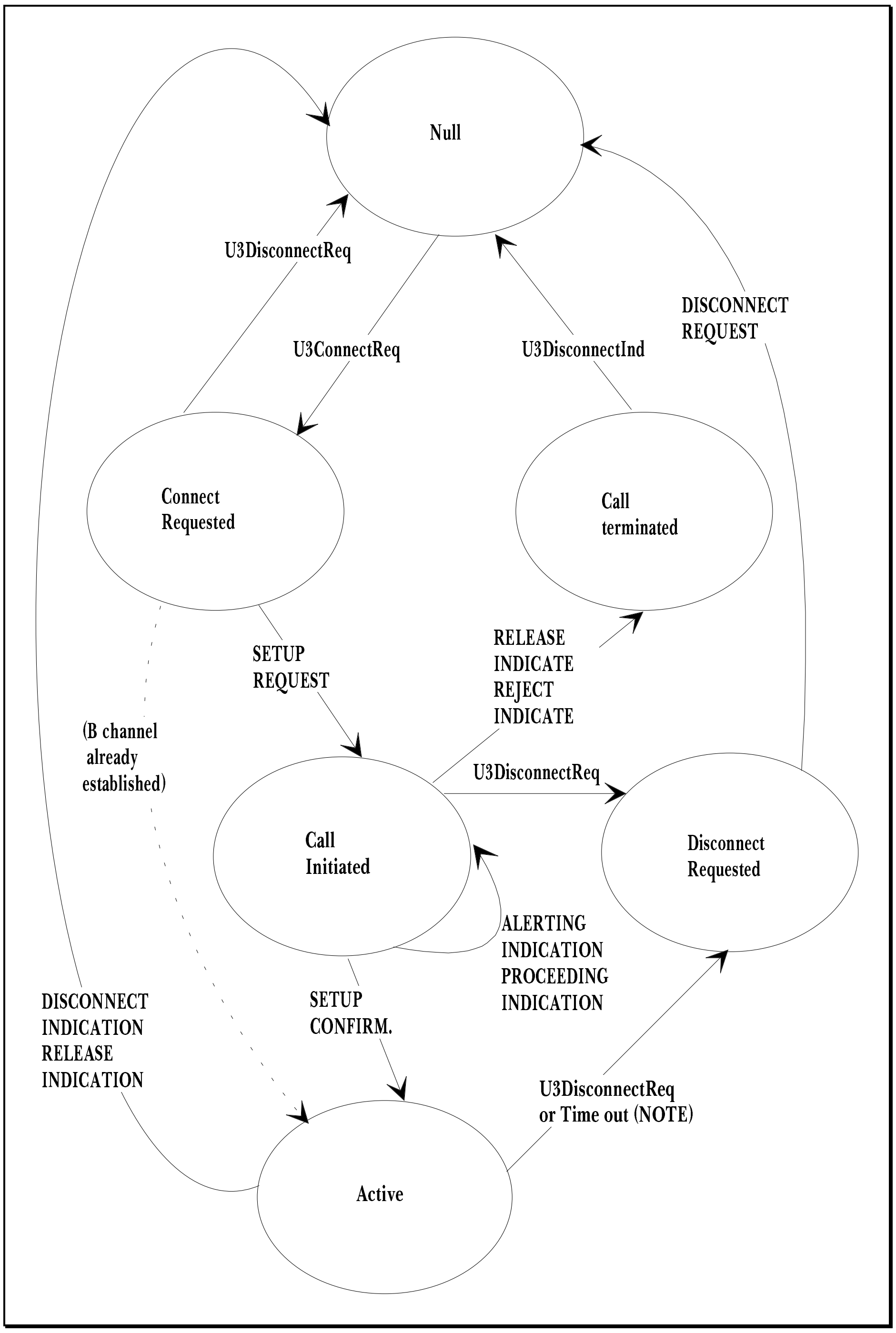

Null
U3DisconnectReq
U3ConnectReq
U3DisconnectInd
DISCONNECT REQUEST
Connect Requested
Call terminated
SETUP REQUEST
RELEASE INDICATE REJECT INDICATE
(B channel already established)
U3DisconnectReq
Call Initiated
Disconnect Requested
ALERTING INDICATION PROCEEDING INDICATION
DISCONNECT INDICATION RELEASE INDICATION
SETUP CONFIRM.
U3DisconnectReq or Time out (NOTE)
Active

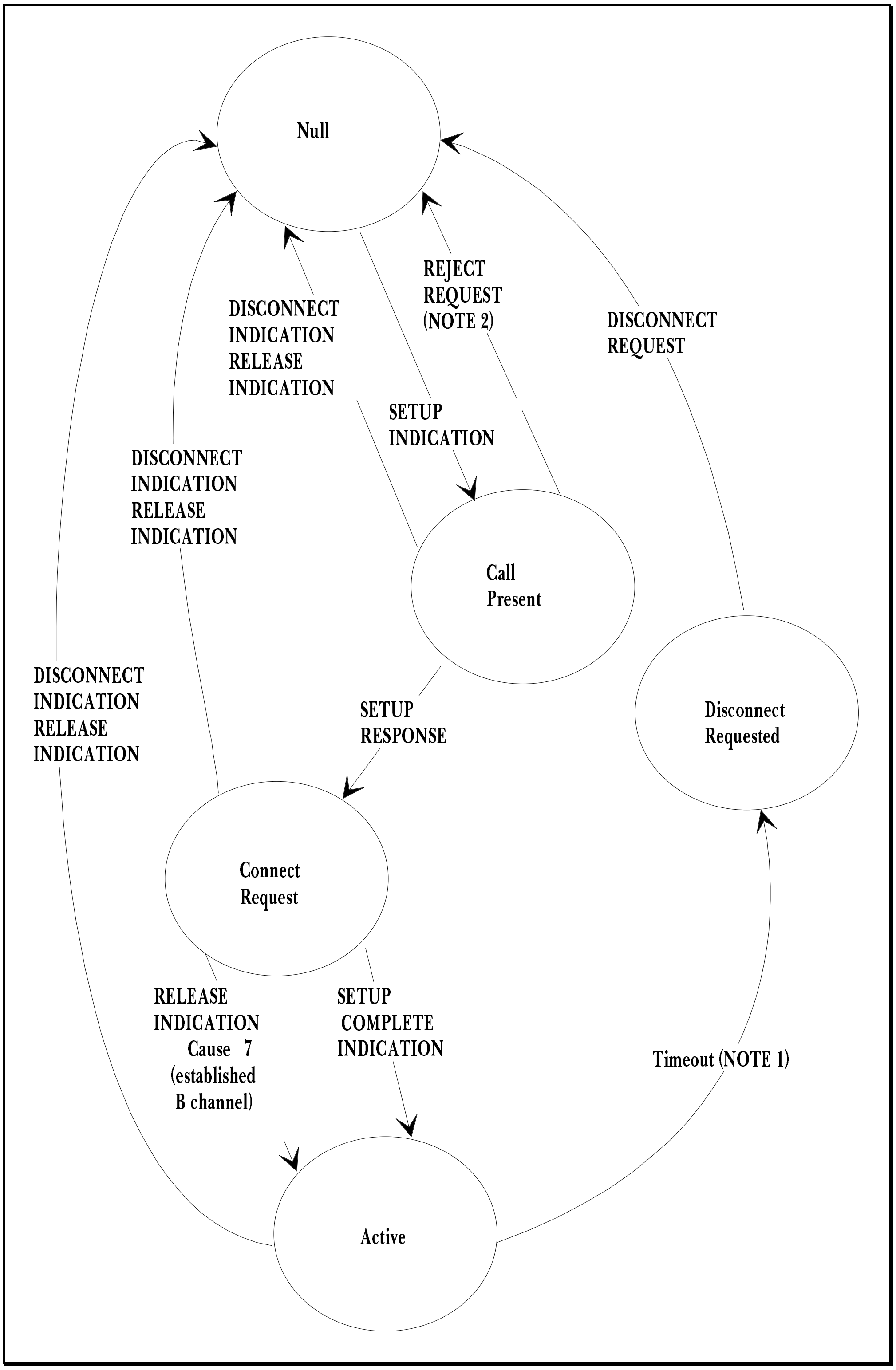
Figure E.14: Co-ordination function - NMA outgoing call and channel establishment

Remarks:
- events in upper case are primitives described in ETS 300 102-2;
- events in mixed case are PCI User Plane messages;
- the states shown are internal to the NAF.

NOTE: Whether the NAF disconnects the ISDN connection following the disconnection of the last User Plane connection on the ISDN connection or sets a time-out is a NAF design consideration.

E.4 Co-ordination function - incoming ISDN call

The following state diagram (figure E.15) shows the establishment of the Control Plane connection. The states indicated are internal to the NAF.


Null
DISCONNECT INDICATION RELEASE INDICATION
REJECT REQUEST (NOTE 2)
DISCONNECT REQUEST
SETUP INDICATION
DISCONNECT INDICATION RELEASE INDICATION
Call Present
DISCONNECT INDICATION RELEASE INDICATION
SETUP RESPONSE
Disconnect Requested
Connect Request
RELEASE INDICATION Cause 7 (established B channel)
SETUP COMPLETE INDICATION
Timeout (NOTE 1)
Active

Remarks:

- events in upper case are primitives described in ETS 300 102-2;
- the states shown are internal to the NAF.

NOTE 1: Whether the NAF disconnects the ISDN connection following the disconnection of the last User Plane connection on the ISDN connection or sets a time-out is a NAF design consideration.

NOTE 2: The NAF may reject the ISDN connection request.

Figure E.15: ISDN incoming call and coordination function

E.5 Suspending/resuming calls

The NAF is required to manage the mapping of NCOID to the call identity which is required by the network when resuming a call. Once the connection is resumed the NAF should ensure the mapping of the NCOID to the Call reference, which may have changed, on the network side.

E.6 Supplementary services

Advice Of Charge (AOC)

If the network supplies AOC without activation by the PUF, the NAF supplies both the transparent coding and the special coding for AOC. For this two facility information elements are sent to the PUF.

In the case of special AOC coding:

- if the PUF asks for AOC, the network may send a return result component in a facility information element specifying that charging information follows. The NAF handles this return result component. It is not handed back to the PUF;

- the NAF handles the different billing identifications. The PUF is not informed of the identification of the charging information;

- currency information can be mapped by using the amount specified in the facility information element coming from the network. The actual value for the value subfield can be calculated by multiplying CurrencyAmount and Multiplier and making the defined fixed point number from the result.

  The Currency is not interpreted. This information should be local knowledge between PUF and NAF;

- unit information can be mapped by using the NumberOfUnits specified in the facility information element coming from the network. The value should be converted to a fixed point number and put in the value subfield;

- if multiple charging information is available within one facility information element, the NAF can map these to several facility information elements, each containing the information for one type of charging. So, if different types of charging units apply, the recorded number of charging units for each type is mapped to a facility information element.

  Which types are available and what relation they have is a local arrangement. The NAF can use any possible local knowledge it has to normalise the units before presenting them to the PUF;

- if the NAF cannot map the AOC information completely because the local knowledge is insufficient, it can indicate this by using "unknown" for the TypeOfCharge subfield. The corresponding value can be mapped using the previous described mappings. So if the NAF cannot normalise the units for different types of charging units, the NAF may indicate this using the "unknown" value.

E.7 Error management

The error indication provided to the PUF only contains sufficient information for the PUF to judge if it is worth continuing with a particular action or not. It is envisaged that more detailed information concerning a particular error will be reported by the NAF, in a NAF specific manner. For example, a NAF may choose to implement an error log in the form of a file where it records detailed information concerning a particular error which provides the information required to debug a PUF which is under development.

The following subclauses provide guidance as to under which conditions the NAF should return a particular error to the PUF.

For messages, in the case of parameters that are repeated where repetition is not allowed, only the first valid number of repetitions of the parameter are processed, further repetitions are ignored.

If an optional parameter is given by the network, the NAF is in charge to provide it to the PUF in the relevant message.

E.7.1 Function return codes

The description of under which conditions these should be issued is described in subclause 6.8.6.

E.7.2 Administration Plane

The description of under which conditions these should be issued is described in subclause 6.8.7. For the ACreateNCOReq the number of possible parameter combinations makes checking complex. It should be approached in the order shown in table E.1.

Table E.1: Checking of ACreateNCOReq message

Parameter	Test	Action
All parameters	Not allowed	InvalidParameter error
	All valid	Continue
NCOType	missing	MissingParameter error
	invalid length	InvalidParameterLength error
	Invalid value	InvalidNCOType error
	Valid value	Continue
Direction	missing	MissingParameter error
	invalid length	InvalidParameterLength error
	Invalid value	InvalidDirectionType error
	Valid value	Continue
AttributeName	missing	AttributeNameMissing error
	invalid length	InvalidParameterLength error
	invalid	AttributeNameError error
	correct	Continue
Attribute or address content	missing	MissingParameter error
	invalid length	InvalidParameterLength error
	invalid	InvalidContent error
	correct	Continue
NafCoordination	present but not required	InvalidParameter error
	invalid length	InvalidParameterLength error
	invalid value	InvalidCoordValue error
	correct	Continue

(continued)

Table E.1: Checking of ACreateNCOReq message (concluded)

Parameter	Test	Action
GroupID	present but not required	GroupIDError error
	missing	GroupIDError error
	invalid length	InvalidParameterLength error
	invalid value	InvalidGroupID error
	correct	Continue
RequestID (if present)	invalid length	InvalidParameterLength error
	present	Process message

E.7.3 Control Plane

The errors returned in the Cause parameter match those in the ETS 300 102-1 [3] Cause information element. This allows the NAF to pass information from the Cause information element into the Cause parameter. If this is done the NAF should map any information element values to parameter values as defined in Annex B.

The following errors should be generated by the NAF as a result of checking parameters on messages passed from PUF to NAF.

Table E.2

Value	ETS 300 102-1 [2] Meaning	PCI Meaning	Generated by	When received from ISDN Processed by
1	Unallocated (unassigned) number		ISDN	PUF
2	No route to specified transit network		ISDN	NAF (NOTE 1)
3	No route to destination		ISDN	PUF
6	Channel not acceptable		ISDN	NAF (NOTE 1)
7	Call placed on an already established channel		ISDN	PUF
16	Normal call clearing		ISDN	PUF
17	User busy		ISDN	PUF
18	No user responding		ISDN	PUF
19	No answer from user (user alerted)		ISDN	PUF
21	Call Rejected		ISDN	PUF
22	Address changed		ISDN	PUF
26	Non selected user clearing		ISDN	PUF
27	Destination out of order		ISDN	PUF
28	Invalid address format	Parameter has invalid address format	NAF, ISDN	PUF
29	Facility rejected	Facility is not provided by this NAF	NAF, ISDN	PUF
30	Response to STATUS ENQUIRY		ISDN	NAF
31	Normal unspecified		ISDN	PUF
34	No circuit/channel available	Temporarily no channel of requested type is available from this NAF	NAF, ISDN	PUF
38	Network out of order		ISDN	NAF (NOTE 1)
41	Temporary failure		ISDN	NAF (NOTE 1)

(continued)

Table E.2 (continued)

Value	ETS 300 102-1 [2] Meaning	PCI Meaning	Generated by	When received from ISDN Processed by
42	Switching equipment congestion		ISDN	PUF
43	Access information discarded		NAF, ISDN	PUF (NOTE 3)
44	Requested channel/circuit not available	No channel of requested type is available from this NAF	NAF, ISDN	PUF
47	Resource unavailable, unspecified	Requested external equipment is not available	NAF, ISDN	PUF
49	Quality of service unavailable		ISDN	PUF
50	Facility requested on Facility parameter is not subscribed		ISDN	PUF
57	Bearer Capability not authorised		ISDN	PUF
58	Bearer Capability not presently available	Service requested by BearerCap is not available. In use by another PUF	NAF, ISDN	PUF
63	Service or option not available, unspecified		ISDN	PUF
65	Service requested by Bearer Capability is not implemented	Service requested by BearerCap Parameter is not provided by NAF	NAF, ISDN	PUF
66	Channel Type not implemented	NAF does not support this type of channel	NAF, ISDN	PUF
69	Facility requested is not implemented	NAF does not support this facility	NAF, ISDN	PUF
70	Only restricted digital information bearer capability is available		ISDN	NAF (NOTE 1)
79	Service or option not implemented, unspecified		ISDN	PUF
81	Invalid call reference	Invalid NCOID	NAF, ISDN	NAF (NOTE 1)
82	Identified channel does not exist	Identified permanent channel is not defined	NAF, ISDN	NAF (NOTE 1)
83	A suspended call exists but this call identity does not		ISDN	NAF (NOTE 1)
85	No call suspended	NCOID does not identify a suspended connection	NAF, ISDN	NAF (NOTE 1)

(continued)

Table E.2 (concluded)

Value	ETS 300 102-1 [2] Meaning	PCI Meaning	Generated by	When received from ISDN Processed by
86	Call having requested call identity has been cleared		ISDN	NAF (NOTE 1)
88	Incompatible destination		ISDN	PUF
91	Invalid transit network selection		ISDN	NAF (NOTE 1)
95	Invalid message, unspecified		ISDN	NAF (NOTE 1)
96	Mandatory parameter is missing	Mandatory parameter is missing	NAF, ISDN	NAF (NOTE 1)
97	Message Identifier non-existent or not implemented on this network	Message Identifier non-existent or not implemented on this NAF	NAF, ISDN	NAF (NOTE 1)
98	Message not compatible with call state or message identifier non-existent or not implemented.	Message not compatible with NCO state or message identifier non-existent or not implemented.	NAF, ISDN	NAF (NOTE 1)
99	Invalid parameter	Invalid parameter	NAF, ISDN	NAF (NOTE 1)
100	Invalid parameter contents	Invalid parameter contents	NAF, ISDN	NAF (NOTE 1)
101	Message not compatible with current state	Message not compatible with current state	NAF, ISDN	NAF (NOTE 1)
102	Recovery on timer expiry		ISDN	NAF (NOTE 2)
111	Protocol Error, unspecified		ISDN	NAF (NOTE 1)
127	Interworking, unspecified		ISDN	PUF

NOTE 1: Where cause values are processed by the NAF. The NAF should attempt to error recovery. If it fails to recover it should indicate to registered PUFs that it is no longer available by the use of the NAFNotAvailable error code.

NOTE 2: NAF should take appropriate action.

NOTE 3: In the case of ISDN generating this cause, it is the responsibility of the NAF to map information element to parameter types in any diagnostic information supplied to PUF.

Table E.3 shows the order of checking for CConnectReq. Information is taken from the CConnectReq message plus the attribute and address sets associated with the mandatory network connection identifier. The table assumes that the initial checking of the message has taken place.

Table E.3: Checking of CConnectReq message

Parameter	Test	Action
NCOID	invalid	CStatusInd Cause parameter value = 81
	valid	Continue
Message state	invalid	CStatusInd Cause parameter value = 101 Diagnostics = MessageID
	valid	Combine parameters from attribute set, address set and CConnectReq message, continue
Mandatory Parameters	BearerCap missing	CDisconnectInd Cause parameter value = 96 Diagnostics = BearerCap
	BearerCap Service is not X25 and CalledNumber missing	CDisconnectInd Cause parameter value = 96 Diagnostics = CalledNumber
	All present	Continue
BearerCap Parameter Content	Invalid contents	CDisconnectInd Cause parameter value = 100 Diagnostics = BearerCap
	Service not available from NAF	CDisconnectInd Cause parameter value = 65
	correct	Continue
CalledNumber Parameter Content (if present)	invalid contents	CDisconnectInd Cause parameter value = 100 Diagnostics = CalledNumber
	correct	Continue
Unrecognised Parameters	present	CDisconnectInd Cause parameter value = 99 Diagnostics = Parameter Type of unrecognised parameter
	not present	Continue

(continued)

Table E.3: Checking of CConnectReq message (concluded)

Parameter	Test	Action
Optional Parameter Content Error	present	CStatusInd Cause parameter value = 100 Diagnostics = Parameter Type of parameter in error Continue (ignore parameter)
	not present	Process Message

E.7.4 NMA User Plane

The error processing for this plane is defined in ISO 9574.

For the U3ConnectReq in the case of the NAF not providing the co-ordination function the checking shown in table E.4 should be performed once parameters from the NCO and message have been combined.

Table E.4: Checking of U3ConnectReq message

Parameter	Test	Action
NCOID	missing	U3DisconnectInd
	invalid parameter length	U3DisconnectInd
	invalid NCOID	U3DisconnectInd
	valid	Continue
Mandatory Parameters	Both BCUG and CalledDTEAddress missing	U3DisconnectInd
	Both BCUG and CalledDTEAddress present	U3DisconnectInd
	BCUG or CalledDTEAddress present	Continue
If CalledDTEAddress present	Content error	U3DisconnectInd
	valid	Continue
If BCUG present	Content error	U3DisconnectInd
	valid	Continue
Other Parameters	Unrecognised	U3DisconnectInd
	Invalid Contents	U3DisconnectInd
	valid	Process Message

E.7.5 TMA User Plane

The error processing for this plane is relatively simple due to the number of messages and parameters supported.

For the U1DataReq message the processing is defined in table E.5.

Table E.5: Checking of U1DataReq message

Parameter	Test	Action
NCOID	missing	U1ErrorInd, error = MissingParameter
	invalid parameter length	U1ErrorInd, error = InvalidParameterLength
	invalid NCOID	U1ErrorInd, error = InvalidNCOID
	valid	Continue
Other Parameters	present	U1ErrorInd, error = MissingParameter
	not present	Process Message

E.8 NAF configuration

The following subclause contains information concerning NAF configuration. This subclause is provided to assist NAF developers and is not intended to be a comprehensive list of configurable items.

E.8.1 Global Configuration

Table E.6: Global Configuration

Parameter	Suggested Default	Comment
Number of PUFs supported	8	

E.8.2 System configuration parameters

Table E.7: System configuration parameters

Parameter	Suggested Default	Comment
DMA		DMA number used by the adapter
I/O address		I/O address used by the adapter
IRQ		IRQ used by the adapter
DRAM		Double RAM access address shared between the adapter and the host environment. This parameter may also contain the size of the frame to be used by the adapter

E.8.3 Control Plane configuration

Table E.8: Control Plane configuration

Parameter	Suggested Default	Comment
Number of D-Channels	1	
D-Channel definitions - type of network - automatic - fixed + number - frame window size (K) - N200 - N201 - N 202 Timers, - T200 - T201 - T202 - T203	1	
Number of B-channels	2	
Number of Permanent B-channels	0	
List of permanent B-channel identifiers	1..256	
Number of permanent (SAPI 16) D-Channels For each D-channel - automatic - fixed + number - same as signalling	0	

E.8.4 User Plane NMA configuration

Table E.9: User Plane NMA configuration

Parameter	Suggested Default	Comment
X25 Network Type	0	Allows NAF to adapt for different country implementations of X.25.
X25 Recommendation	CCITT88	Level of X.25 Recommendation supported.
Layer 3 sequence numbering	8	
Layer 3 Maximum Window Size	7	
Layer 3 Default Window Size	3	
Layer 3 Maximum Packet Size	4096	
Layer 3 Default Packet Size	128	
Layer 3 Default Connection Mode	Auto	Auto - Act as DTE when calling, act as DCE when called. DXE - use Restart Packet to determine DTE or DCE role as in ISO 8208 [3]. DTE - Act as DTE. DCE - Act as DCE.
Lowest number of Incoming SVC (LIC)	1	
Highest number of incoming SVC (HIC)	1	
Lowest number of Two way SVC (LTC)	0	
Highest number of Two way SVC (HTC)	0	
Lowest number of outgoing SVC (LOC)	0	
Highest number of outgoing SVC (HOC)	0	
Layer 3 Timers		NAF may wish to provide PUF user the ability to configure timers.
Layer 2 Default Connection Mode	Auto	Auto - When calling act as DTE, when called act as DCE. DTE as defined in ISO 7776 [4]. DCE as defined in ISO 7776 [4].
Layer 2 B channel modulus	8	NOTE shall be 128 for X.25 on D-channel.

(continued)

Table E.9: User Plane NMA configuration (concluded)

Parameter	Suggested Default	Comment
Layer 2 Window Size	7	
Layer 2 Frame Size	128	
Layer 2 activation type	Case 1	Case1 - send SABM/SABME when calling, do not send when called. Case2 - send SABM/SABME when called, do not send when calling. Passive - do not send SABM/SABME when initiated. Active - send SABM/SABME when initiated.
Layer 2 Timers		
- T1	5	Expressed in second.
- T2	1	Expressed in second.
- N2	5	Maximum number of unsuccessful retransmission.

E.8.5 User Plane TMA configuration

Table E.10: User Plane TMA configuration

Parameter	Suggested Default	Comment
Idle Flag default value	0xFF	

E.9 Buffer management

Buffers passed by the PUF to the NAF are copied by the NAF into internal space as soon as provided. So, the buffers may be reused by the PUF immediately after the function returns.

The exact instant when the message is processed is dependent on the NAF and is outside the scope of this ETS.

In the PUF to NAF direction, the message and the data associated with, if any, are provided together, in one step. If one of the message or the data buffers is too small to contain, respectively, a complete message or the data information, the NAF returns an error and the message is not be provided to the PUF. To help the PUF, the size of the biggest message is established during the registration phase. The size of a data buffer is closely dependent of the type of connection and its protocol. The PUF refers to the protocol initialisation to get the correct value of the biggest data packet.

If a NAF needs to make internal buffering action, it is in charge to get buffers by itself. This can be achieved via a configuration operation, provided by the NAF manufacturer, which is outside the scope of this ETS. The NAF manufacturer may describe how the operation can be realised and which consequences are expected.

E.10 Extension of ISDN-PCI

This Clause describes how the ISDN PCI can be extended. It will indicate which possibilities for extension are available throughout the ISDN PCI. The major part of the description is concerned with the extension of the protocols supported by the ISDN PCI.

E.10.1 Basic mechanism for extension

This subclause identifies the basic mechanisms for extension provided in the ISDN PCI. The following mechanisms are available in the ISDN PCI for extension:

- use of Manufacturer specific messages;
- use of message coding;
- incorporation of new protocols.

The first two mechanisms are provided within the ISDN PCI and provides for minor extension of the functionality of the ISDN PCI. The last mechanism provides a way to extend the functionality of the ISDN PCI to new protocols, not yet covered in this version of the ISDN PCI.

The following subclauses cover the mechanisms in more detail.

E.10.2 Manufacturer specific messages

The Manufacturer specific messages, provided in the Administration Plane of the ISDN PCI provide the basic mechanism for making small extensions. The Manufacturer specific messages can be found in subclauses 6.2.12 and 6.2.13. The extension by Manufacturer specific messages is completely within the scope of the ISDN PCI and implementations using this mechanism conform to this ETS.

By implementing these messages a NAF manufacturer can extend the ISDN PCI to give the PUF access to specific functionality provided by the NAF manufacturer below the ISDN PCI. In this way a NAF manufacturer can give, for example, a PUF access to special hardware connected to the ISDN interface board.

Although the main purpose of the Manufacturer specific messages is to provide a mechanism to incorporate specific local management, there is no limit to the use of the Manufacturer specific messages.

E.10.3 Message coding

The message coding provides for a second mechanism to extend the ISDN PCI. Care should be taken in using this mechanism because it is outside the scope of the ISDN PCI. Implementations using the mechanism do not conform to this ETS.

The ISDN PCI is intended to be an open interface. This is reflected in the definition of the messages. All possible care has been taken to give access to all the important information elements provided with EuroISDN and the supported User Plane protocols.

If additional access to information elements is needed an additional parameter can be added. The coding mechanism provided within the ISDN PCI can be used for this extension.

Within the ISDN PCI all parameters are coded using Type Length and Value (TLV). Please refer to subclause 6.5 for information on the TLV coding. By defining a new TLV coded parameter and adding this to the message, the necessary information can be transported by the PUF to the NAF. The exchange mechanism of the ISDN PCI has no notion of the information it transports and will, therefore, also transport the new parameter provided in the message buffer.

When using this mechanism to extend the ISDN PCI the possibility to port the PUF to a different NAF may be lost. The error mechanism of the ISDN PCI is defined to return an error on invalid parameters and the PUF which sends a newly defined parameter may, therefore, receive an error when running on NAF which does not support this parameter.

E.10.4 Extension of supported protocols

The ISDN PCI is intended to provide access to the data functionality in the ISDN D- and B-channels. For this, an interface to several protocols has been identified and has been incorporated in the ISDN PCI. The protocols described in the ISDN PCI are all related to CCITT Recommendation X.213 [7].

The ISDN PCI does not contain a specific mechanism for extending the functionality of the ISDN PCI with additional protocols. Therefore, extending the ISDN PCI with additional protocols should be regarded as revising the ETS.

The actual extension of an implementation of the ISDN PCI in this way is straightforward. Some of the new protocols may even be incorporated transparently into the ISDN PCI. The same mechanism, as described in subclause E.10.3, can be used for doing the actual extension. New parameters and even new messages can be added to an implementation of the ISDN PCI. When extending the ISDN PCI in this way a distinction should be made between "CCITT Recommendation X.213 [7]" like and non-"CCITT Recommendation X.213 [7]" like protocols. In the next two subclauses these two types of protocols are covered in greater detail.

E.10.4.1 Support of new "CCITT Recommendation X.213" like protocols

The basic operation on the User plane is based upon an "CCITT Recommendation X.213 [7]" like interface to the protocols. By mapping the different protocols to the "CCITT Recommendation X.213 [7]" like interface, these protocols can be supported. Within the definition of the User Plane messages and parameters it is indicated how this mapping has been performed for the different protocols.

When extending the ISDN PCI with another protocol which can be mapped to the "CCITT Recommendation X.213 [7]" like interface the same exercise needs to be repeated. It should be made clear which messages and parameters are applicable for the new "CCITT Recommendation X.213 [7]" like protocol.

If this mapping can be performed without changing the described messages and parameters, the new protocol can even be supported transparently.

E.10.4.2 Support of different type of protocols

For protocols which cannot be mapped to a "CCITT Recommendation X.213 [7]" like interface the extension will be more complicated.

Currently two types of User Plane accesses have been identified. The first is based on transparent access to the ISDN B-channel and is called the TMA. See subclauses 5.3.4.2 and 6.4 for more information. The second access is based on the "CCITT Recommendation X.213 [7]" like interface and is called NMA. See subclauses 5.3.4.1 and 6.4 for more information.

When a non-"CCITT Recommendation X.213 [7]" like protocol has to be supported a new access has to be defined to cover this protocol. Such an extension will be a major revision of the ISDN PCI.

When making the definition of this new access, several kinds of information need to be provided:

- messages in the User Plane;
- parameters for these messages;
- a description of the sequencing of the new messages like provided in clause 6.4.1 for the NMA;
- attributes for this protocol in the ACreateNCOReq (subclause 6.2.1);
- return values and error applicable for the new protocol;
- static attribute sets.

The implementation of a new protocol within the ISDN PCI is the same mechanism, as described in subclause E.10.3 which can be used for doing the actual extension. New parameters and new messages can be added to an implementation of the ISDN PCI. The exchange mechanism allows the exchange of the new information between the PUF and NAF.

Annex F (normative): Operation system specific implementation

This annex describes the operating system specific implementation. The operating systems addressed are:

DOS	stands for operating systems compatible to the MS-DOS[1] operating system. The base MS-DOS version is the version number 3.1.
UNIX	stands for the UNIX[2] operating system. The based version is the version number: System V, Release 3.
Windows	stands for the Windows[3] operating system. The based version is the version number 3.0.

F.1 DOS

A NAF implementation under DOS shall offer the functionality of the exchange functions described in a generic way in Clause 7.

In this annex, the mapping and implementation of these functions is described on a function per function basis. For each function, a coding example in C language is given. Since the implementation of the functions is DOS system dependent, the C language used is based on Microsoft C Version 5.x or higher.

F.1.1 Mechanism

Except for the function **PciGetHandles**, the implementation of the exchange method for DOS is based on a direct access mechanism. The access point is a far function address provided by the NAF. This function address is mapped to the generic type PCI_HANDLE.

To make sure that the function address provided by the NAF is correct, the PUF shall check a signature located in front of the function address before calling the NAF.

To perform this check, the PUF shall examine the memory area located just in front of the function address. The signature is located there which shall contain the eight character constant "ISDN PCI". If this signature is available, the PUF can assume the NAF function address is correct.

Only one access point shall be provided by the NAF. A supplied parameter shall indicate the function to be invoked. This parameter is named **function code**.

Parameters are passed from the PUF to the NAF using the stack. The PUF shall ensure a minimum stack space of 128 bytes on call. When the NAF receives the control of the CPU, the first parameter on the stack is the function code, followed by parameters based on the particular function.

The function code is passed as a 2 byte integer value.

The NAF shall place the return code in the AX register. The NAF procedure is not in charge of cleaning the stack on return. The C call convention is used: the calling PUF pushes parameters right to left and restores the stack on return.

The alignment of the PCIMPB generic structure is **byte**.

1) MS-DOS is a trademark of the Microsoft Corporation.
2) UNIX is a trademark of AT&T.
3) Windows is a trademark of the Microsoft Corporation.

F.1.2 Mapping of generic types and constants

Under DOS, the following mapping shall be used for the generic types described in Clause 7:

Generic Type	DOS Mapping
PCI_INTEGER	2 byte integer (a word)
PCI_BYTEARRAY	far pointer (segment :offset address)
PCI_EXID	Unique identifier provided by NAF (2 byte integer)
PCI_HANDLE	far function address (segment : offset address)
PCI_PROCEDURE	far function address (segment : offset address)

As usual for DOS, all values are in little endian (low byte - high byte) order.

The **function code**, used to invoke the exchange functions, shall be assigned as follows:

Function	Function code value
PciGetProperty	1
PciRegister	2
PciDeregister	3
PciPutMessage	4
PciGetMessage	5
PciSetSignal	6

C presentation of these definitions looks as follows:

```
/*
 * Generic type mappings
 */
typedef short int           PCI_INTEGER;
typedef char far *                  PCI_BYTEARRAY;
typedef short int           PCI_EXID;
typedef short int (far *    PCI_HANDLE) ();
typedef void                (far *      PCI_PROCEDURE) ();

/*
 * Function code constants
 */
#define PCIGETPROPERTY              1
#define PCIREGISTER                 2
#define PCIDEREGISTER               3
#define PCIPUTMESSAGE               4
#define PCIGETMESSAGE               5
#define PCISETSIGNAL                        6

/*
 * Signature
 */
#define PCISIGNATURE                'ISDN PCI'      /* multi characters constant */
```

F.1.3 Description of functions

The PUF is in charge to provide a minimal stack during a function call. The minimal stack size is 128 bytes.

In the description, the access to one is describe for simplicity in the coding examples. However the access of a PUF to multiple NAFs is not excluded. How access to multiple NAFs can be achieved is shown in Annex J.

F.1.3.1 PciGetHandles

Under DOS, the implementation of the **PciGetHandles** function shall use a character device driver named "PCIDD$" to retrieve the available PCI-Handles. This function call is the exception on the basic principle - direct access - under DOS.

The maximum theoretical amount of PCI-Handles which can be retrieved is 4 096. However, the implemented device driver will probably have a practical limit which lies far below and depends on the implementation of the device driver itself.

The following operation shall be performed by the PUF, in order:

- open the "PCIDD$" character device driver;
- prepare a buffer in memory, big enough to hold the maximum amount of PCI-Handles to be retrieved;
- issue a IOCTL system read call: Receive Control Data from Character Device;
  - BX shall contain the dos handle of the device driver;
  - CX shall contain the length of the memory buffer prepared above;
  - DS:DX shall point to the memory buffer;
- check the success of the operation (check carry flag);
- in case of error, optionally issue a DOS Get Extended Error function call to receive a more comprehensive error code;
- on successful return, AX contains the number of bytes provided by the device driver, the buffer contains the available PCI-Handles in a row. The number of available PCI-Handles is calculated by dividing the AX value by 4, the size of a far address function pointer;
- close the device driver.

<u>C coding example:</u>

```
...
#include        <dos.h>         /* declarations for IOCTL call */
#include        <fcntl.h>       /* declarations for open mode */
...
#define SUCCESS        0        /* No error */
#define MAXHANDLES 64           /* max amount of handles to be read */
...
PCI_HANDLE PCIHandles[MAXHANDLES]       /* buffer for receiving PCI-Handles */
...
PCI_INTEGER MaxHandles;                 /* max amount of handles to be read */
PCI_HANDLE far * PCIHandles; /* far pointer to buffer of PCI-Handles */
PCI_INTEGER far * ActualHandles;        /* far ptr to amount of PCI-Handles received */
{
int fildes;                             /* file descriptor */
int error;
union _REGS regs;
struct _SREGS segregs;
struct _DOSERROR errorinfo;
```

```
                                        /* open the driver */
if (_dos_open ("PCIDD$", _O_RDWR, &fildes) != SUCCESS)
    {
                                        /* device driver not accessible; perform error processing */
    error = ...
    }
else
    {
                                        /* prepare IOCTL read from device driver */
    _segread (&segregs);
    segregs.ds = FP_SEG (PCIHandles);       /* set-up segment address */
    regs.x.dx = FP_OFF (PCIHandles);        /* and offset */
    regs.x.cx = MaxHandles * sizeof(PCI_HANDLE);
    regs.x.bx = fildes;                     /* set dos file handle */
    regs.x.ax = 0x4402;                     /* IOCTL read from character device */

                                            /* issue IOCTL read from device driver */
    _intdosx (&regs, &regs, &segregs);

                                            /* close the driver */
    _dos_close (fildes);

                                            /* check for error */
    if (regs.x.cflag & 1)                   /* check processors carry flag */
        {
                                            /* error has occurred; perform error processing */
        _dosexterr (&errorinfo);
        error = doserror.exterror;
        ...
        }
    else
        {
        /* successful operation; set count of handles received */
        ActualHandles = regs.x.ax / sizeof(PCI_HANDLE);
        error = SUCCESS;
        }
...
```

F.1.3.2 PciGetProperty

This function is in charge to retrieve the NAF-Property from the NAF. To issue the function call, the PUF shall possess the PCI-Handle of the NAF it wants to access. Before accessing the NAF, the PUF shall check, if the PCI-Handle it uses is valid by checking the signature of the access point the PCI-Handle is pointing to.

The following operation shall be carried out by the PUF, in order:

- examine memory area pointed to by the PCIHandle to find out if NAF is loaded. Check the signature for the character constant "ISDN PCI";
- call the address with the PciGetProperty **function code** and the parameters provided by the PUF;
- check return code.

C coding example:

```
...
#include        <memory.h>                    /* memory compare func declarations */
...
#define SUCCESS             0          /* No error */
#define PCIGETPROPERTY      1
#define PCISIGNATURE        'ISDN PCI'
#define SIGNATURESIZE       8
...
PCI_HANDLE PCIHandle;
PCI_INTEGER MaximumSize;
PCI_BYTEARRAY Property;
PCI_INTEGER far * ActualSize;
{
PCI_INTEGER error;
char far * signature;

signature = (char far *) PCIHandle - SIGNATURESIZE;
if (_fmemcmp (signature,PCISIGNATURE,SIGNATURESIZE) == SUCCESS)
        {
        /* signature is correct; call the entry point */
        error = (*PCIHandle) (PCIGETPROPERTY, MaximumSize, Property, ActualSize);
        ...
        }
else
        {
        /* signature wrong; process error */
        error = ...
...
```

F.1.3.3 PciRegister

This function is in charge to provide an association between a PUF and a NAF. To issue the function call, the PUF shall possess the PCI-Handle of the NAF it wants to access Before accessing the NAF, the PUF shall check, if the PCI-Handle it uses is valid by checking the signature of the access point the PCI-Handle is pointing to.

For this function call, 2 structures shall be prepared by the PUF and shall be passed on the function stack. The first structure is the PCIRegisterInfo structure as declared in Clause 7. The second is the operating system dependent PCIOpSysInfo structure, which, for DOS, has the following layout:

Structure Element Name	Type	Valid on Call or Return	Explanation
MaxNCOCount	2 byte integer	call	Shall be set to the maximum amount of NCOs the PUF intends to create during the association.
MaxPacketSize	2 byte integer	call	Shall be set to the maximum size of a data packet the NAF shall accept on a user connection.
MaxPacketCount	2 byte integer	call	Shall be set to the maximum amount of packets of the above size the NAF shall buffer per user connection.
AddBufferSize	4 byte integer	call	If the PUF wants to provide buffer space to the NAF, it shall set this value to the size of the buffer space it donates. Otherwise the value shall be set to zero (0).
AddBufferSpace	far address (segment : offset)	call	If the structure element AddBufferSize is non-zero, this element shall point (far) to the donated, additional buffer space.
BufferNeeded	4 byte integer	return	In case the NAF does not have enough buffer space available to guarantee the requested connection characteristics, the amount of additional buffer needed is returned into this element by the NAF.

The information provided with this structure helps the NAF to optimize its internal resources. Therefore the information given by the PUF shall be carefully weighted. This is especially true in an environment, where a NAF serves several PUFs at the same time.

In the case where a NAF does not have available enough memory resources to fulfil the requested characteristics, the PciRegister function shall fail and return a OutOfBuffers error code. In this case, the amount of buffer missing can be taken from the BufferNeeded element of the above structure.

On successful return of the PciRegister function, the Exchange-ID becomes available, which shall be used as a parameter on subsequent exchange mechanism function calls.

The following operation shall be carried out by the PUF, in order:

- examine memory area pointed to by the PCIHandle to find out if NAF is loaded. Check the signature for characters "ISDN PCI";
- allocate and set-up the two structures: PCIRegisterInfo and PCIOpSysInfo. The PCIOpSysInfo structure may, optionally, contain a pointer to additional buffer space which shall be donated to the NAF;
- call the exchange function with the PciRegister **function code** and the parameters provided by the PUF;
- check return code. If the return code indicates OutOfBuffers then the call may be repeated with correct adjusted buffer space to be donated to the NAF;
- keep the returned Exchange-ID for later calling.

<u>C coding example:</u>

```
...
#include        <memory.h>                          /* memory compare func declarations */
#include        <malloc.h>                          /* memory allocation functions */
...
#define SUCCESS                   0                 /* No error */
#define PCIREGISTER               2
#define PCISIGNATURE              'ISDN PCI'
#define SIGNATURESIZE             8
#define E_OUT_OF_BUFFERS          128               /* OutOfBuffers error code */
...
struct pci_register {                               /* structure containing registering info */
                PCI_INTEGER  PUFVersion;            /* optional: give PUF version */
                PCI_INTEGER  PUFType;               /* optional: give PUF type */
                PCI_INTEGER  MaxMsgSize;            /* return: max size of a message */
};

struct pci_opsys {                                  /* structure containing registering info */
                short intMaxNCOCount;/* optional: give max count of NCOs */
                short intMaxPacketSize;/* optional: give expected max size and */
                short intMaxPacketCount;      /* max count of packets to buffer */
                long int        AddBufferSize;      /* optional: give to NAF size and */
                void far *      AddBufferSpace;     /* pointer to additional buffer */
                long int        BufferNeeded;       /* return: amount of add buffer needed */
};
...

/*
 * before calling the PCIRegister function further down, allocate and prepare the structures
 * requested by this function call
 */

struct pci_register PCIRegisterInfo {
                1,                        /* Set PUF version to 1, equalling current ETS Version*/
                0,                        /* Set PUF type to 0 as indicated in clause 7 */
                0                         /* Initialise (expected) return value of MaxMsgSize */
};

struct pci_opsys PCIOpSysInfo {
                2,                        /* Set max amount NCOs PUF intends to create */
                1024,                     /* Set max size of data packets NAF shall accept */
                8,                        /* Set max count of packets NAF shall buffer per NCO */
                0,                        /* Set size of memory PUF wants to donate to NAF */
                (void far *) NULL,        /* Set pointer to (donated) buffer space */
                0                         /* Initialise (expected) return value of BufferNeeded */
};
...
PCI_HANDLE PCIHandle;
struct pci_register far * RegisterStruct;;
PCI_EXID far * ExchangeID
{
PCI_INTEGER error;
char far * signature;
void far * buffer;
```

```
signature = (char far *) PCIHandle - SIGNATURESIZE;
if (_fmemcmp (signature,PCISIGNATURE,SIGNATURESIZE) != SUCCESS)
        {
        /* signature wrong. process error */
        error = ...
        }
else
        {
        /* signature is correct. call the entry point */
        error = (*PCIHandle) (PCIREGISTER, &PCIRegisterInfo, &PCIOpSysInfo, ExchangeID);
        if (error == E_OUT_OF_BUFFERS)
                {
                /* NAF needs more buffer space; try to allocate */
                buffer = _fmalloc ((size_t) PCIOpSysInfo.BufferNeeded);
                if (buffer)
                        {
                        /* there is buffer, so it is worth another try; adjust PCIOpSysInfo structure */
                        PCIOpSysInfo.AddBufferSize = PCIOpSysInfo.BufferNeeded;
                        PCIOpSysInfo.AddBufferSpace = buffer;
                        PCIOpSysInfo.BufferNeeded = 0;
                        /* call PciRegister again ... */
                        }
                }
        error=(*PCIHandle)(PCIREGISTER,&PCIRegisterInfo,&PCIOpSysInfo,ExchangeID);
        if (error)
                {
                /* Process error */
                ...
...
```

F.1.3.4 PciDeregister

This function is in charge to disassociate a PUF and a NAF.

The following operation shall be carried out by the PUF, in order:

- Call the address with the PciDeregister **function code** and the Exchange-ID related to the current association.
- Check return code

C coding example:

```
...
#define PCIDEREGISTER              3
...
PCI_HANDLE PCIHandle;
PCI_EXID ExchangeID;
{
PCI_INTEGER error;

/* call the entry point */
error = (*PCIHandle) (PCIDEREGISTER, ExchangeID);
...
```

F.1.3.5 PciPutMessage

This function is in charge to provide a message from a PUF to a NAF. Parameters shall be provided in the same order as indicated in the generic description of the PciPutMessage function.

The following operation shall be carried out by the PUF, in order:

- call the address with the PciPutMessage **function code** and the Exchange-ID related to the current association as well as the correct set-up PCI Message Parameter Block and the associated buffers;
- check return code.

C coding example:

```
...
#define PCIPUTMESSAGE          4
...
struct pci_mpb {
            PCI_INTEGER  MessageID;
            PCI_INTEGER  MessageMaximumSize;
            PCI_INTEGER  MessageActualUsedSize;
            PCI_INTEGER  DataMaximumSize;
            PCI_INTEGER  DataActualUsedSize;
};
...
PCI_HANDLE PCIHandle;
PCI_EXID ExchangeID;
struct pci_mbp far * PCIMbp;
PCI_BYTEARRAY Message;
PCI_BYTEARRAY Data;
{
PCI_INTEGER error;

/* call the entry point */
error = (*PCIHandle) (PCIPUTMESSAGE, ExchangeID, PCIMbp, Message, Data);
...
```

F.1.3.6 PciGetMessage

This function is in charge to provide the PUF with a message coming from the NAF. Parameters shall be provided in the same order as indicated in the generic description of the PciGetMessage function.

The following operation shall be carried out by the PUF, in order:

- call the address with the PciGetMessage **function code** and the Exchange-ID related to the current association as well as the correct set-up PCI Message Parameter Block and the associated buffers;
- check return code.

C coding example:

```
...
#define PCIGETMESSAGE                 5
...
struct pci_mpb {
                    PCI_INTEGER  MessageID;
                    PCI_INTEGER  MessageMaximumSize;
                    PCI_INTEGER  MessageActualUsedSize;
                    PCI_INTEGER  DataMaximumSize;
                    PCI_INTEGER  DataActualUsedSize;
};
...
PCI_HANDLE PCIHandle;
PCI_EXID ExchangeID;
struct pci_mbp far * PCIMbp;
PCI_BYTEARRAY Message;
PCI_BYTEARRAY Data;
{
PCI_INTEGER error;

/* call the entry point */
error = (*PCIHandle) (PCIGETMESSAGE, ExchangeID, PCIMbp, Message, Data);
...
```

F.1.3.7 PciSetSignal

This function is in charge to provide the NAF with the address of a function located inside the PUF, which shall be called-back if a message becomes available for the PUF.

The following operation shall be carried out by the PUF, in order:

- call the address with the PciSetSignal **function code** and the Exchange-ID related to the current association as well as the correct set-up function address of the call-back routine;
- check return code.

C coding example:

```
...
#define PCISETSIGNAL                  6

/*
 * CallBack function called in interrupt context
 */

void far CallBackFunc ()
{
...
return;
}
```

```
/*
 * Code to set up the notification process
 */
...
PCI_HANDLE PCIHandle;
PCI_EXID ExchangeID;
{
PCI_INTEGER error;

/* call the entry point */
error = (*PCIHandle) (PCISETSIGNAL, ExchangeID, &CallBackFunc);
...
```

The NAF calls-back the PUF with the following conventions applying:

- the NAF provides a minimal stack size of 128 bytes;
- the values of the DS and ES segments are undefined;
- interrupts are disabled.

Gained control, the PUF:

- may or may not enable interrupts;
- is allowed call the NAF via the PciGetMessage or the PciPutMessage function;
- shall not invoke other exchange function calls besides the PciGetMessage and the PciPutMessage functions;
- shall not issue MS-DOS system calls;
- shall not let interrupts disabled over an extended period of time and shall return from the call-back function as quick as possible.

The NAF called via the PCIGetMessage or the PciPutMessage function may enable interrupts. However, the NAF shall not call the call-back routine again, until the call-back routine has returned normally.

At the end of the call-back routine the PUF shall return to the NAF. Only the SS:SP register pair shall be preserved by the PUF.

F.2 Windows

F.2.1 Mechanism

Except for the PciGetHandles function call, the DLL mechanism is the basic mechanism used to support the ISDN PCI exchange method under Windows. Every NAF have to be DLL and have to export an entry point per ISDN PCI function using the same name (PciGetProperty, PciRegister, PciGetMessage, PciPutMessage, PciSetSignal, PciDeregister).

NOTE: Function name exported by the NAF is the same as the description made in Clause 7 but parameters are different.

PciGetHandles needs an access to the PCI.INI file.

PciRegister and PciGetProperty check if the DLL, accessible by its name, is available.

To access a NAF the only need for a PUF is to know the name of the DLL. The address access to the DLL may be provided transparently to the PUF inside the Pci's exchange mechanism functions as described in the Annex J.

The PciRegister function dynamically loads the NAF. It needs to keep trace of the handle of the NAF as a DLL, so this handle is part of the Exchange Identifier. The NAF also needs to keep trace of the PUF, so it assigns an Identifier to the PUF at registration time, this NAF-provided Identifier is the other part of the Exchange Identifier.

The common calling conventions to provide parameter to a DLL is the PASCAL calling convention. This convention is also used by the ISDN PCI exchange method under Windows.

Pointer parameters are far. The PCIMPB structure is always passed via a pointer. The Exchange Identifier structure is always passed via a pointer.

The structure alignment is **byte**.

F.2.2 Implementation of basic type

Under Windows, the following values shall be used:

PCI_HANDLE	name of the DLL
PCI_EXID	Structure contents
	handle provided by Windows when the DLL is loaded (hInstance)
	Unique Identifier provided by NAF to identify the PUF
PCI_PROCEDURE	exported function address (FARPROC) provided by the PUF
PCI_INTEGER	2 bytes
PCI_BYTEARRAY	far pointer

F.2.3 C Function prototypes

```
/*
 * Basic types
 */
typedef SHORT                    PCI_INTEGER;
typedef LPSTR                    PCI_BYTEARRAY;
typedef LPSTR                    PCI_HANDLE;
typedef struct                   {
                                 HINSTANCE        DLLInstance;
                                 PCI_INTEGER      Exchange_Id;
                                 } PCI_EXID;


typedef void (far pascal *PCI_PROCEDURE)(void);

/*
 * Structures
 */
struct pci_mpb {
            PCI_INTEGER  MessageID;
            PCI_INTEGER  MessageMaximumSize;
            PCI_INTEGER  MessageActualUsedSize;
            PCI_INTEGER  DataMaximumSize;
            PCI_INTEGER  DataActualUsedSize;
};

struct pci_register {                       /* structure containing registering info */
            PCI_INTEGER PUFVersion;         /* optional: give PUF version */
            PCI_INTEGER PUFType;            /* optional: give PUF type */
            PCI_INTEGER MaxMsgSize;         /* return: max size of a message */
};
struct pci_opsys {                    /* structure containing specific operating system info */
      int     DummyParameter;         /* No specific requirement for WINDOWS */
};

/*
 * Exchange functions prototypes
 */
```

```
PCI_INTEGER far PASCAL PciGetHandles (     PCI_INTEGER MaxHandles,
                                           PCI_BYTEARRAY PCIHandles,
                                           PCI_INTEGER far * ActualHandles
);

PCI_INTEGER far PASCAL PciGetProperty (     PCI_HANDLE PCIHandle,
                                           PCI_INTEGER MaximumSize,
                                           PCI_BYTEARRAY Property,
                                           PCI_INTEGER far * ActualSize
);

PCI_INTEGER far PASCAL PciRegister (        PCI_HANDLE PCIHandle,
                                           struct pci_register * PCIRegisterInfo,
                                           struct pci_opsys * PCIOpSysInfo,
                                           PCI_EXID far *ExID
);

PCI_INTEGER far PASCAL PciDeregister (      PCI_EXID far *ExID
);

PCI_INTEGER far PASCAL PciPutMessage (      PCI_EXID far *ExID,
                                           struct pci_mpb far *PCIMBP,
                                           PCI_BYTEARRAY Message,
                                           PCI_BYTEARRAY Data
);

PCI_INTEGER far PASCAL PciGetMessage (      PCI_EXID far *ExID,
                                           struct pci_mpb far *PCIMBP,
                                           PCI_BYTEARRAY Message,
                                           PCI_BYTEARRAY Data
);

PCI_INTEGER far PASCAL PciSetSignal (       PCI_EXID far *ExID,
                                           PCI_INTEGER Signal,
                                           PCI_PROCEDURE SignalProcedure
);
```

F.2.4 Description of functions

This subclause describes the implementation, under Windows, of the ISDN PCI exchange method functions. During a PUF to NAF call, the size of the stack shall be at least 1 024 bytes deep.

F.2.4.1 PciGetHandles

Under WINDOWS, the PciGetHandles uses a PCI.INI file in the WINDOWS directory to get available PCI_HANDLEs.

The section [Drivers] in the PCI.INI file contains all entries of installed NAFs. Each entry has the format:

```
pciDriver<number>= DLLName (number=1..32)
```

The following operations shall get all names of installed NAF drivers:

- loops from 1 to 32
  - constructs of the keyName "pciDriver" associated to the current loop value;
  - issue a GetPrivateProfileString using:
    sectionKey = "DRIVERS",
    the keyName contructs before,
    no default value,
    a maximum size equal to 128,
    FileName = "PCI.INI".

F.2.4.2 PciGetProperty

This function is in charge of providing to the PUF the PROPERTY of the NAF. Implicitly, it checks if the NAF is available - loading the library via the LoadLibrary function.

The following operations shall take place, in order:

- load the DLL;
- get the address of the PciGetProperty function exported by the NAF;
- call to this address with the parameters provided by the PUF;
- free the loaded library.

F.2.4.3 PciRegister

This function is in charge of providing an association between a PUF and a NAF. The NAF is loaded and the DLLInstance part of the Exchange Identifier is provided. The availability of the chosen NAF is checked during the load of the library. The library is identified by its name. Parameters for the registration operation are brought together a structure:

- PUFType (PCI_INTEGER);
- PUFVersion (PCI_INTERGER);
- MaxMsgSize (PCI_INTEGER) where the NAF will give the maximum size for a message.

The following operations shall take place, in order:

- load the DLL;
- provide the DLLInstance part of the Exchange Identifier with the DLL Instance;
- get the address of the PciRegister function exported by the NAF;
- call to this address to inform the NAF of a new PUF. The address of the registration parameters structure and the address of the Exchange Identifier structure are passed to the NAF as parameters;
- on return from the NAF, the Exchange_Id part of the Exchange Identifier and the maximum message size parameter of the registration parameter structure have been provided by the NAF;
- return to the PUF with the return code from the NAF.

F.2.4.4 PciDeregister

This function is charged of disassociating a PUF and a NAF. The DLL usage number shall be decremented by Windows but the DLL is not freed from the memory each time a PUF deregisters a NAF.

The following operations shall take place, in order:

- get the address of the PciDeregister function exported by the NAF;
- call to this address to inform the NAF of the end of the association. The PCI_EXID is passed to the NAF by address;
- free the DLL.

F.2.4.5 PciPutMessage

This function is in charge to provide a message, and associated data if any, from a PUF to a NAF. Parameters are provided in the same order as in the description of the PciGetMessage.

The following operations shall take place, in order:

- get the address of the PciPutMessage function exported by the NAF;
- call this address to pass parameter to the NAF (including the address of the PCI_EXID).

F.2.4.6 PciGetMessage

This function is in charge of providing a message, and associated data if any, from a PUF to a NAF. Parameters are provided in the same order as in the description of the PciGetMessage. Buffers provided by the PUF are directly used by the NAF.

The following operations shall take place, in order:

- get the address of the PciGetMessage function exported by the NAF;
- call this address to pass parameter to the NAF (including the address of the PCI_EXID).

F.2.4.7 PciSetSignal

This function allows a PUF to provide a direct information mechanism to be used by the NAF in case of incoming event. Two mutually exclusive mechanism are offered under Windows:

- a signal procedure mechanism;
- a user message mechanism.

Once a mechanism is chosen by the PUF, the other is de-activated by the NAF for that particular PUF. Both mechanism have to be supported by a NAF.

The first mechanism does not used the Signal parameter. This parameter shall be set to 0.

The second mechanism used the Signal parameter to identify the value associated with the WM_USER WINDOWS message. In that case, the Signal parameter shall not be equal to 0.

F.2.4.7.1 Signal mechanism procedure

The routine address, provided by the PUF in the SignalProcedure parameter, is used directly by the NAF. It shall to be made accessible to the NAF before it is provided by the PUF. The routine is called without any parameters.

In that case, the Signal parameter is not used but the parameter shall be passed to the NAF with the 0 value.

The stack used during the call to the SignalProcedure is not that of the PUF. The SignaProcedure shall be compiled without assuming SS=DS, like a DLL.

The NAF is allowed to call the PUF to reissue a signal call. To avoid big stack requirement, the NAF shall wait the return from the PUF signal procedure before reissuing the next signal call.

The PUF call back to the NAF during the signal procedure treatment shall not be allowed. The stack size is not guaranteed when the NAF calls the PUF. Consequently, the stack requirements for the PUF treatment shall be as small as possible.

F.2.4.7.2 User message mechanism procedure

The Signal parameter contains a PUF value to be added to the WM_USER WINDOWS message constant. This message is sent to a PUF Window. The HANDLE for this Window is provided by the PUF in the low word of the SignalProcedure parameter of the PciSetSignal function. It shall be a valid HANDLE WINDOW (HWND).

When the NAF issues the WM_USER + Signal message to the PUF, it uses a WINDOWS API PostMessage call. The PUF will find as third parameter (known as wParam) the type of the message received. In the fourth parameter (lParam), the PUF will find, as high word, the size of the Message associated to this message and as low word, the size of the Data associated. The call will look like:

```
PostMessage(    LOWORD( SignalProcedure),
                WM_USER+Signal,
                MessageID,
                (DWORD) (MessageSize << 16) | (DataSize));
```

As the PostMessage WINDOWS API is used, the PUF is allowed to call back the NAF during the message treatment.

This mechanism is simple to be implemented but an important constraint has to be given:

> under WINDOWS, a PostMessage call can fail due to a lack of room available in the message queue. The PUF is in charge to treat fast enough messages to insure that no NAF message will be lost. The PUF cannot rely on a failed message to be reissued by the NAF.

F.2.4.7.3 De-activation mechanism

To deactivate any signal mechanism the PciSetSignal function Signal and SignalProcedure parameters shall be NULL. Once deactivated, the previous mechanism shall no longer be used by the NAF to call the PUF.

F.3 Unix

F.3.1 Mechanism

The binary compatible interface to a NAF running under the UNIX·operating system shall be implemented using the STREAMS kernel mechanism.

The PCI exchange functions, described in Clause 7, shall be mapped to appropriate functions supplied by the UNIX STREAMS kernel mechanism.

Since "C" is the natural language in the UNIX environment, descriptions should preferably be made using "C" language.

F.3.2 Implementation of basic types

Table F.1 shows the mapping of the basic types of the exchange method to "C" language types:

Table F.1

Basis type	Mapping and usage
PCI_INTEGER	Can be implemented as 2 or 4 byte signed integer, whatever is defined within the underlying UNIX system as system constant for the 'int' type.
PCI_BYTEARRAY	Implemented as pointer to 'char' type.
PCI_EXID	Implemented as 'int' type. Since the exchange method is implemented using STREAMS, the Exchange-ID has the same value and type as the UNIX file descriptor provided by the STREAMS kernel mechanism.
PCI_HANDLE	Implemented as pointer to 'char' type, in fact a UNIX character-string. The string shall to contain the name of the STREAMS device the NAF is implemented in.
PCI_PROCEDURE	Implemented as address of a function returning 'void' type, as defined by UNIX signal () system call.

F.3.3 Parameter passing conventions

For parameter passing the usual 'C' conventions are applying:

- call values are either passed by value (e.g. PCI_INTEGER, PCI_EXID) , or by usage of a pointer (e.g. PCI_BYTEARRAY, PCI_HANDLE);
- return values are passed by giving a pointer for filling in the value (passing by reference).

Errors occurring inside the NAF driver are returned as positive integers (PCI_INTEGER). Their values are defined in Clause 5. As defined there, a value of 0 stands for "no error" (Success).

Errors occurred inside of the PCI exchange functions should be returned as negative integers (PCI_INTEGER). Their values are not defined. They are NAF implementation dependent.

F.3.4 Definition of types, constants and function-prototypes

The types, constants and function prototypes are presented using 'C' language.

If alignment is necessary on the UNIX target system, the size of the int type is employed.

```
/*
 * Basic types
 */

typedef int                          PCI_INTEGER;
typedef char *                       PCI_BYTEARRAY;
typedef int                          PCI_EXID;
typedef char *                       PCI_HANDLE;
typedef void                   (*    PCI_PROCEDURE) ();


/*
 * Structures
 */

struct pci_mpb {
                                     PCI_INTEGER  MessageID;
                                     PCI_INTEGER  MessageMaximumSize;
                                     PCI_INTEGER  MessageActualUsedSize;
                                     PCI_INTEGER  DataMaximumSize;
                                     PCI_INTEGER  DataActualUsedSize;
};


/*
 * Exchange functions prototypes
 */

PCI_INTEGER PciGetHandles (           PCI_INTEGER MaxHandles,
                                     PCI_BYTEARRAY PCIHandles,
                                     PCI_INTEGER * ActualHandles
);

PCI_INTEGER PciGetProperty ( PCI_HANDLE PCIHandle,
                                     PCI_INTEGER MaximumSize,
                                     PCI_BYTEARRAY Property,
                                     PCI_INTEGER * ActualSize
);
```

```
PCI_INTEGER PciRegister (           PCI_HANDLE PCIHandle,
                                    PCI_INTEGER PUFVersion,
                                    PCI_INTEGER PUFType,
                                    PCI_EXID * ExID,
                                    PCI_INTEGER * MaxMsgSize
);

PCI_INTEGER PciDeregister (         PCI_EXID ExID
);

PCI_INTEGER PciPutMessage   (       PCI_EXID ExID,
                                    struct pci_mpb * PCIMBP,
                                    PCI_BYTEARRAY Message,
                                    PCI_BYTEARRAY Data
);

PCI_INTEGER PciGetMessage  (        PCI_EXID ExID,
                                    struct pci_mpb * PCIMBP,
                                    PCI_BYTEARRAY Message,
                                    PCI_BYTEARRAY Data
);

PCI_INTEGER PciSetSignal (          PCI_EXID ExID,
                                    PCI_INTEGER Signal,
                                    PCI_PROCEDURE SignalProcedure
);
```

F.3.5 Adaptation to the STREAMS kernel mechanism

F.3.5.1 General

A NAF implemented into the UNIX kernel shall oppose its ISDN PCI interface via the STREAMS kernel mechanism. For each implemented NAF one STREAMS access shall be provided, independent of the amount of ISDN accesses the NAF provides. Such a STREAMS access can in principle, if implemented by the NAF, be used by several PUFs. Furthermore, as a consequence of the UNIX architecture, one PUF can access several STREAMS and thus several NAFs simultaneously. NAFs shall be defined as CLONE Streams.

The UNIX STREAMS kernel mechanism provides two queues, a write queue and a read queue. Information sent by the exchange functions to the stream driver (downstream information) are placed into the write queue by a STREAMS component called the stream head. Stimulating the stream head to do so is achieved by issuing the STREAMS putmsg() system call.

The stream driver can access the information of the write queue, processes it and places resulting information into the read queue. The content of the read queue (upstream information) is available to the exchange functions by use of the STREAMS getmsg() system call.

F.3.5.2 Communication between PUF exchange functions and NAF stream driver

The communication between an exchange function and the NAF stream driver is carried out by the exchange function by means of getmsg() or putmsg() in the case of PciGetMessage() and PciPutMessage(), and ioctl() in the case of all other functions.

The information transported through this stream is called a STREAMS message. STREAMS messages should not be confused with the messages defined in the ISDN PCI.

The STREAMS mechanism divides the PCI message into two parts: a control part and a data part. For messages exchanged via PciGetMessage() and PciPutMessage(), the control part of the STREAMS message contains the PCI message and the data part will contain the data part of the PCI message. The NAF driver receives the lengths of the individual parts of the PCI message by means of the standard UNIX getmsg() and putmsg() mechanism.

For all other messages, the individual command is passed to the NAF driver in the ioc_cmd field of the struct iocblk structure. The data part associated with this command is passed to the NAF driver in the data blocks of the M_IOCTL message.

Definitions of terms:

mp	is of type mblkt_t * (see /usr/include/sys/stream.h)
struct iocblk	type defined in /usr/include/sys/stream.h

The NAF STREAMS driver can obtain the information necessary for its operation by using the following mechanisms:

1. PCI Messages exchanged via PciPutMessage()

Information	**Availability**
Length of control part	mp->b_wptr - mp->b_rptr
Contents of control part	mp->b_rptr
Presence of a data part	mp->b_cont != NULL
Length of data part	msdgsize(mp)
Contents of data part	mp->b_cont->b_rptr

2. PCI Messages exchanged via the ioctl() mechanism

Information	**Availability**
Requested function	((struct iocblk *)mp->b_rptr)->ioc_cmd
Length of control part	((struct iocblk *)mp->b_rptr)->ioc_count
Contents of control part	mp->b_cont->b_rptr
Room for returned data	mp->b_cont->b_rptr
	((struct iocblk *)mp->b_rptr)->ioc_rval

The requested function shall be defined as follows:

```
#define PCI_PROPERTY          (('Z' << 8) | 1)

#define PCI_REGISTER          (('Z' << 8) | 2)

#define PCI_DEREGISTER        (('Z' << 8) | 3)

#define PCI_SETSIGNAL         (('Z' << 8) | 4)
```

F.3.5.3 Special considerations

Several NAF implementation aspects shall be considered by the PUF implementing the exchange functions:

- the PUF grants the NAF the permission to put incoming PCI messages on the read-side queue, thereby using this queue for buffering. Flow control is achieved by the standard UNIX highwater-lowwater mark mechanism which allows the NAF STREAMS driver to handle flow control transparently on the driver level;
- the size of a stream queue element is limited. A NAF stream driver shall be able to provide 4 096 bytes as data part of the stream message on the PUF's request, but it shall also guarantee this amount as the maximum delivered value; However, data block sizes of more than 4 096 bytes can be supported if the stream is put into "message non-discard mode" (see streamio(7)). Should a message with a data block size of more than 4 096 bytes arrive at the stream head, a call to PciGetMessage shall return the first 4 096 bytes of the data block and successive calls to PciGetMessage shall return the additional data blocks. Each of the additional calls to PciGetMessage shall return a message whose control part length will be zero;
- only the UNIX SIGPOLL signal shall be issued by the NAF implementation.

F.3.6 Description of functions

This subclause describes the implementation of the PCI exchange functions using the UNIX STREAMS mechanism. The description of each function is divided into 3 parts:

1)	Function body:	function body description, including general description of the function behaviour;
2)	STREAMS putmsg():	Structure set-up for call to putmsg();
3)	STREAMS getmsg():	Structure contents after return from getmsg().

The prototypes of putmsg () and getmsg () functions are:

```
int putmsg (fd, ctlptr, dataptr, flags)

        int fd;                         /* File descriptor                      */

        struct strbuf *ctlptr;          /* Control part of the message          */

        struct strbuf *dataptr;         /* Data part of the message             */

        int flags;                      /* Message priority.                    */




int getmsg (fd, ctlptr, dataptr, flags)

        int fd;                         /* File descriptor                      */

        struct strbuf *ctlptr;          /* Control part of the message          */

        struct strbuf *dataptr;         /* Data part of the message             */

        int *flags;                     /* Message priority.                    */
```

with

```
struct strbuf

        {

        int maxlen                      /*Maximum buffer length         */

        int len                         /* Length of data               */

        char *buf                       /* Pointer to buffer            */

        }
```

Alternatively, for PCI exchange functions which use the ioctl() mechanism the description of each function is divided into 2 parts:

1) Function body: function body description, including general description of the function behaviour;

2) ioctl(): Structure set-up for call to ioctl()
.

The prototype of ioctl () is:

```
int ioctl (fd, command, arg)

        int fd;                 /* File descriptor                                   */

        int command;            /*  ioctl command  as defined in streamio(7)         */

        char *arg;              /* command specific argument                         */
```

Whenever command is I_STR arg should point to a structure of type strioctl, where strioctl is defined as:

```
struct strioctl

        {

        int  ic_cmd;            /* User-defined command                 */

        int ic_timeout;         /* Timeout for command          */

        int ic_len;             /* Length of data part to follow    */

        char  *ic_dp;           /* Command-specific arguments */

        }
```

F.3.6.1 PciGetHandles

Function body:

```
PCI_INTEGER PciGetHandles (        PCI_INTEGER          MaxHandles,
                                   PCI_BYTEARRAY        PCIHandles,
                                   PCI_INTEGER          *ActualHandles)


{


...


}
```

MaxHandle contains the maximum number of PCI_HANDLE the PCIHandles parameter can receive. On return, ActualHandles, which is a pointer to an integer value, shall contain the number of PCI_HANDLE copied into the PCIHandles parameter.

This function shall:

- examine the directory **/etc/pcidd** to get the number and the PCI_HANDLEs available;
- update the PCIHandles and the ActualHandles parameters;
- return appropriate error code.

F.3.6.2 PciGetProperty

Function body:

```
PCI_INTEGER PciGetProperty   (     PCI_HANDLE        PCIHandle,
                                   PCI_INTEGER       MaximumSize,
                                   PCI_BYTEARRAY     NAFProperty,
                                   PCI_INTEGER       *ActualSize)
{
        struct strioctl   strioctl;
        extern int        errno;
        int               filedes;
}
```

PCIHandle points to the path name of the STREAMS device, MaximumSize is the size of the buffer to hold the properties. NAFProperty is the pointer to this buffer and ActualSize is a pointer to a integer value receiving the actual size of the property information in the NAF on return.

This function shall:

- open the STREAMS device using PCIHandle;
- issue the ioctl() call;
- retrieve the value of ActualSize and the error code;
- close the STREAMS device;
- return appropriate error code.

STREAMS ioctl():

The ic_cmd component shall be set to PCI_PROPERTY, the ic_len component shall be set to MaximumSize and the ic_dp component shall be set to point to the NAFProperty buffer.

Upon return from the ioctl() call the return value shall be checked against 0 which will indicate success. Any other return value indicates an error condition, which indicates that the errno variable contains the error condition. The ic_len component of the strioctl structure contains the number of bytes returned by the ioctl call. The ic_dp component points to the property returned.

NOTE: The size returned is always the size of the property inside the NAF.

```
strioctl.ic_cmd         = PCI_PROPERTY;

strioctl.ic_timout      = 0;

strioctl.ic_len         = MaximumSize;

strioctl.ic_dp          = (char *) NAFProperty;


if (ioctl (filedes, I_STR, &strioctl) == 0) {

        *ActualSize = strioctl.ic_len;

        return 0;

}

else {

        *ActualSize = 0;

        return errno;

}
```

F.3.6.3 PciRegister

Function body:

```
PCI_INTEGER PciRegister     (PCI_HANDLE          PCIHandle,
                            PCI_INTEGER         PUFVersion,
                            PCI_INTEGER         PUFType,
                            PCI_EXID            *ExID,
                            PCI_INTEGER         *MaxMsgSize)
{
        struct strioctl         strioctl;
        struct pci_register_t   pci_register;
        extern int              errno;
}
```

PCIHandle points to the path name of the STREAMS device. PUFVersion and PUFType are integers and set as indicated in Clause 7. ExID is a pointer to an integer receiving the returned Exchange-ID, which shall be equal to the UNIX file descriptor returned by the open() system call. MaxMsgSize is an integer receiving the message size of the NAF as described in Clause 7.

This function shall:

- open the STREAMS device using PCIHandle;
- issue the ioctl() call;
- retrieve the return values from the pci_control structure;
- leave the STREAMS device open. Assign file descriptor of open() call to ExID;
- return appropriate error code.

STREAMS ioctl():

The ic_cmd component shall be set to PCI_REGISTER, the ic_len component shall be set to the size of the pci_register structure and the ic_dp component shall be set to point to the pci_register structure which is set up with the values of PUFVersion and PUFType. Upon return from the ioctl() call the return value shall be checked against -1 which will indicate an error condition. The external variable errno will be set to indicate the specific error condition. Any other return value indicates success, and the return value of the ioctl call shall indicate the maximum PCI message size the NAF supports.

```
struct pci_register_t {

        int             puf_version;

        int             puf_type;

} pci_register;


pci_register.puf_version = PUFVersion;

pci_register.puf_type           = PUFType;


strioctl.ic_cmd                 = PCI_REGISTER;

strioctl.ic_timout      = 0;

strioctl.ic_len                 = sizeof (pci_register);

strioctl.ic_dp                  = (char *) &pci_register;


if ((*ExID = open (PCI_HANDLE, O_RDWR)) == -1) {

        *ExID = 0;

        return <cant_open_device : errno provides more information>;

}


if ((*MaxMsgSize = ioctl (*ExID, I_STR, &strioctl)) < 0) {

        *MaxMsgSize = 0;

        return errno;

}

else {

        return 0;

}
```

F.3.6.4 PciDeregister

Function body:

```
PCI_INTEGER PciDeregister     (PCI_EXID         *ExID)

{

            struct strioctl             strioctl;

            extern int                  errno;

}
```

ExID identifies the open STREAMS device. It is identical with the file descriptor returned by the open() system call.

This function shall:

- issue the ioctl() call;
- retrieve the error return code;
- close the STREAMS device;
- return appropriate error code.

STREAMS ioctl():

The ic_cmd component shall be set to PCI_DEREGISTER, the ic_len component shall be set to zero; the ic_dp component shall be set to NULL. Upon return from the ioctl() call, the return value shall be checked against -1 which will indicate an error condition. The external variable errno shall be set to indicate the specific error condition. Any other return value indicates success.

```
strioctl.ic_cmd          = PCI_DEREGISTER;

strioctl.ic_timout       = 0;

strioctl.ic_len          = 0;

strioctl.ic_dp           = (char *) NULL;


if (ioctl (*ExID, I_STR, &strioctl) == -1) {

        return errno;

}

else {

        close (*ExID);

        return 0;

}
```

F.3.6.5 PciPutMessage

Function body:

```
PCI_INTEGER PciPutMessage        (PCI_EXID                  ExID,
                                  struct  pci_mbp           *PCIMBP,
                                  PCI_BYTEARRAY             Message,
                                  PCI_BYTEARRAY             Data)
        struct strbuf ctlbuf;     /* stream message control part pointer */
        struct strbuf databuf;    /* stream message data part pointer */
```

ExID identifies the STREAMS device. PCIMPB is a pointer to the PCI Message Parameter Block. Message and Data are the part of the PCI message to be sent to the NAF driver. Either Message or Data might be optional. In this case they are specified as NULL. In order to be more efficient (see STREAMS putmsg hereafter), it is recommended that the PCIMBP be stored contiguously before the Message, this allows to avoid a copy in memory.

This function shall:

- prepare the ctlbuf and databuf structures;
- issue the putmsg() call;
- retrieve the error return;
- return appropriate error code.

STREAMS putmsg():

```
        /* The general idea is to pass in ctlbuf a buffer containing the PCIMBP followed by the content of
        Message, and in databuf the content of Data */
          if (Message && ((char *)Message != (char *)PCIMBP + sizeof(pci_mbp))) {
        /* there is a Message not NULL, and PCIMBP and Message are not contiguous in memory,Have to
        build a buffer where PCIMBP is followed by the Message content */
        char *buffer;
        /* pointer to a buffer,large enough to receive PCIMBP and  the Message content */
          ...
          /* Here a memory allocation process may take place */
          ...
          memcpy (buffer, PCIMBP, sizeof(pci_mbp));
          memcpy ((buffer + sizeof(pci_mbp), Message, PCIMB->MessageActualUsedSize);
          ctlbuf->buf        = buffer;
          ctlbuf->len        = PCIMB->MessageActualUsedSize + sizeof(pci_mbp);
}
else {
          /* either there is no Message, or the PCIMPB and the Message are contiguous in memory */
          ctlbuf->buf        = PCIMBP;
          ctlbuf->len        = Message ? PCIMB->MessageActualUsedSize + sizeof(pci_mbp) :
          sizeof(pci_mbp);
}

databuf->buf     = Data;
databuf->len     = Data ? PCIMB->DataActualUsedSize : 0;


if (putmsg (ExID, &ctlbuf, &databuf, flags) != 0) {
          /* Error condition, errno will be set          */
          ....
}
else {
          /* Operation OK             */
          ....
}
```

F.3.6.6 PciGetMessage

Function body:

```
PCI_INTEGER PciGetMessage            (PCI_EXID                ExID,
                                      struct pci_mbp          *PCIMPB,
                                      PCI_BYTEARRAY           *Message,
                                      PCI_BYTEARRAY           *Data)
          struct strbuf      ctlbuf;  /* stream message control part pointer */
          struct strbuf      databuf; /* stream message data part pointer */
```

ExID identifies the STREAMS device. PCIMPB is a pointer to the PCI Message Parameter Block. Message and Data are the part of the PCI message to be received from the NAF driver. Either Message or Data may be optional. In this case they are specified as NULL. In order to be more efficient (see STREAMS getmsg hereafter), the PCIMBP should be stored contiguously before the Message, this allows to avoid a copy in memory.

This function shall:

- prepare the ctlbuf and databufstructures;
- issue the getmsg () call;
- retrieve the return values from the ctlbuf and databuf structures;
- return appropriate error code.

STREAMS getmsg():

/* The general idea is to pass in ctlbuf a buffer large enough for containing the PCIMBP followed by the content of Message , and in databuf the content of Data. The error code of the NAF is available in the errno variable. */

```
if (Message && ((char *)Message != (char *)PCIMBP + sizeof(pci_mbp))) {
        /* there is a Message not NULL and,  PCIMBP and Message are not contiguous in memory,
        have to reserve a buffer where PCIMBP can be followed by the Message content */
        char  *buffer; /* pointer to a buffer,large enough to receive PCIMBP and the Message        content
*/
        /* Here a memory allocation process may take place */
        ctlbuf->buf        = buffer;
}
else {
        /* either there is no Message, or the PCIMPB and the Message are  contiguous in memory
        */
        ctlbuf->buf                = PCIMPB;
}
ctlbuf->maxlen          = Message ? PCIMB->MessageMaximumSize
                  +sizeof(pci_mbp):sizeof(pci_mbp);
databuf->buf                    = Data;
databuf->maxlen                 = Data ? PCIMPB->DataMaximumSize : 0;

if (getmsg (ExID, &ctlbuf, &databuf, flags) != 0) {
        /* Error condition, errno will be set          */
        PCIMPB->c_error         = errno;
        ....
}
else {  /* Operation OK          */
        if (ctlbuf->len != -1 && ctlbuf->len >= sizeof(pci_mbp)) {
                /* Message, possibly of size 0 is present */
                PCIMPB->MessageActualUsedSize       = ctlbuf->len - sizeof(pci_mbp);
                if (Message && ((char *)Message != (char *)PCIMBP + sizeof(pci_mbp)))
                        {
                        /* there is a Message not NULL and,  PCIMBP and Message are not
                        contiguous in memory, a buffer where PCIMBP can be followed
                        by the Message content, has been used */
                        memcpy (PCIMBP, buffer, sizeof(pci_mbp));
                        memcpy (Message,(buffer + sizeof(pci_mbp)),
                                        (ctlbuf->len - sizeof(pci_mbp)));
                }
                else {
                        /* the PCIMPB and the Message are contiguous in memory,
                         no additional buffer used */
                Message = PCIMPB + sizeof(pci_mbp);
                }
        }
        else {
        /* No Message present or too small message: error at least PCIMPB should be there */
        .............
        }
if (databuf->len != -1) {
        /* Data block, possibly of size 0 is present */
        PCIMPB->DataActualUsedSize  = databuf->len;
}
else {
        /* No Data present */
        PCIMPB->DataActualUsedSize = 0;
}
}
```

F.3.6.7 PciSetSignal

Function body:

```
PCI_INTEGER  PciSetSignal(   PCI_EXID            *ExID,

                             PCI_INTEGER         Signal,

                             PCI_PROCEDURE       SignalProcedure)

{

        extern int           errno;

}
```

ExID identifies the STREAMS device, Signal the UNIX signal number. SignalProcedure is the address of the signal handler ('C' function) inside of the PUF. Only the UNIX SIGPOLL signal shall be issued by the NAF implementation. Consequently, any non-zero value in Signal shall turn on emission of UNIX SIGPOLL signals, a zero value shall turn emission off.

The SignalProcedure defined by the PUF shall re-issue the signal via the signal() system call - see below. This mechanism shall be mandatory otherwise the next signal provided by the NAF shall kill the PUF.

More than one signal can be sent by a NAF to a PUF before the PUF accesses the NAF. An access to the NAF by the PUF during signal procedure treatment is not recommended.

This function shall:

- issue the ioctl() call;
- retrieve the error code;
- register UNIX SIGPOLL signalling with the stream head using: ioctl (..., I_SETSIG, S_MSG) system call;
- register UNIX SIGPOLL signalling with the operating system using: signal (SIGPOLL, SignalProcedure) system call;
- return appropriate error code.

STREAMS ioctl():

The function shall check the Signal parameter and shall, if Signal equals zero, set up the Signal_options variable to zero to turn off signalling. Furthermore, the signal function shall be de-registered by issuing the appropriate signal() call.

If Signal is non-zero, Signal_options shall be set to enable SIGPOLL signalling and any other options mandated by the implementation (see sigaction()). Furthermore, the signal function shall be registered using the signal() system call.

```
if (Signal == 0) {

        Signal_options = 0;


        if (ioctl (ExID, I_SETSIG, &Signal_options) == -1)

                return errno;

        signal (SIGPOLL, SIG_DFL);


        return 0;

}

else {

        Signal_options = <SETSIG options>


        if (ioctl (ExID, I_SETSIG, &Signal_options) == -1)

                return errno;

        signal (SIGPOLL, SignalProcedure);


        return 0;

}
```

F.4 Availability of NAF's PCI_HANDLE

To be accessible via the PciGetHandles function call, a NAF shall issue a declaration action. The inverse action - extraction from the list of available NAFs - is described too. These actions are operating system specific.

F.4.1 DOS

F.4.1.1 Declaration action

The NAF uses the PCIDD$ Device Driver to declare itself, issuing an IOCTL write command, passing a structure containing the action code (Declare) and the handle of the NAF.

The maximum number of NAF than the 'PCIDD$' Device Driver can register is 32.

The following operation takes place in order:

- open the 'PCIDD$' driver;
- prepare the following structure:
  - one word: command code, 0x4544 (characters 'DE', DEclaration);
  - one double-word: address of the NAF entry point;
- issue a IOCTL system call write command:
  - CX contains the size of the declaration structure (6);
  - DS:DX point to the structure;
- check the success of the operation (check CARRY FLAG);
- in case of error, issue a Get Extended Error function call to get a more comprehensive error code;
- close the driver.

The command shall end successfully even if the NAF is already declared. In this case, no action takes place.

The command gives an error on the following cases. In these cases, no action takes place.

- Standard DOS errors (Invalid handle, Invalid function number, etc...).
- The length of the buffer passed (register CX) is not correct (extended error 24, Bad request structure length).
- The command code is invalid (extended error 31, General failure).
- Already 32 NAF are declared **and** the NAF to be declared is not already declared (extended error 29, Write fault).

F.4.1.2 Extraction action

The NAF uses the PCIDD$ Device Driver to extract itself, issuing an IOCTL write command, passing a structure containing the action code (Extract) and the handle of the NAF.

The following operation takes place in order:

- open the 'PCIDD$' driver;
- prepare the following structure:
  - one word: command code, 0x5845 (characters 'EX', EXtraction);
  - one double-word: address of the NAF entry point;
- issue a IOCTL system call write command;
  - CX contains the size of the extraction structure (6);
  - DS:DX point to the structure;
- check the success of the operation (check CARRY FLAG);
- in case of error, issue a Get Extended Error function call to get a more comprehensive error code;
- close the driver.

The command shall be successful even if the NAF has not already been declared. In this case, no action takes place.

The command gives an error on the following cases. In these cases, no action takes place.

- Standard DOS errors (Invalid handle, Invalid function number, etc...).
- The length of the buffer passed (register CX) is not correct (extended error 24, Bad request structure length).
- The command code is invalid (extended error 31, General failure).

F.4.2 Windows

F.4.2.1 Declaration action

First, the NAF may get the list of available PCI_HANDLEs to check if not already declared. The mechanism the NAF uses is the same as any PUF to get available NAF: PciGetHandles (see subclause F.2.4.1).

If not yet declared, the NAF includes its own PCI_HANDLE into the list.

```
PCI_BYTEARRAY        ownDLLName = "xxx";
PCI_BYTE             driverName[128];
WORD                 index;
char                 keyName[20];

/* Check if NAF not already installed */
for (index = 1; index <=  32; index++)
       {
       sprintf( keyName, "pciDriver%d", index);
       if (GetPrivateProfileString(  "DRIVERS",      /* Section name */
                                     keyName,        /* "pciDriver"+1..n */
                                     NULL,           /* No default needed */
                                     driverName,
                                     sizeof( driverName),
                                     "PCI.INI") > 0)
          {
          if (strcmpi(driverName, ownDLLName) == 0) return; /* NAFinstalled, OK return */
          }
       }

/* Search a free pciDriver position */
for (index = 1, index <= 32; index++)
       {
       sprintf( keyName, "pciDriver%d", index);
       if (GetPrivateProfileString(  "DRIVERS",      /* Section name */
                                     keyName,        /* "pciDriver"+1..n */
                                     NULL,           /* No default needed */
                                     driverName,
                                     sizeof( driverName),
                                     "PCI.INI") == 0)
          {
          /* Entry does not exist, add own NAF Driver name */
          WritePrivateProfileString( "DRIVERS", keyName, ownDLLName, "PCI.INI");
          return;
          }
       }
```

Maximum number of NAF than can register is 32.

F.4.2.2 Extraction action

First, the NAF gets the list of available PCI_HANDLEs to check if it is declared. If so, the NAF removes its own PCI_HANDLE from the driver list in "PCI.INI".

```
PCI_BYTEARRAY       ownDLLName = "xxx";
PCI_BYTE            driverName[128];
WORD                index;
char                keyName[20];


for (index = 1, index <= 32; index++)
    {
    sprintf( keyName, "pciDriver%d", index);
    if (GetPrivateProfileString(  "DRIVERS",      /* Section name */
                                  keyName,        /* "pciDriver"+1..n
                                  NULL,           /* No default needed */
                                  driverName,
                                  sizeof( driverName),
                                  "PCI.INI") > 0)
      {
      /* Check for own name */
      if (strcmpi(driverName, ownDLLName) == 0)
           {
           /* Remove the name of the Driver */
           WritePrivateProfileString( "DRIVERS", keyName, "", "PCI.INI");
           }
      }
    }
```

F.4.3 UNIX

F.4.3.1 Declaration action

During the installation script of the STREAM driver, the directory **/etc/pcidd** is updated by a dummy file which is the name of the new NAF. The installation script may check the availability of the NAF before the creation of the new dummy file.

F.4.3.2 Extraction action

During the de-installation script of the STREAM driver, the directory **/etc/pcidd** is updated by removing the dummy file name of the NAF.

Annex G (normative): PCI ICS Proforma

G.1 Copyright release for PCI ICS Proforma

Notwithstanding the provisions of the copyright Clause related to the text of the present ETS, ETSI grants users of this ETS to freely reproduce the PCI Implementation Conformance Statement (ICS) Proforma in this annex so that it can be used for its intended purposes and may further publish the completed ICS.

G.2 Introduction

This annex contains the PCI ICS Proforma. The PCI ICS Proforma lists all mandatory, conditional and optional items of the ISDN PCI specification relating to the exchange mechanism and the supported messages. It shall be used in the process of evaluating a particular implementation when claiming conformance to, or support of, the ISDN PCI specification. The implementation which claims conformance can either be a PUF or a NAF. For the PUF the PCI ICS Proforma indicates if it uses the item. For the NAF, this is indicated if the item is supported.

To evaluate conformance of a particular implementation, it shall be necessary to have a statement of which capabilities and options have been implemented. This annex contains such a statement.

The supplier of an implementation which is conforming to this ETS shall complete a copy of the PCI ICS Proforma and shall provide information necessary to identify both the supplier and the implementation.

G.3 PCI ICS Proforma cover page

G.3.1 Identification of PCI ICS

PCI ICS serial no :
Date :

G.3.2 Identification of implementation

Name :
Version :
Special configuration :
Other information :

G.3.3 Identification of the system supplier

Name	:	Contact	:
Street	:	Phone no	:
City	:	Telex no	:
Country	:	Fax no	:

G.3.4 Global statement of conformance

Are all mandatory features implemented? (Yes or No) :

Answering "No" to this question indicates non-conformance to the ISDN-PCI interface specification. Non-supported mandatory capabilities are to be indicated in the PCI ICS, with an explanation of why the implementation is non-conforming.

G.4 Instructions for completing the PCI ICS Proforma

Each line within the PCI ICS Proforma which requires implementation details to be entered is numbered at the left hand edge of the line. This numbering is included as a means of uniquely identifying all possible implementation details within the PCI ICS Proforma.

The N/P column in this annex separates the capabilities for the Network Access Facility (NAF) and the PCI User Facility (PUF).

N	Network Access Facility (NAF);
P	PCI User Facility (PUF).

The D column in this annex reflects the definition of the items in this ETS. Each entry in this column is chosen from the following list:

M	Mandatory support is required;
O	Optional support is permitted. If implemented, it shall conform to this ETS;
C	Conditional support. The support of this element is subject to a condition which is described in the note column.

The I column in this annex describes the actual capabilities of the implementation and shall be completed by the supplier using a symbol chosen from the following list:

Y	item is implemented (subject to stated constraints);
N	item is not implemented;
-	item is not applicable.

The ref./note column in this annex contains the references to the location in the main body of this ETS where the items are described or a note explaining why the item is conditional.

The following abbreviations are used in the headings of this PCI ICS Proforma:

P/N	PCI User Facility (PUF)/Network Access Facility (NAF);
D	Defined;
I	Implemented;
ref.	reference.

G.5 Exchange Mechanism

	Item of ISDN PCI	N/P	D	I	ref./note
1	PciGetHandles	N	M	[ ]	7.1.2
		P	O	[ ]	7.1.2
2	PciGetProperty	N	M	[ ]	7.1.3
		P	O	[ ]	7.1.3
3	PciRegister	N	M	[ ]	**Error! Not a valid result for table.**
		P	M	[ ]	**Error! Not a valid result for table.**
4	PciPutMessage	N	M	[ ]	7.3.6
		P	M	[ ]	7.3.6
5	PciGetMessage	N	M	[ ]	7.3.7
		P	M	[ ]	7.3.7
6	PciSetSignal	N	M	[ ]	7.3.8
		P	O	[ ]	7.3.8
7	PciDeRegister	N	M	[ ]	7.2.1
		P	M	[ ]	7.2.1

G.6 Administration Plane

	Item of ISDN PCI	N/P	D	I	ref./note
8	Administration Plane message class 1	N	M	[ ]	6.2
	Basic class	P	M	[ ]	6.2
9	Administration Plane message class 2	N	O	[ ]	6.2
	Security features	P	O	[ ]	6.2
10	Administration Plane message class 3	N	O	[ ]	6.2
	Manufacturer specific features	P	O	[ ]	6.2

G.7 Control Plane

	Item of ISDN PCI	N/P	D	I	ref./note
11	Control Plane message class 1	N	M	[ ]	6.3
	Basic class	P	C	[ ]	6.3 Use of co-ordination function
10	Control Plane message class 2	N	O	[ ]	6.3
	Overlap sending	P	O	[ ]	6.3
13	Control Plane message class 3	N	O	[ ]	6.3
	User-to-user information transfer	P	O	[ ]	6.3
14	Control Plane message class 4	N	O	[ ]	6.3
	Call adjournment for telephony	P	O	[ ]	6.3
15	Control Plane message class 5	N	O	[ ]	6.3
	Facility invocation	P	O	[ ]	6.3
11	Control Plane message class 6	N	O	[ ]	6.3
	External Equipment handling	P	O	[ ]	6.3

G.8 User Plane

	Item of ISDN PCI	N/P	D	I	ref./note
12	User Plane message class 1	N	M	[ ]	6.4
	Basic class	P	M	[ ]	6.4

G.9 User Plane Protocols

	Item of ISDN PCI	N/P	D	I	ref./note
13	Network layer protocol according to ETS 300 080 [2]	N	M	[ ]	
		P	O	[ ]	
14	Network layer protocol according to ISO/IEC 8208	N	M	[ ]	
		P	O	[ ]	
15	Transparent User Plane protocol	N	M	[ ]	
		P	O	[ ]	
16	Network layer protocol according to network layer of T.70	N	O	[ ]	
		P	O	[ ]	
17	Network layer protocol using Null layer 3 with access to X.75 on layer 2	N	O	[ ]	
		P	O	[ ]	
18	Network layer protocol using Null layer 3 with transparent access to HDLC framing	N	O	[ ]	
		P	O	[ ]	

G.10 Miscellaneous features

	Item of ISDN PCI	N/P	D	I	ref./note
19	Transparent coding of facility information element.	N	O	[ ]	5.3.3.2,
		P	O	[ ]	5.3.3.2,

Annex H (normative): Static attribute content

This annex contains a complete description of the static attribute that a NAF shall provide. Rules to establish these attributes are in relation with:

- protocol requirement;
- services.

H.1 Control plane static attribute sets

The basic document used for this annex is ETR 018 [9].

The attribute sets described below use following conventions:

Name:	to be used with the ANcoCreateReq message;
BC:	content of the BearerCap parameter, in hexadecimal octet;
LLC	content of the LLC parameter, in hexadecimal octet - decimal in parenthesis;
HLC:	content of the HLC parameter - decimal in parenthesis.

H.1.1 Generic circuit bearer service

H.1.1.1 Speech

Name	: "SPEECH_A-LAW"
BearerCap	: 80 90 A3
LLC	: Not used
HLC	: Not used

Name	: 'SPEECH_µ-LAW'
BearerCap	: 80 90 A2
LLC	: Not used
HLC	: Not used

H.1.1.2 Unrestricted digital information

Name	: 'UNRESTRICTED'
BearerCap	: 88 90
LLC	: Not used
HLC	: Not used

H.1.1.3 Restricted digital information

Name	: 'UNRESTRICTED/56'
BearerCap	: 88 90 01 8F
LLC	: Not used
HLC	: Not used

H.1.1.4 3,1 Khz audio information transfer

Name	: 'AUDIO_A-LAW'
BearerCap	: 90 90 A3
LLC	: Not used
HLC	: Not used

Name	: 'AUDIO_µ-LAW'
BearerCap	: 90 90 A2
LLC	: Not used
HLC	: Not used

H.1.2 Packet mode bearer service

Name	: 'D_CHANNEL_HDL'
BearerCap	: 88 C0 C6 E6
LLC	: Not used
HLC	: Not used

H.1.3 Teleservices

Name	: 'TELEPHONYA_LAW'
BearerCap	: 80 90 A3
LLC	: Not used
HLC	: Standard = 0 Identification = 1

Name	: 'TELEPHONYμ_LAW'
BearerCap	: 80 90 A2
LLC	: Not used
HLC	: Standard = 0 Identification = 1

Name	: 'TELEFAX_G4'
BearerCap	: 88 90
LLC	: Depending on Terminal Equipment: octet 3a. Not used: octet 4 and 5. Octet 6 (layer 2) = 0D (13) Octet 7 (layer 3) = 07
HLC	: Standard = 0 Identification = 21 (33)

H.2 User Plane static attribute sets

The attribute sets described below use following conventions:

- Name shall be used with the ANcoCreateReq message;
- all numeric values are in decimal.

Name	: U_TELEMATIC_TERM
WindowSize	: 2
PacketSIze	: 128 (byte)
U3Protocol	: ETS 300 080
L3ConnectionMode	: DXE
L3TwoWayVCCount	: local arrangement
L3IncomingVCCount	: 0
L3OutgoingVCCount	: 0
L2ConnectionMode	: Auto
L2WindowSize	: 7
L2FrameSize	: 128 (byte)
L2XID	: none

Annex J (informative): Operating system implementation coding samples

These samples present a way to implement the exchange mechanism function call from the PUF point of view.

J.1 Sample DOS 'C' Code

```
/*************************************************************
This library code may be linked to a PUF to allow uniform access to multiple NAFs. The access to the
different NAFs by use of an unique ExID is achieved by the use of a local table, which allows MAX_EXID
entries.
*************************************************************/
/*
 * Include files
 */
#include <dos.h>
#include <fcntl.h>
#include <memory.h>
#include <malloc.h>
#include <stdio.h>

/*
 * General typedefs
 */
typedef void         ( *   PFRV) ();        /* Pointer to Function Returning Void        */
typedef short int( far *  FPFRS) ();    /* Far Pointer to Function Returning Short    */
typedef void         ( far *  FPFRV) ();    /* Far Pointer to Function Returning Void*/
typedef int          ( far *  FPFRI) ();    /* Far Pointer to Function Returning Int   */

/*
 * Mapping of generic type definitions
 */

typedef short int PCI_INTEGER;
typedef char far *     PCI_BYTEARRAY;
typedef short int PCI_EXID;
typedef FPFRI          PCI_HANDLE;
typedef FPFRV          PCI_PROCEDURE;

/*
* Definition of function codes
*/

#define PCIGETPROPERTY     (short) (1)
#define PCIREGISTER        (short) (2)
#define PCIDEREGISTER      (short) (3)
#define PCIPUTMESSAGE      (short) (4)
#define PCIGETMESSAGE      (short) (5)
#define PCISETSIGNAL       (short) (6)

/*
 * Error definitions
 */

#define E_DEVICE_DRIVER_NOT_FOUND      128
#define E_DEVICE_DRIVER_CONTROL                128
#define E_NAF_NOT_FOUND                130
#define E_NAF_INVALID_ADDRESS          130
#define E_TOO_MANY_ASSOCIATIONS        133
#define E_INVALID_EXCHANGE_ID          136
```

```
/*
 * Other definitions
 */

#define SUCCESS         0
#define MAX_EXID        32 /* allow 32 PUF_NAF associations */

/*
 * Structures
 */

struct pci_mpb {
        PCI_INTEGER MessageID;
        PCI_INTEGER MessageMaximumSize;
        PCI_INTEGER MessageActualUsedSize;
        PCI_INTEGER DataMaximumSize;
        PCI_INTEGER DataActualUsedSize;
};

struct pci_register {                           /* structure containing registering info */
        PCI_INTEGER PUFVersion;                 /* optional: give PUF version */
        PCI_INTEGER PUFType;                    /* optional: give PUF type */
        PCI_INTEGER MaxMsgSize;                 /* return: max size of a message */
};

struct pci_opsys {                              /* structure containing registering info */
        short intMaxNCOCount;/* optional: give max count of NCOs */
        short intMaxPacketSize;/* optional: give expected max size and */
        short intMaxPacketCount;        /* max count of packets to buffer */
        long int        AddBufferSize;          /* optional: give to NAF size and */
        void far *      AddBufferSpace;         /* pointer to additional buffer */
        long int        BufferNeeded;           /* return: amount of add buffer needed */
};

struct loc_exid_map {                           /* locally used structure for ExIDs */
        PCI_HANDLE      pci_handle;
        PCI_EXID        exchange_id;
};

/*
* Functional constants
*/
const char PCIsign[8]="ISDN PCI";

/*
 * Local variables
 */
static struct loc_exid_map _exid_map[MAX_EXID];         /* table holding MAX_EXID ExID entries */
static short int _exid_cnt = MAX_EXID;          /* count of free places inside ExID table */
```

```
/***
PciGetHandles :         Asks the "PCIDD$" device driver for a list of available
        PCI-Handles (NAF entry points).
        Returns available PCI-Handles into the given PCIHandles buffer.
        The maximum amount of PCI-Handles requested is given in MaxHandles.
        Function will fail, if MaxHandles is less than the Handles available in the driver.
***/
short int PciGetHandles (        short int MaxHandles,
                                 FPFRI * PCIHandles,
                                 short int * ActualHandles )
{
short int fildes;               /* file descripror */
union _REGS regs;
struct _SREGS segregs;

        /* open the driver */
if (_dos_open ("PCIDD$", _O_RDWR, &fildes) != SUCCESS)
        return E_DEVICE_DRIVER_NOT_FOUND;       /* device driver not accessible; return error */

/* prepare IOCTL read from device driver */
_segread (&segregs);
segregs.ds = FP_SEG (PCIHandles);               /* set-up segment address */
regs.x.dx = FP_OFF (PCIHandles);                /* and offset */
regs.x.cx = MaxHandles * sizeof(PCI_HANDLE);
regs.x.bx = fildes;                             /* set dos file handle */
regs.x.ax = 0x4402;                             /* IOCTL read from character device */

/* issue IOCTL read from device driver */
_intdosx (&regs, &regs, &segregs);

/* close the driver */
_dos_close (fildes);

/* check for error */
if (regs.x.cflag & 1)                                   /* check processors carry flag */
        return E_DEVICE_DRIVER_CONTROL; /* error has occurred; return error */

/* Successful operation. Compute count of PCI-Handles received */
*ActualHandles = regs.x.ax / sizeof(PCI_HANDLE);

return SUCCESS;
}       /* End of PciGetHandles() */
```

```
/***
PciGetProperty :            Asks the NAF for it's properties, which is a list of TLV coded topics.
            Returns the properties into the given Property buffer.
            The maximum size of the Property buffer is given in MaximumSize.
            Function will fail, if MaximumSize is less than the size of the Property the
            NAF can deliver.
***/

short int PciGetProperty ( FPFRI PCIHandle,
            short int MaximumSize,
            char * Property,
            short int * ActualSize )
{
register short int error;
unsigned char far * signature;

/* Check if NAF is available */
if ( PCIHandle == NULL)
            return E_NAF_INVALID_ADDRESS;       /* NAF inaccessible, invalid address */

/* compute address of signature and check it */
signature = (unsigned char far *) PCIHandle - sizeof(PCIsign);
if ( _fmemcmp (PCIsign, signature, sizeof(PCIsign) != SUCCESS))
            return E_NAF_NOT_FOUND;     /* NAF inaccessible invalid signature */

/* Call the NAF to obtain the property information */
error = (*PCIHandle) (  PCIGETPROPERTY,
            MaximumSize,
            (char far *) Property,
            (short int far *) ActualSize );

return error;
}           /* End of PciGetProperty() */

/***
PciRegister :      Tries to associate calling PUF with selected NAF.
            Delivers the ExID, which has to be used in subsequent calls.
            Two structures have to be provided by the calling PUF:
            -           The PCIRegisterInfo and
            -           the PCIOpSysInfo structure.
***/

short int PciRegister ( FPFRI PCIHandle,
            struct pci_register * PCIRegisterInfo,
            struct pci_opsys * PCIOpSysInfo,
            short int * ExID )
{
register short int error;
register short int exchange_id;
unsigned char far * signature;
struct loc_exid_map *exid_map; /* dynamic pointer to local _exid_map tab */

/* Check if NAF is available */
if ( PCIHandle == NULL)
            return E_NAF_INVALID_ADDRESS;       /* NAF inaccessible, invalid address */

/* compute address of signature and check it */
signature = (unsigned char far *) PCIHandle - sizeof(PCIsign);
if ( _fmemcmp (PCIsign, signature, sizeof(PCIsign) != SUCCESS))
            return E_NAF_NOT_FOUND;                 /* NAF inaccessible invalid signature */
/* check if there is still room in our local _exid_map table */
if ( ! _exid_cnt)
        return E_TOO_MANY_ASSOCIATIONS;         /* Indicate table exhausted */

/* Call the NAF to inform it of a new association PUF */
error = (*PCIHandle) (  PCIREGISTER,
        (struct pci_register far *) PCIRegisterInfo,
        (struct pci_opsys far *) PCIOpSysInfo,
        (short int far * ) ExID );
if ( ! error)
        {
        /* Association was successful; record it in local table */
        exchange_id = 0;
        exid_map = &_exid_map[0];       /* setup pointer into local _exid_map table */
        while (exid_map->pci_handle)
                {
                exid_map++;
                exchange_id += 1;
                }
        exid_map->exchange_id = *ExID;
        exid_map->pci_handle = PCIHandle;
        *ExID = exchange_id;    /* compute and set Exchange-ID */
        _exid_cnt -= 1;
        }

return error;
}       /* End of PciRegister() */

/***
PciDeregister :   Terminates an existing association wit a NAF.
        The ExID of an existing association has to be provided.
***/
```

short int **PciDeregister** (PCI_EXID ExID)

```
{
register short int error;
struct loc_exid_map *exid_map; /* dynamic pointer to local _exid_map tab */

/* Check if ExID is valid and setup pointer into local _exid_map table */
exid_map = &_exid_map[ExID];
if (ExID < 0 || ExID >= MAX_EXID || ! exid_map->pci_handle)
        return E_INVALID_EXCHANGE_ID;

/* Call the NAF to inform it of the end of the association */
error = (*exid_map->pci_handle) (   PCIDEREGISTER,
        exid_map->exchange_id );

/* delete association from local table */
exid_map->pci_handle = NULL;
_exid_cnt += 1;

return error;
}       /* End of PciDeregister() */
```

```
/***
PciPutMessage :         Transfers a Message and associated Data to the NAF.
***/

short int PciPutMessage (  short int ExID,
        struct pci_mpb * PCIMBP,
        char * Message,
        char * Data )
{
register short int error;
struct loc_exid_map *exid_map; /* dynamic pointer to local _exid_map tab */

/* Check if ExID is valid and setup pointer into local _exid_map table */
exid_map = &_exid_map[ExID];
if (ExID < 0 || ExID >= MAX_EXID || ! exid_map->pci_handle)
        return E_INVALID_EXCHANGE_ID;
/* Call the NAF and provide the message */
error = (*exid_map->pci_handle) (   PCIPUTMESSAGE,
        exid_map->exchange_id,
        (struct pci_mbp far *) PCIMBP,
        (char far *) Message,
        (char far *) Data );
return error;
}       /* End of PciPutMessage() */
/***
PciGetMessage :         Receives a Message and associated Data from the NAF.
***/

short int PciGetMessage (  short int ExID,
        struct pci_mpb * PCIMBP,
        char * Message,
        char * Data )
{
register short int error;
struct loc_exid_map *exid_map; /* dynamic pointer to local _exid_map tab */

/* Check if ExID is valid and setup pointer into local _exid_map table */
exid_map = &_exid_map[ExID];
if (ExID < 0 || ExID >= MAX_EXID || ! exid_map->pci_handle)
        return E_INVALID_EXCHANGE_ID;

/* Call the NAF and receive the message */
error = (*exid_map->pci_handle) (   PCIGETMESSAGE,
        exid_map->exchange_id,
        (struct pci_mbp far *) PCIMBP,
        (char far *) Message,
        (char far *) Data );

return error;
}       /* End of PciGetMessage() */
```

```
/***
PciSetSignal : Hands the address of a SignalProcedure to the NAF.
        The SignalProcedure then will receive notification on communication
        events (i.e. Message available for retrieval)
***/

short int PciSetSignal (   short int ExID,
        short int Signal,
        PFRV SignalProcedure )
{
register short int error;
struct loc_exid_map *exid_map; /* dynamic pointer to local _exid_map tab */

/* Check if ExID is valid and setup pointer into local _exid_map table */
exid_map = &_exid_map[ExID];
if (ExID < 0 || ExID >= MAX_EXID || ! exid_map->pci_handle)
        return E_INVALID_EXCHANGE_ID;

/* Call the NAF to set the signal function */
error = (*exid_map->pci_handle) (   PCISETSIGNAL,
        exid_map->exchange_id,
        (FPFRV) SignalProcedure );

return error;
}       /* End of PciSetSignal() */
```

J.2 Sample Windows "C" code

The following code shows a sample implementation of PUF exchange functions for the Windows environment. The sample is illustrated using "C" language:

```
/*
 * standard includes
 */
#include <windows.h>


/*
 * Basic types
 */
typedef short               PCI_INTEGER;
typedef LPSTR               PCI_BYTEARRAY;
typedef LPSTR               PCI_HANDLE;
typedef struct {
        HINSTANCE           hDLLInstance;
        PCI_INTEGER         Exchange_Id;
        } PCI_EXID;
typedef void (far pascal *PCI_PROCEDURE)();


/*
 * PCI Structures
 */
struct pci_mpb {
        PCI_INTEGER   MessageID;
        PCI_INTEGER   MessageMaximumSize;
        PCI_INTEGER   MessageActualUsedSize;
        PCI_INTEGER   DataMaximumSize;
        PCI_INTEGER   DataActualUsedSize;
        };
typedef struct pci_mpb PCI_MPB;

struct pci_register {                   /* structure containing registering info */
        PCI_INTEGER PUFVersion;         /* optional: give PUF version */
        PCI_INTEGER PUFType;            /* optional: give PUF type */
        PCI_INTEGER MaxMsgSize;         /* return: max size of a message */
};

struct pci_opsys {                      /* structure containing registering info */
        int      DummyParameter;        /* No specific requirement for WINDOWS */
};

/*
 * PCI defines
 */
#define PCI_HANDLE_LENGTH           128/* size of each handle in the buffer from PciGetHandles*/
#define PCI_E_SUCCESS                               0
#define PCI_E_QUERY_ENTITY_NOT_AVAILABLE            128
#define PCI_E_INVALID_PCI_HANDLE                    130
#define PCI_E_NAF_NOT_AVAILABLE                     255
```

```
/*
/////////////////////////////////////////////////////////////////////////////////////////
/// PciGetHandles()
*/
PCI_INTEGER far PASCAL PciGetHandles (      PCI_INTEGER MaxHandles,
                                            PCI_HANDLE PCIHandles,
                                            PCI_INTEGER far * ActualHandles)
        {
        int                     nafNumber;
        int                     nafFound;
        int                     size;
        char                    keyName[20];
        PCI_BYTEARRAY           buffer;

        buffer = PCIHandles;
        for (nafNumber = 1, nafFound = 0; nafNumber <= MaxHandles; nafNumber++)
                {
                wsprintf( keyName, "pciDriver%d", nafNumber);
                size = GetPrivateProfileString(     "DRIVERS",      /* Section name*/
                                                    keyName,        /* 'pciDriver'+1..n */
                                                    NULL,           /* No default string needed */
                                                    buffer,         /* Address where to put the result */
                                                    128,            /* Maxi. size for the result */
                                                    "PCI.INI");     /* INI FileName */
                if (size > 0)
                        {
                        nafFound++;                                 /* One more NAF found */
                        buffer += 128;/* Next location for a PCIHandle (128 octets fixed size) */
                        }
                }
        *ActualHandles = nafFound;
        }
```

```
/*
////////////////////////////////////////////////////////////////////////////////////////
/// PciGetProperty()
*/
PCI_INTEGER far PASCAL PciGetProperty (      PCI_HANDLE PCIHandle,
                                             PCI_INTEGER MaximumSize,
                                             PCI_BYTEARRAY Property,
                                             PCI_INTEGER far * ActualSize)
        {
        PCI_INTEGER iReturnCode;
        HINSTANCE hDLLInstance;
        FARPROC lpfnGetProperty;

        /* load the NAF's DLL */
        hDLLInstance = LoadLibrary(PCIHandle);
        if (hDLLInstance < HINSTANCE_ERROR)
                return PCI_E_INVALID_PCI_HANDLE;   /* error in LoadLibrary */

        /* get the "PciGetProperty" entry point of the dll */
        lpfnGetProperty = GetProcAddress(hDLLInstance, "PciGetProperty");
        if (lpfnGetProperty == NULL)
                {
                FreeLibrary(hDLLInstance);
                return PCI_E_NAF_NOT_AVAILABLE; /* error in GetProcAddress */
                }

        /* call the "PciGetProperty" entry point of the dll */
        iReturnCode = lpfnGetProperty(PCIHandle, MaximumSize, Property, ActualSize);

        /* free the DLL in any case */
        FreeLibrary(hDLLInstance);

        /* return with the DLL's return code */
        return iReturnCode;
        }
```

```
/*
///////////////////////////////////////////////////////////////////////////////////////
/// PciRegister()
///         The PCIOpSysInfo is kept for compatibility only
*/
PCI_INTEGER far PASCAL PciRegister (            PCI_HANDLE PCIHandle,
                                                struct pci_register * PCIRegisterInfo,
                                                struct pci_opsys * PCIOpSysInfo,
                                                PCI_EXID far *ExID)
        {
        PCI_INTEGER iReturnCode;
        FARPROC lpfnRegister;
        HINSTANCE hDLLInstance;

        /* load the NAF's DLL */
        hDLLInstance = LoadLibrary(PCIHandle);
        if (hDLLInstance < HINSTANCE_ERROR)
                return PCI_E_INVALID_PCI_HANDLE;   /* error in LoadLibrary */

        /* put the DLL instance in ExID */
        ExID->hDLLInstance = hDLLInstance;

        /* get the "PciRegister" entry point of the dll */
        lpfnRegister = GetProcAddress(hDLLInstance, "PciRegister");
        if (lpfnRegister == NULL)
                {   /* error in GetProcAddress */
                FreeLibrary(hDLLInstance);
                return PCI_E_NAF_NOT_AVAILABLE;
                }

        /* call the "PciRegister" entry point of the dll */
        iReturnCode = lpfnRegister(PCIRegisterInfo, ExID);

        if (iReturnCode != 0)
                {   /* error in PciRegister : free the DLL */
                FreeLibrary(hDLLInstance);
                }

        /* return with the DLL's return code */
        return iReturnCode;
        }
```

```
/*
/////////////////////////////////////////////////////////////////////////////////////////
/// PciDeRegister()
*/
PCI_INTEGER far PASCAL PciDeregister(PCI_EXID far *ExID)
        {
        PCI_INTEGER iReturnCode;
        FARPROC lpfnDeregister;

        /* get the "PciDeregister" entry point of the dll */
        lpfnDeregister = GetProcAddress(ExID->hDLLInstance, "PciDeregister");
        if (lpfnDeregister == NULL)   /* error in GetProcAddress */
                return PCI_E_NAF_NOT_AVAILABLE;

        /* call the "PciDeRegister" entry point of the dll */

        iReturnCode = lpfnDeregister(ExID);

        /* free the DLL in any case */
        FreeLibrary(ExID->hDLLInstance);

        /* return with the DLL's return code */
        return iReturnCode;
        }


/*
/////////////////////////////////////////////////////////////////////////////////////////
/// PciPutMessage()
*/
PCI_INTEGER far PASCAL PciPutMessage(   PCI_EXID far *ExID,
                                        PCI_MPB far *PCIMBP,
                                        PCI_BYTEARRAY Message,
                                        PCI_BYTEARRAY Data)
        {
        FARPROC lpfnPutMessage;

        /* get the "PciPutMessage" entry point of the dll */
        lpfnPutMessage = GetProcAddress(ExID->hDLLInstance, "PciPutMessage");
        if (lpfnPutMessage == NULL)   /* error in GetProcAddress */
                return PCI_E_NAF_NOT_AVAILABLE;

        /* call the "PciPutMessage" entry point of the dll */
        /* and return with the DLL's return code */

        return lpfnPutMessage(ExID, PCIMBP, Message, Data);
        }
```

```
/*
////////////////////////////////////////////////////////////////////////////////////////////////////////
/// PciGetMessage()
*/
PCI_INTEGER far PASCAL PciGetMessage(      PCI_EXID far *ExID,
                                           PCI_MPB far *PCIMBP,
                                           PCI_BYTEARRAY Message,
                                           PCI_BYTEARRAY Data)
        {
        FARPROC lpfnGetMessage;
        /* get the "PciGetMessage" entry point of the dll */

        lpfnGetMessage = GetProcAddress(ExID->hDLLInstance, "PciGetMessage");
        if (lpfnGetMessage == NULL)   /* error in GetProcAddress */
                return PCI_E_NAF_NOT_AVAILABLE;

        /* call the "PciGetMessage" entry point of the dll */
        /* and return with the DLL's return code */

        return lpfnGetMessage(ExID, PCIMBP, Message, Data);
        }


/*
////////////////////////////////////////////////////////////////////////////////////////////////////////
/// PciSetSignal()
*/
PCI_INTEGER far PASCAL PciSetSignal(       PCI_EXID far *ExID,
                                           PCI_INTEGER Signal,
                                           PCI_PROCEDURE SignalProcedure)
        {
        FARPROC lpfnSetSignal;

        /* get the "PciSetSignal" entry point of the dll */
        lpfnSetSignal = GetProcAddress(ExID->hDLLInstance, "PciSetSignal");
        if (lpfnSetSignal == NULL)   /* error in GetProcAddress */
                return PCI_E_NAF_NOT_AVAILABLE;

        /* call the "PciSetSignal" entry point of the dll */
        /* and return with the DLL's return code */

        return lpfnSetSignal(ExID, Signal, SignalProcedure);
        }
```

J.3 Sample UNIX "C" code

The **PciGetHandles** function call is not presented.

```
/*

 * Include files and basic definitions

 */

#include        <stddef.h>

#include        <fcntl.h>

#include        <signal.h>

#include        <stropts.h>

#include        <errno.h>

#include        <stdlib.h>


#define ERROR           (-1)            /* Error value */

#define Success         (0)             /* Success value */


/*

 * Basic types

 */


typedef int             PCI_INTEGER;

typedef char *          PCI_BYTEARRAY;

typedef int             PCI_EXID;

typedef char *          PCI_HANDLE;

typedef void            (* PCI_PROCEDURE)();
```

```
/*

 * Structures

 */



struct pci_mpb {

        PCI_INTEGER  MessageID;

        PCI_INTEGER  MessageMaximumSize;

        PCI_INTEGER  MessageActualUsedSize;

        PCI_INTEGER  DataMaximumSize;

        PCI_INTEGER  DataActualUsedSize;

};



struct pci_register {                   /* structure containing registering info */
        PCI_INTEGER PUFVersion;         /* optional: give PUF version */
        PCI_INTEGER PUFType;            /* optional: give PUF type */
        PCI_INTEGER MaxMsgSize;         /* return: max size of a message */
};

struct pci_opsys {                      /* structure containing registering info */
        int     DummyParameter;         /* No specific requirement for WINDOWS */
};

/*

 * Function definitions

 */



#define PCI_PROPERTY                    (('Z' << 8) | 1)

#define PCI_REGISTER                            (('Z' << 8) | 2)

#define PCI_DEREGISTER                  (('Z' << 8) | 3)

#define PCI_SETSIGNAL                   (('Z' << 8) | 4)
```

```
/***

 ***   Functions

 ***/




/*
 *          PciGetProperty function
 */

PCI_INTEGER PciGetProperty (PCIHandle, MaximumSize, NAFProperty, ActualSize)

            PCI_HANDLE PCIHandle;                  /* char **/

            PCI_INTEGER MaximumSize;               /* int              */

            PCI_BYTEARRAY NAFProperty;             /* char **/

            PCI_INTEGER * ActualSize;              /* int *            */

{

register int filedes;                 /* filedescriptor */

struct strioctl     strioct;          /* stream message control part pointer */


*ActualSize = ERROR;   /* preset with error value */


if ((filedes = open (PCIHandle, O_RDWR)) < Success)

            return ERROR;

strioct.ic_cmd              = PCI_PROPERTY;

strioct.ic_timout           = 0;

strioct.ic_len              = MaximumSize;

strioct.ic_dp               = (char *) NAFProperty;
if (ioctl (filedes, I_STR, &strioct) == 0)
          {
          *ActualSize = strioct.ic_len;
          close (filedes);
          return 0;
          }
else
          {
          *ActualSize = 0;
          close (filedes);
          return errno;
          }
}
```

```
/*
 *          PciRegister function
 */

PCI_INTEGER PciRegister (PCIHandle, pci_register, pcidummy, ExID)

            PCI_HANDLE PCIHandle;         /* char **/

            struct pci_register pciregister;

            struct pci_opsys pcidummy;

            PCI_EXID   * ExID;            /* int *    */

{

struct strioctl     strioctl;


struct pci_register_t {

            int                     puf_version;

            int                     puf_type;

} pci_reg;


pci_reg.puf_version         = pciregister.PUFVersion;

pci_reg.puf_type            = pciregister.PUFType;


strioctl.ic_cmd             = PCI_REGISTER;

strioctl.ic_timout= 0;

strioctl.ic_len             = sizeof (pci_reg);

strioctl.ic_dp              = (char *) &pci_reg;


if ((*ExID = open (PCIHandle, O_RDWR)) == -1)

            {

            *ExID = 0;

            return errno;

            }
if ((pciregister.MaxMsgSize = ioctl (*ExID, I_STR, &strioctl)) < 0)

        {

        pciregister.MaxMsgSize = 0;

        return errno;

        }

else

        {
        return 0;

        }

}

/*
 *      PciDeregister function
 */

PCI_INTEGER PciDeregister (ExID)

        PCI_EXID ExID;          /* int              */

{

struct strioctl    strioctl;


strioctl.ic_cmd             = PCI_DEREGISTER;

strioctl.ic_timout          = 0;

strioctl.ic_len             = 0;

strioctl.ic_dp               = (char *) NULL;


if (ioctl (ExID, I_STR, &strioctl) == -1)

        {

        return errno;

        }

else

        {

        close (ExID);

        return 0;

        }

}
```

```
/*
 *          PciPutMessage function
 */

PCI_INTEGER PciPutMessage (ExID, PCIMPB, Message, Data)

            PCI_EXID ExID;                  /* int              */

            struct pci_mpb * PCIMPB;

            PCI_BYTEARRAY Message;          /* char **/

            PCI_BYTEARRAY Data;             /* char **/

{

struct strbuf ctlbuf;

struct strbuf databuf;

char  *buffer = NULL;     /* pointer to a buffer, large enough to receive PCIMBP and Message
                          contents */

int nErr;


if (Message && ((char *)Message != (char *)PCIMPB + sizeof(struct pci_mpb)))

            {

            /* there is a Message not NULL, and PCIMPB Band Message are not contiguous
             in memory,Have to build a buffer where PCIMPB is followed by the Message content */

            /* Here a memory allocation process may take place */

            buffer = (char *) (malloc( sizeof(struct pci_mpb) + PCIMPB->MessageActualUsedSize));


            memcpy (buffer, PCIMPB, sizeof(struct pci_mpb));

            memcpy (buffer + sizeof(struct pci_mpb), Message, PCIMPB->MessageActualUsedSize);

            ctlbuf.buf          = buffer;

            ctlbuf.len          = PCIMPB->MessageActualUsedSize + sizeof(struct pci_mpb);

}

else

            {

            /* either there is no Message, or the PCIMPB and the Message are contiguous in memory */

            ctlbuf.buf          = (char *)PCIMPB;

            ctlbuf.len          = Message ? PCIMPB->MessageActualUsedSize + sizeof(struct pci_mpb)
                                            : sizeof(struct pci_mpb);

            }
```

```
databuf.buf       = Data;

databuf.len       = Data ? PCIMPB->DataActualUsedSize : 0;




if (putmsg (ExID, &ctlbuf, &databuf, 0) != 0)

        {

        nErr = errno;     /* errno contents the error code
 */
        }

else

        {

        nErr = 0;

        }

if (buffer != NULL) free(buffer);

return nErr;

}

/*
 *      PciGetMessage function
 */

PCI_INTEGER PciGetMessage (ExID, PCIMPB, Message, Data)

        PCI_EXID ExID;                 /* int         */

        struct pci_mpb * PCIMPB;

        PCI_BYTEARRAY Message;         /* char **/

        PCI_BYTEARRAY Data;            /* char **/

{

struct strbuf ctlbuf;

int flags;

struct strbuf databuf;

char  *buffer = NULL;     /* pointer to a buffer,large enough to receive PCIMBP and  the Message
                  content */

int nErr = 0;
```

```
if (Message && ((char *)Message != (char *)PCIMPB + sizeof(struct pci_mpb)))

        {

        /* there is a Message not NULL and,  PCIMPB and Message are not contiguous in memory,

        have to reserve a buffer where PCIMBP can be followed by the Message content */

        /* Here a memory allocation process may take place */

        buffer = (char *) (malloc( sizeof(struct pci_mpb) + PCIMPB->MessageMaximumSize ));

        ctlbuf.buf        = buffer;

        }

else {

        /* either there is no Message, or the PCIMPB and the Message are contiguous in memory */

        ctlbuf.buf  =  (char *)PCIMPB;

        }

ctlbuf.maxlen    =        Message ? PCIMPB->MessageMaximumSize + sizeof(struct
                          pci_mpb):sizeof(struct pci_mpb);

databuf.buf      =        Data;

databuf.maxlen =          Data ? PCIMPB->DataMaximumSize : 0;


if (getmsg (ExID, &ctlbuf, &databuf, &flags) != 0)

        {

        /* Error condition, errno will be set          */

        nErr = errno;

        }

else    {

        /* Operation OK             */

        if (ctlbuf.len != -1 && ctlbuf.len >= sizeof(struct pci_mpb)) {

                /* Message, possibly of size 0 is present */

                PCIMPB->MessageActualUsedSize       = ctlbuf.len - sizeof(struct pci_mpb);

                if (Message && ((char *)Message != (char *)PCIMPB + sizeof(struct pci_mpb)))

                        {

                        /* there is a Message not NULL and, PCIMPB and Message are not
                        contiguous in memory, a buffer where PCIMPB can be followed
                        by the Message content, has been used */

                        memcpy ( PCIMPB, buffer, sizeof(struct pci_mpb));
                memcpy ( Message,(buffer + sizeof(struct pci_mpb)), (ctlbuf.len -
                sizeof(struct pci_mpb)));

                }

            else

                {

                /* PCIMPB and Message are contiguous in memory, no more buffer used */

                Message = (char *) (PCIMPB + sizeof(struct pci_mpb));

                }

            }

        else

            {

            /* No Message present or too small message: error at least PCIMPB
            should be there */

            PCIMPB->MessageID = 0;

            PCIMPB->MessageActualUsedSize = 0;

            }


    if (databuf.len != -1)

        {

        /* Data block, possibly of size 0 is present */

        PCIMPB->DataActualUsedSize  = databuf.len;

        }

    else

        {

        /* No Data present */

        PCIMPB->DataActualUsedSize = 0;

        }

    }

    if (buffer != NULL) free( buffer);


    return nErr;

    }
```

```
/*
 *          PciSetSignal function
 */

PCI_INTEGER PciSetSignal (ExID, Signal, SignalProcedure)


PCI_EXID ExID;                          /* int              */

PCI_INTEGER Signal;                     /* int              */

PCI_PROCEDURE SignalProcedure;          /* void (*) ()      */

{

int Signal_options;

if (Signal == 0)

        {

        Signal_options = 0;


        if (ioctl (ExID, I_SETSIG, &Signal_options) == -1)

                return errno;

        signal (SIGPOLL, SIG_DFL);

        return 0;

        }

else

        {

        Signal_options = S_MSG;


        if (ioctl (ExID, I_SETSIG, &Signal_options) == -1)

                return errno;

        signal (SIGPOLL, SignalProcedure);

        return 0;

        }

}
```

Annex K (informative): TLV Coder/decoder sample

```
/*
//////////////////////////////////////////////////////////////////////////
///
///         SAMPLES
///
///         TLV coder and decoder
///
//////////////////////////////////////////////////////////////////////////
*/

#include <memory.h>
#include <stdarg.h>

/*
 * Definition of Types
 */
typedef int         BOOL;
#define FALSE  0
#define TRUE   1

#define LG_MAX_MESSAGE 128

/* Definition of structures */
struct sParameter /* Intermediate structure which receive the parameter to be added */
          {
          int iMessageLength;
          char scMessage[LG_MAX_MESSAGE];
          };

/*
//////////////////////////////////////////////////////////////////////////
///
///         Function            : AddOctetParameter
///
///         Rule                : Add an octet parameter in a message
///
///         Parameters          :
///                               structure sParameter pointer
///                               parameter type
///                               parameter value
///
///         Return  :
///                   TRUE   : Success
///                   FALSE  : Error during processing
///
//////////////////////////////////////////////////////////////////////////
*/
BOOL AddOctetParameter(      struct sParameter *pMessage,
                             unsigned char cType,
                             unsigned char cValue)
          {
          if (pMessage->iMessageLength + 3 > LG_MAX_MESSAGE) /* Buffer is too small */
                    {
                    /* Process message size error */
                    return FALSE;
                    }/* if */
	/* TLV coding */
	pMessage->scMessage[pMessage->iMessageLength++] = cType;
	pMessage->scMessage[pMessage->iMessageLength++] = 1; /* length = 1 for octet */
	pMessage->scMessage[pMessage->iMessageLength++] = cValue; /* content */

	/* Success */
	return TRUE;
	}/* AddOctetParameter */

/*
/////////////////////////////////////////////////////////////////////////
///
///	Function		: AddStringParameter
///
///	Rule	: Add a string (octet-string) parameter in a message
///
///	Parameters	:
///		structure sParameter pointer
///		parameter type
///		parameter length
///		parameter value (pointer)
///
///	Return	:
///		TRUE	: Success
///		FALSE	: Error during processing
///
/////////////////////////////////////////////////////////////////////////
*/
BOOL AddStringParameter(	struct sParameter *pMessage,
				unsigned char cType,
				int iLg,
				unsigned char *lpValue)
	{
	if (iLg == 0) return FALSE;

	if (pMessage->iMessageLength + iLg + 2 > LG_MAX_MESSAGE) /* Buffer is too small */
		{
		/* Process message size error */
		return FALSE;
		}/* if */

	/* TLV coding */
	pMessage->scMessage[pMessage->iMessageLength++] = cType;		/* Add the type */
	pMessage->scMessage[pMessage->iMessageLength++] = iLg;	/* Length */
	memcpy(pMessage->scMessage+pMessage->iMessageLength, lpValue, iLg);/* Value */
	pMessage->iMessageLength += iLg;

	/* Success */
	return TRUE;
	}/* AddStringParameter */
```

```
/*
//////////////////////////////////////////////////////////////////////
///
///     Function        : ExtractParameter
///
///     Rule            : Find a specific parameter and provide its location
///
///     Parameters      :
///             address to the message
///             current message length
///             parameter type we are looking for
///             pointer of pointer where to find value
///             pointer of an integer where to find the length of the parameter
///
///     Return  :
///             TRUE   : Success
///             FALSE  : Error during processing
///
//////////////////////////////////////////////////////////////////////
*/
BOOL ExtractParameter(          unsigned char *lpMessage,
                                unsigned int iLgMessage, unsigned char cType,
                                unsigned char * *lplpValue, unsigned int *lpiLgValue)
        {
        while (iLgMessage > 0) /* for all message parameters */
                {
                if (*lpMessage != cType)
                        {
                        /* process the next parameter */
                        iLgMessage -= lpMessage[1] + 2;
                        lpMessage += lpMessage[1] + 2;
                        continue;
                        }/* if */

                        /* the parameter type is found update information for the caller */
                        *lplpValue = lpMessage + 2;
                        *lpiLgValue = lpMessage[1];

                /* Success */
                return TRUE;
                }/* while */

        return FALSE;
        }/* ExtractParameter */
```

History

Document history	
March 1994	First Edition
February 1996	Converted into Adobe Acrobat Portable Document Format (PDF)