INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

T.200

(10/96)

SERIES T: TERMINALS FOR TELEMATIC SERVICES

Programmable communication interface for terminal equipment connected to ISDN

ITU-T Recommendation T.200
Superseded by a more recent version

(Previously CCITT Recommendation)

Superseded by a more recent version

ITU-T T-SERIES RECOMMENDATIONS

TERMINALS FOR TELEMATIC SERVICES

For further details, please refer to ITU-T List of Recommendations.

Superseded by a more recent version

FOREWORD

The ITU-T (Telecommunication Standardization Sector) is a permanent organ of the International Telecommunication Union (ITU). The ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Conference (WTSC), which meets every four years, establishes the topics for study by the ITU-T Study Groups which, in their turn, produce Recommendations on these topics.

The approval of Recommendations by the Members of the ITU-T is covered by the procedure laid down in WTSC Resolution No. 1 (Helsinki, March 1-12, 1993).

ITU-T Recommendation T.200 was prepared by ITU-T Study Group 8 (1993-1996) and was approved by the WTSC (Geneva, 9-18 October 1996).

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

CONTENTS

Recommendation T.200

PROGRAMMABLE COMMUNICATION INTERFACE FOR TERMINAL EQUIPMENT CONNECTED TO ISDN

(Geneva, 1996)

An important aspect for the success of ISDN is the availability of user applications (in particular, applications based on Personal Computers) making use of ISDN.

Various national, regional and manufacturer specifications or standards, respectively, have appeared in the market during recent years.

One of these specifications has been elaborated in ITU-T. It is recommended to take this specification, which is attached hereto as an appendix, into consideration when different programmable communication interface solutions are evaluated for implementation.

Appendix I

Programming communication interface for terminal equipment connected to ISDN

This appendix contains nine parts:

Part 1: General architecture

Part 2: Basic services

Part 3: User plane protocols management architecture

Part 4: Layer One protocols

Part 5: Layer Two protocols

Part 6: Layer Three protocols

Part 7: DOS exchange mechanism

Part 8: Windows exchange mechanism

Part 9: UNIX exchange mechanism

Superseded by a more recent version

CONTENTS

PART 1

Superseded by a more recent version

PART 1: GENERAL ARCHITECTURE

Summary

An important aspect for the success of ISDN will be the availability of user applications (in particular, applications based on Personal Computers) making use of ISDN. This specification is the introduction to the multi-part specification that defines a standard Programming Communication Interface (PCI) allowing applications to access and manage the services provided by an ISDN. It provides mechanisms to support the most protocols used for communication between ISDN applications.

This part of the specification introduces a general description of the PCI; in particular it provides an overview of the contents of each part of this specification.

Introduction

This ITU-T ISDN Application Programming Interface (API), called ISDN Programming Communication Interface (PCI) is an application interface for accessing and administering ISDN. It is a multi-part specification in which this specification is the introduction.

ISDN-PCI has been defined in order to provide a standard for terminal equipment providers that makes possible 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 future ISDN extensions will not affect the operation of existing applications.

1 Scope

This part describes the Integrated Services Digital Network Programming Communication Interface (ISDN-PCI) specification organization.

It describes the structure of the set of specifications with a short description of each.

2 References

[1] Part 2, *Basic services.*

[2] Part 3, *User plane protocols management architecture.*

[3] Part 4, *Layer One protocols.*

[4] Part 5, *Layer Two protocols.*

[5] Part 6, *Layer Three protocols.*

[6] Part 7, *DOS exchange mechanism.*

[7] Part 8, *Windows exchange mechanism.*

[8] Part 9, *UNIX exchange mechanism.*

3 Definitions

This specification defines the following terms:

3.1 **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.

3.2 **exchange mechanism**: Means provided for the PUF to interchange messages with the NAF.

3.3 **ISDN programming communication interface (ISDN-PCI)**: ISDN oriented software interface providing access provisions for programming network signalling and user data exchange.

3.4 **message**: Unit of information transferred through the interface between the Network Access Facility (NAF) and the PCI User Facility (PUF).

3.5 **network access facility (NAF)**: Functional unit located between the ISDN-PCI and the network related layers.

3.6 **PCI user facility (PUF)**: Functional unit using the ISDN-PCI to access a NAF, e.g. the local application using the interface.

3.7 **user plane**: A logical grouping of functionality providing access for user protocols and data.

3.8 **user protocol**: The protocol running and conforming to User Plane functionality.

4 Abbreviations

This specification uses the following abbreviations.

API	Application Programming Interface
ISDN	Integrated Services Digital Network
NAF	Network Access Facility
NCO	Network Connection Object
PCI	Programming Communication Interface
PUF	Programming communication interface User Facility

5 Overview of the ISDN-PCI set of specifications

This specification is intended to assist software developers, implementors of applications and equipment manufacturers understand the organization of the ISDN-PCI set of specifications.

Figure 1 shows the relation between the ISDN-PCI specifications.

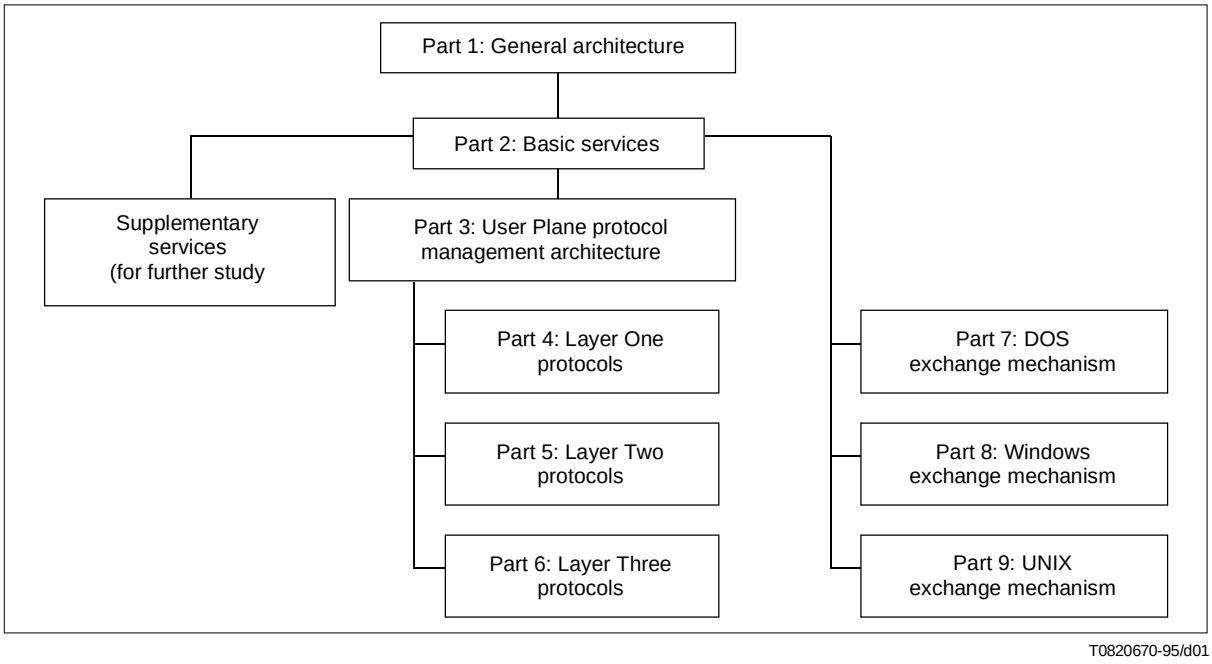


Figure 1 – ISDN-PCI organizational chart

– Part 2, "Basic Services" [1], contains the common set of information for all parts. It should be read first. It includes the way to manipulate a Network Connection Object (NCO) and general set of parameters which apply to it and also the way to establish an ISDN connection. It covers the description of the exchange mechanism which helps to select and to exchange messages with an NAF. In addition, it describes the use of external equipment such as telephony equipment. Frequent references to the content of this part are incorporated in the other parts of the ISDN-PCI series.

NOTE – A version control exchange mechanism is defined in ISDN-PCI. The version number which refers to the current specification is 2.

– Part 3, "User plane protocols management architecture" [2], contains the description of the use of User Plane protocols for a D- or a B-channel. It describes the protocol location referring to the OSI model. It includes the description of the way to select a particular protocol, via the NCOtype and UProtocol parameters. The concept of the coordination function is explained in detail in Part 3. It presents general rules in a protocol independent manner, such as selection criteria of an NCO or error management.

– Part 4, "Layer One protocols" [3], contains the description of the use of Layer One User Plane protocols over a B-channel. It describes how to access transparently the bytes exchanged on the channel.

– Part 5, "Layer Two protocols" [4], contains the description of the use of Layer Two User Plane protocols over a B-channel. It describes how to access and make use of PPP, SDLC, HDLC with or without framing and V.110 protocols. It contains message and parameter utilization and values, the sequencing and the mapping of the appropriate messages in a per protocol manner, the error management rules and some configuration information.

– Part 6, "Layer Three protocols" [5], contains the description of the use of Layer Three User Plane protocols over a B-channel. It describes how to access and make use of ISO 8208, T.90 and T.70 protocols. It contains the coordination function application, message and parameter utilization and values, the sequencing and the mapping of the appropriate messages in a per protocol manner, the error management rules and some configuration information.

– Part 7, "DOS exchange mechanism" [6], contains the DOS operating system dependent information of the exchange method described in [2]. It explains how to access and make use of an NAF, what are the DOS considerations and gives an example of an implementation using the C-language.

– Part 8, "Windows exchange mechanism" [7], contains the Windows™ operating system dependent information of the exchange method described in [2]. It explains how to access and make use of an NAF, what are the Windows considerations and gives an example of an implementation using the C-language.

– Part 9, "UNIX exchange mechanism" [8], contains the UNIX™ operating system dependent information of the exchange method described in [2]. It explains how to access and make use of an NAF, what are the UNIX considerations and gives an example of an implementation using the C-language.

Superseded by a more recent version

CONTENTS

PART 2

PART 2: BASIC SERVICES

Summary

This part of the specification provides a technical overview and defines the Basic Functions supported by the ISDN-PCI. It defines the PCI architecture and includes a detailed definition of the PCI messages and parameters used for Administration and Connection Control. It explains how to make use of these messages and parameters via a generic exchange mechanism.

1 Scope

This part constitutes a part of the Integrated Services Digital Network Programming Communication Interface (ISDN-PCI) for accessing and administering the ISDN services as indicated in [1] and specifies the "Basic Functions" provided by ISDN-PCI.

Basic functions provided in this part specify:

- an administration and control interface for applications requiring direct control of ISDN;
- supports for application access to multiple channels on multiple ISDN accesses;
- supports concurrent applications;
- general mechanisms to support multiple and concurrent protocol stacks related to data exchange;
- bindings to common operating system environments;
- the access to security features via the interface.

2 References

[1] ITU-T Recommendation Q.931 (1993), *ISDN user-network interface layer 3 specification for basic call control.*

[2] ITU-T Recommendation X.213 (1995), *Information technology – Open Systems Interconnection – Network service definition.*

[3] Part I, *General architecture.*

[4] Part 3, *User plane protocol management architecture.*

[5] Part 4, *Layer One protocols.*

[6] Part 5, *Layer Two protocols.*

[7] Part 6, *Layer Three protocols.*

[8] Part 7, *DOS exchange mechanism.*

[9] Part 8, *Windows exchange mechanism.*

[10] Part 9, *UNIX exchange mechanism.*

For further references to publications, which are of interest when reading this part, refer to the bibliography at the end of this part.

3 Definitions

This part defines the following terms:

3.1 **address set**: Set of parameters containing remote and local user layer or signalling addresses.

3.2 **administration plane**: Logical grouping of functionality for management of PUF-NAF dialogue as well as for access to local or network related NAF resources.

3.3 **attribute set**: Set of parameters driving user protocols and ISDN signalling.

3.4 **B-channel**: Logical ISDN channel for the use of data transfer.

3.5 **control plane**: Logical grouping of functionality for access of ISDN signalling.

3.6 **D-channel**: Logical ISDN channel used for signalling and in some cases, for data transfer.

3.7 **exchange function**: PUF functionality realizing the exchange mechanism.

3.8 **exchange mechanism**: Means provided for the PUF to interchange messages with the NAF.

3.9 **ISDN access**: Set of ISDN channels provided by a single Network Access Facility (NAF) to access ISDN services.

3.10 **ISDN programming communication interface (ISDN-PCI)**: ISDN oriented software interface providing access provisions for programming network signalling and user data exchange.

3.11 **message**: Unit of information transferred through the interface between the Network Access Facility (NAF) and the PCI User Facility (PUF).

3.12 **network access facility (NAF)**: Functional unit located between the ISDN-PCI and the network related layers.

3.13 **network connection object (NCO)**: Abstract object within the NAF that shall be created by the PUF to gain access to network signalling or data.

3.14 **NULL layer**: Describes an empty layer of the OSI reference model. Such a layer does not contain any functionality and passes requests and responses transparently to adjoining layers.

3.15 **PCI user facility (PUF)**: Functional unit using the ISDN-PCI to access a NAF, e.g. the local application using the interface.

3.16 **type-length-value coding (TLV coding)**: Coding scheme used for binary presentation of messages.

3.17 **user connection**: Connection accessible through User Plane functionality.

3.18 **user plane**: Logical grouping of functionality for access of user protocols and data.

3.19 **user protocol**: Protocol running and conforming to User Plane functionality.

4 Abbreviations

This part uses the following abbreviations:

API	Application Programming Interface
CONS	Connection Oriented Network Service
HLC	High Layer Compatibility
ISDN	Integrated Services Digital Network
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
NAF	Network Access Facility
NCO	Network Connection Object
PCI	Programming Communication Interface
PciMPB	Pci Message Parameter Block
PUF	Programming communication interface User Facility
TLV coding	Type-Length-Value coding (used for presentation of ISDN-PCI messages)
X.25 PLP	X.25 Packet Layer Protocol

5 Functional model

5.1 Introduction

This clause describes the functional model for the ISDN-PCI. It introduces the architecture of the ISDN-PCI. 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 enables a higher layer entity to 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 5.2.1.

5.2.1 ISDN-PCI and its components

This subclause identifies the functional components relevant for the definition of the ISDN-PCI. Their relationship is described in Figure 1.

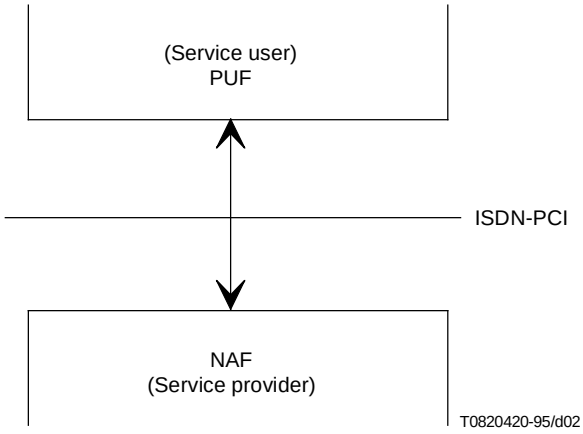


Figure 1 – Functional picture of ISDN-PCI with surrounding components

The functional components relevant for ISDN-PCI are:

PUF

Throughout this part, 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.

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 that is described in Figure 2. 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 via the NAF is 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. Furthermore, the control plane is responsible for managing special equipment accessible through the ISDN-PCI.

- **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.

Figure 2 gives a representation of the three planes.

5.2.3 Coordination cases

PCI-ISDN provides for two mechanisms to coordinate the functionalities associated with the ISDN signalling and the user connection.

In the PUF coordination case, the PUF shall handle the establishment of a user connection by using the basic call control provided by the control plane. As a result of controlling the signalling connection, the PUF can use the supplementary services.

In the NAF coordination case, an abstraction is provided by a coordination function, which maps the primitives of CONS X.213 [2] in the User Plane according to the primitives of the control plane and User Plane protocols. A detailed description of the condition and procedures for the use of the coordination function is provided in Part 3 [4]. Since the NAF manages the coordination between signalling and user connection, the PUF shall not access the control plane.

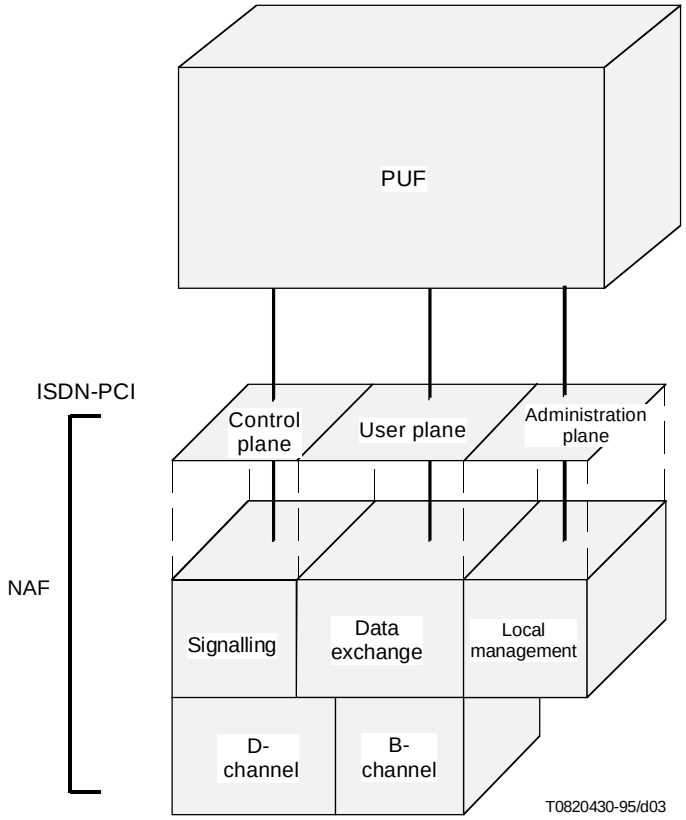


Figure 2 – Relation between planes and ISDN

5.3 Functionality

5.3.1 Introduction

As described in 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 5.5.

In order to access ISDN signalling or data, the PUF has to request the NAF for creation of an NCO. The creation and destruction of network connection is the main part of the functionality of the resource management.

After having performed this successfully, the PUF is in an "idle" state and may subsequently access ISDN signalling (except in case of NAF coordination) 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 resource management functionality is needed to be able to use the ISDN-PCI for communication. 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 evolves 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 principal 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. In this part, Annex C gives the list of standardized attribute sets for the control plane.

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- 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 configuring user protocols, for executing the ISDN signalling protocol and for collecting some management information relevant to the NCOs (statistics, cost, ...). User protocols and ISDN signalling are 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.

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;

– the information retrieval on an NCO.

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 an NCO, a unique identifier, the 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 functionality is User Plane protocol dependent. A detailed description of the condition and procedures for the use of the grouping functionality is provided in Part 3.

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-coordinatedNCO (C/U3), user and control plane 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 5.3.3) and data management (see 5.3.4) provide functionality to manage connections with these NCOs.

Five types of external equipments are defined:

1) External equipment without telephony hook control – This type of external equipment only contains the transceivers. In this case, the PUF is in charge of managing the ISDN connection.

2) External equipment with telephony hook control – In this case, all telephony hook events are available at the PCI level and the PUF is in charge of managing the ISDN connection.

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

4) External equipment with keypad and with or without telephony hook control – In this case all dialling events, all telephony hook control events are available at the PCI and the PUF is in charge of managing the ISDN connection.

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

All these types of external equipment are connected to the NAF by the means of a proprietary connection which is out of the scope of this part 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, DTMF codes can be sent via the keypad.

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

Type 5 external equipment is 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 a 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 goes off-hook via the handset and the PUF issues the overlap or block dialling;
  - the PUF issues the overlap or block dialling and the user goes off-hook via the handset.

– *Case of incoming calls:*

  - the user goes off-hook via the handset and the PUF receives a control plane message to inform it;
  - the PUF answers the incoming call and the user goes off-hook via the handset.

– *Case of local close down:*

  - 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 goes on-hook via the handset.

– *Case of remote close down:*

  - the PUF receives a control plane message and the user goes on-hook via 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 closed down in different ways:

– *Case of the outgoing calls:*

  - the user goes off-hook via the handset and the PUF issues the overlap or block dialling;
  - the user goes off-hook via 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 goes off-hook via 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 goes off-hook via the handset and the PUF receives a control plane message to inform it of the off-hook state;
- the PUF answers to the incoming call and the user goes off-hook via the handset.

– *Case of local breakdown:*

- the user goes on-hook via the handset and the PUF receives a control plane message to inform it of the on-hook state;
- the PUF releases the call and the user goes on-hook via the handset.

– *Case of remote breakdown:*

- the PUF receives a control plane message and the user goes on-hook via 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 dependent of the implementation of the manufacturer specific feature and is, therefore, not covered in this part.

5.3.3 Connection management

The connection management functionality covers two aspects:

– the set-up and breakdown of physical connections;

– the access and usage of supplementary services.

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

The control plane of the ISDN-PCI provides the functionality defined by the physical 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 reuse 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 designated 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.

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 by the signalling part. Furthermore, some external equipments are able to manage ISDN signalling when the host is off.

In 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 can not be accommodated.

5.3.3.2 Support of supplementary services

Supplementary services provide additional functionality related to the connection management. The description and the use of supplementary services is for further study.

Supplementary services, as provided by the connection management of ISDN, are available to the PUF when PUF coordination case 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 the coordination function is handled by the NAF is for further study.

5.3.4 Data management

The data management functionality covers two aspects:

- 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 access to various User Plane protocols running in the ISDN network layer. Depending on the selected User Plane protocol, the User Plane provides access to a network layer (Layer 3), link layer (Layer 2) or transparent (Layer 1) connections.

Selection of the User Plane protocol is possible using the resource management functionalities (creation or modification of an NCO).

For every type of connection, it is important that there exists a signalling connection before any data access can be done. In general, unless a PUF makes use of the coordination function provided by the NAF, the establishment of the signalling connection is achieved by using the control plane functionality, whereas the establishment of data access is achieved by use of the User Plane functionality.

When using a connection with a transparent User Plane protocol and 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 User Plane messages.

A detailed description of the available protocols and of the corresponding User Plane messages, sequencing and parameters can be found in Part 3 [4].

5.4 Relating functionality to planes

When relating the functionality as described in 5.3, 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 are not mandatory for a NAF to supply.

In the description of the planes there are indications on 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 like 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 1 provides an overview on administration plane operations.

Table 1 – 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
Change protocol (Note)	Change the User Plane protocol on the established B-channel
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, there exists no clear separation of operations like in the administration plane. It shall be possible to distinguish between a number of operational groups in the control plane. Table 2 provides an overview on control plane operational groups.

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. A detailed description of the operations available for the various possible access (transparent, link layer, network layer) can be found in Parts 3 to 6.

Table 2 – 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
Additional information (Note)	Provide access to additional information during a call
NOTE - These operational groups are optional for the NAF.	

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 within a system that 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 sends 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 an 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 which use this method are more complex in design than those that do not .

– *Deregistration Phase*

When a PUF does not need to exchange messages with a NAF, it disassociates from the NAF. This phase is called the Deregistration 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 resources of the global system.

Table 3 gives the list of functions grouped into their respective phases.

Table 3 – 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 3 presents a state diagram for the ISDN-PCI function calls.

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

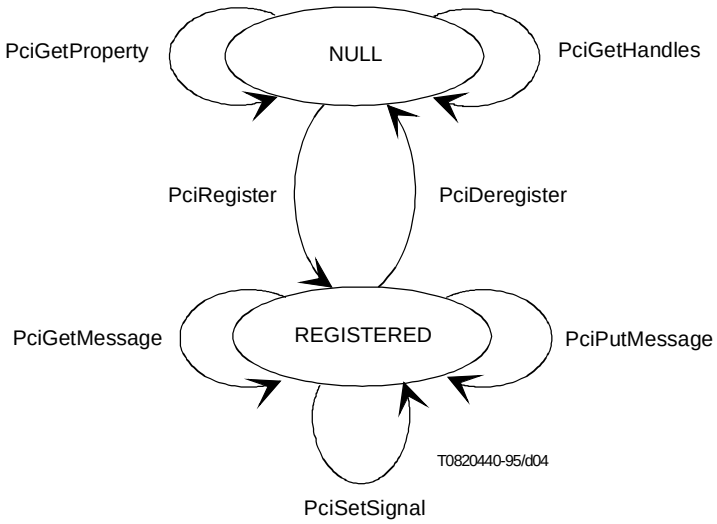


Figure 3 – ISDN-PCI function calls order

5.6 Total interaction overview

As an example of the sequencing of operations, Figures 4 and 5 present a chronological interaction overview of the actions the PUF shall 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 7.

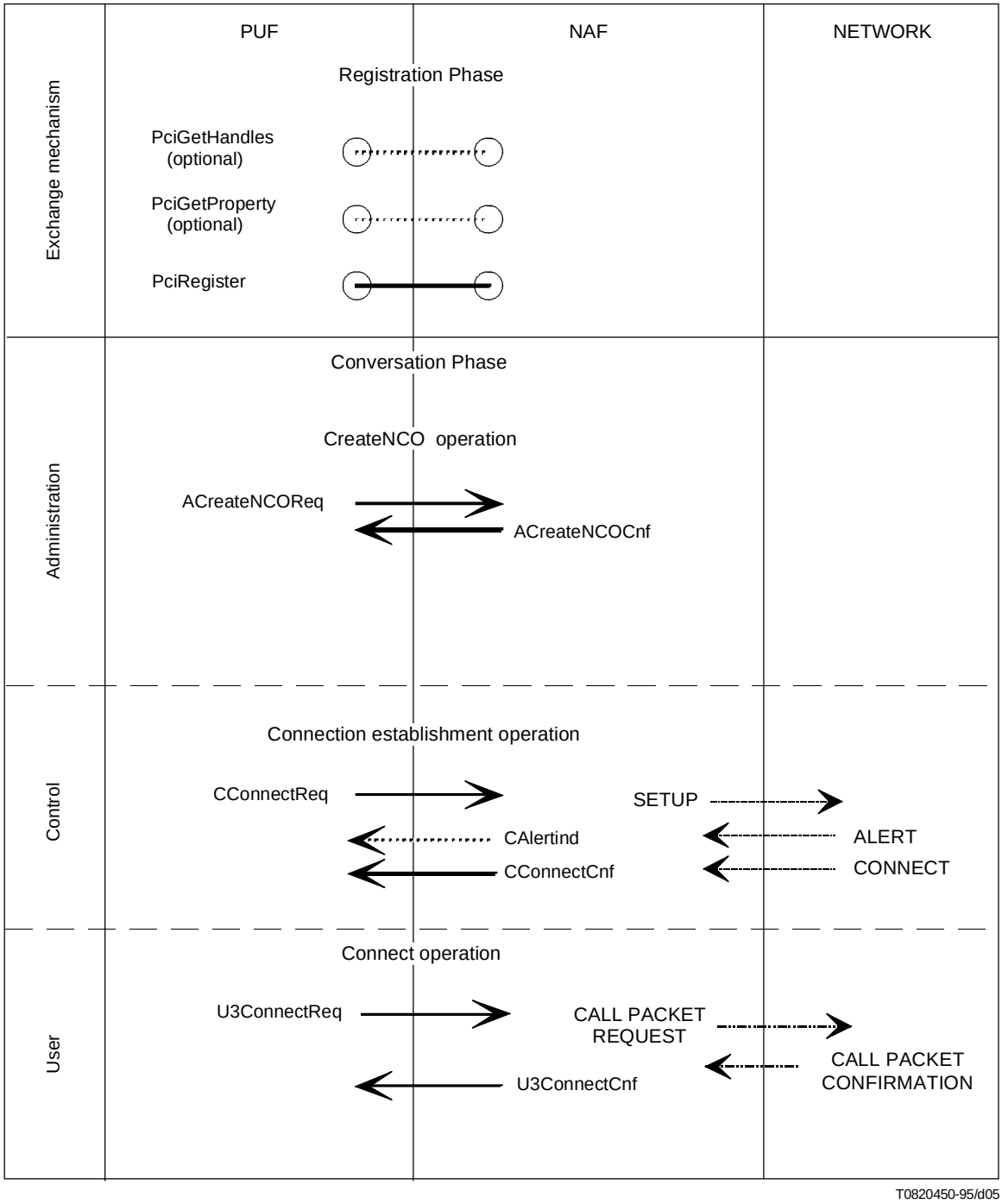


Figure 4 – Sample of sequencing operations – PUF coordination case

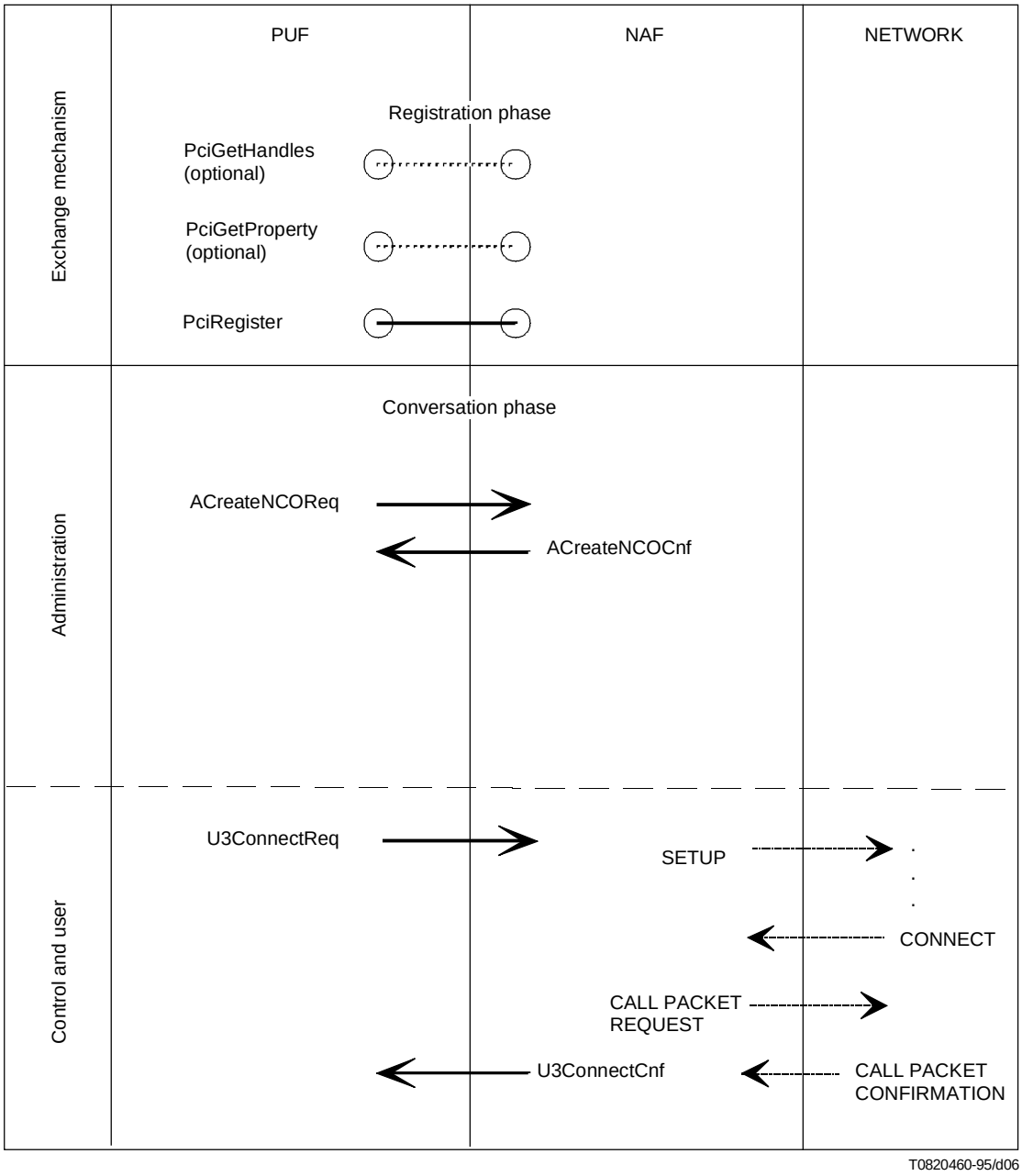


Figure 5 – Sample of sequencing operations – NAF coordination case

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 8 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 association 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 association.

Identifiers related to the Conversation Phase of the communication between PUF and NAF. For details of these identifiers refer to clause 7.

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 a Network Connection Object (NCO). The complete list of static attribute sets is defined in Annex F.

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 a Network Connection Object (NCO). The complete list of static attribute sets is defined in Annex F.

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 a Network Connection Object (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 Network Connection Objects. It is User Plane protocol dependent.

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 will 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 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 dependent. Error information is returned by a mechanism particular to the protocol in use. These mechanisms are described in 7.8.

6 Information encodings

In 7.6, 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 Type-Length-Value (TLV) coding as follows:

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

Fields included into the parameter are coded as structured information. The order of this structured information is defined by the order of the parameter itself in 7.6. Omitted fields reduce the size of the parameter.

Values in parenthesis are decimal.

7 Description of ISDN-PCI messages

As described in 5.5, "PUF-NAF Interactions", the exchange of messages is realized through two 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 contain 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 clause 6, "Information encodings".

7.1 Conventions

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

7.1.1 Address conventions

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

- The called address refers to the address the sender desires to be connected to.
- The calling address refers to the local address of the sender.

7.1.2 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.

7.1.3 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 of the same plane 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.

7.1.4 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 dependent implementation. It refers to the basic information unit in all systems.

7.1.4.1 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 7.6 of this part of this appendix.

7.1.4.2 Parameter repetition

Parameters in a message can be repeated. The number of repetitions is fixed by the Network or the user protocol used.

7.1.4.3 Parameter checking

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

7.1.5 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 will take place.

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

7.2 Administration plane messages

The administration plane messages are divided into the following classes:

1) management of network connection objects and error report message;

2) management of connection security;

3) NAF manufacturer messages;

4) protocol change messages.

For management of Network Connection Objects (NCOs) there are messages available for creating and destroying a connection object. During creation of a NCO static or dynamic attribute and address sets are linked to the created NCO. On conclusion of the creation of an NCO, an NCO identifier (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 F. For reporting of error information a single message is provided by the NAF. This is used to report general error conditions.

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.

To access manufacturer specific features there are messages available. 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. The information exchanged between PUF and NAF is dependent on the implementation of the feature and is, therefore, not covered in this part.

There are messages available to request a change of the User Plane protocol associated with an NCO. 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.

All request messages of the administration plane may contain a request identifier (RequestID). This identifier, if assigned by a PUF on a request message, is returned by the NAF on the related confirm message.

Table 4 gives an overview of administration plane messages. The messages themselves are described in detail in the following subclauses.

Table 4 – Administration plane messages

Mess. identificateur	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
113	4	AChangeNCOReq	Request for a change on an existing NCO
114	4	AChangeNCOCnf	Confirmation on changing the existing NCO

7.2.1 ACreateNCOReq

Class: 1 (Basic Class).

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

The PUF has to provide a 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. It is absent if the NCOType value is U3 or else it is optional.
UDirection	C	Determines how NCO shall be used, for the User Plane. This parameter is User Plane Protocol dependent.
CAttributeName	C	Name of static control plane attribute
CAttribute parameters	C	Control plane attribute parameters Mutually exclusive with CAttributeName; see Table 10 for more details.
UAttributeName	C	Name of static User Plane attribute
UAttribute parameters	C	User Plane attribute parameters Mutually exclusive with UAttributeName; see the ISDN-PCI relevant User Plane protocol specification for more details.
CAddress parameters	O	Control plane address; see Table 13 for more details.
UAddress parameters	O	User Plane address; see the ISDN-PCI relevant User Plane protocol specification for more details.
GroupID	C	Required if NCO is to be grouped. This parameter is User Plane Protocol dependent.
SelectorID	O	Helps the NAF to select the right NCO
CPMessageMask	O	ISDN message filter. If not provided, the PUF will receive any control plane message.
CPParameterMask	O	ISDN control plane parameter filter. If not provided, the PUF will receive any control plane parameter.

Remarks: See also 7.7 on usage of the NCO.

Related: ACreateNCOCnf.

7.2.2 NCOType and conditional parameter specification

This subclause defines the NCOTypes usable within a ACreateNCOReq.

Currently there are 4 types of NCOs defined. These types are shown in Table 5.

For NCOTypes supporting a User Plane access, Tables 5 and 6 only show the general forms of the NCOtype. A detailed description of the specific NCOType usable with a specific User Plane protocol can be found in Part 3 [4].

Table 5 – NCOTypes

NCOType	NCO allows PUF …
C	… signalling access only
C/U	... signalling and User Plane access (PUF coordination functionality)
U3	… U3 User Plane access with signalling in charge to the NAF (NAF coordination functionality)
U3G	… User Plane access to additional virtual circuits. This NCO shall be grouped to an already created U3 or C/U type NCO.

Table 6 shows which conditional parameters shall be specified in the ACreateNCOReq message in relation to the selected NCOType.

Table 6 – Specification of conditional ACreateNCOReq message parameters

NCOType	SigAttribute type	UsrAttribute type	SigAddress type	UsrAddress type	GroupID
C	C Attribute		C Address		
C/U	C Attribute	U Attribute	C Address	U Address	
U3	C Attribute	U Attribute	C Address	U Address	
U3G		U Attribute		U Address	Reference to NCO (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/U shall be provided. Therefore, the creation of such an NCO type shall have been carried out successfully in order to have the GroupID available.					

7.2.3 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.

7.2.4 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.

7.2.5 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.

7.2.6 AGetNCOInfoReq

Class: 1 (Basic Class).

Description: Request message for getting information about an NCO. Each NCO is characterized by some attributes (see 7.6.47, "Administration Attribute set parameters") which are accessible from the PUF thanks to this request and its confirmation.

Parameter:

Name	Required	Comment
NCOID	M	Identifier of NCO requested on

Related: AGetNCOInfoCnf.

7.2.7 AGetNCOInfoCnf

Class: 1 (Basic Class).

Description: Confirmation message sent by the NAF to the PUF for answering an AGetNCOInfoReq. It contains the information (see 7.6.47, "Administration Attribute set parameters") 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.

7.2.8 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.

7.2.9 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 of 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 provided by the NAF. 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 dependent on the security algorithm activated.

Parameters:

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

Related: ASecurityCnf.

7.2.10 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.

7.2.11 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)

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

Related: None.

7.2.12 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

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

Related: AManufacturerReq.

7.2.13 AChangeNCOReq

Class: 4 (Additional class).

Description: This message is used to change parameter(s) of an existing NCO.

Only NCO of C, C/U1 or C/U3 type may have the NCOType parameter changed. If the NCO refers to an active connection, the NAF changes the protocol only in stable conditions.

Parameters:

Name	Required	Comment
RequestID	O	Request identifier generated by the PUF
NCOID	M	NCO to change
UDirection	O	Determines how the NCO shall be used for the User Plane
UAttributeName	O	Name of static User Plane attribute
UAttribute Parameters	O	User Plane Attribute parameters Exclusive with UAttributeName; see the ISDN-PCI relevant User Plane protocol specification for more details.
NCOType	O	Specification of the new NCO type
UAddress parameters	C	User Plane address; see the ISDN-PCI relevant User Plane protocol specification for more details.

Related: AChangeNCOCnf.

7.2.14 AChangeNCOCnf

Class: 4 (Additional class).

Description: Confirmation message sent by the NAF to the PUF for answering a AChangeNCOReq. If successful, the changes requested are immediately operational.

Parameters:

Name	Required	Comment
RequestID	O	Request identifier generated by the PUF
NCOID	M	NCO to change
CompletionStatus	M	Completion status of the change NCO operation

Related: AChangeNCOReq.

7.3 Control plane messages

7.3.1 Introduction

7.3.1.1 Control Messages classes

The control plane messages are divided into seven 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;
7) additional information.

As described in 7.1.3, 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 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 7 gives an overview of control plane messages, the class they belong to and their message identifier.

Table 7 – 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
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 is 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 depressed key
238	6	CExtEquipOffHookInd	Indicate that the handset is off-hook
240	6	CExtEquipOnHookInd	Indicate that the handset is on-hook
241	7	CAddInfoReq	Request to send additional information related to a call
242	7	CAddInfoInd	Indicate that additional information related to a call has been received

7.3.1.2 Sequencing of control plane messages

Figures 6, 7, 8 and 9 present the state diagrams affecting the state of a PUF connection.

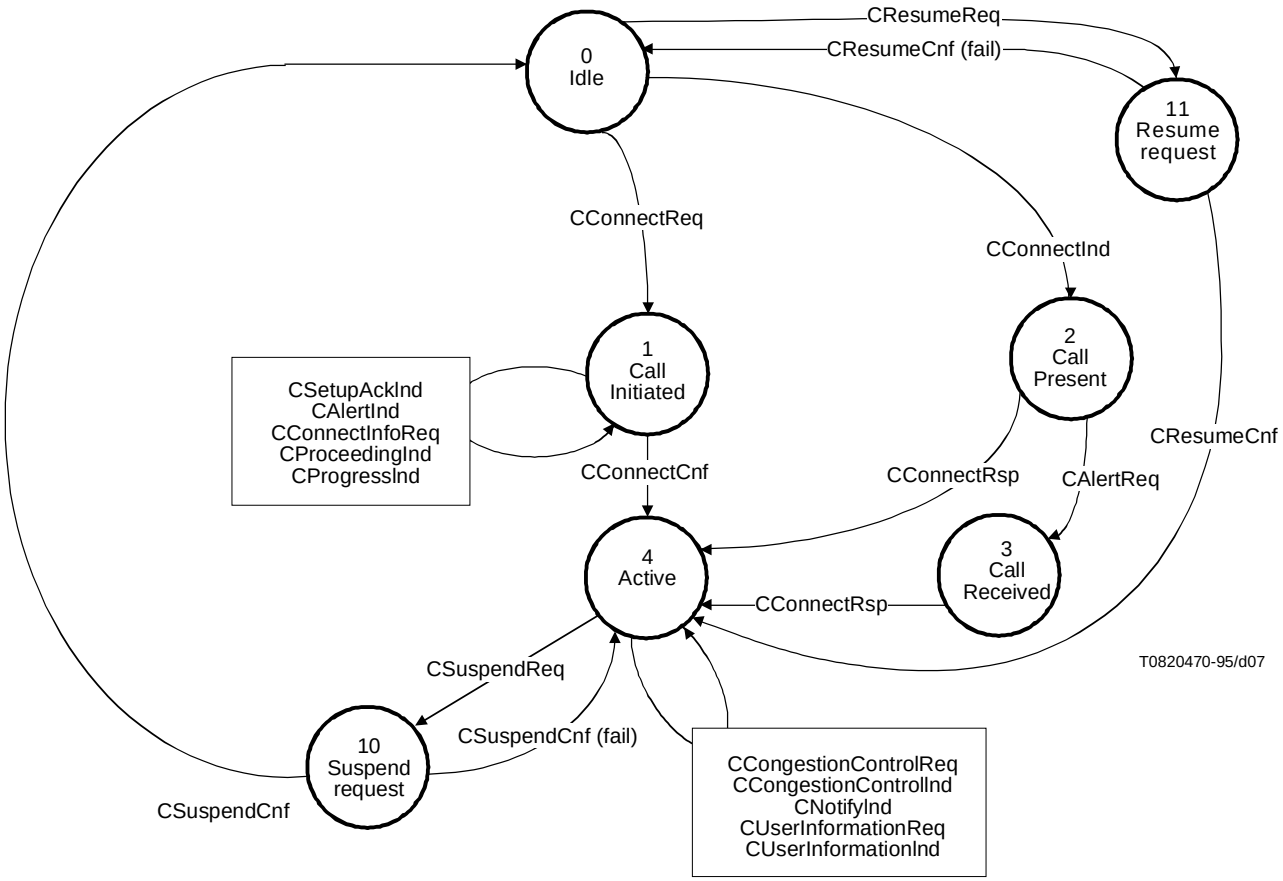


Figure 6 – State diagram of a control plane (no external equipment or external equipment type 1)

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

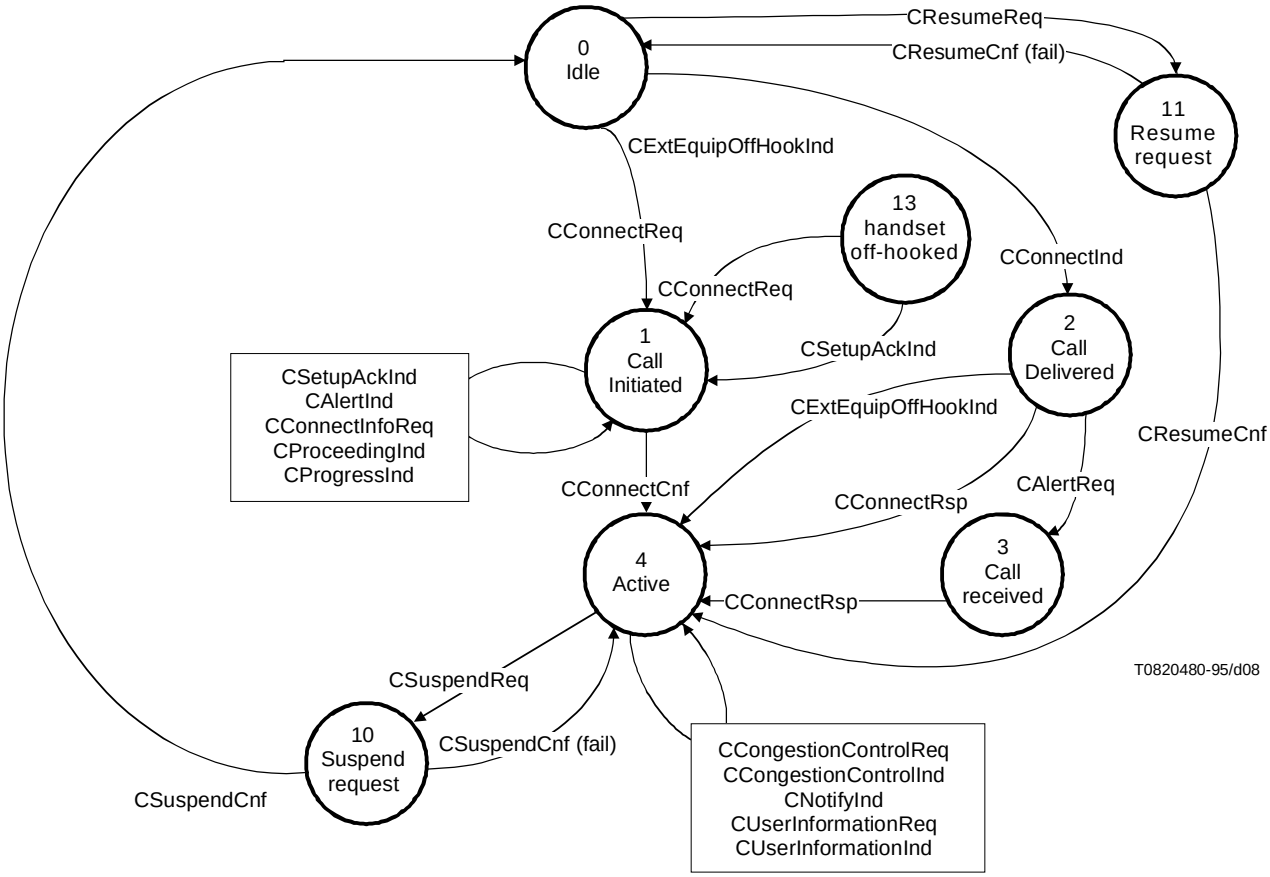


Figure 7 – State diagram of a control plane (external equipment type 2 or 3)

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

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

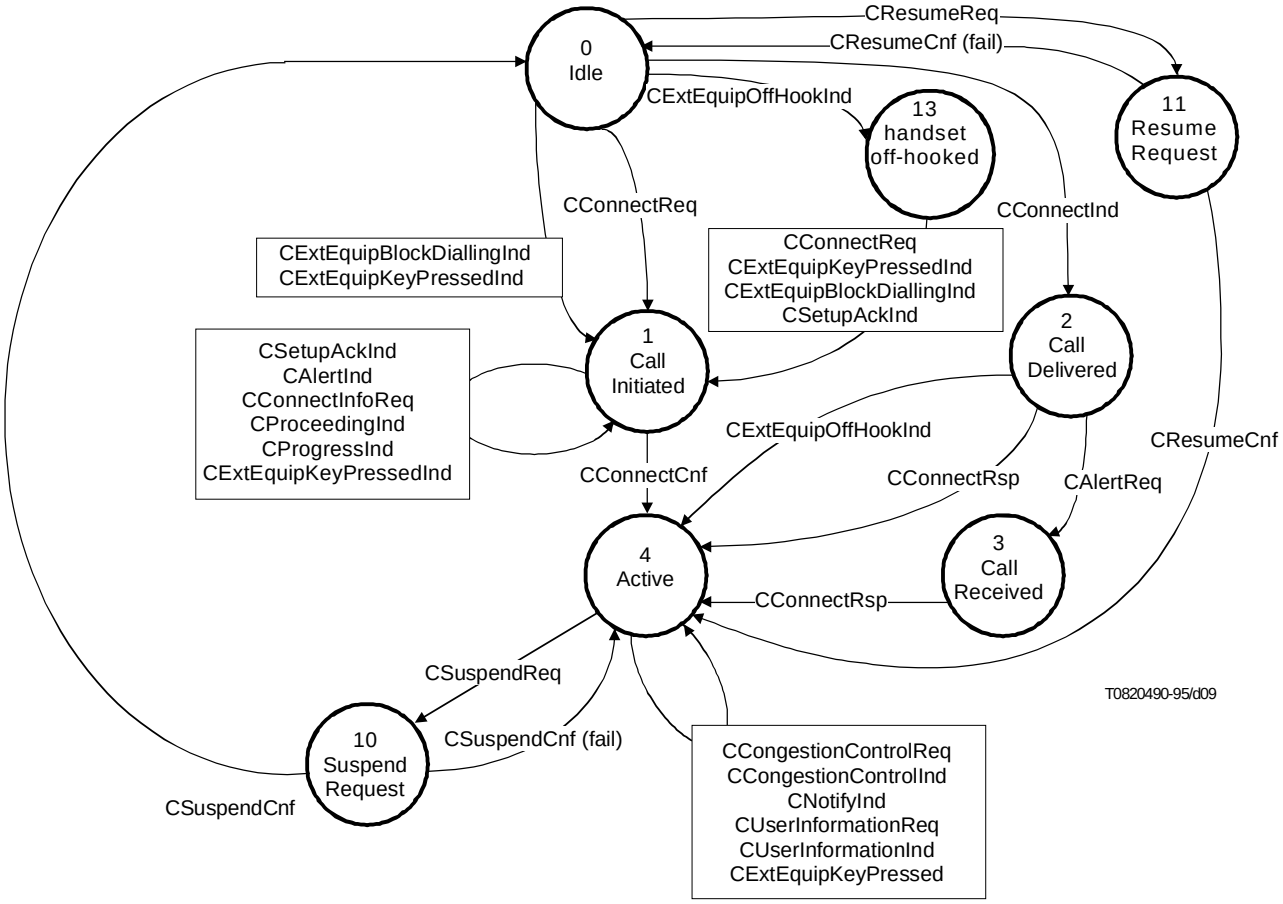


Figure 8 – State diagram of a control plane (external equipment type 4 or 5)

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

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

NOTE 3 – In Figures 6, 7 and 8 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.

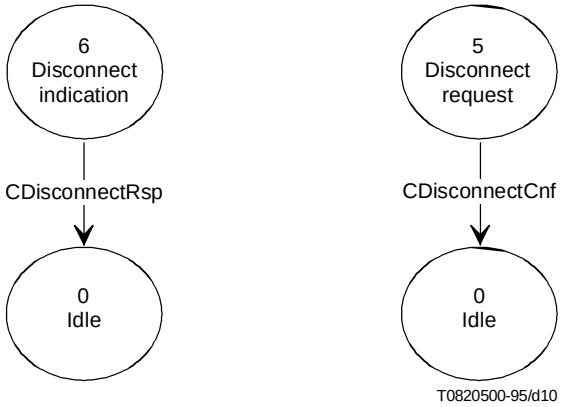


Figure 9 – State diagram of a control plane (connection) – Disconnection

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 I.1.

Any CStatusInd message may cause a transition to state 0, if provided by the network.

CAddInfoReq or CAddInfoInd message do not change the state of a NCO.

NOTE 1 – Figures 6, 7, 8 and 9 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.

7.3.2 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
ProgressIndicator	O	Details concerning call progress
UserToUserInfo	O	Information to be exchanged between ISDN users

Remarks: The availability of UserToUserInfo depends on the user-to-user service level. See 7.3.35 for details on user-to-user information.

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

7.3.3 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
ProgressIndicator	O	Details concerning call progress
Display	O	Information provided by the Network to be displayed
Signal	O	Information provided by the Network regarding tones
UserToUserInfo	O	Information to be exchanged between ISDN users

Remarks: The availability of UserToUserInfo depends on the user-to-user service level. See 7.3.35 for details on user-to-user information.

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

7.3.4 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 either specified in the message, or have been specified in the address parameters used 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 2)
CalledSubaddress	O	Remote subaddress (Notes 1 and 2)
ChannelIdentification	O	Used by PUF to indicate type of requested Channel. See ChannelIdentification parameter in 7.6.12 for details of supported values. If not provided default is any channel (Note 1).
BearerCap	O	Transmission capability required from channel (Note 1)
LLC	O	Lower Layer Compatibility information element (Note 1)
HLC	O	High Layer Compatibility information element (Note 1)
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 – Either a CalledNumber or a CalledSubaddress parameter – in the message or during the NCO creation – shall be supplied.		

Remarks: The availability of UserToUserInfo depends on the user-to-user service level. See 7.3.35 for details on user-to-user information.

Related: CConnectCnf, CAlertReq, CAlertInd, CConnectInfoReq.

7.3.5 CConnectInd

Class: 1 (Basic class).

Description: This message offers an incoming call to all appropriate PUFs (see 7.7.1). 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
Signal	O	Information provided by the Network regarding tones
UserToUserInfo	O	Information to be exchanged between ISDN users

Remarks: 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 7.3.35 for details on user-to-user information.

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

7.3.6 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 7.6.12 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 present.
LLC	O	Lower Layer Compatibility information element
Facility	O	Supplementary services operation or information
ProgressIndicator	O	Details concerning call progress
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 7.3.35 for details on user-to-user information.

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

7.3.7 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	Lower 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
ProgressIndicator	O	Details concerning call progress
Signal	O	Information provided by the Network regarding tones
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 7.3.35 for details on user-to-user information.

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

7.3.8 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 7.3.35 for details on user-to-user information.

Related: CDisconnectInd, CDisconnectRsp, CDisconnectCnf.

7.3.9 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
ProgressIndicator	O	Details concerning call progress
Signal	O	Information provided by the Network regarding tones
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 7.3.35 for details on user-to-user information.

Related: CDisconnectReq, CDisconnectRsp, CDisconnectCnf.

7.3.10 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 reused 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.

7.3.11 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
Signal	O	Information provided by the Network regarding tones

Related: CDisconnectReq, CDisconnectInd, CDisconnectRsp.

7.3.12 CProgressInd

Class: 1 (Basic class).

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

Parameters:

Name	Provided	Comment
NCOID	M	Identifies the call
ChannelIdentification	O	Identification of the channel used
CauseToPUF	O	Reason for the message
Display	O	Information provided by the network to be displayed
ProgressIndicator	M	Details concerning call progress
UserToUserInfo	O	Information to be exchanged between ISDN users

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

7.3.13 CStatusInd

Class: 1 (Basic class).

Description: With this message, the PUF shall be informed that a signalling protocol error, as defined in 7.8.8, has occurred.

Parameters:

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

Related: None.

7.3.14 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.

7.3.15 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 similar to dialling on a keypad.

Parameters:

Name	Required	Comment
NCOID	M	Identifies the call
CalledNumber	M	Part of the remote address (Note 1)
NumberComplete	O	Indicates this message contains the last part of the called number from a PUF point of view (Note 2).
NOTE 1 – 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). NOTE 2 – If this parameter is included, no more CConnectInfoReq messages will be accepted by the NAF for this call.		

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

Related: CConnectReq, CProceedingInd, CSetupAckInd.

7.3.16 CProceedingInd

Class: 2 (Additional class).

Description: The PUF receives this message when no more establishment information will 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.

7.3.17 CUserInformationReq

Class: 3 (Additional class).

Description: This message allows a PUF to request user-to-user information be sent 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 7.3.35 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 follows
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 7.3.35 for details on user-to-user information.

Related: CUserInformationInd, CCongestionControlReq, CCongestionControlInd.

7.3.18 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 7.3.35 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 follows.
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 7.3.35 for details on user-to-user information.

Related: CUserInformationReq, CCongestionControlReq, CCongestionControlInd.

7.3.19 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 7.3.35 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. This message has only a local meaning.

Related: CUserInformationReq, CUserInformationInd, CCongestionControlInd.

7.3.20 CCongestionControlInd

Class: 3 (Additional class).

Description: This message allows a NAF to indicate to a PUF that a flow control operation 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 is 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 7.3.35 for details on user-to-user information.

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

This message has only a local meaning. The flow control for each direction shall be operated independently.

Related: CUserInformationReq, CUserInformationInd, CCongestionControlReq.

7.3.21 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.

Parameter:

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.

7.3.22 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 reused. 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.

7.3.23 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 get informed if the suspended connection is reconnected.

Parameter:

Name	Required	Comment
NCOID	M	Identifies the call

Related: CSuspendReq, CSuspendCnf, CResumeCnf, CNotifyInd.

7.3.24 CResumeCnf

Class: 4 (Additional class).

Description: This message is the answer to a 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 succeeds; – 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 might 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.

7.3.25 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.

7.3.26 CFacilityReq

Class: 5 (Additional class).

Description: This message allows the PUF 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 of the facility parameter refer to 7.6.26.

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.

7.3.27 CFacilityInd

Class: 5 (Additional class).

Description: This message provides to the PUF the facility coming 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 7.6.26.

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.

7.3.28 CExtEquipAvailabilityInd

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.

7.3.29 CExtEquipBlockDiallingInd

Class: 6 (Additional class).

Description: With this message, the PUF gets the dialling information input 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.

7.3.30 CExtEquipKeyPressedInd

Class: 6 (Additional class).

Description: With this message, the PUF gets the dialling information input 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.

7.3.31 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 6, 7 and 8).

Parameter:

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

Related: CExtEquipOnHookInd.

7.3.32 CExtEquipOnHookInd

Class: 6 (Additional class).

Description: With this message, the PUF is informed that the handset of the external equipment is on-hook. 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 6, 7 and 8).

Parameter:

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

Related: CExtEquipOffHookInd.

7.3.33 CAddInfoReq

Class: 7 (Additional class).

Description: This message allows a PUF to send additional information related to a call. The overlap sending information is handled via the class 2 CConnectInfoReq message. This message may be used to convey network related information (e.g. particular identification procedure).

Parameters:

Name	Required	Comment
NCOID	M	Identifies the call
AdditionalInformation	M	Conveys network related information

Related: CAddInfoInd.

7.3.34 CAddInfoInd

Class: 7 (Additional class).

Description: This message allows a NAF to send additional information related to a call provided by the network (e.g. particular identification procedure).

Parameters:

Name	Required	Comment
NCOID	M	Identifies the call
Display	O	Information provided by the network to be displayed
AdditionalInformation	M	Conveys network related information

Related: CAddInfoReq.

7.3.35 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 Recommendation Q.931 [1] three user-to-user service levels are defined:

– *Service 1*:

user-to-user information exchanged during the set-up and clearing phase of a call.

– *Service 2*:

user-to-user information exchanged during call establishment.

– *Service 3*:

user-to-user information exchanged 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 Recommendation Q.931 [1].

7.4 Implementation of supplementary services

An ISDN-PCI Recommendation describing in detail the way to make use of supplementary services is for further study.

7.5 User Plane messages

The User Plane messages provide access to different User Plane protocol stacks. Depending on the selected User Plane protocol the User Plane provides access to a network layer (Layer 3), link layer (Layer 2) or transparent (Layer 1) connection. A detailed description of the available protocols and of the corresponding User Plane messages, sequencing and parameters can be found in Parts 3 to 6.

7.6 Message parameters

This subclause describes parameters for messages of the administration and the control plane. Parameters for messages of the User Plane are described in Parts 3 to 6.

Parameters within this subclause are alphabetically ordered. The type numbering is taken from the Appendix III.

7.6.1 AdditionInformation

Description: This parameter is used to pass additional information from/to the PUF. The information is transparently conveyed to/from the network. See related network or NAF documentation for more details.

Type: 80.

Field	Field type	Direction	Required	Comment
Value	IA5-string	B	M	128 is the maximum length. It may be reduced due to network constraints.

7.6.2 Algorithm

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

Type: 1.

Field	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 provided by the NAF. "nosecurity": this value for this parameter indicates that security is no longer needed for the connection. 16 bytes is the maximum length
NOTE – A detailed description of the possible use of this parameter can be found in Part 3 [2].				

7.6.3 BearerCap

Description: This parameter is used to pass bearer capability to/from the PUF and optionally layer 1 information if the LLC parameter, described in 7.6.31, is provided in the call.

Type: 3.

Field	Field type	Direction	Required	Comment
BearerCap	Octet-string	B	M	Bearer capability information element Maximum length is 12 octets
NOTE – Values for this field are defined in the Recommendation Q.931.				

7.6.4 CalledNumber

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

Type: 7.

Field	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	C	20 bytes is the maximum length. May be absent in the case of overlapping numbering or if associated with the CConnectReq message. Otherwise mandatory.
NOTE – In the message exchange from PUF to NAF, this parameter shall either be supplied in the NCO or in the appropriate message.				

7.6.5 CalledSubaddress

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

Type: 8.

Field	Field type	Direction	Required	Comment
NumberType	Octet	B	M	Default (255) – Default is nsap 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 bytes is the maximum length

7.6.6 CallingNumber

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

Type: 11.

Field	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 Calling Line Identification Presentation supplementary service.

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 Calling Line Identification Presentation supplementary service 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 Calling Line Identification Presentation supplementary service and specifying an incomplete number for Direct-Dialling-In.

7.6.7 CallingSubaddress

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

Type: 12.

Field	Field type	Direction	Required	Comment
NumberType	Octet	B	M	Default (255) – Default is nsap 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

7.6.8 CAttributeName

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

Type: 13.

Field	Field type	Direction	Required	Comment
AttributeName	IA5-string	P	M	16 bytes is the maximum length

7.6.9 CauseToNAF

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

Type: 14.

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

7.6.10 CauseToPUF

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

Type: 15.

Field	Field type	Direction	Required	Comment
Cause	Octet	N	M	Cause value
Standard	Octet	N	M	Default (255) – Default is ITU ITU (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 Q.931 Q.931 (0) X.21 (3) X.25 (4)
Diagnostics	Octet-string	N	C	Depends on the cause value Length is fixed to 2 The lower octet contents least significant byte

7.6.11 CDirection

Description: This parameter is used to pass information concerning the usage of a particular NCO to the NAF, for the control plane part. If this parameter is absent at the NCO creation time, the value retained for this NCO will be both (3).

Type: 16.

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

7.6.12 ChannelIdentification

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

Type: 17.

Field	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.

7.6.13 ChargingInfo

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

Type: 18.

Field	Field type	Direction	Required	Comment
Tag	Octet	N	M	charginginfo (3) chargingerror (4) (See 7.6.26: coding of FacilityTag).
Value	Octet-string	N	C	Length and content depend on the Tag. Absent if Tag is chargingerror. (See 7.6.26: coding of FacilityValue).

7.6.14 CompletionStatus

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

Type: 19.

Field	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 7.8 for more details. Length shall be in the range 0 to 16 octets.

7.6.15 CongestionLevel

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

Type: 20.

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

7.6.16 ConnectedNumber

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

Type: 21.

Field	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 bytes is the maximum length

7.6.17 ConnectedSubaddress

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

Type: 22.

Field	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 bytes is the maximum length

7.6.18 CPMessageMask

Description: This parameter is set by the PUF to indicate which Control Plane messages are not intended to be received from the NAF. Not all Control Plane messages can be filtered.

This parameter is coded as a bit field. The default value is 0 for all bits, which means that the PUF will receive any message coming from the network.

Type: 73

Field	Field type	Direction	Required	Comment
CPMessageMask	Octet String	P	M	Fixed length is 2 Bit 1 CAlertInd Bit 2 CProgressInd Bit 3 CSetupAckInd Bit 4 CProceedingInd Bit 5 CUserInformationInd Bit 6 CCongestionControlInd Bit 7 CNotifyInd Bit 8 CFacilityInd Bits 9 to 16 are reserved

7.6.19 CPParameterMask

Description: This parameter is set by the PUF to indicate which Control Plane message parameters are not intended to be received from the NAF. Not all Control Plane message parameters can be filtered.

It is coded as a bit field. The default value is 0 for all bits, which means that the PUF will receive any Control Plane message parameter coming from the network.

Type: 72

Field	Field type	Direction	Required	Comment
CPParameterMask	Octet String	P	M	Fixed length is 4 Bit 1 CauseToPUF Bit 2 ChannelIdentification Bit 3 DateTime Bit 4 Display Bit 5 Facility Bit 6 High Layer Compatibility Bit 7 Low Layer Compatibility Bit 8 UserToUserInfo Bit 9 Signal Bit 10 ProgressIndicator Bits 11 to 31 are reserved

7.6.20 DateTime

Description: This parameter is used to pass date and time information to the PUF. This information is provided by the Network in the call establishment operation or by the NAF at the NCO creation time.

Type: 23.

Field	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

7.6.21 Display

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

Type: 24.

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

7.6.22 ExtEquipAvailability

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

Type: 25.

Field	Field type	Direction	Required	Comment
Availability	Boolean	N	M	State of the external equipment: TRUE – Equipment available FALSE – Equipment unavailable

7.6.23 ExtEquipBlockDialling

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

Type: 26.

Field	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 bytes is the maximum length

7.6.24 ExtEquipKeypressed

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

Type: 27.

Field	Field type	Direction	Required	Comment
Keypressed	Octet	N	M	Keypad information: (0 to 9) Numeric key (10) "*" key (11) "#" key (12) "A" key (13) "B" key (14) "C" key (15) "D" key

7.6.25 ExtEquipName

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

Type: 28.

Field	Field type	Direction	Required	Comment
Type	Octet	B	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 types are described in Annex A
Name	IA5-string	B	M	maximum length is 16 "DEFAULT" – Use first defined external equipment of specified type

7.6.26 Facility

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

Type: 30.

Table 9 – 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 only be invoked in the CConnectReq.
		NoChargingInfoAvailable (128)	There is no charging information available

7.6.27 GroupID

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

Type: 33.

Field	Field type	Direction	Required	Comment
GroupID	Octet string	B	M	The value is unique for a PUF/NAF relation 4 octets is the fixed length
NOTE – A detailed description of the possible use of this parameter can be found in Parts 3 to 6.				

7.6.28 High Layer Compatibility (HLC)

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

Type: 34.

Field	Field type	Direction	Required	Comment
Standard	Octet	B	M	Default (255) – Default is ITU-T ITU-T (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)

7.6.29 Key

Description: Key to be used for the security algorithm.

Type: 36.

Field	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

7.6.30 Keypad

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

Type: 37.

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

7.6.31 Low Layer Compatibility (LLC)

Description: This parameter is used to pass a subset of 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.

Field	Field type	Direction	Required	Comment
Negotiation	Boolean	B	M	TRUE – Negotiation is allowed FALSE – Negotiation is not allowed
Layer2protocol	Octet	B	M	0-31 255 = unspecified It refers to the Octet 6 of the LLC information element
layer2optional	Octet	B	M	0-127 255 = unspecified It refers to the Octet 6a of the LLC information element
Layer3protocol	Octet	B	M	0-31 255 = unspecified It refers to the Octet 7 of the LLC information element
layer3optional	Octet	B	M	0-127 255 = unspecified It refers to the Octet 7a of the LLC information element

7.6.32 ManufacturerCode

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

Type: 47.

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

7.6.33 NCOID

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

Type: 49.

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

7.6.34 NCOType

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

Type: 50.

Field	Field type	Direction	Required	Comment
Identifier	Octet	B	M	cset (1) – Signalling access only (Note)
NOTE – More values of the NCOType to be used in case of different types of User Plane Access can be found in Parts 3 to 6.				

7.6.35 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.

Field	Field type	Direction	Required	Comment
Value	Octet	N	M	suspended (1) resumed (2) callwaiting (3)
NOTE – Other values are network dependent.				

7.6.36 NumberComplete

Description: This parameter is used by the PUF to indicate to the NAF that a called number is complete.

Type: 79.

Field	Field type	Direction	Required	Comment
Value	Octet	P	M	Number complete (1)
NOTE – Only one value is available.				

7.6.37 ProgressIndicator

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

Type: 53.

Field	Field type	Direction	Required	Comment
Standard	Octet	N	M	ITU-T (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.

7.6.38 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.

Field	Field type	Direction	Required	Comment
Identifier	Octet-string	B	M	Internal ID provided by the PUF Length is fixed to 4 octets.

7.6.39 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.

Field	Field type	Direction	Required	Comment
Identifier	Octet-string	P	M	Internal ID provided by the PUF Length is fixed to 4 octets.

7.6.40 Signal

Description: This parameter is used to optionally convey information to the PUF regarding tones and alerting signals.

Type: 81.

Field	Field type	Direction	Required	Comment
Signal value	Octet	N	M	dial tone on (0) ring back tone on (1) intercept tone on (2) network congestion tone on (3) busy tone on (4) confirm tone on (5) answer tone on (6) call waiting tone on (7) off-hook warning tone on (8) tones off (63) alerting on – Pattern 0 (64) (Note) alerting on – Pattern 1 (65) (Note) alerting on – Pattern 2 (66) (Note) alerting on – Pattern 3 (67) (Note) alerting on – Pattern 4 (68) (Note) alerting on – Pattern 5 (69) (Note) alerting on – Pattern 6 (70) (Note) alerting on – Pattern 7 (71) (Note) alerting off (79)
NOTE – The use of these patterns is network dependent.				

7.6.41 TEI

Description: This parameter is used to access a permanent link to a data packet switch (packet connection in D-channel).

Type: 61.

Field	Field type	Direction	Required	Comment
Value	Octet	B	M	

7.6.42 UProtocol

Description: This parameter is used to select the User Plane protocol.

Type: 62.

Field	Field type	Direction	Required	Comment
L3Protocol	Octet	P	M	Default (255) – (Note)
L2Protocol	Octet	P	C	Mandatory if L3Protocol is NULL (4). Default (255) – ISO 7776 (Note)
L1Protocol	Octet	P	C	Mandatory if L2Protocol is NULL (8). Absent if L2Protocol is absent. (Note)
NOTE – A detailed description of the possible use and values for this parameter can be found in Parts 3 to 6.				

7.6.43 UAttributeName

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

Type: 63.

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

7.6.44 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.

Field	Field type	Direction	Required	Comment
Direction	Octet	P	M	listen (1) call (2) both (3)
NOTE – If absent, the value assumed by the NAF is the value of the CDirection parameter.				

7.6.45 UserToUserInfo

Description: This parameter is used to pass user-to-user information to/from the PUF.

Type: 65.

Field	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.

7.6.46 AttributeSet Parameters

AttributeSet parameters depend on the type of parameters to provide with the ACreateNCO request. Tables 10 and 11 show the content of such parameters.

Table 10 – Signalling Attribute Set (CAttributeSet) parameters

Parameter	Required	Comment
ChannelIdentification	O*	See 7.6.12
HLC	O	See 7.6.28
LLC	O	See 7.6.31
BearerCap	O	See 7.6.3

Table 11 – External Equipment related parameters (within the UAttributeSet)

Parameter	Required	Comment
ExtEquipName	O	Name of external equipment to be used. If provided connection shall be established to identified external equipment and User Plane messages shall not be provided. See 7.6.25.

Tables providing further detailed description of the User Plane Attribute Set (UAttributeSet) Parameters available to be used with the ACreateNCO request for the various possible user protocols can be found in Parts 3 to 6.

7.6.47 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 12 shows the content of this parameter.

Table 12 – Administration attribute set parameters

Parameter	Required	Comment
NCOType	O	See 7.6.34
CDirection	O	See 7.6.11
CAttributeName	O	See 7.6.8
CAttribute parameter	O	See Table 10
UDirection	O	See 7.6.44
UAttributeName	O	See the relevant ISDN-PCI User Plane protocol specification
UAttribute parameter	O	See the relevant ISDN-PCI User Plane protocol specification
CAddress parameters	O	See Table 13
UAddress parameters	O	See the relevant ISDN-PCI User Plane protocol specification
GroupID	O	Providing at the NCO creation time. See 7.6.27.
SelectorID	O	See 7.6.39
ChargingInfo	O	See 7.6.13
DateTime	O	Date and Time of the NCO creation. See 7.6.20.
CauseToPUF	O	See 7.6.10

7.6.48 Address set parameter

Table 13 shows the structures of the Address.

Table 13 – Signalling address set (CAddressSet) parameters

Parameter	Required	Comment
CalledNumber	O	See 7.6.4 for parameter definition
CalledSubaddress	O	See 7.6.5 for parameter definition
CallingNumber	O	See 7.6.6 for parameter definition
CallingSubaddress	O	See 7.6.7 for parameter definition

Tables providing further detailed description of the User Plane Address Set (UAddressSet). Parameters available to the PUF for the various possible user protocols can be found in Parts 3 to 6.

7.7 Selection criteria

7.7.1 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. Table 14 summarizes the matching operation.

Table 14 – 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	Not provided	(no operation)	Match

The NAF shall broadcast an incoming call to all PUFs, which have indicated compatibility within a 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 coordinated NCOs have a higher priority than PUF coordinated 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-coordinated NCOs.

If CAlertReq message is sent by a PUF, only the first shall be sent to the Network. All others are ignored. When a CDisconnectReq message is sent by a PUF, it does 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.

The SelectorID parameter impacts on the incoming call broadcasting operation when more than one NCO per PUF is selected.

7.7.1.1 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 Table 15, only the NCO2 shall be chosen because the NCO1 is waiting for a subaddress information element different than that provided in the incoming call and because all the expected information element included in the NCO2 matches with the incoming call information elements.

Table 15 – Matching NCO on an incoming call

Field	Incoming call	NCO1	NCO2
Called addr.	123456789	Not provided	123456789
Called sub-addr.	1002	1001	Not provided
Bearer cap.	Speech	Not provided	Speech
LLC	No outband negotiation	Not provided	Not provided
HLC	Telephony	Telephony	Telephony

User Plane protocol specific Information element to be considered during the NCO selection process are specified for the various possible user protocols in Parts 3 to 6.

7.7.2 Action if no NCO available

7.7.2.1 Control plane incoming call

A disconnection cause #88 "incompatible destination" is issued by the NAF.

7.7.2.2 User Plane incoming call

The disconnection procedure for the various possible user protocols are specified in Parts 3 to 6.

7.8 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.

7.8.1 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.

7.8.2 Control plane

When Mandatory parameters are missing, or a content error occurs in a mandatory parameter, or a parameter is unrecognized, the NAF indicates the error to the PUF as given in 7.8.2.1 to 7.8.2.3.

7.8.2.1 Invalid state for message

CStatusInd, no change of state for connection.

7.8.2.2 Mandatory parameters

In case of mandatory parameters missing, mandatory parameters content error or unrecognized parameter, the NAF indicates this error to the PUF as follows:

- for CConnectReq the PUF is sent a CDisconnectInd;
- for CDisconnectReq the PUF is sent a CDisconnectCnf;
- for any other message the PUF is sent a CStatusInd, no operation is performed, and no change of state occurs.

7.8.2.3 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.

7.8.3 Errors in facility requests

Errors related to facility requests depend on the facility being requested. In the case of 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 Table 9.

When a PUF uses facilities in the transparent form it is up to the PUF to understand how errors will be reported and what processing may have occurred within the network.

7.8.4 User Plane

Errors are dealt with according to procedures defined in Parts 3 to 6 for the various possible User Plane protocols.

7.8.5 Function return codes

Table 16 defines function return codes.

Table 16 – Function return codes

Return code		Meaning
Success	0	Function completed successfully
QueryEntityNotAvailable	128	The Query entity is not available or an error occurred 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 fixed condition.
NAFBusy	132	NAF is unable, currently, to process this request (lack of resource or other reason). The function may work correctly if reused 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 recognize 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 in an implementation, this value is returned.
InvalidHandlesBuffer	146	The PCIHandles buffer address is invalid
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)

7.8.6 Administration plane return code

The following values (see Table 17) are returned in the CompletionStatus parameter. The ErrorSpecific information column indicates what, if any, information shall be in the ErrorSpecific field.

Table 17 – 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 resource (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 recognized or is not supported by a message	Parameter Type
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 recognized 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

7.8.7 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 18 – Control plane causes

Value	Q.931 [1] meaning	ISDN-PCI meaning	Generated by	NAF provided diagnostics
1	Unallocated (unassigned) number		ISDN	N/A
2	No Route to Specified Transit Network		ISDN	N/A
3	No route to destination		ISDN	N/A
6	Channel Unacceptable		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
41	Temporary Failure		ISDN	N/A
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
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 authorized		ISDN	N/A
58	Bearer Capability not presently available		ISDN	N/A

Table 18 – Control plane causes *(concluded)*

Value	Q.931 [1] meaning	ISDN-PCI meaning	Generated by	NAF provided diagnostics
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
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.

For some ISDN, other values may be provided in the NAF to PUF direction.

Table 19 – Content of the CauseToNAF parameter

Value	Meaning
16	Normal call clearing
21	Call rejected
31	Normal unspecified
88	Incompatible destination

7.8.8 User Plane causes

Values returned as completion causes for messages of the User Plane can be found in Parts 3 to 6.

8 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 Exchange Method for a real environment is contained in Parts 7 to 9 for various Operating systems. They include the binary representation, some code samples in Programming Language C and all information attached to a particular Operating System.

Since the NAF side implementation of the exchange functions depends on the underlying operating system, the PUF code calling this function is operating system dependent as well. To be source code portable between different operating systems, the PUF should encapsulate the code calling the NAF by a functional interface, which resembles the generic exchange functions described in this clause.

The exchange functions pass and return parameter values. These values are based on the generic types shown in Table 20.

Table 20 – 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

Dependent of the operating system, the parameters are passed either by value or by reference. The way parameters are passed is defined in the relevant specification.

General conventions for this generic presentation:

- 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.

8.1 Registration phase

8.1.1 Overview

Before a PUF and a NAF can interchange information the PUF has to 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 documentation.

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 10.

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 has to give 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 will become 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 clause 6). 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 might be a name, a file-path or a function address.

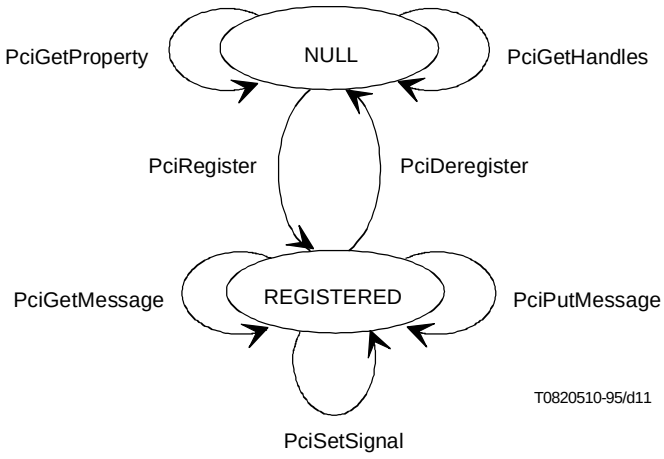


Figure 10 – ISDN-PCI exchange function calls order including the optional registration phase functions

NOTE – In many NAF environments, a PUF may use the optional functions PciGetHandles and PciGetProperty as described above to select the NAF which suits best to it's needs. For more details on the **PciGetProperty** function refer to 8.1.3.

1) The use of this function is optional for the PUF, but it's implementation (provision) is mandatory for the NAF.

8.1.2 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 its 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.

8.1.3 PciGetProperty

This function allows a PUF to obtain the NAF-Property. The PUF has to 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 (Table 21).
ActualSize	PCI_INTEGER	Return Value	Actual size of property returned

The parameters of NAF-Property are shown in Table 21.

Table 21 – TLV coded NAF-Property parameter

Parameter	Provided	TLV Coding (Note)			Comment and value
		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 a)	M	4	1..4	Octet	Give the supported layer 3 protocols. May be a NAF selection criteria. For defined value, see 7.6.42.
UserProtocolL2 a)	M	5	1..2	Octet	Give the supported layer 2 protocols. May be a NAF selection criteria. For defined value, see 7.6.42.
B-Channels	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 a)	O	9	1	Octet	Additional administration plane functions supported, identified by class. Valid value: 2..4.
CplaneClass a)	O	10	1	Octet	Additional control plane functions supported, identified by class: Valid values are in range 2..7.
SuppService a)	O	11	1..16	Octet	Specification of supplementary services. For future study.
ExtEquipName a)	O	12	2..17	Octet	In order, type and name of external equipment. See 7.6.25.
AdditionalUserProtocol a)	O	13	1..16	Octet	Additional User Plane protocols. For further extension.
PCIVersion a)	O	15	1	Octet	PCI version supported

a) Parameter may be repeated.

NOTE – There may be other TypeID values defined in other specifications of the ISDN-PCI series

8.1.4 PciRegister

This function allows a PUF to be associated to an NAF.

As a calling parameter, the PUF provides the PCI-Handle of the NAF it wants to register with. In addition 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 have to be passed between the two entities to ensure proper cooperation. The PCIRegisterInfo structure is shown in Table 22.

The PCIOpSysInfo structure contains operating system dependent information to be exchanged between PUF and NAF.

Table 22 – 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 will neither deliver messages bigger in size nor does it guarantee to accept bigger ones from PUF.
NOTE – The PUFVersion number which equals the major revision number is defined in Part 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 functions of the exchange method 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 (Table 22)
PCIOpSysInfo	PCIOpSysInfo structure	Call Value	Contains Operating system dependent information. For details, refer to Parts 7 to 9
ExID	PCI_EXID	Return Value	Exchange-ID

[1] For more (operating system specific) error codes, refer to Parts 7 to 9.

8.2 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 is mandatory prior to PUF termination.

When the PUF disassociates using this function, the NAF will free any resources allocated for this PUF, such as clearing already existing connections.

8.2.1 PciDeregister

This function disassociates a PUF from an NAF. The association between the PUF and NAF is identified by the ExID.

Function Name: PciDeregister

Function Return Value: Errorcode (PCI_INTEGER)

Success

InvalidExID
NAFnotAvailable
NAFBusy

Parameter:

Name	Generic type	Call or Return Value	Comment
ExID	PCI_EXID	Call Value	Exchange-ID 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 is possible using this ExID.

8.3 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 (PCI-MPB) 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.

8.3.1 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-MPB 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 PCI-MPB contains the Message Identifier and details concerning the usage of the message and data buffers.

8.3.2 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 if 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 will 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.

8.3.3 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 will be indicated in a special way.

To be able to receive a message the PUF provides the NAF with a correctly set-up PCI-MPB, which shall contain the addresses of a message and a data buffer respectively. The size of the message and data buffers have to 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 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 will be indicated by the value zero for the length used. The absence of a message is indicated by the NOMESSAGE (0) type in the MessageID field of the PCI-MPB.

8.3.4 Receiving messages using signal method

To receive a message using this method, first, the PUF has to 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 an 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 will do so until the **PciSetSignal** function is used to remove the notification mechanism.

To receive the message from the NAF, the PUF has to 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. These restrictions are depending on the operating system.

8.3.5 PCI Message Parameter Block (PCI-MPB)

Table 23 shows the structure of the PCI Message Parameter Block (PCI-MPB).

Table 23 – Structure of the PCI Message Parameter Block (PCI-MPB)

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 11 shows how messages are sent or received.

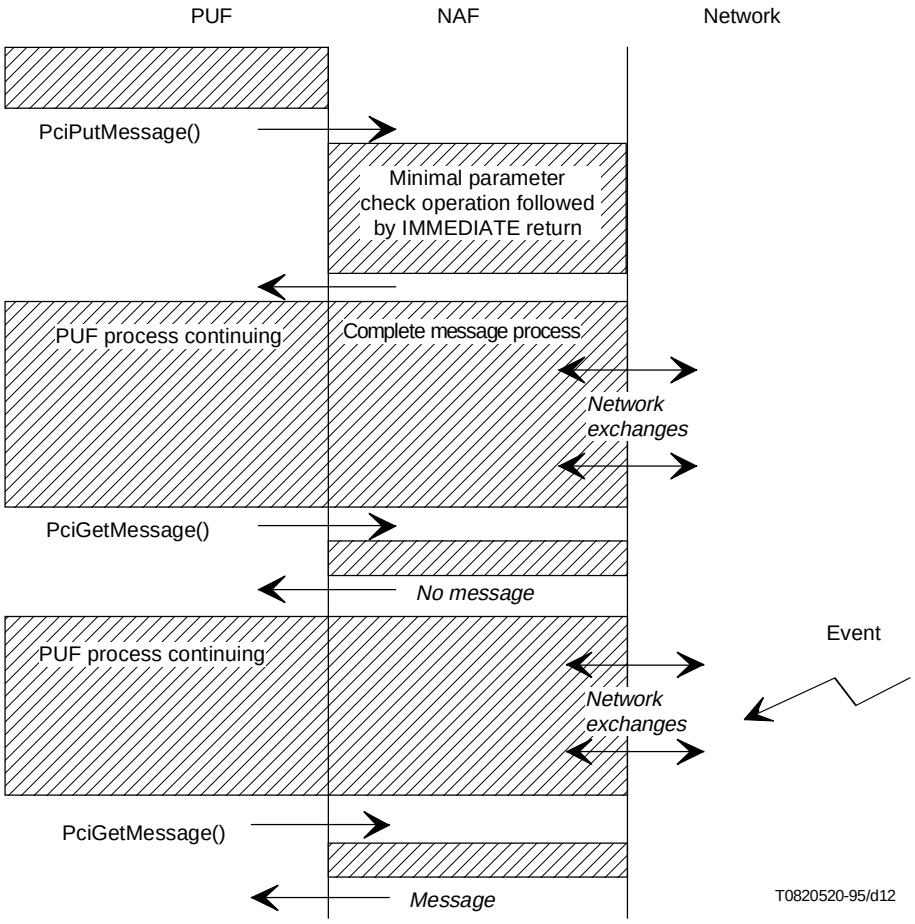


Figure 11 – Process to send or to receive messages

8.3.6 PciPutMessage

This function allows a PUF to transmit a message to an 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 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 PCI-MPB.

The PUF indicates the size of the buffer(s) in the ActualUsedSize field of the PCI-MPB 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 PCI-MPB structure is always required. To indicate absence of a buffer, the PUF may specify no buffer address instead by supplying a NULL (0) value.

8.3.7 PciGetMessage

This function allows a PUF to get a message from an 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 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 will return an error.

The PUF indicates the maximum size of the buffer(s) in the MaximumSize fields of the PCI-MPB for the respective buffers, i.e. MessageMaximumSize for the message buffer and DataMaximumSize for the data buffer.

On return, the NAF will indicate the size of the buffer(s) in the ActualUsedSize field of the PCI-MPB for the respective buffers used.

To indicate no message, the NAF fills the MessageID field in the PCI-MPB with NOMESSAGE (0).

8.3.8 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 will 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

InvalidExID
NAFnotAvailable
NAFBusy
InvalidSignalNumber

Parameters:

Name	Generic type	Call or Return Value	Comment
ExID	PCI_EXID	Call Value	Exchange-ID 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.

Any new **PciSetSignal** call overwrites the previous one.

The signal mechanism can be stopped by supplying a NULL (0) value instead of Signal and SignalProcedure values during call.

9 Security

This clause addresses communication security using the ISDN-PCI.

9.1 General aspects of security in ISDN

The digital nature of ISDN facilitates adding security, but ISDN has been developed without support for security features in the lower layers. The deployment of ISDN in the public network is well under way and this constrains how security features may be added.

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 adapt it to ISDN and incorporate it in standards.

9.2 Security in the ISDN-PCI

Although there are no lower layer ISDN security standards available, the ISDN-PCI offers access to security functionalities which may be available in the NAF. This access offers an initial approach to use security on the 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 subaddress information. The ISDN number and subaddress information are provided by the network, and therefore, may be used to identify the calling user. This supplementary service provides the PUF with a method to identify the other party.

– *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 7.2.9 and 7.2.10 for information on the use of these administration plane messages.

9.3 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 9.2.

Although the standards do not exist, the impact of introducing security in the ISDN-PCI can be estimated. Three approaches to introducing security can be seen:

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 on one of the lower layers a secure protocol is operating, 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 to implement such extensions:

   – adding a message;

   – extending the attribute sets to contain the security information;

   – NCOs contain the security information.

3) *Definition of a new secure protocol stack for ISDN*

   If new secure ISDN protocols are established, the ISDN-PCI must 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.

Superseded by a more recent version

Annex A

Telephony

This annex presents different types of external equipment handled in this part.

A.1 Type 1 external equipment

This external equipment is the simplest form of telephony equipment. It does not contain hook control 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 of the external equipment (external equipment connected or not 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.

If the external equipment is in use, a CConnectReq that attempts to use the external equipment should be rejected with a 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).

A.2 Type 2 external equipment

This external equipment contains hook control 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 transitions in the PUF for incoming and outgoing calls.

A control plane message is defined to indicate to the PUF the availability of the external equipment (external equipment connected or not 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 is the responsibility of the PUF to ensure that the hook control is in the desired 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 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 47 (resource unavailable).

A.3 Type 3 external equipment

This contains hook control but not 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 means of two control plane messages. Therefore, this external equipment can cause state transitions in the PUF for incoming and outgoing calls.

A control plane message is defined to indicate to the PUF the availability of the external equipment (external equipment connected or not 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 then determines whether or not to make the external equipment available for use. If it does not, an attempt to connect shall be denied with CDisconnectInd with cause 47 (resource unavailable).

A.4 Type 4 external equipment

This external equipment contains a dialling mechanism and or not a hook control. This external equipment does not manage ISDN signalling. This kind of external equipment supports 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 control, it can provide some information to the PUF about the state of the handset by means of two control plane messages.

A control plane message is defined to indicate to the PUF the availability of the external equipment (external equipment connected or not to the NAF).

All dialling actions and handset actions (if available) realized on this external equipment can cause state transition in the PUF for incoming and 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 then determines whether or not to make the external equipment available for use. If it does not, an attempt to connect shall be denied with CDisconnectInd with cause 47 (resource unavailable).

A.5 Type 5 external equipment

This external equipment contains a dialling mechanism and or not a hook control. This external equipment is connected to the ISDN network; therefore, it is able to manage ISDN signalling in the case where the host (e.g. personal computer) is off. Type 5 external equipment allows placing outgoing calls from it and answering incoming calls.

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 hook control, it can provide some information to the PUF about the state of the handset by means of two control plane messages.

A control plane message is defined to indicate to the PUF the availability of the external equipment (external equipment connected or not to the NAF).

All Type 5 dialling actions and handset operations (if available) can cause state transitions in the PUF for incoming and 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 then determines whether or not to make the external equipment available for use. If it does not, an attempt to connect shall be denied with CDisconnectInd with cause 47 (resource unavailable).

Superseded by a more recent version

Annex B

Mapping between ISDN-PCI messages and parameters and the ISDN

This annex provides the mapping between protocols used and the ISDN-PCI messages.

B.1 Control plane messages

Table B.1 – Control plane message to Q.931 mapping

PCI message	Q.931 [1] 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	
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	

Table B.1 – Control plane message to Q.931 mapping *(concluded)*

PCI message	Q.931 [1] message	Direction	Notes
CFacilityReq	FACILITY	user-to-network	
CFacilityInd	FACILITY	network-to-user	
CAddInfoReq	INFORMATION	user-to-network	
CAddInfoInd	INFORMATION	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 the Recommendation Q.931 [1] information elements is defined in Table B.2.

Table B.2 – Control plane parameters

Control plane parameter	Q.931 [1] 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
NumberComplete	Sending complete
ProgressIndicator	Progress Indicator
Signal	Signal
UserToUserInfo	User-user

Annex C

Static attribute content

This annex contains a complete description of the static attributes that an NAF shall provide. Rules to establish these attributes are related to the protocol requirement.

C.1 Control plane static attribute sets

The attribute sets described below use the following conventions:

Name: To be used with the ANCOCreateReq message

BC: Content of the BearerCap parameter, in hexadecimal octets

LLC: Content of the LLC parameter, in hexadecimal octets – decimal if in parenthesis

HLC: Content of the HLC parameter – decimal in parenthesis.

C.1.1 Generic circuit bearer service

C.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

C.1.1.2 Unrestricted digital information

Name: "UNRESTRICTED"

BearerCap: 88 90

LLC: Not used

HLC: Not used

C.1.1.3 Restricted digital information

Name: "UNRESTRICTED/56"

BearerCap: 88 90 01 8F

LLC: Not used

HLC: Not used

C.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

C.1.2 Packet mode bearer service

Name: "D_CHANNEL_HDL"

BearerCap: 88 C0 C6 E6

LLC: Not used

HLC: Not used

C.1.3 Teleservices

Name: "TELEPHONY_A-LAW"

BearerCap: 80 90 A3

LLC: Not used

HLC: Standard = 0
Identification = 1

Name: "TELEPHONY_μ-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: octets 4 and 5.
Octet 6 (layer 2) = 0D (13)
Octet 7 (layer 3) = 07

HLC: Standard = 0
Identification = 21 (33)

Appendix I

NAF development guidelines

The main body of this part 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 appendix. It gives guidelines for the development of the NAF in accordance with the main body of this part.

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.

Consider this appendix from the following viewpoints:

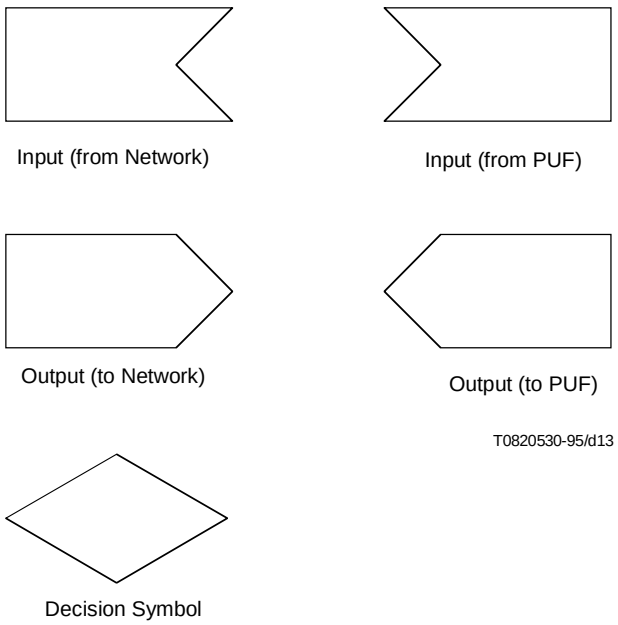
- This appendix gives additional points. The NAF has to be implemented using this part. It should implement the ISDN-PCI in such a way that the functionality described is provided.
- The main body of this part should be given priority if there is anything not clear in this appendix or the interpretation between the main body of this part and this appendix is different.
- This appendix 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 part.

I.1 NAF-SDL diagrams

The following SDL diagrams show, as example, the internal states of the call control section of the NAF. They are provided for clarification only. Not all the possible cases are shown in these diagrams, for simplification.

The primitives shown in upper case are those defined in Recommendation Q.931 [1].

The following symbols are used within this description. A full description of the symbols and their meaning is given in Recommendation Z.100.



State Symbol

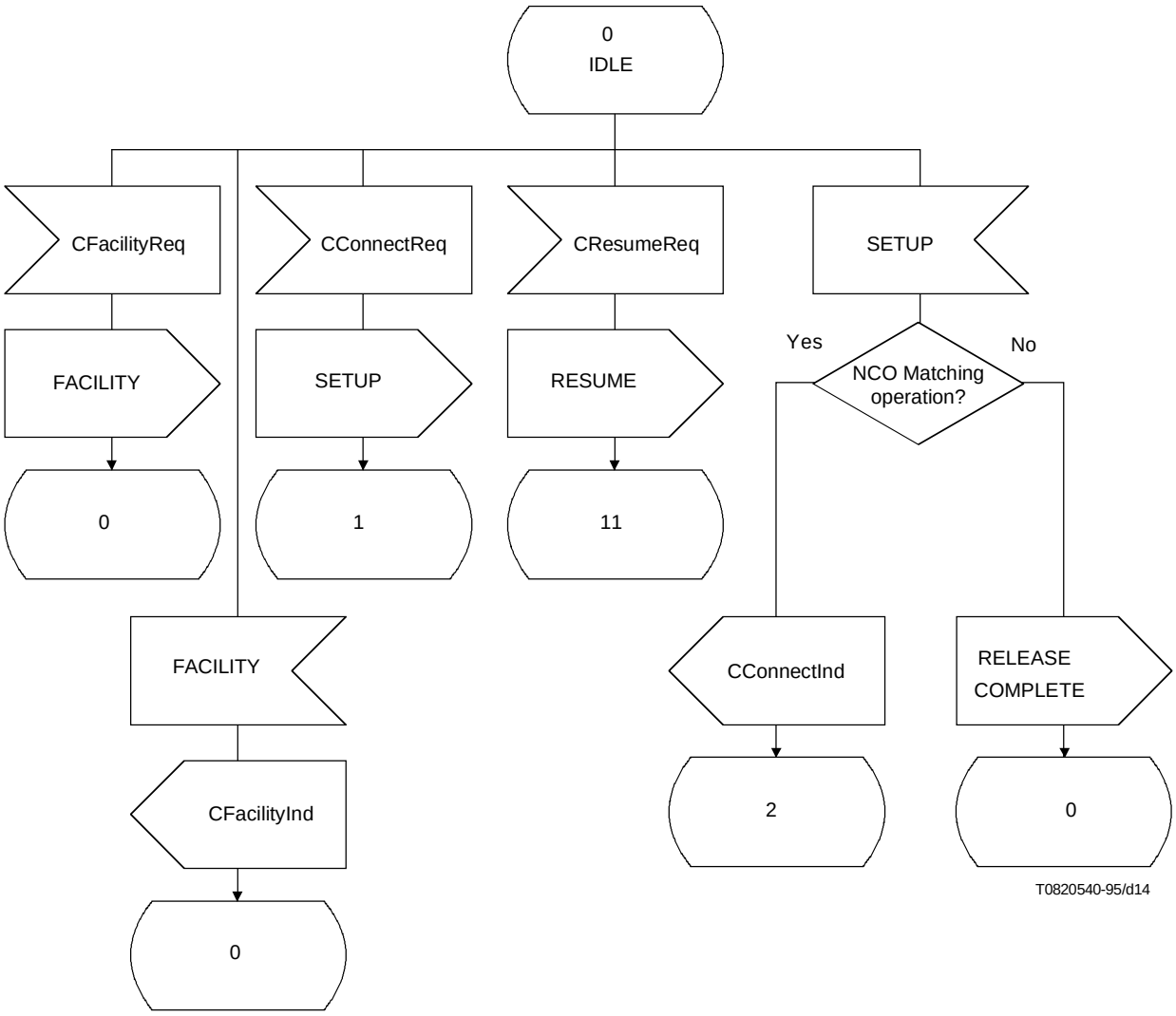


Figure I.1 – IDLE

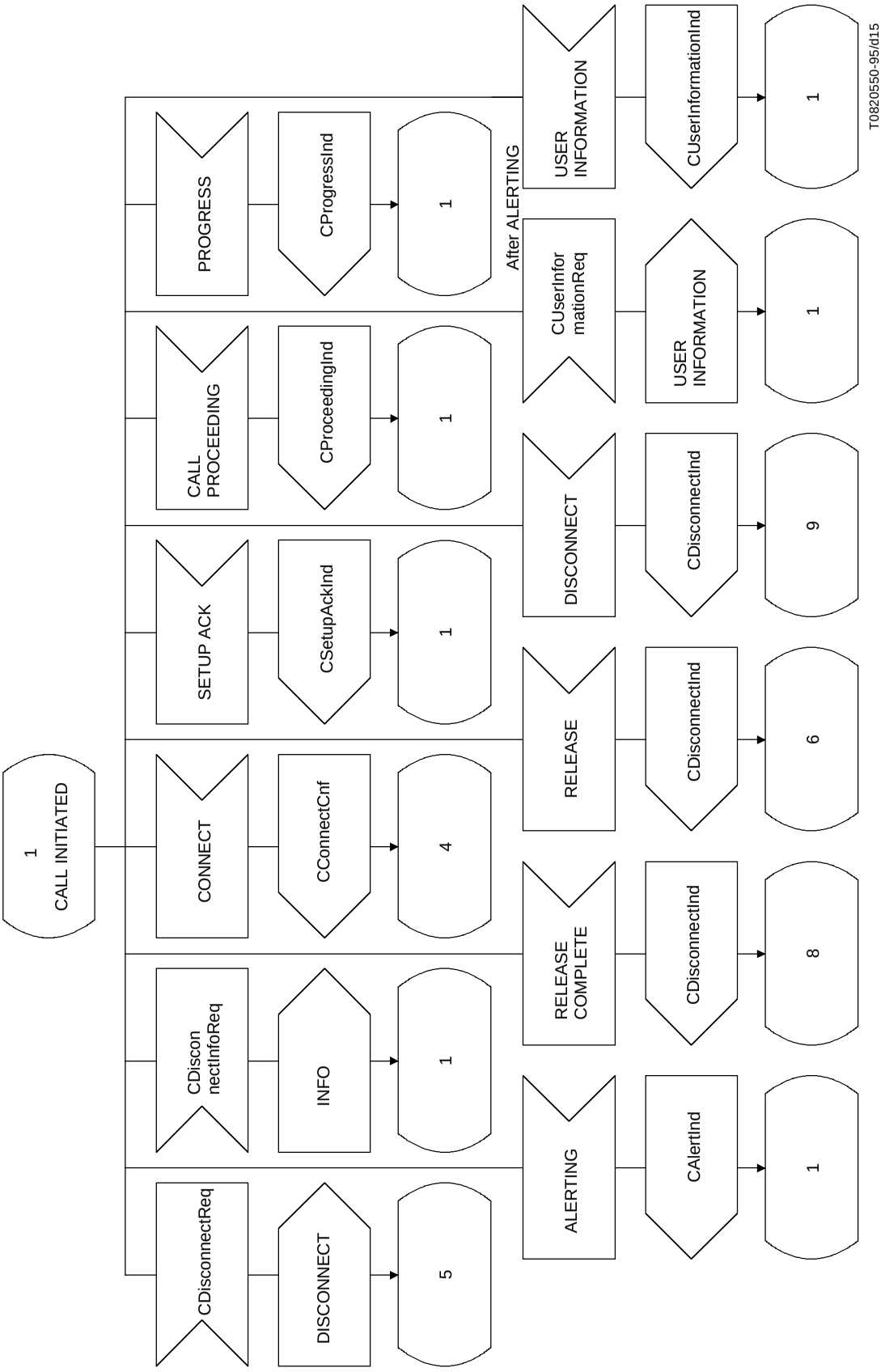


Figure I.2 – CALL INITIATED

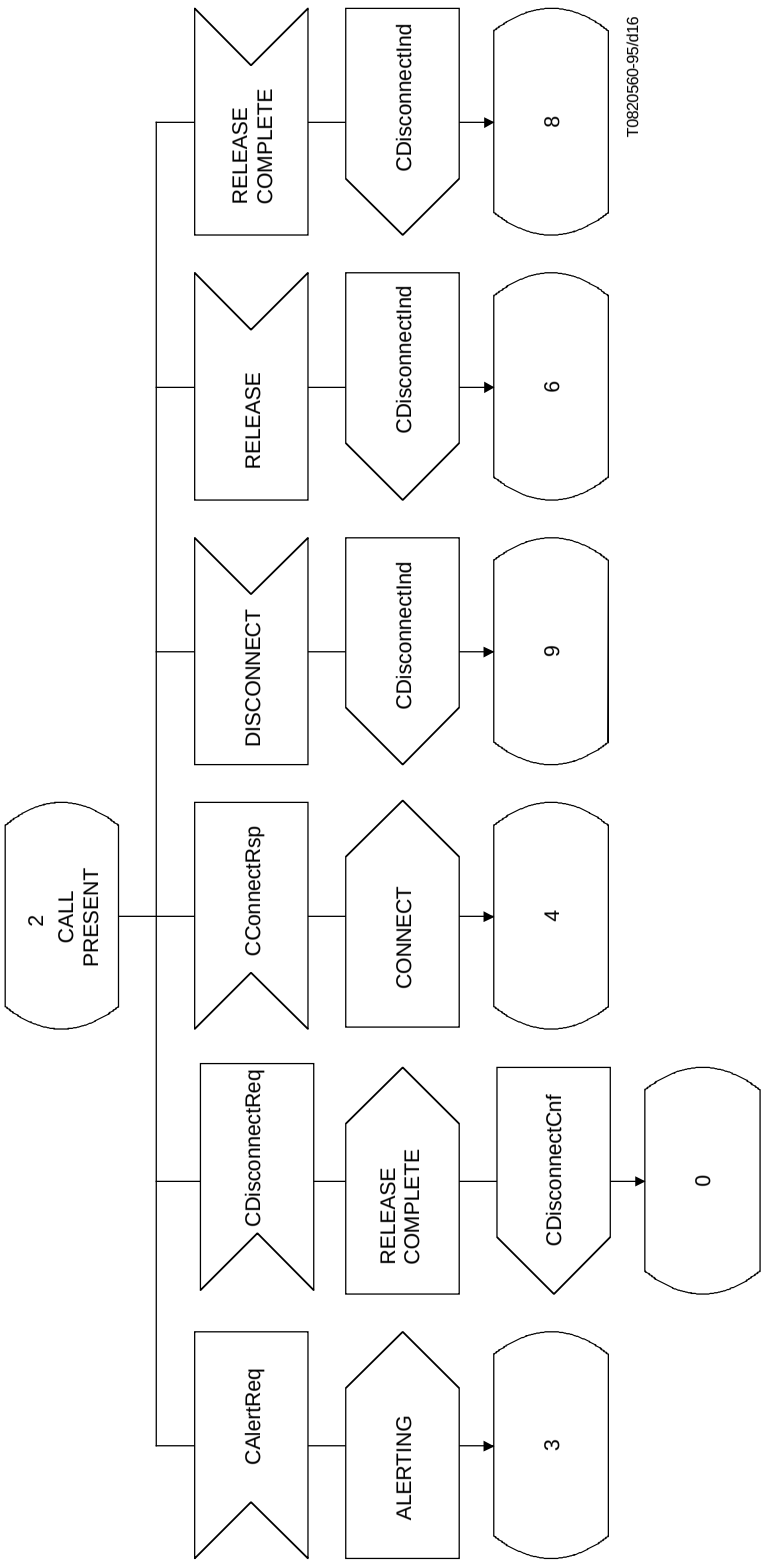


Figure I.3 – CALL PRESENT

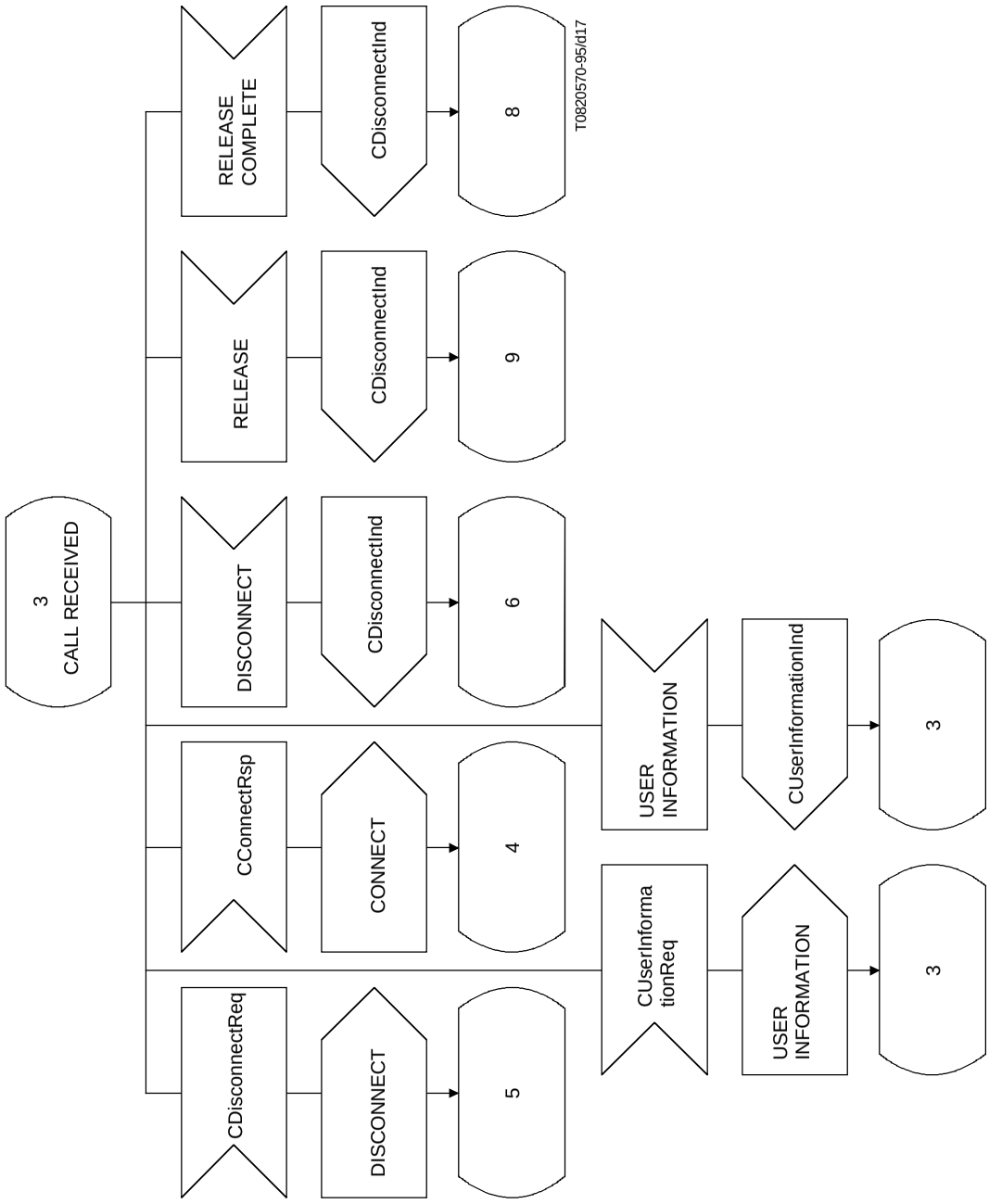


Figure I.4 – CALL RECEIVED

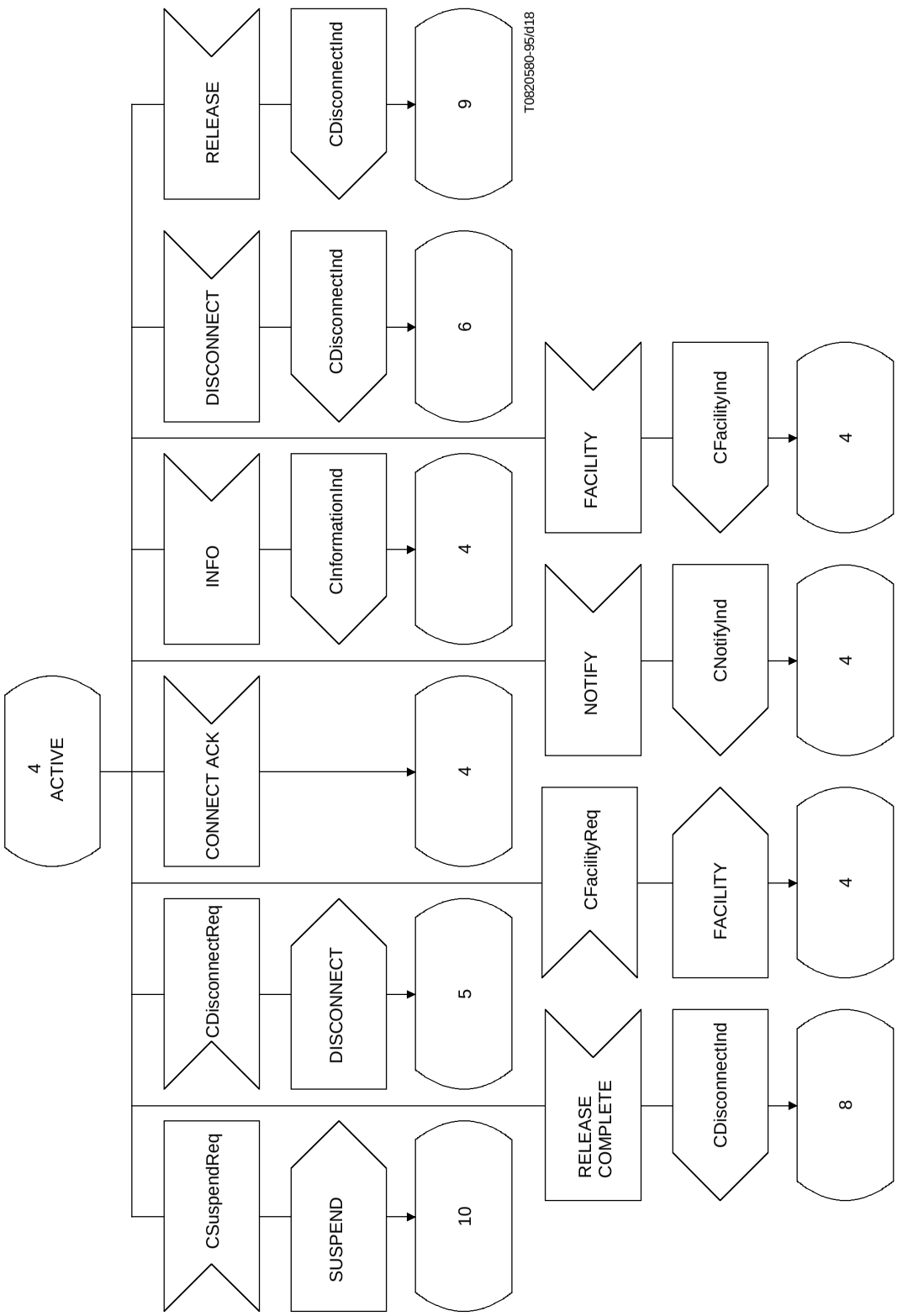


Figure I.5 – ACTIVE connection

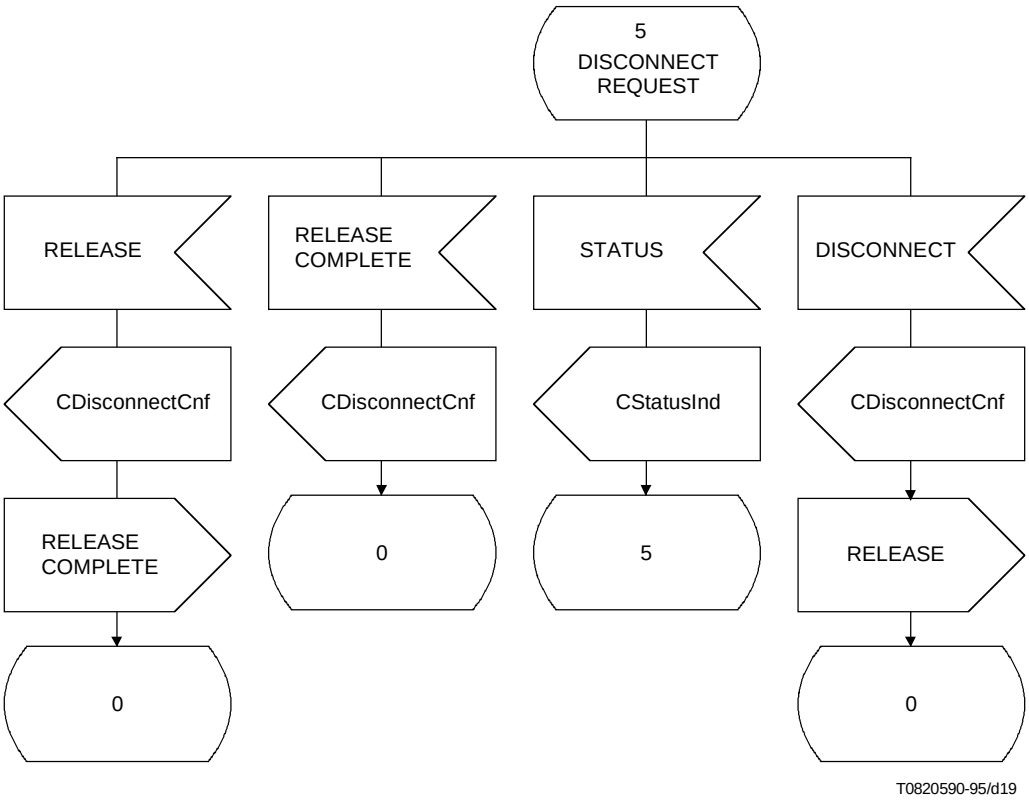


Figure I.6 – DISCONNECT request

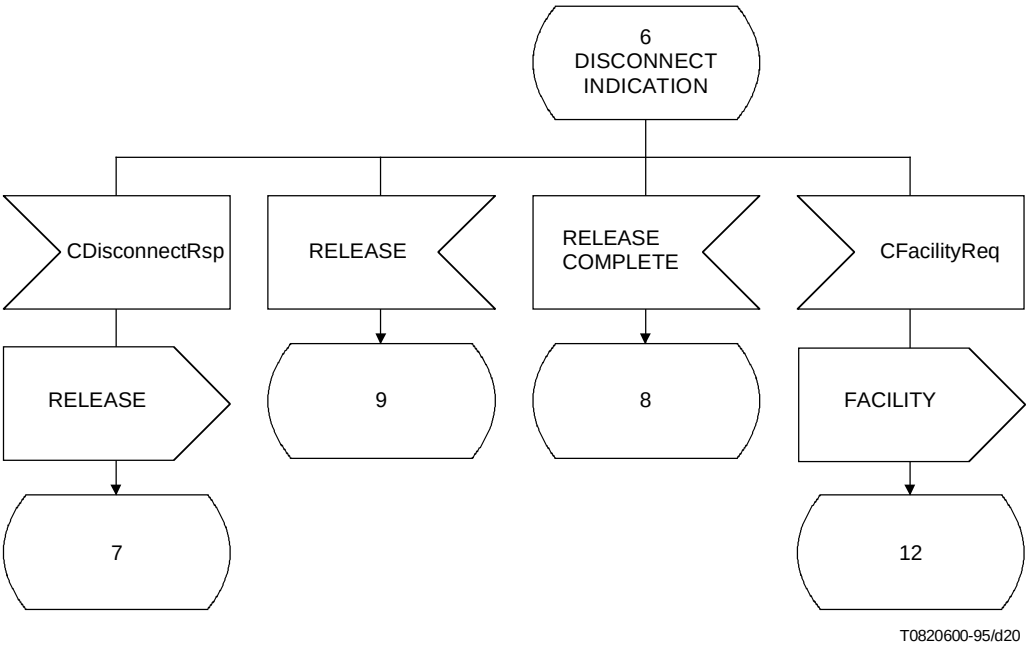


Figure I.7 – DISCONNECT indication

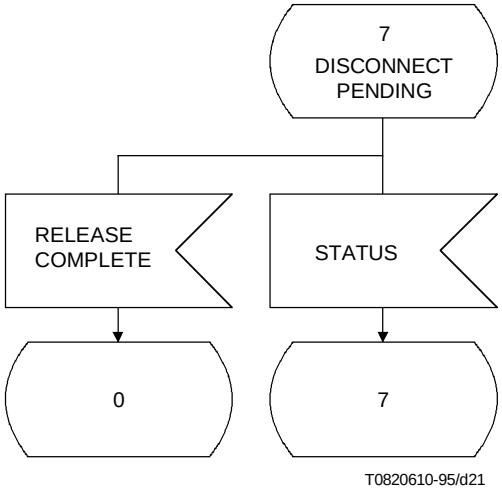


Figure I.8 – DISCONNECT pending

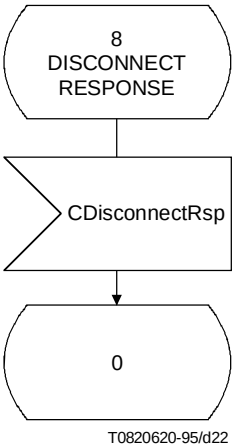


Figure I.9 – DISCONNECT response

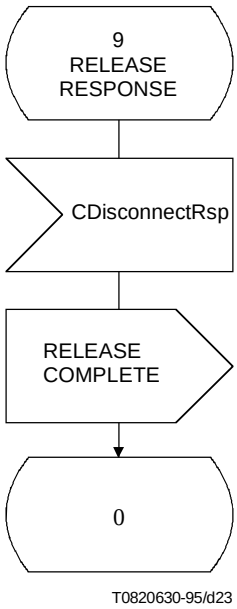


Figure I.10 – RELEASE response

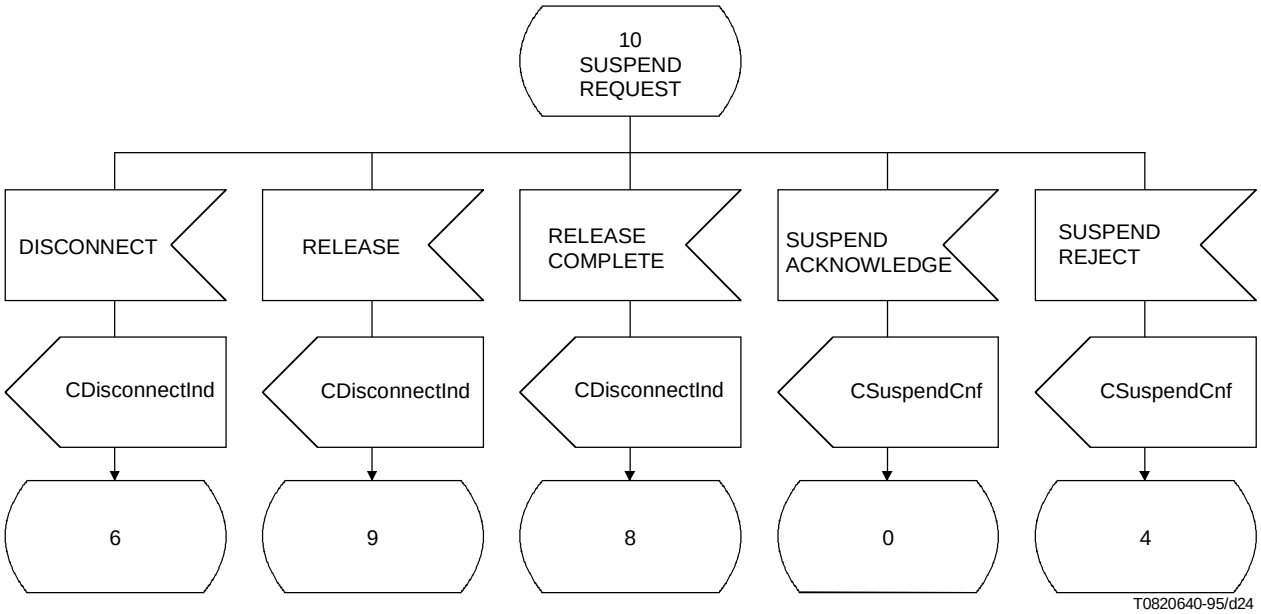


Figure I.11 – SUSPEND request

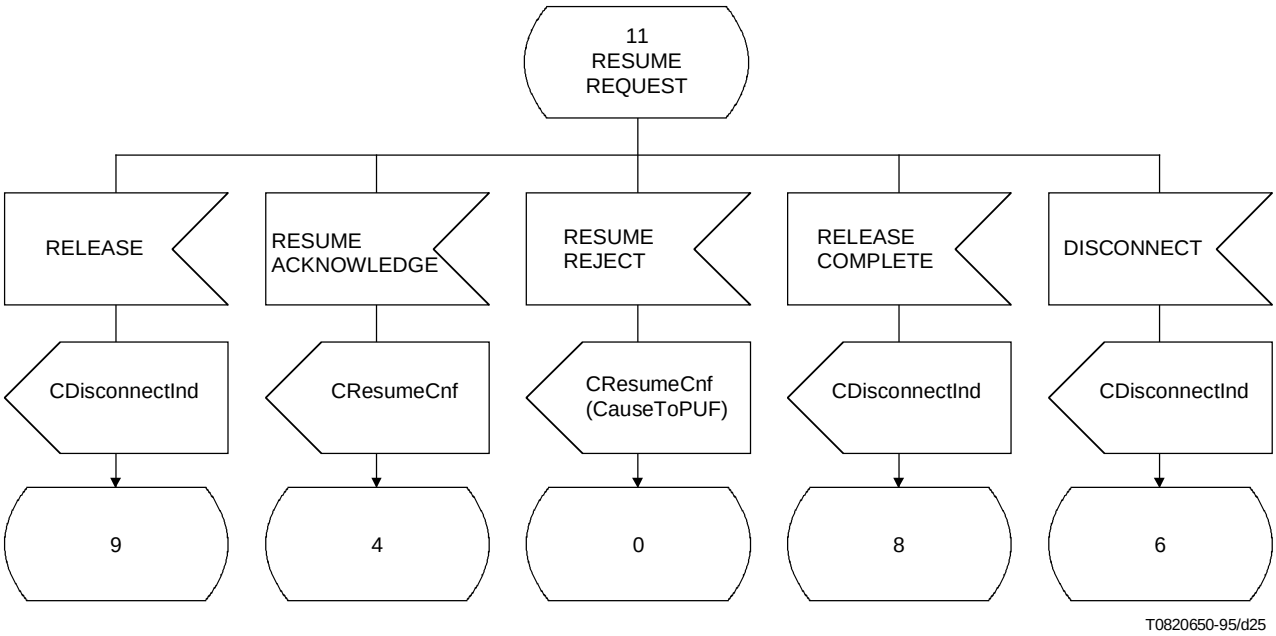


Figure I.12 – RESUME request

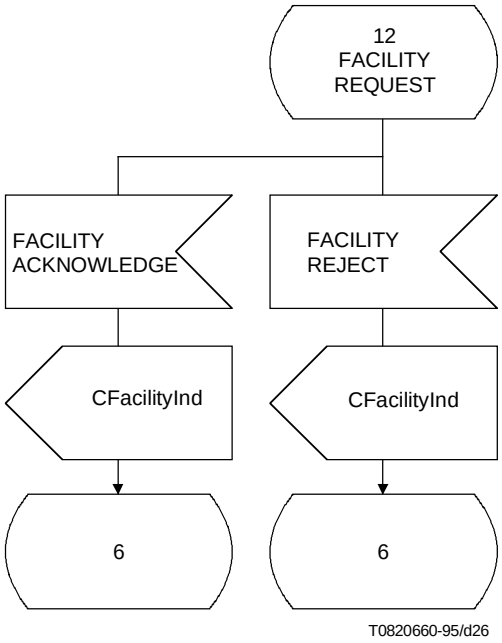


Figure I.13 – FACILITY request

I.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 on if they are available to the NAF.

I.3 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.

I.4 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 an NAF specific manner. For example, an NAF may choose to implement an error log in the form of a file. This file is where it records detailed information concerning a particular error. This provides the information required to debug a PUF which is under development.

The following subclauses provide guidance as to the conditions under which 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 of providing it to the PUF in the relevant message.

I.4.1 Function return codes

The description of the conditions under which these should be issued is described in 7.8.5.

I.4.2 Administration plane

The description of the conditions under which these should be issued is described in 7.8.6. For the ACreateNCOReq, the number of possible parameter combinations makes checking complex. It should be approached in the order shown in Table I.1.

Table I.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

Table I.1 – Checking of ACreateNCOReq message *(concluded)*

Parameter	Test	Action
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
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

I.4.3 Control plane

The errors returned in the Cause parameter match those in Recommendation Q.931 [1] 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 I.2 – Control plane Cause parameter matching

Value	Q.931 [1] 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

Table I.2 – Control plane Cause parameter matching *(continued)*

Value	Q.931 [1] meaning	PCI meaning	Generated by	When received from ISDN processed by
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)
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

Table I.2 – Control plane Cause parameter matching *(concluded)*

Value	Q.931 [1] meaning	PCI meaning	Generated by	When received from ISDN processed by
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)
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 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 I.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 I.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 X.25 and CalledNumber is 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
Unrecognized Parameters	Present	CDisconnectInd Cause parameter value = 99 Diagnostics = Parameter Type of unrecognized parameter
	Not present	Continue
Optional Parameter Content Error	Present	CStatusInd Cause parameter value = 100 Diagnostics = Parameter Type of parameter in error Continue (ignore parameter)
	Not present	Process Message

I.5 NAF configuration

The following subclause contains information concerning NAF configuration. That subclause is provided to assist NAF developers and is not intended to be a comprehensive list of configurable items.

I.5.1 Global configuration

Table I.4 – Global configuration

Parameter	Suggested default	Comment
Number of PUFs supported	8	

I.5.2 System configuration parameters

Table I.5 – 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

I.5.3 Control plane configuration

Table I.6 – 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; – N202. 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	

I.6 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 part.

In the PUF-to-NAF direction, the message and the associated data, if any, are provided together, in one step. If one of the messages or the data buffers is too small to contain, respectively, a complete message or the data information, the NAF shall return an error and the message shall 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 has to refer to the User Plane protocol initialization to get the correct value of the longest data packet.

If a NAF needs new internal buffers, it is in charge of this action. This can be achieved via a configuration operation, provided by the NAF manufacturer, which is outside the scope of this part. The NAF manufacturer may describe how the operation can be realized and which consequences are expected.

Appendix II

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 receives 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 */
```

Appendix III

List of parameters

Table III.1 contains the complete list of parameters defined in ISDN-PCI. The first column gives the parameter code (type). The second column provides the name of the parameter. The last column indicates in which plane the parameter is used. Some parameters may appear in more than one plane.

Table III.1 – List of ISDN-PCI parameters

Parameter identificator	Parameter name	Use in the plane
1	Algorithm	Administration
2	Bcug	User and Administration
3	BearerCap	Control and Administration
4	Bit_DQM	Control and Administration
5	CalledDTEAddress	User and Administration
6	CalledDTEAddressExt	User and Administration
7	CalledNumber	Control and Administration
8	CalledSubaddress	Control and Administration
9	CallingDTEAddress	User and Administration
10	CallingDTEAddressExt	User and Administration
11	CallingNumber	Control and Administration
12	CallingSubaddress	Control and Administration
13	CAttributeName	Control and Administration
14	CauseToNAF	Control
15	CauseToPUF	Control and Administration
16	CDirection	Administration
17	ChannelIdentification	Control
18	ChargingInfo	Administration
19	CompletionStatus	Administration, Control and User
20	CongestionLevel	Control
21	ConnectedNumber	Control
22	ConnectedSubaddress	Control
23	DateTime	Control and Administration
24	Display	Control
25	ExtEquipAvailability	Control
26	ExtEquipBlockDialling	Control
27	ExtEquipKeyPressed	Control
28	ExtEquipName	Control and Administration
29	ExpeditedData	Control
30	Facility	Control
31	FacilityData	User
32	FastSelect	User and Administration
33	GroupID	Administration

Table III.1 – List of ISDN-PCI parameters *(continued)*

Parameter identificator	Parameter name	Use in the plane
34	HLC	Control and Administration
35	IdleFlag	Administration
36	Key	Administration
37	Keypad	Control
38	L2ConnectionMode	Administration
39	L2FrameSize	Administration
40	L2WindowSize	Administration
41	L2XID	Administration
42	L3ConnectionMode	Administration
43	L3IncomingCount	Administration
44	L3OutgoingVCCount	Administration
45	L3TwoWayCount	Administration
46	LLC	Control and Administration
47	ManufacturerCode	Administration
48	MoreData	Control
49	NCOID	Administration, Control and User
50	NCOType	Administration
51	NotificationIndicator	Control
52	PacketSize	User and Administration
53	ProgressIndicator	Control
54	QOSParameters	User and Administration
55	ReadyFlag	User
56	RequestID	Administration
57	ReceiptConfirm	User
58	RespondingDTEAddress	User
59	RespondingDTEAddressExt	User
60	SelectorID	Administration
61	TEI	User and Administration
62	UProtocol	User and Administration
63	UAttributeName	Administration
64	UDirection	Administration
65	UserData	User
66	UserToUserInfo	Control
67	WindowSize	User and Administration
68	X213Cause	User
69	X213Origin	User
70	X25Cause	User
71	X25Diagnostic	User
72	CPParameterMask	Administration
73	CPMessageMask	Administration

Table III.1 – List of ISDN-PCI parameters *(concluded)*

Parameter identificator	Parameter name	Use in the plane
74	PPPNegotiation	User and Administration
75	FlowControlMechanism	Administration
76	FlowControlCharacters	Administration
77	MomentNumber	Administration
78	V110BChannelDisconnection	Administration
79	NumberComplete	Control
80	AdditionalInfo	Control
81	Signal	Control

Bibliography

This bibliography contains references to documents which may be of importance to the PUF and NAF developers. The documents can be useful when reading or implementing this part.

– ISO/IEC 9574:1992, *Information technology – Provision of the OSI connection-mode network service by packet mode terminal equipment to an integrated services digital network (ISDN)*.

– ISO/IEC 8878:1992, *Information technology – Telecommunications and information exchange between systems – Use of X.25 to provide the OSI Connection-mode Network Service.*

– ITU-T Recommendation Q.932 (1993), *Generic procedures for the Control of ISDN Supplementary Services*.

– ITU-T Recommendations Q.951.1, Q.951.2 and Q.951.3 (1992), Q.951.3, Q.951.4, Q.951.5 and Q.951.6 (1993), Q.951.7 (1997), *Stage 3 description for number identification supplementary services using DSS 1*.

– ITU-T Recommendations Q.953.1 (1992), Q.953.2 (1993), Q.953.3 (1997) and Q.953.4 (1995), *Stage 3 description for call completion supplementary services using DSS 1*.

– ITU-T Recommendation X.211 (1995), *Information technology – Open Systems Interconnection – Physical service definition.*

– ITU-T Recommendation Z.100 (1993), *CCITT specification and description language (SDL)*.

Superseded by a more recent version

CONTENTS

PART 3

Superseded by a more recent version

PART 3: USER PLANE PROTOCOLS MANAGEMENT ARCHITECTURE

Summary

This part of the multi-part specification provides the general aspects for the management and access to the user plane protocol supported by the PCI. In particular, it includes the user plane protocol architecture and the detailed description of the protocol selection mechanism.

Introduction

The use of different Integrated Services Digital Network (ISDN) programming interfaces by terminal equipment has hindered the development of common applications using ISDN which, in turn, has constrained deployment of ISDN applications on terminal equipment such as personal computers.

This ITU-T ISDN Application Programming Interface (API), called ISDN Programming Communication Interface (PCI), is an application interface for accessing and administering ISDN services. The ISDN-PCI comprises a set of specifications in which this part is the protocol usage introduction.

ISDN-PCI has been defined in order to provide a standard that terminal equipment providers that makes possible 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 future ISDN extensions will not affect the operation of existing applications.

1 Scope

This specification describes the Integrated Services Digital Network Programming Communication Interface (ISDN-PCI) User Plane access.

This part describes the ISDN-PCI protocol management provided by the User Plane. The User Plane provides operations which facilitate establishment, data exchange and/or release of logical communication channels. It provides messages that allow the use of underlying protocols. It 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.

The ISDN-PCI is located between layers 3 and 4 of the OSI reference model. In the case of transparent access, the NAF considers layers 2 and 3 as Null layers. In case of link access, the NAF considers layer 3 as Null layer and for network access layers 2 and 3 are implemented as shown in Figure 1.

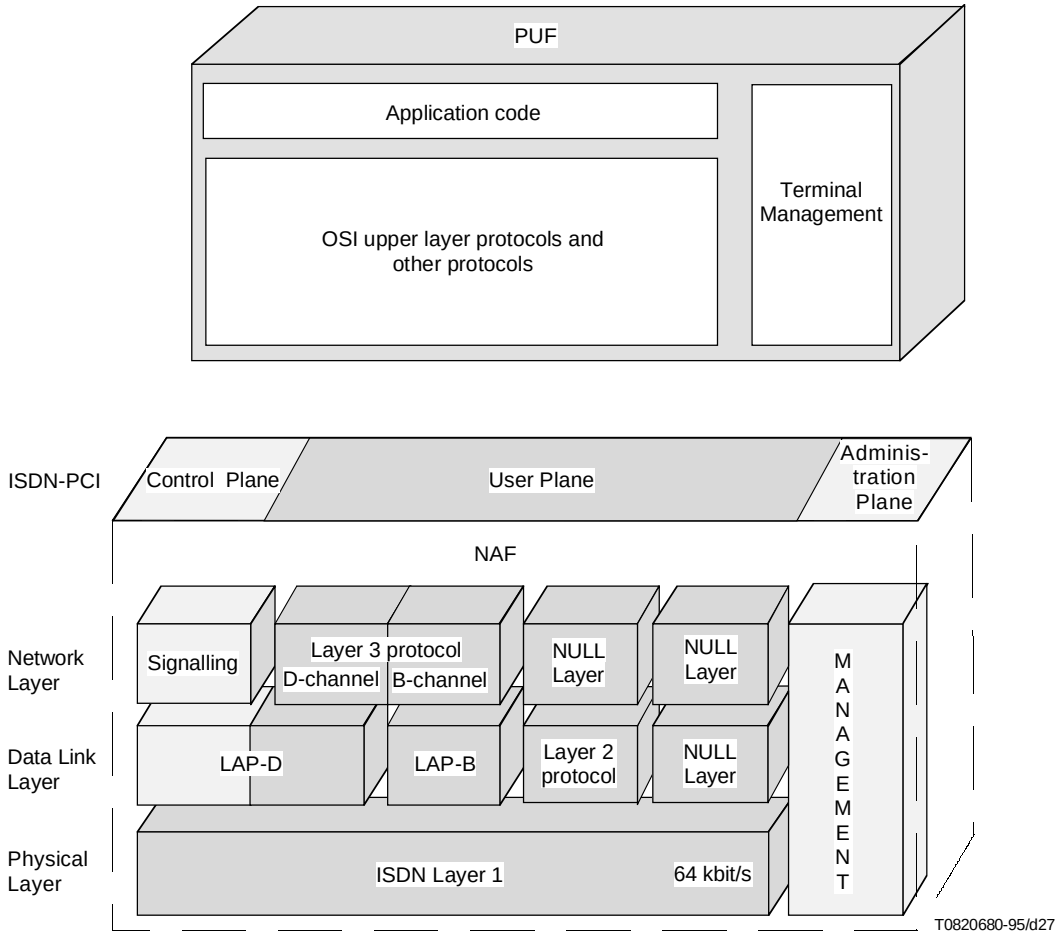


Figure 1 – OSI location

For the support of transparent access to the ISDN B-channel, the User Plane provides access to the Physical Service Access Point (Ph-SAP). The User Plane also allows for access of another Service Access Point (SAP). It provides the services defined in Recommendation X.213 and is, therefore, located at the Network layer Service Access Point (N-SAP).

This part specifies the usage of possible protocols for the data transfer service. The User Plane provides the services for a variety of protocols. They all are optional and may be divided into the following groups:

- layer 1 protocols;
- layer 2 protocols;
- layer 3 protocols.

This part is one of a multi-part document covering User Plane description. Each protocols group is described in a separate part:

Part 3: Describes general mechanism, coordination functionality and protocol selection mechanism.

Part 4: Defines protocol usage for layer one protocols:

- Transparent B-channel access.

Part 5: Defines protocol usage for layer two protocols:

- ISO/IEC 7776;
- HDLC;
- PPP;
- SDLC;
- V.110.

Part 6: Defines protocol usage for layer three protocols:

- T.90;
- ISO/IEC 8208;
- T.70 NL.

2 References

[1] ITU-T Recommendation X.213 (1995), *Information technology – Open Systems Interconnection – Network service definition.*

[2] Part 1, *General architecture*.

[3] Part 2, *Basic services.*

[4] Part 4, *Layer One protocols.*

[5] Part 5, *Layer Two protocols*.

[6] Part 6, *Layer Three protocols*.

3 Definitions

This part defines the following terms:

3.1 administration plane: Logical grouping of functionality for management of PUF-NAF dialogue as well as for access to local or network related NAF resources.

3.2 attribute set: Set of parameters driving user protocols and ISDN signalling.

3.3 B-channel: Logical ISDN channel for the use of data transfer.

3.4 control plane: Logical grouping of functionality for access of ISDN signalling.

3.5 D-channel: Logical ISDN channel used for signalling and, in some cases, for data transfer.

3.6 ISDN access: Set of ISDN channels provided by a single Network Access Facility (NAF) to access ISDN services.

3.7 ISDN programming communication interface (ISDN-PCI): Network (ISDN) oriented software interface providing access provisions for programming network signalling and user data exchange.

3.8 message: Unit of information transferred through the interface between the Network Access Facility (NAF) and the PCI User Facility (PUF).

3.9 network access facility (NAF): Functional unit located between the ISDN-PCI and the network related layers.

3.10 **network connection object (NCO)**: Abstract object within the NAF that shall be created by the PUF to gain access to network signalling or data.

3.11 **NULL layer**: 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.

3.12 **PCI user facility (PUF)**: Functional unit using the ISDN-PCI to access a NAF. In fact, the local application using the interface.

3.13 **user connection**: Connection accessible through User Plane functionality.

3.14 **user plane**: Logical grouping of functionality for access of user protocols and data.

3.15 **user protocol**: Protocol running and conforming to User Plane functionality.

4 Abbreviations

This part uses the following abbreviations:

API	Application Programming Interface
CONS	Connection Oriented Network Service
ISDN	Integrated Services Digital Network
LAP-B	Link Access Procedure Balanced
LAP-D	Link Access Procedure for D-channel
N-SAP	Network layer – Service Access Point
NAF	Network Access Facility
NCO	Network Connection Object
PCI	Programming Communication Interface
Ph-SAP	Physical layer – Service Access Point
PUF	Programming communication interface User Facility
SAP	Service Access Point

5 Reader's guidance

5.1 Reader's guide

This part is intended for software developers, implementors of applications and equipment manufacturers by providing them the general usage description of User Plane protocols.

5.2 How to use this part

Readers who:

- need a quick overview over the described User Plane protocols management need to refer to this part;
- intend to implement an application using the ISDN-PCI interface should read the other Parts [4], [5] and [6] depending on the desired protocol;
- intend to build an ISDN adaptor card or equipment using the ISDN-PCI interface should refer to Appendix I and read the other Parts [4], [5] and [6] depending on the desired protocol. SDL diagrams and configuration are provided in these specific parts.

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

Table 1 – List of this Part contents

Clause, Appendix	Contains …
Clause 1	… the scope of this part. This describes what this part covers
Clause 2	… references
Clause 3	… definitions of the terms used throughout this part
Clause 4	… definitions of the abbreviations used throughout this part
Clause 5	… an overview
Clause 6	… general presentation of protocol extension architecture
Appendix I	… NAF development guidelines
Appendix II	… key information on user protocols

This part provides:

- the definition of message accesses (see clause 2);
- the list of supported protocols and the selection method (see clause 3);
- coordination function information (see clause 4);
- common NCO selection criteria (see clause 5);
- error checking principles (see clause 6);
- AttributeSet content (see clause 7).

6 User Plane protocol management architecture

6.1 Introduction

The data management covers the functionality used to:

- establish data connections on already established physical connections, if needed;
- exchange data.

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

So far, three sets of messages are defined in the User Plane. One set allows access to User Plane protocols providing the OSI Network-layer service interface. The second one provides access to link-layer service interface. The last set provides a transparent interface where the PUF implements the protocol to be run over the connection.

For both types of access it is important that there exists a signalling connection before any data access can be granted. In general, establishment of that signalling connection is achieved by use of Control Plane functionality, described in [3], whereas the establishment of the data access is achieved, if necessary, using User Plane functionality.

In the following subclauses, the different methods for message access which are supported by the ISDN-PCI are explained.

6.2 Message access

6.2.1 The physical layer access (transparent access)

The ISDN-PCI supports a transparent message access. It provides access to the transparent layers 2 and 3 (NULL Layers) and thus provides direct access to the physical layer of ISDN, providing a byte synchronized control over a B-channel. The bearer service provided by the network (Bearer Capability) is not limited to digital service. For example, Bearer Capability may be "speech".

As with any message access, this type of message access offers its own set of operations. Table 2 provides an overview on User Plane operations for this message access.

Due to the nature of the access, only operations allowing direct byte stream access are provided for this message access; no user protocol is running. Thus, by establishment of a signalling connection, the transparent data access becomes available. Unlike the access via the network layer, only one data connection is accessible per signalling connection.

When using a connection on the transparent access with an NCO (NCOType C) which is associated with external equipment, the data generated on the connection shall be sent to the external equipment rather than used to generate transparent access messages. This case is outside the scope of this part.

Table 2 – User plane operations for transparent access

Operation name	Purpose of operation
Data	Data transfer
Error	Indicates an error has occurred

6.2.2 The link layer access

The ISDN-PCI supports a link layer message access. It provides access to the transparent layer 3 (NULL Layer) and thus provides direct access to the link layer of ISDN.

This message access offers its own set of operations. Depending on the user protocol, these operations are available or not. Table 3 provides an overview on User Plane operations for this message access.

Table 3 – User plane operations for link 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.
Disconnect	Disconnect connection
ReadyToReceive	Control the normal data flow
Error	Indicates an error has occurred

6.2.3 The network layer access

The ISDN-PCI supports a network layer message access. It provides access to the User Plane protocols running in the ISDN network layer. Thus, it provides access to a network layer connection.

This message access offers its own set of operations. Depending on the user protocol, these operations are available or not. Table 4 provides an overview on User Plane operations for this message access.

The operational set is based on Recommendation X.213 [1].

The ISDN-PCI provides, for some protocols, coordination functionality which removes the need for the PUF to use Control Plane functionality. This coordination functionality, which is available to the PUF demand, implicitly builds a signalling connection when a user connection is requested.

Table 4 – 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 Recommendation X.213 [1].	

6.3 Protocols

6.3.1 Supported User Plane protocol

There are different user layer protocols which can be accessed through the various accesses. One of these User Plane protocols is selectable at the creation of the NCO.

Table 5 summarizes protocols usage defined.

Table 5 – Supported user plane protocols

Protocol	Layer
Network layer according to Recommendation T.90	3
ISO/IEC 8208	3
Network layer of Recommendation T.70 NL	3
Null layer 3 with access to ISO/IEC 7776 on layer 2	2
Null layer 3 with transparent access to HDLC framing	2
Null layer 3 with transparent access to HDLC framing with error indication	2
PPP	2
SDLC	2
V.110 asynchronous (Note)	2
V.110 synchronous (Note)	2
Transparent B-Channel access with byte framing from the network	1
NOTE – A V.110 access is offered to a PUF at the level 2 but other ways to use this protocol are possible.	

6.3.2 Protocol selection

The protocol selection is made during the creation of the NCO, by the use of two parameters: NCOType and UProtocol parameters (see description of the ACreateNCOReq function in [3]).

6.3.2.1 NCOType parameter

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

Type: 50.

Field	Field type	Direction	Required	Comment
Identifier	Octet	P	M	C (1) – Signalling access only (Note) U3 (2) – Network user access with NAF signalling coordination (NAF coordination functionality) C/U (3) – Signalling and network, link or physical user access. U3/G (4) – Network user access to additional virtual circuits. This NCO must be grouped to an already created U3 or C/U type NCO.
NOTE – NCO type C is outside the scope of Parts 3 to 6.				

6.3.2.2 UProtocol parameter

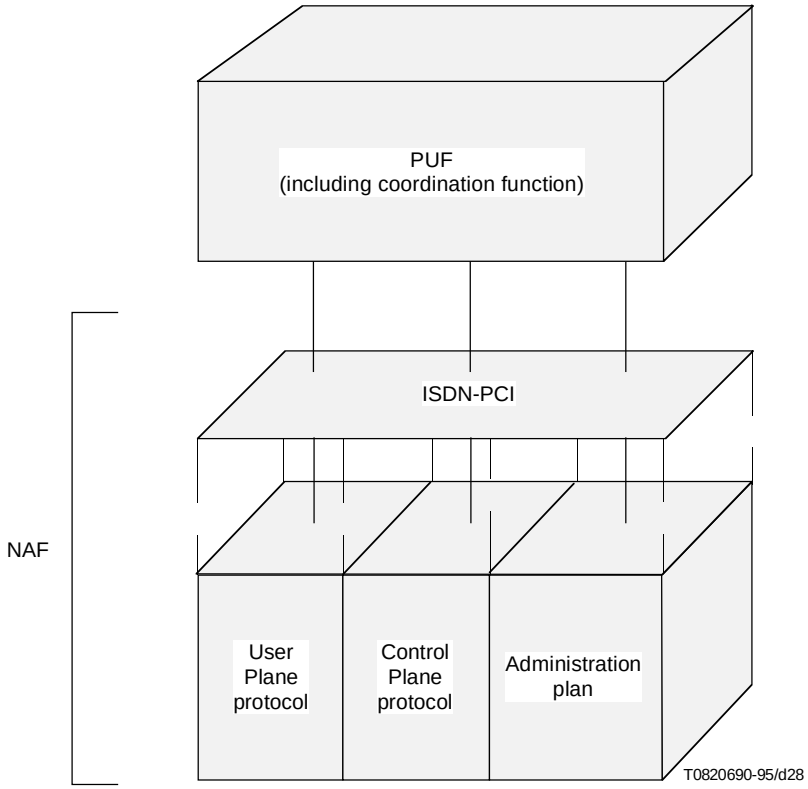
Description: This is used to select the User Plane protocol. If the length is 3, the first octet contains the layer 3 protocol requested, the second octet contains the layer 2 protocol requested and the third octet contains the layer 1 protocol requested.

Type: 62.

Field	Field type	Direction	Required	Comment
L3Protocol	Octet	P	M	Default (255) – T.90 T.90 (1) ISO/IEC 8208 (2) T.70 NL (3) NULL (4)
L2Protocol	Octet	P	C (Note 1)	Default (255) – ISO/IEC 7776 ISO/IEC 7776 (1) Frame oriented transparent (2) Frame oriented transparent with error indication (3) PPP (4) SDLC (5) V.110 asynchronous (6) V.110 synchronous (7) NULL (8)
L1Protocol	Octet	P	C (Note 2)	Default (255) – Transparent access with byte framing from the network Transparent access with byte framing from the network (1)
NOTE 1 – Mandatory if L3Protocol is NULL. NOTE 2 – Mandatory if L3Protocol and L2Protocol are NULL.				

6.4 Coordination function

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 coordination between signalling and user connection is handled only by the PUF. Figure 2 shows the PUF provided coordination function. As a result of controlling the signalling connection, the PUF can use the supplementary services.



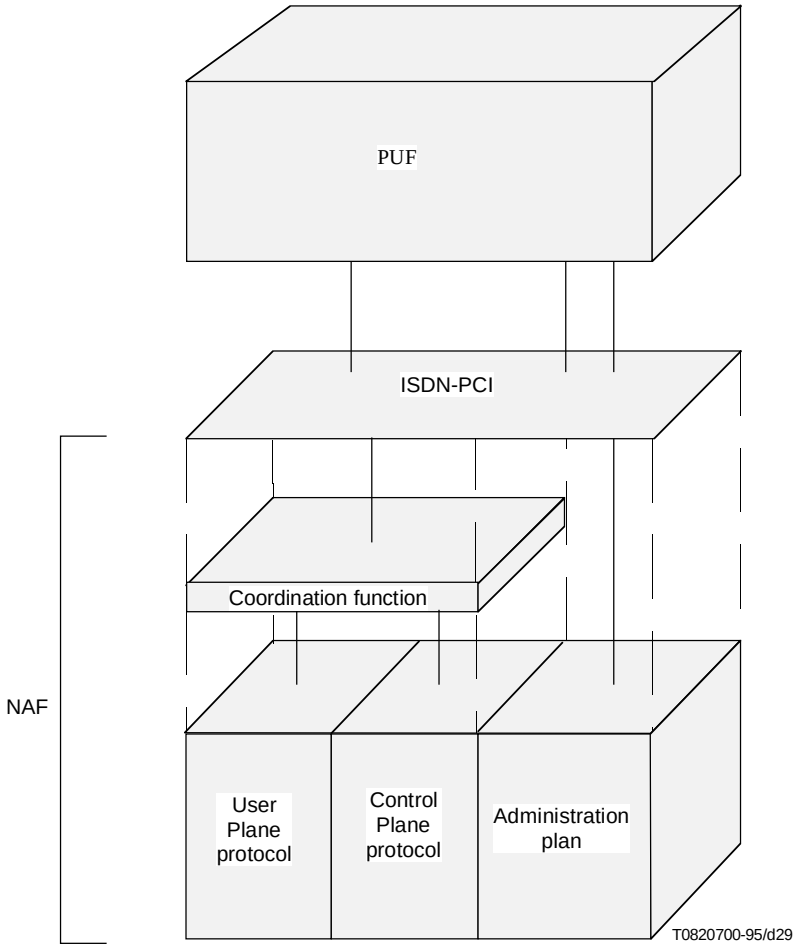
NOTE – The existence of a coordination function inside the PUF is outside the scope of this part.

Figure 2 – PUF coordination

The PUF may be offered an ISO Connection-mode Network Service (CONS) as defined in Recommendation X.213. This abstraction is provided by a coordination function, which maps the primitives of CONS in the User Plane according to the primitives of the Control Plane and User Plane protocols. The coordination function can only be used with the User Plane protocols relating to Recommendation X.213. The coordination function is provided as part of the NAF. Since the NAF manages the coordination between signalling and user connection, the PUF shall not access the Control Plane. Figure 3 shows the NAF provided coordination function.

The coordination function does not affect the Administration Plane.

Even if the coordination function is used by the PUF, the layers 2 and 3 protocols used are the selected protocols for the Network and Link access.



NOTE – The coordination function is only defined for User Plane protocols related to Recommendation X.213.

Figure 3 – NAF coordination

For achieving a connection which is to be NAF coordinated, the PUF exchanges the following message:

ACreateNCOReq, with NCOType U3 and the relevant information.

A connection can then be requested using the UConnectReq. 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 the State diagrams, see Parts 4 to 6 [4], [5] and [6].

The coordination function may not be available for all the User Plane protocols. Therefore the coordination function availability is noted in each relevant user protocol part.

6.5 Selection criteria

6.5.1 NCO Selection – User Plane information element

In order to apply the right NCO on an incoming call, the NAF uses various criterias. General mechanism and Control Plane information elements are described in Part 2 [3].

Some User Plane information elements can also be applied for the NCO selection. Useful elements are particular to the protocol in use. For example, in case of ISO/IEC 8208, User Plane information elements are the following:

– packet size negotiation;

– window size negotiation.

See other Parts (Parts 4 to 6) for User Plane information elements to be used.

6.5.2 Action if no NCO available – User Plane incoming call

A disconnect with the protocol specific reason is issued by the NAF. The exact reason is provided in the protocol relevant subclause.

6.6 User plane error checking

Administrative message error information is returned in the Administration Plane error message. For details on the message error handling, refer to Part 2 [3].

The protocol error detection takes place after administrative checking and the mechanism used to return error information depends on the protocol. These mechanisms are described in the relevant subclause of each protocol description.

Invalid content parameter and invalid length of user data are considered as protocol error.

6.7 User plane attribute sets

Attribute sets are used to keep together important parameters for driving user protocols. A collection of attribute sets exists for this plane. They are defined in each protocol subclause [4], [5] and [6].

Appendix I

NAF development guidelines

The main body of this part contains the description of the ISDN-PCI from the PUF point of view. Following this approach, some 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 appendix. It gives guidelines for the development of the NAF in accordance with the main body of this part.

Consider this appendix from the following viewpoints:

– This appendix gives additional points. The NAF has to be implemented using this part. It should implement the ISDN-PCI in such a way that the functionality described is provided.

– The main body of this part should be given priority if there is anything not clear in this appendix or the interpretation between the main body of this part and this appendix is different.

– This appendix 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 part.

I.1 User plane error management

The error processing for this plane is defined for each protocol in the relevant part [4], [5], [6].

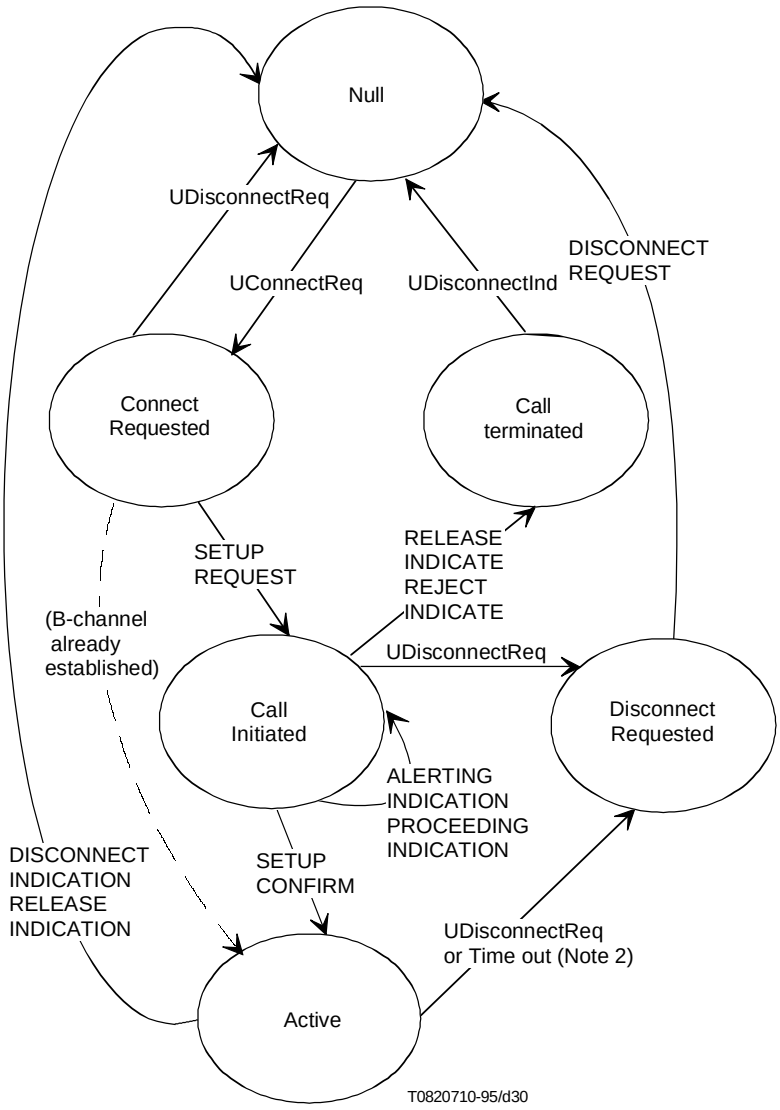
I.2 NAF configuration

Global Configuration, system configuration and Control plane configuration are provided in Part 2.

User plane configuration can be found in Parts 4 to 6 [4], [5], [6], depending on the protocol.

I.3 Coordination function – Outgoing User Plane call

Figure I.1 shows the establishment of the Control Plane connection. The states indicated are internal to the NAF.



NOTE 1

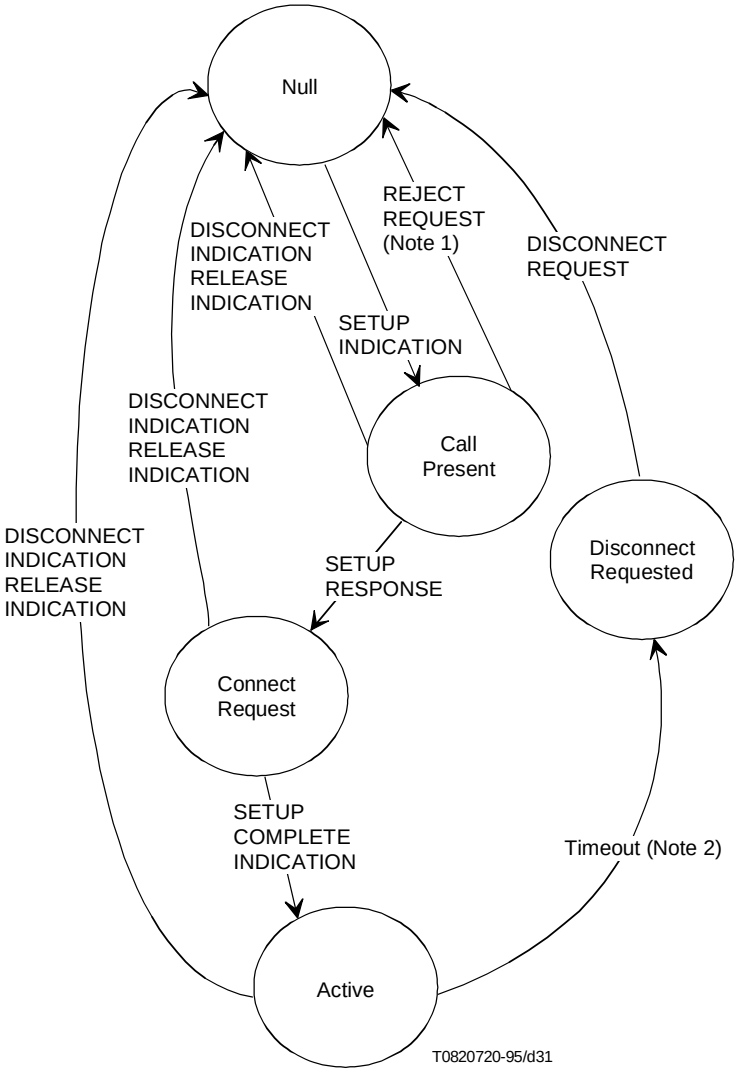
– events in upper case are primitives described in Recommendation Q.931;
– events in mixed case are PCI User Plane messages;
– the states shown are internal to the NAF.

NOTE 2 – 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 an NAF design consideration.

Figure I.1 – Coordination function – Outgoing call and channel establishment

I.4 Coordination function – Incoming ISDN call

The following state diagram (see Figure I.2) shows the establishment of the Control Plane connection. The states indicated are internal to the NAF.



NOTE 1

– events in upper case are primitives described in Recommendation Q.931;

– the states shown are internal to the NAF.

NOTE 2 – 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 an NAF design consideration.

NOTE 3 – The NAF may reject the ISDN connection request.

Figure I.2 – ISDN incoming call and coordination function

Superseded by a more recent version

Appendix II

User protocols

This appendix summarizes key information to make use of User Plane protocols. Values in parenthesis are decimal coded.

Protocol	NCOType	UProtocol	UDirection	Function coordination
Transparent access	. C/U	. NULL (4) . NULL (8) . Transparent access (1)	"Both" or absent	No
V.110	. C/U	. NULL (4) . V.110 (6 or 7) . (...)	"Both" or absent	No
PPP	. C/U	. NULL (4) . PPP (4) . (...)	"Both" or absent	No
SDLC	. C/U	. NULL (4) . SDLC (5) . (...)	"Both" or absent	No
HDLC	. C/U	. NULL (4) . HDLC (2 or 3) . (...)	"Both" or absent	No
ISO/IEC 7776	. C/U	. NULL (4) . ISO/IEC 7776 (1) . (...)	"Both" or absent	No
T.70	. C/U	. T.70 (3) . / . (...)	"Both" or absent	No
ISO/IEC 8208	. C/U . U3 . U3/G	. ISO/IEC 8208 (2) . (...) . (...)	Used	Yes
T.90	. C/U . U3 . U3/G	. T.90 (1) . ISO/IEC 7776 (1) . (...)	Used	Yes
NOTE – When (...) is used, it means that there is not a fixed value.				

Superseded by a more recent version

CONTENTS

PART 4

PART 4: LAYER ONE PROTOCOLS

Summary

This part of the specification details procedures, messages and parameters used to access the ISDN-PCI's User Plane protocols that provide a layer 1 communication service.

Introduction

The use of different Integrated Services Digital Network (ISDN) programming interfaces by terminal equipment has hindered the development of common applications using ISDN which, in turn, has constrained deployment of ISDN applications on terminal equipment such as personal computers.

This ITU-T ISDN Application Programming Interface (API), called ISDN Programming Communication Interface (PCI), is an application interface for accessing and administering ISDN services. The ISDN-PCI comprises a set of specifications in which this part is the layer 1 protocol usage description.

ISDN-PCI has been defined in order to provide a standard for terminal equipment providers that makes possible 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 future ISDN extensions will not affect the operation of existing applications.

Superseded by a more recent version

1 Scope

This part describes the Integrated Services Digital Network Programming Communication Interface (ISDN-PCI) layer 1 protocols provided by the ISDN-PCI User Plane. It forms a part of specification on ISDN-PCI.

It describes the specific elements (messages, parameters, attribute sets, etc.) relating to the layer 1 user protocols. This part covers the Transparent B-channel access user protocol. ITU-T Recommendations describing other layer 1 user protocols are for further study.

2 References

[1] Part 1, *General architecture.*

[2] Part 2, *Basic services.*

[3] Part 3, *User plane protocols management architecture.*

3 Definitions

This part defines the following terms:

3.1 **attribute set**: Set of parameters driving user protocols and ISDN signalling.

3.2 **B-channel**: Logical ISDN channel for the use of data transfer.

3.3 **control plane**: Logical grouping of functionality for access of ISDN signalling.

3.4 **D-channel**: Logical ISDN channel used for signalling and in some cases, for data transfer.

3.5 **ISDN access**: Set of ISDN channels provided by a single Network Access Facility (NAF) to access ISDN services.

3.6 **ISDN programming communication interface (ISDN-PCI)**: ISDN oriented software interface providing access provisions for programming network signalling and user data exchange.

3.7 **message**: Unit of information transferred through the interface between the Network Access Facility (NAF) and the PCI User Facility (PUF).

3.8 **network access facility (NAF)**: Functional unit located between the ISDN-PCI and the network related layers.

3.9 **network connection object (NCO)**: Abstract object within the NAF that is created by the PUF to gain access to network signalling or data.

3.10 **NULL layer**: Describes an empty layer of the OSI reference model. Such a layer does not contain any functionality and passes requests and responses transparently to adjoining layers.

3.11 **PCI user facility (PUF)**: Functional unit using the ISDN-PCI to access a NAF, e.g. the local application using the interface.

3.12 **user connection**: Connection accessible through User Plane functionality.

3.13 **user plane**: Logical grouping of functionality for access of user protocols and data.

3.14 **user protocol**: Protocol running and conforming to User Plane functionality.

4 Abbreviations

This part uses the following abbreviations:

API	Application Programming Interface
ISDN	Integrated Services Digital Network
LAP-B	Link Access Procedure Balanced
LAP-D	Link Access Procedure for D-channel
NAF	Network Access Facility

NCO Network Connection Object

PCI Programming Communication Interface

PUF Programming communication interface User Facility

SAP Service Access Point

5 Reader's guidance

5.1 Reader's guide

This part is intended for software developers, implementors of applications and equipment manufacturers by providing them the usage description of layer 1 User Plane protocols.

5.2 How to use this part

Readers who:

- need a quick overview over the described User Plane protocols need to refer to the Part 3 [3];
- intend to implement an application using the ISDN-PCI interface with a layer 1 user protocol should read this specification. Protocol usage is provided in clause 6;
- intend to build an ISDN adaptor card or equipment using the ISDN-PCI interface with a layer 1 user protocol should read this specification. Protocol usage is provided in clause 6, while Appendix I describes informative default configuration values.

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

Table 1 – List of contents

Clause, Annex, Appendix	**Contains** …
Clause 1	… the scope of this part. This describes what this part covers
Clause 2	… references
Clause 3	… definitions of the terms used throughout this part
Clause 4	… definitions of the abbreviations used throughout this part
Clause 5	… gives an overview
Clause 6	… transparent access
Appendix I	… informative default configuration values

For each supported protocol, this part provides:

- description of available user messages (see clause 2);
- description of useful user parameters (see clause 3);
- the protocol state diagram (see clause 4);
- coordination function information (see clause 5);
- specific NCO selection criteria (see clause 6);
- specific error handling and codes (see clause 7);
- AttributeSet definition (see clause 8).

Appendix I gives default protocol configuration values if any.

6 Transparent B-channel access with byte framing from the network

6.1 Introduction

This clause deals with the Transparent B-channel access protocol with byte framing from the network. The BearerCap parameter indicates if the B-channel is used at 64 kbit/s or 56 bit/s.

For this access, the NAF considers layers 2 and 3 as a Null layers, as shown in Figure 1.

The OSI location of the 64 kbit/s transparent protocol is shown in Figure 1.

General description conventions are provided in [2].

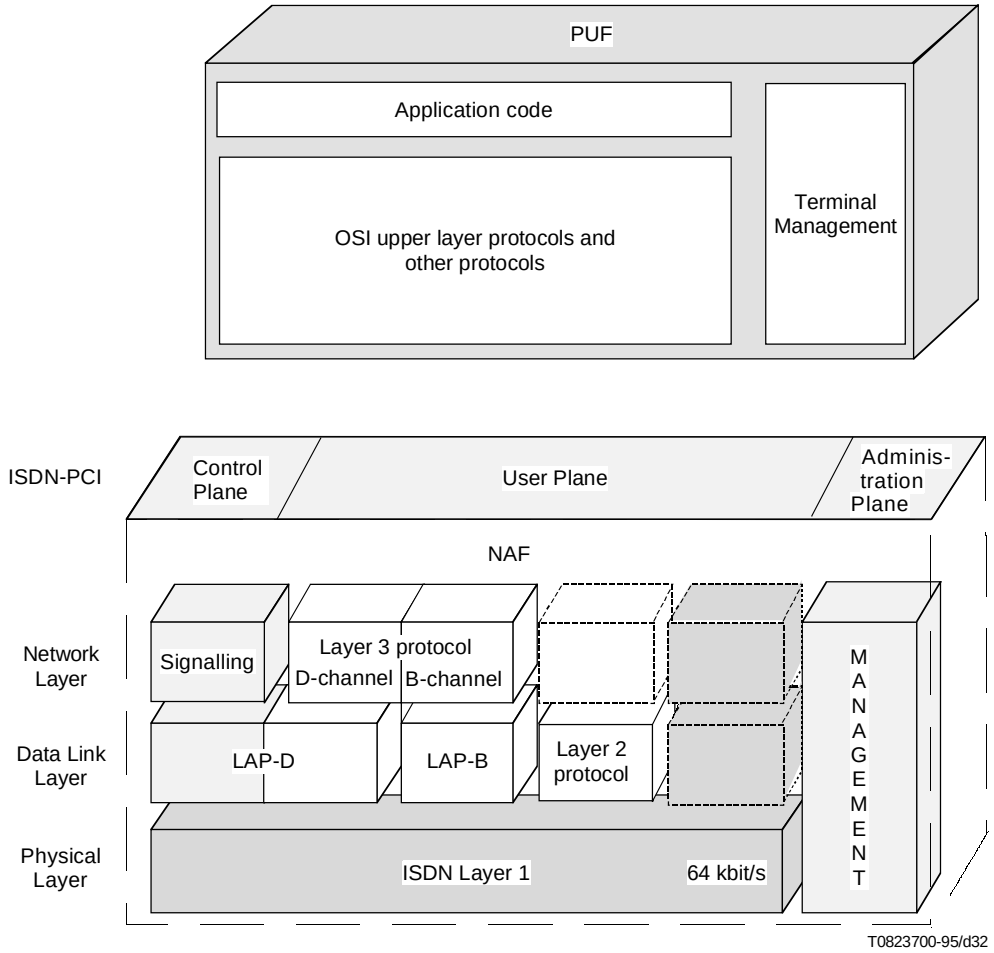


Figure 1 – OSI location

6.2 Messages

Table 2 gives an overview of user messages.

Table 2 – Overview of user messages

Message identifier	Class	Message name	Purpose of message
307	1	UDataReq	Request transfer of data
308	1	UDataInd	Indicate arrival of transferred data
319	1	UErrorInd	Indicate an error

6.2.1 UDataReq

Class: 1 (Basic class).

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

Parameter:

Name	Required	Comment
NCOID	M	Identifies the Control Plane connection

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

Mandatory data shall be provided in the data buffer.

Related: None.

6.2.2 UDataInd

Class: 1 (Basic class).

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

Parameter:

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: UErrorInd.

6.2.3 UErrorInd

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
Cause	M	Identifies type of error

Related: None.

6.3 Messages parameters

This subclause describes parameters for the plane. Table 3 summarizes used parameters.

Table 3 – Overview of user parameters

Parameter identifier	Parameter name	Used in user messages	Used in UAttribute Set	Other use
35	IdleFlag		X	
50	NCOType			X
62	UProtocol		X	
63	UAttributeName			X
64	UDirection			X
68	Cause	X		

6.3.1 IdleFlag

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

Type: 35.

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

6.3.2 NCOType

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

Type: 50.

Field	Field type	Direction	Required	Comment
Identifier	Octet		M	C/U (3) – Signalling and transparent user access

6.3.3 UProtocol

Description: This is used to select the User Plane protocol. The first byte contains the layer 3 protocol requested, the second contains the layer 2 protocol requested and the third contains the layer 1 protocol requested.

Type: 62.

Field	Field type	Direction	Required	Comment
L3Protocol	Octet	P	M	NULL (4)
L2Protocol	Octet	P	M	NULL (8)
L1Protocol	Octet	P	M	Transparent B-channel access (1)

Remark: Other values (for other protocols) are provided in [3].

6.3.4 UAttributeName

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

Type: 63.

Field	Field type	Direction	Required	Comment
AttributeName	IA5-string	P	M	16 bytes is the maximum length

6.3.5 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.

Field	Field type	Direction	Required	Comment
Direction	Octet	P	O	both (3)

6.3.6 Cause

Description: This parameter is used to pass Cause information for disconnection to the PUF.

Type: 68.

Field	Field type	Direction	Required	Comment
Value	Octet	N	M	210 – Overflow

6.4 State diagram

User messages do not change the state of the connection.

6.5 Coordination function

The coordination function cannot be used with the User Plane protocol relating to the transparent B-channel access.

6.6 Selection criteria

No specific parameters are used. General NCO criteria are provided in [2].

6.7 Specific error handling

Errors are dealt with in the following manner: in case of overflow of Incoming Data, PUF is sent UErrorInd.

6.8 Static attributes

6.8.1 AttributeSet parameters

Table 4 – User plane Attribute Set (UAttributeSet) parameters

Parameter	Required	Comment
IdleFlag	C	Flag byte to be sent while idle. See 6.3.1.
Uprotocol	O	See 6.3.3

Remark: It is only possible to use these parameters during NCO creation containing Control Plane information. Refer to subclause – ACreateNCO operation – in [2] for details.

If parameters are omitted, defaults shall be used. The default values described in Appendix I.

6.8.2 Static attribute content

Name:	U-TRANSPARENT
UProtocol:	64 kbit/s
IdleFlag:	0xFF

Appendix I

Configuration

I.1 Transparent B-channel access

Table I.1 – User Plane configuration for the transparent B-channel access

Parameter	Suggested default	Comment
Idle flag default value	0xFF	

Superseded by a more recent version

CONTENTS

PART 5

Superseded by a more recent version

PART 5: LAYER TWO PROTOCOLS

Summary

This part of the specification details Procedures, Messages and Parameters used to access the ISDN-PCI's User Plane Protocols that provide a Layer 2 communication service.

Introduction

The use of different Integrated Services Digital Network (ISDN) programming interfaces by terminal equipment has hindered the development of common applications using ISDN which, in turn, has constrained deployment of ISDN applications on terminal equipment such as personal computers.

This ITU-T ISDN Application Programming Interface (API), called ISDN Programming Communication Interface (PCI), is an application interface for accessing and administering ISDN services. The ISDN-PCI comprises a set of specifications in which this part is the Layer 2 protocol usage description.

ISDN-PCI has been defined in order to provide a standard for terminal equipment providers that makes possible 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 future ISDN extensions will not affect the operation of existing applications.

1 Scope

Part 5 describes the Integrated Services Digital Network Programming Communication Interface (ISDN-PCI) layer 2 protocols provided by the ISDN-PCI User Plane. It forms a part of the specification on ISDN-PCI.

It describes the specific elements (messages, parameters, attribute sets, etc.) relating to the layer 2 user protocols. This part covers ISO/IEC 7776, HDLC with or without error indication, PPP, SDLC and V110 user protocols. Other layer 2 user protocol descriptions are for further study.

2 References

[1] Part 1, *General architecture.*

[2] Part 2, *Basic services.*

[3] Part 3, *Protocol extension architecture.*

[4] ISO/IEC 7776:1995, *Information technology – Telecommunications and information exchange between systems – High-level data link control procedures – Description of the X.25 LAPB-compatible DTE data link procedures.*

[5] ITU-T Recommendation V.110 (1996), *Support of data terminal equipments with V-Series type interfaces by an integrated services digital network.*

3 Definitions

This part defines the following terms:

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

3.2 **B-channel**: Logical ISDN channel for the use of data transfer.

3.3 **control plane**: Logical grouping of functionality for access of ISDN signalling.

3.4 **D-channel**: Logical ISDN channel used for signalling and, in some cases, for data transfer.

3.5 **ISDN access**: Set of ISDN channels provided by a single Network Access Facility (NAF) to access ISDN services.

3.6 **ISDN programming communication interface (ISDN-PCI)**: Network (ISDN) oriented software interface providing access provisions for programming network signalling and user data exchange.

3.7 **message**: Unit of information transferred through the interface between the Network Access Facility (NAF) and the PCI User Facility (PUF).

3.8 **network access facility (NAF)**: Functional unit located between the ISDN-PCI and the network related layers.

3.9 **network connection object (NCO)**: Abstract object within the NAF that shall be created by the PUF to gain access to network signalling or data.

3.10 **NULL layer**: 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.

3.11 **PCI user facility (PUF)**: Functional unit using the ISDN-PCI to access an NAF. In fact, the local application using the interface.

3.12 **user connection**: Connection accessible through User Plane functionality.

3.13 **user plane**: Logical grouping of functionality for access of user protocols and data.

3.14 **user protocol**: Protocol running and conforming to User Plane functionality.

4 Abbreviations

This part uses the following abbreviations:

API	Application Programming Interface
ISDN	Integrated Services Digital Network
LAP-B	Link Access Procedure Balanced
LAP-D	Link Access Procedure for D-channel
NAF	Network Access Facility
NCO	Network Connection Object
PCI	Programming Communication Interface
PUF	Programming communication interface User Facility

5 Reader's guidance

5.1 Reader's guide

This part is intended for software developers, implementors of applications and equipment manufacturers by providing them the usage description of layer 2 User Plane protocols.

5.2 How to use this part

Readers who:

- need a quick overview of the described User Plane protocols should refer to Part 3;
- intend to implement an application using the ISDN-PCI interface with a layer 2 user protocol should read this part. Protocol usage is provided in clauses 6 to 11;
- intend to build an ISDN adaptor card or equipment using the ISDN-PCI interface with a layer 2 user protocol should read this part. Protocol usage is provided in clauses 6 to 11, while Appendices I and II describe default configuration values and NAF diagrams.

Table 1 provides a list showing the major clauses in this part.

Table 1 – List of contents

Clause, Annex, Appendix	Contains ...
Clause 1	... the scope of this part. This describes what this part covers
Clause 2	... references
Clause 3	... definitions of the terms used throughout this part
Clause 4	... definitions of the abbreviations used throughout this part
Clause 5	... gives an overview
Clause 6	... ISO/IEC 7776 protocol
Clause 7	... HDLC protocol
Clause 8	... HDLC protocol with error indication
Clause 9	... PPP protocol
Clause 10	... SDLC protocol
Clause 11	... V.110 protocol
Appendix I	... informative default configuration values
Appendix II	... informative NAF SDL diagrams

For each supported protocol, this part provides:

- description of available user messages (see clause 2);
- description of useful user parameters (see clause 3);
- the protocol state diagram (see clause 4);
- coordination function information (see clause 5);
- specific NCO selection criteria if it exists (see clause 6);
- specific error handling and codes (see clause 7);
- AttributeSet definition (see clause 8).

Appendix I gives default protocol configuration values if any.

Appendix II shows NAF-SDL diagrams describing most of the situations.

6 ISO/IEC 7776 protocol

6.1 Introduction

This clause deals with the ISO/IEC 7776 protocol.

The User Plane provides the services for ISO/IEC 7776 protocol using the User Plane protocols on a connection on B-channel. For this access, the NAF considers layer 3 as a Null, as shown in Figure 1.

The OSI location of the ISO/IEC 7776 protocol is shown in Figure 1.

The general description conventions are provided in [2].

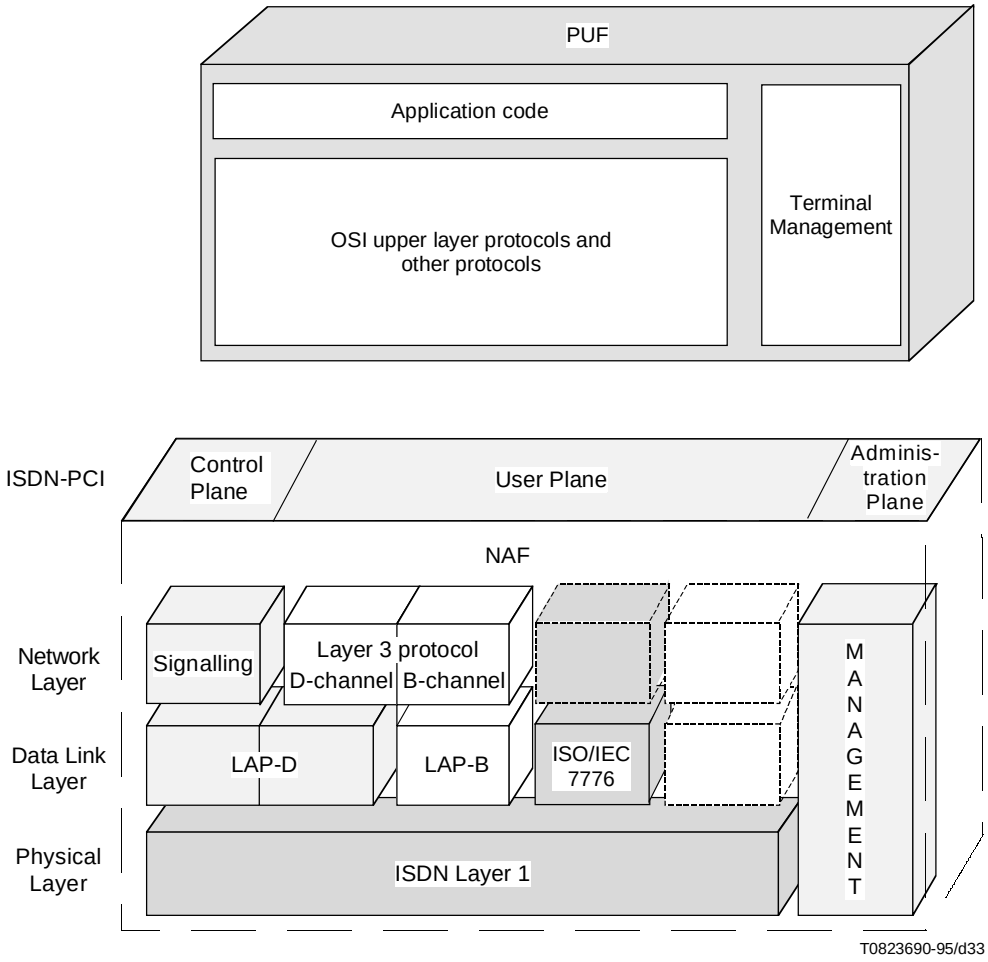


Figure 1 – OSI location

6.2 Messages

The user plane messages provide an access to ISO/IEC 7776 protocol stacks. Following is a list and short description of relevant user plane messages. Table 2 gives an overview of these messages.

Table 2 – Overview of user messages

Message identifier	Class	Message name	Purpose of message
301	1	UConnectReq	Request establishment of a user connection
302	1	UConnectInd	Indicate establishment of a user connection has been requested
303	1	UConnectRsp	Indicate acceptance of user connection establishment
304	1	UConnectCnf	Confirm user connection has been established
305	1	UDisconnectReq	Request removal of user connection
306	1	UDisconnectInd	Indicate removal of user connection
307	1	UDataReq	Request data transfer on an established user connection
308	1	UDataInd	Indicate arrival of transferred data on an established user connection
317	1	UReadyToReceiveReq	Used to perform flow control for a user connection
318	1	UReadyToReceiveInd	Used to indicate flow control status on a user connection

6.2.1 UConnectReq

Class: 1 (Basic class).

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

Parameter:

Name	Required	Comment
NCOID	M	Identifies the user connection

Related: UConnectCnf.

6.2.2 UConnectInd

Class: 1 (Basic class).

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

Parameter:

Name	Provided	Comment
NCOID	M	Identifies the user connection

Related: UConnectRsp.

6.2.3 UConnectRsp

Class: 1 (Basic class).

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

Parameter:

Name	Required	Comment
NCOID	M	Identifies the user connection

Related: UConnectInd.

6.2.4 UConnectCnf

Class: 1 (Basic class).

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

Parameter:

Name	Provided	Comment
NCOID	M	Identifies the user connection

Related: UConnectReq.

6.2.5 UDisconnectReq

Class: 1 (Basic class).

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

Parameter:

Name	Required	Comment
NCOID	M	Identifies the user connection

Related: None.

6.2.6 UDisconnectInd

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
Origin	M	Identifies the initiator of the user connection removal
Cause	M	Identifies the reason of the user connection removal

Related: None.

6.2.7 UDataReq

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 layer 2 data packet size defined at the NCO creation time.

Parameter:

Name	Required	Comment
NCOID	M	Identifies the user connection

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

Mandatory data shall be provided in the data buffer.

Related: None.

6.2.8 UDataInd

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 described at the NCO creation time.

Parameter:

Name	Provided	Comment
NCOID	M	Identifies the user connection

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

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

Related: None.

6.3 Messages parameters

This subclause describes parameters for the ISO/IEC 7776 protocol. Table 3 summarizes used parameters. They are alphabetically ordered.

Table 3 – Overview of user parameters

Parameter identifier	Parameter name	Use in user messages	Use in UAttributeSet	Other use
38	L2ConnectionMode		X	
40	L2WindowSize		X	
41	L2XID		X	
50	NCOType			X
62	UProtocol		X	
63	UAttributeName			X
64	UDirection			X
68	Cause	X		
69	Origin	X		

6.3.1 L2ConnectionMode

Description: This parameter is used only if it is not defined in L2XID value field. It is used to pass details of the layer connection mode to the NAF.

Type: 38.

Field	Field type	Direction	Required	Comment
Value	Octet	P	M	dte (1) – Act as secondary link station (non negotiable) dce (2) – Act as primary link station (non negotiable) auto (3) – Link station role is negotiable by XID exchange

6.3.2 L2WindowSize

Description: This parameter is used only if it is not defined in L2XID value field. It is used to pass details of the layer 2 window size to the NAF.

Type: 40.

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

6.3.3 L2XID

Description: This is used to pass details of the layer 2 XID value and its use. XID information field may include values that override some parameters defined elsewhere.

Type: 41.

Field	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 octets.

6.3.4 NCOType

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

Type: 50.

Field	Field type	Direction	Required	Comment
Identifier	Octet	P	M	C/U (3) – Signalling and network layer user access

6.3.5 UProtocol

Description: This is used to select the User Plane protocol.

Type: 62.

Field	Field type	Direction	Required	Comment
L3Protocol	Octet	P	M	NULL (4)
L2Protocol	Octet	P	M	ISO/IEC 7776 (0)
L1Protocol	Octet	P	O	Default (255) – Transparent B-channel access

Remark: Other possible values (for other protocols) are provided in Part 2 [2].

6.3.6 UAttributeName

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

Type: 63.

Field	Field type	Direction	Required	Comment
AttributeName	IA5-string	P	M	16 bytes is the maximum length

6.3.7 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.

Field	Field type	Direction	Required	Comment
Direction	Octet	P	M	both (3)

6.3.8 Cause

Description: This parameter is used to pass Cause information for disconnection to the PUF.

Type: 68.

Field	Field type	Direction	Required	Comment
Value	Octet	N	M	Value provided in Table 4

6.3.9 Origin

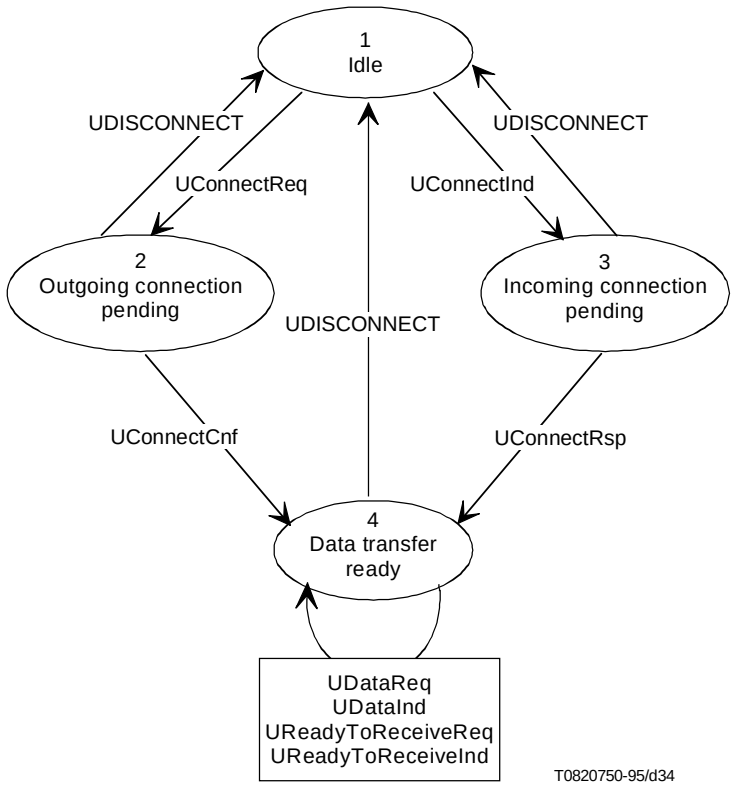
Description: This parameter is used to pass the origin information of the disconnection to the PUF.

Type: 69.

Field	Field type	Direction	Required	Comment
Value	Octet	N	M	undefined (1) NAF Provider (2) Remote User (3)

6.4 State diagram

Figure 2 shows the different states a user connection can get, using the U-messages, and in which order these messages shall be used.



NOTE – Where UDISCONNECT appears, it can be either UDisconnectReq or UDisconnectInd.

Figure 2 – Overview of the user plane messages

6.5 Coordination function

The coordination function cannot be used with the User Plane protocol relating to the ISO/IEC 7776 protocol.

6.6 Selection criteria

No ISO/IEC 7776 protocol specific parameters are used. General NCO criteria are provided in [2].

6.7 Specific error handling and codes

In case of invalid length of UDataReq UserData, parameter PUF is sent UDisconnectInd.

The Table 4 gives possible values of the Cause parameter.

Table 4 – Cause parameter value

Return code		Meaning	ErrorSpecific information
Undefined	220	Undefined error situation	Not present
DiscNorm	241	Disconnection – Normal condition	Not present
InvalidSequence	244	Connection rejected – Invalid sequencing in the frame numbering (transient condition)	Not present
FrameTooBig	245	Connection rejected – Reception of a frame bigger than defined in the NCO value (fixed condition)	Not present

6.8 Static attributes

6.8.1 AttributeSet parameters

Table 5 – User plane Attribute Set (UAttributeSet) parameters

Parameter	Required	Comment
Uprotocol	O	See Remark. See also 6.5.
L2ConnectionMode	O	See Remark. See also 6.1.
L2WindowSize	O	See Remark. See also 6.2.
L2XID	O	See Remark. See also 6.3.

Remark: It is only possible to use these parameters during NCO creation containing control plane information. Refer to subclause – ACreateNCO operation – in [2] for details.

If parameters are omitted, defaults shall be used by the NAF. Default values are described in Appendix I.

6.8.2 Static attribute content

Name:	U_ISO7776
L2FrameSize:	128
L2WindowSize:	7
L2ConnectionMode:	Auto
L2XID:	Send and match

7 HDLC protocol

7.1 Introduction

This clause deals with the HDLC protocol.

For this access, the NAF considers layer 3 as a Null, as shown in Figure 3.

The OSI location of the HDLC protocol is shown in Figure 3.

General description conventions are provided in [2].

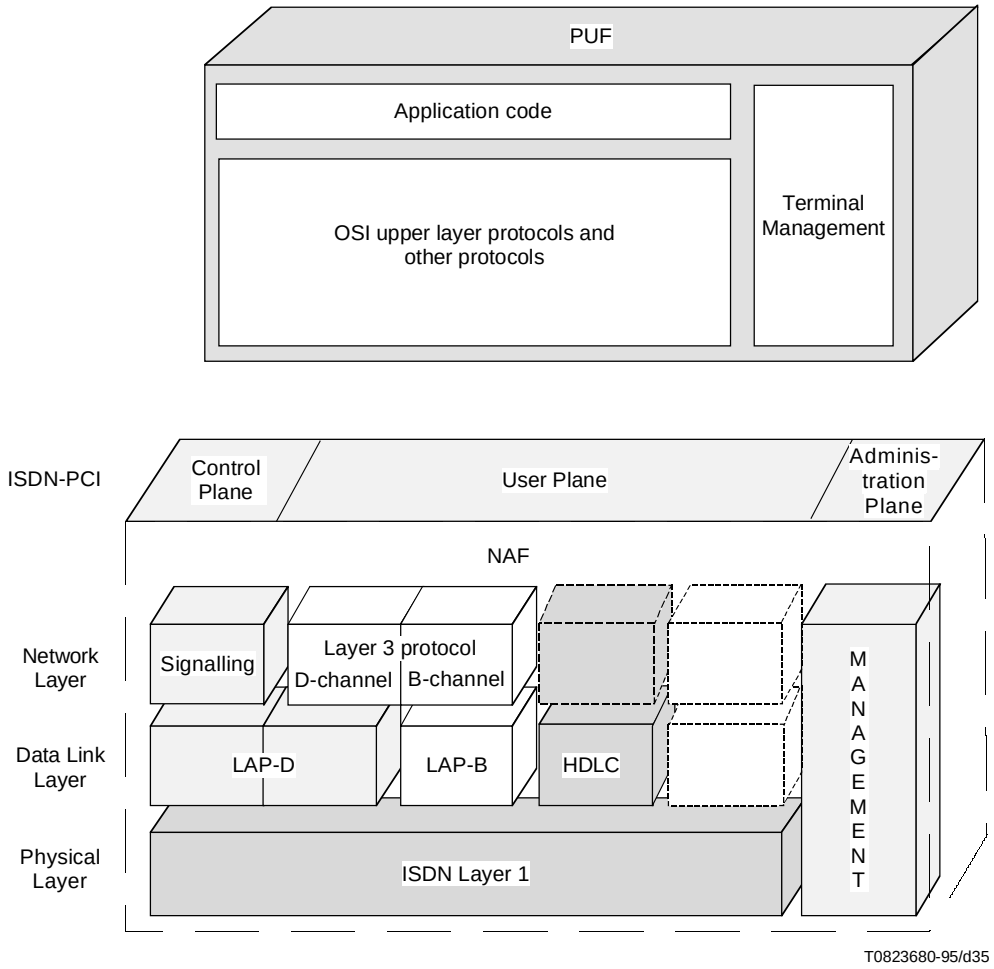


Figure 3 – OSI location

7.2 Messages

The User Plane messages provide an access to the protocol stacks. Following is a list and short description of significant User Plane messages. Table 6 gives an overview of these messages.

Table 6 – Overview of user messages

Message identifier	Class	Message name	Purpose of message
307	1	UDataReq	Request data transfer on an established user connection
308	1	UDataInd	Indicate arrival of transferred data on an established user connection

7.2.1 UDataReq

Class: 1 (Basic class).

Description: This message allows a PUF to send a data packet. The size of a data packet is limited by the maximum allowed at the ISDN-PCI interface, i.e. 4096 octets.

Parameter:

Name	Required	Comment
NCOID	M	Identifies the user connection

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

Mandatory data shall be provided in the data buffer.

Related: None.

7.2.2 UDataInd

Class: 1 (Basic class).

Description: This message indicates the presence of received data to a PUF. The size of a data packet is limited by the maximum allowed at the ISDN-PCI interface, i.e. 4096 octets.

Parameter:

Name	Provided	Comment
NCOID	M	Identifies the user connection

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

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

Related: None.

7.3 Messages parameters

This subclause describes parameters for the HDLC plane. Table 7 summarizes used parameters. They are alphabetically ordered.

Table 7 – Overview of user parameters

Parameter Identifier	Parameter name	Use in user messages	Use in UAttributeSet	Other use
50	NCOType			X
62	UProtocol		X	
63	UAttributeName			X
64	UDirection			X

7.3.1 NCOType

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

Type: 50.

Field	Field type	Direction	Required	Comment
Identifier	Octet		M	C/U (3) – Signalling and link layer user access

7.3.2 UProtocol

Description: This is used to select the User Plane protocol.

Type: 62.

Field	Field type	Direction	Required	Comment
L3Protocol	Octet	P	M	NULL (4)
L2Protocol	Octet	P	M	HDLC (2)
L1Protocol	Octet	P	O	Default (255) – Transparent B-channel access

Remark: Other possible values (for other protocols) are provided in [3].

7.3.3 UAttributeName

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

Type: 63.

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

7.3.4 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.

Field	Field type	Direction	Required	Comment
Direction	Octet	P	O	both (3)

7.4 State diagram

User messages do not change the state of the connection.

7.5 Coordination function

The coordination function cannot be used with this User Plane protocol.

7.6 Selection criteria

No specific parameters are used. General NCO criteria are provided in [2].

7.7 Specific error handling and codes

Protocol errors are not available at the interface.

7.8 Static attributes

7.8.1 AttributeSet parameters

Table 8 – User plane Attribute Set (UAttributeSet) parameters

Parameter	Required	Comment
Uprotocol	O	See Remark. See also 7.3.2.

Remark: It is only possible to use these parameters during NCO creation containing Control Plane information. Refer to subclause – ACreateNCO operation – in [2] for details.

If parameters are omitted, defaults shall be used by the NAF. Default values are described in Appendix I.

7.8.2 Static attribute content

Name:	U_HDLC
UProtocol:	HDLC

8 HDLC protocol with error

8.1 Introduction

This clause deals with the HDLC protocol with error.

For this access, the NAF considers layer 3 as a Null, as shown in Figure 4.

The OSI location of the HDLC protocol is shown in Figure 4.

General description conventions are provided in [2].

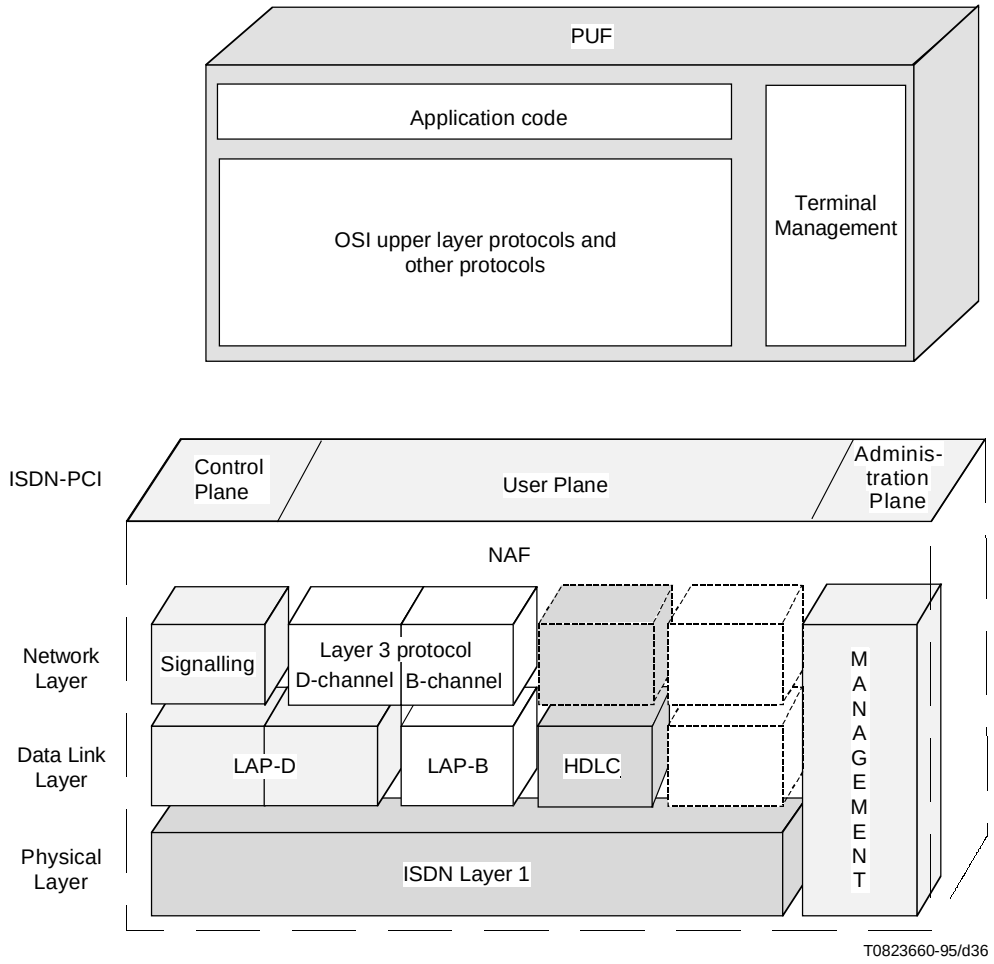


Figure 4 – OSI location

8.2 Messages

The User Plane messages provide an access to the protocol stacks. Following is a list and short description of significant User Plane messages. Table 9 gives an overview of these messages.

Table 9 – Overview of user messages

Message identifier	Class	Message name	Purpose of message
307	1	UDataReq	Request data transfer on an established user connection
308	1	UDataInd	Indicate arrival of transferred data on an established user connection

8.2.1 UDataReq

Class: 1 (Basic class).

Description: This message allows a PUF to send a data packet. The size of a data packet is limited by the maximum allowed at the ISDN-PCI interface, i.e. 4096 octets.

Parameter:

Name	Required	Comment
NCOID	M	Identifies the user connection

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

Mandatory data shall be provided in the data buffer.

Related: None.

8.2.2 UDataInd

Class: 1 (Basic class).

Description: This message indicates the presence of received data to a PUF. The size of a data packet is limited by the maximum allowed at the ISDN-PCI interface, i.e. 4096 octets.

Parameters:

Name	Provided	Comment
NCOID	M	Identifies the user connection
Cause	O	Identifies the type of error

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

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

Related: None.

8.3 Messages parameters

This subclause describes parameters for the HDLC plane. Table 10 summarizes used parameters. They are alphabetically ordered.

Table 10 – Overview of user parameters

Parameter identifier	Parameter name	Use in user messages	Use in UAttributeSet	Other use
50	NCOType			X
62	UProtocol		X	
63	UAttributeName			X
64	UDirection			X
68	Cause	X		

8.3.1 NCOType

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

Type: 50.

Field	Field type	Direction	Required	Comment
Identifier	Octet		M	C/U (3) – Signalling and link layer user access

8.3.2 UProtocol

Description: This is used to select the User Plane protocol.

Type: 62.

Field	Field type	Direction	Required	Comment
L3Protocol	Octet	P	M	NULL (4)
L2Protocol	Octet	P	M	HDLC protocol with error (3)
L1Protocol	Octet	P	O	Default (255) – Transparent access

8.3.3 UAttributeName

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

Type: 63.

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

8.3.4 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.

Field	Field type	Direction	Required	Comment
Direction	Octet	P	O	both (3)

8.3.5 Cause

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

Type: 68.

Field	Field type	Direction	Required	Comment
Value	Octet	B	M	210 – Overflow 211 – Framing error

8.4 State diagram

User messages do not change the state of the connection.

8.5 Coordination function

The coordination function cannot be used with this User Plane protocol.

8.6 Selection criteria

No specific parameters are used. General NCO criteria are provided in [2].

8.7 Specific error handling

In case of invalid length of UDataReq UserData parameter, PUF is sent UDataInd with Cause parameter.

8.8 Static attributes

8.8.1 AttributeSet parameters

Table 11 – User plane Attribute Set (UAttributeSet) parameters

Parameter	Required	Comment
Uprotocol	O	See Remark. See also 8.3.2.

Remark : It is only possible to use these parameters during NCO creation containing Control Plane information. Refer to subclause – ACreateNCO operation – in [2] for details.

If parameters are omitted, defaults shall be used by the NAF. Default values are described in Appendix I.

8.8.2 Static attribute content

Name:	U_HDLC_E
UProtocol:	HDLC

9 PPP protocol

9.1 Introduction

This clause deals with the PPP protocol.

The User Plane provides the services for PPP using the User Plane protocols on a connection on B-channel. For this access, the NAF considers layer 3 as a Null, as shown in Figure 5.

The OSI location of the PPP protocol is shown in Figure 5.

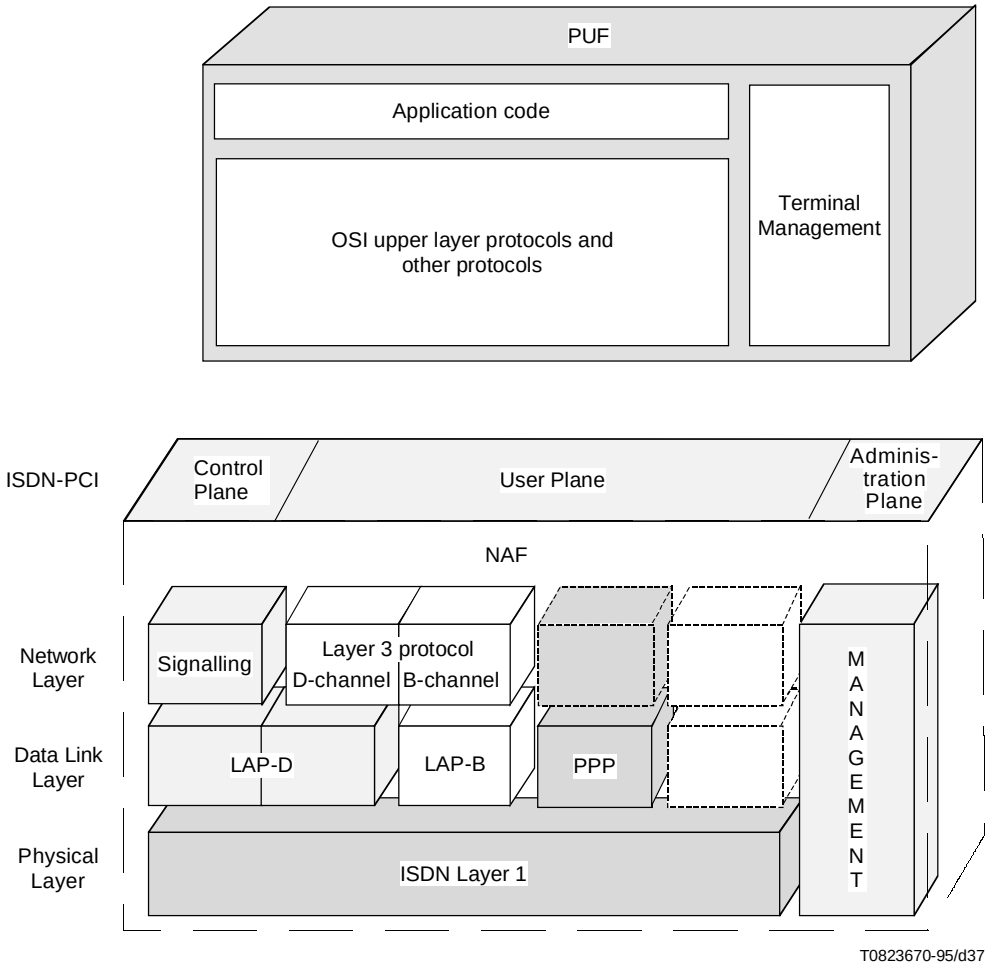


Figure 5 – OSI location

NOTE – PPP is defined as a set of protocols which can be divided into groups:

- Link Control Protocols (LCP) in charge of the establishment, configuration and testing of the data link connection;
- a family of Network Control Protocols (NCP) in charge of the establishment and configuration of the different network layers protocols.

The PPP implementation in the ISDN-PCI applies only on the Link Control Protocols LCP (RFC 1548), PPP Link Quality Monitoring (RFC 1333) and PPP Authentification Protocols (RFC 1334) and RFC 1570 PPP-LCP Extensions (RFC 1570).

The NCPs which handle problems concerning the configuration of network protocols are defined in specific documents. These are not covered by this specification.

General description conventions are provided in [2].

9.2 Messages

The User Plane messages provide an access to PPP protocol stacks. Following is a list and short description of relevant User Plane messages. Table 12 gives an overview of these messages.

Table 12 – Overview of user messages

Message identifier	Class	Message name	Purpose of message
301	1	UConnectReq	Request establishment of a user connection
302	1	UConnectInd	Indicate establishment of a user connection has been requested
303	1	UConnectRsp	Indicate acceptance of user connection establishment
304	1	UConnectCnf	Confirm user connection has been established
305	1	UDisconnectReq	Request removal of user connection
306	1	UDisconnectInd	Indicate removal of user connection
307	1	UDataReq	Request data transfer on an established user connection
308	1	UDataInd	Indicate arrival of transferred data on an established user connection
319	1	UErrorInd	Indicate an error

9.2.1 UConnectReq

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
PPPNegotiation	M	Indicates the requested value

Related: UConnectCnf.

9.2.2 UConnectInd

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
PPPNegotiation	M	Indicates the value proposed for this user connection

Related: UConnectRsp.

9.2.3 UConnectRsp

Class: 1 (Basic class).

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

Parameter:

Name	Required	Comment
NCOID	M	Identifies the user connection

Related: UConnectInd.

9.2.4 UConnectCnf

Class: 1 (Basic class).

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

Parameter:

Name	Provided	Comment
NCOID	M	Identifies the user connection

Related: UConnectReq.

9.2.5 UDisconnectReq

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
PPPCause	O	PPP reason to remove the user connection

Related: None.

9.2.6 UDisconnectInd

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
PPPOrigin	M	Identifies the initiator of the user connection removal
PPPCause	O	PPP reason to remove the user connection
PPPDiagnostic	C	Complementary information for PPPCause. Optional if PPPCause parameter supplied

Related: None.

9.2.7 UDataReq

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.

Parameter:

Name	Required	Comment
NCOID	M	Identifies the user connection

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

Mandatory data shall be provided in the data buffer.

Address field is set to "11111111" (All-Station address) and control field is set to "00000011" (Unnumbered Information) with bit P/F set to zero. The FCS is inserted at the end of each data block with the flag, transparently by the NAF.

Related: None.

9.2.8 UDataInd

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.

Parameter:

Name	Provided	Comment
NCOID	M	Identifies the user connection

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

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

Address field is set to "11111111" (All-Station address) and control field is set to "00000011" (Unnumbered Information) with bit P/F set to zero. The FCS is inserted at the end of each data block with the flag, transparently by the NAF.

Related: None.

9.2.9 UErrorInd

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
PPPCause	M	Identifies type of error

Related: None.

9.3 Messages parameters

This subclause describes parameters for the PPP plane. Table 13 summarizes these parameters.

Table 13 – Overview of user parameters

Parameter identifier	Parameter name	Use in user messages	Use in UAttributeSet	Other use
50	NCOType			X
62	UProtocol		X	
63	UAttributeName			X
64	UDirection			X
69	PPPOrigin	X		
70	PPPCause	X		
71	PPPDiagnostic	X		
74	PPPNegotiation	X	X	

9.3.1 NCOType

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

Type: 50.

Field	Field type	Direction	Required	Comment
Identifier	Octet		M	C/U (3) – Signalling and network layer user access

9.3.2 UProtocol

Description: This is used to select the User Plane protocol.

Type: 62.

Field	Field type	Direction	Required	Comment
L3Protocol	Octet	P	M	NULL (4)
L2Protocol	Octet	P	M	PPP (4)
L1Protocol	Octet	P	O	Default (255) – Transparent B-channel access

Remark: Other possible values (for other protocols) are provided in [3].

9.3.3 UAttributeName

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

Type: 63.

Field	Field type	Direction	Required	Comment
AttributeName	IA5-string	P	M	16 bytes is the maximum length

9.3.4 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.

Field	Field type	Direction	Required	Comment
Direction	Octet	P	O	both (3)

9.3.5 PPPCause

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

Type: 70.

Field	Field type	Direction	Required	Comment
Value	Octet	B	M	Value provided in Table 14

9.3.6 PPPDiag

Description: This parameter is used to pass PPP Diagnostic information associated to a PPP Cause.

Type: 71.

Field	Field type	Direction	Required	Comment
DiagType	Octet	N	M	Indicate the type of diagnostic associated with the PPPCause ConfNoConverging (0)
NoConvergingDiag	Octet-string	N	M	Diagnostic associated with ConfNoConverging (Note)
NOTE – These elements are ordered in the same way that they have been defined in the PPPNegotiation message (see PPPNegotiation). Furthermore, the bits corresponding to the non-acknowledged options are set and those corresponding to the acknowledged options are reset.				

9.3.7 PPPNegotiation

Description: This parameter is used to indicate the PPP negotiation to perform.

Type: 74.

Field		Field type	Direction	Required	Comment
PPPNegiotiation	Usage	Octet-string	B	M	Indicates if the following values are included Length is fixed to 2 octets (Note 1)
	MRUlocal	Octet-string	B	C	Indicates the size of the Maximum Receive Unit of the local peer Default (0) Length is fixed to 2 (Note 2)
	MRUremote	Octet-string	N	C	Indicates the size of the Maximum Receive Unit of the remote peer Length is fixed to 2 octets (Note 2)
	Authentproto	Octet	B	C	Indicates the type of the authentication to perform. These values are exclusive Default (0) PAP (1) CHAP (2) (Note 2)
	Qualityproto	Octet-string	B	C	Indicates the value of the reporting period for quality protocol Default (0) Length is fixed to 4 octets (Note 2)
	Magicnumber	Octet-string	B	C	Indicates the value of the magic number to use Default (0) Length is fixed to 4 (Note 2)
	Protocolcomp	Octet	B	C	Indicates if the protocol field compression is to be set (Note 2)
	Addresscomp	Octet	B	C	Indicates if the address field compression is to be set (Note 2)
	FCSAlternatives	Octet-string	B	C	Indicates the value of the FCS format to use Default (0) Length is fixed to 4 octets (Note 2)

Field		Field type	Direction	Required	Comment
PPPNegiotiation *(cont.)*	SelfDescPadding	Octet-string	B	C	Indicates the value of the Self Describing Padding to use Default (0) Length is fixed to 4 octets
	CallBack	Octet	B	C	Indicates if the callback option is to be set (Note 2)
	CompoundFrame	Octet	B	C	Indicates if the CompoundFrame option is to be set (Note 2)
NOTE 1 – Each bit of the indicator corresponds to an option ordered as indicated in the array (that means the first bit refers to the MRUlocal option, the second to the MRUremote option, the third to Authentproto option and so on). When the indicator concerning an option is not set, it means that the PPP option has not to be negotiated. NOTE 2 – Before defining a negotiation parameter, the PUF has to check if the functionality is served by the NAF by using the PciGetProperty. See Part 2 [2].					

9.3.8 PPPOrigin

Description: This parameter is used to pass PPP origin information to/from the PUF.

Type: 69.

Field	Field type	Direction	Required	Comment
Value	Octet	B	M	undefined (1) NAF Provider (2) PUF User (3)

9.4 State diagram

Figure 6 shows the different states a user connection can get, using the U-messages, and in which order these messages shall be used.

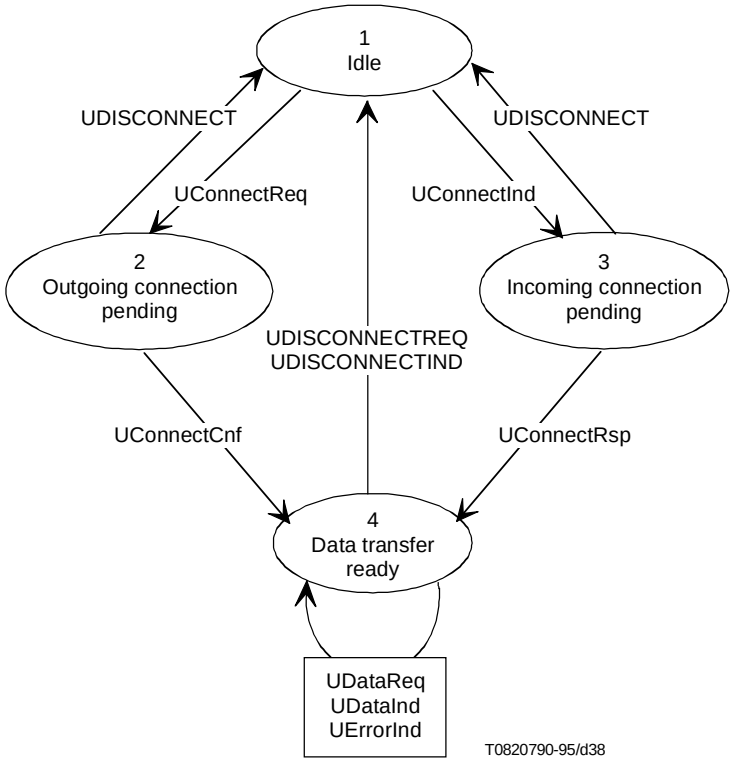


Figure 6 – Overview of the User Plane messages

9.5 Coordination function

The coordination function cannot be used with the User Plane protocol relating to PPP.

9.6 Selection criteria

No PPP specific parameters are used. General NCO criteria are provided in [2].

9.7 Specific error handling and codes

Errors are dealt with in the following manner:

9.7.1 Errors

In case of protocol reject information from the remote part, PUF is sent UErrorInd.

In case of invalid length of UDataReq UserData parameter, data are ignored.

9.7.2 Causes

These values can be specified and are returned in the PPPCause parameter.

Table 14 – PPPCause parameter value

Return code		Meaning	ErrorSpecific information
Undefined	220	Undefined error situation	Not present
DiscNorm	241	Disconnection – Normal condition	Not present
ConfNoConverging	244	Connection rejected – Host no responding (transient condition)	Not present
Hostunreachable	245	Connection rejected – Configurations cannot match (fixed condition)	Not present
Protocol Error	212	Protocol Error	Not present

9.8 Static attributes

9.8.1 AttributeSet parameters

Table 15 – User plane Attribute Set (UAttributeSet) parameters

Parameter	Required	Comment
UProtocol	O	See Remark. See also 9.3.2.
MRUlocal	O	See Remark. See also 9.3.7 (PPPNegotiation).
MRUremote	O	See Remark. See also 9.3.7.
Authentproto	O	See Remark. See also 9.3.7.
Qualityproto	O	See Remark. See also 9.3.7.
MagicNumber	O	See Remark. See also 9.3.7.
Protocolcomp	O	See Remark. See also 9.3.7.
Addresscomp	O	See Remark. See also 9.3.7.
FCSAlternatives	O	See Remark. See also 9.3.7.
SelfDescPadding	O	See Remark. See also 9.3.7.
CallBack	O	See Remark. See also 9.3.7.
CompoundFrame	O	See Remark. See also 9.3.7.

Remark: It is only possible to use these parameters during NCO creation containing Control Plane information. Refer to subclause – ACreateNCO operation – in [2] for details.

If parameters are omitted, defaults shall be used by the NAF. Default values are described in Appendix I.

9.8.2 Static attribute content

Name:	U_PPP
UProtocol:	PPP
MRUlocal:	1500
MRUremote:	1500
Authentproto:	1500
Qualityproto:	None
MagicNumber:	None
Protocolcomp:	None
AddressComp	None
FCSAlternatives:	None
SelfDescPadding:	None
CallBack:	None
CompoundFrame:	None

9.9 Protocol specific NAF property information

The PPP specific parameters of NAF-Property are shown in Table 16.

See also the PciGetProperty function in Part 2 (Basic Services).

Table 16 – TLV coded NAF-Property parameter

Parameter	Provided	TLV coding			Comment and value
		TypeID	Length	Value	
PPPNegotiation	M	14	2.27	Octet	Indicates the PPP options provided by the NAF

10 SDLC protocol

10.1 Introduction

This clause deals with the SDLC protocol.

The User Plane provides the services for SDLC using the User Plane protocols on a connection on B-channel. For this access, the NAF considers layer 3 as a Null, as shown in Figure 7.

SDLC protocol supported is an SDLC link in normal response mode having a point-to-point configuration. Overview and protocol information is provided in IBM publication "IBM Synchronous Data Link Control Concepts" (GA27-3093).

The OSI location of the SDLC protocol is shown in Figure 7.

General description conventions are provided in [2].

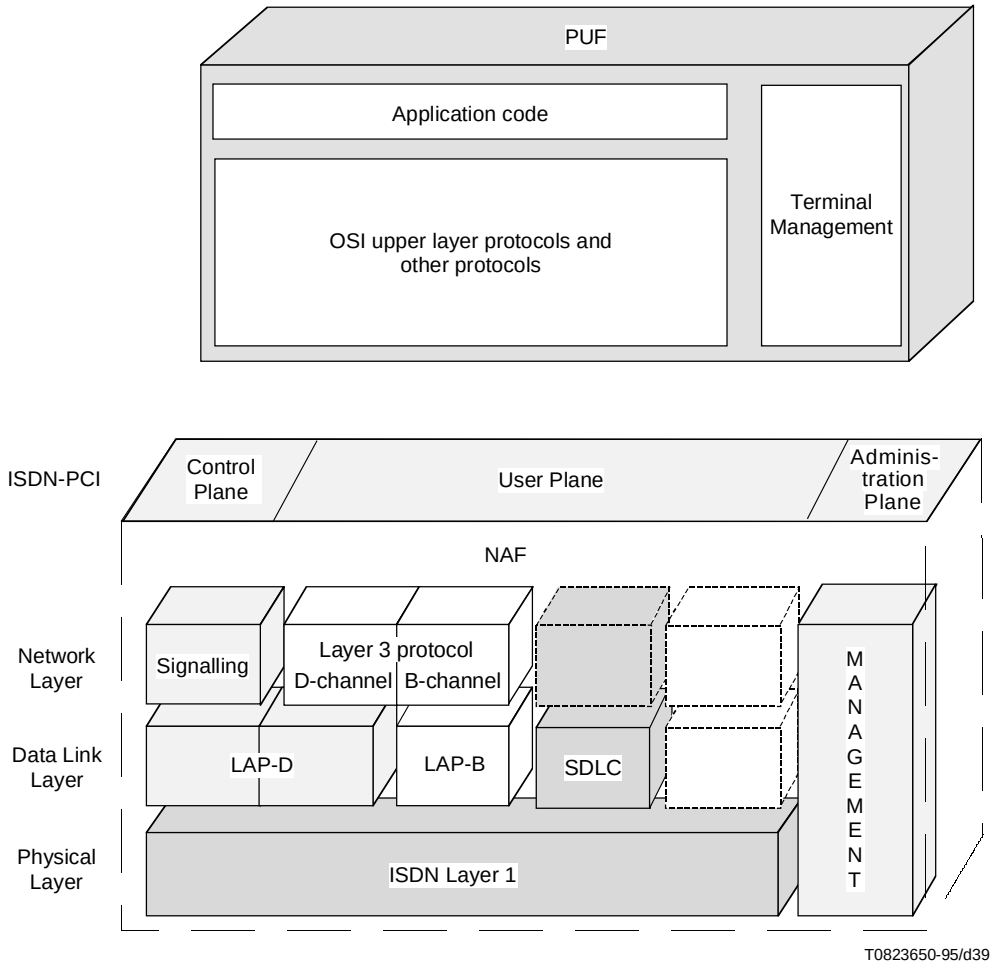


Figure 7 – OSI location

10.2 Messages

The User Plane messages provide an access to SDLC protocol stacks. Following is a list and short description of relevant User Plane messages. Table 17 gives an overview of these messages.

Table 17 – Overview of user messages

Message identifier	Class	Message name	Purpose of message
301	1	UConnectReq	Request establishment of a user connection
302	1	UConnectInd	Indicate establishment of a user connection has been requested
303	1	UConnectRsp	Indicate acceptance of user connection establishment
304	1	UConnectCnf	Confirm user connection has been established
305	1	UDisconnectReq	Request removal of user connection
306	1	UDisconnectInd	Indicate removal of user connection
307	1	UDataReq	Request data transfer on an established user connection
308	1	UDataInd	Indicate arrival of transferred data on an established user connection
309	1	UExpeditedDataReq	Request expedited data transfer on an established user connection
310	1	UExpeditedDataInd	Indicate presence of transferred expedited data on an established user connection
317	1	UReadyToReceiveReq	Used to perform flow control for a user connection
318	1	UReadyToReceiveInd	Used to indicate flow control status on a user connection

10.2.1 UConnectReq

Class: 1 (Basic class).

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

Parameter:

Name	Required	Comment
NCOID	M	Identifies the user connection

Related: UConnectCnf.

10.2.2 UConnectInd

Class: 1 (Basic class).

Description: This message informs a PUF of an incoming demand to establish a user connection. This message informs the PUF of the end of transient idle state of the user connection caused by Data Link Layer resetting.

Parameter:

Name	Provided	Comment
NCOID	M	Identifies the user connection

Related: UConnectRsp

10.2.3 UConnectRsp

Class: 1 (Basic class).

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

Parameter:

Name	Provided	Comment
NCOID	M	Identifies the user connection

Related: UConnectInd.

10.2.4 UConnectCnf

Class: 1 (Basic class).

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

Parameter:

Name	Provided	Comment
NCOID	M	Identifies the user connection

Related: UConnectReq.

10.2.5 UDisconnectReq

Class: 1 (Basic class).

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

Parameter:

Name	Provided	Comment
NCOID	M	Identifies the user connection

Related: None.

10.2.6 UDisconnectInd

Class: 1 (Basic class).

Description: This message informs a PUF that a user connection has been removed. This message may inform the PUF of transient idle state of the user connection caused by Data Link Layer reset. In this case, the SDLCCause parameter value is DiscTrans (disconnection – transient condition) i.e. 225 in decimal.

Parameters:

Name	Provided	Comment
NCOID	M	Identifies the user connection
SDLCOrigin	M	Identifies the initiator of the user connection removal
SDLCCause	C	SDLC reason to remove the user connection

Related: None.

10.2.7 UDataReq

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. No fragmentation mechanism is available at the SDLC link layer level.

Parameter:

Name	Required	Comment
NCOID	M	Identifies the user connection

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

Mandatory data shall be provided in the data buffer.

Related: None.

10.2.8 UDataInd

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. No fragmentation mechanism is available at the SDLC link layer level.

Parameter:

Name	Provided	Comment
NCOID	M	Identifies the user connection

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

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

Related: None.

10.2.9 UExpeditedDataReq

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 UDataReq messages. SDLC expedited data is transmitted in an Unnumbered Information frame.

Parameter:

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

Related: None.

10.2.10 UExpeditedDataInd

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 UDataInd messages. SDLC expedited data are received in an Unnumbered Information frame.

Parameters:

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

Related: None.

10.2.11 UReadyToReceiveReq

Class: 1 (Basic class).

Description: This message allows the PUF to indicate to the NAF if it can accept incoming data (UDataInd 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.

Related: UDataInd.

10.2.12 UReadyToReceiveInd

Class: 1 (Basic class).

Description: This message allows the NAF to indicate to the PUF if the user connection permits the sending of data (UDataReq messages). This message can only apply to an already established user connection. If the ReadyFlag parameter value is FALSE, the NAF can not 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

Related: UDataReq.

10.3 Messages parameters

This subclause describes parameters for the SDLC plane. Table 18 summarizes used parameters.

Table 18 – Overview of user parameters

Parameter identifier	Parameter name	Use in user messages	Use in UAttributeSet	Other use
38	L2ConnectionMode		X	
39	L2FrameSize		X	
40	L2WindowSize		X	
41	L2XID		X	
50	NCOType			X
55	ReadyFlag	X		
62	UProtocol		X	
63	UAttributeName			X
64	UDirection			X
65	UserData	X		
68	SDLCCause	X		
69	SDLCOrigin	X		

10.3.1 L2ConnectionMode

Description: This parameter is used only if it is not defined in L2XID value field. It is used to pass details of the layer connection mode to the NAF.

Type: 38.

Field	Field type	Direction	Required	Comment
Value	Octet	P	M	dte (1) – Act as secondary link station (non-negotiable) dce (2) – Act as primary link station (non negotiable) auto (3) – Link station role is negotiable by XID exchange

10.3.2 L2FrameSize

Description: This parameter is used only if it is not defined in L2XID value field. It is used to pass details of the layer 2 frame size to the NAF.

Type: 39.

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

10.3.3 L2WindowSize

Description: This parameter is used only if it is not defined in L2XID value field. It is used to pass details of the layer 2 window size to the NAF.

Type: 40.

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

10.3.4 L2XID

Description: This is used to pass details of the layer 2 XID value and its use. XID information field may include values that override some parameters defined elsewhere. DLC-XID information field formats are described in IBM publication "Systems Network Architecture – Formats" (GA27-3136-11).

Type: 41.

Field	Field type	Direction	Required	Comment
Use	Octet	P	M	Not relevant for SDLC protocol
Value	Octet-string	P	M	XID value (Identifier and signature) Formatted DLC-XID information field for SDLC protocol. Maximum length is 127.

10.3.5 NCOType

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

Type: 50.

Field	Field type	Direction	Required	Comment
Identifier	Octet		M	C/U (3) – Signalling and network layer user access

Remark : An SDLC connection can only be defined by a C/U type NCO. No U3/G type NCO can be grouped to an NCO defining an SDLC connection.

10.3.6 ReadyFlag

Description: This parameter is used to request and indicate flow control status on a user connection.

Type: 55.

Field	Field type	Direction	Required	Comment
Usage	Boolean	B	M	TRUE – Data transfer is allowed FALSE – Data transfer is not allowed

10.3.7 UProtocol

Description: This is used to select the User Plane protocol.

Type: 62.

Field	Field type	Direction	Required	Comment
L3Protocol	Octet	P	M	NULL (4)
L2Protocol	Octet	P	M	SDLC (5)
L1Protocol	Octet	P	O	Default (255) – Transparent B-channel access

Remark: Other possible values (for other protocols) are provided in [3].

10.3.8 UAttributeName

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

Type: 63.

Field	Field type	Direction	Required	Comment
AttributeName	IA5-string	P	M	16 bytes is the maximum length

10.3.9 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.

Field	Field type	Direction	Required	Comment
Direction	Octet	P	O	both (3)

10.3.10 UserData

Description: This parameter is used to pass Data that is limited in size to/from the PUF.

Type: 65.

Field	Field type	Direction	Required	Comment
Data	Octet-string	B	M	128 octets is the maximum size

10.3.11 SDLCCause

Description: This parameter is used to pass SDLC Cause information to/from the PUF.

Type: 68.

Field	Field type	Direction	Required	Comment
Value	Octet	B	M	See User Plane return code values in 10.7.2

10.3.12 SDLCOrigin

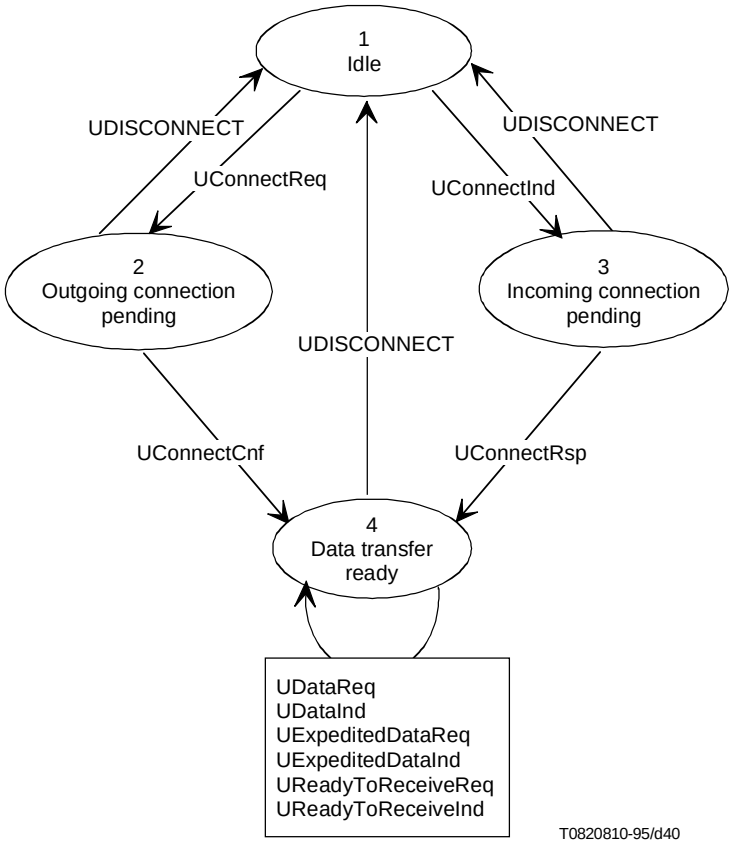
Description: This parameter is used to pass SDLC origin information to/from the PUF.

Type: 69.

Field	Field type	Direction	Required	Comment
Value	Octet	B	M	Undefined (1) NAF Provider (2) PUF User (3)

10.4 State diagram

Figure 8 shows the different states a user connection can get, using the U-messages, and in which order these messages shall be used.



NOTE – Where UDISCONNECT appears, it can be either UDisconnectReq or UDisconnectInd.

Figure 8 – Overview of the User Plane messages

10.5 Coordination function

The coordination function cannot be used with the User Plane protocol relating to SDLC.

10.6 Selection criteria

No SDLC specific parameters are used. General NCO criteria are provided in [2].

10.7 Specific error handling and codes

Errors are dealt with in the following manner.

10.7.1 Invalid use of user messages

In case of:

– Invalid length of UDataReq UserData parameter;

– Invalid use of ExpeditedData,

action is:

– PUF is sent UDisconnectInd.

10.7.2 Causes

These values are specified in Table 19 and are returned in the SDLCCause parameter.

Table 19 – SDLCCause parameter value

Return code		Meaning	ErrorSpecific information
Undefined	220	Undefined error situation	Not present
DiscTrans	225	Disconnection – Transient condition Indicates that Data Link layer is resetting	Not present
DiscPerm	226	Disconnection – Permanent condition Indicates that the remote station is no longer reachable	Not present
DiscNorm	241	Disconnection – Normal condition Indicates that the disconnection has been requested by the remote station	Not present
ConRejectTrans	244	Connection rejection – Transient condition Indicates that Data Link layer activation has been denied by the remote station	Not present
ConRejectPerm	245	Connection rejection – Fixed condition Indicates that the remote station is unreachable	Not present

10.8 Static attributes

10.8.1 AttributeSet parameters

Table 20 – User Plane Attribute Set (UAttributeSet) parameters

Parameter	Required	Comment
UProtocol	O	See Remark. See also 10.3.7.
L2ConnectionMode	O	See Remark. See also 10.3.1.
L2WindowSize	O	See Remark. See also 10.3.3.
L2FrameSize	O	See Remark. See also 10.3.2.
L2XID	O	See Remark. See also 10.3.4.

Remark: It is only possible to use these parameters during NCO creation containing Control Plane information. Refer to subclause – ACreateNCO operation – in [2] for details.

If parameters are omitted, defaults shall be used by the NAF. Default values are described in Appendix I.

L2ConnectionMode, L2WindowSize and L2FrameSize parameters may contain user defined values or default configuration values when not provided by the NAF, or may contain values resulting of XID exchange negotiation. L2XID parameter contains the DLC-XID information field.

10.8.2 Static attribute content

Name:	U_SDLC
UProtocol:	SDLC
L2ConnectionMode:	dte
L2WindowSize:	7
L2FrameSize:	265
L2XID:	Not use

11 V.110 protocol

11.1 Introduction

This clause deals with the V.110 protocol. This protocol allows a PUF to request a NAF for a B-channel running the V.110 protocol. Both synchronous and asynchronous options of V.110 are allowed.

This protocol uses NULL layer 3 protocol as shown in Figure 9.

The OSI location of the V.110 protocol is shown in Figure 9.

General description conventions are provided in [2].

A V.110 negotiation may occur via the parameter "BearerCap". In this case, octets 5a, 5b, 5c and 5d of the parameter may be concerned (see [2]).

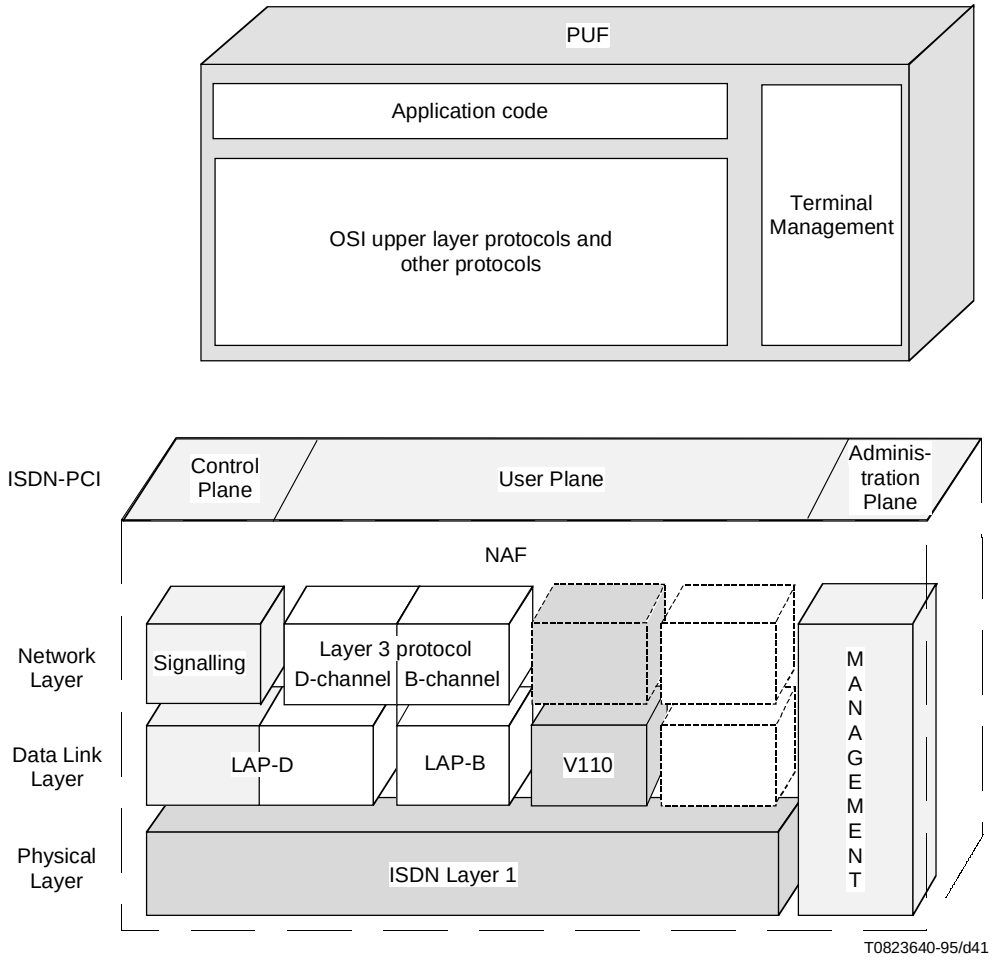


Figure 9 – OSI location

11.2 Messages

The User Plane messages provide an access to V.110 protocol stacks. Following is a list and short description of relevant User Plane messages. Table 21 gives an overview of these messages.

Table 21 – Overview of user messages

Message Identifier	Class	Message name	Purpose of message
301	1	UConnectReq	Request establishment of a user connection
302	1	UConnectInd	Indicate establishment of a user connection has been requested
303	1	UConnectRsp	Indicate acceptance of user connection establishment
304	1	UConnectCnf	Confirm user connection has been established
305	1	UDisconnectReq	Request removal of user connection
306	1	UDisconnectInd	Indicate removal of user connection
307	1	UDataReq	Request data transfer on an established user connection
308	1	UDataInd	Indicate arrival of transferred data on an established user connection
317	1	UReadyToReceiveReq	Used to perform flow control for a user connection
318	1	UReadyToReceiveInd	Used to indicate flow control status on a user connection

11.2.1 UConnectReq

Class: 1 (Basic class).

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

Parameter:

Name	Required	Comment
NCOID	M	Identifies the user connection

Related: UConnectCnf.

11.2.2 UConnectInd

Class: 1 (Basic class).

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

Parameter:

Name	Provided	Comment
NCOID	M	Identifies the user connection

Related: UConnectRsp.

11.2.3 UConnectRsp

Class: 1 (Basic class).

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

Parameter:

Name	Provided	Comment
NCOID	M	Identifies the user connection

Related: UConnectInd.

11.2.4 UConnectCnf

Class: 1 (Basic class).

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

Parameter:

Name	Provided	Comment
NCOID	M	Identifies the user connection

Related: UConnectReq.

11.2.5 UDisconnectReq

Class: 1 (Basic class).

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

Parameter:

Name	Provided	Comment
NCOID	M	Identifies the user connection

Related: None.

11.2.6 UDisconnectInd

Class: 1 (Basic class).

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

Parameter:

Name	Provided	Comment
NCOID	M	Identifies the user connection
V.110Origin	M	Identifies the initiator of the user connection removal
V.110Cause	C	V.110 reason to remove the user connection

Related: None.

11.2.7 UDataReq

Class: 1 (Basic class).

Description: This message allows a PUF to send a data packet. The size of a data packet is limited by the maximum allowed at the ISDN-PCI interface, i.e. 4096 octets.

Parameter:

Name	Required	Comment
NCOID	M	Identifies the user connection

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

Related: None.

11.2.8 UDataInd

Class: 1 (Basic class).

Description: This message indicates the presence of received data to a PUF. The size of a data packet is limited by the maximum allowed at the ISDN-PCI interface, i.e. 4096 octets.

Parameter:

Name	Provided	Comment
NCOID	M	Identifies the user connection

Remark: Data received are always provided, but not as a parameter of the message. Data are provided in the data buffer. This buffer, in this case, is mandatory.

Related: None.

11.2.9 UReadyToReceiveReq

Class: 1 (Basic class).

Description: This message allows the PUF to indicate to the NAF if it can accept incoming data (UDataInd 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.

Related: UDataInd.

11.2.10 UReadyToReceiveInd

Class: 1 (Basic class).

Description: This message allows the NAF to indicate to the PUF if the user connection permits the sending of data (UDataReq messages). This message can only apply to an already established user connection. If the ReadyFlag parameter value is FALSE, the NAF can not 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

Related: UDataReq.

11.3 Messages parameters

This subclause describes parameters for the V.110 plane. Table 22 summarizes used parameters.

Table 22 – Overview of user parameters

Parameter Identifier	Parameter Name	Used in user messages	Used in UAttributeSet	Other use
50	NCOType			X
55	ReadyFlag	X		
62	UProtocol		X	
63	UAttributeName			X
64	UDirection			X
68	V.110Cause	X		
69	V.110Origin	X		
75	FlowControlMechanism		X	
76	FlowControlCharacters		X	
77	MomentNumber		X	
78	V.110BChannelDisconnection		X	

11.3.1 NCOType

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

Type: 50.

Field	Field type	Direction	Required	Comment
Identifier	Octet		M	C/U (3) – Signalling and transparent user access

11.3.2 ReadyFlag

Description: This parameter is used to request and indicate flow control status on a user connection.

Type: 55.

Field	Field type	Direction	Required	Comment
Usage	Boolean	B	M	TRUE – Data transfer is allowed FALSE – Data transfer is not allowed

11.3.3 UProtocol

Description: This is used to select the User Plane protocol. If the length is 3, the first octet contains the layer 3 protocol requested, the second octet contains the layer 2 protocol requested and the third octet contains the layer 1 protocol requested.

Type: 62.

Field	Field type	Direction	Required	Comment
L3Protocol	Octet	P	M	NULL (4)
L2Protocol	Octet	P	M	V.110 asynchronous (6) V.110 synchronous (7)
L1Protocol	Octet	P	O	Default (255) – Transparent B-channel access

Remark: Other possible values (for other protocols) are provided in [3].

11.3.4 UAttributeName

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

Type: 63.

Field	Field type	Direction	Required	Comment
AttributeName	IA5-string	P	M	16 bytes is the maximum length

11.3.5 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.

Field	Field type	Direction	Required	Comment
Direction	Octet	P	O	both (3)

11.3.6 V.110Cause

Description: This parameter is used to pass V.110 Cause information to/from PUF.

Type: 67.

Field	Field type	Direction	Required	Comment
Value	Octet	B	M	See values in Table 23

11.3.7 V.110Origin

Description: This parameter is used to pass V.110 origin information to/from PUF.

Type: 68.

Field	Field type	Direction	Required	Comment
Value	Octet	B	M	Undefined (1) NAF Provider (2) PUF User (3)

11.3.8 FlowControlMechanism

Description: This parameter is used to negotiate the flow control mechanism for a V.110 connection. Two possibilities exist: first is the XON/XOFF characters, second is via V.24 105/106 signals. This parameter is used for end-to-end negotiation.

Type: 75.

Field	Field type	Direction	Required	Comment
Value	Octet	P	M	Type of mechanism to use: – 0 XON/XOFF characters; – 1 105/106 signals. Default value is 0 (XON/XOFF).

11.3.9 FlowControlCharacters

Description: This parameter is used to set the characters to define flow control characters for a V.110 connection. The characters may be different for each direction, so two characters are mandatory to be provided, even if they have the same value. This parameter has only a local meaning.

Type: 76.

Field	Field type	Direction	Required	Comment
Value	Octet String	P	M	Fixed length is 2. The first character identifies the XON character. Default value is 16. The second character identifies the XOFF character. Default value is 18.

11.3.10 MomentNumber

Description: This parameter is used to set the number of moments for a V.110 connection. It has only a local meaning.

Type: 77.

Field	Field type	Direction	Required	Comment
Value	Octet	P	M	Number of moments

11.3.11 V.110BChannelDisconnection

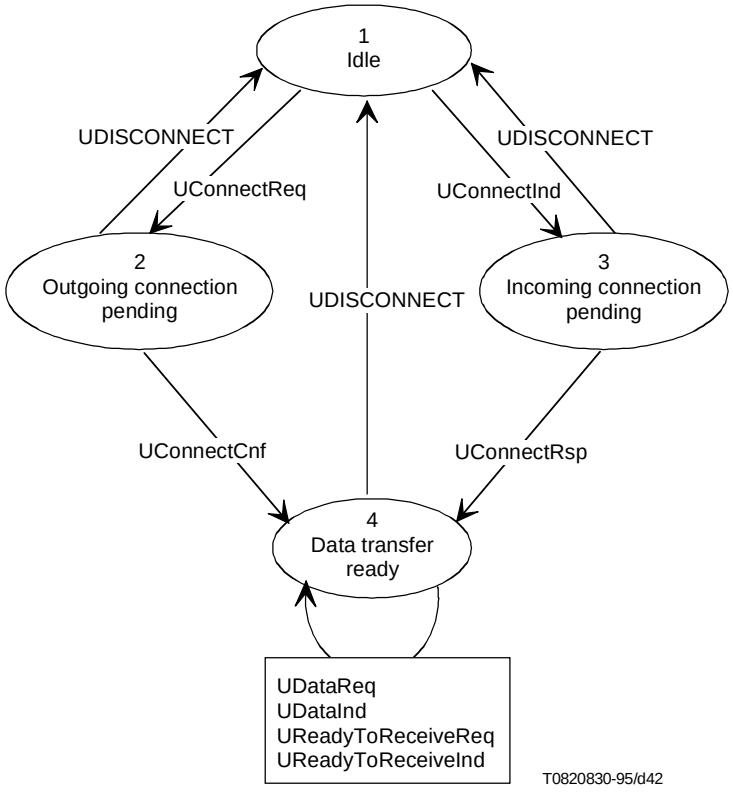
Description : A V.110 disconnection may imply the B-channel disconnection. This parameter is used to set this information. It has only a local meaning.

Type: 78.

Field	Field type	Direction	Required	Comment
Value	Octet	P	M	V.110 implies B-channel disconnection: – 0 No disconnection; – 1 Disconnection.

11.4 State diagram

Figure 10 shows the different states a user connection can get, using the U-messages, and in which order these messages shall be used.



NOTE – Where UDISCONNECT appears, it can be either UDisconnectReq or UDisconnectInd.

Figure 10 – Overview of the User Plane messages

11.5 Coordination function

The coordination function cannot be used with the User Plane protocol relating to V.110 access.

11.6 Selection criteria

No specific parameters are to be used. General NCO criteria are provided in [2].

11.7 Specific error handling and codes

Errors are dealt with in the following manner.

11.7.1 Invalid use of User Plane messages

In case of invalid length of UDataReq UserData parameter, PUF is sent UDisconnectInd.

11.7.2 Causes

These values can be specified and are returned in the V.110Cause parameter.

Table 23 – V.110Cause parameter value

Return code		Meaning	ErrorSpecific information
Undefined	220	Undefined error situation.	Not present
DiscNorm	241	Disconnection – Normal condition.	Not present
ConRejectTrans	244	Connection rejected (transient condition)	Not present
ConRejectPerm	245	Connection rejected (permanent condition)	Not present

11.8 Static attributes

11.8.1 AttributeSet parameters

Table 24 – User Plane Attribute Set (UAttributeSet) parameters

Parameter	Required	Comment
UProtocol	O	See 11.3.3
FlowControlMechanism	O	See 11.3.8
FlowControlCharacters	O	See 11.3.9
MomentNumber	O	See 11.3.10
V.110BChannelDisconnection	O	See 11.3.11

Remark: It is only possible to use these parameters during an NCO creation containing Control Plane information. Refer to subclause – ACreateNCO operation – in [2] for details.

If parameters are omitted, defaults shall be used. The default values described in Appendix I.

11.8.2 Static attribute content

Name:	U_V.110
UProtocol:	V.110a
FlowControlMechanism:	0
FlowControlCharacters:	17 19
V.110BChannelDisconnection:	0

Appendix I

Configuration

The following subclauses give the default values parameters to use if they are absent from the parameter list during the NCO creation operation.

I.1 ISO/IEC 7776 protocol

Table I.1 – User Plane ISO/IEC 7776 configuration

Parameter	Suggested default	Comment
L2FrameSize	128	
L2WindowSize	7	
L2ConnectionMode	Auto	
L2XID		Not used

1.2 PPP protocol

Table I.2 – User Plane PPP configuration

Parameter	Suggested default	Comment
Link Control Protocol Parameters		
– Maximum-Receive-Unit	1500 (bytes)	Enables a peer to inform the maximum packet size accepted in reception
– Restart timeout	3 (second)	Waiting time for a response to a request packet
– Max terminate	2	Counter for number of terminate requests sent without response
– Max configure	10	Counter for number of configure requests sent without response
– Max failure	10	Counter for number of Configure-Nak received sent before sending a Configure-Reject assuming that the configuration is not converging
– Magic number	None	
– Protocol compression	None	
– Address and Control field Compression	None	

Table I.2 – User Plane PPP configuration *(cont)*

Parameter	Suggested default	Comment
Authentification Protocol – Type of authentification protocol to be set – Local-ID – Local password – List of the remote of couple peers "ID/Password" – Algorithm *Line Quality Monitoring* – Reporting period	None None	 Enables use of the PPP Authentification Protocols Length and name of the local Peer ID Length and name of the local Password List of length and name of the remote couple of "ID/Password" Type of CHAP algorithm used Value of maximum time of sending informations
FCSAlternatives		Indicates the value of the FCS format to use
SelfDescPadding	None	Indicates the value of the Self Describing Padding to use
Callback – Use of the callback – Number to call back	 None	 Indicates if the callback option is to be set
CompoundFrame	None	Indicates if the CompoundFrame option is to be set

I.3 SDLC protocol

Table I.3 – User Plane SDLC configuration

Parameter	Suggested default	Comment
SDLC connection mode	Case 1	SDLC default link station role Case 1 – Secondary link station Case 2 – Primary link station
SDLC initialization mode	Case 1	SDLC default initialization mode Case 1 – Send / answer to SNRM Case 2 – Send RIM/SIM
SDLC address	0xC1	SDLC default station address
SDLC modulus	8	SDLC default frame numbering modulus
SDLC window size	7	SDLC default window size
SDLC frame size	265	SDLC default maximum frame size excluding the link header and the link trailer
SDLC timers – T1 – T2 – N2	 2 1 10	SDLC protocol timers Expressed in second Expressed in second Maximum number of unsuccessful retransmission

I.4 V.110 protocol

Table I.4 – User Plane V.110 configuration

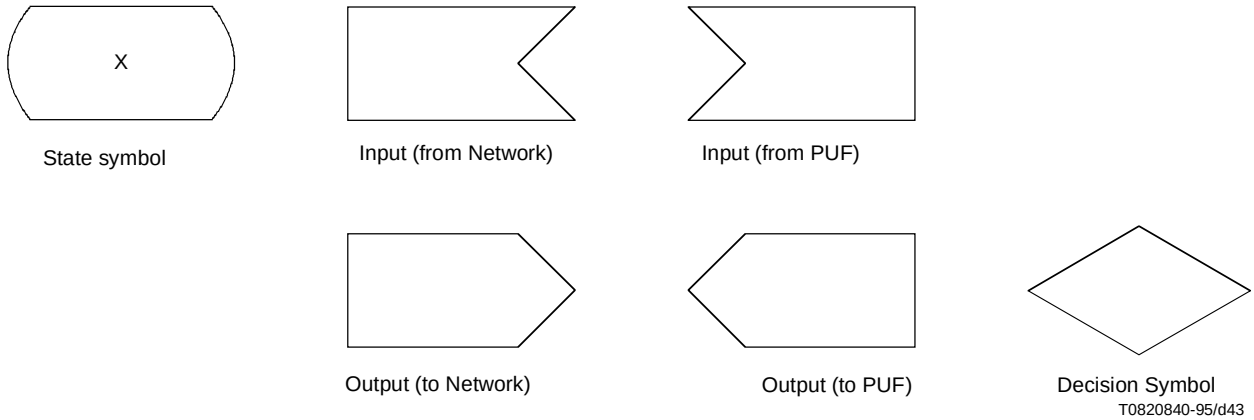
Parameter	Suggested default	Comment
Out of synchronization timer	3 s	Maximum time for resynchronization
Synchronization timer	10 s	Maximum time for Synchronization

Appendix II

NAF-SDL diagrams

The mapping of User Plane messages to protocol messages is provided in the following tables. Some SDL diagrams, or other kind of schemes, are given to explain more clearly the relation between user messages and network primitives. These diagrams do not cover every case. They only present some of the possible cases.

The following symbols are used within this description. A full description of the symbols and their meaning is given in Recommendation Z.100.



II.1 ISO/IEC 7776 protocol

Table II.1 shows the mapping of User Plane messages to service primitives.

Table II.1 – Mapping between User Plane message and protocol messages

ISDN-PCI	ISO/IEC 7776
UConnectReq	Send SABM(E)
UConnectInd	Received SABM(E)
UConnectRsp	Send UA
UConnectCnf	Received UA
UDisconnectReq	Send DISC
UDisconnectInd	Received DISC or FRMR
UDataReq	Send I frame
UDataInd	Received I frame
UReadyToReceiveReq (busy)	Send RNR
UReadyToReceiveReq (free)	Send RR
UReadyToReceiveInd (busy)	Received RNR
UReadyToReceiveInd (free)	Received RR

NOTE – REJ frames and frames numbering are handled transparently by the NAF.

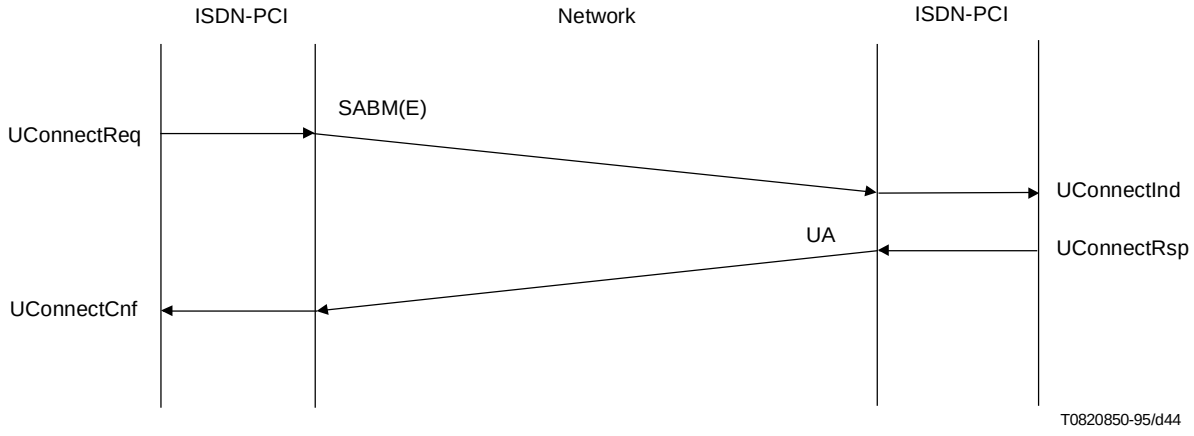


Figure II.1 – Connection phase

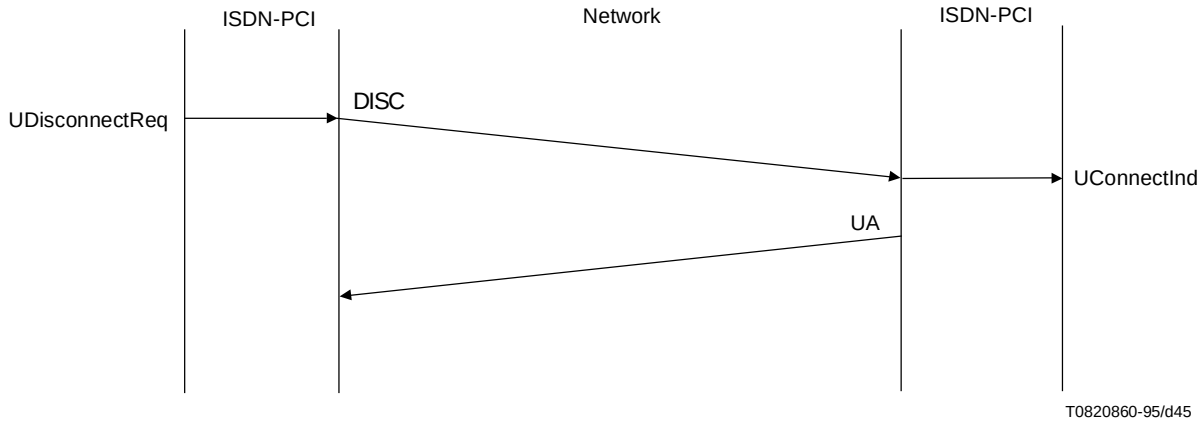


Figure II.2 – Disconnection phase

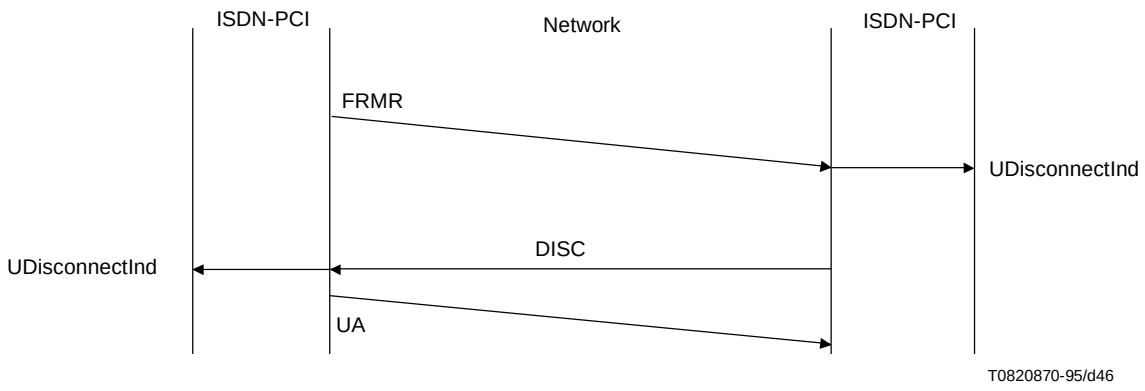


Figure II.3 – Error situation

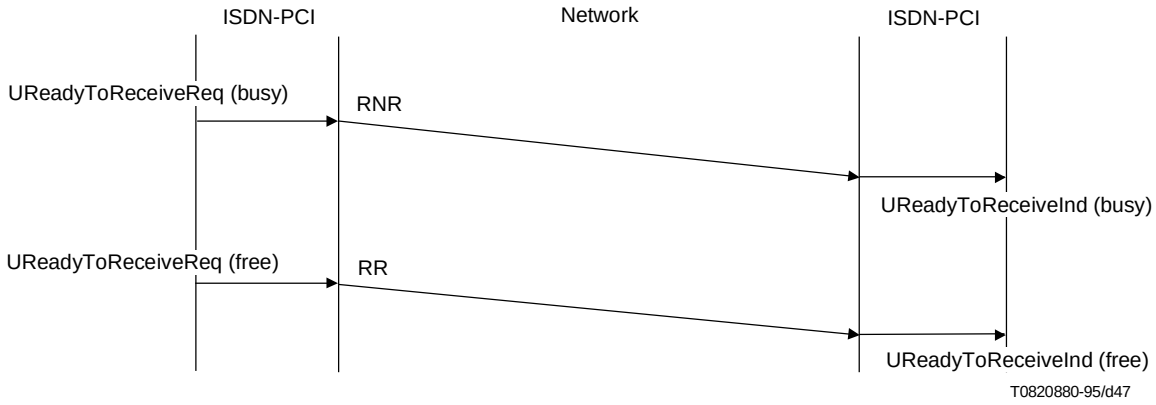


Figure II.4 – Flow control situation

II.2 HDLC protocol

Table II.2 shows the mapping of User Plane messages to service primitives.

Table II.2 – Mapping between User Plane message and protocol messages

PCI Message	Primitive
UDataReq	I
UDataInd	I

II.3 HDLC protocol with error

Table II.3 shows the mapping of User Plane messages to service primitives.

Table II.3 – Mapping between User Plane message and protocol messages

PCI Message	Primitive
UDataReq	I
UDataInd	I

II.4 PPP protocol

Table II.4 shows the mapping of User Plane messages to service primitives.

Table II.4 – Mapping between User Plane message and protocol messages

PCI Message	Primitive
UConnectReq	UI-CONFIGURE REQUEST
UConnectInd	UI-CONFIGURE REQUEST
UConnectRsp	UI-CONFIGURE ACK/NACK
UConnectCnf	UI-CONFIGURE ACK/NACK
UDisconnectReq	UI-TERMINATE REQUEST
UDisconnectInd	UI-CONFIGURE REJECT/ TERMINATE REQUEST
UDataReq	UI-INFO (NCP)
UdataInd	UI-INFO (NCP)
UerrorInd	UI-PROTOCOL REJECT CODE REJECT

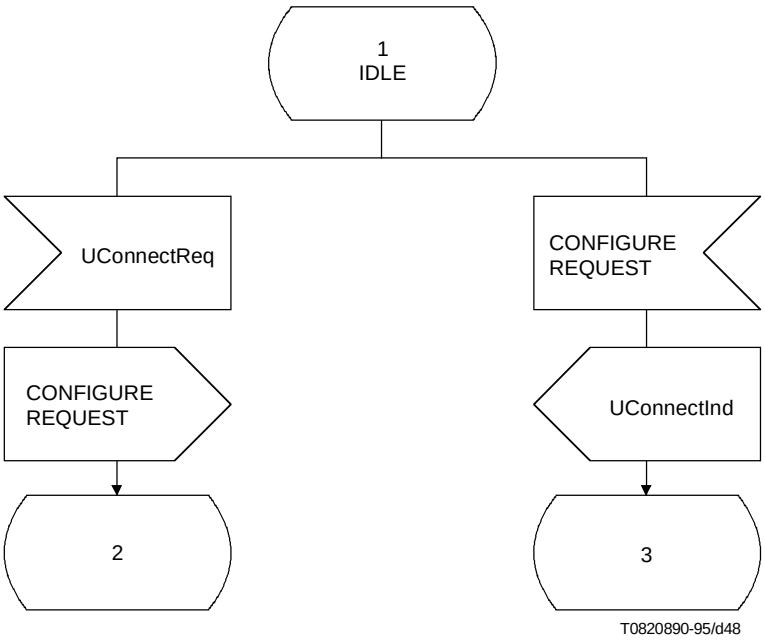


Figure II.5 – Idle

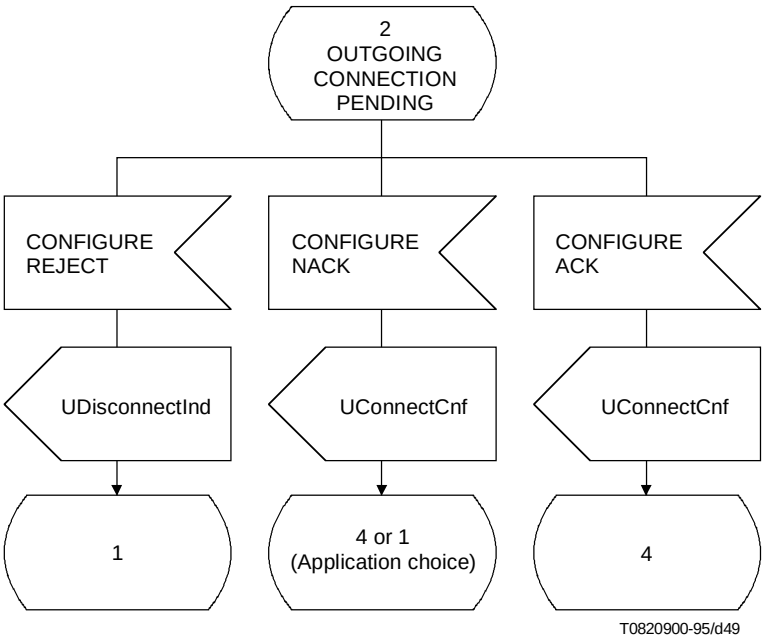


Figure II.6 – Outgoing connection pending

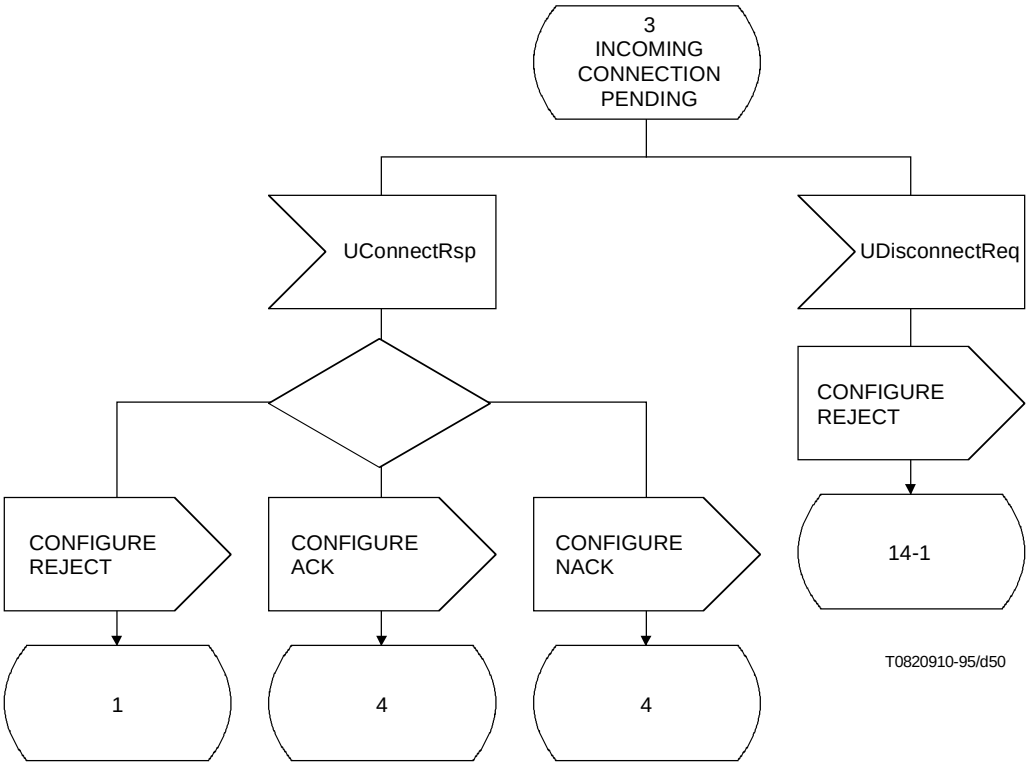


Figure II.7 – Incoming connection pending

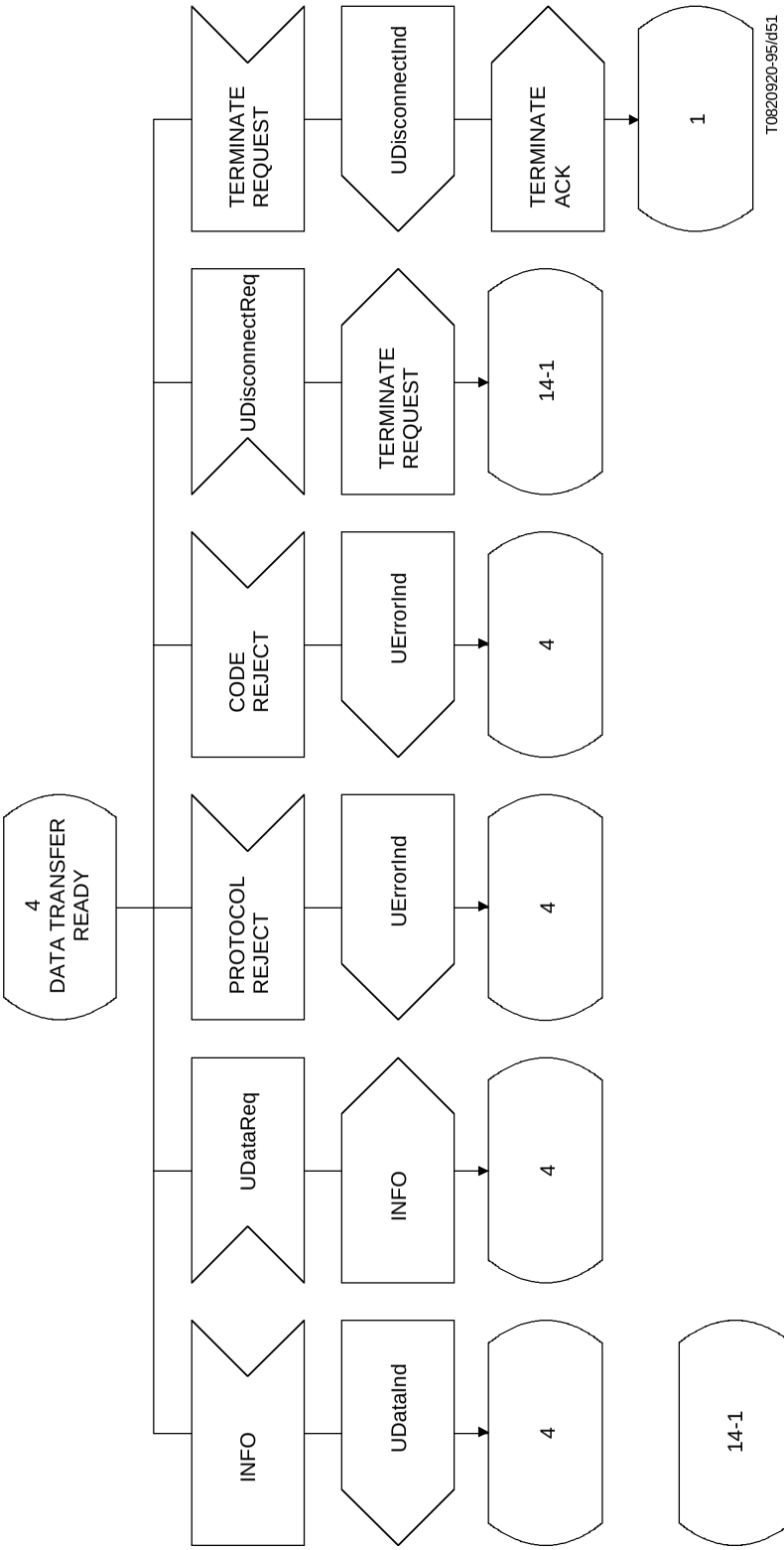


Figure II.8 – Data transfer ready

II.5 SDLC protocol

Table II.5 shows the mapping of User Plane messages to service primitives.

Table II.5 – Mapping between User Plane message and protocol messages

PCI Message	Primitive
UConnectReq	XID-P
UConnectInd	XID-P
UConnectRsp	XID-F
UConnectCnf	XID-F
UDisconnectReq	DISC-P
UDisconnectInd	RD-F or DM-F
UDataReq	I
UDataInd	I
UExpeditedDataReq	UI
UExpeditedDataInd	UI
UReadyToReceiveReq	Local meaning
UReadyToReceiveInd	Local meaning

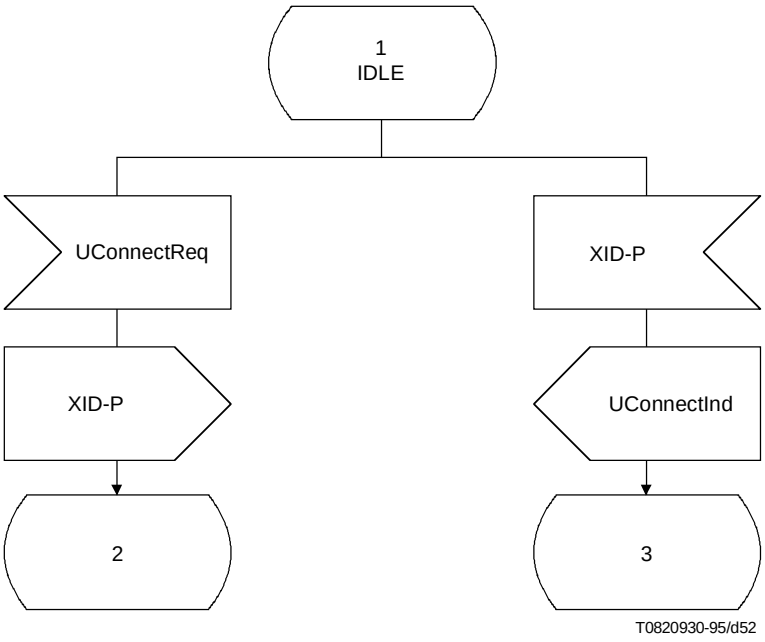


Figure II.9 – Idle

Ix-y states are intermediate states.

Ox-y states are optional states.

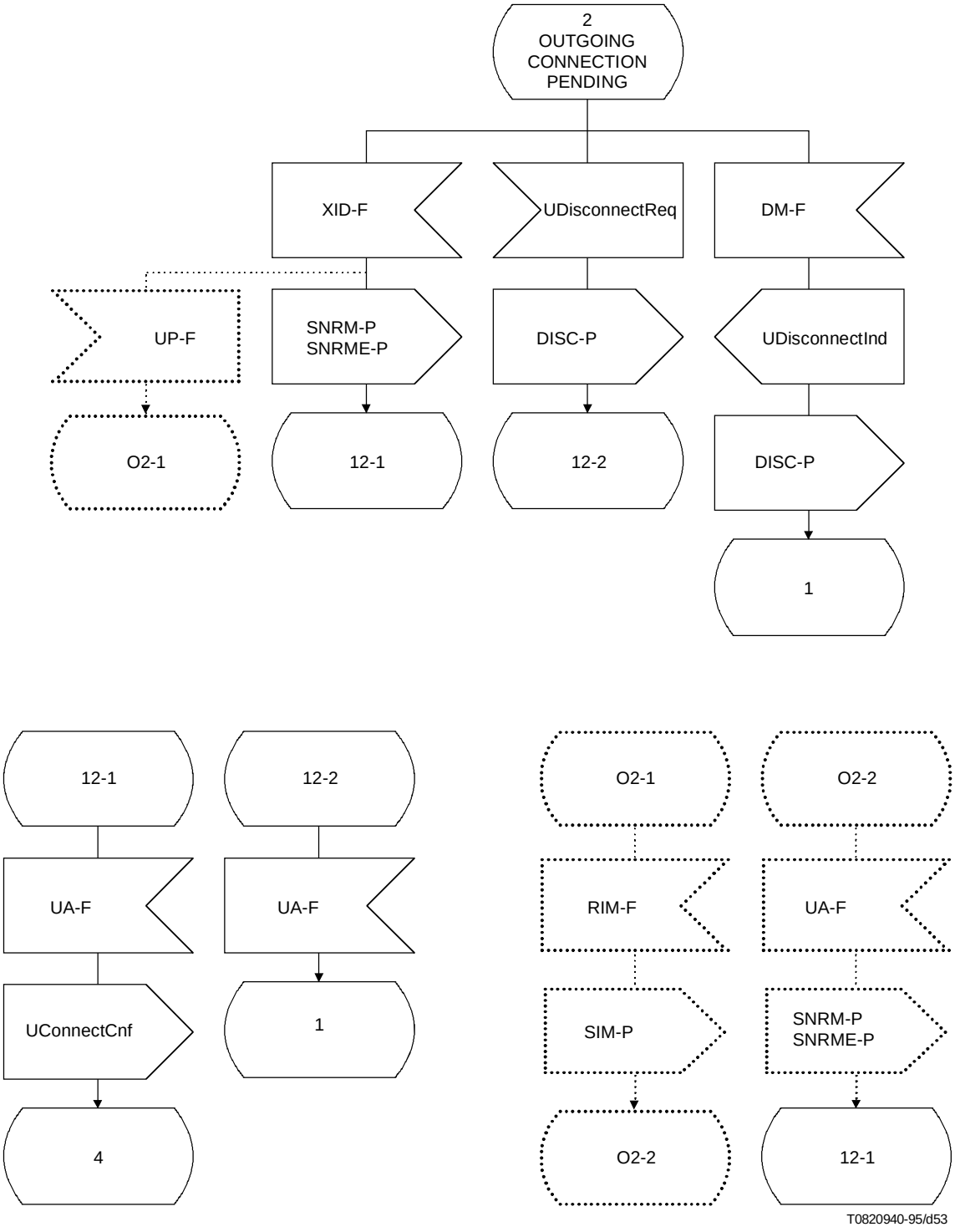


Figure II.10 – Outgoing connection pending (primary)

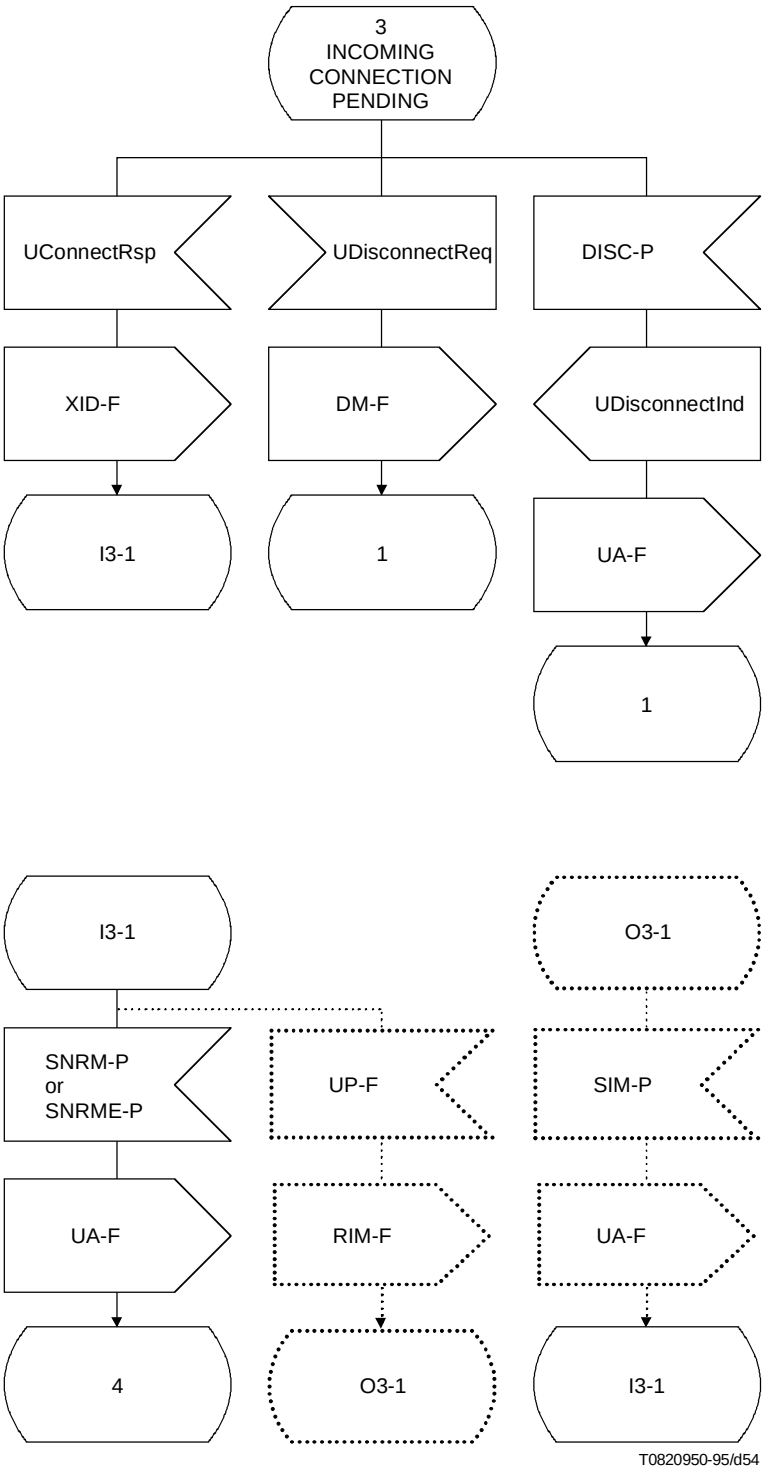


Figure II.11 – Incoming connection pending (secondary)

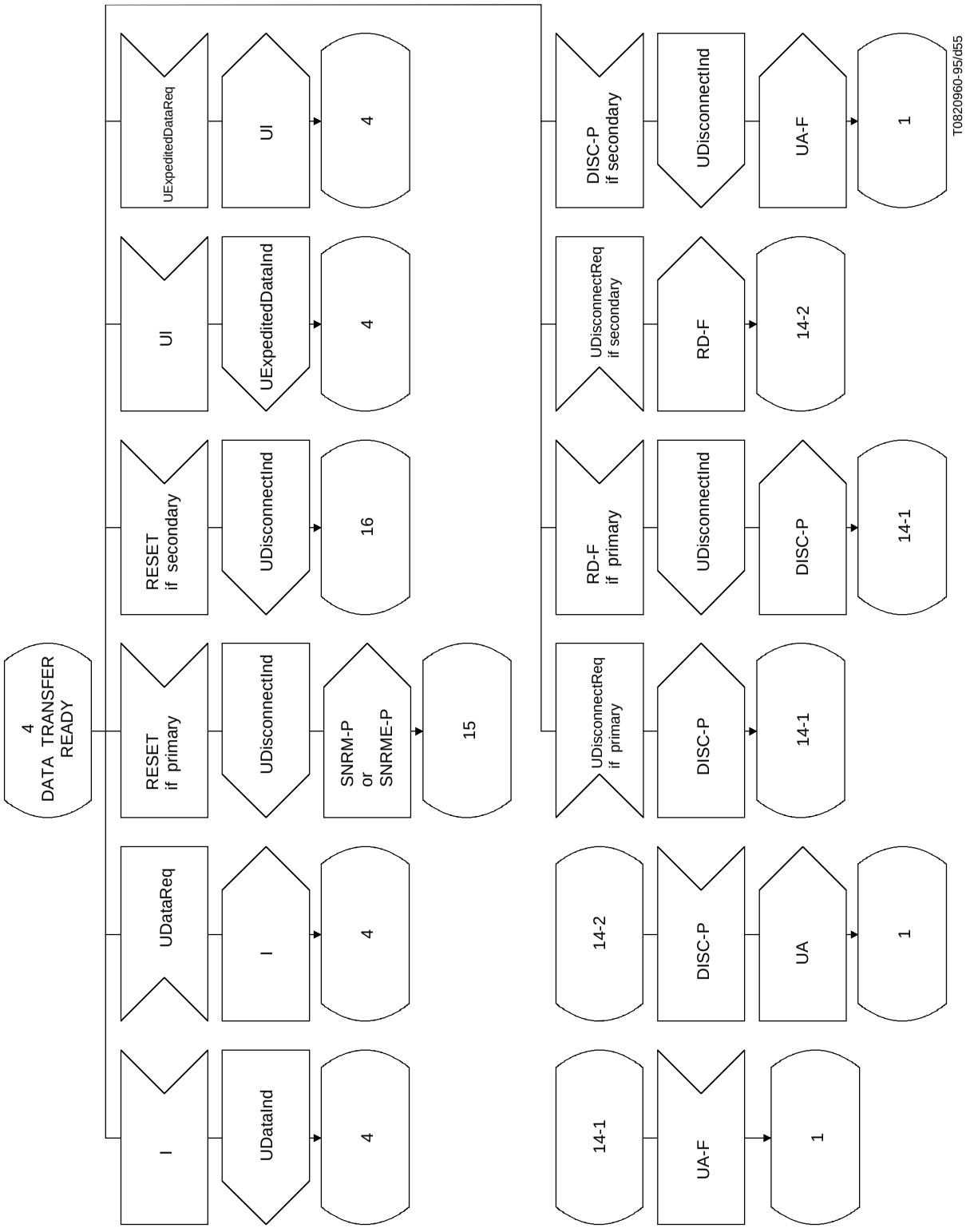


Figure II.12 – Data transfer ready

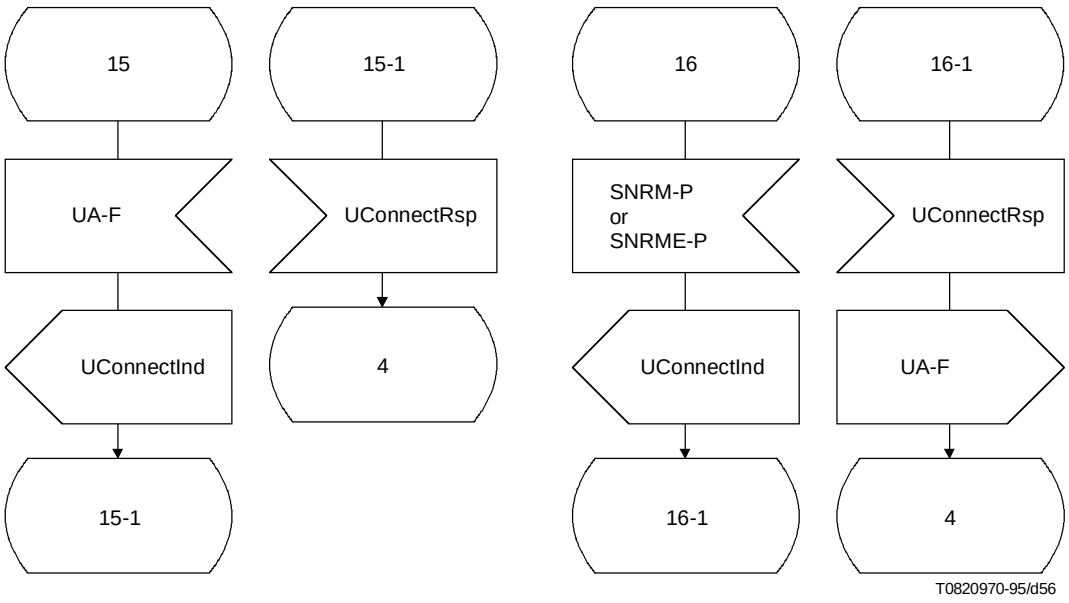


Figure II.13 – Reset

II.6 V.110 protocol

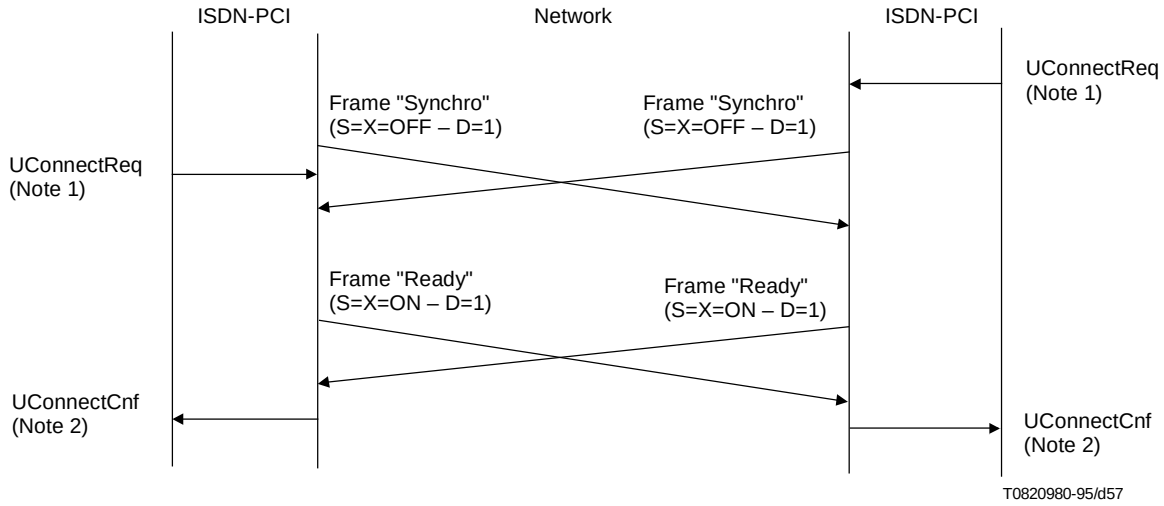
Table II.6 shows the mapping of User Plane messages to service elements. For V.110 protocol, there is no direct link between user connection messages and protocol frames. The connection phase consists of synchronization and negotiation. It begins without application demand, when ISDN channel is established.

To make PCI messages easier to understand, Table II.6 shows a theoretic mapping between User Plane messages and V.110 frames.

Table II.6 – Mapping between User Plane message and V.110 frame

PCI Message	Frame
UConnectReq	Frame “Synchronization” (bit S = bit X = OFF) Local meaning.
UConnectInd	Frame “Ready” (bit S = bit X = ON) – Local meaning: Remote synchronization has been received.
UConnectRsp	Frame “Synchronization” (bit S = bit X = OFF) – Local meaning. Note
UConnectCnf	Frame “Ready” (bit S = bit X = ON) (A negotiation delay may be necessary before data transfer is ready.)
UDisconnectReq	Frame with bit S = OFF, bit X = ON, D = 0
UDisconnectInd	Frame with bit S = OFF, bit X = ON, D = 0
UDataReq	Data frame
UDataInd	Data frame
UReadyToReceiveReq	Local meaning
UReadyToReceiveInd	Local meaning
NOTE – Sending a UConnectResponse does not mean negotiation is finished. Data transfer can be unavailable for a time. In this case, the PUF is sent a NAFBusy error (see Figure II.15).	

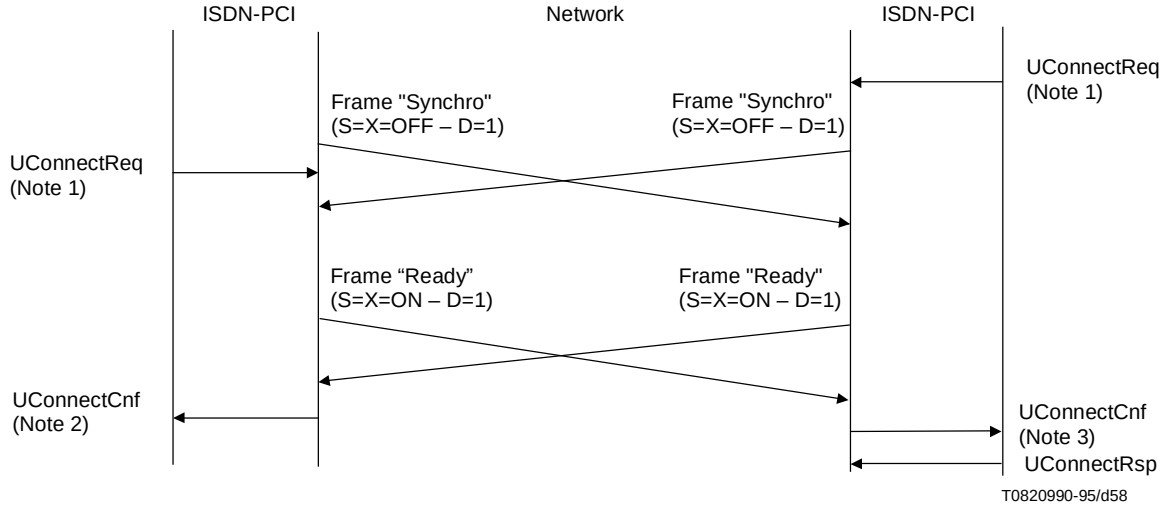
Figures II.14 to II.19 show more general cases, but not every possible situation.



NOTE 1 – UConnectReq is not really linked with the frame "synchronization". It can be sent before or after.

NOTE 2 – UConnectConf means the two sides are synchronized.

Figure II.14 – Connection phase



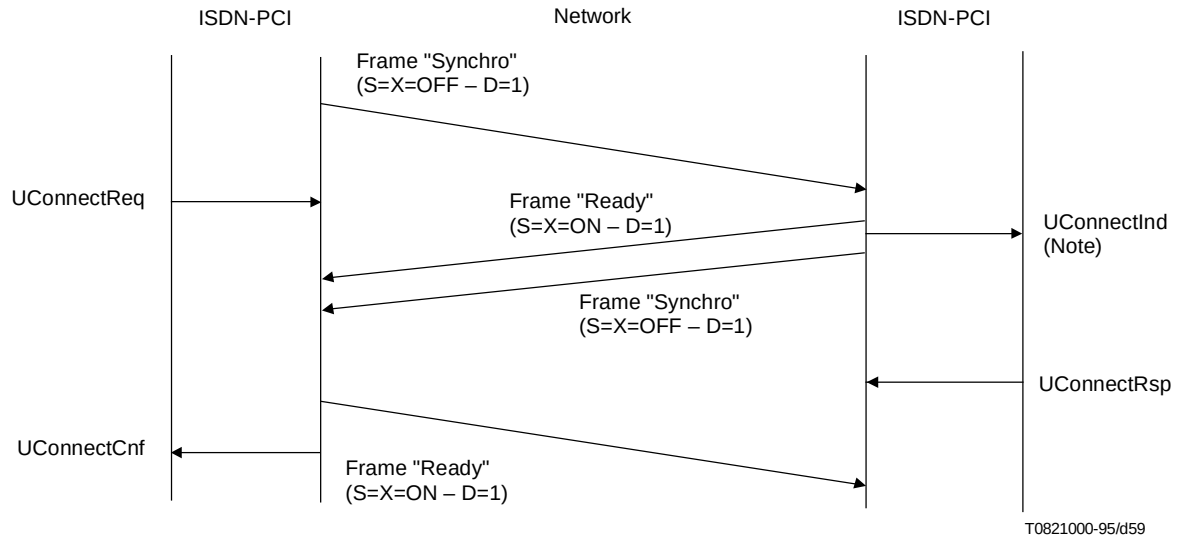
NOTE 1 – UConnectReq is not really linked with the frame "synchronization". It can be sent before or after.

NOTE 2 – UConnectCnf means the two sides are synchronized.

NOTE 3 – UconnectInd means the two sides are synchronized.

Figure II.15 – Connection phase

Figure II.16 shows another possible situation. It is a theoretic situation: low layer V.110 module generally begins synchronization phase just when the B-channel is established.



NOTE – UConnectInd means "the other side is going ready".

Figure II.16 – Connection phase

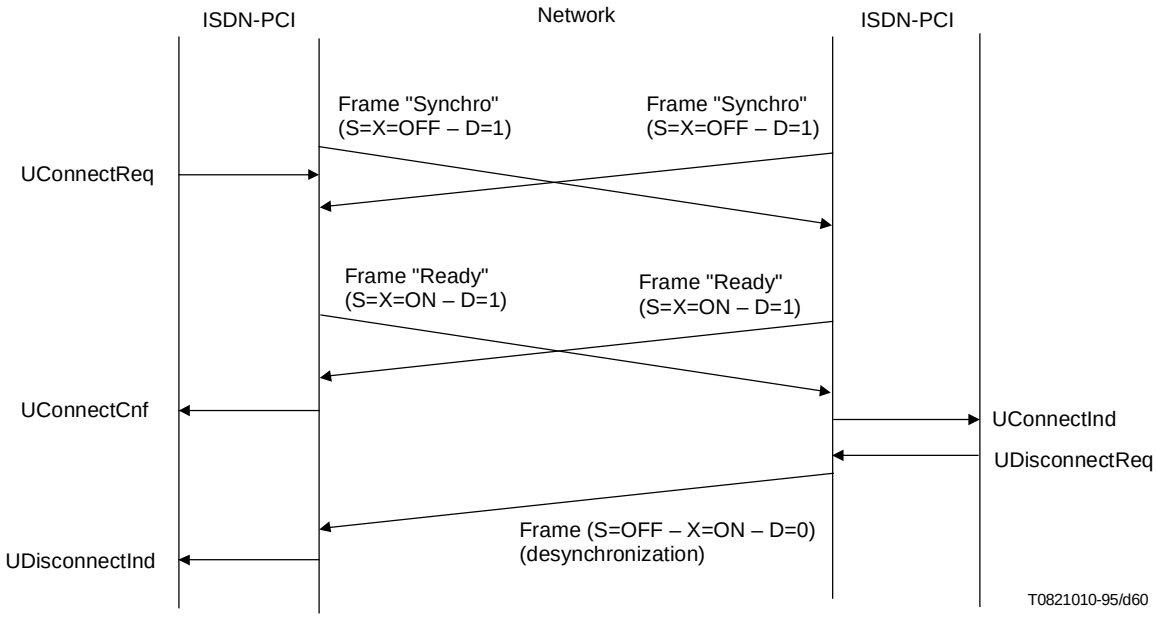


Figure II.17 – Remote disconnection during connection phase

NOTE – Depending on V.110 BChannelDisconnection parameter, a user disconnection may imply the B-channel disconnection.

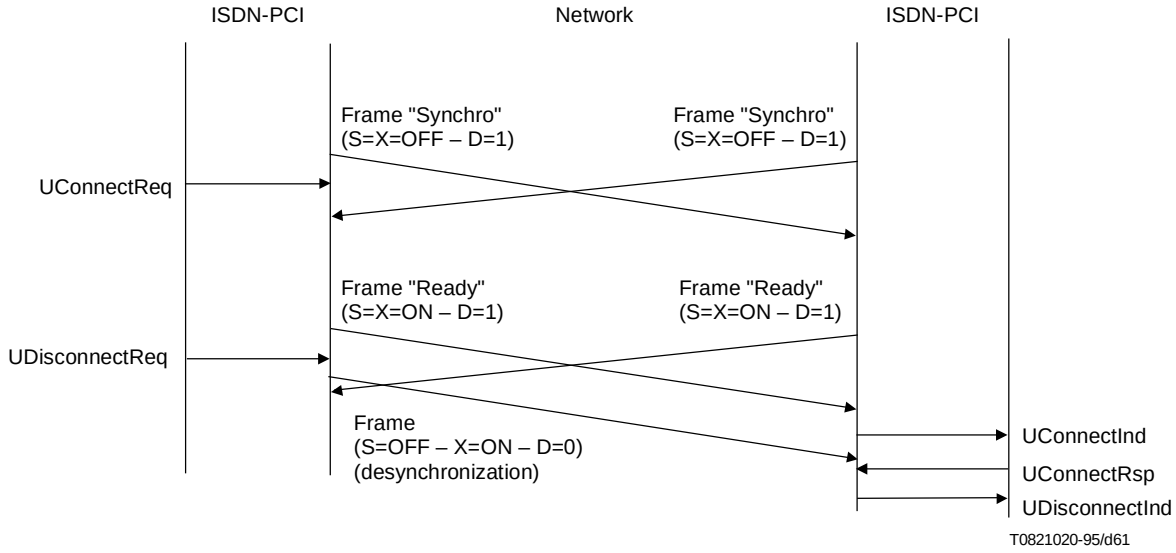


Figure II.18 – Disconnection during connection phase

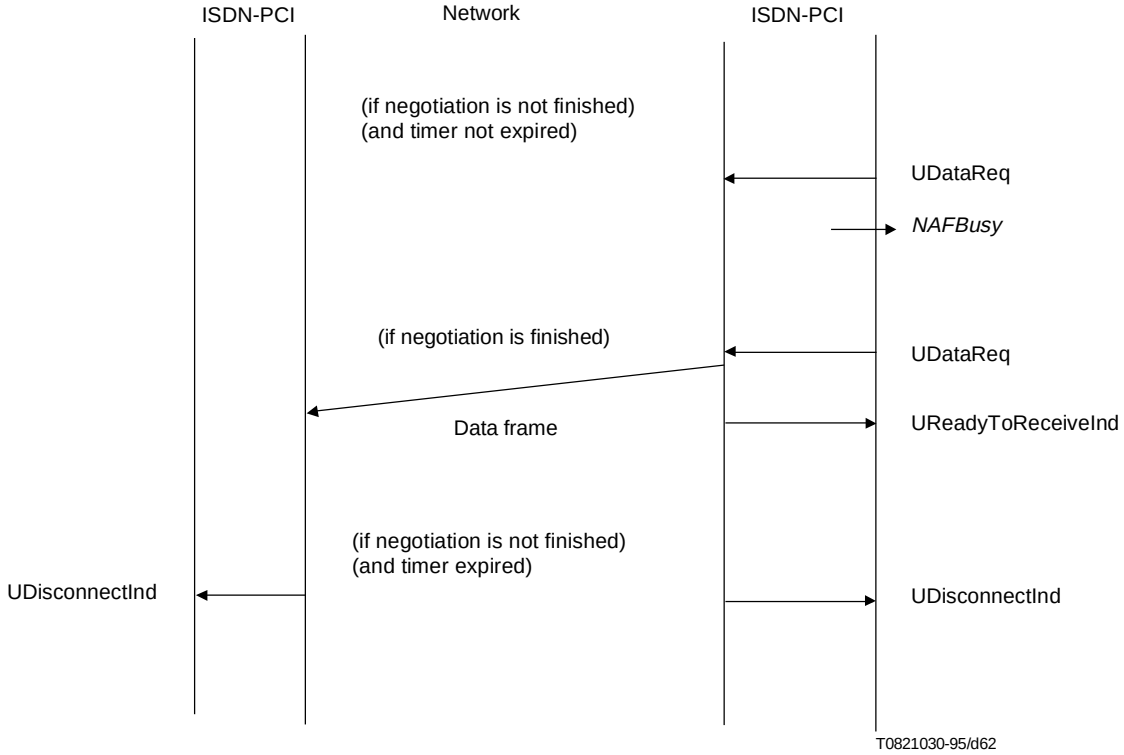


Figure II.19 – Data transfer

Superseded by a more recent version

CONTENTS

PART 6

Superseded by a more recent version

PART 6: LAYER THREE PROTOCOLS

Summary

This part of the specification details Procedures, Messages and Parameters used to access the ISDN-PCI's User Plane Protocols that provide a Layer 3 communication service.

Introduction

The use of different Integrated Services Digital Network (ISDN) programming interfaces by terminal equipment has hindered the development of common applications using ISDN which, in turn, has constrained deployment of ISDN applications on terminal equipment such as personal computers.

This ITU-T ISDN Application Programming Interface (API), called ISDN Programming Communication Interface (PCI), is an application interface for accessing and administering ISDN services. The ISDN-PCI is a set of specifications in which this part is the layer 3 protocol usage description.

ISDN-PCI has been defined in order to provide a standard for terminal equipment providers that makes possible 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 future ISDN extensions will not affect the operation of existing applications.

1 Scope

This part describes the Integrated Services Digital Network Programming Communication Interface (ISDN-PCI) layer 3 protocols provided by the ISDN-PCI User Plane. It forms a part of specifications on ISDN-PCI.

It describes the specific elements (messages, parameters, attribute sets, etc.) relating to the layer 3 user protocols. This part covers T.90, ISO/IEC 8208 and T.70NL user protocols. ITU-T Recommendation describing other layer 3 user protocols are for further study.

2 References

[1] ITU-T Recommendation T.90 (1992), *Characteristics and protocols for terminals for telematic services in ISDN.*

[2] ISO/IEC 8208:1995, *Information technology – Data communications – X.25 Packet Layer Protocol for Data Terminal Equipment.*

[3] ITU-T Recommendation X.213 (1995), *Information technology – Open Systems Interconnection – Network service definition.*

[4] Part 1, *General architecture.*

[5] Part 2, *Basic services.*

[6] Part 3, *User Plane protocol Management Architecture.*

[7] ITU-T Recommendation X.31 (1995), *Support of packet mode terminal equipment by an ISDN.*

3 Definitions

This part defines the following terms:

3.1 **attribute set**: Set of parameters driving user protocols and ISDN signalling.

3.2 **B-channel**: Logical ISDN channel for the use of data transfer.

3.3 **control plane**: Logical grouping of functionality for access of ISDN signalling.

3.4 **D-channel**: Logical ISDN channel used for signalling and, in some cases, for data transfer.

3.5 **ISDN access**: Set of ISDN channels provided by a single Network Access Facility (NAF) to access ISDN services.

3.6 **ISDN programming communication interface (ISDN-PCI)**: Network (ISDN) oriented software interface providing access provisions for programming network signalling and user data exchange.

3.7 **message**: Unit of information transferred through the interface between the Network Access Facility (NAF) and the PCI User Facility (PUF).

3.8 **network access facility (NAF)**: Functional unit located between the ISDN-PCI and the network related layers.

3.9 **network connection object (NCO)**: Abstract object within the NAF that shall be created by the PUF to gain access to network signalling or data.

3.10 **NULL layer**: 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.

3.11 **PCI User Facility (PUF)**: Functional unit using the ISDN-PCI to access an NAF. In fact, the local application using the interface.

3.12 **user connection**: Connection accessible through User Plane functionality.

3.13 **user plane**: Logical grouping of functionality for access of user protocols and data.

3.14 **user protocol**: Protocol running and conforming to User Plane functionality.

4 Abbreviations

This part uses the following abbreviations:

API	Application Programming Interface
ISDN	Integrated Services Digital Network
LAP-B	Link Access Procedure Balanced
LAP-D	Link Access Procedure for D-channel
N-SAP	Network layer – Service Access Point
NAF	Network Access Facility
NCO	Network Connection Object
PCI	Programming Communication Interface
PUF	Programming communication interface User Facility
SAP	Service Access Point
X.25 PLP	X.25 Packet Layer Protocol

5 Reader's guidance

5.1 Reader's guide

This part is intended for software developers, implementors of applications and equipment manufacturers by providing them the usage description of layer 3 User Plane protocols.

5.2 How to use this part

Readers who:

- need a quick overview of the described User Plane protocols should refer to Part 3;
- intend to implement an application using the ISDN-PCI interface with a layer 3 user protocol should read this part. Protocol usage is provided in clauses 6 and 7;
- intend to build an ISDN adaptor card or equipment using the ISDN-PCI interface should read this part. Protocol usage is provided in clauses 6 and 7, while Appendices I and II describe informative default configuration values and NAF diagrams.

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

Table 1 – List of contents

Clause, Annex, Appendix	Contains ...
Clause 1	... the scope of this part. This describes what this part covers
Clause 2	... references
Clause 3	... definitions of the terms used throughout this part
Clause 4	... definitions of the abbreviations used throughout this part
Clause 5	... gives an overview
Clause 6	... ISO/IEC 8208 and T.90 protocol
Clause 7	... T.70 protocol
Appendix I	... informative default configuration values
Appendix II	... informative NAF-SDL diagrams
Appendix III	... information on X.25 usage

For each supported protocol, this part provides:

- description of available user messages (see clause 2);
- description of useful user parameters (see clause 3);
- the protocol state diagram (see clause 4);
- coordination function information (see clause 5);
- specific NCO selection criteria if it exists (see clause 6);
- specific error handling and codes (see clause 7);
- AttributeSet definition (see clause 8).

Appendix I gives default protocol configuration values, if any.

Appendix II shows NAF-SDL diagrams describing most of the situations.

6 ISO/IEC 8208 protocol and T.90 protocol

6.1 Introduction

This clause deals with the ISO/IEC 8208 protocol [2] and with the T.90 protocol [1]. Figure 1 shows the localization of the user protocol access.

General description conventions are provided in [5].

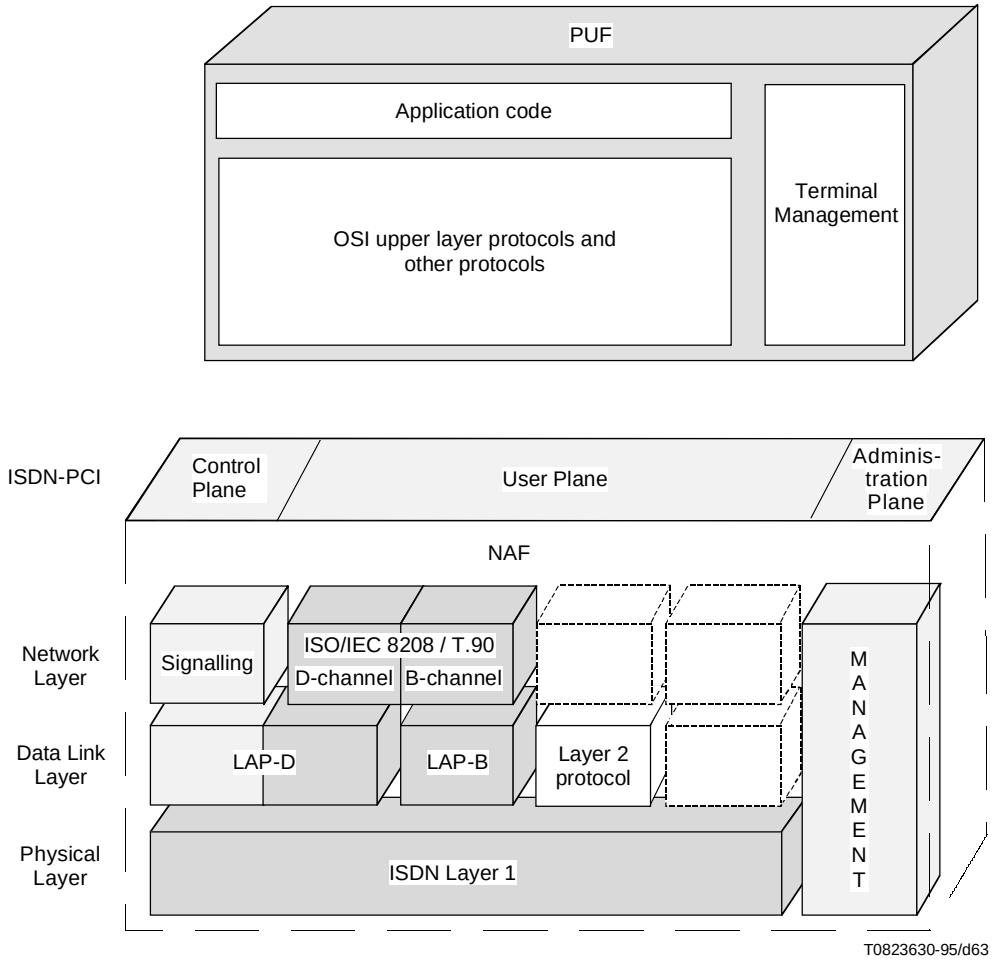


Figure 1 – OSI location

6.2 Description of messages

The User Plane messages provide an X.213-access to ISO/IEC 8208 or T.90 protocol stacks. Following is a list and short description of significant User Plane messages. Table 2 gives an overview of these messages.

Table 2 – Overview of user messages

Message identifier	Class	Message name	Purpose of message	Used for ISO/IEC 8208	Used for T.90
301	1	UConnectReq	Request establishment of a user connection	X	X
302	1	UConnectInd	Indicate establishment of a user connection has been requested	X	X
303	1	UConnectRsp	Indicate acceptance of user connection establishment	X	X
304	1	UConnectCnf	Confirm user connection has been established	X	X
305	1	UDisconnectReq	Request removal of user connection	X	X
306	1	UDisconnectInd	Indicate removal of user connection	X	X
307	1	UDataReq	Request data transfer on an established user connection	X	X
308	1	UDataInd	Indicate arrival of transferred data on an established user connection	X	X
309	1	UExpeditedDataReq	Request expedited data transfer on an established user connection	X	
310	1	UExpeditedDataInd	Indicate presence of transferred expedited data on an established user connection	X	
311	1	UResetReq	Request reset to initial state of an established user connection	X	X
312	1	UResetInd	Indicate reset to initial state of an established user connection	X	X
313	1	UResetRsp	Indicate acceptance of reset to initial state of an established user connection	X	X
314	1	UResetCnf	Confirm acceptance of reset to initial state of an established user connection	X	X
315	1	UDataAcknowledgeReq	Request acknowledgement of data received on an established user connection	X	
316	1	UDataAcknowledgeInd	Indicate acknowledgement of data transferred on an established user connection	X	
317	1	UReadyToReceiveReq	Used to perform flow control for a user connection	X	X
318	1	UReadyToReceiveInd	Used to indicate flow control status on a user connection	X	X

6.2.1 UConnectReq

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 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.
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 facilities If present, the following facilities shall be overridden by information found elsewhere in this message: – BCUG; – FastSelect; – Called Address Extension; – Calling Address Extension.

Related: UConnectCnf.

Protocols: This message is used in the two User Plane protocols T.90 and ISO/IEC 8208.

6.2.2 UConnectInd

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 Bilateral Closed User Group facility information. If present, then addressing information shall not be present.
FastSelect	O	Authorization 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 facilities The following facilities, if present, are presented by the use of specific parameters: – BCUG; – FastSelect; – Called Address Extension; – Calling Address Extension.

Related: UConnectRsp.

Protocols: This message is used in the two User Plane protocols T.90 and ISO/IEC 8208.

6.2.3 UConnectRsp

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 UConnectInd.
PacketSize	O	Used to indicate agreed value
WindowSize	O	Used to indicate agreed value
FacilityData	O	Used to supply facilities

Related: UConnectInd.

Protocols: This message is used in the two User Plane protocols T.90 and ISO/IEC 8208.

6.2.4 UConnectCnf

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.
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 UConnectReq.
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 facilities

Related: UConnectReq.

Protocols: This message is used in the two User Plane protocols T.90 and ISO/IEC 8208.

6.2.5 UDisconnectReq

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.
UserData	O	Only allowed if FastSelect parameter was specified during the user connection establishment Maximum size of 128 octets
X25Cause	O	Reason to remove the user connection Both X213Cause and X25Cause cannot be used on the same message
X25Diagnostic	C	Complementary information for reason. Optional if X25Cause parameter supplied; otherwise, not allowed.
FacilityData	O	Used to supply facilities
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.		

Related: None.

Protocols: This message is used in the two User Plane protocols T.90 and ISO/IEC 8208.

6.2.6 UDisconnectInd

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	Reason to remove the user connection. Both X213Cause and X25Cause cannot be used on the same message.
X25Diagnostic	C	Complementary information for Reason. Optional if X25Cause parameter supplied; else, not allowed.
FacilityData	O	Used to supply facilities
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.		

Related: None.

Protocols: This message is used in the two User Plane protocols T.90 and ISO/IEC 8208.

6.2.7 UDataReq

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 Qualifier bit, the More Bit and to request confirmation of receipt of data.

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

Mandatory data shall be provided in the data buffer.

Related: UReadyToReceiveInd.

Protocols: This message is used in the two User Plane protocols T.90 and ISO/IEC 8208.

6.2.8 UDataInd

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 Qualifier bit value, the More Bit value and the need of confirmation of data reception.

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

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

Related: UReadyToReceiveReq.

Protocols: This message is used in the two User Plane protocols T.90 and ISO/IEC 8208.

6.2.9 UExpeditedDataReq

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 UDataReq messages.

Parameters:

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

Related: None.

Protocol: This message is used in the User Plane protocol ISO/IEC 8208.

6.2.10 UExpeditedDataInd

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 UDataInd messages.

Parameters:

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

Related: None.

Protocol: This message is used in the User Plane protocol ISO/IEC 8208.

6.2.11 UResetReq

Class: 1 (Basic class).

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

Parameters:

Name	Provided	Comment
NCOID	M	Identifies the user 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	Reason to reset the user connection
X25Diagnostic	C	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.		

Related: UResetCnf.

Protocols: This message is used in the two User Plane protocols T.90 and ISO/IEC 8208.

6.2.12 UResetInd

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	Reason to reset the user connection
X25Diagnostic	O	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.		

Related: UResetRsp.

Protocols: This message is used in the two User Plane protocols T.90 and ISO/IEC 8208.

6.2.13 UResetRsp

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.

Parameter:

Name	Required	Comment
NCOID	M	Identifies the user connection

Related: UResetInd.

Protocols: This message is used in the two User Plane protocols T.90 and ISO/IEC 8208.

6.2.14 UResetCnf

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.

Parameter:

Name	Provided	Comment
NCOID	M	Identifies the user connection

Related: UResetReq.

Protocols: This message is used in the two User Plane protocols T.90 and ISO/IEC 8208.

6.2.15 UDataAcknowledgeReq

Class: 1 (Basic class).

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

Parameter:

Name	Required	Comment
NCOID	M	Identifies the user connection

Related: UDataInd.

Protocol: This message is used in the User Plane protocol ISO/IEC 8208.

6.2.16 UDataAcknowledgeInd

Class: 1 (Basic class).

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

Parameter:

Name	Provided	Comment
NCOID	M	Identifies the user connection

Related: UDataReq.

Protocol: This message is used in the User Plane protocol ISO/IEC 8208.

6.2.17 UReadyToReceiveReq

Class: 1 (Basic class).

Description: This message allows the PUF to indicate to the NAF if it can accept incoming data (UDataInd 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.

Related: UDataInd.

Protocols: This message is used in the two User Plane protocols T.90 and ISO/IEC 8208.

6.2.18 UReadyToReceiveInd

Class: 1 (Basic class).

Description: This message allows the NAF to indicate to the PUF if the user connection permits the sending of data (UDataReq 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

Related: UDataReq.

Protocols: This message is used in the two User Plane protocols T.90 and ISO/IEC 8208.

6.3 Messages parameters

This subclause describes parameters for the ISO/IEC 8208 User Plane and the T.90 User Plane. They are alphabetically ordered.

Information presentation is provided in [5].

Table 3 – Overview of user parameters

Parameter identifier	Parameter name	Use in ISO/IEC 8208 user messages	Use in T.90 user messages	Use in UAttributeSet	Other use
1	Algorithm				X
2	Bilateral closed user group	X	X		
4	Bit_DQM	X	X (Note)		
5	CalledDTEAddress	X	X		
6	CalledDTEAddressExt	X	X		
9	CallingDTEAddress	X	X		
10	CallingDTEAddressExt	X	X		
29	ExpeditedData	X			
31	FacilityData	X	X (Note)		
32	FastSelect	X	X (Note)	X	
33	GroupID				X
38	L2ConnectionMode			X	
39	L2FrameSize			X	
40	L2WindowSize			X	
41	L2XID			X	
42	L3ConnectionMode			X	
43	L3IncomingCount			X	
44	L3OutgoingVCCount			X	

Table 3 – Overview of user parameters *(concluded)*

Parameter identifier	Parameter name	Use in ISO/IEC 8208 user messages	Use in T.90 user messages	Use in UAttributeSet	Other use
45	L3TwoWayCount			X	
50	NCOType				X
52	PacketSize	X	X	X	
54	QOSParameters	X	X	X	
55	ReadyFlag	X	X		
57	ReceiptConfirm	X			
58	RespondingDTEAddress	X	X		
59	RespondingDTEAddressExt	X	X		
61	TEI			X	
62	UProtocol			X	
63	UAttributeName				X
64	UDirection				X
65	UserData	X			
67	WindowSize	X	X	X	
68	X213Cause	X	X		
69	X213Origin	X	X		
70	X25Cause	X	X		
71	X25Diagnostic	X	X		
NOTE – This parameter shall be used in accordance with the rules of Recommendation T.90 [1].					

6.3.1 Algorithm

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

Type: 1.

Field	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

6.3.2 Bilateral closed user group (Bcug)

Description: This parameter is used to pass Bilateral closed user group information to/from the PUF.

Type: 2.

Field	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

6.3.3 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;

– Qualifier bit value (bit 2);

– More Data bit value (bit 3).

Each information uses a binary position. The Most Significant Bit (MSB) is 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 the sum of the value for each bit (logical OR).

Type: 4.

Field	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 Qualifier bit 0 – Reset Qualifier bit Bit 3: 1 – Set More bit 0 – Reset More bit

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

For T.90 protocol, this parameter shall be used in accordance with the rules of Recommendation T.90 [1].

6.3.4 CalledDTEAddress

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

Type: 5.

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

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.

6.3.5 CalledDTEAddressExt

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

Type: 6.

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

Remark: The BCD translation is provided by the NAF.

6.3.6 CallingDTEAddress

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

Type: 9.

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

Remark: The BCD translation is provided by the NAF.

6.3.7 CallingDTEAddressExt

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

Type: 10.

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

Remark: The BCD translation is provided by the NAF.

6.3.8 ExpeditedData

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

Type: 29.

Field	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

6.3.9 FacilityData

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

Type: 31.

Field	Field type	Direction	Required	Comment
FacilityData	Octet string	B	M	Encoded as facility information defined in ISO/IEC 8208 [2] 109 octets is the maximum length

6.3.10 FastSelect

Description: This parameter is used to pass Fast Select Facility information to/from the PUF.

Type: 32.

Field	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

Remarks: When specified on a UConnectReq 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 UDisconnectInd 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 UConnectCnf can have a maximum UserData parameter of 128 octets. Subsequent to this, both the UDisconnectInd and UDisconnectReq fields may also have maximum UserData parameters of 128 octets.

When received on a UConnectInd 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 UDisconnectReq. 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 UConnectRsp with a maximum UserData parameter of 128 octets. Subsequent to this, both the UDisconnectInd and UDisconnectReq fields may also have maximum UserData parameters of 128 octets.

6.3.11 GroupID

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

Type: 33.

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

6.3.12 L2ConnectionMode

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

Type: 38.

Field	Field type	Direction	Required	Comment
Value	Octet	P	M	dte (1) – Acts as DTE as defined in ISO/IEC 7776 dce (2) – Acts as DCE as defined in ISO/IEC 7776 auto (3) – When calling, acts as DTE; when called, acts as DCE.

6.3.13 L2FrameSize

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

Type: 39.

Field	Field type	Direction	Required	Comment
Value	Octet-string	P	M	Frame size in octets Length is fixed to 2 octets The first octet contains the most significant byte of the 2 bytes containing the value

6.3.14 L2WindowSize

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

Type: 40.

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

6.3.15 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 octets

6.3.16 L3ConnectionMode

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

Type: 42.

Field	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/IEC 8208 "auto"

6.3.17 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.

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

6.3.18 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.

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

6.3.19 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.

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

6.3.20 NCOType

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

Type: 50.

Field	Field type	Direction	Required	Comment
Identifier	Octet		M	U3 (2) – Network user access with NAF signalling coordination (NAF coordination functionality) C/U (3) – Signalling and network layer user access U3/G (4) – Network user access to additional virtual circuits. This NCO must be grouped to an existing U3 or C/U type NCO.

6.3.21 PacketSize

Description: This parameter is used to pass 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 4) Maximum size of data that can be received with UDataInd
Outvalue	Octet	B	M	Outbound maximum user data length (see Table 4) Maximum size of data that can be passed with UDataReq

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

– on UConnectReq, the PUF may specify the values it wishes to use;

– on UConnectCnf, the NAF shall always specify the values to be used for the user connection;

– on UConnectInd, 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 UConnectRsp, the PUF can specify values if the UConnectInd indicated that negotiation was possible.

For T.90 protocol, this parameter shall be used in accordance with the rules of Recommendation T.90 [1].

Table 4 – Precoded packet size values

Precoded value	Packet size (octet)	Precoded value	Packet size (octet)
4	16	9	512
5	32	10	1024
6	64	11	2048
7	128	12	4096
8	256		

6.3.22 QOSParameters

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

Type: 54.

Field		Field type	Direction	Required	Comment
Throughput	Usage	Boolean	B	M	Indicates if following values are included
	InTarget	Octet	B	C	Values provided in Table 5
	InLowest	Octet	B	C	Values provided in Table 5
	InAvailable	Octet	B	C	Values provided in Table 5
	InSelected	Octet	B	C	Values provided in Table 5
	OutTarget	Octet	B	C	Values provided in Table 5
	OutLowest	Octet	B	C	Values provided in Table 5
	OutAvailable	Octet	B	C	Values provided in Table 5
	OutSelected	Octet	B	C	Values provided in Table 5
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, otherwise 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, otherwise 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 the least significant byte. 65 535 means not used. Delay is expressed in milliseconds.

Remark: For T.90 protocol, this parameter shall be used in accordance with the rules of Recommendation T.90 [1].

Table 5 – Throughput precoding value

Precoding value	Throughput class	Precoding value	Throughput class
3	75	9	4800
4	150	10	9600
5	300	11	19 200
6	600	12	48 000
7	1200	13	64 000
8	2400	0	unused

6.3.23 ReadyFlag

Description: This parameter is used to request and indicate flow control status on a user connection.

Type: 55.

Field	Field type	Direction	Required	Comment
Usage	Boolean	B	M	TRUE – Data transfer is allowed FALSE – Data transfer is not allowed

6.3.24 ReceiptConfirm

Description: This parameter is used to request confirmation of data receipt for a User Plane connection.

Type: 57.

Field	Field type	Direction	Required	Comment
Value	Boolean	B	M	TRUE – Confirmation requested FALSE – Confirmation not requested

6.3.25 RespondingDTEAddress

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

Type: 58.

Field	Field type	Direction	Required	Comment
Address	IA5-string		M	16 octets is the maximum length

6.3.26 RespondingDTEAddressExt

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

Type: 59.

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

6.3.27 TEI

Description: This parameter is used to access a permanent link to a data packet switch (packet connection in D-channel).

Type: 61.

Field	Field type	Direction	Required	Comment
Value	Octet	B	M	

6.3.28 UProtocol

Description: This is used to select the User Plane protocol.

Type: 62.

Field	Field type	Direction	Required	Comment
L3Protocol	Octet	P	M	Default (255) – T.90 [1] T.90 (1) ISO/IEC 8208 (2)
L2Protocol	Octet	P	O	Default (255) – ISO/IEC 7776
L1Protocol	Octet	P	O	Default (255) – Transparent B-channel access

Remarks: Other possible values are described in [6].

6.3.29 UAttributeName

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

Type: 63.

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

6.3.30 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.

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

6.3.31 UserData

Description: This parameter is used to pass Data that is limited in size to/from the PUF.

Type: 65.

Field	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

6.3.32 WindowSize

Description: This parameter is used to pass window size information to/from the PUF.

Type: 67.

Field	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

Remarks: This parameter is used to determine the window sizes to be used for a user connection.

- On UConnectReq, the PUF may specify the values it wishes to use.
- On UConnectCnf, the NAF shall always specify the values to be used for the user connection.
- On UConnectInd, 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 UConnectRsp, the PUF can specify values if the UConnectInd indicated that negotiation was possible.

6.3.33 X213Cause

Description: This parameter is used to pass X.213 Cause information to/from the PUF.

Type: 68.

Field	Field type	Direction	Required	Comment
Value	Octet	B	M	See User Plane return code values in 6.7

6.3.34 X213Origin

Description: This parameter is used to pass X.213 origin information to/from the PUF.

Type: 69.

Field	Field type	Direction	Required	Comment
Value	Octet	B	M	undefined (1) NAF Provider (2) PUF User (3)

6.3.35 X25Cause

Description: This parameter is used to pass X.25 Cause information to/from the PUF.

Type: 70.

Field	Field type	Direction	Required	Comment
Value	Octet	B	M	See ISO/IEC 8208 [2] cause code values

6.3.36 X25Diagnostic

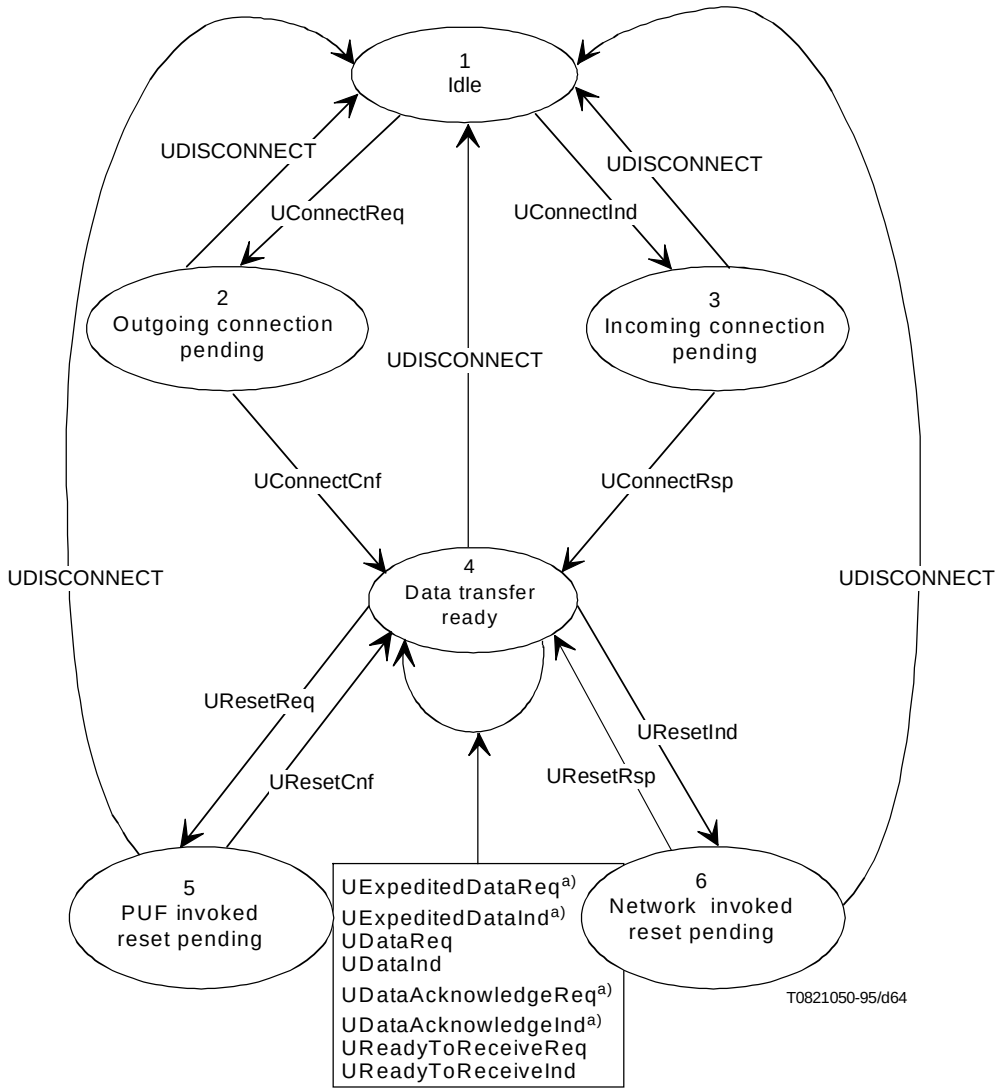
Description: This parameter is used to pass X.25 Diagnostic information to/from the PUF.

Type: 71.

Field	Field type	Direction	Required	Comment
Value	Octet	B	M	See ISO/IEC 8208 diagnostic values

6.4 State diagram

Figure 2 shows the different states a user connection using the user messages can get and in which order these messages shall be used.



a) This message is used for the ISO/IEC 8208 protocol only.

NOTE – Where UDISCONNECT appears, it can be either UDisconnectReq or UDisconnectInd.

Figure 2 – Overview of the User Plane messages

6.5 Coordination function

The coordination function may be used. See [6] for details.

6.6 Selection criteria

This subclause deals with ISO/IEC 8208 specific parameters. General NCO criteria are provided in [5].

6.6.1 NCO Selection

To select an NCO, the NAF uses the following parameters:

- packet size negotiation;
- window size negotiation.

6.6.1.1 Packet size negotiation

In the INCOMING CALL packet, if the packet size is not provided, the default value, i.e. 128 octets, is 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.

6.6.1.2 Window size negotiation

In the INCOMING CALL packet, if the Window size is not provided, the default value, i.e. 2, is 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.

6.6.1.3 Effective packet size and window size negotiation

In the UConnectRsp, if packet size is not provided, the packet size provided in the incoming call – i.e. UConnectInd – is accepted by the PUF. The same rules apply to the window size.

In the UConnectCnf, if the packet size/window size is not provided, the packet size/window size provided during the outgoing call – i.e. UConnectReq – is approved for use by the PUF.

6.6.2 Action if no NCO available

A disconnect with the X213reason “Connection rejection – reason unspecified transient” is issued by the NAF.

6.7 Specific error handling and codes

Errors are dealt with in the following manner.

6.7.1 Invalid use of User Plane messages

In case of:

- invalid use of Receipt Confirmation Service;
- invalid use of Confirmation request on UDataReq;
- invalid length of UDataReq UserData parameter;
- invalid use of Expedited Data;
- invalid issuing of messages while in Reset state,

action is:

- PUF is sent UDisconnectInd.

In case of:

– invalid use of Bit_DQM (association between More and Qualifier bits) parameters on subsequent UDataReq messages,

action is:

– PUF is sent UResetInd.

6.7.2 Other errors

In case of parameter content error, PUF is sent UDisconnectInd.

6.7.3 Causes

These values can be specified and are returned in the X213Cause parameter.

Table 6 – X213Cause parameter value

Return code		Meaning	ErrorSpecific information
Undefined	220	Undefined error situation	Not present
NSAPunreachablePerm	221	Connection Rejection – NSAP unreachable/fixed condition	Not present
DiscTrans	225	Disconnection – Transient condition	Not present
DiscPerm	226	Disconnection – Fixed condition	Not present
NoReasonTrans	227	Connection Rejection – Reason unspecified/transient condition	Not present
NoReasonPerm	228	Connection Rejection – Reason unspecified/fixed condition	Not present
QOSnotavailTrans	229	Connection Rejection – QOS not available/transient condition	Not present
QOSnotavailPerm	230	Connection Rejection – QOS not available/fixed condition	Not present
NSAPunreachableTrans	231	Connection Rejection – NSAP unreachable/transient condition	Not present
NSAPunknown	232	Connection Rejection – NSAP address unknown (fixed condition)	Not present
DiscNorm	241	Disconnection – Normal condition	Not present
DiscAbnorm	242	Disconnection – Abnormal condition	Not present
ConRejectTrans	244	Connection rejection – Transient condition	Not present
ConRejectPerm	245	Connection rejection – Fixed condition	Not present
ConRejectUserData	248	Connection rejection – Incompatible information in Userdata parameter	Not present

6.8 AttributeSet

6.8.1 AttributeSet parameters

Table 7 – User Plane Attribute Set (UAttributeSet) parameters

Parameter	Required	Comment
WindowSize	O	Layer 3 window size. See 6.3.32.
PacketSize	O	Layer 3 packet size. See 6.3.21.
FastSelect	O	Fast select facility. See 6.3.10.
QOSParameters	O	Quality of service. See 6.3.22.
UProtocol	O	See Remark. See also 6.3.28.
L3ConnectionMode	O	See Remark. See also 6.3.16.
L3TwoWayVCCount	O	See Remark. See also 6.3.19.
L3IncomingVCCount	O	See Remark. See also 6.3.17.
L3OutgoingVCCount	O	See Remark. See also 6.3.18.
TEI	O	See Remark. See also 6.3.27.
L2ConnectionMode	O	See Remark. See also 6.3.12.
L2WindowSize	O	See Remark. See also 6.3.14.
L2FrameSize	O	See Remark. See also 6.3.13.
L2XID	O	See Remark. See also 6.3.15.

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 – ACreateNCO operation – in [5] 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. Default values are described in Appendix I.

Table 8 – User Plane Address Set (UAddressSet) parameters

Parameter	Required	Comment
CalledDTEAddress	O	See 7.3.4 for parameter definition
CalledDTEAddressExt	O	See 7.3.5 for parameter definition
CallingDTEAddress	O	See 7.3.6 for parameter definition
CallingDTEAddressExt	O	See 7.3.7 for parameter definition

6.8.2 Static attribute content

The attribute sets described below use the following conventions:

– Name shall be used with the ACreateNCOReq message.

– All numeric values are in decimal.

Name:	U_ISO8208
WindowSize:	2
PacketSize:	128 (byte)
UProtocol:	ISO/IEC 8208
L3ConnectionMode:	DXE
L3TwoWayVCCount:	local arrangement
L3IncomingVCCount:	1
L3OutgoingVCCount:	1
L2ConnectionMode:	Auto
L2WindowSize:	7
L2FrameSize:	128 (byte)
L2XID:	none

Name:	U_TELEMATIC_TERM
WindowSize:	2
PacketSize:	128 (byte)
UProtocol:	T.90
L3ConnectionMode:	DXE
L3TwoWayVCCount:	local arrangement
L3IncomingVCCount:	0
L3OutgoingVCCount:	0
L2ConnectionMode:	Auto
L2WindowSize:	7
L2FrameSize:	128 (byte)

7 T.70NL protocol

7.1 Introduction

This clause deals with the T.70 protocol. In this part, whenever Recommendation T.70 is referenced, T.70NL is implied (Null Layer).

The OSI location of the T.70 protocol is shown in Figure 3.

General description conventions are provided in [5].

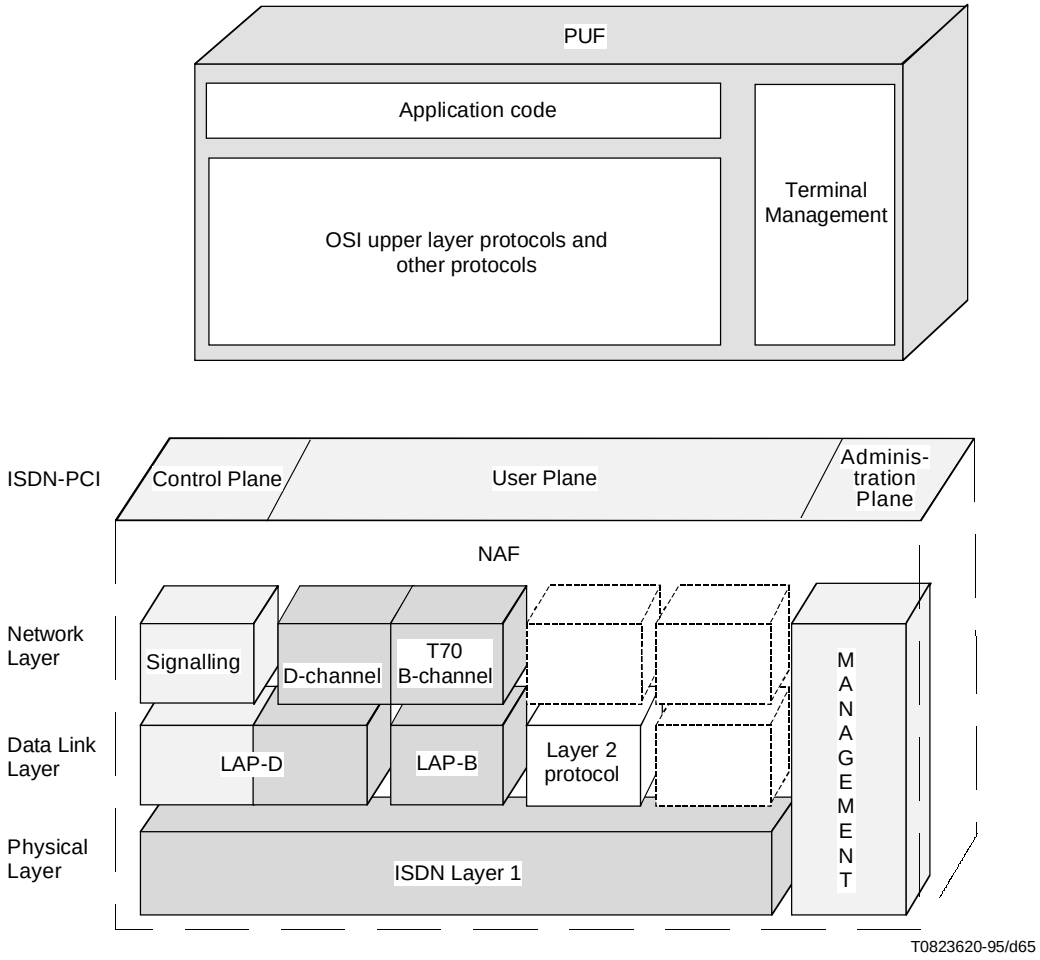


Figure 3 – OSI location

7.2 Messages

The User Plane messages provide an access to T.70 protocol stacks. Following is a list and short description of relevant User Plane messages. Table 9 gives an overview of these messages.

Table 9 – Overview of user messages

Message identifier	Class	Message name	Purpose of message
307	1	UDataReq	Request data transfer on an established user connection
308	1	UDataInd	Indicate arrival of transferred data on an established user connection

7.2.1 UDataReq

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 T.70 More Bit and Qualifier bit

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

Mandatory data shall be provided in the data buffer.

Related: UReadyToReceiveInd.

7.2.2 UDataInd

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 T.70 More Bit and Qualifier bit reception

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

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

Related: UReadyToReceiveReq.

7.3 Messages parameters

This subclause describes parameters for the T.70 User Plane. They are ordered by parameter identifiers. Information presentation is provided in [4].

Table 10 – Overview of user parameters

Parameter identifier	Parameter name	Use in user messages	Use in UAttributeSet	Other use
4	Bit_DQM	X		
50	NCOType			X
52	PacketSize		X	
62	UProtocol		X	
63	UAttributeName			
64	UDirection			X

7.3.1 Bit_DQM

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

– More Data bit value (bit 3).

Each information uses a binary position. The Most Significant Bit (MSB) is 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 the sum of the value for each bit (logical OR).

Type: 4.

Field	Field type	Direction	Required	Comment
DQM	Octet	B	M	Bit 1: 0 – Confirmation of data reception is not allowed or not required Bit 2: 0 – Reset Qualifier bit Bit 3: 1 – Set More bit 0 – Reset More bit

7.3.2 NCOType

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

Type: 50.

Field	Field type	Direction	Required	Comment
Identifier	Octet		M	U3 (2) – Network user access with NAF signalling coordination (NAF coordination functionality)

7.3.3 PacketSize

Description: This parameter is used to pass packet size information to/from the PUF.

Type: 52.

Field	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 12) Maximum size of data that can be received with UDataInd
Outvalue	Octet	B	M	Outbound maximum user data length (see Table 12) Maximum size of data that can be passed with UDataReq

Remark: This parameter is used to determine the maximum size of data buffers that can be passed with the UDataReq and UDataInd messages.

Table 11 – Precoded packet size values

Precoded value	Packet size (octet)	Precoded value	Packet size (octet)
4	16	8	256
5	32	9	512
6	64	10	1024
7	128	11	2048

7.3.4 UProtocol

Description: This is used to select the User Plane protocol.

Type: 62.

Field	Field type	Direction	Required	Comment
L3Protocol	Octet	P	M	T.70 (3)
L2Protocol	Octet	P	O	Default (255) – ISO/IEC 7776
L1Protocol	Octet	P	O	Default (255) – Transparent B-channel access

Remark: Other possible values are described in [6].

7.3.5 UAttributeName

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

Type: 63.

Field	Field type	Direction	Required	Comment
AttributeName	IA5-string	P	M	16 bytes is the maximum length

7.3.6 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.

Field	Field type	Direction	Required	Comment
Direction	Octet	P	O	both (3)

7.4 State diagram

User messages do not change the state of the connection.

7.5 Coordination function

The coordination function cannot be used.

7.6 Selection criteria

No T.70 specific parameters are used. General NCO criteria are provided in [5].

7.7 Specific error handling and codes

Errors are dealt with in the following manner.

7.8 Specific error handling and codes

Protocol errors are not available at the interface.

7.9 Static attributes

7.9.1 AttributeSet parameters

Table 12 – User Plane Attribute Set (UAttributeSet) parameters

Parameters	Required	Comment
UProtocol	O	See Remark. See also 7.3.4
L3PacketSize	O	See Remark. See also 7.3.3

Remark: It is only possible to use these parameters during NCO creation containing Control Plane information. Refer to subclause – ACreateNCO operation – in [5] for details.

If parameters are omitted, defaults shall be used by the NAF. Default values are described in Appendix I.

7.9.2 Static attribute content

Name:	U_T70
UProtocol:	T.70
L3PacketSize:	128

Appendix I

Configuration

I.1 T.90 protocol

Table I.1 – User Plane T.90 configuration

Parameters	Suggested default	Comment
X.25 Network Type	0	Allows NAF to adapt for different country implementations of X.25
Rec. X.25	CCITT88	Level of Rec. X.25 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	

Table I.1 – User Plane T.90 configuration *(end)*

Parameter	Suggested default	Comment
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/IEC 8208 [2] 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/IEC 7776 dce as defined in ISO/IEC 7776
Layer 2 B-channel modulus	8	NOTE – It shall be 128 for X.25 on D-channel
Layer 2 Window Size	7	
Layer 2 Frame Size	128	
Layer 2 activation type	Case1	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 seconds
– T1	1	Expressed in seconds
– N2	5	Maximum number of unsuccessful retransmissions

I.2 ISO/IEC 8208 protocol

Table I.2 – User Plane ISO/IEC 8208 configuration

Parameter	Suggested default	Comment
X.25 Network Type	0	Allows NAF to adapt for different country implementations of X.25
Rec. X.25	CCITT88	Level of Rec. X.25 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/IEC 8208 [2] 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/IEC 7776 dce as defined in ISO/IEC 7776
Layer 2 B-channel modulus	8	NOTE – It shall be 128 for X.25 on D-channel
Layer 2 Window Size	7	
Layer 2 Frame Size	128	
Layer 2 activation type	Case1	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 seconds
– T2	1	Expressed in seconds
– N2	5	Maximum number of unsuccessful retransmissions

I.3 T.70 protocol

Table I.3 – User Plane T.70 configuration

Parameter	Suggested default	Comment
Layer 3 Maximum Packet Size	2048	
Layer 3 Default Packet Size	128	
Layer 2 Timers – T1 – T1 – N2	 5 1 5	 Expressed in seconds Expressed in seconds Maximum number of unsuccessful retransmissions

Appendix II

NAF-SDL diagrams

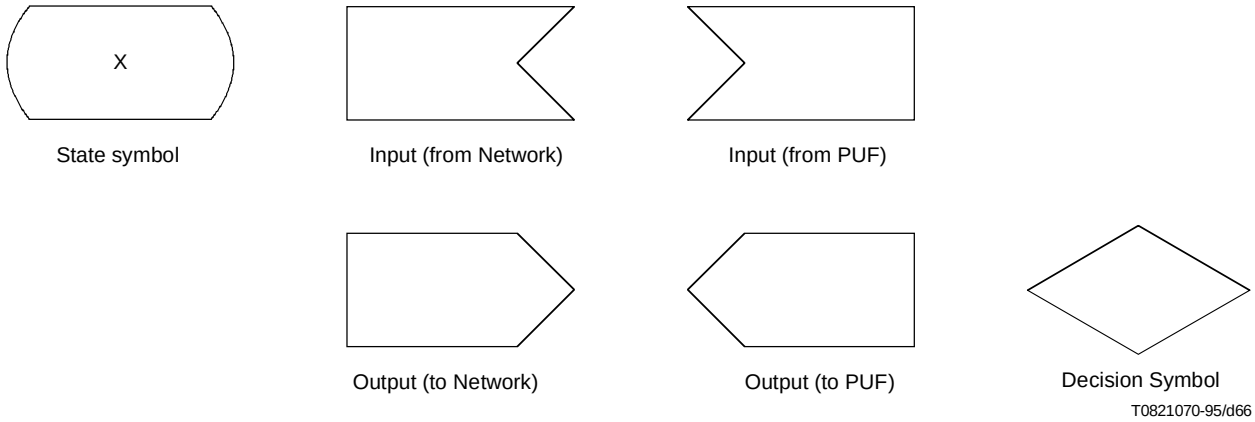
The mapping of User Plane messages to protocol messages depends on whether the NAF is providing the coordination function for a particular Control Plane connection.

When the NAF is providing the coordination function, the mapping of X.213 service primitives to Q.931 [3] messages and X.25 packets is explained in ISO/IEC 9574 and ISO/IEC 8878.

When the NAF is not providing the coordination function, the mapping of X.213 service primitives to X.25 packets is explained in ISO/IEC 8878.

Some SDL diagrams are given to explain more clearly the relation between user messages and network primitives. These diagrams do not cover every case. They only present some of the possible cases.

The following symbols are used within this description. A full description of the symbols and their meaning is given in Recommendation Z.100.



II.1 T.90 protocol

Table II.1 shows the mapping of User Plane messages to X.213 service primitives.

Table II.1 – Mapping between User Plane message

PCI message	X.213 primitive
UConnectReq	N-CONNECT request
UConnectInd	N-CONNECT indication
UConnectRsp	N-CONNECT response
UConnectCnf	N-CONNECT confirm
UDisconnectReq	N-DISCONNECT request
UDisconnectInd	N-DISCONNECT indication
UDataReq	N-DATA request
UDataInd	N-DATA indication
UResetReq	N-RESET request
UResetInd	N-RESET indication
UResetRsp	N-RESET response
UResetCnf	N-RESET confirm
UReadyToReceiveReq	Not equivalent to an X.213 primitive
UReadyToReceiveInd	Not equivalent to an X.213 primitive

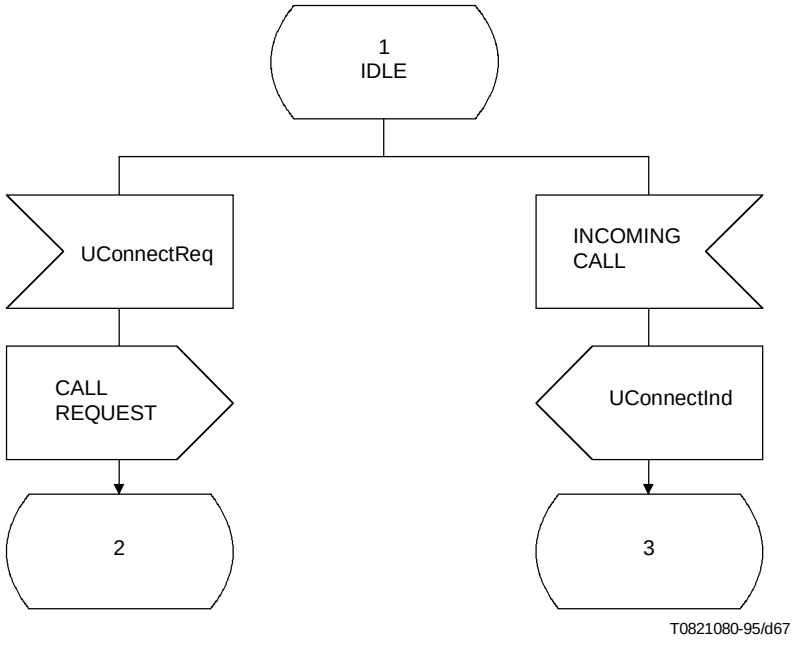


Figure II.1 – Idle

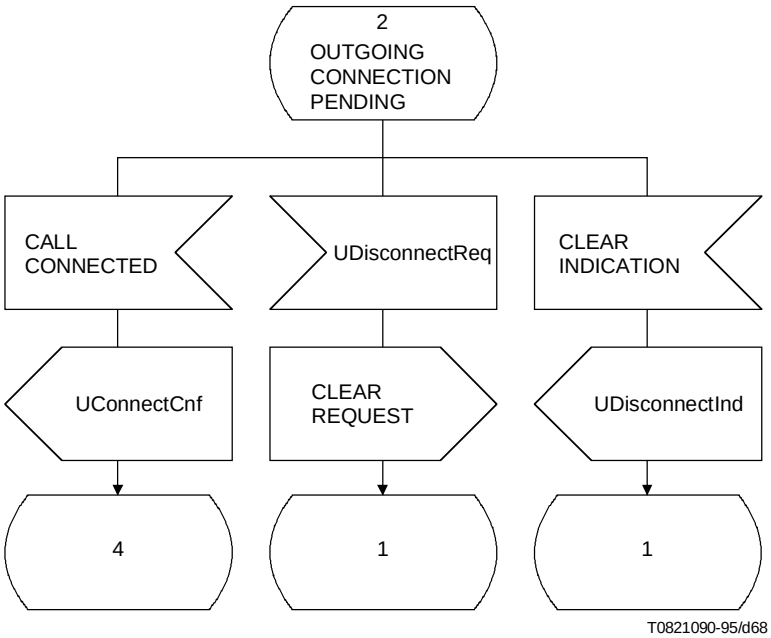


Figure II.2 – Outgoing connection pending

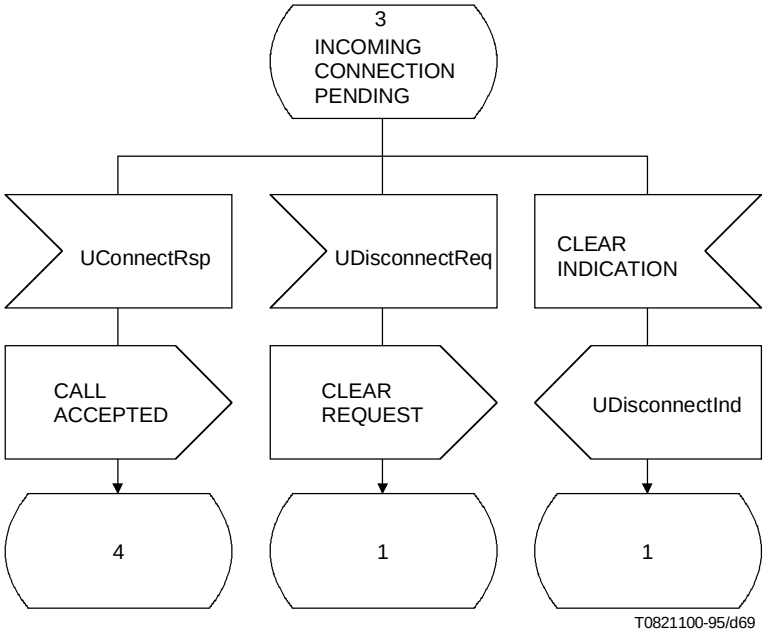


Figure II.3 – Incoming connection pending

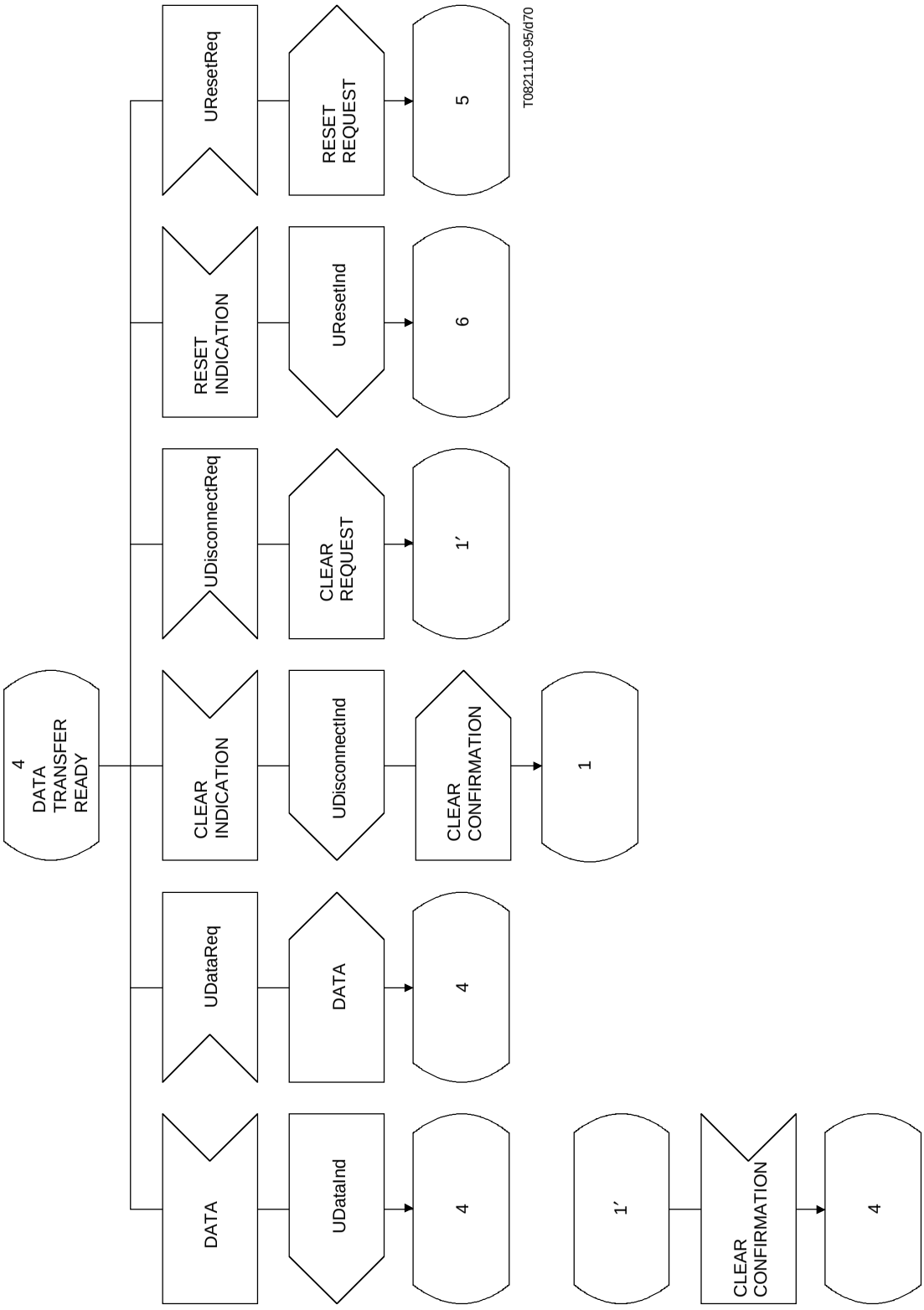


Figure II.4 – Data transfer ready

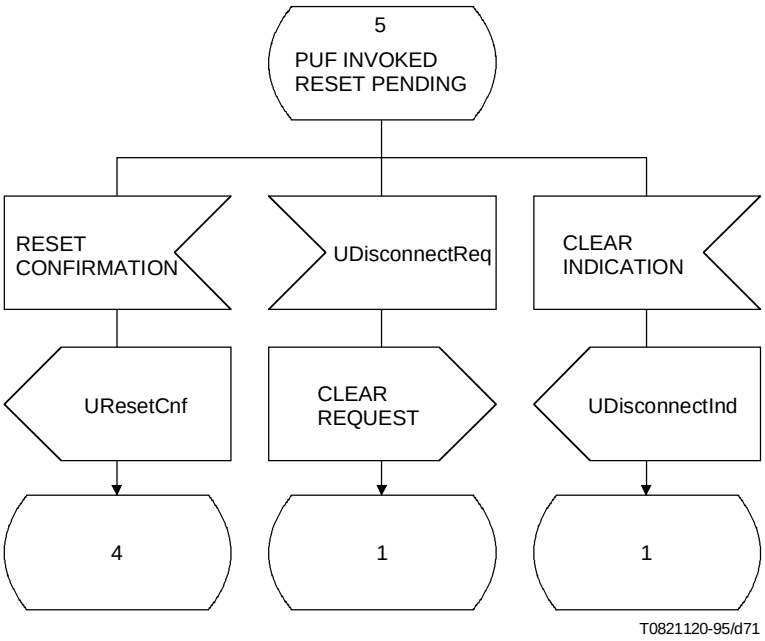


Figure II.5 – PUF invoked reset pending

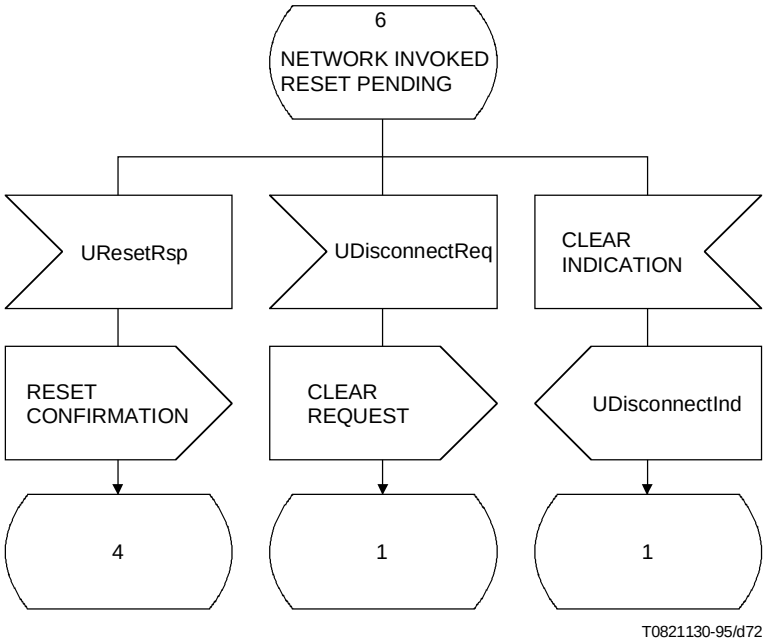


Figure II.6 – Network invoked reset pending

II.2 ISO/IEC 8208 protocol

Table II.2 shows the mapping of User Plane messages to X.213 service primitives.

Table II.2 – Mapping between User Plane message and protocol messages

PCI message	X.213 primitive
UConnectReq	N-CONNECT request
UConnectInd	N-CONNECT indication
UConnectRsp	N-CONNECT response
UConnectCnf	N-CONNECT confirm
UDisconnectReq	N-DISCONNECT request
UDisconnectInd	N-DISCONNECT indication
UDataReq	N-DATA request
UDataInd	N-DATA indication
UExpeditedDataReq	N-EXPEDITED-DATA request
UExpeditedDataInd	N-EXPEDITED-DATA indication
UResetReq	N-RESET request
UResetInd	N-RESET indication
UResetRsp	N-RESET response
UResetCnf	N-RESET confirm
UDataAcknowledgeReq	N-DATA-ACKNOWLEDGE request
UDataAcknowledgeInd	N-DATA-ACKNOWLEDGE indication
UReadyToReceiveReq	Not equivalent to an X.213 primitive
UReadyToReceiveInd	Not equivalent to an X.213 primitive

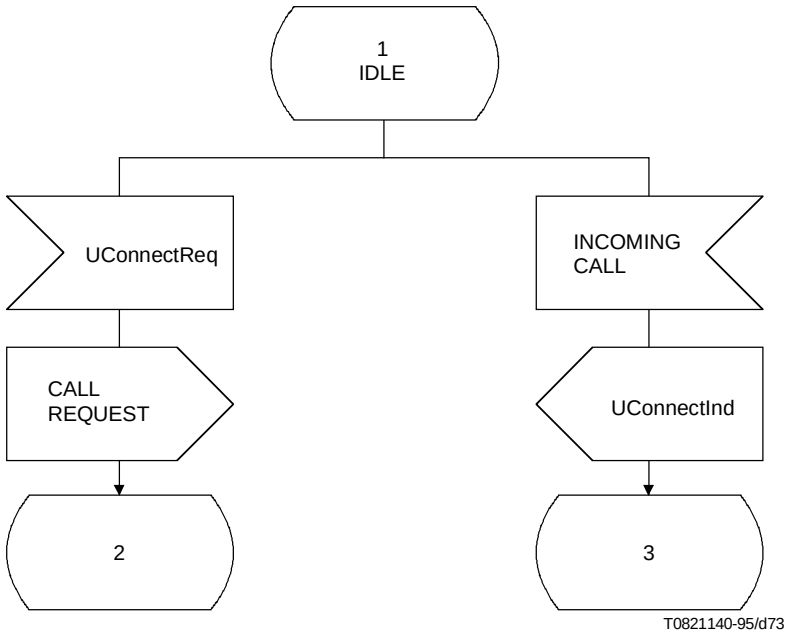


Figure II.7 – Idle

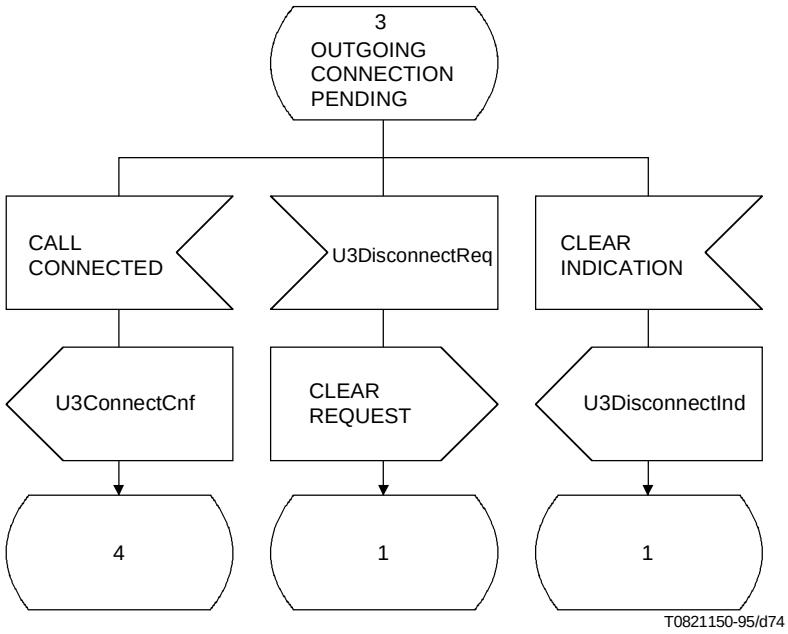


Figure II.8 – Outgoing connection pending

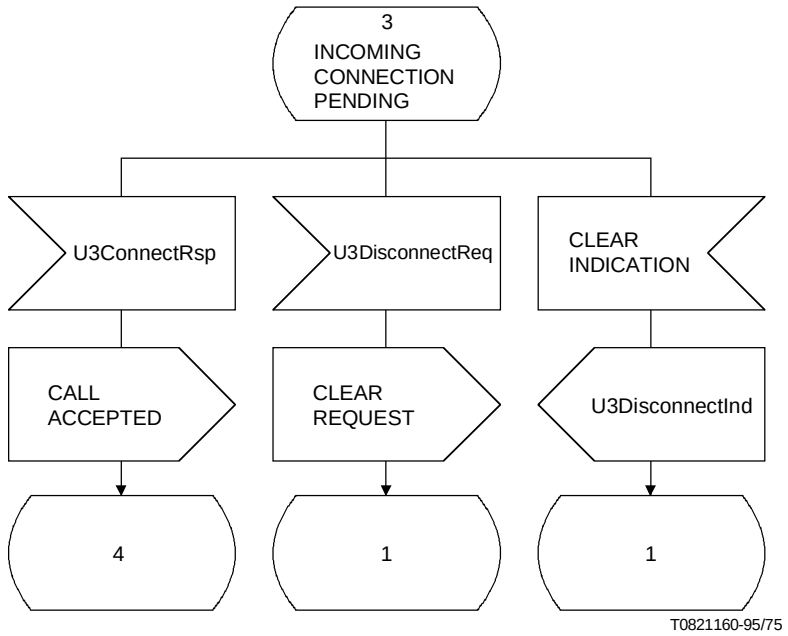


Figure II.9 – Incoming connection pending

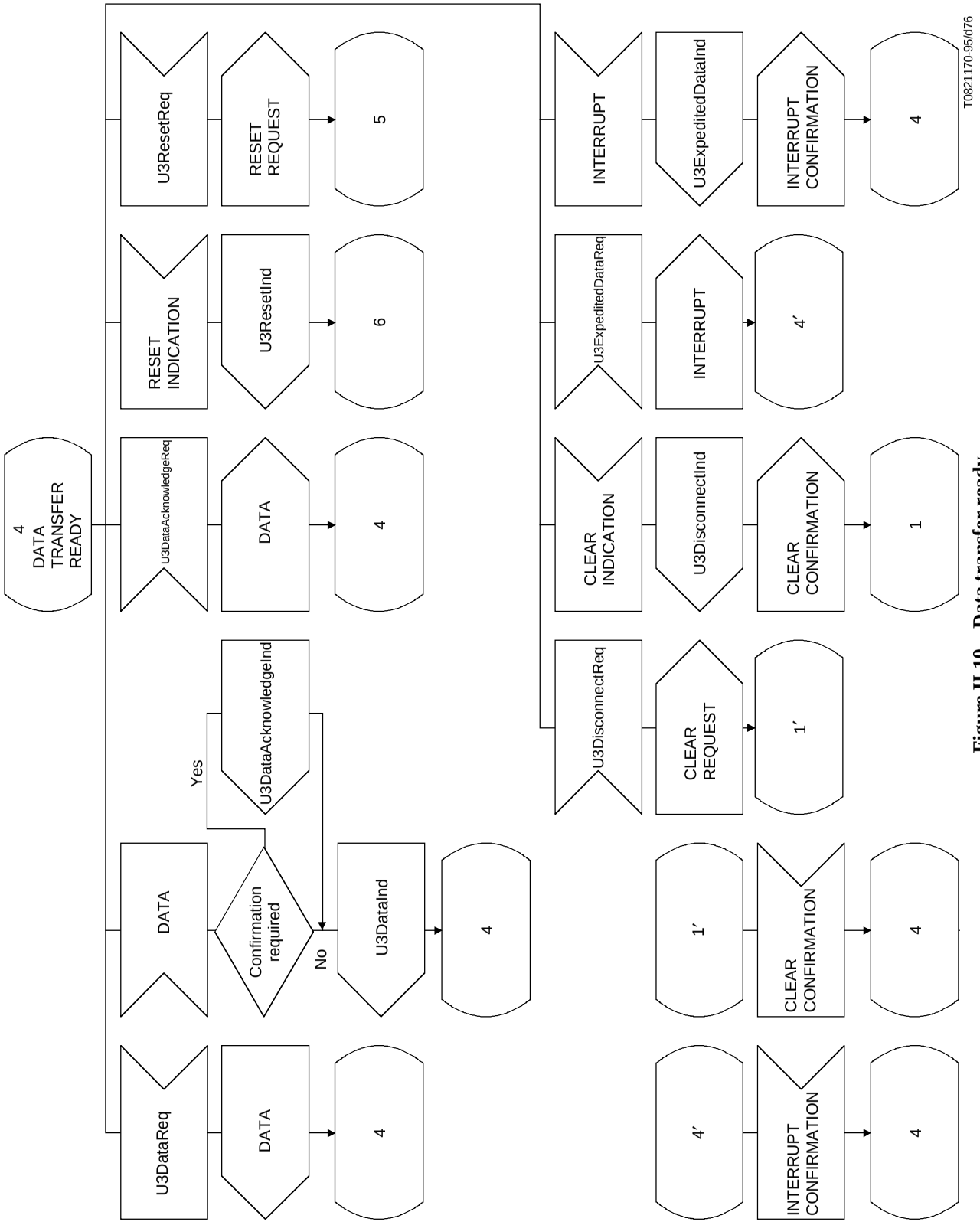


Figure II.10 – Data transfer ready

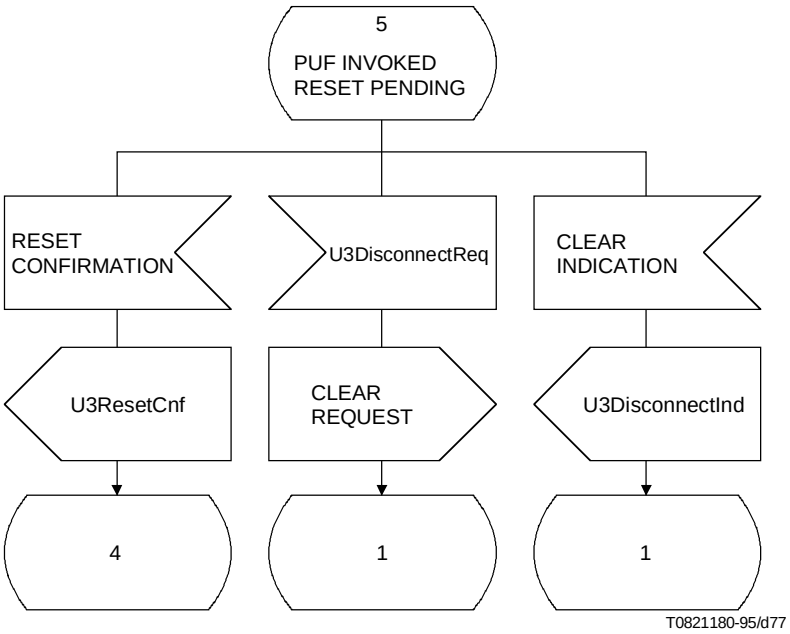


Figure II.11 – PUF invoked reset pending

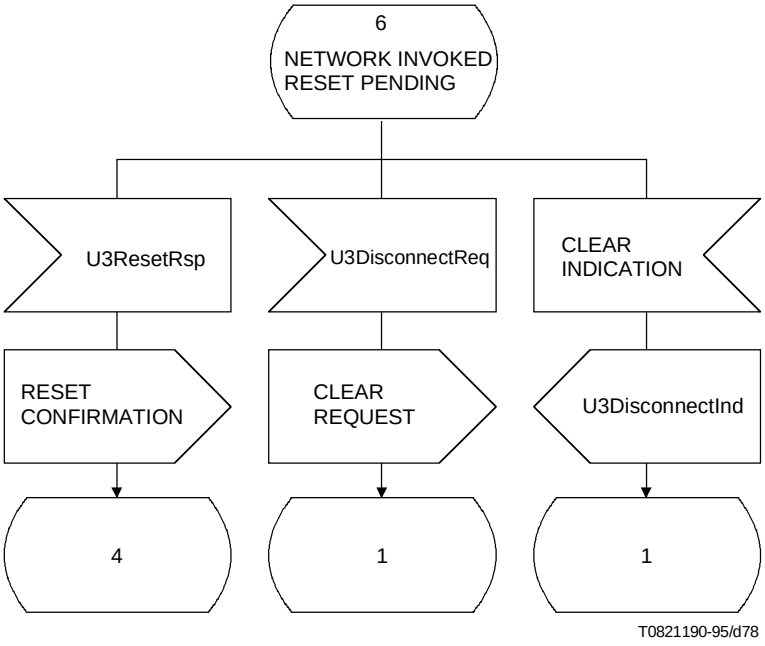


Figure II.12 – Network invoked reset pending

Appendix III

X.25 usage

III.1 Parameter Values for Recommendation X.25 use

Table III.1 shows the required setting of parameters to achieve different types of Recommendation X.31 operation.

Table III.1 – Types of 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	X.25	Not required	Not required	Not required
X.31 Case B, B-channel permanent	X.25	B-channel	Required	Not required
X.31 Case B, D-channel	X.25	D-channel	Required	Not required

III.2 Disconnection of ISDN channel with established Recommendation X.25 connections

In the coordination case, this is covered by ISO/IEC 9574.

In the non-coordination case, the following should be provided to the PUF:

- CDisconnectInd message with cause for channel disconnection;
- For each established Recommendation X.25 Connection:
  - U3DisconnectInd message with X213Cause and X213Origin as defined by ISO/IEC 9574 and X25Cause.
- For each Recommendation X.25 Connection in the process of being established:
  - U3DisconnectInd message with X213Cause and X213Origin as defined by ISO/IEC 9574 and X25Cause.

Superseded by a more recent version

CONTENTS

PART 7

PART 7: EXCHANGE MECHANISM DOS

Summary

This part of the specification defines all details of the Operating System binding for a MS-DOS™ environment (a general presentation of the binding mechanism can be found in Part 2.

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 specification defines the ITU-T ISDN Application Programming Interface (API), called ISDN Programming Communication Interface (PCI). The ISDN-PCI is an application interface for accessing and administering ISDN.

It has been defined in order to provide a standard for terminal equipment providers that makes possible 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 future ISDN extensions will not affect the operation of existing applications.

Superseded by a more recent version

1 Scope

This part specifies the Integrated Services Digital Network Programming Communication Interface (ISDN-PCI) exchange mechanism for the MS-DOS™ Operating System. It forms a part of the specification on ISDN-PCI.

It describes the way a PUF or a NAF, as described in Part 2, should dialogue, exchange messages and parameters to make ISDN connection.

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

2 References

[1] Part 1, *General architecture.*

[2] Part 2, *Basic services.*

3 Definitions

This part defines the following terms:

3.1 **exchange function**: PUF functionality realizing the exchange mechanism.

3.2 **exchange mechanism**: Means provided for the PUF to interchange messages with the NAF.

3.3 **ISDN programming communication interface (ISDN-PCI)**: Network (ISDN) oriented software interface providing access provisions for programming network signalling and user data exchange.

3.4 **message**: Unit of information transferred through the interface between the Network Access Facility (NAF) and the PCI User Facility (PUF).

3.5 **network access facility (NAF)**: Functional unit located between the ISDN-PCI and the network related layers.

3.6 **PCI user facility (PUF)**: Functional unit using the ISDN-PCI to access a NAF. In fact, the local application using the interface.

4 Abbreviations

This part uses the following abbreviations:

API	Application Programming Interface
DOS	stands for operating systems compatible to the MS-DOS operating system
ISDN	Integrated Services Digital Network
MS-DOS	Trade Mark of Microsoft Corporation, INC
NAF	Network Access Facility
PCI	Programming Communication Interface
PciMPB	Pci Message Parameter Block
PUF	Programming communication interface User Facility

5 Reader’s guidance

5.1 Reader’s guide

This part is intended for software developers, implementors of applications and equipment manufacturers by providing them the exchange mechanims description and coding examples as described in [2] for the DOS.

5.2 How to use this part

Readers who:

- need a quick overview over the exchange mechanism in general need to refer to Part 2: "Basic services" [2]. Other operating systems are described. The reader should consult Part 1: "General description" [1] to get information on other Operating System availability. General description of the exchange mechanism for DOS is located in 6.1;
- intend to implement an application using this ISDN-PCI interface for DOS should inspect clauses 5 and 6. Clauses 3 and 4 provide useful information on the definitions of terms and abbreviations used. Coding examples are provided in Appendix I. To get information on parameters and return codes list and description, the reader should refer to Part 2: "Basic services" [2];
- intend to build an ISDN adapter card or equipment should also first read clauses 5 and 6. Clauses 3 and 4 provide useful information on the definitions of terms and abbreviations used. For more detailed information regarding the NAF is contained in 6.4. Coding examples provided in Appendix I show how a PUF may access a NAF. To get information on parameters and return codes list and description, the reader should refer to Part 2: "Basic services" [2].

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

Table 1 – List of contents

Clause, Appendix	Contains ...
Clause 1	... the scope of this part. This describes what this part covers
Clause 2	... references
Clause 3	... definitions of the terms used throughout this part
Clause 4	... definitions of the abbreviations used throughout this part
Clause 5	... gives an overview and reader's guidance
Clause 6	... operating system dependencies and implementation rules for DOS
Appendix I	... sample coding in C-language illustrating operating system specific implementation of the exchange mechanism for DOS

6 DOS Operation System specific implementation

This part describes the operating system specific implementation for the DOS operating system. For the following description, the base MS-DOS version is the version number 3.1.

A NAF implementation under DOS shall offer the functionality of the exchange functions described in a generic way in Part 2 [2].

In this part, the mapping and implementation of these functions are described on a function per function basis. For each function, a coding example in C-language is given.

6.1 Introduction

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 may 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. There the signature is located, which shall contain the eight character constant "ISDN-PCI". If this signature is available, the PUF assumes 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 has to 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 PCI-MPB generic structure is **byte**.

6.2 Mapping of generic types and constants

Under DOS, the following mapping shall be used for the generic types described in Part 2: "Basic Services" [2]:

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 */
```

6.3 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 described for simplicity in the coding examples. However, the access of a PUF to multiple NAFs is not excluded.

6.3.1 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 4096. 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 to:

- 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 an 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.

C-coding example:

```
...
#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 PCIHandlesArray[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;
                }
...
```

6.3.2 PciGetProperty

This function is in charge to retrieve the NAF-Property from the NAF. To issue the function call, the PUF must possess the PCI-Handle of the NAF it wants to access. Before accessing the NAF, the PUF may 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:

- may examine memory area pointed to by the PCIHandle to find out if NAF is loaded and check the signature for the character constant "ISDN-PCI" in that case;
- 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 = ...

...
```

6.3.3 PciRegister

This function is in charge to provide an association between a PUF and a NAF. To issue the function call, the PUF must possess the PCI-Handle of the NAF it wants to access. Before accessing the NAF, the PUF may 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, two 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 Part 2 [2]. The second is the operating system dependent PCIOpSysInfo structure, which for DOS has the following layout:

Element name	Type	Validity	Explanation
MaxNCOCount	2-byte integer	On call	Shall be set to the maximum amount of NCOs the PUF intends to create during the association
MaxPacketSize	2-byte integer	On call	Shall be set to the maximum size of a data packet the NAF shall accept on a user connection
MaxPacketCount	2-byte integer	On 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	On 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)	On call	If the structure element AddBufferSize is non-zero, this element shall point (far) to the donated, additional buffer space.
BufferNeeded	4-byte integer	On return	In case the NAF has not 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 case a NAF has not enough memory resources available to fulfill the requested characteristics, the PciRegister function will fail and return a BuffersTooSmall error code. In this case the amount of buffer missing can be taken from the BufferNeeded element of the above structure.

On the 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 to:

- Examine memory area pointed to by the PCIHandle to find out if NAF is loaded. Check the signature for characters "ISDN-PCI".
- Allocate and setup 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.

C-coding example:

```
...
#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       148     /* BuffersTooSmall 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 int       MaxNCOCount;            /* optional: give max count of NCOs */
        short int       MaxPacketSize;          /* optional: give expected max size and */
        short int       MaxPacketCount;         /* 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 {
        2,                      /* Set PUF version to 2, equaling current Recommendation Version */
        0,                      /* Set PUF type to 0 as indicated in [2] */
        0                       /* Initialize (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                       /* Initialize (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´s 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 */
            ...
…
```

6.3.4 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);
...
```

6.3.5 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 to:

- 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_mpb far * PCIMpb;
PCI_BYTEARRAY Message;
PCI_BYTEARRAY Data;
{
PCI_INTEGER error;

/* call the entry point */
error = (*PCIHandle) (PCIPUTMESSAGE, ExchangeID, PCIMpb, Message, Data);
...
}
```

6.3.6 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 to:

- 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_mpb far * PCIMpb;
PCI_BYTEARRAY Message;
PCI_BYTEARRAY Data;
{
PCI_INTEGER error;

/* call the entry point */
error = (*PCIHandle) (PCIGETMESSAGE, ExchangeID, PCIMpb, Message, Data);
...
}
```

6.3.7 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 to:

- 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 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 has to be preserved by the PUF.

6.4 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.

6.4.1 Declaration action

Under DOS, 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 will take place in order to:

- 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 will 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).

6.4.2 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 will take place in order to:

- 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 will 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).

Appendix I

DOS Operating System implementation coding samples

These samples present a way to implement the exchange mechanism function call from the PUF point of view.

```
/*****************************************************************************************************
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 a 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 int       MaxNCOCount;            /* optional: give max count of NCOs */
        short int       MaxPacketSize;          /* optional: give expected max size and */
        short int       MaxPacketCount;         /* 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 descriptor */
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 occured; 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 with 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 * PCIMPB,
        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 set up 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_mpb far *) PCIMPB,
            (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 * PCIMPB,
            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_mpb far *) PCIMPB,
            (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 set up 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() */
```

Superseded by a more recent version

CONTENTS

PART 8

PART 8: WINDOWS EXCHANGE MECHANISM

Summary

This part of the multi-part specification defines all details of the Operating System binding for a Windows™ environment (a general presentation of the binding mechanism can be found in Part 2).

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 Specification defines the ITU-T ISDN Application Programming Interface (API), called ISDN Programming Communication Interface (PCI). The ISDN-PCI is an application interface for accessing and administering ISDN.

It has been defined in order to provide a standard for terminal equipment providers that make possible 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 future ISDN extensions not affect the operation of existing applications.

Superseded by a more recent version

1 Scope

This part specifies the Integrated Services Digital Network Programming Communication Interface (ISDN-PCI) exchange mechanism for the Windows Operating System. It forms a part of the specification on ISDN-PCI.

It describes the way a PUF or a NAF as described in Part 2: "Basic services", should exchange messages and parameters to make ISDN connection.

Further specifications specify the method of testing and detailled application specific requirements to determine conformance based on this part.

2 References

[1] Part 1, *General architecture.*

[2] Part 2, *Basic services.*

3 Definitions

This part defines the following terms:

3.1 **exchange function**: PUF functionality realizing the exchange mechanism.

3.2 **exchange mechanism**: Means provided for the PUF to interchange messages with the NAF.

3.3 **ISDN programming communication interface (ISDN-PCI)**: ISDN oriented software interface providing access provisions for programming network signalling and user data exchange.

3.4 **message**: Unit of information transferred through the interface between the Network Access Facility (NAF) and the PCI User Facility (PUF).

3.5 **network access facility (NAF)**: Functional unit located between the ISDN-PCI and the network related layers.

3.6 **PCI user facility (PUF)**: Functional unit using the ISDN-PCI to access a NAF. E.g. the local application using the interface.

4 Abbreviations

This part uses the following abbreviations:

API	Application Programming Interface
ISDN	Integrated Services Digital Network
NAF	Network Access Facility
PCI	Programming Communication Interface
PciMPB	Pci Message Parameter Block
PUF	Programming communication interface User Facility
WINDOWS	Stands for the Windows™ Operating Systems based on version 3.0. Windows is Trade Mark of Microsoft Corporation, Inc.

5 Reader’s guidance

5.1 Reader’s guide

This part is intended for software developers, implementors of applications and equipment manufacturers by providing them the exchange mechanisms description and coding examples as described in [2] for the Windows™ Operating System.

5.2 How to use this part

Readers who:

- need a quick overview over the exchange mechanism in general should refer to Part 2: "Basic services" [2]. Other operating systems are described. The readers should consult Part 1: "General architecture" [1] to get information on other Operating System availability. General description of the exchange mechanism for Windows™ is located in 6.1;
- intend to implement an application using this ISDN-PCI interface for Windows™ should inspect clauses 5 and 6. Clauses 3 and 4 provide useful information on the definitions of terms and abbreviations used. Coding examples are provided in Appendix I. To get information on parameters and return codes list and description, the reader should refer to Part 2: "Basic services" [2];
- intend to build an ISDN adapter card or equipment should also first read clauses 5 and 6. Clauses 3 and 4 provide useful information on the definitions of terms and abbreviations used. More detailed information regarding the NAF is contained in 6.4. Coding examples provided in Appendix I show how a PUF may access a NAF. To get information on parameters and return codes list and description, the reader should refer to Part 2: "Basic services" [2].

Table 1 provides a list showing the major clauses of this part.

Table 1 – List of contents

Clause, Appendix	Contains ...
Clause 1	... the scope of this part. This describes what this part covers
Clause 2	... references
Clause 3	... definitions of the terms used throughout this part
Clause 4	... definitions of the abbreviations used throughout this part
Clause 5	... gives an overview and reader's guidance
Clause 6	... operating system dependencies and implementation rules for Windows™
Appendix I	... sample coding in C-language illustrating operating system specific implementation of the exchange mechanism for Windows™

6 Windows Operating System specific implementation

6.1 Introduction

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 has to be DLL and has to export an entry point per ISDN-PCI function using the same name (PciGetProperty, PciRegister, PciGetMessage, PciPutMessage, PciSetSignal, PciDeregister).

NOTE – Function names exported by the NAF are the same as the description made in Part 2 [2] 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 to provide transparently to the PUF inside the Pci's exchange mechanism functions as shown in Appendix I.

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 needs also 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.

Under Windows, 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 in that case.

Pointer parameters are far. The PCI-MPB structure is always passed via a pointer. The Exchange Identifier structure is always passed via a pointer too.

The structure alignment is **byte**.

6.2 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

6.3 C-structures and 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 *PCIMPB,
                                            PCI_BYTEARRAY Message,
                                            PCI_BYTEARRAY Data);

PCI_INTEGER far PASCAL PciGetMessage (      PCI_EXID far *ExID,
                                            struct pci_mpb far *PCIMPB,
                                            PCI_BYTEARRAY Message,
                                            PCI_BYTEARRAY Data);

PCI_INTEGER far PASCAL PciSetSignal (       PCI_EXID far *ExID,
                                            PCI_INTEGER Signal,
                                            PCI_PROCEDURE SignalProcedure);
```

6.4 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 must be at least 1024 bytes deep.

6.4.1 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 entrys 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 constructs before;

  no default value;

  a maximum size equal to 128;

  FileName = 'PCI.INI'.

6.4.2 PciGetProperty

This function is in charge to provide the PUF with the PROPERTY of the NAF. Implicitly it checks if the NAF is available, when loading the library via the LoadLibrary function.

The following operations shall take place, in order to:

- 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.

6.4.3 PciRegister

This function is in charge to provide 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 according to the following structure:

- PUFType (PCI_INTEGER)
- PUFVersion (PCI_INTEGER)
- MaxMsgSize (PCI_INTEGER) where the NAF will give the maximum size for a message.

The following operations shall take place, in order to:

- 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.

6.4.4 PciDeregister

This function is in charge to disassociate 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 to:

- 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.

6.4.5 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 to:

- 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).

6.4.6 PciGetMessage

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. Buffers provided by the PUF are directly used by the NAF.

The following operations shall take place, in order to:

- 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).

6.4.7 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 mechanisms are offered under Windows:

- a signal procedure mechanism;
- a user message mechanism.

Once a mechanism is chosen by the PUF, the other is desactivated by the NAF for that particular PUF. Both mechanisms have to be supported by a NAF.

The first mechanism does not use 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 must not be equal to 0.

6.4.7.1 Signal mechanism procedure

The routine address, provided by the PUF in the SignalProcedure parameter, is used directly by the NAF. It has 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 the PUF's one. The SignalProcedure must be compiled without assuming SS equal to DS, i.e. as a DLL.

The NAF is allowed to call the PUF to reissue a signal call. To avoid big stack requirement, the NAF has to 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 is not allowed. The stack size is not guaranty when the NAF calls the PUF. Consequently, the stack requirements for the PUF treatment have to be as small as possible.

6.4.7.2 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 must 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 point out:

- 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.

6.4.7.3 Desactivation mechanism

To deactivate any signal mechanism the PciSetSignal function Signal and SignalProcedure parameters shall be set to NULL. Once deactivated, the previous mechanism shall no longer be used by the NAF to call the PUF.

6.5 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.

6.5.1 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 6.4.1).

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;
                }
        }
```

The maximum number of NAF that can be registered is 32.

6.5.2 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");
                        }
                }
        }
```

Appendix I

WINDOWS Operating System implementation coding samples

These samples present a way to implement the exchange mechanism function call from the PUF point of view.

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 *PCIMPB,
                                            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, PCIMPB, Message, Data);
        }


/*
//////////////////////////////////////////////////////////////////////////////////////
/// PciGetMessage()
*/
PCI_INTEGER far PASCAL PciGetMessage(       PCI_EXID far *ExID,
                                            PCI_MPB far *PCIMPB,
                                            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, PCIMPB, 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);
        }
```

CONTENTS

PART 9

Page

PART 9: UNIX EXCHANGE MECHANISM

Summary

This part of the multi-part specification defines all details of the operating system binding for a UNIX™ environment (a general presentation of the binding mechanism can be found in Part 2).

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 specification defines the ITU-T ISDN Application Programming Interface (API), called ISDN Programming Communication Interface (PCI). The ISDN-PCI is an application interface for accessing and administering ISDN.

It has been defined in order to provide a standard for terminal equipment providers that makes possible 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 future ISDN extensions will not affect the operation of existing applications.

Superseded by a more recent version

1 Scope

This part specifies the Integrated Services Digital Network Programming Communication Interface (ISDN-PCI) exchange mechanism for the UNIX operating system. It forms a part of the specification on ISDN-PCI.

It describes the way a PUF or a NAF as described in Part 2: "Basic services", should dialogue, exchange messages and parameters to make ISDN connection.

Further specifications specify the method of testing and detailled application specific requirements to determine conformance based on this part.

2 References

[1] Part 1, *General architecture*.

[2] Part 2, *Basic services.*

3 Definitions

This part defines the following terms:

3.1 **exchange function**: PUF functionality realising the exchange mechanism.

3.2 **exchange mechanism**: Means provided for the PUF to interchange messages with the NAF.

3.3 **ISDN programming communication interface (ISDN-PCI)**: Network (ISDN) oriented software interface providing access provisions for programming network signalling and user data exchange.

3.4 **message**: Unit of information transferred through the interface between the Network Access Facility (NAF) and the PCI User Facility (PUF).

3.5 **network access facility (NAF)**: Functional unit located between the ISDN-PCI and the network related layers.

3.6 **PCI User Facility (PUF)**: Functional unit using the ISDN-PCI to access a NAF. In fact, the local application using the interface.

4 Abbreviations

This part uses the following abbreviations:

API	Application Programming Interface
ISDN	Integrated Services Digital Network
NAF	Network Access Facility
PCI	Programming Communication Interface
PciMPB	Pci Message Parameter Block
PUF	Programming communication interface User Facility
UNIX	Stands for operating systems compatible to the UNIX operating system

5 Reader’s guidance

5.1 Reader’s guide

This part is intended for software developers, implementors of applications and equipment manufacturers by providing them the exchange mechanism description and coding examples as described in [2] for the UNIX operating system. An understanding of the STREAMS concept is a help to get the details of the subsequent mechanism description.

5.2 How to use this part

Readers who:

- need a quick overview over the exchange mechanism in general should refer to Part 2: "Basic services" [2]. Other operating systems are described. The reader should consult Part 1: "General architecture" [1] to get information on other operating system availability. General description of the exchange mechanism for UNIX is located in 6.1;

- intend to implement an application using this ISDN-PCI interface for UNIX should inspect clauses 5 and 6. Clauses 3 and 4 provide useful information on the definitions of terms and abbreviations used. Coding examples are provided in Appendix I. To get information on parameters and return codes list and description, the reader should refer to Part 2: "Basic services" [2];

- intend to build an ISDN adapter card or equipment should also first read clauses 5 and 6. Clauses 3 and 4 provide useful information on the definitions of terms and abbreviations used. More detailed information regarding the NAF is contained in 6.5 and 6.6. Coding examples provided in Appendix I show how a PUF may access a NAF. To get information on parameters and return codes list and description, the reader should refer to Part 2: "Basic services" [2].

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

Table 1 – List of contents

Clause, Appendix	Contains ...
Clause 1	... the scope of this part. This describes what this part covers
Clause 2	... references
Clause 3	... definitions of the terms used throughout this part
Clause 4	... definitions of the abbreviations used throughout this part
Clause 5	... gives an overview and reader’s guidance
Clause 6	... operating system dependencies and implementation rules for UNIX
Appendix I	... sample coding in C-language illustrating operating system specific implementation of the exchange mechanism for UNIX

6 UNIX operating system specific implementation

6.1 Introduction

The PCI exchange functions described in Part 2: "Basic services" [2] have to be mapped to appropriate functions supplied by the UNIX STREAMS kernel mechanism.

The binary compatible interface to a NAF running under the UNIX operating system shall be implemented using the STREAMS kernel mechanism.

Descriptions were made using C-language because it is the natural language in the UNIX environment.

6.2 Implementation of basic types

The following table shows the mapping of the basic types of the exchange method to C-language types:

Basic 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 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

6.3 Parameter passing conventions

For parameter passing, the usual C-conventions apply:

- Call values are either passed by value (e.g. PCI_INTEGER, PCI_EXID), or by use 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 occurred inside the NAF driver are returned as positive integers (PCI_INTEGER). Their values are defined in [2]. As defined there, a value of 0 stands for "no error" (Success).

Errors occurred inside the PCI exchange functions should be returned as negative integers (PCI_INTEGER). Their values are not defined. They are NAF implementation dependent.

6.4 Definition of types, constants and function-prototypes

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 *    PCIMPB,
                                PCI_BYTEARRAY       Message,
                                PCI_BYTEARRAY       Data);

PCI_INTEGER PciGetMessage (     PCI_EXID            ExID,
                                struct pci_mpb *    PCIMPB,
                                PCI_BYTEARRAY       Message,
                                PCI_BYTEARRAY       Data);

PCI_INTEGER PciSetSignal (      PCI_EXID            ExID,
                                PCI_INTEGER         Signal,
                                PCI_PROCEDURE       SignalProcedure);
```

6.5 Adaptation to the STREAMS kernel mechanism

6.5.1 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.

6.5.2 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)

6.5.3 Special considerations

Several NAF implementation aspects have to 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 acheived 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 4096 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 4096 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 4096 bytes arrive at the stream head, a call to PciGetMessage will return the first 4096 bytes of the data block and successive calls to PciGetMessage will return the additional data blocks. Each of the additional calls to PciGetMessage will return a message whose control part length will be zero.
- Only the UNIX SIGPOLL signal shall be issued by the NAF implementation.

6.6 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 three 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               */
        }
```

6.6.1 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, will 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.

6.6.2 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 an 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_timeout     = 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;
}
```

6.6.3 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 [2]. 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 [2].

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 and 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_timeout         = 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;
}
```

6.6.4 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 will be set to indicate the specific error condition. Any other return value indicates success.

```
strioctl.ic_cmd         = PCI_DEREGISTER;
strioctl.ic_timeout     = 0;
strioctl.ic_len         = 0;
strioctl.ic_dp          = (char *) NULL;

if (ioctl (*ExID, I_STR, &strioctl) == –1) {
        return errno;
}
else {
        close (*ExID);
        return 0;
}
```

6.6.5 PciPutMessage

Function body:

```
PCI_INTEGER PciPutMessage(      PCI_EXID                ExID,
                                struct  pci_mpb         *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. PCI-MPB 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 PCI-MPB 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 PCIMPB followed by the content of Message, and in
databuf the content of Data */
if (Message && ((char *)Message != (char *)PCIMPB + sizeof(pci_mpb))) {
        /*  There is a Message not NULL, and PCIMPB and Message are not contiguous in memory,
            Have to build a buffer where PCIMPB is followed by the Message content */
        char  *buffer; /* pointer to a buffer, large enough to receive PCIMPB and  the Message content */
        ...
        /* Here a memory allocation process may take place */
        ...
        memcpy (buffer, PCIMPB, sizeof(pci_mpb));
        memcpy ((buffer + sizeof(pci_mpb), Message, PCIMPB->MessageActualUsedSize);
        ctlbuf->buf     = buffer;
        ctlbuf->len     = PCIMPB->MessageActualUsedSize + sizeof(pci_mpb);
}
else {
        /* either there is no Message, or the PCIMPB and the Message are contiguous in memory */
        ctlbuf->buf         = PCIMPB;
        ctlbuf->len         = Message ? PCIMPB->MessageActualUsedSize + sizeof(pci_mpb) : sizeof(pci_mpb);
}

databuf->buf    = Data;
databuf->len    = Data ? PCIMPB->DataActualUsedSize : 0;

if (putmsg (ExID, &ctlbuf, &databuf, flags) != 0) {
        /* Error condition, errno will be set    */
        ....
}
else {
        /* Operation OK          */
        ....
}
```

6.6.6 PciGetMessage

Function body:

```
PCI_INTEGER PciGetMessage(      PCI_EXID            ExID,
                                struct pci_mpb      *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. PCI-MPB 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 might be optional. In this case they are specified as NULL. In order to be more efficient (see STREAMS getmsg hereafter), it is recommended that the PCI-MPB 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 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 PCI MPB 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 *)PCIMPB + sizeof(pci_mpb))) {
        /* there is a Message not NULL and,  PCIMPB and Message are not contiguous in memory,
        have to reserve a buffer where PCIMPB can be followed by the Message content */
        char  *buffer; /* pointer to a buffer, large enough to receive PCIMPB 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 ? PCIMPB->MessageMaximumSize + sizeof(pci_mpb):sizeof(pci_mpb);
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_mpb)) {
                /* Message, possibly of size 0 is present */
                PCIMPB->MessageActualUsedSize = ctlbuf->len - sizeof(pci_mpb);
                if (Message && ((char *)Message != (char *)PCIMPB + sizeof(pci_mpb)))
                        {
                        /* there is a Message not NULL and,  PCIMPB and Message are not contiguous in memory,
                        a buffer where PCIMPB is followed by the Message content, has been used */
                        memcpy (PCIMPB, buffer, sizeof(pci_mpb));
                        memcpy (Message,(buffer + sizeof(pci_mpb)), (ctlbuf->len - sizeof(pci_mpb)));
                }
                else {
                        /* the PCIMPB and the Message are contiguous in memory,  no additional buffer used */
                        Message = PCIMPB + sizeof(pci_mpb);
                }
        }
        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;
}
}
```

6.6.7 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 reissue the signal via the signal() system call – see below. This mechanism is 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 deregistered 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;
}
```

6.7 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.

6.7.1 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 availability of the NAF before the creation of the new dummy file.

6.7.2 Extraction action

During the deinstallation script of the STREAM driver, the directory **/etc/pcidd** is updated by removing the dummy file name of the NAF.

Appendix I

UNIX operating system implementation coding samples

These samples present a way to implement the exchange mechanism function call from the PUF point of view.

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)


/*****************************************************************
 *      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_timeout      = 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_timeout             = 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_timeout             = 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 PCIMPB and Message contents */
int nErr;
```

```
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 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 PCIMPB 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 PCIMPB 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;
        }
}
```