COMMON-ISDN-API

Version 2.0

Part I

4th Edition

June 2001

Author:
CAPI Association e.V.

Editor:
AVM GmbH, Germany
E-mail: hj.ortmann@avm.de

4th Edition / June 2001

Publisher:
CAPI Association e.V.
http://www.capi.org/

Contents (Part I)

SPECIAL NOTICES

Reader's Guide

This document defines **COMMON-ISDN-API Version 2.0**. Readers should be generally familiar with ISDN concepts.

Chapter 1 provides an introduction into the general concepts of the application interface **COMMON-ISDN-API** from a global point of view. Chapter 2 provides a detailed look at **COMMON-ISDN-API**'s position relative to the OSI models layers and introduces the different protocol options supported. Chapter 3 describes the basic mechanisms that ensure operating system independence, such as messages, message structures and the message protocol used. Chapter 4 describes the mechanisms which are necessary for messages to be exchanged between **COMMON-ISDN-API** and applications. Chapter 5 and 6 specify in detail the function and coding of each message and parameter. Chapter 7 illustrates the actions allowed in different states of a connection through state diagrams. Chapter 8 of **COMMON-ISDN-API** (moved to **COMMON-ISDN-API Part II**) includes all operating system-dependent **COMMON-ISDN-API** operations needed to exchange messages. It is divided into a subchapter for each operating system supported by **COMMON-ISDN-API**. Annex A contains arrow diagrams to facilitate an intuitive understanding of how to connect, exchange data and disconnect. Annex B is added to provide a coding scheme used to exchange G3 fax documents between **COMMON-ISDN-API** and applications. Annex C (included in **COMMON-ISDN-API Part III**) describes the use of supplementary services not covered in Part I. Annex D clarifies some implementation details.

The present edition of **COMMON-ISDN-API Version 2.0** consists of four parts: Part I (Chapters 1 to 8, Annexes A, B and D) defines the basics of **COMMON-ISDN-API Version 2.0** (messages, exchange mechanism, and parameters). Part II (Chapter 8) describes the operating system-dependent exchange mechanisms. Part III (Annex C) deals with supplementary service support not handled in Part I. Part IV deals with interoperability between CAPI implementations based on USB and is not intended for CAPI applications.

Disclaimer

While all due care has been taken in the preparation and publication of this document, errors in content, typographical or otherwise, may occur. If you have comments concerning its accuracy, please contact the CAPI Association.

The CAPI Association makes no representations or warranties with respect to the contents or use of this manual, and explicitly disclaims all express or implied warranties of merchantability or fitness for any particular purpose. Furthermore, the CAPI Association reserves the right to revise this publication and to make amendments to its content, at any time, without obligation to notify any person or entity of such revisions and changes.

Trademarks

The following terms are trademarks of companies, even though this is not explicitly indicated in the text.

MS-DOS is a registered trademark of Microsoft Corporation.
NetWare is a registered trademark of Novell, Inc.
Novell is a registered trademark of Novell, Inc.
OS/2 is a trademark of International Business Machines Corporation.
UNIX is a registered trademark of UNIX Systems Laboratories Inc.
Windows is a trademark of Microsoft Corporation.

PREFACE

COMMON-ISDN-API (**CAPI**) is an application programming interface standard used to access ISDN equipment connected to basic rate interfaces (**BRI**) and primary rate interfaces (**PRI**). By adhering to this standard, application developers can take advantage of a well-defined mechanism for communications over ISDN lines without having to adjust to the idiosyncrasies of specific hardware vendors' implementations. ISDN equipment vendors benefit in turn from a wealth of applications ready to run with their equipment.

COMMON-ISDN-API is now a well-established standard. Potential cost savings were the driving force for **COMMON-ISDN-API** controller and application development. Commercial users are rapidly migrating to ISDN (**I**ntegrated **S**ervices **D**igital **N**etwork) as the principal medium for exchanging data in a wide range of formats.

In 1989, manufacturers began defining an application interface that would be accepted in the growing ISDN market. For practical reasons, the focus of this standard was on the national ISDN protocol, since an ETSI ISDN protocol standard was not available at that time. Work on the application interface was completed in 1990 by a CAPI working group consisting of application providers, ISDN equipment manufacturers, bulk customers / user groups and DBP Telekom. **COMMON-ISDN-API** Version 1.1 was a great step in the development of the national ISDN market in Germany. Today almost every German ISDN solution, as well as a growing number of international products, is based on **COMMON-ISDN-API**.

Since then, the international protocol specification has been completed, and almost every telecommunications provider now offers BRI / PRI lines with protocols based on Q.931 / ETS 300 102. Experience in ISDN application interface design, knowledge of the market requirements and a large base of installed **COMMON-ISDN-API 1.1** solutions (hardware controllers and applications on a variety of operating systems) made it clear that a new application interface was needed for use in international ISDNs.

COMMON-ISDN-API Version 2.0 reflects more than ten years of ISDN business implementation experience in an exploding market. It incorporates all the benefits of CAPI Version 1.1 as well as all actual aspects of ISDN (such as Group 3 fax connectivity or video-telephony). It is based on Q.931 / ETS 300 102, but not limited to these protocols. It simplifies the development of ISDN applications through many default values which do not need to be programmed. It keeps applications free of ISDN protocol knowledge, thus making a great variety of applications possible.

By using **COMMON-ISDN-API Version 2.0** the international market can take advantage of available know-how and thus accelerate growth.

1 INTRODUCTION

COMMON-ISDN-API enables applications to access ISDN adapters in a straightforward manner and allows unrestricted use of their functions through a standardized software interface.

Future expansions or hardware changes will not affect applications which use this interface. **COMMON-ISDN-API** makes such changes transparent to user applications. Future extensions can preserve compatibility with the existing software base.

COMMON-ISDN-API provides an abstraction of ISDN services that is independent of the underlying network and of the adapter used to connect to the network. It provides an easy-to-use interface for applications and offers a unified access to different ISDN services such as data, voice, fax, video, telephony, etc.

COMMON-ISDN-API provides a basis for modular applications development in ISDN systems.

1.1 Scope

This document describes **COMMON-ISDN-API**, the application programming interface for ISDN. **COMMON-ISDN-API**'s design is message-oriented and event driven. **COMMON-ISDN-API** is described in three parts: the main part defines each message used and its message parameters. This part is entirely operating system independent. Part II deals with the operations needed to exchange these messages. Part III describes extensions of **COMMON-ISDN-API** to support ISDN supplementary services.

The specification of **COMMON-ISDN-API** as such is an application *interface*. The given *implementation* of **COMMON-ISDN-API,** however, which is actually seen by an application dealing with ISDN communications, represents a kind of *instantiation*. The state diagrams in Chapter 7 explain the behavior of **COMMON-ISDN-API** from a point of view set at the interface level, but also take the implementation of **COMMON-ISDN-API** as an instantiation (for real states) into consideration.

1.2 Features

COMMON-ISDN-API includes a number of important features.

- Support for basic call features, such as call set-up and clear-down
- Support for multiple B channels, for data and/or voice connections
- Support for several logical data link connections within a physical connection
- Selection of specific services and protocols during connection set-up and on answering incoming calls
- Transparent interface for protocols above Layer 3
- Support for one or more Basic Rate Interfaces as well as Primary Rate Interfaces on one or more ISDN adapters
- Support for multiple applications
- Operating system-independent messages
- Operating system-dependent exchange mechanism for optimum operating system integration
- Asynchronous, event-driven mechanism for high throughput
- Well-defined mechanism for manufacturer-specific extensions

2 OVERVIEW

COMMON-ISDN-API provides a standardized interface which allows any number of application programs to use any number of ISDN drivers and ISDN controllers. Applications can be freely assigned to specific drivers and controllers:

- One application can use one controller
- One application can use more than one controller
- Several applications can share a single controller
- Several applications can share more than one controller

Applications can use different protocols at different protocol levels; **COMMON-ISDN-API** provides a selection mechanism to support this. **COMMON-ISDN-API** also performs an abstraction from different protocol variants, creating a standardized network access. All connection-related data such as connection state, display messages etc. is available to applications at any time.

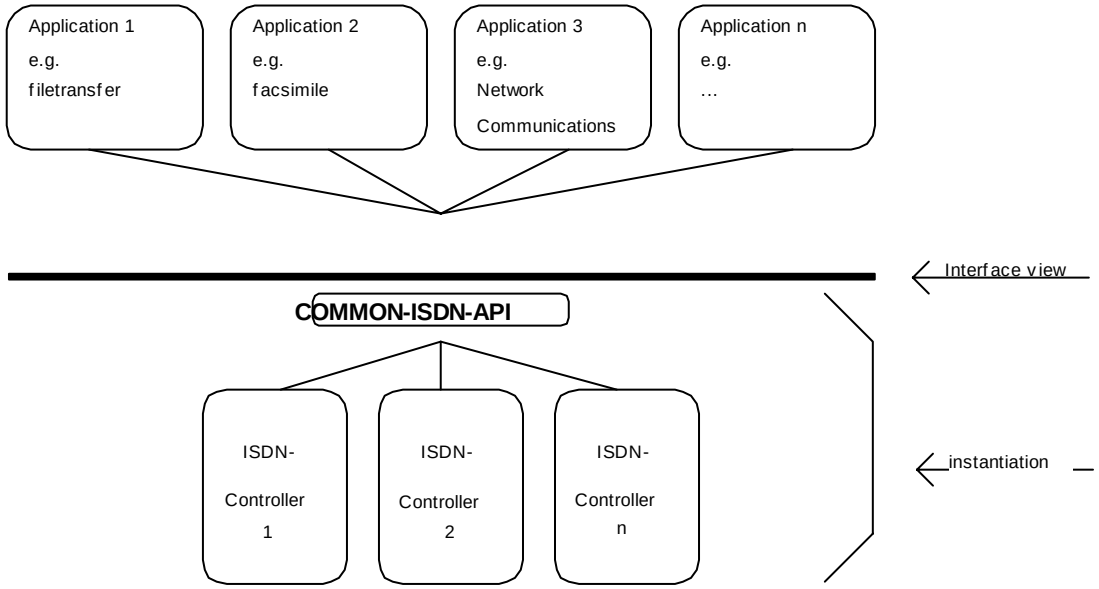


Figure 1: Position of **COMMON-ISDN-API**

COMMON-ISDN-API covers the entire signaling protocol as well as protocol layers 1 to 3 (physical and framing layer, data link layer and network layer) of the data channels. The interface of **COMMON-ISDN-API** is located between Layer 3 and Layer 4, and provides a point of reference for applications and higher level protocols.

COMMON-ISDN-API offers many commonly used protocols to applications without low-level protocol knowledge. The default protocol is **ISO 7776** (X.75 SLP), i.e. framing protocol **HDLC**, data link protocol **ISO 7776** (**X.75** SLP), and a transparent network layer.

Other supported framing layer variants are **HDLC inverted**, **PCM** (bit-transparent with byte framing) **64**/**56** kbit, and **V.110** sync / async. **COMMON-ISDN-API** integrates the following data link and network layer protocols: **LAPD** in accordance with Q.921 for **X.25 D-channel** implementation, **PPP** (Point-to-Point Protocol), **ISO 8208** (**X.25** DTE-DTE), **X.25 DCE**, **T.90NL** (with **T.70NL** compatibility) and **T.30** (Group 3 fax).

Even if not all protocols can be fit completely into the OSI scheme, **COMMON-ISDN-API** always supports three layers. Applications can configure each layer. In case of illegal or meaningless protocol stack combinations (e.g. bit-transparent 56 kbit/s and X.25 DCE), **COMMON-ISDN-API** reports an error.

The following chapter presents the basic mechanism used by **COMMON-ISDN-API**, based on message queues for the exchange of commands and data. The operations on these message queues and the structure of the messages exchanged are described. Afterward, other functions for identification and the mechanism for manufacturer-specific extensions are described.

3 MESSAGE OVERVIEW

The term *message* is a fundamental one in the definition of **COMMON-ISDN-API**. *Messages* are information defined by **COMMON-ISDN-API,** exchanged between the application and **COMMON-ISDN-API** by an asynchronous mechanism. This technique achieves operating system independence.

3.1 General Message Protocol

Communication between the application and **COMMON-ISDN-API** always uses the following general protocol:

A message is always followed by an appropriate reply. Messages going from an application to **COMMON-ISDN-API** are called **Requests**; the corresponding answer from **COMMON-ISDN-API** is called a **Confirmation**. Messages initiated by **COMMON-ISDN-API** are called **Indications**; the corresponding answer from an application is called a **Response**. This is also reflected in the naming convention for messages: every message name ends with the appropriate suffix (_REQ, _CONF, _IND and _RESP).

Each message contains a message number. **COMMON-ISDN-API** always returns the number used in the REQUEST message in the corresponding CONFIRMATION. Applications may choose unique message numbers to identify message correlation before interpreting incoming messages. INDICATIONS from **COMMON-ISDN-API** are numbered so that an application is guaranteed to get a different message number in every incoming INDICATION.

An application is not allowed to send RESPONSE messages without having received an INDICATION. **COMMON-ISDN-API** ignores such illegal messages.

3.2 Type Definitions

Parameters are associated with every message exchanged. In the description of messages and their parameters, only few basic data types are used:

- byte — 8bit value, coded as one octet
- word — 16bit value, coded as two contiguous octets, least significant first
- dword — 32bit value, coded as two contiguous words, least significant first
- qword — 64bit value, coded as two contiguous dwords, least significant first
- struct — coded as an array of octets, the first octet containing the length of the subsequent array. If the first octet has the value **255** (0xFF), this is an escape code: the following word is then interpreted as containing the length of the following data. An empty struct is coded as a single octet with value 0.

Every message is described in terms of these basic types.

3.3 Message Structure

All messages exchanged between applications and **COMMON-ISDN-API** consist of a fixed-length header and a parameter area of variable length, with one parameter immediately following another. No padding occurs in the message header or parameter area.

Message header	Parameter 1	Parameter 2	...	Parameter n

Figure 2: Message Layout

In order to facilitate future extensions, messages containing more parameters than defined shall be treated as valid messages. **COMMON-ISDN-API** implementations and applications shall ignore all such additional parameters.

The message header has the following layout:

Total length	ApplID	Command	Sub-command	Message number

Figure 3: Message Header Layout

The message header is composed of the following functional elements:

Header element	**Type**	**Contents**
Total length	word	Total length of the message including the complete message header.
ApplID	word	Unique identification number of the application. The application ID is assigned to the application by **COMMON-ISDN-API** in the CAPI_REGISTER operation
Command	byte	Command
Subcommand	byte	Command extension
Message number	word	Message number as described in 3.1 above

3.4 Manufacturer-Specific Extensions

Manufacturer-specific extensions of **COMMON-ISDN-API** are possible without altering the basic structure. An appropriate value in the command/subcommand field in the message header identifies such extensions.

3.5 Table of Messages

Messages are logically grouped into three kinds:

- Messages concerning the ISDN signaling protocol (D-channel)
- Messages concerning logical connections (B or D-channel)
- Administrative and other messages

The following table gives an overview of the defined messages and their functions. The complete description of each message is given in Chapter 5.

Message	Value	Description
CONNECT_REQ	0x02 / 0x80	initiates an outgoing physical connection
CONNECT_CONF	0x02 / 0x81	local confirmation of the request
CONNECT_IND	0x02 / 0x82	indicates an incoming physical connection
CONNECT_RESP	0x02 / 0x83	response to the indication
CONNECT_ACTIVE_IND	0x03 / 0x82	indicates the activation of a physical connection
CONNECT_ACTIVE_RESP	0x03 / 0x83	response to the indication
DISCONNECT_REQ	0x04 / 0x80	initiates clearing down of a physical connection
DISCONNECT_CONF	0x04 / 0x81	local confirmation of the request
DISCONNECT_IND	0x04 / 0x82	indicates the clearing of a physical connection
DISCONNECT_RESP	0x04 / 0x83	response to the indication
ALERT_REQ	0x01 / 0x80	initiates sending of ALERT, i.e. compatibility with call
ALERT_CONF	0x01 / 0x81	local confirmation of the request
INFO_REQ	0x08 / 0x80	initiates sending of signaling information
INFO_CONF	0x08 / 0x81	local confirmation of the request
INFO_IND	0x08 / 0x82	indicates specified signaling information
INFO_RESP	0x08 / 0x83	response to the indication

Table 1: Messages concerning the signaling protocol

Message	Value	Description
CONNECT_B3_REQ	0x82 / 0x80	initiates an outgoing logical connection
CONNECT_B3_CONF	0x82 / 0x81	local confirmation of the request
CONNECT_B3_IND	0x82 / 0x82	indicates an incoming logical connection
CONNECT_B3_RESP	0x82 / 0x83	response to the indication
CONNECT_B3_ACTIVE_IND	0x83 / 0x82	indicates the activation of a logical connection
CONNECT_B3_ACTIVE_RESP	0x83 / 0x83	response to the indication
CONNECT_B3_T90_ACTIVE_IND	0x88 / 0x82	indicates switching from T.70NL to T.90NL
CONNECT_B3_T90_ACTIVE_RESP	0x88 / 0x83	response to the indication
DISCONNECT_B3_REQ	0x84 / 0x80	initiates clearing down of a logical connection
DISCONNECT_B3_CONF	0x84 / 0x81	local confirmation of the request
DISCONNECT_B3_IND	0x84 / 0x82	indicates the clearing down of a logical connection
DISCONNECT_B3_RESP	0x84 / 0x83	response to the indication
DATA_B3_REQ	0x86 / 0x80	initiates sending of data over a logical connection
DATA_B3_CONF	0x86 / 0x81	local confirmation of the request
DATA_B3_IND	0x86 / 0x82	indicates incoming data over a logical connection
DATA_B3_RESP	0x86 / 0x83	response to the indication
RESET_B3_REQ	0x87 / 0x80	initiates the resetting of a logical connection
RESET_B3_CONF	0x87 / 0x81	local confirmation of the request
RESET_B3_IND	0x87 / 0x82	indicates the resetting of a logical connection
RESET_B3_RESP	0x87 / 0x83	response to the indication

Table 2: Messages concerning logical connections

Message	Value	Description
LISTEN_REQ	0x05 / 0x80	activates call and info indications
LISTEN_CONF	0x05 / 0x81	local confirmation of the request
FACILITY_REQ	0x80 / 0x80	requests additional facilities (e.g. ext. equipment)
FACILITY_CONF	0x80 / 0x81	local confirmation of the request
FACILITY_IND	0x80 / 0x82	indicates additional facilities (e.g. ext. equipment)
FACILITY_RESP	0x80 / 0x83	response to the indication
SELECT_B_PROTOCOL_REQ	0x41 / 0x80	selects protocol stack used for a logical connection
SELECT_B_PROTOCOL_CONF	0x41 / 0x81	local confirmation of the request
MANUFACTURER_REQ	0xFF / 0x80	manufacturer-specific operation
MANUFACTURER_CONF	0xFF / 0x81	manufacturer-specific operation
MANUFACTURER_IND	0xFF / 0x82	manufacturer-specific operation
MANUFACTURER_RESP	0xFF / 0x83	manufacturer-specific operation

Table 3: Administrative and other messages

4 EXCHANGE MECHANISM OVERVIEW

4.1 Message Queues

Communication between an application program and **COMMON-ISDN-API** takes place via message queues. As shown in Figure 4, there is exactly one message queue for **COMMON-ISDN-API** and one for each registered application program. Messages between application programs and **COMMON-ISDN-API** are exchanged via these message queues. In data transfer, the messages are used for signaling purposes only, and the data itself is transferred via a memory space shared by the application and **COMMON-ISDN-API**. The queues are organized first in, first out, so that **COMMON-ISDN-API** processes messages in the order of their arrival.

An application issues commands to an ISDN driver or controller by placing an appropriate message in the **COMMON-ISDN-API** message queue. In the reverse direction, a message from an ISDN driver or controller is transferred to the message queue of the addressed application.

This method, also used in higher-layer protocols and modern operating systems, allows flexible access by several applications to different ISDN drivers and controllers. It also provides a powerful mechanism for processing events that arrive asynchronously, which is a paramount requirement for high-speed data transfer.

The message queue structure is not specified. It is manufacturer-specific and transparent to the application program. **COMMON-ISDN-API** simply defines the necessary access operations.

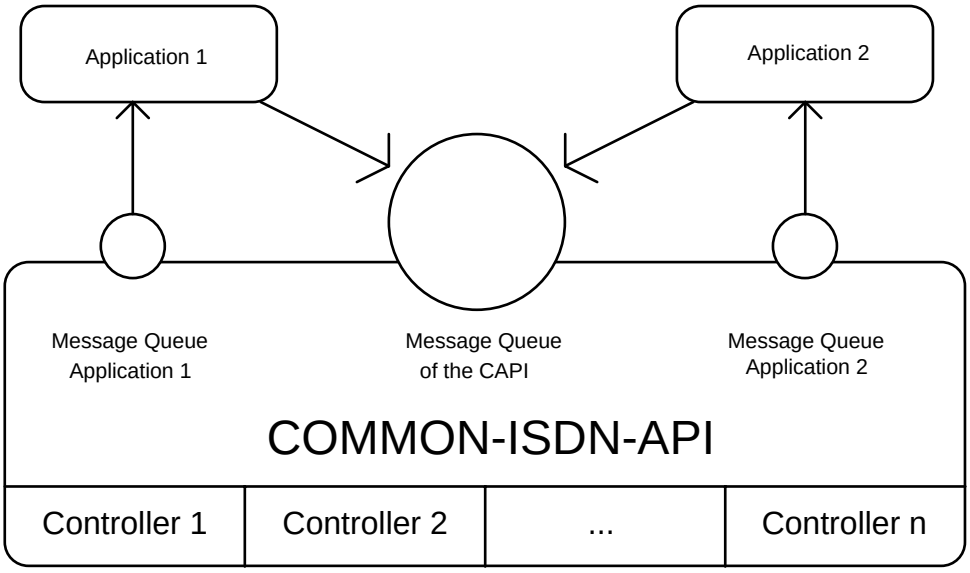


Figure 4: Message queues in **COMMON-ISDN-API**

4.2 Operations on Message Queues

4.2.1 Overview

The message queues described represent the link between an application and **COMMON-ISDN-API**, with its subordinate ISDN controllers and drivers. Only four operations are required to use the message queues. The operations on the message queues are not restricted to a particular system specification. Their respective characteristics and implementation are specific to each given operating system. This chapter describes the operating system-independent functions of **COMMON-ISDN-API**'s exchange operations. **COMMON-ISDN-API Part II**, Chapter 8 defines the operating system-specific implementation.

4.2.1.1 Registering an Application

Before an application can issue commands to **COMMON-ISDN-API,** it must be registered with **COMMON-ISDN-API**. The CAPI_REGISTER function is used to do this. **COMMON-ISDN-API** uses this function to assign a unique application number (ApplID) to the application. The message queue to the application is set up at the same time.

4.2.1.2 Messages from Application to COMMON-ISDN-API

All messages from an application to **COMMON-ISDN-API** are placed in **COMMON-ISDN-API**'s message queue. The operation CAPI_PUT_MESSAGE is provided for this purpose. The application transfers the message by calling this function. If **COMMON-ISDN-API**'s message queue cannot accept any more messages, the CAPI_PUT_MESSAGE function returns an error.

4.2.1.3 Messages from COMMON-ISDN-API to Application

COMMON-ISDN-API manages a message queue for each application and puts all messages to the application in this queue. Applications use the operation CAPI_GET_MESSAGE to read new messages from this queue. When this function is called, it returns the received message. If the application's message queue is empty, the operation CAPI_GET_MESSAGE returns an error. If an application does not retrieve these messages or the message queue size was configured too small, this queue may overflow. In this case, one or more messages from **COMMON-ISDN-API** are lost. The application is informed of this error in the next CAPI_GET_MESSAGE operation.

It is recommended that applications not 'poll' the queue for incoming messages. Instead, **COMMON-ISDN-API** provides mechanisms to inform an application that messages are present: CAPI_SET_SIGNAL or CAPI_WAIT_FOR_SIGNAL, depending on the resources offered by the operating system.

4.2.1.4 Releasing an Application

If a registered application wants to quit using **COMMON-ISDN-API**, its connection to **COMMON-ISDN-API** must be released. This is done by the CAPI_RELEASE operation. Releasing the application frees the previously used message queue. An application must disconnect all existing connections before issuing a CAPI_RELEASE: otherwise, the behavior of **COMMON-ISDN-API** is undefined. This applies only to non-external equipment. External devices controlled by **COMMON-ISDN-API** (such as telephones) may allow release from **COMMON-ISDN-API** without terminating existing calls.

4.2.1.5 Other Operations

Additional operations are available to obtain information about the manufacturer, software releases, configuration and serial numbers. Depending on the operating system, callback functions can be registered, which are activated when a new message is placed in the application's message queue.

4.2.2 Operations

4.2.2.1 CAPI_REGISTER

Applications use *CAPI_REGISTER* to register their presence with **COMMON-ISDN-API**. Registration parameters specify the maximum number of logical ISDN connections, the message buffer size, the number of data buffers and the data buffer size required by the application. The message buffer size is normally calculated in accordance with the following formula:

Message buffer size = 1024 + (1024 * number of logical ISDN connections)

Successful registration causes **COMMON-ISDN-API** to assign and return to the caller a unique application identifier. This application identifier is used in subsequent **COMMON-ISDN-API** function calls as well as in defined **COMMON-ISDN-API** messages.

Some operating systems require applications to pass a memory buffer or other additional information to **COMMON-ISDN-API**.

4.2.2.2 CAPI_RELEASE

Applications use CAPI_RELEASE to terminate their registration with **COMMON-ISDN-API**. All memory and other resources allocated to the application by **COMMON-ISDN-API** are released.

4.2.2.3 CAPI_PUT_MESSAGE

Applications call CAPI_PUT_MESSAGE to transfer a single message to **COMMON-ISDN-API**. When the function call returns, the message memory area can be reused by the application.

4.2.2.4 CAPI_GET_MESSAGE

Applications call CAPI_GET_MESSAGE to retrieve a single message from **COMMON-ISDN-API**. If a message is available, the message's address is returned to the application. If there are no messages, CAPI_GET_MESSAGE returns an error indication (see also CAPI_SET_SIGNAL and CAPI_WAIT_FOR_SIGNAL to avoid polling for messages).

The contents of the message block returned by CAPI_GET_MESSAGE is valid until the same application calls CAPI_GET_MESSAGE again. Applications which process the message asynchronously or need to preserve the message beyond the next call to CAPI_GET_MESSAGE must make a local copy of the message.

4.2.2.5 CAPI_SET_SIGNAL

Applications call CAPI_SET_SIGNAL to install a mechanism which signals that a **COMMON-ISDN-API** message is present, or that a **COMMON-ISDN-API** internal busy/queue full condition has been cleared. All restrictions pertinent to an interrupt context apply to the callback function.

4.2.2.6 CAPI_WAIT_FOR_SIGNAL

Applications call CAPI_WAIT_FOR_SIGNAL to wait for an asynchronous event from **COMMON-ISDN-API**. This function returns as soon as a message from **COMMON-ISDN-API** is available.

4.2.2.7 CAPI_GET_PROFILE

Applications call CAPI_GET_PROFILE to retrieve capability information from **COMMON-ISDN-API**. **COMMON-ISDN-API** copies information about the implemented features, the total number of controllers and the protocols supported by the specified controller to a 64-byte buffer passed by the calling application. The application must ignore unknown bits. **COMMON-ISDN-API** sets every reserved field to zero. CAPI_GET_PROFILE fills the buffer with the following structure:

Type	Description
2 bytes	Number of installed controllers, least significant byte first
2 bytes	Number of supported B-channels, least significant byte first
4 bytes	Global options (bit field): [0]: Internal controller supported [1]: External equipment supported [2]: Handset supported (external equipment must also be set) [3]: DTMF supported [4]: Supplementary Services (see Part III) [5]: Channel allocation supported (leased lines) [6]: Parameter *B channel operation* supported [7]: Line Interconnect supported [8]...[31]: reserved
4 bytes	B1 protocol support (bit field): [0]: 64 kbit/s with HDLC framing, always set. [1]: 64 kbit/s bit-transparent operation with byte framing from the network [2]: V.110 asynchronous operation with start/stop byte framing [3]: V.110 synchronous operation with HDLC framing [4]: T.30 modem for Group 3 fax [5]: 64 kbit/s inverted with HDLC framing. [6]: 56 kbit/s bit-transparent operation with byte framing from the network [7]: Modem with all negotiations [8]: Modem asynchronous operation with start/stop byte framing [9]: Modem synchronous operation with HDLC framing [10]...[31]: reserved
4 bytes	B2 protocol support (bit field): [0]: ISO 7776 (X.75 SLP), always set [1]: Transparent [2]: SDLC [3]: LAPD in accordance with Q.921 for D-channel X.25 (SAPI 16) [4]: T.30 for Group 3 fax [5]: Point-to-Point Protocol (PPP) [6]: Transparent (ignoring framing errors of B1 protocol) [7]: Modem error correction and compression (V.42 bis or MNP5) [8]: ISO 7776 (X.75 SLP) modified supporting V.42 bis compression [9]: V.120 asynchronous mode [10]: V.120 asynchronous mode supporting V.42 bis [11]: V.120 bit-transparent mode [12]: LAPD in accordance with Q.921 including free SAPI selection [13]...[31]: reserved

4 bytes	B3 protocol support (bit field): [0]: Transparent, always set [1]: T.90NL with compatibility to T.70NL in accordance to T.90 Appendix II. [2]: ISO 8208 (X.25 DTE-DTE) [3]: X.25 DCE [4]: T.30 for Group 3 fax [5]: T.30 for Group 3 fax with extensions [6]: reserved [7]: Modem [8]...[31]: reserved
24 bytes	reserved for **COMMON-ISDN-API** use
20 bytes	Manufacturer-specific information

CAPI_GET_PROFILE information structure

4.2.2.8 CAPI_GET_MANUFACTURER

Applications call CAPI_GET_MANUFACTURER to retrieve information about the manufacturer of the specified ISDN controller. **COMMON-ISDN-API** copies this information to a 64-byte buffer passed by the calling application.

4.2.2.9 CAPI_GET_VERSION

Applications call CAPI_GET_VERSION to retrieve version information from the specified ISDN controller. Major and minor version numbers are returned for both **COMMON-ISDN-API** and the manufacturer-specific implementation.

4.2.2.10 CAPI_GET_SERIAL_NUMBER

Applications call CAPI_GET_SERIAL_NUMBER to retrieve the serial number of the specified ISDN controller. **COMMON-ISDN-API** copies this information to a buffer of eight bytes passed by the calling application.

4.2.2.11 CAPI_INSTALLED

Applications call CAPI_INSTALLED to determine whether the ISDN controller and all necessary drivers are installed.

4.2.2.12 CAPI_MANUFACTURER

The usage of CAPI_MANUFACTURER is manufacturer-dependent.

4.3 Table of Operations

Operation	Description
CAPI_REGISTER	Register an application
CAPI_RELEASE	Release an application
CAPI_PUT_MESSAGE	Transfer message to CAPI
CAPI_GET_MESSAGE	Get message from CAPI
CAPI_SET_SIGNAL *	Register call-back function
CAPI_WAIT_FOR_SIGNAL *	Wait for new message to be made available
CAPI_GET_PROFILE	Get capabilities of CAPI implementation
CAPI_GET_MANUFACTURER	Get manufacturer identification
CAPI_GET_VERSION	Get CAPI version numbers
CAPI_GET_SERIAL_NUMBER	Get serial number
CAPI_INSTALLED *	Check whether CAPI is installed
CAPI_MANUFACTURER *	Manufacturer-specific function

Table 4: Operations defined in **COMMON-ISDN-API**

* Operations marked with an asterisk (*) are only available in implementations for certain operating systems.

5 MESSAGE DESCRIPTIONS

The following section defines all **COMMON-ISDN-API** messages with their respective parameters. Parameters are explained in more detail in Chapter 6.

Messages are listed alphabetically, but disregarding the extension (_REQ, _CONF, _IND, _RESP), which indicates the initiator of the exchange and the direction of the message. For each basic message name, the following order is observed: REQUEST, CONFIRMATION, INDICATION, RESPONSE.

5.1 ALERT_REQ

Description

This message should be used by applications to indicate compatibility with an incoming call. This command sends an ALERT to the network to prevent the call from expiring (“No user responding”). If an application is able to accept the call immediately, it need not use this message, but can issue a CONNECT_RESP to **COMMON-ISDN-API** directly.

ALERT_REQ	Command	**0x01**
	Subcommand	**0x80**

Parameter	**Type**	**Comment**
PLCI	dword	Physical Link Connection Identifier
Additional info	struct	Additional information elements

Note

The parameter *Additional info* is coded as an empty structure if no additional information (such as user-user data) needs to be transmitted.

If the parameter *Sending Complete* (part of parameter *Additional info*) is set to 1, a CALL PROCEEDING is sent to the network instead of an ALERT to indicate, that all needed information is available.

5.2 ALERT_CONF

Description

This message confirms the reception of an ALERT_REQ.

ALERT_CONF	Command	**0x01**
	Subcommand	**0x81**

Parameter	Type	Comment
PLCI	dword	Physical Link Connection Identifier
Info	word	**0:** Alert initiated **0x0003:** Alert already sent by another application **0x2001:** Message not supported in current state **0x2002:** Illegal PLCI **0x2007:** Illegal message parameter coding

Note

Info 0x0003 is returned if another application has already initiated the sending of an ALERT message to the network. In this case, the *Additional info* parameter of the corresponding ALERT_REQ has been ignored.

See also

Description of *broadcast mechanism* in LISTEN_REQ

5.3 CONNECT_REQ

Description

This message initiates the setting up of a physical connection. An application only needs to provide the relevant parameters (i.e. *Controller*, *CIP value* and usually *called party number*). Every other structure can be empty (length zero). In this case, the default values as described in Chapter 6 are used.

CONNECT_REQ	Command	**0x02**
	Subcommand	**0x80**

Parameter	**Type**	**Comment**
Controller	dword	
CIP Value	word	Compatibility Information Profile
Called party number	struct	Called party number
Calling party number	struct	Calling party number
Called party subaddress	struct	Called party subaddress
Calling party subaddress	struct	Calling party subaddress
B protocol	struct	B protocol to be used
BC	struct	Bearer Capability
LLC	struct	Low Layer Compatibility
HLC	struct	High Layer Compatibility
Additional Info	struct	Additional information elements

Note

If an application provides *BC*, *LLC* and/or *HLC*, the parameters are used without checking the resulting combination.

The absence (i.e. coding as an empty structure) of the *B protocol* parameter results in the default protocol behavior: ISO 7776 (X.75) and window size seven. This is a recommended selection to obtain general connectivity with the benefits of HDLC error recovery. Note that ISO 7776 defines a default maximum data length of 128 octets, whereas **COMMON-ISDN-API** is able to handle up to at least 2048 octets, depending on the **CAPI_REGISTER** parameters values of the given application.

5.4 CONNECT_CONF

Description

This message confirms the initiation of a call set-up. This connection is assigned a *PLCI*, which serves as an identifier in further processing. Errors are returned in the parameter *Info*.

CONNECT_CONF	Command	**0x02**
	Subcommand	**0x81**

Parameter	Type	Comment
PLCI	dword	Physical Link Connection Identifier
Info	word	**0:** Connect initiated **0x2002:** Illegal controller **0x2003:** No PLCI available **0x2007:** Illegal message parameter coding **0x3001:** B1 protocol not supported **0x3002:** B2 protocol not supported **0x3003:** B3 protocol not supported **0x3004:** B1 protocol parameter not supported **0x3005:** B2 protocol parameter not supported **0x3006:** B3 protocol parameter not supported **0x3007:** B protocol combination not supported **0x3009:** CIP Value unknown

Note

Upon confirmation, the connection is in the set-up phase. Successful switching is indicated by the subsequent message **CONNECT_ACTIVE_IND**.

If an application needs to identify which CONNECT_REQ corresponds to this message, it can use the message number mechanism described in Chapter 3.

5.5 CONNECT_IND

Description

This message indicates an incoming call for a physical connection. The incoming call is assigned a *PLCI* which is used to identify this connection in subsequent messages.

CONNECT_IND	Command	**0x02**
	Subcommand	**0x82**

Parameter	Type	Comment
PLCI	dword	Physical Link Connection Identifier
CIP Value	word	Compatibility Information Profile
Called party number	struct	Called party number
Calling party number	struct	Calling party number
Called party subaddress	struct	Called party subaddress
Calling party subaddress	struct	Calling party subaddress
BC	struct	Bearer capability
LLC	struct	Low Layer Compatibility
HLC	struct	High Layer Compatibility
Additional Info	struct	Additional information elements
Calling party number	struct	Second calling party number (see ETS 300-092 Annex B)

Note

Incoming calls are only signaled if the application has sent the message LISTEN_REQ to **COMMON-ISDN-API**.

All information available from the network at this point is signaled to the application. Empty structs indicate the absence of this information.

Incoming calls are not signaled for security reason if the combination of Calling party number, Calling party subaddress and CIP Value is not allowed by Call Control Supervision (see Annex D.2).

5.6 CONNECT_RESP

Description

This message is used by the application to react to an incoming call. The incoming call is identified by the parameter *PLCI*. The parameter *Reject* is used to accept, reject or ignore the call.

CONNECT_RESP	Command	**0x02**
	Subcommand	**0x83**

Parameter	Type	Comment
PLCI	dword	Physical Link Connection Identifier
Reject	word	**0:** Accept call **1:** Ignore call **2:** Reject call, normal call clearing **3:** Reject call, user busy **4:** Reject call, requested circuit/channel not available **5:** Reject call, facility rejected **6:** Reject call, channel unacceptable **7:** Reject call, incompatible destination **8:** Reject call, destination out of order **0x34xx:** The content of the low byte 'xx' will be signaled to the network in a cause information element (octet 4). It is the application's responsibility to provide a value that is properly coded in accordance with Q.931/ETS 300 102-1. The controller will send this cause value indicating coding standard CCITT (octet 3).
B protocol	struct	B protocol to be used
Connected number	struct	Connected number
Connected subaddress	struct	Connected subaddress
LLC	struct	Low Layer Compatibility
Additional Info	struct	Additional information elements

Note

The parameter *LLC* can optionally be used for LLC negotiation if supported by the network.

Any unknown *Reject* value is mapped to *normal call clearing*.

Any *Reject* value other than *accept call* causes a DISCONNECT_IND to be sent to the application.

The absence (i.e. coding as an empty structure) of the parameter *B protocol* results in the default protocol behavior: ISO 7776 (X.75) and window size seven. This is a rec-

ommended selection to obtain general connectivity with the benefits of HDLC error recovery. Note that ISO 7776 describes a default maximum data length of 128 octets, whereas **COMMON-ISDN-API** is able to handle up to at least 2048 octets, depending on the **CAPI_REGISTER** values of an application.

5.7 CONNECT_ACTIVE_IND

Description

This message indicates the physical connection of a B channel. The connection is identified by the parameter *PLCI*.

CONNECT_ACTIVE_IND	Command	**0x03**
	Subcommand	**0x82**

Parameter	Type	Comment
PLCI	dword	Physical Link Connection Identifier
Connected number	struct	Connected number
Connected subaddress	struct	Connected subaddress
LLC	struct	Low Layer Compatibility

Note

The parameters *Connected number/subaddress* and *LLC* are filled in completely if the network provides this information. The absence of network information is indicated by empty structures.

5.8 CONNECT_ACTIVE_RESP

Description

With this message the application acknowledges the receipt of a CONNECT_ACTIVE_IND.

CONNECT_ACTIVE_RESP	Command	**0x03**
	Subcommand	**0x83**

Parameter	Type	Comment
PLCI	dword	Physical Link Connection Identifier

5.9 CONNECT_B3_ACTIVE_IND

Description

This message indicates the logical connection of a B channel. The connection is identified by the parameter *NCCI*. The parameter *NCPI* is used to transfer additional protocol-dependent information.

CONNECT_B3_ACTIVE_IND	Command	**0x83**
	Subcommand	**0x82**

Parameter	Type	Comment
NCCI	dword	Network Control Connection Identifier
NCPI	struct	Network Control Protocol Information

Note

The meaning of the parameter *NCPI* depends on the protocol used.

After this message, incoming data can be indicated to the application.

In the case outgoing calls using the protocol T.30, this message does not imply successful training between both sides. This is to enable an application to send data to **COMMON-ISDN-API** without waiting for termination of the training phase. If the training phase is not successful, **COMMON-ISDN-API** indicates this in the message DISCONNECT_B3_IND.

5.10 CONNECT_B3_ACTIVE_RESP

Description

With this message the application acknowledges the receipt of a CONNECT_B3_ACTIVE_IND.

CONNECT_B3_ACTIVE_RESP	Command	**0x83**
	Subcommand	**0x83**

Parameter	Type	Comment
NCCI	dword	Network Control Connection Identifier

5.11 CONNECT_B3_REQ

Description

This message initiates the setting up of a logical connection. The physical connection is identified by the parameter *PLCI*. Protocol-dependent information can be transferred using the parameter *NCPI*.

CONNECT_B3_REQ	Command	**0x82**
	Subcommand	**0x80**

Parameter	Type	Comment
PLCI	dword	Physical Link Connection Identifier
NCPI	struct	Network Control Protocol Information

Note

The meaning of the parameter *NCPI* depends on the protocol used.

5.12 CONNECT_B3_CONF

Description

This message confirms the initiation of a logical connection set-up. The logical connection is assigned an *NCCI* for subsequent identification. Error information is supplied in the parameter *Info*.

CONNECT_B3_CONF	Command	**0x82**
	Subcommand	**0x81**

Parameter	Type	Comment
NCCI	dword	Network Control Connection Identifier
Info	Word	**0:** Connect initiated **0x0001:** NCPI not supported by current protocol, NCPI ignored **0x2001:** Message not supported in current state **0x2002:** Illegal PLCI **0x2004:** No NCCI available **0x3008:** NCPI not supported

Note

This confirmation means that the connection is now in the set-up phase. Successful set-up is indicated by the subsequent message **CONNECT_B3_ACTIVE_IND**.

If the value **0x0001** is returned in the parameter *Info*, then the set-up of a logical connection has been initiated, but the parameter *NCPI* was ignored. In that case, the Layer 3 protocol used does not support the specified value of *NCPI* (e.g. the transparent mode of Layer 3).

5.13 CONNECT_B3_IND

Description

This message indicates an incoming logical connection in a physical connection (incoming call). The incoming connection is assigned an *NCCI* for subsequent identification. Protocol-dependent information, if available, is transferred by the parameter *NCPI*.

CONNECT_B3_IND	Command	**0x82**
	Subcommand	**0x82**

Parameter	Type	Comment
NCCI	dword	Network Control Connection Identifier
NCPI	struct	Network Control Protocol Information

Note

The meaning of the parameter *NCPI* depends on the protocol.

This message means that the connection is in the set-up phase. Successful set-up is indicated by the subsequent CONNECT_B3_ACTIVE_IND message.

In case of B3 protocol 5 (T.30 for Group 3 fax with extensions), this message is sent to the application after receipt of the T.30 DTC or DCS frame. If these frames are not received within the defined time-out, a CONNECT_B3_IND directly followed by a DISCONNECT_B3_IND shall be sent to the application.

For modem operation with B3 protocols 0 or 7, this message shall be sent to the application when modem training and negotiation starts.

Incoming logical connections with B2 Protocol = 3 (“LAPD in accordance with Q.921 for D channel X.25”) are not signaled for security reason if the combination of X.25 Calling DTE address (see X.25 Incoming Call in *NCPI*) and TEI in the corresponding CONNECT_REQ is not allowed by Call Control Supervision (see Annex D.2).

5.14 CONNECT_B3_RESP

Description

With this message the application accepts or rejects an incoming logical connection. The incoming call is identified by the parameter *NCCI*. The call is accepted or rejected using the parameter *reject*. The parameter *NCPI* can be used to transfer additional protocol-dependent information.

CONNECT_B3_RESP	Command	**0x82**
	Subcommand	**0x83**

Parameter	**Type**	**Comment**
NCCI	dword	Network Control Connection Identifier
Reject	word	**0:** Accept call **2:** Reject call, normal call clearing
NCPI	struct	Network Control Protocol Information

Note

The meaning of the parameter *NCPI* depends on the protocol used.

Any other value of parameter *Reject* results in the call being rejected.

5.15 CONNECT_B3_T90_ACTIVE_IND

Description

This message indicates the change from T.70 to T.90 within a logical connection on a B channel. The connection is identified by the parameter *NCCI*. The parameter *NCPI* is used to transfer additional T.90 information.

CONNECT_B3_T90_ACTIVE_IND	Command	**0x88**
	Subcommand	**0x82**

Parameter	Type	Comment
NCCI	dword	Network Control Connection Identifier
NCPI	struct	Network Control Protocol Information

Note

This message is only generated if the selected B3 protocol is T.90NL with T.70NL compatibility in accordance with T.90 Appendix II. In this case, the protocol initially used is T.70. This message indicates the negotiation and change to T.90.

5.16 CONNECT_B3_T90_ACTIVE_RESP

Description

With this message, the application acknowledges the receipt of a CONNECT_B3_-T90_ACTIVE_IND.

CONNECT_B3_T90_ACTIVE_RESP	Command	**0x88**
	Subcommand	**0x83**

Parameter	Type	Comment
NCCI	dword	Network Control Connection Identifier

5.17 DATA_B3_REQ

Description

This message sends data within the logical connection identified by the *NCCI*. Data to be sent are referenced by the parameters *Data*/*Data length*. The data is not contained in the message: a 32-bit pointer is used to transfer the address of the data area. The application issues a unique identifier for this data block in the parameter *Data handle*. This handle is used in a later DATA_B3_CONF. It is possible to set additional information, such as an indication that more data follows, delivery confirmation etc. in the parameter *Flags*. The flags are not supported by all protocols.

DATA_B3_REQ	Command	**0x86**
	Subcommand	**0x80**

Parameter	Type	Comment
NCCI	dword	Network Control Connection Identifier
Data	dword	Pointer to the data to be sent
Data length	word	Size of data area to be sent
Data handle	word	Referenced in **DATA_B3_CONF**
Flags	word	**[0]:** Qualifier bit **[1]:** More-data bit **[2]:** Delivery confirmation bit **[3]:** Expedited data **[4]:** UI frame **[5] to [15]:** reserved
Data64	qword	For 64bit applications: 64-bit pointer to the data to be sent. All other applications: reserved (see note)

Note

The data transfer does not support assembly or re-assembly of data.

An application must not change or free the data area before the corresponding DATA_B3_CONF is received.

Flags are protocol-dependent. If an application sets reserved bits in the *Flags* parameter, **COMMON-ISDN-API** rejects the **DATA_B3_REQ**. This is to allow future expansion of this parameter. If an application sets bits in the *Flags* parameter which are not supported by the current protocol, **COMMON-ISDN-API** accepts the **DATA_B3_REQ,** but returns error information in the corresponding DATA_B3_CONF.

B2 protocols 9 and 11 (V.120 asynchronous/bit transparent mode): The application must limit *Data length* to 259 bytes to conform to V.120.

If delivery confirmation is requested for transparent B2/B3 protocol, the DATA_B3_CONF is generated when the send data has been completely transmitted.

64bit applications have to use the *Data64* parameter. In this case the *Data* parameter has to be coded as 0.

5.18 DATA_B3_CONF

Description

This message confirms the acceptance of a data package to be sent. The logical connection is identified by the parameter *NCCI*. The parameter *Data handle* contains the identifier given by the application in the associated DATA_B3_REQ. After receiving this message, the application can reuse the referenced data area.

DATA_B3_CONF	Command	**0x86**
	Subcommand	**0x81**

Parameter	Type	Comment
NCCI	dword	Network Control Connection Identifier
Data handle	word	Identifies the corresponding **DATA_B3_REQ**
Info	word	**0:** Data transmission initiated **0x0002:** Flags not supported by current protocol, flags ignored **0x2001:** Message not supported in current state **0x2002:** Illegal NCCI **0x2007:** Illegal message parameter coding **0x300A:** Flags not supported (reserved bits) **0x300C:** Data length not supported by current protocol

Note

Every DATA_B3_REQ results in a corresponding **DATA_B3_CONF**, with one exception: after transmitting the message DISCONNECT_B3_IND to an application, **COMMON-ISDN-API** is not allowed to send any other message concerning this logical connection. Therefore the application must ensure correct management of resources or buffers.

If an application sets the delivery confirmation bit in the corresponding DATA_B3_REQ, the application will receive the confirmation after delivery of the sent packet is confirmed by the protocol used (if the selected protocol supports this mechanism).

COMMON-ISDN-API supports up to seven unconfirmed DATA_B3_REQ messages.

5.19 DATA_B3_IND

Description

This message indicates incoming data within a logical connection. The logical connection is identified by the *NCCI*. The length of the incoming data area is indicated by the parameter *Data length*. The incoming data area itself can be referenced by the parameter *Data*. The data is not contained in the message: a 32-bit pointer is used to communicate the address of the data area. **COMMON-ISDN-API** also issues a handle corresponding to this data area in the parameter *Data handle*. When the application confirms receipt of the data by sending a DATA_B3_RESP message, it must also supply this handle. Additional protocol-specific information available—such as an indication that more data follows, delivery confirmation etc.—is supplied in the parameter *Flags*.

DATA_B3_IND	Command	**0x86**
	Subcommand	**0x82**

Parameter	**Type**	**Comment**
NCCI	dword	Network Control Connection Identifier
Data	dword	Pointer to data received (see note)
Data length	word	Size of data area received
Data handle	word	Handle of data area, referenced in **DATA_B3_RESP**
Flags	word	[0]: Qualifier bit [1]: More-data bit [2]: Delivery confirmation bit [3]: Expedited data [4]: Break (B2 protocols 9,10 and 11) [5 to 14]: reserved [15]: Framing error bit: data may be invalid (B2 protocols 6, 9 and 11)
Data64	qword	For 64bit applications: 64-bit pointer to the data received. All other applications: reserved (see note)

Note

The data transfer does not support re-assembly functions.

The data area that contains the data remains allocated until the corresponding DATA_B3_RESP is received. However, expedited data is only valid until the next **CAPI_GET_MESSAGE** is called by the application.

On receiving **DATA_B3_IND** messages with reserved bits set in the parameter *Flags*, the application must ignore the data area but process the message, i.e. send a DATA_B3_RESP to **COMMON-ISDN-API**. This is to allow future expansion of the parameter *Flags*.

In case of 64bit applications the *Data64* parameter is used. In this case the *Data* parameter is coded as 0.

5.20 DATA_B3_RESP

Description

With this message, the application acknowledges receipt of an incoming data package. The logical connection is identified by the parameter *NCCI*. The parameter *Data handle* identifies the corresponding DATA_B3_IND.

DATA_B3_RESP	Command	**0x86**
	Subcommand	**0x83**

Parameter	**Type**	**Comment**
NCCI	dword	Network Control Connection Identifier
Data handle	word	Identifies the corresponding **DATA_B3_IND**

Note

This message frees the data buffer referenced by *Data handle* for reuse by **COMMON-ISDN-API**.

High data throughput requires that the application respond promptly to DATA_B3_IND messages. Failure to do so triggers flow control on the line, or may cause loss of incoming data in case of protocols without flow control mechanisms.

5.21 DISCONNECT_B3_REQ

Description

This message initiates the clearing down of the logical connection identified by the parameter *NCCI*. The parameter *NCPI* can be used to transfer additional protocol dependent information.

DISCONNECT_B3_REQ	Command	**0x84**
	Subcommand	**0x80**

Parameter	Type	Comment
NCCI	dword	Network Control Connection Identifier
NCPI	struct	Network Control Protocol Information

Note

The meaning of the parameter *NCPI* depends on the protocol used.

In the case of Group 3 fax (B protocol T.30) and voice (B1 protocol bit-transparent, B2/B3 protocol transparent), outgoing data received from the application by means of DATA_B3_REQ messages is sent before the logical connection is disconnected.

5.22 DISCONNECT_B3_CONF

Description

This message confirms that a logical connection clear-down has been initiated. Any errors are coded in the parameter *Info*.

DISCONNECT_B3_CONF	Command	**0x84**
	Subcommand	**0x81**

Parameter	Type	Comment
NCCI	dword	Network Control Connection Identifier
Info	word	**0:** Disconnect initiated **0x0001:** NCPI not supported by current protocol: NCPI ignored **0x2001:** Message not supported in current state **0x2002:** Illegal NCCI **0x2007:** Illegal message parameter coding **0x3008:** NCPI not supported

5.23 DISCONNECT_B3_IND

Description

This message indicates the clearing down of the logical connection identified by the parameter *NCCI*. The parameter *Reason_B3* indicates whether this clear-down was caused by incorrect protocol behavior or by Call Control Supervision (see Annex D.2). The parameter *NCPI* is used to indicate additional protocol-dependent information, if available.

DISCONNECT_B3_IND	Command	**0x84**
	Subcommand	**0x82**

Parameter	Type	Comment
NCCI	dword	Network Control Connection Identifier
Reason_B3	word	**0:** Clearing in accordance with protocol **0x3301:** Protocol error, Layer 1 **0x3302:** Protocol error, Layer 2 **0x3303:** Protocol error, Layer 3 **0x3305:** Cleared by Call Control Supervision Protocol dependent values are described in Chapter 6.
NCPI	struct	Network Control Protocol Information

Note

The meaning of the *NCPI* parameter depends on the protocol used.

After this message, no further messages concerning this *NCCI* are sent to the application. The application must answer this message with a DISCONNECT_B3_RESP in order to free the resources allocated to the *NCCI*.

Outgoing logical connections with B2 Protocol = 3 (“LAPD in accordance with Q.921 for D channel X.25”) could be cleared for security reason (*Reason_B3* = 0x3305) if the combination of X.25 Called DTE address (see X.25 Call Request in NCPI) of the corresponding CONNECT_B3_REQ and TEI of the corresponding CONNECT_REQ is not allowed by Call Control Supervision (see Annex D.2).

5.24 DISCONNECT_B3_RESP

Description

With this message, the application acknowledges the clearing down of a logical connection.

DISCONNECT_B3_RESP	Command	**0x84**
	Subcommand	**0x83**

Parameter	Type	Comment
NCCI	dword	Network Control Connection Identifier

Note

With this message, resources allocated to the *NCCI* are released.

If an application fails to send this message after receiving DISCONNECT_B3_IND, **COMMON-ISDN-API** will eventually reject subsequent CONNECT_B3_REQ messages with the info value **No NCCI available** (0x2004).

5.25 DISCONNECT_REQ

Description

This message initiates the clearing of a physical connection, identified by the parameter *PLCI*.

DISCONNECT_REQ	Command	**0x04**
	Subcommand	**0x80**

Parameter	Type	Comment
PLCI	dword	Physical Link Connection Identifier
Additional Info	struct	Additional information elements

Note

COMMON-ISDN-API clears existing logical connections and issues a DISCONNECT_B3_IND message containing the cause **0x3301** (protocol error, Layer 1) before clearing the physical connection .

5.26 DISCONNECT_CONF

Description

This message confirms the initiation of clearing a physical connection. Any errors are coded in the parameter *Info*.

DISCONNECT_CONF	Command	**0x04**
	Subcommand	**0x81**

Parameter	Type	Comment
PLCI	dword	Physical Link Connection Identifier
Info	word	**0:** Disconnect initiated **0x2001:** Message not supported in current state **0x2002:** Illegal PLCI **0x2007:** Illegal message parameter coding

5.27 DISCONNECT_IND

Description

This message indicates the clearing of the physical channel identified via the parameter *PLCI*. The parameter *Reason* indicates the cause for this clearing.

DISCONNECT_IND	Command	**0x04**
	Subcommand	**0x82**

Parameter	Type	Comment
PLCI	dword	Physical Link Connection Identifier
Reason	word	**0:** No cause available **0x3301:** Protocol error, Layer 1 **0x3302:** Protocol error, Layer 2 **0x3303:** Protocol error, Layer 3 **0x3304:** Another application got that call **0x3305:** Cleared by Call Control Supervision **0x34xx**: Disconnect cause from the network in accordance with Q.850/ETS 300 102-1. The cause value received from the network in a cause information element (Octet 4) is indicated in the field 'xx'.

Note

After this message, no further messages concerning this *PLCI* are sent to the application. The application must answer this message with DISCONNECT_RESP to free the resources allocated to the *PLCI*.

Outgoing physical connections could be cleared for security reason (*Reason* = 0x3305) if the combination of Called party number, Called party subaddress and CIP Value of the corresponding CONNECT_REQ is not allowed by Call Control Supervision. In case of overlap sending security clearing could occur after any INFO_REQ that builds a Called party number which is not allowed. In case of overlap receiving security clearing could occur after any INFO_IND that builds a Calling party number which is not allowed (see Annex D.2).

5.28 DISCONNECT_RESP

Description

With this message, the application acknowledges the clearing down of the physical channel.

DISCONNECT_RESP	Command	**0x04**
	Subcommand	**0x83**

Parameter	Type	Comment
PLCI	dword	Physical Link Connection Identifier

Note

With this message, the *PLCI* is released.

If an application fails to send this message after receiving DISCONNECT_IND**,** any resources associated with this *PLCI* are not freed. This may lead to **COMMON-ISDN-API** resource problems affecting other applications too, indicated by info value **0x2003**: **No PLCI available**.

5.29 FACILITY_REQ

Description

This message is used to handle optional facilities on the controller or facilities related to connections identified by Controller, PLCI or NCCI. At the moment, facility support is defined for handsets, DTMF, V.42 bis, Supplementary Services and power management wakeup.

Handset, DTMF, V.42 bis, Supplementary Services and power management wakeup support are optional **COMMON-ISDN-API** features. In the case that **COMMON-ISDN-API** does not support these facilities, an appropriate information value is returned in the FACILITY_CONF.

DTMF can not be used with all B protocols. Normally it is used with 64 kbit/sec speech and T.30 audio. Supplementary Services may be used with all B protocols. Normally they are used with speech services. However, hold/retrieve, terminal portability functions and especially call forwarding are defined operations for other services such as data communications as well. Line Interconnect is also primarily intended for speech services but may also be used for data applications. The use of power management wakeup is independent of the selected B channel protocol.

FACILITY_REQ	Command	**0x80**
	Subcommand	**0x80**

Parameter	Type	Comment
Controller/PLCI/NCCI	dword	Depending on the facility selector
Facility selector	word	**0x0000:** Handset **0x0001:** DTMF **0x0002:** V.42 bis **0x0003:** Supplementary Services (see Part III) **0x0004:** Power management wakeup **0x0005**: Line Interconnect
Facility request parameter	struct	Facility-dependent parameters

5.30 FACILITY_CONF

Description

This message confirms the acceptance of the FACILITY_REQ. Any error is coded in the parameter *Info*.

FACILITY_CONF	Command	**0x80**
	Subcommand	**0x81**

Parameter	Type	Comment
Controller/PLCI/NCCI	dword	Depending on the facility selector
Info	word	**0:** Request accepted **0x2001:** Message not supported in current state **0x2002:** Incorrect Controller/PLCI/NCCI **0x2007:** Illegal message parameter coding **0x2008:** No interconnection resources available **0x300B:** Facility not supported **0x3011:** Facility specific function not supported
Facility selector	word	**0x0000:** Handset **0x0001:** DTMF **0x0002:** V.42 bis **0x0003:** Supplementary Services (see Part III) **0x0004:** Power management wakeup **0x0005**: Line Interconnect
Facility confirmation parameter	struct	Facility-dependent parameters

Note

In case of facility selector **3** (Supplementary Services) this message may allocate a new PLCI (in the case of resuming a suspended call). This PLCI must be released later by means of DISCONNECT_IND / DISCONNECT_RESP.

If a **COMMON-ISDN-API** implementation supports the facility selector **4** (power management wakeup) its behavior has to differ from one that does not support the facility selector **4**.

5.31 FACILITY_IND

Description

This message is used to indicate a facility-dependent event originating on a controller or connection identified by the facility-dependent parameter *Controller*/*PLCI*/*NCCI*.

FACILITY_IND	Command	**0x80**
	Subcommand	**0x82**

Parameter	Type	Comment
Controller/PLCI/NCCI	dword	Depending on the facility selector
Facility selector	word	**0x0000:** Handset Support **0x0001:** DTMF **0x0002:** V.42 bis **0x0003:** Supplementary Services (see Part III) **0x0004**: reserved **0x0005**: Line Interconnect
Facility indication parameter	struct	Facility-dependent parameters

Note

In case of facility selector **0** (Handset Support) this message may allocate a new PLCI (in the case that the handset goes off-hook). This PLCI must be released later by means of DISCONNECT_IND / DISCONNECT_RESP.

5.32 FACILITY_RESP

Description

With this message, the application acknowledges receipt of a facility indication message.

FACILITY_RESP	Command	0x80
	Subcommand	0x83

Parameter	Type	Comment
Controller/PLCI/NCCI	dword	Depending on the facility selector
Facility selector	word	**0x0000:** Handset Support **0x0001:** DTMF **0x0002:** V.42 bis **0x0003:** Supplementary Services (see Part III) **0x0004**: reserved **0x0005**: Line Interconnect
Facility response parameters	struct	Facility-dependent parameters

5.33 INFO_REQ

Description

This message permits sending of protocol information, such as overlap sending, for a physical connection.

INFO_REQ	Command	**0x08**
	Subcommand	**0x80**

Parameter	Type	Comment
Controller/PLCI	dword	See note
Called party number	struct	Called party number
Additional Info	struct	Additional information elements

Note

The first parameter identifies a physical connection (if a PLCI is given) or the addressed controller (if the PLCI field of parameter *Controller/PLCI* is zero). Different messages are sent to the network depending on this parameter.

5.34 INFO_CONF

Description

This message confirms acceptance of an INFO_REQ. If a controller is given as an addressing parameter in the corresponding INFO_REQ, this connection may be assigned a *PLCI*, which serves as an identifier in further processing. Any error is coded in the parameter *Info*.

INFO_CONF	Command	**0x08**
	Subcommand	**0x81**

Parameter	**Type**	**Comment**
PLCI	dword	Physical Link Connection Identifier
Info	word	**0:** Transmission of information initiated **0x2001:** Message not supported in current state **0x2002:** Illegal Controller/PLCI **0x2003:** No PLCI available **0x2007:** Illegal message parameter coding

5.35 INFO_IND

Description

This message indicates an event for a physical connection as expressed by an information element (*Info element*) whose coding is described by the parameter *Info number*. The connection is identified via the parameter *Controller/PLCI*.

INFO_IND	Command	**0x08**
	Subcommand	**0x82**

Parameter	Type	Comment
Controller/PLCI	dword	Controller / Physical Link Connection Identifier
Info number	word	Information element identifier
Info element	struct	Information element dependent structure

Note

An individual **INFO_IND** is sent for each information element. To enable indication of events, the info mask parameter of the message **LISTEN_REQ** has to be used.

If the *PLCI* field in the *Controller/PLCI* parameter is zero, the network has sent information not associated with a physical connection.

When information is received from the network which leads to other **COMMON-ISDN-API** messages (as when the controller receives a RELEASE from the network which includes charge information), it is guaranteed that an application receives the **INFO_IND** messages before the other **COMMON-ISDN-API** messages. There is one exception: information related to new connections (e.g. information included in an incoming SETUP) will be indicated after the corresponding message (CONNECT_IND).

5.36 INFO_RESP

Description

With this message, the application acknowledges the receipt of an **INFO_IND**.

INFO_RESP	Command	**0x08**
	Subcommand	**0x83**

Parameter	**Type**	**Comment**
Controller/PLCI	dword	Controller / Physical Link Connection Identifier (as in INFO_IND)

5.37 LISTEN_REQ

Description

This message is used to activate signaling of incoming events from **COMMON-ISDN-API** to the application. *Info mask* is used to define which signaling protocol events are to be indicated to the application. These events are normally associated with physical connections. *CIP Mask* defines selection criteria based upon *Bearer Capability* and *High Layer Compatibility*, thus specifying which incoming calls are signaled to an application.

More than one application may listen to the same *CIP Values*. Every application listening to a matching value is informed about incoming calls. If more than one application attempts to accept the call, the first CONNECT_RESP received by **COMMON-ISDN-API** is accepted. Every other application receives a DISCONNECT_IND message which indicates this situation in the *Reason* parameter.

LISTEN_REQ	Command	**0x05**
	Subcommand	**0x80**

Parameter	**Type**	**Comment**
Controller	dword	
Info mask	dword	Bit field, coding as follows: **[0]:** Cause **[1]:** Date/time **[2]:** Display **[3]:** User-user information **[4]:** Call progression **[5]:** Facility **[6]:** Charging **[7]:** Called party number **[8]:** Channel information **[9]:** Early B3 connect **[10]**: Redirecting/redirection information **[11]:** reserved **[12]:** Sending Complete **[13 to 31]:** reserved
CIP Mask	dword	explained below
CIP Mask 2	dword	reserved for additional services
Calling party number	struct	Calling party number
Calling party subaddress	struct	Calling party subaddress

Explanation of *CIP Mask*:

Parameter	Type	Comment
CIP Mask	dword	Bit field, coding as follows: **[0]:** Any match **[1]:** Speech **[2]:** Unrestricted digital information **[3]:** Restricted digital information **[4]:** 3.1 kHz audio **[5]:** 7.0 kHz audio **[6]:** Video **[7]:** Packet mode **[8]:** 56 kbit/s rate adaptation **[9]:** Unrestricted digital information with tones/announcements **[10 to 15]:** reserved **[16]:** Telephony **[17]:** Group 2/3 fax **[18]:** Group 4 fax class 1 **[19]:** Teletex service (basic and mixed), fax group 4 class 2 **[20]:** Teletex service (basic and processable) **[21]:** Teletex service (basic) **[22]:** Videotex **[23]:** Telex **[24]:** Message handling systems according X.400 **[25]:** OSI applications according X.200 **[26]:** 7 kHz Telephony **[27]:** Video Telephony F.721, first connection **[28]:** Video Telephony F.721, second connection **[29 to 31]:** reserved

Note

Clearing all bits in the *CIP Mask* disables the signaling of incoming calls to the application.

Calling party number/subaddress are used only for external ISDN equipment (handsets), which might need the (local) line's own address to handle *outgoing* calls.

5.38 LISTEN_CONF

Description

This message confirms the acceptance of the LISTEN_REQ. Any errors are coded in the parameter *Info*.

LISTEN_CONF	Command	**0x05**
	Subcommand	**0x81**

Parameter	Type	Comment
Controller	dword	
Info	word	**0:** Listen is active **0x2002:** Illegal controller **0x2005:** No Listen resources available **0x2007:** Illegal message parameter coding

5.39 MANUFACTURER_REQ

Description

This message is used to transfer manufacturer-specific information.

MANUFACTURER_REQ	Command	0xFF
	Subcommand	0x80

Parameter	Type	Comment
Controller	dword	
Manu ID	dword	Manufacturer-specific ID (should be unique)
Manufacturer-specific		Manufacturer-specific data

Note

This message should be avoided since it is a non-compatible message. Applications which use this message can only work with ISDN equipment by one manufacturer.

A manufacturer shall choose *one* manufacturer-specific ID for all of its **COMMON-ISDN-API** implementations. This manufacturer-specific ID shall be unique. An abbreviation or nickname based on the manufacturer's name might be a good choice.

The behavior of **COMMON-ISDN-API** after receiving any MANUFACTURER_REQ is **not defined**.

5.40 MANUFACTURER_CONF

Description

This message confirms receipt of a MANUFACTURER_REQ.

MANUFACTURER_CONF	Command	**0xFF**
	Subcommand	**0x81**

Parameter	Type	Comment
Controller	dword	
Manu ID	dword	Manufacturer-specific ID (should be unique)
Manufacturer-specific		Manufacturer-specific data

5.41 MANUFACTURER_IND

Description

This message is used to indicate manufacturer-specific information to an application. **COMMON-ISDN-API** must not generate this message unless it is requested by a MANUFACTURER_REQ.

MANUFACTURER_IND	Command	**0xFF**
	Subcommand	**0x82**

Parameter	**Type**	**Comment**
Controller	dword	
Manu ID	dword	Manufacturer-specific ID (should be unique)
Manufacturer-specific		Manufacturer-specific data

Note

This message shall not be sent by **COMMON-ISDN-API** except on prior request from an application by means of MANUFACTURER_REQ.

5.42 MANUFACTURER_RESP

Description

With this message, an application confirms receipt of a MANUFACTURER_IND.

MANUFACTURER_RESP	Command	**0xFF**
	Subcommand	**0x83**

Parameter	Type	Comment
Controller	dword	
Manu ID	dword	Manufacturer-specific ID (should be unique)
Manufacturer-specific		Manufacturer-specific data

5.43 RESET_B3_REQ

Description

With this message, the application initiates a reset of the specified logical connection. The logical connection is identified by the parameter *NCCI*.

RESET_B3_REQ	Command	**0x87**
	Subcommand	**0x80**

Parameter	Type	Comment
NCCI	dword	Network Control Connection Identifier
NCPI	struct	Network Control Protocol Information

Note

The meaning of the parameter *NCPI* depends on the protocol used.

The reaction to a **RESET_B3_REQ** depends on the selected Layer 3 protocol. If ISO 8208, T.90 or X.25 DCE was selected, the reset procedure is performed in accordance with the protocol recommendations. In the case of a transparent Layer 3, a reset procedure in Layer 2 is initiated.

In the case of bit-transparent data, i.e. B1 Protocol 1 or B1 Protocol 6 (64 or 56 kbit/s bit-transparent operation with byte framing from the network), B2 Protocol 1 (transparent) and B3 Protocol 0 (transparent), all pending transmit data located in internal buffers is invalidated. The exact amount of data which is invalidated depends on the given implementation and is therefore not predictable.

If a reset procedure is not defined for the protocol, a **RESET_B3_REQ** causes the controller to generate a RESET_B3_CONF with info value **Reset procedure not supported by current protocol** (0x300D). No further action is taken.

After successfully initiating a reset on a logical connection, an application is not allowed to transmit data until the resulting RESET_B3_IND (or DISCONNECT_B3_IND) message is received.

Data loss may occur during the reset procedure!

5.44 RESET_B3_CONF

Description

With this message, the controller confirms the initiation of a logical connection reset.

RESET_B3_CONF	Command	**0x87**
	Subcommand	**0x81**

Parameter	Type	Comment
NCCI	dword	Network Control Connection Identifier
Info	word	0: Reset initiated 0x0001: NCPI not supported by current protocol, NCPI ignored 0x2001: Message not supported in current state 0x2002: Illegal NCCI 0x2007: Illegal message parameter coding 0x3008: NCPI not supported 0x300D: Reset procedure not supported by current protocol

5.45 RESET_B3_IND

Description

This message indicates that a logical connection has been reset. The logical connection is identified by the *NCCI*.

RESET_B3_IND	Command	**0x87**
	Subcommand	**0x82**

Parameter	Type	Comment
NCCI	dword	Network Control Connection Identifier
NCPI	struct	Network Control Protocol Information

Note

The meaning of the parameter *NCPI* depends on the protocol used.

In the case of a transparent Layer 3, the re-establishment of Layer 2 is indicated.

This message indicates a possible loss of data!

5.46 RESET_B3_RESP

Description

With this message, the application acknowledges the resetting of a logical connection.

RESET_B3_RESP	Command	**0x87**
	Subcommand	**0x83**

Parameter	Type	Comment
NCCI	dword	Network Control Connection Identifier

5.47 SELECT_B_PROTOCOL_REQ

Description

This message allows an application to change the current protocol during a physical connection, i. e. after receipt of the message CONNECT_ACTIVE_IND. Support for this message is optional. If a particular **COMMON-ISDN-API** implementation does not support this change, the *Info* parameter of the corresponding SELECT_B_-PROTOCOL_CONF is set to **Message not supported in current state** (0x2001).

SELECT_B_PROTOCOL_REQ	Command	**0x41**
	Subcommand	**0x80**

Parameter	**Type**	**Comment**
PLCI	dword	Physical Link Connection Identifier
B protocol	struct	Protocol definition

5.48 SELECT_B_PROTOCOL_CONF

Description

This message confirms the change of protocol stack for a physical connection. Any error is shown in the *Info* parameter.

SELECT_B_PROTOCOL_CONF	Command	**0x41**
	Subcommand	**0x81**

Parameter	Type	Comment
PLCI	dword	Physical Link Connection Identifier
Info	word	**0:** Protocol change successful **0x2001:** Message not supported in current state **0x2002:** Illegal PLCI **0x2007:** Illegal message parameter coding **0x3001:** B1 protocol not supported **0x3002:** B2 protocol not supported **0x3003:** B3 protocol not supported **0x3004:** B1 protocol parameter not supported **0x3005:** B2 protocol parameter not supported **0x3006:** B3 protocol parameter not supported **0x3007:** B protocol combination not supported

6 PARAMETER DESCRIPTION

This section describes the parameters used in **COMMON-ISDN-API** messages. Each parameter is listed with its data type, possible values and reference to the messages in which the parameter appears.

Some parameter values are defined in accordance with ETS 300 102-1 or Q.931. There is no private **COMMON-ISDN-API** coding for such parameters. They are coded as **COMMON-ISDN-API** structures starting with a length octet and the remainder of the parameter coded as defined in ETS 300 102-1 / Q.931 from octet three onwards. References to the contents of a structure in this chapter always use the index 0 to identify the first octet of information, i.e. the octet *following* the length octet.

Parameters may not be omitted. Instead, an empty structure shall be used. An empty structure is coded as a single length octet containing a value of zero.

Default values as described in the following section must be implemented in **COMMON-ISDN-API**. They need not be valid for external ISDN equipment. In case they are not, the external equipment defines the default values for its use.

Parameters may themselves contain parameters, which are then referred to as “sub-parameters”.

6.1 Protocol-Independent Parameters

Additional Info (struct)

The purpose of the parameter *additional info* is to exchange information specific to the signaling protocol of the network. Depending on the signaling protocol, only the relevant elements of this structure are used. For example, the B channel information is ignored in the message DISCONNECT_REQ.

The parameter has the following structure:

struct	**B channel information**
struct	**Keypad facility (coded in accordance with ETS 300 102-1 / Q.931)**
struct	**User-user data (coded in accordance with ETS 300 102-1 / Q.931)**
struct	**Facility data array, which is used to transfer additional parameters coded in accordance with ETS 300 102-1 / Q.931 starting from octet 1. This field is used to transport one or more complete facility data information elements.**
struct	**Sending complete**

This information element appears in:

ALERT_REQ
CONNECT_REQ
CONNECT_IND
CONNECT_RESP
DISCONNECT_REQ

B Channel Information (struct)

The purpose of the sub-parameter *B channel information* is to choose between B channel data exchange, D channel data exchange or pure user-user data exchange. If this struct is empty, the default value is assumed.

This sub-parameter is coded as a structure. Depending on the parameter *Channel* (the first element of the structure), additional information is included. Parameter *Channel* can have following values:

0: **Use B channel**
1: **Use D channel**
2: **Use neither B channel nor D channel**
3: **Use channel allocation (leased lines only)**
4: **Use channel identification information element**

The struct *B channel information* is coded as follows:

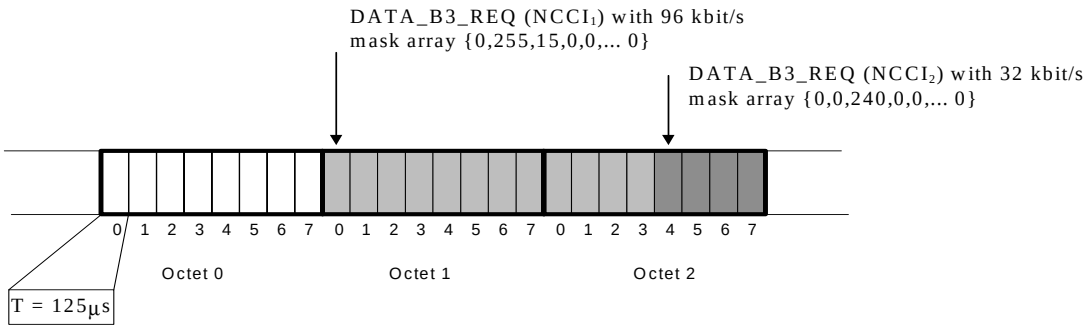
Use B channel; use D channel; use neither B channel nor D channel:

Channel	word	0: Use B channel (default value) 1: Use D channel 2: Use neither B channel nor D channel

Use channel allocation (leased lines only):

Channel	word	3: Use channel allocation (leased lines only)
Operation	word	0: DTE (originate) mode (default value) 1: DCE (answer) mode
Channel mask array	byte array[31]	0: D Channel mask (default value: 0) BRI (16 kbit/s): bits 6-7 are used bits 0-5 are reserved PRI (64 kbit/s): bits 0-7 are used 1: B channel 1 mask (default : 255) 2: B channel 2 mask (default : 0) 3…30: B channel 3...30 mask ignored in case of BRI (default value: 0)

The parameter *Operation* defines the mode (DTE or DCE) in which the B-channel protocols (e.g. X.75 or X.25, etc.) are operated. The *Channel mask array* specifies the channels and subchannels to be bundled to one physical connection. In each mask byte, the least significant bit (LSB) corresponds to the LSB of the respective channel on the BRI or PRI. Bandwidth values may be selected within the range 0..64 kbit/s in units of 8 kbit/s. The default value for this mask array is {0, 255, 0, 0, …, 0}, thus allocating B Channel 1 with 64 kbit/s on a BRI. In the case of a BRI, the unused *channel mask array* bytes 3..30 shall be ignored. See the example below for different channel allocations:



Two connections with different bandwidths (96 kbit/s and 32 kbit/s)

A leased line connection is always initiated by a CONNECT_REQ. Unused parameters (e.g. Called Party Number) are ignored. The message flow is as described in Annex A, A.1.1 Outgoing call.

The message CONNECT_CONF with *Info* value 0x300E indicates overlapping channel masks (such as {0,255,15,0,...} and {0,0,255,0,…}) in different CONNECT_REQ messages.

Channel allocation can be used on leased lines only.

Use channel identification information element:

Channel	word	4: Use channel identification information element
Channel Identification	struct	Coded in accordance with ETS 300 102-1 / Q.931

The purpose of the parameter *Channel Identification* is to identify a channel within the interface(s) controlled by these signaling procedures.

This sub-parameter appears in the parameter:

Additional information

B Protocol (struct)

The purpose of the parameter *B protocol* is to select and configure the B channel protocols. The parameter *B protocol* is protocol-dependent: see Subclause 6.2.

B1 Protocol (word)

The purpose of the sub-parameter *B1 protocol* is to specify the physical layer and framing used for this connection. The sub-parameter *B1 protocol* is protocol-dependent: see Subclause 6.2.

B2 Protocol (word)

The purpose of the sub-parameter *B2 protocol* is to specify the data link layer used for this connection. The sub-parameter *B2 protocol* is protocol-dependent: see Subclause 6.2.

B3 Protocol (word)

The purpose of the sub-parameter *B3 protocol* is to specify the network layer used for this connection. The sub-parameter *B3 protocol* is protocol-dependent: see Subclause 6.2.

B1 Configuration (struct)

The purpose of the sub-parameter *B1 configuration* is to offer additional configuration information for the B1 protocol. The sub-parameter *B1 configuration* is protocol-dependent: see Subclause 6.2.

B2 Configuration (struct)

The purpose of the sub-parameter *B2 configuration* is to offer additional configuration information for the B2 protocol. The sub-parameter *B2 configuration* is protocol-dependent: see Subclause 6.2.

B3 Configuration (struct)

The purpose of the sub-parameter *B3 configuration* is to offer additional configuration information for the B3 protocol. The sub-parameter *B3 configuration* is protocol-dependent: see Subclause 6.2.

BC (struct)

The purpose of the parameter *Bearer Capability (BC)* is to indicate a requested CCITT Recommendation I.231 bearer service to be provided by the network. It contains information which may be used only by the network. The information element is coded in accordance with ETS 300 102-1 / Q.931.

This information element appears in:

CONNECT_IND
CONNECT_REQ

Called Party Number (struct)

The purpose of the parameter *Called party number* is to identify the party called in the call establishment process. The information element is coded in accordance with ETS 300 102-1 / Q.931.

Byte 0	Type of number and numbering plan identification (byte 3 of the *Called party number* information element: see ETS 300 102). The value supplied by the application at the calling end is transmitted over the network. **0x80** is the suggested default value. At the called end, the value received from the network is passed to the application.
Bytes 1…n	Digits of the *Called party number* information element.

This information element appears in:

CONNECT_IND
CONNECT_REQ

Called Party Subaddress (struct)

The purpose of the parameter C*alled party subaddress* is to identify the subaddress of the party called in the call establishment process. The information element is coded in accordance with ETS 300 102-1 / Q.931.

Byte 0	Type of subaddress The value supplied by the application at the calling end is transmitted over the network. **0x80** is the suggested default value (NSAP in accordance with X.213). In this case, the first subaddress information octet should have the value **0x50**. At the called end, the value received from the network is passed to the application.
Bytes 1…n	Contents of the information element *Called party subaddress*.

This information element appears in:

CONNECT_IND
CONNECT_REQ

Calling Party Number (struct)

The purpose of the parameter *Calling party number* is to identify the origin of a call. The information element is coded in accordance with ETS 300 102-1 / Q.931.

Byte 0	Type of number and numbering plan identification (byte 3 of the *Calling party number* information element, see ETS 300 102). The value supplied by the application at the calling end is transmitted over the network. **0x00** is the suggested default value. At the called interface, the value received from the network is passed to the application. The extension bit is always cleared.
Byte 1	Presentation and screening indicator (byte 3a of the *Calling party number* information element). This byte may be used to allow or suppress the presentation of the caller's number in an incoming call. The value supplied by the application at the originating interface is transmitted over the network. **0x80** is the suggested default value, which allows presentation of the caller's number. **0xA0** suppresses presentation of the calling number if the network supports this mechanism. At the called interface, the value received from the network is passed to the application. If this byte was not transmitted from the network, the controller inserts the valid default value **0x80** (user-provided, not screened).
Bytes 2…n	Digits of the information element *Calling party number*.

This information element appears in:

CONNECT_IND
CONNECT_REQ
LISTEN_REQ

Calling Party Subaddress (struct)

The purpose of the parameter *Calling party subaddress* is to identify a subaddress associated with the origin of a call. The information element is coded in accordance with ETS 300 102-1 / Q.931.

Byte 0	Type of subaddress The value supplied by the application at the calling end is transmitted over the network. **0x80** is the suggested default value (NSAP in accordance with X.213). In this case, byte 1 should have the value **0x50**. At the called end, the value received from the network is passed to the application.
Bytes 1…n	Contents of the calling party subaddress information element.

This information element appears in:

CONNECT_IND
CONNECT_REQ
LISTEN_REQ

CIP Value (word)

The purpose of parameter *CIP Value* is to identify a complete profile of the compatibility information *Bearer Capability*, *Low Layer Compatibility* and *High Layer Compatibility*. With this parameter, standard applications are not required to do complex coding and decoding of these individual information elements.

Some of the *CIP* values only define a *Bearer Capability* (*CIP* 1 to 9), and some define a combination of *Bearer Capability* and *High Layer Compatibility* (*CIP* 16 to 28). A *Low Layer Compatibility* information element is not defined by the *CIP*, but must be provided by the application if necessary.

The following *CIP* values are defined:

CIP value	Service	Relation to BC/HLC
0		No predefined profile
1	**Speech**	Bearer capability: Coding standard: CCITT Information transfer capability: speech Transfer mode: circuit mode Information transfer rate: 64 kbit/s User information Layer 1 protocol: G.711 Coding of BC: <0x04, 0x03, 0x80, 0x90, 0xA3> or <0x04, 0x03, 0x80, 0x90, 0xA2>(see note)

2	**Unrestricted digital information**	Bearer capability: Coding standard: CCITT Information transfer capability: unrestricted digital information Transfer mode: circuit mode Information transfer rate: 64 kbit/s Coding of BC: <0x04, 0x02, 0x88, 0x90>
3	**Restricted digital information**	Bearer capability: Coding standard: CCITT Information transfer capability: restricted digital information Transfer mode: circuit mode Information transfer rate: 64 kbit/s Coding of BC: <0x04, 0x02, 0x89, 0x90>
4	**3.1 kHz audio**	Bearer capability: Coding standard: CCITT Information transfer capability: 3.1 kHz audio Transfer mode: circuit mode Information transfer rate: 64 kbit/s User information Layer 1 protocol: G.711 Coding of BC: <0x04, 0x03, 0x90, 0x90, 0xA3> or <0x04, 0x03, 0x80, 0x90, 0xA2>(see note)
5	**7 kHz audio**	Bearer capability: Coding standard: CCITT Information transfer capability: unrestricted digital information with tones/announcements Transfer mode: circuit mode Information transfer rate: 64 kbit/s Coding of BC: <0x04, 0x02, 0x91, 0x90>
6	**Video**	Bearer capability: Coding standard: CCITT Information transfer capability: video Transfer mode: circuit mode Information transfer rate: 64 kbit/s Coding of BC: <0x04, 0x02, 0x98, 0x90>
7	**Packet mode**	Bearer capability: Coding standard: CCITT Information transfer capability: unrestricted digital information Transfer mode: packet mode Information transfer rate: packet mode Layer 2 protocol: X.25 Layer 2 Layer 3 protocol: X.25 Layer 3 Coding of BC: <0x04, 0x04, 0x88, 0xC0, 0xC6, 0xE6>
8	**56 kbit/s rate adaptation**	Bearer capability: Coding standard: CCITT Information transfer capability: unrestricted digital information Transfer mode: circuit mode Layer 1 protocol: CCITT standardized rate adaptation V.110/X.30 Information transfer rate: packet mode Rate: 56 kbit/s Coding of BC: <0x04, 0x04, 0x88, 0x90, 0x21, 0x8F>
9	**Unrestricted digital information with tones/announcements**	Bearer capability: Coding standard: CCITT Information transfer capability: unrestricted digital information with tones/announcements Transfer mode: circuit mode Information transfer rate: 64 kbit/s Layer 1 protocol: H.221, H.242 Coding of BC: <0x04, 0x03, 0x91, 0x90, 0xA5>
10..15	**Reserved**	

16	**Telephony**	Bearer Capability as for CIP 1. High Layer Compatibility: Coding standard: CCITT Interpretation: First characteristics identification is to be used Presentation: High layer protocol profile High layer characteristics identification: Telephony Coding of HLC: <0x7D, 0x02, 0x91, 0x81>
17	**Group 2/3 facsimile**	Bearer Capability as for CIP 4. High Layer Compatibility: Coding standard: CCITT Interpretation: First characteristics identification is to be used Presentation: High layer protocol profile High layer characteristics identification: Group 2/3 facsimile Coding of HLC: <0x7D, 0x02, 0x91, 0x84>
18	**Group 4 facsimile Class 1**	Bearer Capability as for CIP 2. High Layer Compatibility: Coding standard: CCITT Interpretation: First characteristics identification is to be used Presentation: High layer protocol profile High layer characteristics identification: Group 4 facsimile Class 1 Coding of HLC: <0x7D, 0x02, 0x91, 0xA1>
19	**Teletex service basic and mixed mode and Group 4 facsimile service Classes II and III**	Bearer Capability as for CIP 2. High Layer Compatibility: Coding standard: CCITT Interpretation: First characteristics identification is to be used Presentation: High layer protocol profile High layer characteristics identification. Teletex service and Group 4 facsimile service Coding of HLC: <0x7D, 0x02, 0x91, 0xA4>
20	**Teletex service basic and processable mode**	Bearer Capability as for CIP 2. High Layer Compatibility: Coding standard: CCITT Interpretation: First characteristics identification is to be used Presentation: High layer protocol profile High layer characteristics identification. Teletex service basic and processable mode Coding of HLC: <0x7D, 0x02, 0x91, 0xA8>
21	**Teletex service basic mode**	Bearer Capability as for CIP 2. High Layer Compatibility: Coding standard: CCITT Interpretation: First characteristics identification is to be used Presentation: High layer protocol profile High layer characteristics identification. Teletex service basic mode Coding of HLC: <0x7D, 0x02, 0x91, 0xB1>
22	**International interworking for Videotex**	Bearer Capability as for CIP 2. High Layer Compatibility: Coding standard: CCITT Interpretation: First characteristics identification is to be used Presentation: High layer protocol profile High layer characteristics identification. International interworking for Videotex Coding of HLC: <0x7D, 0x02, 0x91, 0xB2>

23	**Telex**	Bearer Capability as for CIP 2. High Layer Compatibility: Coding standard: CCITT Interpretation: First characteristics identification is to be used Presentation: High layer protocol profile High layer characteristics identification: Telex Coding of HLC: <0x7D, 0x02, 0x91, 0xB5>
24	**Message Handling Systems in accordance with X.400**	Bearer Capability as for CIP 2. High Layer Compatibility: Coding standard: CCITT Interpretation: First characteristics identification is to be used Presentation: High layer protocol profile High layer characteristics identification: Message Handling Systems (in accordance with X.400) Coding of HLC: <0x7D, 0x02, 0x91, 0xB8>
25	**OSI application in accordance with X.200**	Bearer Capability as for CIP 2. High Layer Compatibility: Coding standard: CCITT Interpretation: First characteristics identification is to be used Presentation: High layer protocol profile High layer characteristics identification: OSI application in accordance with X.200 Coding of HLC: <0x7D, 0x02, 0x91, 0xC1>
26	**7 kHz telephony**	Bearer Capability as for CIP 9. High Layer Compatibility: Coding standard: CCITT Interpretation: First characteristics identification is to be used Presentation: High layer protocol profile High layer characteristics identification: Telephony Coding of HLC: <0x7D, 0x02, 0x91, 0x81>
27	**Video telephony, first connection**	Bearer Capability as for CIP 9. High Layer Compatibility: Coding standard: CCITT Interpretation: First characteristics identification is to be used Presentation: High layer protocol profile High layer characteristics identification: Video telephony (F.721) Extended high layer characteristics identification: Capability set of initial channel per H.221 Coding of HLC: <0x7D, 0x03, 0x91, 0x60, 0x01>
28	**Video telephony, second connection**	Bearer Capability as for CIP 2 High Layer Compatibility: Coding standard: CCITT Interpretation: First characteristics identification is to be used Presentation: High layer protocol profile High layer characteristics identification: Video telephony (Rec. F.721) Extended high layer characteristics identification: Capability set of subsequent channel per H.221 Coding of HLC: <0x7D, 0x03, 0x91, 0x60, 0x02>

Note: This coding applies to ISDN with A-Law default coding for speech/audio. For ISDN with μ-Law default coding, the corresponding values are used.

This information element appears in:

CONNECT_IND
CONNECT_REQ

CIP Mask (dword)

The purpose of the parameter *CIP Mask* is to select basic classes of incoming calls. The bit position within this mask identifies the related CIP value. When an incoming call is received, **COMMON-ISDN-API** tries to match this incoming call to the enabled CIP values (more than one value may match). A CONNECT_IND message is sent to the application if the bit position within the *CIP Mask* of any matching CIP value is set to 1. The CIP value in the CONNECT_IND message is set to the highest matching CIP value.

The following rules are defined for determining matching CIP values:

1. CIP values defining a Bearer Capability only (CIP values 1 to 9) generate a match with any incoming call that includes this Bearer Capability information. Additional information included in the Bearer Capability information element is ignored. The match is generated regardless of any Low Layer Compatibility or High Layer Compatibility received.

2. CIP values defining a Bearer Capability and a High Layer Compatibility (CIP values 16 to 28) generate a match with any incoming call that includes a Bearer Capability and a High Layer Compatibility with the same information. The match is generated regardless of any Low Layer Compatibility received.

Bit 0 in the *CIP Mask* has a special meaning. When no other matching bit is set in the *CIP Mask* except bit 0, a CONNECT_IND is sent to the application with a CIP value of 0. In this case, the application must evaluate the parameters Bearer Capability, Low Layer Compatibility and High Layer Compatibility to decide whether or not it is compatible with the call.

Examples:

Service	Bits to be set in the CIP mask	
Telephony	1	For calls within ISDN from equipment which does not send High Layer Compatibility info.
	4	For calls from the analog network.
	16	For call within ISDN from equipment which sends High Layer Compatibility info.
Group 2/3 fax	4	For calls from the analog network.
	17	For calls within ISDN.
Non-standard 64 kbit/s data application	2	No checking of High Layer Compatibility information is provided. The application should verify that no High Layer Compatibility information is received.
Non-standard 56 kbit/s data application	8	No checking of High Layer Compatibility information is provided. The application must verify that no High Layer Compatibility information is received.
Group 4 fax	2	For calls from equipment which does not send High Layer Compatibility information. The application must verify that no High Layer Compatibility information is received.
	18	For calls from equipment which sends High Layer Compatibility information.

This information element appears in:

LISTEN_REQ

Connected Number (struct)

The purpose of the parameter *Connected number* is to indicate which number is connected to a call. The information element is coded in accordance with ETS 300 097.

Byte 0	Type of number and numbering plan identification (byte 3 of the connected number information element; see ETS 300 097). In the direction from application to **COMMON-ISDN-API**, the value supplied by the application is transmitted over the network. **0x00** is the suggested default value. In the direction from **COMMON-ISDN-API** to application, the value received from the network is passed to the application. The extension bit is always cleared.
Byte 1	Presentation and screening indicator (byte 3a of the connected number information element). In the direction from application to **COMMON-ISDN-API**, the value supplied by the application is transmitted over the network. **0x80** is the suggested default value. In the direction from **COMMON-ISDN-API** to application, the value received from the network is passed to the application. If this byte was not received over the network, the controller supplies the value **0x80** (user-provided, not screened).
Bytes 2...n	Digits of the connected number information element.

This information element appears in:

CONNECT_ACTIVE_IND
CONNECT_RESP

Connected Subaddress (struct)

The purpose of the connected subaddress is to identify the subaddress of the connected user answering a call. The information element is coded in accordance with ETS 300 097.

Byte 0	Type of subaddress The value supplied by the application at the calling end is transmitted over the network. **0x80** is the suggested default value (NSAP in accordance with X.213). In this case, byte 1 should have the value **0x50**. At the called end, the value received from the network is passed to the application.
Bytes 1...n	Contents of the connected subaddress information element.

This information element appears in:

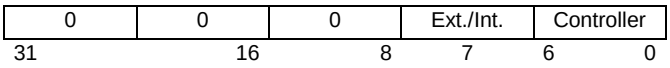
CONNECT_ACTIVE_IND
CONNECT_RESP

Controller (dword)

The purpose of the parameter *Controller* is to address a hardware unit that provides the application with access to ISDN. A *controller* may support zero, one or several physical and logical connections. The parameter *Controller* is a dword (to be compatible in size with PLCI and NCCI) in the range from 1 to 127 (0 is reserved). Bit 7 additionally indicates whether the message applies to internal (0) or external (1) equipment. Controllers are numbered sequentially and can be designed to handle external equipment in addition to internal capabilities, or may provide access exclusively to external equipment, such as a handset, for example.

External equipment behavior, such as B channel handling, is not defined by **COMMON-ISDN-API**.

Format of *Controller*:



This information element appears in:

CONNECT_REQ
FACILITY_REQ
FACILITY_CONF
FACILITY_IND
FACILITY_RESP
LISTEN_REQ
LISTEN_CONF
MANUFACTURER_REQ
MANUFACTURER_CONF
MANUFACTURER_IND
MANUFACTURER_RESP

Data (dword)

The purpose of the parameter *Data* is to hold a 32-bit pointer to the data area containing the information.

This information element appears in:

DATA_B3_REQ
DATA_B3_IND

Data64 (qword)

The purpose of the parameter *Data64* is to hold a 64-bit pointer to the data area containing the information (64bit applications only).

This information element appears in:

DATA_B3_REQ
DATA_B3_IND

Data Length (word)

The purpose of the parameter *Data length* is to specify the length of the data area.

This information element appears in:

DATA_B3_REQ
DATA_B3_IND

Data Handle (word)

The purpose of the parameter *Data handle* is to identify the data area referred to in data exchange messages.

This information element appears in:

DATA_B3_REQ
DATA_B3_CONF
DATA_B3_IND
DATA_B3_RESP

DTMF Characteristics (struct)

The purpose of the sub-parameter *DTMF Characteristics* is to specify the characteristics of the DTMF recognition.

DTMF Selectivity	word	0 (default): default behavior (implementation dependent) 1 to 100: specifies the desired frequency selectivity (tolerance) characteristic for the DTMF recognizer, where the value 1 indicates the narrowest selectivity (as close as possible to the Q.24 specification) and the value 100 indicates the broadest selectivity. Any value in the range 1..100 is mapped to an appropriate behavior, which is dependent on the given implementation.

This sub-parameter appears in parameter:

Facility Request Parameter

Facility Awake Request Parameter (struct)

The purpose of the parameter *Facility awake request parameter* is to offer additional information concerning the parameter *Facility request parameter* in case of Facility Selector **4**. It includes a combination of *Called party number* and *CIP Mask* which enables the generation of a CONNECT_IND.

This parameter is coded as a structure with the following elements:

Called party number	struct	Called party number
CIP mask	dword	Compatibility Information Profile mask

This information element appears in:

Facility Request Parameter

Facility Selector (word)

The purpose of the parameter *Facility selector* is to identify the requested **COMMON-ISDN-API** facility.

The defined values are:

0 **Handset (external ISDN equipment)**
1 **DTMF (Dual Tone Multi-Frequency)**
2 **V.42 bis Compression**
3 **Supplementary Services (described in COMMON-ISDN-API Part III)**
4 **Power management wakeup**
5 **Line Interconnect**

This information element appears in:

FACILITY_REQ
FACILITY_CONF
FACILITY_IND
FACILITY_RESP

Facility Request Parameter (struct)

The purpose of the parameter *Facility request parameter* is to offer additional information concerning the message FACILITY_REQ.

This parameter is coded as a structure with the following elements depending on the value of *Facility selector*:

Facility selector:

0 **Parameter does not apply (coded as empty structure)**

1 **DTMF (Dual Tone Multi-Frequency):**

Function	word	1: Start DTMF listen on B channel data 2. Stop DTMF listen 3: Send DTMF digits 4 to n: Reserved
Tone-Duration	word	Time in ms for one digit; default is 40 ms
Gap-Duration	word	Time in ms between digits; default is 40 ms
DTMF-Digits	struct	Characters to be sent, coded as IA5-char. '0' to '9', '*', '#', 'A', 'B', 'C' or 'D'. Each character generates a unique DTMF signal.
DTMF Characteristics	struct	Characteristics of DTMF recognition (interpreted for Function 1 only).

Sending DTMF characters interrupts the transmission of **DATA_B3_REQ** data. After DTMF generation, the data transmission is resumed.

2 **V.42 bis compression:**

Function	word	0: Get compression information

A FACILITY_REQ with *Facility selector* 2 (V.42 bis compression) is valid in all states except State N0 (see Chapter 7: State Diagram).

3 **Supplementary Services: see COMMON-ISDN-API Part III**

4 **Power management wakeup:**

Number of awake request parameters	word	Number of wake up conditions
Facility awake request parameter	struct	Facility awake request parameter

To reduce the frequency of wake-ups (CONNECT_IND) due to line activity, a **COMMON-ISDN-API** application can define a set of *Awake request parameters* for which it wants to receive CONNECT_IND. A **COMMON-ISDN-API** implementation must accept at least a list of 10 *Awake request parameter* structures.

5 **Line Interconnect:**

Function	word	0x0000: Get Supported Services 0x0001: Connect 0x0002: Disconnect
LI Request Parameter	struct	Line Interconnect request parameter

This information element appears in:

FACILITY_REQ

Facility Confirmation Parameter (struct)

The purpose of the parameter *Facility confirmation parameter* is to offer additional information concerning the message FACILITY_CONF.

This parameter is coded as a structure with the following elements depending on the value of *Facility selector*:

Facility selector:

0 **Parameter does not apply (coded as structure with a length of 0)**

1 **DTMF (Dual Tone Multi-Frequency):**

DTMF information	word	0: Sending of DTMF info successfully initiated 1: Incorrect DTMF digit 2: Unknown DTMF request

2 V.42 bis compression:

V.42 bis information	word	0: Information available <>0: Information not available
Compression mode	word	0: No compression (remote site does not support XID exchange in accordance with **COMMON-ISDN-API** specification) 1: V.42 bis
Number of code words	word	Actual value used
Maximum string length	word	Actual value used
Tx total	dword	Number of octets to be transmitted
Tx compressed	dword	Number of octets transmitted after compression
Rx total	dword	Number of octets received
Rx uncompressed	dword	Number of octets after decompression

3 Supplementary Services: see COMMON-ISDN-API Part III

4 Power management wakeup:

Number of accepted awake request parameters	word	0: Not accepted > 0: Number of accepted wake up conditions

5 Line Interconnect:

Function	word	0x0000: Get Supported Services 0x0001: Connect 0x0002: Disconnect
LI Confirmation Parameter	struct	Line Interconnect confirmation parameter

This information element appears in:

FACILITY_CONF

Facility Indication Parameter (struct)

The purpose of the parameter *Facility indication parameter* is to offer additional information concerning the message FACILITY_IND.

This parameter is coded as a structure with the following elements depending on the value of *Facility selector*:

Facility selector:

0 Handset Support:

Handset digits	byte array	Characters received, coded as IA5-char. '0' to '9', '*', '#', 'A', 'B', 'C' or 'D'; or '+': Handset off-hook '-': Handset on-hook

1 DTMF (Dual Tone Multi-Frequency):

DTMF digits	byte array	Received characters, coded as IA5-char. '0' to '9', '*', '#', 'A', 'B', 'C' or 'D'; or 'X': Recognition of fax tone CNG (1.1 kHz) 'Y': Recognition of fax tone CED (2.1 kHz)

2 **V.42 bis compression:**
Parameter does not apply (coded as structure with a length of 0)

3 **Supplementary Services:**
see COMMON-ISDN-API Part III

4 **Power management wakeup:**
Parameter does not apply (coded as structure with a length of 0)

5 **Line Interconnect:**

Function	word	0x0001: Connect Active 0x0002: Disconnect
LI Indication Parameter	struct	Line Interconnect indication parameter

This information element appears in:

FACILITY_IND

Facility Response Parameter (struct)

The purpose of the parameter *Facility response parameter* is to offer additional information concerning the message FACILITY_RESP.

This parameter is coded as a structure with the following elements depending on the value of *Facility selector*:

Facility selector:

0 **Handset (external ISDN equipment) support**
1 **DTMF (Dual Tone Multi-Frequency)**
2 **V.42 bis compression:**
Parameter does not apply (coded as structure with a length of 0)

3 **Supplementary Services:**
see COMMON-ISDN-API Part III

4 **Power management wakeup:**
Parameter does not apply (coded as structure with a length of 0)

This information element appears in:

FACILITY_RESP

Flags (word)

The purpose of the parameter *Flags* is to communicate additional, protocol-dependent information about the data.

Bit 0 Qualifier bit
Bit 1 More-data bit

Bit 2	Delivery confirmation bit
Bit 3	Expedited data bit
Bit 4	Break / UI frame
Bit 15	Framing error bit: data may be invalid (only with appropriate B2 protocol)

This information element appears in:

DATA_B3_REQ
DATA_B3_IND

HLC (struct)

The purpose of the *High Layer Compatibility (HLC)* information element is to provide a means for compatibility checking by the remote user. The information element is coded in accordance with ETS 300 102-1 / Q.931.

This information element appears in:

CONNECT_IND
CONNECT_REQ

Info (word)

The purpose of the parameter *Info* is to provide error information to the application. A unique code is defined for each error which can be detected by the controller. This code is independent of the error context.

COMMON-ISDN-API shall not generate other information values than those defined below. In case additional information values are defined in future, however, an application should interpret any information value except class **0x00xx** as an indication that the corresponding request was rejected by **COMMON-ISDN-API**. Class **0x00xx** indicates successful handling of the corresponding request and returns additional information.

Class 0x00xx: Informative values (the corresponding request message was processed)

Value	Reason
0	Request accepted
0x0001	NCPI not supported by current protocol, NCPI ignored
0x0002	Flags not supported by current protocol, flags ignored
0x0003	Alert already sent by another application

Class 0x10xx: Error information concerning CAPI_REGISTER

Value	Reason
0x1001	Too many applications
0x1002	Logical block size too small; must be at least 128 bytes
0x1003	Buffer exceeds 64 kbytes
0x1004	Message buffer size too small, must be at least 1024 bytes
0x1005	Max. number of logical connections not supported
0x1006	reserved
0x1007	The message could not be accepted because of an internal busy condition
0x1008	OS resource error (e.g. no memory)
0x1009	**COMMON-ISDN-API** not installed
0x100A	Controller does not support external equipment

0x100B	Controller does only support external equipment

Class 0x11xx: Error information concerning message exchange functions

Value	Reason
0x1101	Illegal application number
0x1102	Illegal command or subcommand, or message length less than 12 octets
0x1103	The message could not be accepted because of a queue full condition. The error code does not imply that **COMMON-ISDN-API** cannot receive messages directed to another controller, PLCI or NCCI.
0x1104	Queue is empty
0x1105	Queue overflow: a message was lost. This indicates a configuration error. The only recovery from this error is to do the CAPI_RELEASE operation.
0x1106	Unknown notification parameter
0x1107	The message could not be accepted because of an internal busy condition
0x1108	OS resource error (e.g. no memory)
0x1109	**COMMON-ISDN-API** not installed
0x110A	Controller does not support external equipment
0x110B	Controller supports only external equipment

Class 0x20xx: Error information concerning resource/coding problems

Value	Reason
0x2001	Message not supported in current state
0x2002	Illegal Controller/PLCI/NCCI
0x2003	No PLCI available
0x2004	No NCCI available
0x2005	No Listen resources available
0x2006	No fax resources available (protocol T.30)
0x2007	Illegal message parameter coding
0x2008	No interconnection resources available

Class 0x30xx: Error information concerning requested services

Value	Reason
0x3001	B1 protocol not supported
0x3002	B2 protocol not supported
0x3003	B3 protocol not supported
0x3004	B1 protocol parameter not supported
0x3005	B2 protocol parameter not supported
0x3006	B3 protocol parameter not supported
0x3007	B protocol combination not supported
0x3008	NCPI not supported
0x3009	CIP Value unknown
0x300A	Flags not supported (reserved bits)
0x300B	Facility not supported
0x300C	Data length not supported by current protocol
0x300D	Reset procedure not supported by current protocol
0x300E	TEI assignment failed / overlapping channel masks
0x300F	Unsupported interoperability (see Part IV)
0x3010	Request not allowed in this state
0x3011	Facility specific function not supported

This information element appears in:

CONNECT_B3_CONF
CONNECT_CONF
INFO_CONF
DATA_B3_CONF
DISCONNECT_B3_CONF

DISCONNECT_CONF
LISTEN_CONF
RESET_B3_CONF
SELECT_B_PROTOCOL_CONF

Info Element (struct)

The purpose of the parameter *Info element* depends on the value of the parameter *Info number*.

If the *Info number* specifies an information element, then *Info element* contains that information element with coding as defined in ETS 300 102-1 / Q.931.

If the *Info number* specifies the charge information *Charge units*, then *Info element* contains a dword indicating the sum of charging units accumulated by the network up to this moment.

If the *Info number* specifies the charging information *National currency* then *Info element* contains the following struct:

Charges	dword	Sum of charges accumulated by the network up to this moment. **Note**: implementations that conform to **COMMON-ISDN-API** v2.0 [Second Edition] may return only this parameter.
Extended charges	dword	Sum of charges accumulated by the network (see ETS 300 182-1 Table 2: ASN.1 variable 'rCurrency')
Multiplier	word	Extended multiplier (see ETS 300 182-1 Table 2: ASN.1 variable 'multiplier'): 0: 1/1000 1: 1/100 2: 1/10 3: 1 4: 10 5: 100 6: 1000
Currency sign	struct	Currency sign (see ETS 300 182-1 Table 2: ASN.1 variable 'currencyAmount'), coded as IA5-characters

If the *Info number* specifies a message type, then the *Info element* is an empty **COMMON-ISDN-API** struct.

This information element appears in:

INFO_IND

Info Mask (dword)

The parameter *Info mask* specifies which type of information for a physical connection or controller is provided by **COMMON-ISDN-API**. The selected information is indicated in INFO_IND messages to the application. A given *Info mask* (set in LISTEN_REQ) is valid until it is superseded by another LISTEN_REQ, and applies to all information concerning the corresponding application. The *Info mask* is coded as a bit field. A bit set to 1 means that corresponding INFO_IND messages are generated. A bit set to 0 means the specified information is suppressed. In the default *Info mask*, all bits are set to 0. If an application wants to change this value, it must send a LISTEN_REQ message, even if it does not want to be informed about incoming calls.

Bit 0	Cause: cause information given by the network during disconnection. The *info element* parameter of the corresponding INFO_IND message is a **COMMON-ISDN-API** struct which contains the cause information element as defined in ETS 300 102-1 and Q.931 (4.5.12 in both).
Bit 1	Date/time: date/time information indicated by the network. The *info element* parameter of the corresponding INFO_IND message contains the date/time information element as defined in ETS 300 102-1 and Q.931 (4.6.1 in both).
Bit 2	Display: information to be displayed to the user. The *info element* parameter of the corresponding INFO_IND message contains the display information element as defined in ETS 300 102-1 and Q.931 (4.5.15 in both).
Bit 3	User-user: user-user information that is carried transparently by the network. The *info element* parameter of the corresponding INFO_IND message contains the user-user information element as defined in ETS 300 102-1 and Q.931 (4.5.29 in both).
Bit 4	Call progress: information regarding to the progress of the call. There are five different INFO_IND messages that correspond to this information type, each with a unique info number. The first INFO_IND contains the progress indicator information element as defined in ETS 300 102-1 and Q.931. The other four messages indicate the occurrence of the network events SETUP ACKNOWLEDGE, CALL PROCEEDING, ALERTING and PROGRESS. In these cases, the *Info number* parameter indicates the event, and the *Info element* is an empty **COMMON-ISDN-API** struct.
Bit 5	Facility: facility information to indicate the invocation and operation of supplementary services. The *Info element* parameter of the corresponding INFO_IND message contains the facility information element as defined in ETS 300 102-1 and Q.931 (4.6.2 in both).
Bit 6	Charge information: connection-oriented charge information provided by the network. There are two different INFO_IND messages, with unique *Info number* values, that correspond to this information type. The first shows the total charge units indicated by the network up to this moment; the second shows the total charges in the national currency indicated by the network up to this moment. In both cases, the *Info element* parameter is coded as a **COMMON-ISDN-API** struct containing a dword. It is highly recommended that only one of these two types of charge information be supplied to the user, and that the application convert one type to the other. However, in some networks this might be impossible due to ambiguous information provided by the network. In such cases it is not defined whether the current charges are represented by only one or by both types of information, or by the sum of the two.
Bit 7	Called Party Number: identifies the destination of a call. The *info element* parameter of the corresponding INFO_IND message contains the *called party number* information element as defined in ETS 300 102-1 and Q.931 (4.5.8 in both).
Bit 8	Channel Identification: identifies the used channel of a call. The *info element* parameter of the corresponding INFO_IND message contains the *channel identification* information element as defined in ETS 300 102-1 and Q.931 (4.5.13 in both).
Bit 9	Enables ‘early B3 connect’ (see note). When this bit is set to 1, a B-channel connection (NCCI) may be established based on a D-channel connection (PLCI) which has not yet been established. Additional information regarding the progress of the call is sent to the application. There are two different INFO_IND messages that correspond to this information type, each with a unique info number: The first INFO_IND contains the progress indicator information element as defined in ETS 300 102-1 and Q.931. The other indicates the occurrence of the network event DISC. In this case, the *Info number* parameter indicates the corresponding event and the *Info element* is an empty **COMMON-ISDN-API** struct.
Bit 10	Redirecting/redirection information: redirecting/redirection information indicated by the network. The *info element* parameter of the corresponding INFO_IND message contains the *redirecting/redirection number* information element defined in ETS 300 207 (7.2).
Bit 11	reserved: must be set to 0.
Bit 12	Sending Complete: indicates the completion of the called party number. The *info element* parameter of the corresponding INFO_IND message contains the *sending complete* information element as defined in ETS 300 102-1 and Q.931 (4.5.27 in both). The *sending complete* information element may be sent by the network after completion of the called party number; this can be helpful especially for the overlap receiving (direct dial in, DDI) case, where the called party number may be spread over multiple INFO_IND messages containing the *called party number* information element (see bit 7).
Bits 13-31	reserved: must be set to 0.

Note (Early B3 Connect): Voice applications need access to the in-band announcements provided by the local exchange, such as ring tones and “number changed” announcements. This can be realized by establishing a (transparent) NCCI connection before the PLCI state machine has reached the P-ACT state. See also Chapter 7 (state diagram) and Annex A (sample flow chart diagrams).

This information element appears in:

LISTEN_REQ

Info Number (word)

The purpose of the parameter *Info number* specifies the coding of the parameter *Info element* and the type of information which is carried by the given INFO_IND message. The high byte is structured as a bit field and indicates which type of information is contained in the low byte:

Bit 15	1: The low byte identifies a message type. 0: The low byte represents an information element type.
Bit 14	1: The low byte indicates supplementary information not covered by network events or information elements. In this case, bit 15 must be set to 0.
Bits 13-8	Reserved: set to 0.

If bit 15 is set, then the low byte containing the message type is coded in accordance with ETS 300 102-1 / Q.931. In this case, the INFO_IND message indicates the occurrence of a network event corresponding to the specified message, and the parameter *Info element* is an empty **COMMON-ISDN-API** struct.

If bits 14 and 15 are cleared, then the low byte represents an information element type, coded in accordance with ETS 300 102-1 / Q.931. The parameter *info element* contains the information element itself.

If bit 14 is set, then the low byte represents supplementary information. The defined values are

0	**Total charges in charge units.** In this case, the *Info element* parameter contains the information element.
1	**Total charges in national currency.** In this case, the *Info element* parameter contains the information element.

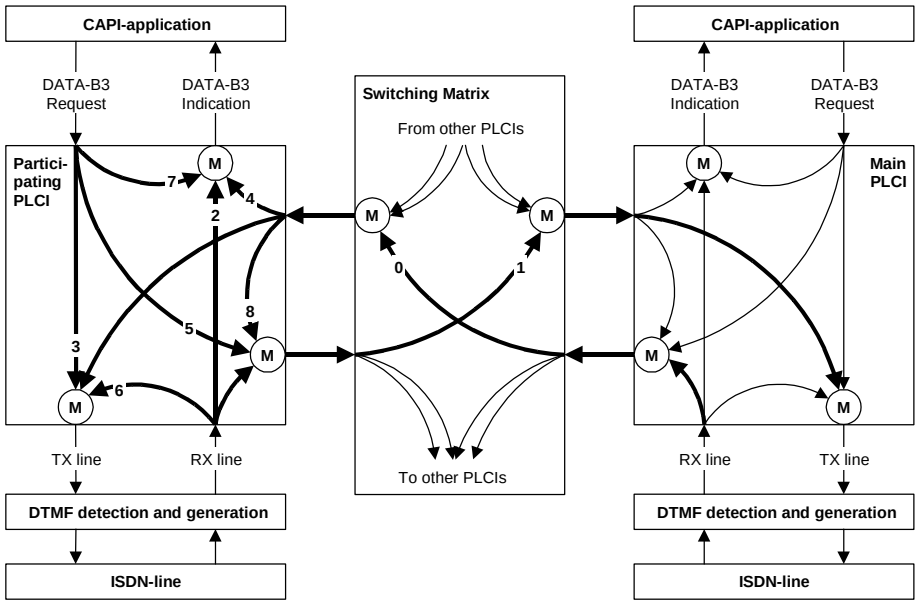
This information element appears in:

INFO_IND

LI Connect Request Participant (struct)

Participating PLCI	dword	Identifier of entity to be interconnected to entity identified by PLCI in main-part of FACILITY_REQ
Data path	dword	see figure below. Bit field, coding as follows: [0]: Enable data-transmission from main PLCI to participating PLCI [1]: Enable data-transmission from participating PLCI to main PLCI [2]: Enable monitoring of channel-data for participating PLCI [3]: Enable mixing for participating PLCI [4]: Enable monitoring of all data which is sent to channel of participating PLCI [5]: Enable mixing of DATA_B3_REQ of participating PLCI to channels of all interconnected PLCI [6]: Incoming line-data will be looped back. [7]: Incoming application-data (DATA_B3_REQ) will be looped back (DATA_B3_IND) [8]: Incoming conference-data will be looped back. [9 to 31]: reserved

Note: If Bit 2 is set, DATA_B3_INDs will be generated for the participating PLCI if it has a layer-3-connection, otherwise DATA_B3_INDs will stop coming in. If Bit 3 is set, all DATA_B3_REQs transferred for participating PCLI will be mixed to all other data sent to the channel of the participating PLCI. If Bit 4 is set, all interconnection data – even of later interconnected entities - which is sent to the channel of the participating PLCI will also be mixed into the DATA_B3_INDs of the participating PLCI. If bit 5 is set, all DATA_B3_REQs which are transferred for the participating PCLI will also be mixed into the channels of all interconnected entities – even if they are interconnected later on.



Model of Line Interconnect facility
Data paths 0..8 denote the connections set through bits [0]..[8] respectively in the *Data Path*
The result of the Mixing operation (M) depends on the B protocol. For speech it enables simultaneous conversation between n parties. Data paths within a PLCI are not subject to clock adaptation. Synchronisation takes place at the transition to/from the switching matrix.

This information element appears in:

LI Request Parameter

LI Connect Confirmation Participant (struct)

Participating PLCI	dword	Identifier of entity to be interconnected to entity identified by PLCI in main-part of FACILITY_REQ
Participating Info	word	0x0000: Request accepted 0x2001: Message not supported in current state 0x2002: Incorrect Controller/PLCI/NCCI 0x2007: Illegal message parameter coding 0x2008: No interconnection resources available 0x3011: Facility specific function not supported

This information element appears in:

LI Confirmation Parameter

LI Disconnect Request Participant (struct)

Participating PLCI	dword	Identifier of entity to be disconnected from entity identified by PLCI in main-part of FACILITY_REQ

This information element appears in:

LI Request Parameter

LI Disconnect Confirmation participant (struct)

Participating PLCI	dword	Identifier of entity to disconnect from entity identified by PLCI in main-part of FACILITY_REQ
Participating Info	word	0x0000: Request accepted 0x2001: Message not supported in current state 0x2002: Incorrect Controller/PLCI/NCCI 0x2007: Illegal message parameter coding 0x3011: Facility specific function not supported

This information element appears in:

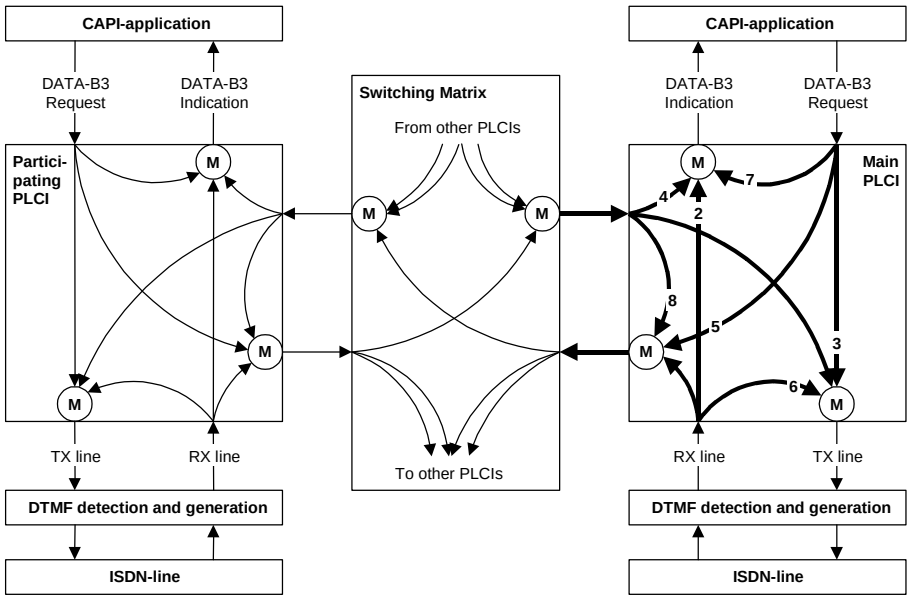
LI Confirmation Parameter

0x0000 Get Supported Services
Parameter does not apply (coded as struct with length 0)

0x0001 Connect

Data Path	dword	See figure below. Bit field, coding as follows: [0]: reserved [1]: reserved [2]: Enable monitoring of channel data for PLCI in main-part of FACILITY_REQ [3]: Enable mixing into data channel of PLCI in main-part of FACILITY_REQ [4]: Enable monitoring of channel data of all PLCIs interconnected to PLCI in main-part of FACILITY_REQ [5]: Enable mixing into data channel of all PLCIs interconnected to PLCI in main-part of FACILITY_REQ [6]: Incoming line-data will be looped back. [7]: Incoming application-data (DATA_B3_REQ) will be looped back (DATA_B3_IND) [8]: Incoming conference-data will be looped back. [9 to 31]: reserved
LI Connect Request Participant	struct	Sequence of participant-structs for the interconnection with the PLCI in main-part of FACILITY_REQ

Note: If Bit 2 is set, DATA_B3_INDs will be generated for the main PLCI if it has a layer-3-connection, otherwise DATA_B3_INDs will stop coming in. If Bit 3 is set, all DATA_B3_REQs transferred for the main PCLI will be mixed to all other data sent to the channel of the main PLCI. If Bit 4 is set, all interconnection data – even of later interconnected entities - which is sent to the channel of the main PLCI will also be mixed into the DATA_B3_INDs of the main PLCI. If bit 5 is set, all DATA_B3_REQs which are transferred for the main PCLI will also be mixed into the channels of all interconnected entities – even if they are interconnected later on. If the two lowest bits of a Participant Interconnect Mask are 0, a Line Interconnect indication “Disconnect” will be generated. In all other bit-combinations a Line Interconnect indication “Connect” will result in case of success. General interconnect-behavior may also change depending on the value of bit 9 of the parameter *Info Mask* in the LISTEN_REQ (early B3).



Model of Line Interconnect facility:
Data paths 2..8 denote the connections set through bits [2]..[8] respectively in the *Data Path.*
The result of the Mixing operation (M) depends on the B protocol. For speech it enables simultaneous conversation between parties. Data paths within a PLCI are not subject to clock adaptation. Synchronisation takes place at the transition to/from switching matrix.

0x0002 Disconnect

LI Disconnect Request Participant	struct	Sequence of participant-structs to be removed from the interconnection to the PLCI in main-part of FACILITY_REQ.

This information element appears in:

Facility Request Parameter

LI Confirmation Parameter (struct)

0x0000 Get Supported Services

Info	word	0x0000: Request accepted 0x2001: Message not supported in current state 0x2002: Incorrect Controller/PLCI/NCCI 0x2007: Illegal message parameter coding 0x3011: Facility specific function not supported
Supported Services	dword	Bit field, coding as follows: [0]: Cross-Controller supported Interconnects across all controller-boundaries (PLCIs of the FACILITY_REQ & of the participating entity/entities can reside on different controllers) [1]: Asymmetric connections supported Different settings for both directions of a link in the Participating Interconnect Mask [2]: Monitoring supported Bit 2 of Participating Interconnect / Main PLCI Data Mask [3]: Mixing supported Bit 3 of Participating Interconnect / Main PLCI Data Mask [4]: Remote Monitoring supported Bit 4 of Participating Interconnect / Main PLCI Data Mask [5]: Remote Mixing supported Bit 5 of Participating Interconnect / Main PLCI Data Mask [6]: Looping of line-data supported Bit 6 of Participating Interconnect / Main PLCI Data Mask [7]: Looping of B3-application-data supported Bit 7 of Participating Interconnect / Main PLCI Data Mask [8]: Looping of conference-data supported Bit 8 of Participating Interconnect / Main PLCI Data Mask [9 to 31]: reserved
Supported Interconnects of this controller	dword	Maximum number of parallel interconnects between any two PLCIs which are supported by the specified controller
Supported Participants of this controller	dword	Maximum number of participants which can be connected to one main PLCIs by the specified controller not including the main-part
Supported Interconnects of all controllers	dword	Maximum number of parallel interconnects between any two PLCIs which are supported accumulated for all available controllers
Supported Participants of all controllers	dword	Maximum number of participants which can be connected to one main PLCIs accumulated for all available controllers not including the main-part.

0x0001 Connect

Main Info	word	0x0000: Request accepted 0x2001: Message not supported in current state 0x2002: Incorrect Controller/PLCI/NCCI 0x2007: Illegal message parameter coding 0x2008: No interconnection resources available 0x3011: Facility specific function not supported
LI Connect Confirmation Participant	struct	Sequence with structs of participant(s) and their corresponding info-values, for the interconnection with the PLCI in main-part of FACILITY_CONF.

0x0002 Disconnect

Main Info	word	0x0000: Request accepted 0x2001: Message not supported in current state 0x2002: Incorrect Controller/PLCI/NCCI 0x2007: Illegal message parameter coding 0x3011: Facility specific function not supported
LI Disconnect Confirmation Participant	struct	Sequence with structs of participant(s) and their corresponding info-values, to be removed from the interconnection to the PLCI in main-part of FACILITY_CONF.

This information element appears in:

Facility Confirmation Parameter

LI Indication Parameter (struct)

0x0001 Connect_Active

Participating PLCI	dword	Identifier of entity interconnected to entity identified by PLCI in main-part of FACILITY_IND

0x0002 Disconnect

Participating PLCI	dword	Identifier of entity disconnected from entity identified by PLCI in main-part of FACILITY_IND.
LI Service Reason	word	0x0000: User initiated 0x3800: PLCI has no b-channel 0x3801: Lines not compatible 0x3802: PLCI(s) is (are) not in any or not in the same interconnection

Note: The participant structs are not available for indications, as the events that trigger these indications will not happen at the same time. The participant structs are only a syntactical tool for ease of use when starting & terminating interconnects. One might as well send multiple requests instead of one with all participants included. In the disconnect-case the main PLCI will not be the instance which caused this message either by a FACILITY-request or due to the corresponding line disconnecting for example but the PLCI of the instance which is affected by this event.

This information element appears in:

Facility Indication Parameter

LI Service Reason (word)

The purpose of the parameter *Line Interconnection Service Reason* is to provide error information to the application regarding establishment of line interconnections. The defined values are:

Value	Reason
0x0000	User initiated
0x3800	PLCI has no B-channel
0x3801	Lines not compatible
0x3802	PLCI(s) is (are) not in any or not in the same interconnection

This information element appears in:

LI Indication Parameter

LLC (struct)

The purpose of the parameter *Low Layer Compatibility (LLC)* is to provide a means for compatibility checking by an addressed entity (such as a remote user, an interworking unit or a network node with a high layer function addressed by the calling party). The *Low Layer Compatibility* information element is transferred transparently by ISDN between the entity originating the call (e.g. the calling user) and the addressed entity. If the network allows *Low Layer Compatibility* negotiation, then the *Low Layer Compatibility* information element is also passed transparently from the addressed entity to the originating entity. The information element is coded in accordance with ETS 300 102-1 / Q.931.

This information element appears in:

CONNECT_ACTIVE_IND
CONNECT_IND
CONNECT_REQ
CONNECT_RESP

Manu ID (dword)

The purpose of the parameter *Manu ID* is to communicate a dword which identifies the manufacturer in MANUFACTURER messages. Every manufacturer supplying MANUFACTURER messages should choose a unique value (such as an abbreviation of the company name).

This information element appears in:

MANUFACTURER_REQ
MANUFACTURER_RESP
MANUFACTURER_IND
MANUFACTURER_CONF

Manufacturer-Specific

The purpose of the parameter *Manufacturer-specific* is to exchange manufacturer-specific information.

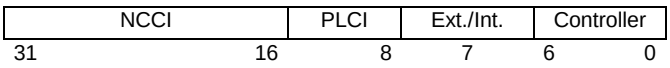
This information element appears in:

MANUFACTURER_REQ
MANUFACTURER_RESP
MANUFACTURER_IND
MANUFACTURER_CONF

NCCI (dword)

The purpose of the parameter *NCCI* is to identify a logical connection. The NCCI (Network Control Connection Identifier) is assigned by **COMMON-ISDN-API** on creation of a logical connection. Depending on the Layer 3 protocol selected (e.g. ISO 8208), it is possible to have multiple NCCIs based on a single PLCI. The *NCCI* parameter is a dword with a range from 1 to 65535 (0 reserved), coded as described below, and also includes the corresponding PLCI and controller number.

Format of *NCCI*:



This information element appears in:

CONNECT_B3_ACTIVE_IND
CONNECT_B3_ACTIVE_RESP
CONNECT_B3_CONF
CONNECT_B3_IND
CONNECT_B3_RESP
DATA_B3_CONF
DATA_B3_IND
DATA_B3_REQ
DATA_B3_RESP
DISCONNECT_B3_CONF
DISCONNECT_B3_IND
DISCONNECT_B3_REQ
DISCONNECT_B3_RESP
FACILITY_REQ
FACILITY_CONF
FACILITY_IND
FACILITY_RESP
RESET_B3_CONF
RESET_B3_IND
RESET_B3_REQ
RESET_B3_RESP

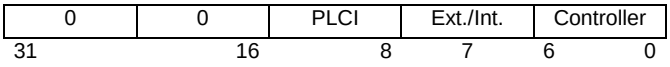
NCPI (struct)

The purpose of the parameter *NCPI* is to provide additional protocol-specific information. The parameter *NCPI* is protocol-dependent, see Subclause 6.2.

PLCI (dword)

The purpose of the parameter *PLCI* is to identify a physical connection between two endpoints. The PLCI (Physical Link Connection Identifier) is assigned by **COMMON-ISDN-API** during creation of the physical connection. The *PLCI* parameter is a dword with the range from 1 to 255 (0 reserved), coded as described below, and also includes the controller number.

Format of *PLCI*:



This information element appears in:

CONNECT_ACTIVE_IND
CONNECT_ACTIVE_RESP
CONNECT_B3_REQ
CONNECT_CONF
CONNECT_IND
CONNECT_RESP
DISCONNECT_REQ
DISCONNECT_CONF
DISCONNECT_IND
DISCONNECT_RESP
FACILITY_REQ
FACILITY_CONF
FACILITY_IND

FACILITY_RESP
INFO_REQ
INFO_CONF
INFO_IND
INFO_RESP
SELECT_B_PROTOCOL_REQ
SELECT_B_PROTOCOL_CONF

Reason (word)

The purpose of the parameter *Reason* is to provide error information to the application regarding the clearing down of a physical connection. The defined values are:

0	**Normal clearing, no cause available**
0x3301	**Protocol error, Layer 1**
0x3302	**Protocol error, Layer 2**
0x3303	**Protocol error, Layer 3**
0x3304	**The call was given to another application (see LISTEN_REQ)**
0x3305	**Cleared by Call Control Supervision (see Annex D.2)**
0x34xx	**Disconnect cause from the network in accordance with ETS 300 102-1 / Q.850. The field 'xx' indicates the cause value received from the network in a cause information element (Octet 4).**

This information element appears in:

DISCONNECT_IND

Reason_B3 (word)

The purpose of the parameter is to provide error information to the application regarding the clearing down of a logical connection. The parameter *Reason_B3* is protocol-dependent: see Subclause 6.2. The defined protocol-independent values are:

0	**Normal clearing, no cause available**
0x3301	**Protocol error Layer 1 (line interrupted)**
0x3302	**Protocol error Layer 2**
0x3303	**Protocol error Layer 3**
0x3305	**Cleared by Call Control Supervision (see Annex D.2)**

This information element appears in:

DISCONNECT_B3_IND

Reject (word)

The purpose of the parameter *reject* is to define the action of **COMMON-ISDN-API** for incoming calls.

The defined values are

0	**Accept the call**
1	**Ignore the call**
2	**Reject call, normal call clearing**
3	**Reject call, user busy**
4	**Reject call, requested circuit/channel not available**
5	**Reject call, facility rejected**
6	**Reject call, channel unacceptable**
7	**Reject call, incompatible destination**
8	**Reject call, destination out of order**

0x34xx	**The content of the low byte 'xx' will be signaled to the network in a cause information element (Octet 4).** It is the application's responsibility to provide a value that is properly coded in accordance with Q.931/ETS 300 102-1. The controller will send this cause value indicating coding standard CCITT (Octet 3).

This information element appears in:

CONNECT_B3_RESP
CONNECT_RESP

Sending Complete (struct)

The purpose of the sub-parameter *Sending Complete* is to enable the origination of a Sending Complete Information Element at the completion of the Called Party Number by means of a CONNECT_REQ or an INFO_REQ. If this struct is empty, the default value is assumed.

Mode	word	0: Do not send the Sending Complete Information Element (default value) 1: Send the Sending Complete Information Element

This sub-parameter appears in the parameter:

Additional information

6.2 Protocol-Dependent Parameters

B Channel Operation (word)

The purpose of the sub-parameter *B Channel Operation* is to specify the mode (DTE or DCE) in which the B channel protocols are operated, regardless of which end initiated the call establishment.

The following values are defined:

0:	**Default (see note)**
1:	**DTE mode (originate)**
2:	**DCE mode (answer)**

Note (Default mode): On the calling side (CONNECT_REQ), the B channel runs in the DTE mode (CONNECT_B3_REQ). On the called side (CONNECT_IND) the B channel runs in the DCE mode (CONNECT_B3_IND).

This sub-parameter appears in parameter:

Global Configuration

B Protocol (struct)

The purpose of the parameter *B protocol* is to select and configure the B channel protocols. The parameter contains a protocol identifier and configuration information for each layer. If this struct is empty, the default value is assumed.

The parameter has the following structure:

word	**B1 protocol: Physical layer and framing**
word	**B2 protocol: Data link layer**
word	**B3 protocol: Network layer**
struct	**B1 configuration: Physical layer and framing parameter**
struct	**B2 configuration: Data link layer parameter**
struct	**B3 configuration: Network layer parameter**
struct	**Global Configuration**

This information element appears in:

CONNECT_REQ
CONNECT_RESP
SELECT_B_PROTOCOL_REQ

B1 Protocol (word)

The purpose of the sub-parameter *B1 protocol* is to specify the physical layer and framing used for this connection.

The following values are defined:

0:	**64 kbit/s with HDLC framing (default)**
1:	**64 kbit/s bit-transparent operation with byte framing from the network**
2:	**V.110 asynchronous operation with start/stop byte framing (see Note 1)**
3:	**V.110 synchronous operation with HDLC framing (see Note 2)**
4:	**T.30 modem for Group 3 fax**
5:	**64 kbit/s inverted with HDLC framing**
6:	**56 kbit/s bit-transparent operation with byte framing from the network**
7:	**Modem with full negotiation (B2 Protocol must be 7)**
8:	**Modem asynchronous operation with start/stop byte framing**
9:	**Modem synchronous operation with HDLC framing**

Note 1: Recommendation V.110 describes two different frame alternatives for 56 kbit/s rate adaptation. **COMMON-ISDN-API** uses frame alternative 1 (V.110 table 7b).

Note 2: For transmission of HDLC framing in 56 kbit/s networks, **COMMON-ISDN-API** B1 Protocol 3 (V.110 synchronous operation with HDLC framing) shall be used.

This sub-parameter appears in parameter:

B protocol

B2 Protocol (word)

The purpose of the sub-parameter *B2 protocol* is to specify the data link layer used for this connection.

The following values are defined:

0:	**ISO 7776 (X.75 SLP) (default)**
1:	**Transparent**
2:	**SDLC**
3:	**LAPD in accordance with Q.921 for D channel X.25 (SAPI 16)**
4:	**T.30 for Group 3 fax**
5:	**Point-to-Point Protocol (PPP)**
6:	**Transparent (ignoring framing errors of B1 protocol)**
7:	**Modem with full negotiation (e.g. V.42 bis, MNP 5)**
8:	**ISO 7776 (X.75 SLP) modified to support V.42 bis compression (see Note 3)**
9:	**V.120 asynchronous mode (see Note 1)**
10:	**V.120 asynchronous mode with V.42 bis compression (see Notes 1, 2, 3)**
11:	**V.120 bit-transparent mode (see Note 1)**
12:	**LAPD in accordance with Q.921 including free SAPI selection**

Note 1:
V.120 multiframe mode supported (data transmission uses I-frames, not UI-frames).
V.120 flow control by Q.921 mechanism supported (RR/RNR, usage of V.120 CS-header byte is implementation-dependent).
V.120 break signal /error handling is indicated.
V.120 Multi-Link operation is not supported.
V.120 in-band negotiation is not supported.

Note 2: **COMMON-ISDN-API** negotiates V.42 bis compression using the defined XID exchange mechanism and obtains the maximum V.120 frame size.

Note 3: Implementation clarifications for **COMMON-ISDN-API's** V.42 bis implementation are described in Annex D.1.

This sub-parameter appears in parameter:

B protocol

B3 Protocol (word)

The purpose of the sub-parameter *B3 protocol* is to specify the network layer used for this connection.

The following values are defined:

0:	**Transparent (default)**
1:	**T.90NL with compatibility to T.70NL in accordance with T.90 Appendix II.**
2:	**ISO 8208 (X.25 DTE-DTE)**
3:	**X.25 DCE**
4:	**T.30 for Group 3 fax**
5:	**T.30 for Group 3 fax extended (see Note 1)**
6:	**reserved**
7:	**Modem (see Note 2)**

Note 1: Includes support for fax-polling mode and detailed status information (see parameter *NCPI*).

Note 2: Modem capability is also possible with B3 Protocol 0 (Transparent), but applications must use B3 Protocol 7 to obtain information about the results of modem negotiation (see parameter *NCPI*).

This sub-parameter appears in parameter:

B protocol

B1 Configuration (struct)

The purpose of the sub-parameter *B1 configuration* is to provide additional configuration information for the B1 protocol.

The coding of the parameter *B1 configuration* for each protocol is described below:

B1 Configuration for B1 protocol 0: 64 kbit/s with HDLC framing

Coded as an empty struct

B1 Configuration for B1 protocol 1: 64 kbit/s bit-transparent operation:

Coded as an empty struct

B1 Configuration for B1 protocol 2: V.110 asynchronous operation with start/stop byte framing:

Maximum bit rate	word	Coded as unsigned integer value (default: 0 : adaptive)
Bits per character	word	Coded as unsigned integer value (default: 8)
Parity	word	0: None (default) 1: Odd 2: Even
Stop bits	word	0: 1 stop bit (default) 1: 2 stop bits

B1 Configuration for B1 protocol 3: V.110 synchronous operation with HDLC framing:

Maximum bit rate	word	Coded as unsigned integer value, default: 56 kbit
Bits per character	word	reserved, coded as 0
Parity	word	reserved, coded as 0
Stop bits	word	reserved, coded as 0

B1 Configuration for B1 protocol 4: T.30 modem for Group 3 fax:

Maximum bit rate	word	Coded as unsigned integer value (default: 0 : adaptive)
Transmit level in dB	word	Coded as signed integer value. If this parameter or its value is not supported by the ISDN controller, it is ignored.
reserved1	word	reserved, coded as 0
reserved2	word	reserved, coded as 0

B1 Configuration for B1 protocol 5: 64 kbit/s inverted with HDLC framing:

Coded as an empty struct

B1 Configuration for B1 protocol 6: 56 kbit/s bit-transparent operation:

Coded as an empty struct

B1 Configuration for B1 protocol 7: Modem with full negotiation:

Maximum bit rate	word	Coded as unsigned integer value (default: 0 : adaptive)
Bits per character	word	Coded as unsigned integer value, default: 8
Parity	word	0: None (default) 1: Odd 2: Even
Stop bits	word	0: 1 stop bit (default) 1: 2 stop bits

Options	word	[Bit 0]: Disable retrain [Bit 1]: Disable ring tone [Bits 3...2]: Guard tone: 00: No guard tone (default) 01:1800 Hz 10: 550 Hz [Bits 5...4]: loudspeaker 00: Off 01: On during dialing and negotiation (default) 10: Always on [Bits 7...6]: Loudspeaker volume 00: Silent 01: Normal low (default) 10: Normal high 11: Maximum
Speed negotiation	word	0: None 1: Within modulation class 2: V.100 3: V.8 (default) Note: The highest implemented negotiation mode is used as default.

B1 Configuration for B1 protocol 8: Modem asynchronous operation with start/stop byte framing:

maximum bit rate	word	Coded as unsigned integer value (default: 0 : adaptive)
bits per character	word	Coded as unsigned integer value, default: 8
Parity	word	0: None (default) 1: Odd 2: Even
Stop bits	word	0: 1 stop bit (default) 1: 2 stop bits
options	word	[Bit 0]: Disable retrain [Bit 1]: Disable ring tone [Bits 3...2]: Guard tone: 00: No guard tone (default) 01:1800hz 10: 550hz [Bits 5...4]: Loudspeaker 00: Off 01: On during dialing and negotiation (default) 10: Always on [Bits 7...6]: Loudspeaker volume 00: Silent 01: Normal low (default) 10: Normal high 11: Maximum
Speed negotiation	word	0: None 1: Within modulation class 2: V.100 3: V.8 (default) Note: The highest implemented negotiation mode is used as default.

B1 Configuration for B1 protocol 9: Modem synchronous operation with HDLC framing:

Maximum bit rate	word	Coded as unsigned integer value (default: 0 : adaptive)
Bits per character	word	reserved, coded as 0
Parity	word	reserved, coded as 0
Stop bits	word	reserved, coded as 0

Options	word	[Bit 0]: Disable retrain [Bit 1]: Disable ring tone [Bit 3...2]: Guard tone: 00: No guard tone (default) 01:1800 Hz 10: 550 Hz [Bits 5...4]: Loudspeaker 00: Off 01: On during dialing and negotiation (default) 10: Always on [Bits 7...6]: Loudspeaker volume 00: Silent 01: Normal low (default) 10: Normal high 11: Maximum
Speed negotiation	word	0: None 1: Within modulation class 2: V.100 3: V.8 (default) Note: The highest implemented negotiation mode is used as default.

This sub-parameter appears in parameter:

B protocol

B2 Configuration (struct)

The purpose of the sub-parameter *B2 configuration* is to provide additional configuration information for the B2 protocol.

The coding of the parameter *B2 configuration* for each protocol is described below:

B2 configuration for B2 protocol 0: ISO 7776 (X.75 SLP):

Address A	byte	Link address A, default is 0x03
Address B	byte	Link address B, default is 0x01
Modulo mode	byte	8: Normal operation (default) 128: Extended operation
Window size	byte	Window size (default: 7)
XID	struct	reserved, coded as an empty struct

B2 Configuration for B2 protocol 1: Transparent:

Coded as an empty struct

B2 Configuration for B2 protocol 2: SDLC:

Address A	byte	Link address (default is 0xC1)
Address B	byte	reserved, coded as 0
Modulo mode	byte	8: Normal operation (default) 128: Extended operation
Window size	byte	Window size (default: 7)
XID	struct	Contents of the XID response when a XID command is received

B2 Configuration for B2 protocol 3: LAPD in accordance with Q.921 for D channel X.25 (SAPI 16):

Address A	byte	[Bit 0]: 0: Automatic TEI assignment procedure shall be used 1: Fixed TEI value shall be used (default) [Bits 1...7]: TEI value in case of fixed TEI (default: 0)
Address B	byte	Reserved, coded as 0
Modulo mode	byte	8: Normal operation 128: Extended operation (default)
Window size	byte	Window size (default: 3)
XID	struct	Reserved, coded as an empty struct

B2 Configuration for B2 protocol 4: T.30 for Group 3 fax:

Coded as an empty struct

B2 Configuration for B2 protocol 5: Point-to- Point Protocol (PPP):

Coded as an empty struct

B2 Configuration for B2protocol 6: Transparent (ignoring framing errors of B1 protocol):

Coded as an empty struct

B2 Configuration for B2 protocol 7: Modem with full negotiation:

Options	word	[Bit 0]: Disable V.42 / V.42 bis [Bit 1]: Disable MNP4/MNP5 [Bit 2]: Disable transparent mode (accept only V.42 / V.42 bis or MNP4/5 connects) [Bit 3]: Disable V.42 negotiation [Bit 4]: Disable compression [other]: reserved

B2 Configuration for B2 protocol 8: ISO 7776 (X.75 SLP) modified supporting V.42 bis compression:

Address A	byte	Link address A, default is 0x03
Address B	byte	Link address B, default is 0x01
Modulo mode	byte	8: Normal operation (default) 128: Extended operation
Window size	byte	Window size (default: 7)
Direction	word	Enable compression/decompression for 0: All directions (default) 1: Incoming data only 2: Outgoing data only
Number of code words	word	Parameter *P1* of V.42 bis (default: manufacturer-dependent). In accordance with V.42 bis, a value of 2048 provides good compression across a wide range of data types. Implementation of a default value of at least 2048 is therefore suggested.
Maximum string length	word	Parameter *P2* of V.42 bis, value in range from 6 to 250 (default: 250)

B2 Configuration for B2 protocol 9: V.120 asynchronous mode:

Address A	byte	Low byte of LLI, default is 0x00
Address B	byte	High byte of LLI, default is 0x01
Modulo mode	byte	128: extended operation (default)
Window size	byte	Window size (default 7)
XID	struct	reserved, coded as an empty struct

B2 Configuration for B2 protocol 10: V.120 asynchronous mode with V.42 bis compression:

Address A	byte	Low byte of LLI, default ix 0x00
Address B	byte	High byte of LLI, default is 0x01
Modulo mode	byte	128: Extended operation (default)
Window size	byte	Window size (default: 7)
Direction	word	Enable compression/decompression for 0: All directions (default) 1: Incoming data only 2: Outgoing data only
Number of code words	word	Parameter *P1* of V.42 bis (default: manufacturer-dependent). In accordance with V.42 bis, a value of 2048 provides good compression across a wide range of data types. Implementation of a default value of at least 2048 is therefore suggested.
Maximum string length	word	Parameter *P2* of V.42 bis, value in range from 6 to 250, default is 250

B2 Configuration for B2 protocol 11: V.120 bit-transparent mode:

Address A	byte	Low byte of LLI, default is 0x00
Address B	byte	High byte of LLI, default is 0x01
Modulo mode	byte	128: extended operation (default)
Window size	byte	Window size (default: 7)
XID	struct	reserved, coded as an empty struct

B2 Configuration for B2 protocol 12: LAPD in accordance with Q.921 including free SAPI selection:

Address A	byte	[Bit 0]: 0: Automatic TEI assignment procedure shall be used 1 : Fixed TEI value shall be used (default) [Bits 1.. 7]: TEI value in case of fixed TEI (default: 0)
Address B	byte	[Bits 0..1]: reserved [Bits 2..7]: SAPI (default: 0)
Modulo mode	byte	8: normal operation 128: extended operation (default)
Window size	byte	Window size (default: 1)
XID	struct	reserved, coded as an empty struct

This sub-parameter appears in parameter:

B protocol

B3 Configuration (struct)

The purpose of the sub-parameter *B3 configuration* is to provide additional configuration information for the B3 protocol. Different structures of this parameter are defined, depending on the B3 protocol:

B3 Configuration for B3 protocol 0: Transparent

Coded as an empty struct

B3 Configuration for B3 protocol 1: T.90NL with compatibility to T.70NL in accordance with T.90 Appendix II:

LIC	word	Lowest incoming channel, default is 0
HIC	word	Highest incoming channel, default is 0
LTC	word	Lowest two-way channel, default is 1
HTC	word	Highest two-way channel, default is 1
LOC	word	Lowest outgoing channel, default is 0
HOC	word	Highest outgoing channel, default is 0
Modulo mode	word	8: Normal operation (default) 128: Extended operation
Window size	word	Used to configure non-standard defaults for the transmit and receive window size; default is 2

The default values of maximum transmit and receive packet size are taken from the CAPI_REGISTER parameters.

B3 Configuration for B3 protocol 2: ISO 8208 (X.25 DTE-DTE):

LIC	word	Lowest incoming channel, default is 0
HIC	word	Highest incoming channel, default is 0
LTC	word	Lowest two-way channel, default is 1
HTC	word	Highest two-way channel, default is 1
LOC	word	Lowest outgoing channel, default is 0
HOC	word	Highest outgoing channel, default is 0
Modulo mode	word	8: Normal operation (default) 128: Extended operation
Window size	word	Used to configure non-standard defaults for the transmit and receive window size, default is 2

The default values of maximum transmit and receive packet size are taken from the CAPI_REGISTER parameters.

B3 Configuration for B3 protocol 3: X.25 DCE:

LIC	word	Lowest incoming channel, default is 0
HIC	word	Highest incoming channel, default is 0
LTC	word	Lowest two-way channel, default is 1
HTC	word	Highest two-way channel, default is 1
LOC	word	Lowest outgoing channel, default is 0
HOC	word	Highest outgoing channel, default is 0
Modulo mode	word	8: Normal operation (default) 128: Extended operation

Window size	word	Used to configure non-standard defaults for the transmit and receive window size, default is 2

The default values of maximum transmit and receive packet size are taken from the CAPI_REGISTER parameters.

B3 Configuration for B3 protocol 4: T.30 for Group 3 fax:

Resolution	word	0: Standard 1: High
Format	word	0: SFF (default, description in Annex B) 1: Plain fax format (modified Huffman coding) 2: PCX 3: DCX 4: TIFF 5: ASCII 6: Extended ANSI 7: Binary-File transfer
Station ID	struct	ID of the calling station, max. 20 IA5-characters. To be compatible with T.30 an application should use only the characters defined in T.30 Table 3 (<space>, '+', '0'..'9').
Head line	struct	Header sent on each fax page, coded in accordance with ISO 8859-1 (extended ASCII)

The headline which is generated at the top of each fax page sent should contain the following fields:

Date: **Current date**
Time: **Current time at start of page transmission**
Headline: **Contents of parameter head line**
Station ID: **Contents of parameter station ID (max. 20 characters)**
Page: **Current page number**
Number of pages: **Optional; total number of pages if specified in SFF document header**

The font, order and position of these fields within the complete headline is implementation-dependent.

B3 Configuration for B3 protocol 5: T.30 for Group 3 fax extended:

Options	word	[Bit 0] : Enable high resolution [Bit 1] : Accept incoming fax-polling requests [Bit 10]: Enable JPEG negotiation (continuous-tone color mode according to T.4 Annex E) (see note 1) [Bit 11]: Enable JBIG color and gray-scale negotiation according to T.43 (see note 1) [Bit 12]: Do not use JBIG progressive bi-level image compression [Bit 13]: Do not use MR compression [Bit 14]: Do not use MMR compression [Bit 15]: Do not use ECM
Format	word	Default data format, if not negotiated (see note 1) 0: SFF (default, description in Annex B) 1: Plain fax format (modified Huffman coding) 2: PCX 3: DCX 4: TIFF 5: ASCII 6: Extended ANSI 7: Binary-File transfer
Station ID	struct	ID of the calling station, max. 20 IA5-characters. To be compatible with T.30 an application should use only the characters defined in T.30 Table 3 (<space>, '+', '0'.. '9').
Head line	struct	Header sent on each fax page, coded in accordance with ISO 8859-1 (extended ASCII)

Note 1: If negotiation is successful, the data format is changed to *native*, see parameter NCPI.

The headline which is generated at the top of each fax page sent should contain the following fields:

Date:	**Current date**
Time:	**Current time at start of page transmission**
Headline:	**Contents of parameter head line**
Station ID:	**Contents of parameter station ID (max. 20 characters)**
Page:	**Current page number**
Number of pages:	**Optional; total number of pages if specified in SFF document header**

The font, order and position of these fields within the complete headline is implementation-dependent.

B3 Configuration for B3 protocol 7: Modem:

Coded as an empty struct

This sub-parameter appears in parameter:

B protocol

Global Configuration (struct)

The purpose of the sub-parameter *Global Configuration* is to provide additional configuration information for all protocol layers.

The parameter has the following structure:

word **B Channel Operation**

This information element appears in:

B Protocol

NCPI (struct)

The purpose of the parameter *NCPI* is to provide additional protocol-specific information.

NCPI for B3 protocol 0: Transparent:

Coded as an empty struct

NCPI for B3 protocol 1: T.90NL with T.70NL compatibility in accordance with T.90 Appendix II:

Coded as an empty struct

NCPI for B3 protocol 2: ISO 8208 (X.25 DTE-DTE):

Flags	byte	**[0]:** Enable the use of the delivery confirmation procedure in call set-up and data packets (D-bit) **[1..7]:** reserved
Group	byte	Logical channel group number of the permanent virtual circuit (PVC) to be used. In the case of virtual calls (VC), this number must be set to zero.
Channel	byte	Logical channel number of the permanent virtual circuit (PVC) to be used. In the case of virtual calls (VC), this number must be set to zero.
Contents	byte array	Bytes following the packet type identifier field in the X.25 PLP packets.

NCPI for B3 protocol 3: X25 (DCE)

Options	byte	**[0]:** Enable the use of the delivery confirmation procedure in call set-up and data packets (D-bit) **[1...7]:** Reserved
Group	byte	Logical channel group number of the permanent virtual circuit (PVC) to be used. In the case of virtual calls (VC), this number must be set to zero.
Channel	byte	Logical channel number of the permanent virtual circuit (PVC) to be used. In the case of virtual calls (VC), this number must be set to zero.
Contents	byte array	Bytes following the packet type identifier field in the X.25 PLP packets.

NCPI for B3 protocol 4: T.30 for Group 3 fax (message DISCONNECT_B3_IND):

Rate	word	Actual bit rate used, coded as unsigned integer value

Resolution	word	0: Standard 1: High
Format	word	0: SFF (default) 1: Plain fax format (modified Huffman coding) 2: PCX 3: DCX 4: TIFF 5: ASCII 6: Extended ANSI 7: Binary file transfer
Pages	word	Number of pages, coded as unsigned integer value
Receive ID	struct	ID of remote station

NCPI for B3 protocol 4: T.30 for Group 3 fax (all messages except DISCONNECT_B3_IND):

Coded as an empty struct

NCPI for B3 protocol 5: T.30 for Group 3 fax extended (message CONNECT_B3_REQ):

Rate	word	reserved, coded as 0
Options	word	[Bit 0]: reserved (already set in B3Configuration) [Bit 1]: Fax-polling request [Bit 2]: Request to send / poll another document after the current document (prevents remote station from disconnecting the D channel)
Format	word	ignored (already set in B3Configuration)
Pages	word	reserved, coded as 0
Receive ID	struct	reserved, coded as an empty struct

NCPI for B3 protocol 5: T.30 for Group 3 fax extended (messages CONNECT_B3_IND, CONNECT_B3_ACTIVE_IND, DISCONNECT_B3_IND):

Rate	word	Actual bit rate used, coded as unsigned integer value • CONNECT_B3_IND: bit rate as coded in *B3 Configuration* • CONNECT_B3_ACTIVE_IND: bit rate currently used • DISCONNECT_B3_IND: last bit rate used
Options	word	[Bit 0]: Enable high resolution [Bit 1]: Fax-polling request / indication [Bit 2]: Request / indication to send / poll another document after the current document (prevents remote station from disconnecting the D channel) [Bit 10]: This is a JPEG continuous-tone colour connection with data format according to T.4 Annex E (see note) [Bit 11]: This is a JBIG colour and gray-scale connection with data format according to T.43 Table 7 (see note) [Bit 12]: This is a connection with JBIG progressive bi-level image compression [Bit 13]: This is a connection with MR compression [Bit 14]: This is a connection with MMR compression [Bit 15]: This is not an ECM connection
Format	word	0: SFF (default) 1: Plain fax format (modified Huffman coding) 2: PCX 3: DCX 4: TIFF 5: ASCII 6: Extended ANSI 7: Binary file transfer 8: Native format (see note)
Pages	word	Number of pages, coded as unsigned integer value • CONNECT_B3_IND: reserved, coded as 0 • CONNECT_B3_ACTIVE_IND: reserved, coded as 0 • DISCONNECT_B3_IND: number of pages
Receive ID	struct	ID of remote station • CONNECT_B3_IND: ID of remote station • CONNECT_B3_ACTIVE_IND: ID of remote station • DISCONNECT_B3_IND: ID of remote station

Note: In case of Format 8 (*Native*) the data format is indicated by the the parameter *Option*.

NCPI for B3 protocol 7: Modem (messages CONNECT_B3_ACTIVE_IND, DISCONNECT_B3_IND):

Rate	word	Actual bit rate used, coded as unsigned integer value. If receive and transmit rates are different the lower rate is displayed.
Protocol	Word	Result of negotiation [Bit 0]: V.42 / V.42 bis successfully negotiated [Bit 1]: MNP4/MNP5 successfully negotiated [Bit 2]: Transparent mode successfully negotiated [Bit 3]: reserved [Bit 4]: Compression successfully negotiated [other]: reserved

NCPI for B3 protocol 7: Modem (messages except CONNECT_B3_ACTIVE_IND, DISCONNECT_B3_IND):

Coded as an empty struct

This information element appears in:

CONNECT_B3_ACTIVE_IND
CONNECT_B3_T90_ACTIVE_IND
CONNECT_B3_IND
CONNECT_B3_REQ
CONNECT_B3_RESP
DISCONNECT_B3_IND
DISCONNECT_B3_REQ
RESET_B3_REQ
RESET_B3_RESP

Reason_B3 (word)

The purpose of the parameter *Reason_B3* is to provide error information to the application regarding the clearing of a logical connection. The defined values are:

Protocol independent:

0	Normal clearing, no cause available
0x3301	Protocol error, Layer 1 (interrupted line or B channel removed by signaling protocol)
0x3302	Protocol error, Layer 2
0x3303	Protocol error, Layer 3

B3 protocols 4, 5 (T.30 / T.30 extended)

0x3311	Connection not successful (remote station is not a G3 fax device)
0x3312	Connection not successful (training error)
0x3313	Disconnected before transfer (remote station does not support transfer mode, such as resolution, or fax-polling server does not send a document)
0x3314	Disconnected during transfer (remote abort)
0x3315	Disconnected during transfer (remote procedure error (e.g. unsuccessful repetition of T.30 commands)
0x3316	Disconnected during transfer (local Tx data underflow)
0x3317	Disconnected during transfer (local Rx data overflow)
0x3318	Disconnected during transfer (local abort)
0x3319	Illegal parameter coding (e.g. SFF coding error)

B3 protocol 7 (Modem):

0x3500	Normal end of connection
0x3501	Carrier lost
0x3502	Error in negotiation, i.e. no modem with error correction at the other end
0x3503	No answer to protocol request
0x3504	Remote modem only works in synchronous mode
0x3505	Framing fails
0x3506	Protocol negotiation fails
0x3507	Other modem sends wrong protocol request
0x3508	Sync information (data or flags) missing
0x3509	Normal end of connection from the other modem
0x350a	No answer from other modem
0x350b	Protocol error
0x350c	Error in compression
0x350d	No connect (timeout or a wrong modulation)
0x350e	No protocol fall-back allowed
0x350f	No modem or fax at requested number
0x3510	Handshake error

This information element appears in:

DISCONNECT_B3_IND

7 STATE DIAGRAMS

7.1 User's Guide

To explain the message exchange between CAPI and the application, a graphic description is in order. In the absence of an international standard for the description of message exchange between two local entities, a new sort of representation was created. On the following pages, the state machines are described in the form of a state diagram depicting application and controller. Such a state diagram is a monitor view of an idealized interface. In reality, CAPI is not only an interface definition but also a concrete instantiation.

The state diagram on the following pages is split into three separate state machines:

1. LISTEN state machine
2. PLCI state machine (physical connections)
3. NCCI state machine (logical connections)

On every physical connection, identified by a PLCI, several logical Layer 3 links could exist, each identified by a NCCI. This necessitates a division into PLCI and NCCI state machines. A description of *n* physical links with *m* logical links at one time in one state machine is not feasible. Therefore, only one PLCI and one NCCI at a time is considered in the state diagram.

The **COMMON-ISDN-API** messages LISTEN_REQ and LISTEN_CONF are described in a separate state diagram because the availability of a successful LISTEN setting exceeds the lifetime of logical and/or physical connections.

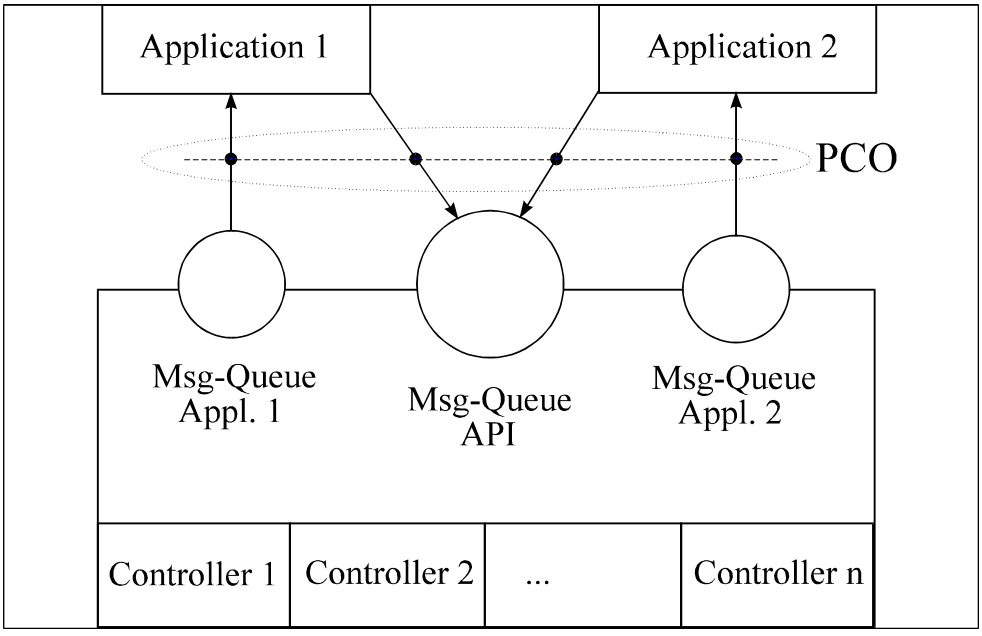


Figure 5: Position of PCO (Point of Control and Observation)

7.2 Explanation

The state diagrams assume an error-free exchange of messages. The point of control and observation (PCO) for the message exchange description is at the level of the CAPI operations. For real implementations, an asynchronous exchange of messages is not allowed to result in an error condition.

The state diagrams describe the flow of the messages at the PCO without consideration of their possible asynchronicity in real implementations.

For the sake of simplicity, confirmations and responses which do not induce a state transition are not shown in these state diagrams.

An expected confirmation of a request or an expected response to an indication is allowed to appear in “ANY” state.

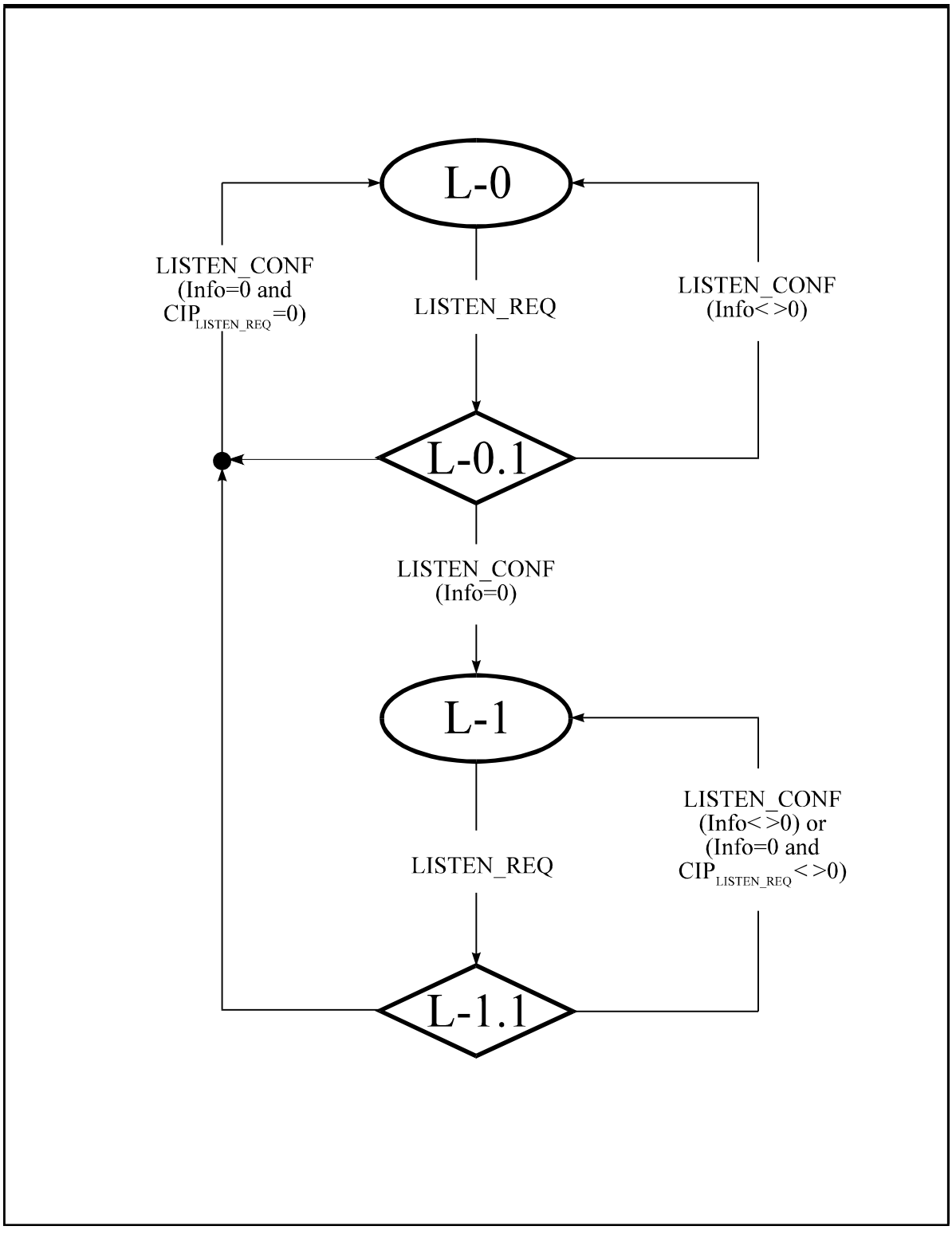
The messages MANUFACTURER_REQ, MANUFACTURER_CONF, MANUFACTURER_IND and MANUFACTURER_RESP may cause incompatibility. They are not described in the state diagrams.

Requests with an invalid PLCI or an invalid NCCI are incorrect messages and are therefore not described in the state diagrams.

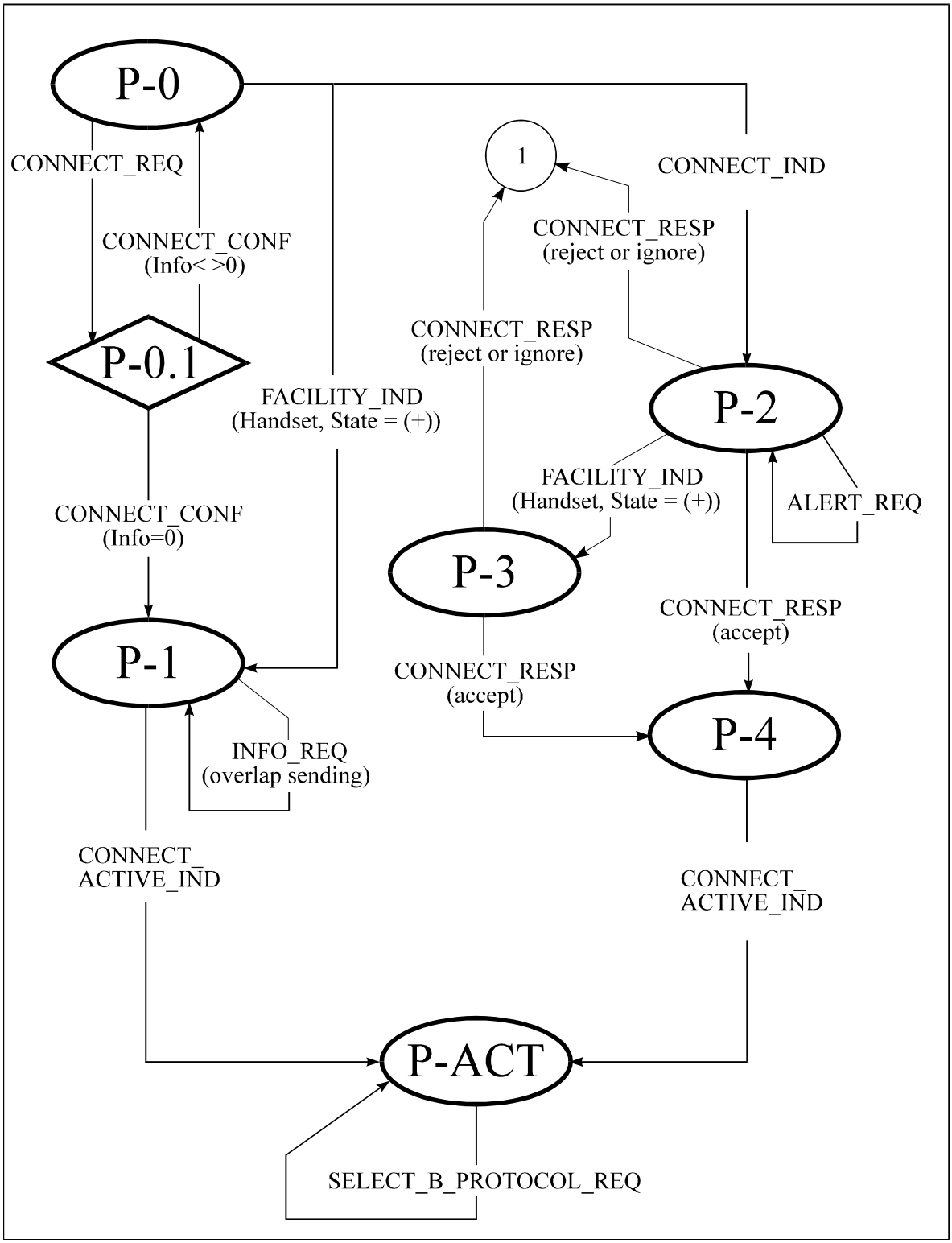
INFO_REQ and INFO_IND are network-specific elements which can appear at any time. The use of INFO_REQ for "overlap sending" in particular is described in the PLCI state machine 1/2.

FACILITY_REQ, FACILITY_CONF, FACILITY_IND and FACILITY_RESP are facility-specific messages which can appear at any time. Therefore they can occur in every state of the LISTEN, PLCI and NCCI state machines. The FACILITY_IND concerning "Handset Support" is described in particular in the PLCI state machine 1/2. The flow of messages for Handset Support depends on the actual handset interface (such as AEI, or the Additional Equipment Interface) or manufacturer-specific codecs. For this reason can occur that only a part of the described message flow for Handset Support is used. However, the FACILITY messages for Handset Support may not be used in a different way from that described in the message definitions and the state machines.

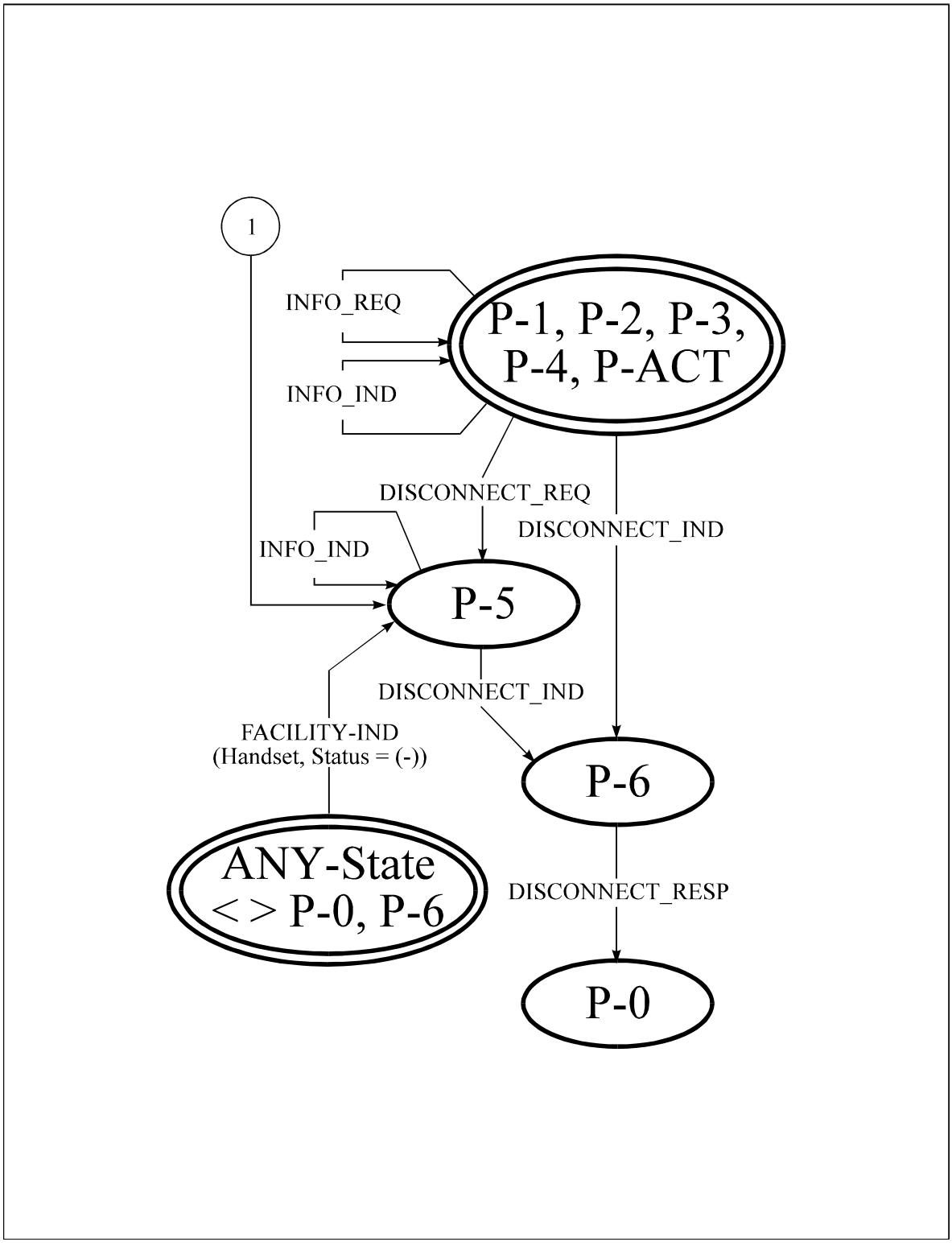
LISTEN - state machine



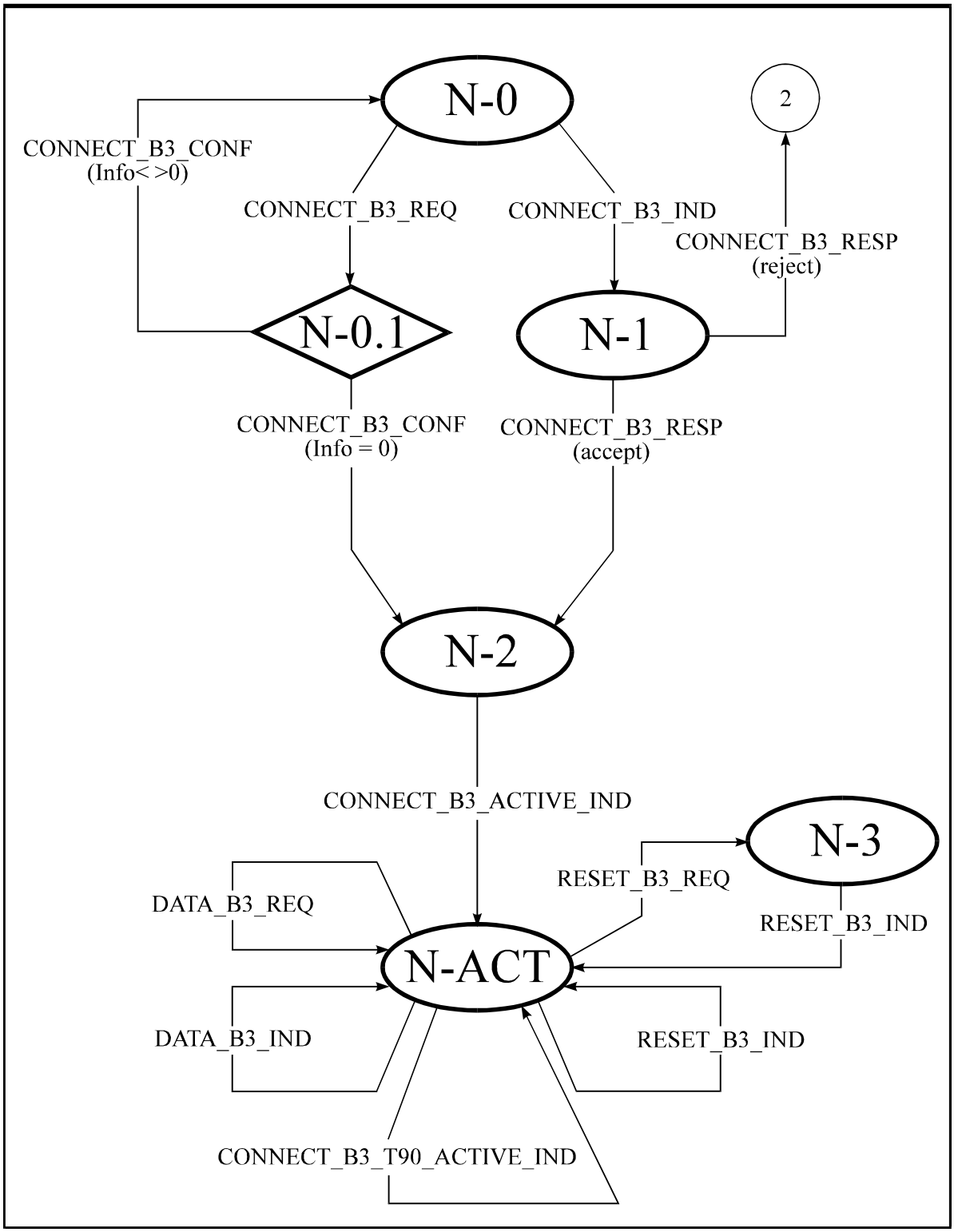
PLCI - state machine 1/2



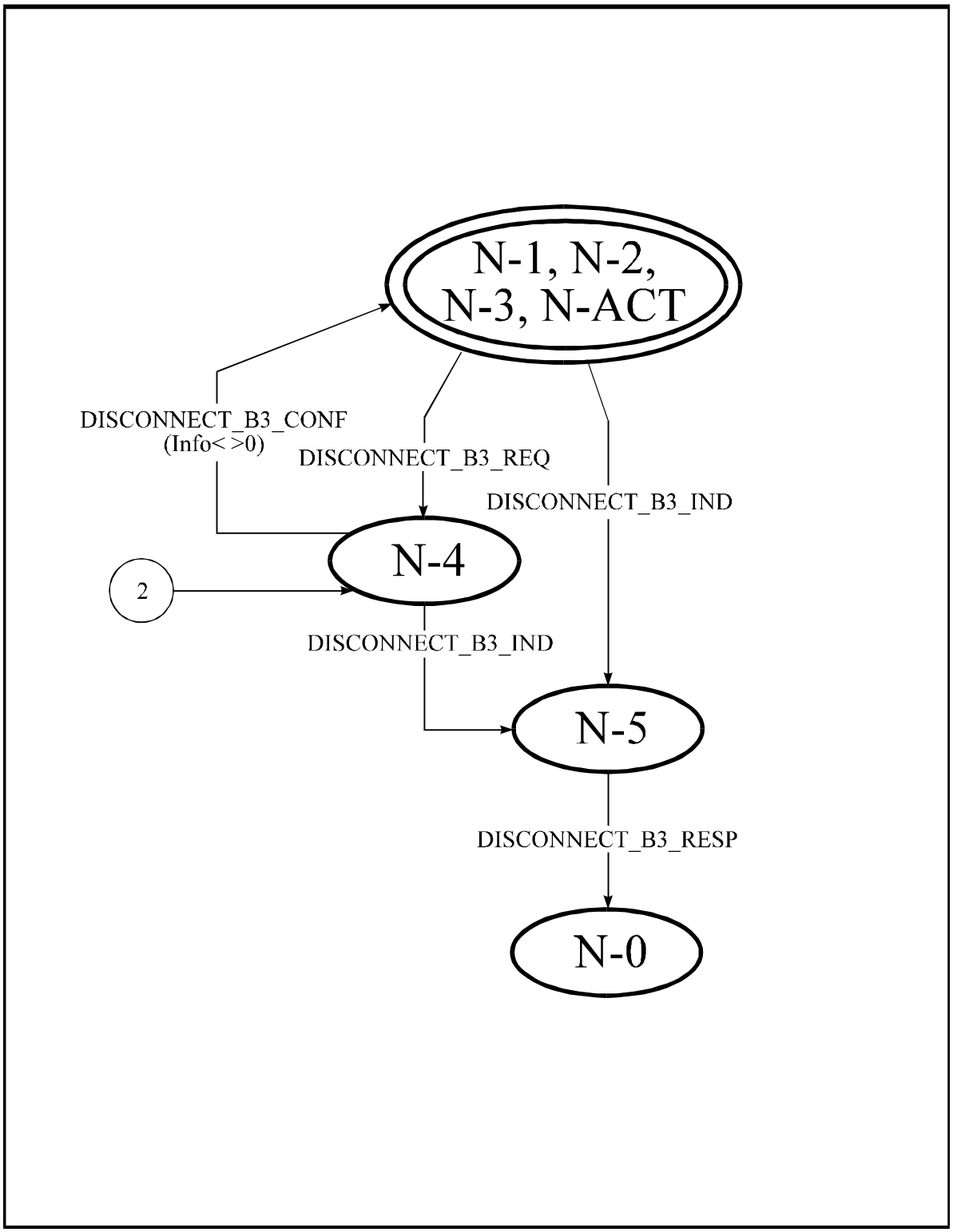
PLCI - state machine 2/2



NCCI - state machine 1/2



NCCI - state machine 2/2



8 SPECIFICATIONS FOR COMMERCIAL OPERATING SYSTEMS

COMMON-ISDN-API can be used with the following operating systems:

MS-DOS
Windows 3.x application level
OS/2 application level
OS/2 device driver level
UNIX
NetWare
Windows NT application level
Windows NT device driver level
Windows 95 application level
Windows 95 device driver level
Windows 95 IOCtl Access
Windows 98 application level
Windows 98 device driver level
Windows 2000 application level
Windows 2000 device driver level
Linux application level
Linux kernel level
Windows XP 32bit application layer
Windows XP 64bit application layer
Windows XP device driver level

All operating systems support the following **COMMON-ISDN-API** operations:

- CAPI_REGISTER — Register application with **COMMON-ISDN-API**
- CAPI_RELEASE — Release application from **COMMON-ISDN-API**
- CAPI_PUT_MESSAGE — Transfer message to **COMMON-ISDN-API**
- CAPI_GET_MESSAGE — Retrieve message from **COMMON-ISDN-API**
- CAPI_GET_MANUFACTURER — Get manufacturer information from **COMMON-ISDN-API**
- CAPI_GET_VERSION — Get version information from **COMMON-ISDN-API**
- CAPI_GET_SERIAL_NUMBER — Get serial number of **COMMON-ISDN-API**
- CAPI_GET_PROFILE — Get capability information from **COMMON-ISDN-API**

Depending on the operating system, the following **COMMON-ISDN-API** operations may also be available:

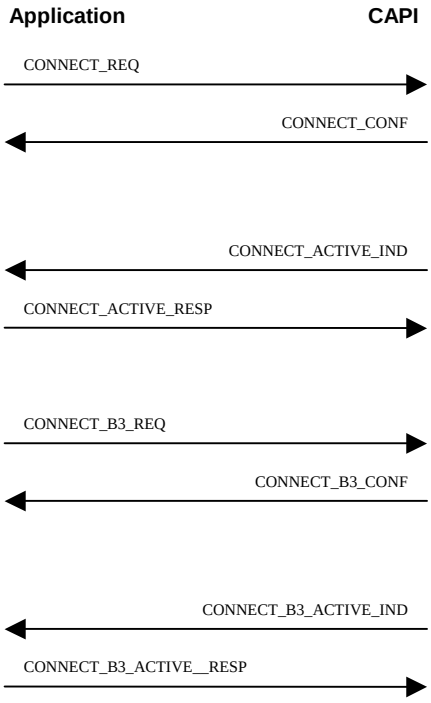
- CAPI_SET_SIGNAL — Install call-back function
- CAPI_WAIT_FOR_SIGNAL — Wait for **COMMON-ISDN-API** message
- CAPI_INSTALLED — Check whether **COMMON-ISDN-API** is installed
- CAPI_MANUFACTURER — Manufacturer-specific **COMMON-ISDN-API** operation

See **COMMON-ISDN-API** Part II for details.

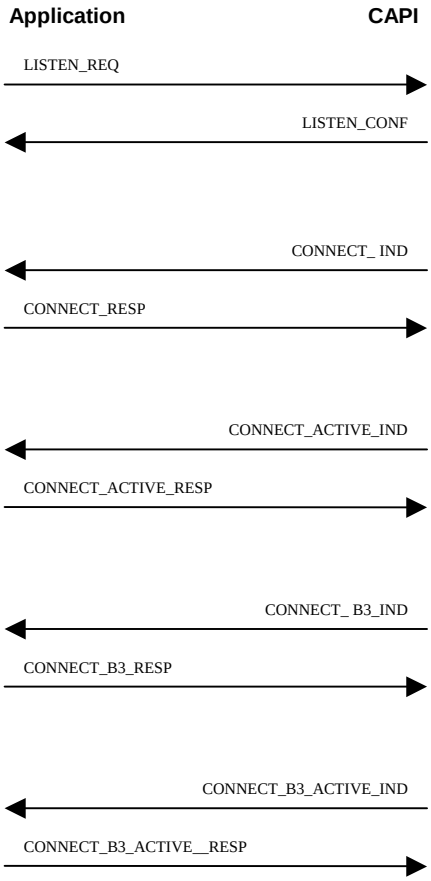
ANNEX A (INFORMATIVE): SAMPLE FLOW CHART DIAGRAMS

A.1 Call Establishment

A.1.1 Outgoing Call

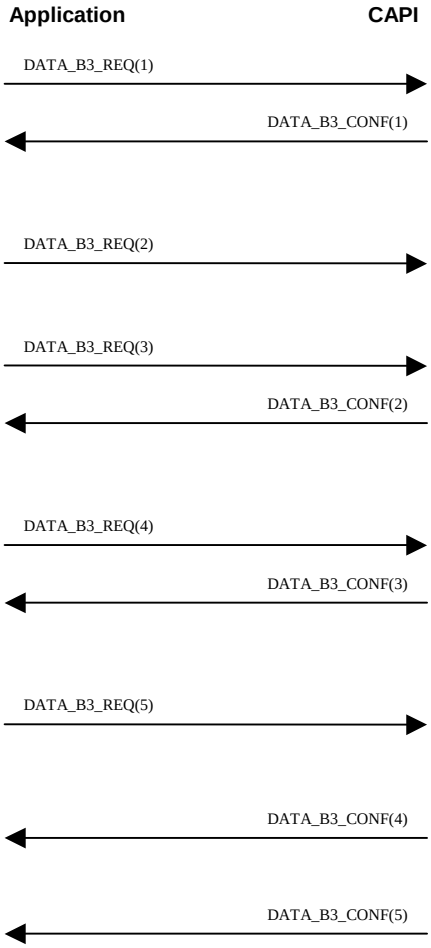


A.1.2 Incoming Call

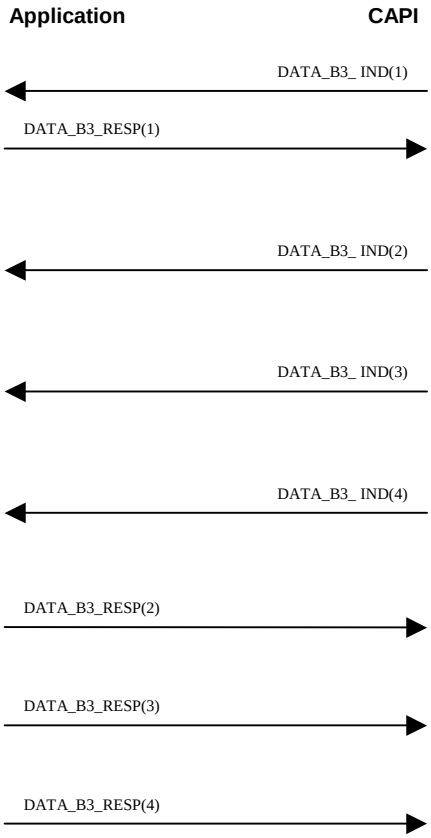


A.2 Data Transfer

A.2.1 Transmitting Data

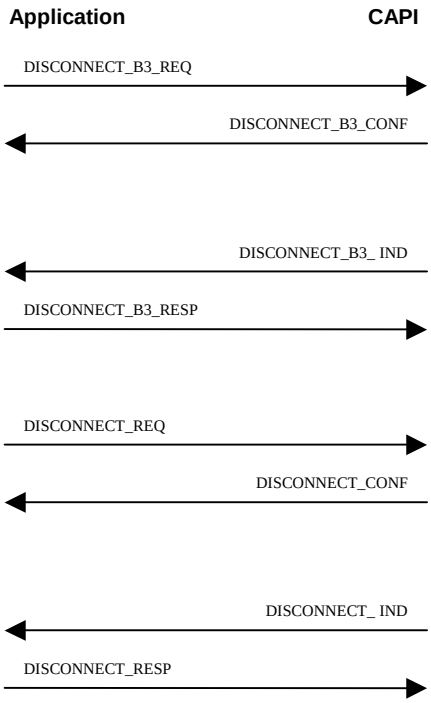


A.2.2 Receiving Data

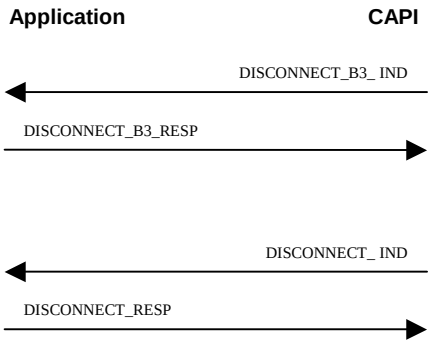


A.3 Call Clearing

A.3.1 Active Disconnect

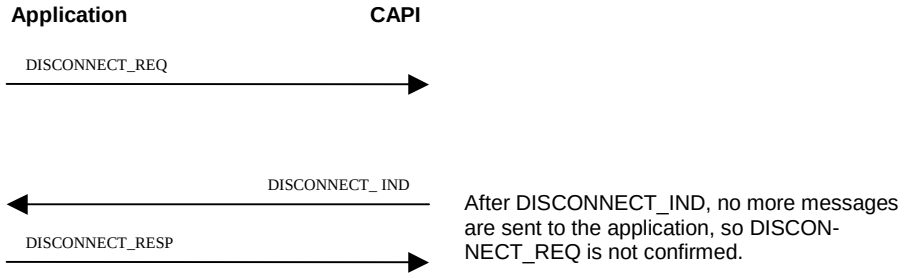


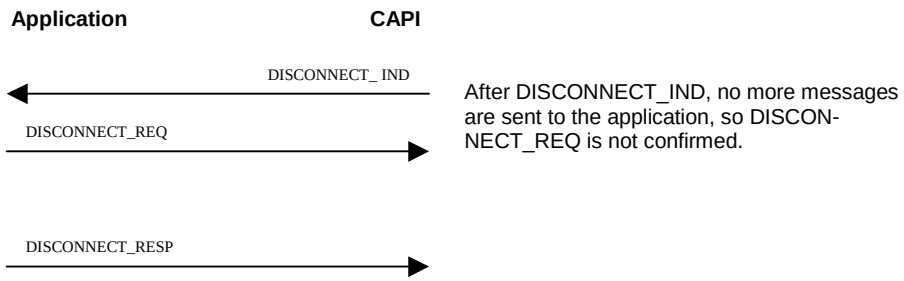
A.3.2 Passive Disconnect

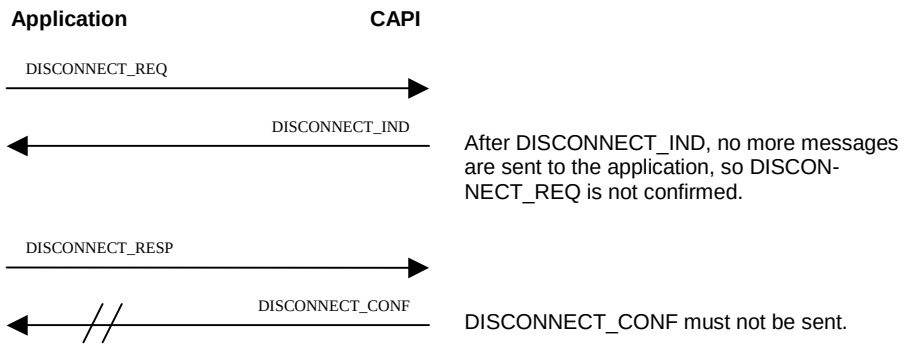


A.3.3 Disconnect Collision

Simultaneous release of a physical connection by the application and **COMMON-ISDN-API**





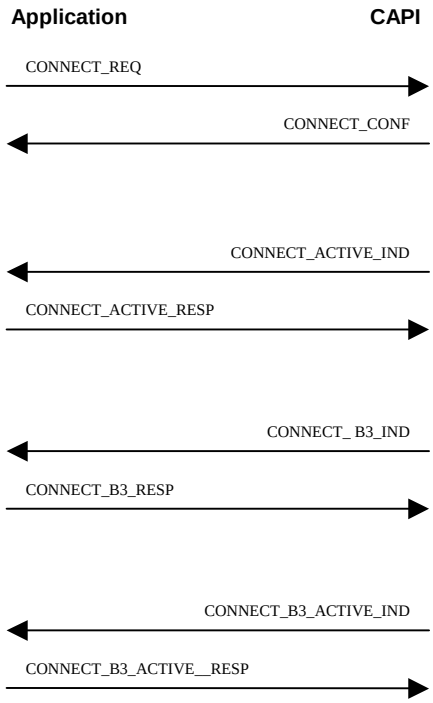


A.4 X.25 D-channel (X.31 case B)

A.4.1 Outgoing Call

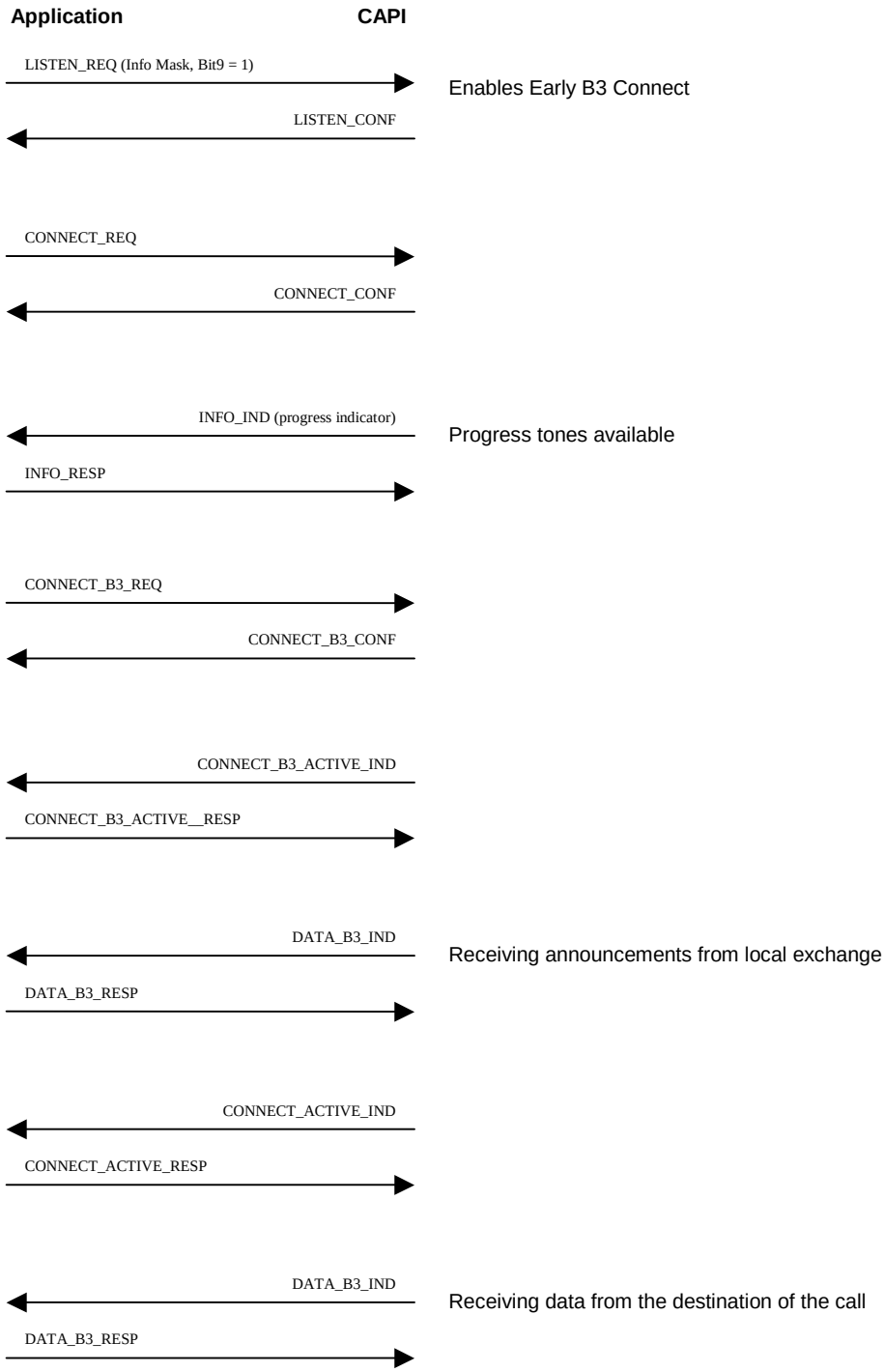
See A.1.1 “Outgoing Call”

A.4.2 Incoming Call

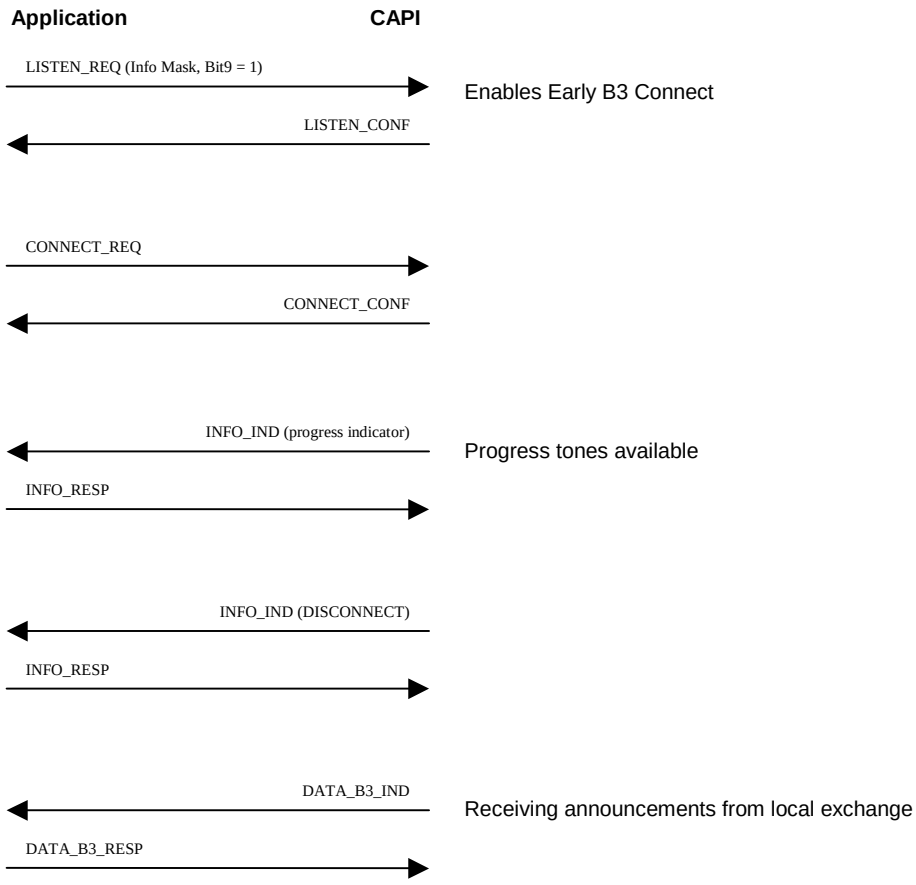


A.5 Early B3 Connect

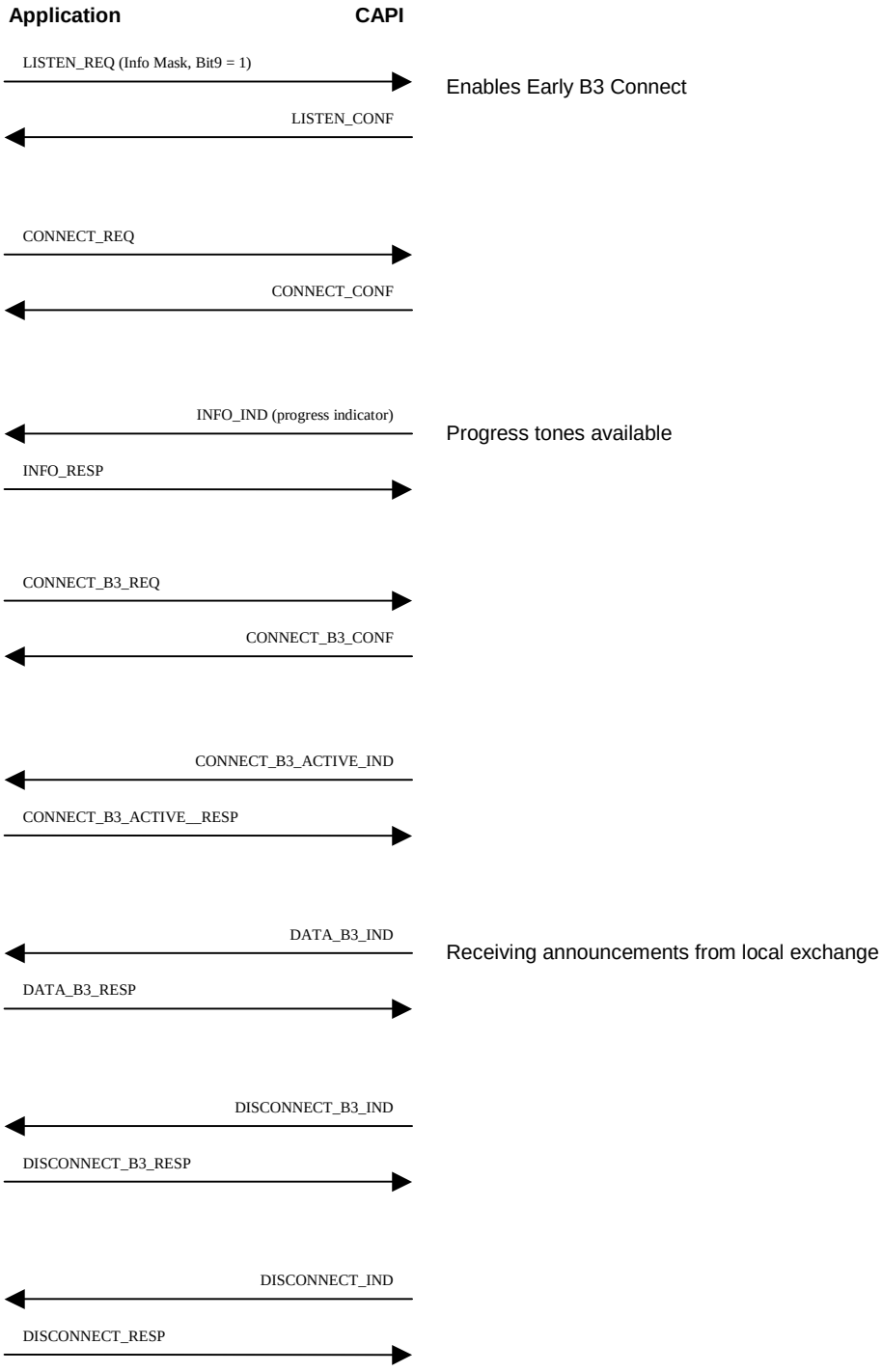
A.5.1 Call Establishment (Successful)



A.5.2 Call Clearing

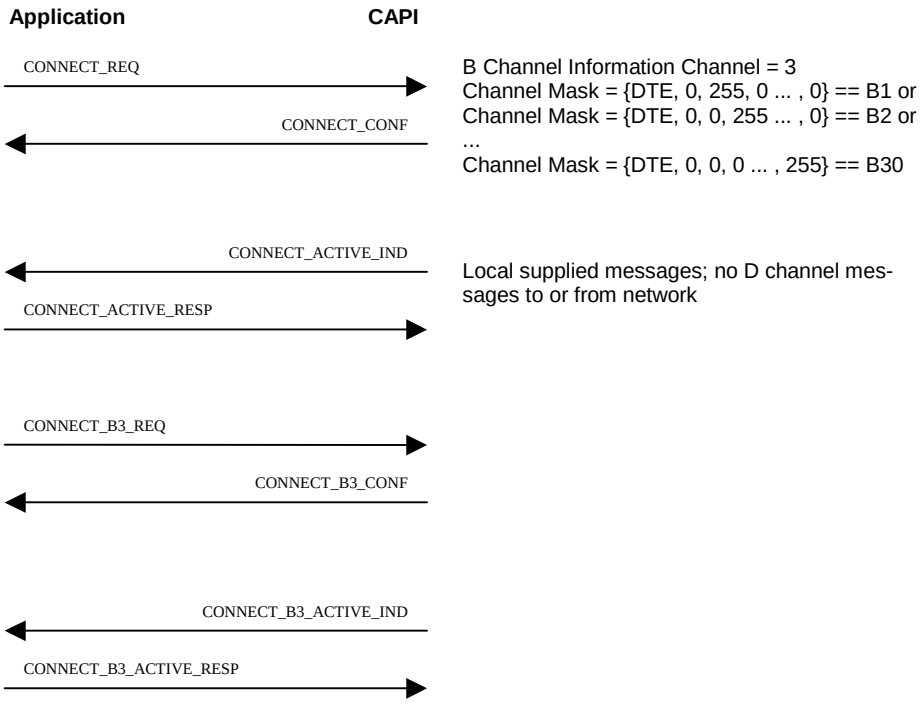


A.5.3 Call Establishment (Unsuccessful)

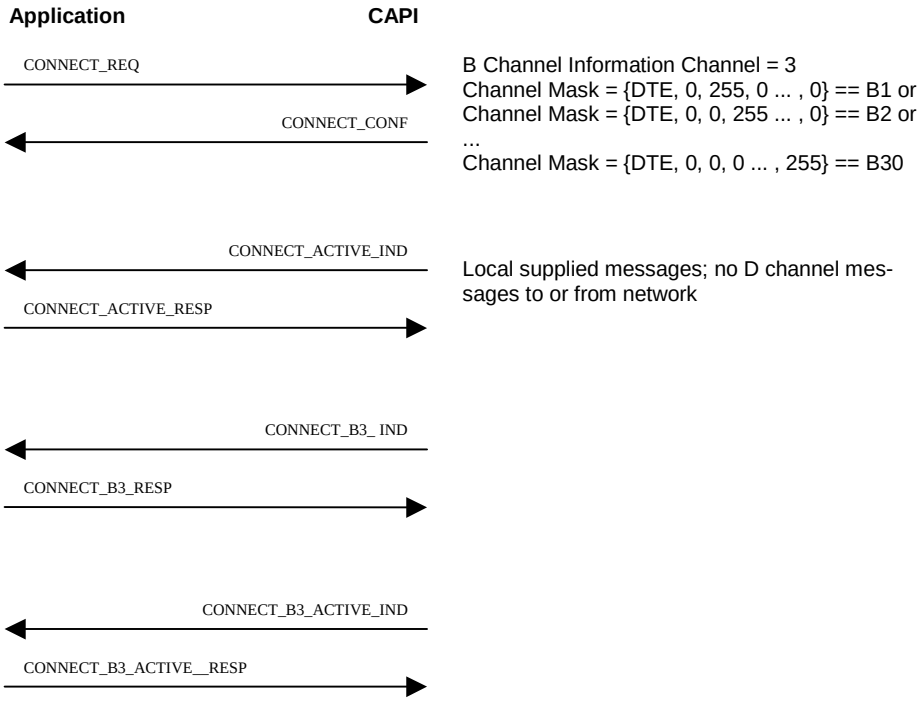


A.6 Permanent Connection

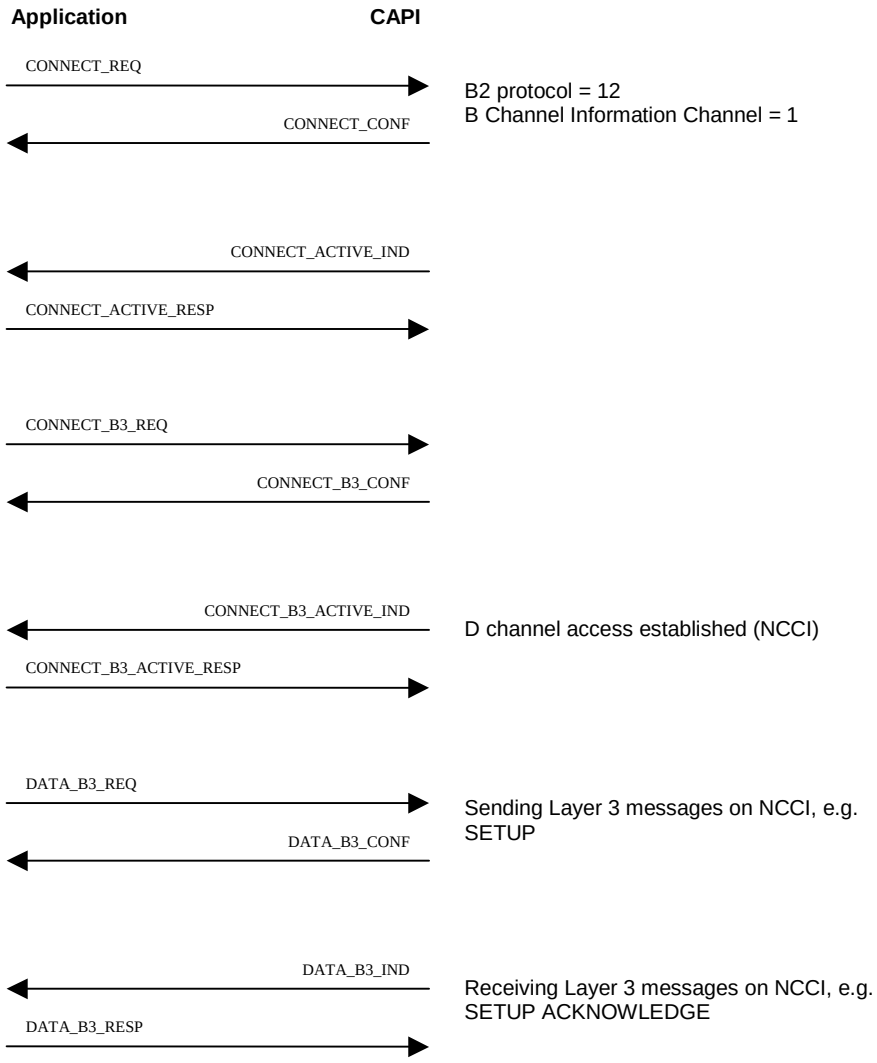
A.6.1 Outgoing Call



A.6.2 Incoming Call

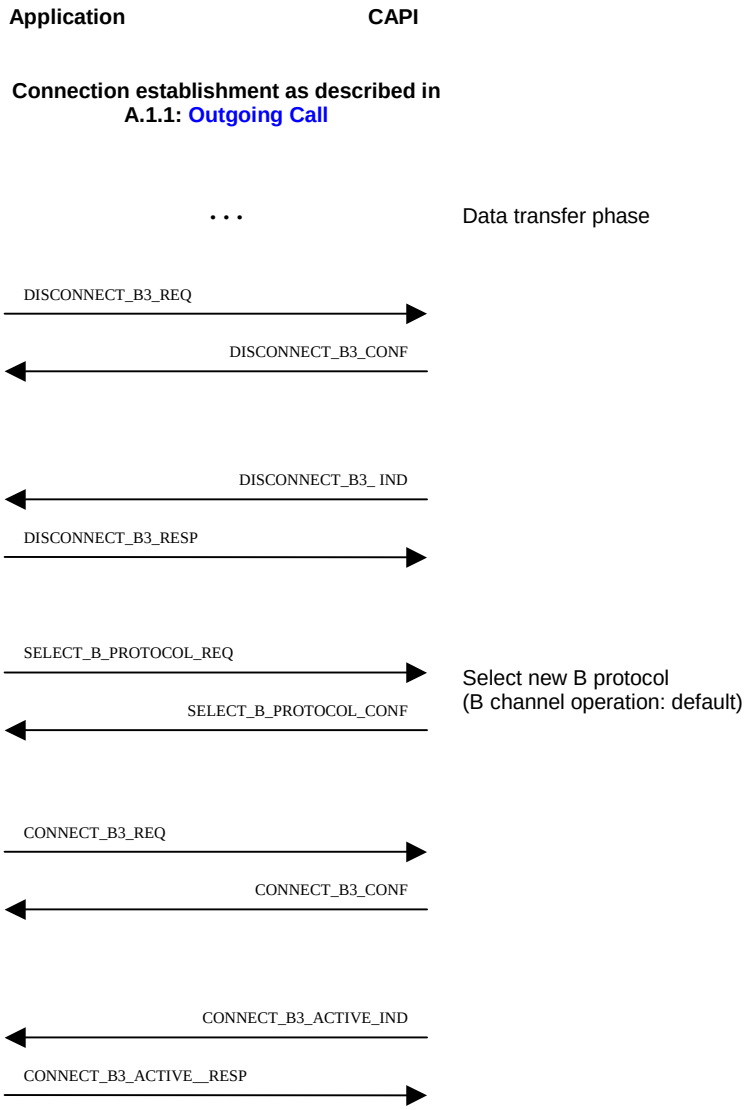


A.7 D Channel Layer 2 Access

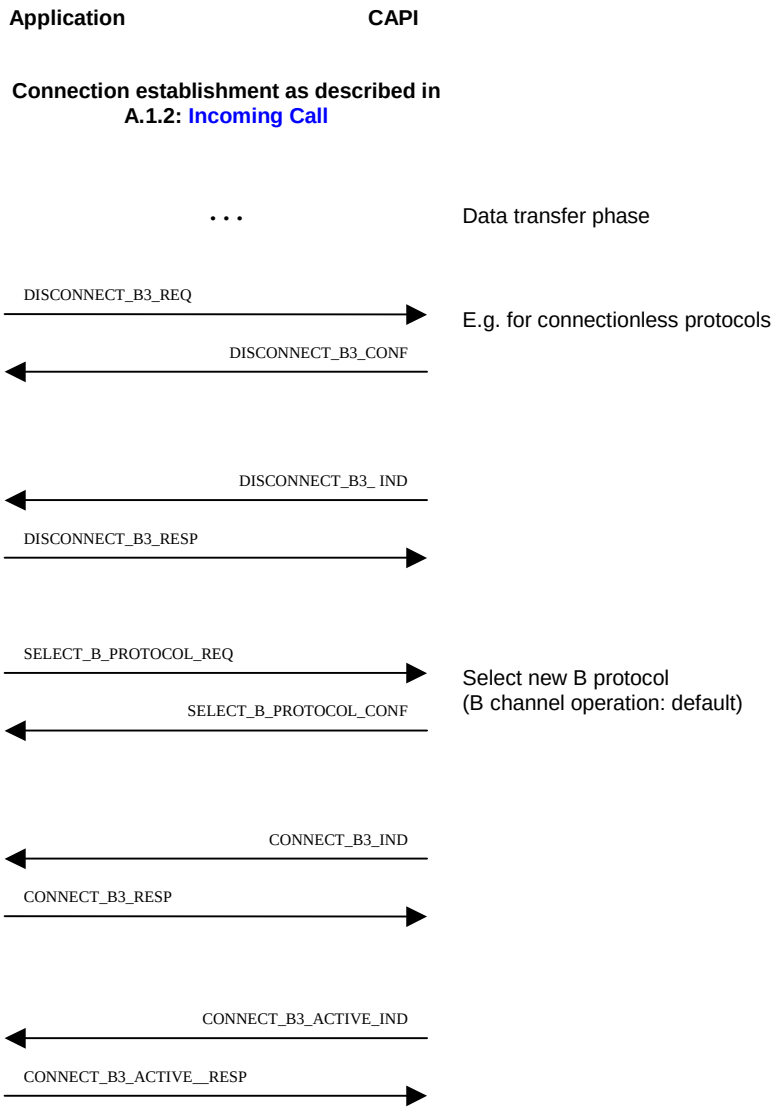


A.8 Change of the B Protocol

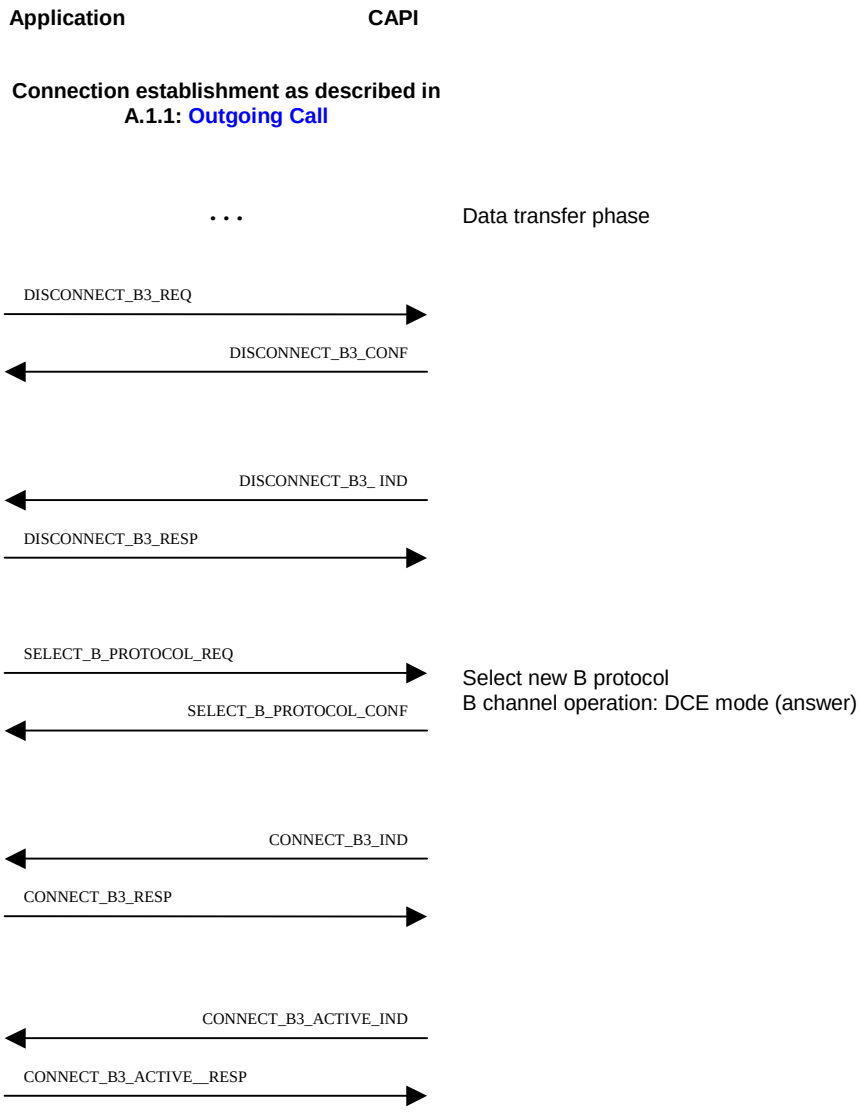
A.8.1 Outgoing Call



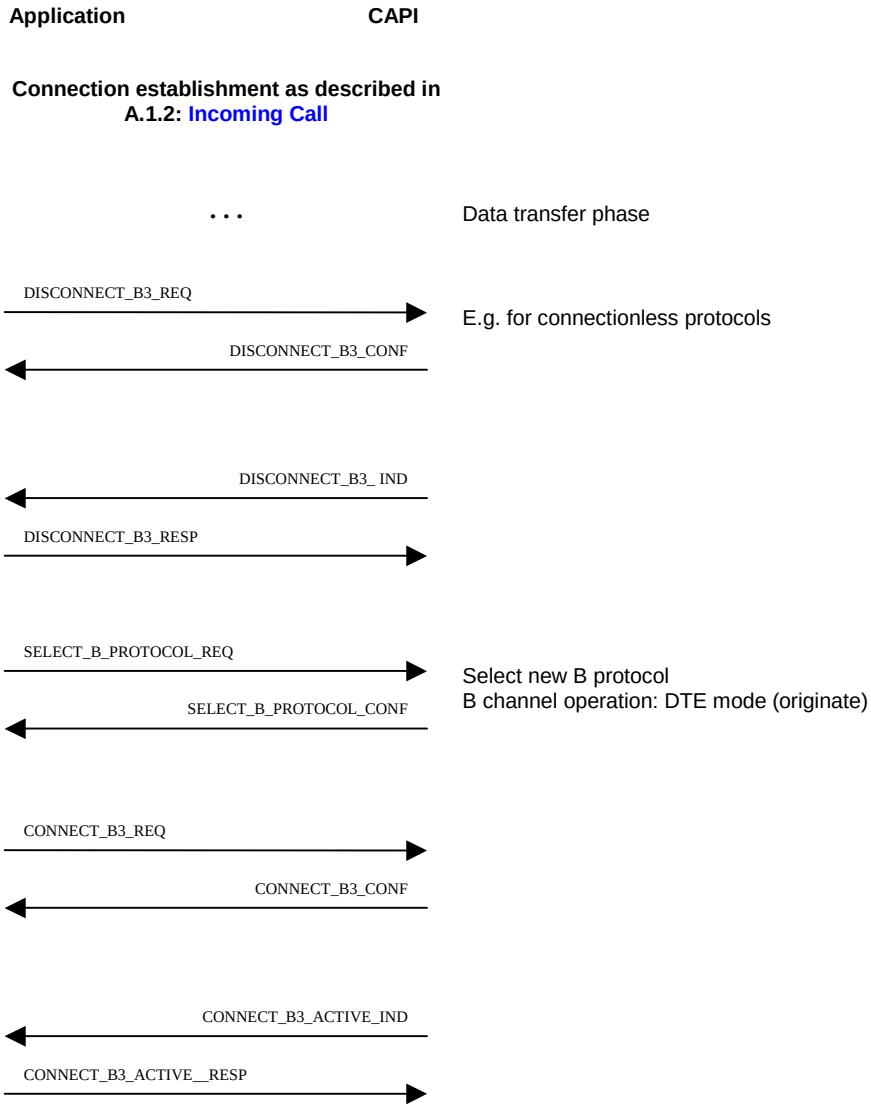
A.8.2 Incoming Call



A.8.3 Outgoing Call - Change of B Channel Operation Mode

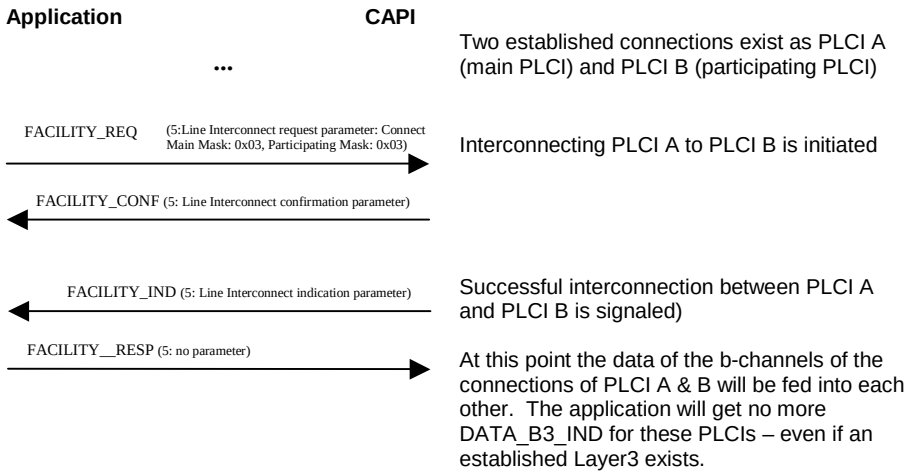


A.8.4 Incoming Call - Change of B Channel Operation Mode

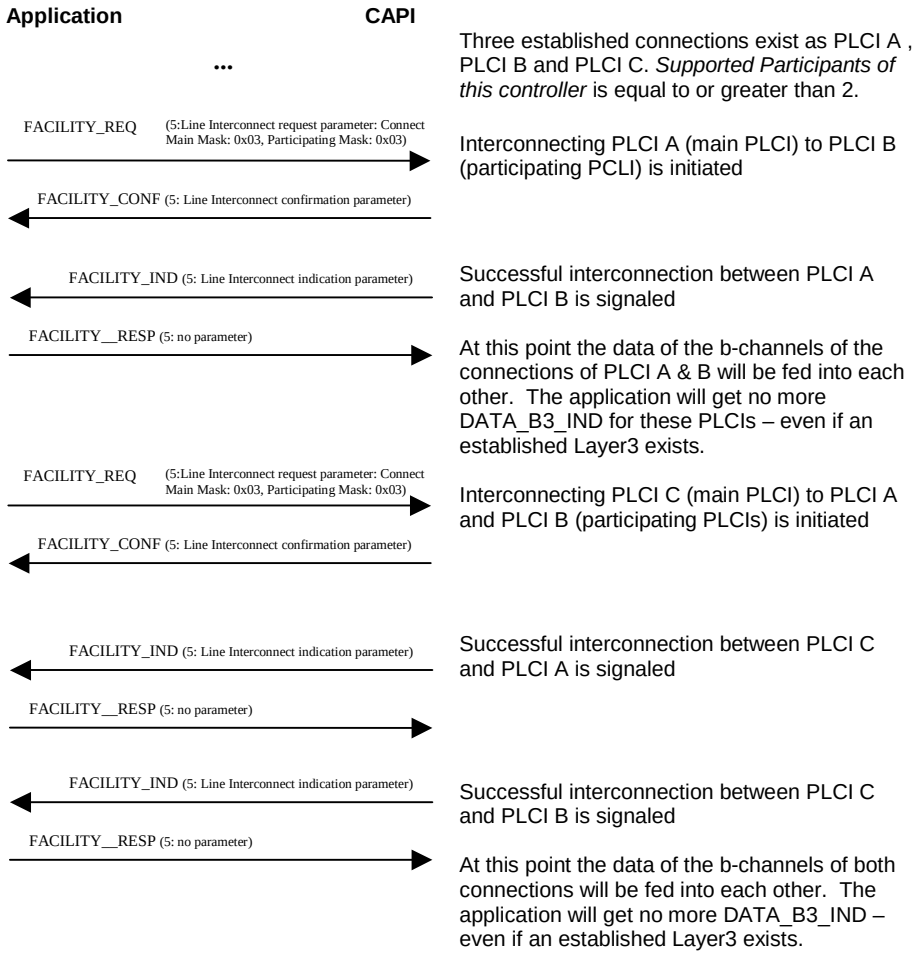


A.9 Line Interconnect

A.9.1 Interconnection (Two Participants)

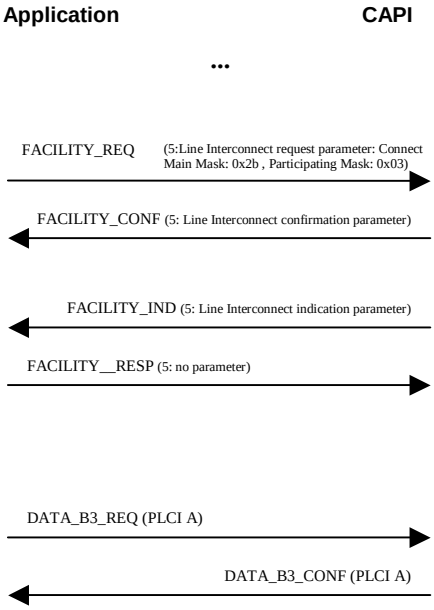


A.9.2 Conferencing (Three Participants)



Establishing a symmetric conference for more than three participants is analog. One would simply add all existing participants to the *Interconnection Connect Request Participants* struct for each new invocation of the line-interconnect-connect FACILITY_REQ (so one more participant for each new invocation)

A.9.3 Switching (Local & Remote Mixing)



Two established connections exist as PLCI A (main PLCI) and PLCI B (participating PLCI). *Supported Services Bit[3]: Mixing supported and Bit[5]: Remote Mixing* are set

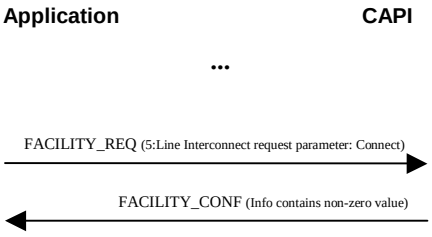
Interconnecting PLCI A to PLCI B is initiated (Bits 3 and 5 of the *Main PLCI Data Mask* are set)

Successful interconnection between PLCI A and PLCI B is signaled

At this point the data of the b-channels of the connections of PLCI A & B will be fed into each other. The application will get no more DATA_B3_IND for these PLCIs – even if an established Layer3 exists.

Data is being mixed with the data of b-channel of PLCI A and streamed into b-channel of PLCI B and mixed with the data of b-channel of PLCI B and streamed into b-channel of PLCI A

A.9.4 Failed Switching (Confirmation)

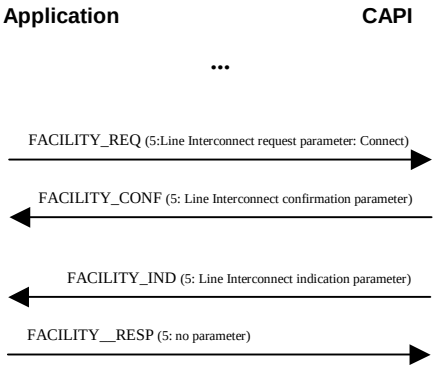


Two established connections exist as PLCI A (main PLCI) and PLCI B (participating PLCI)

Interconnecting PLCI A to PLCI B is initiated

Application is informed that the interconnection could not be initiated (due to a lack of free resources or not implemented functionality for example)

A.9.5 Failed Switching (Indication)

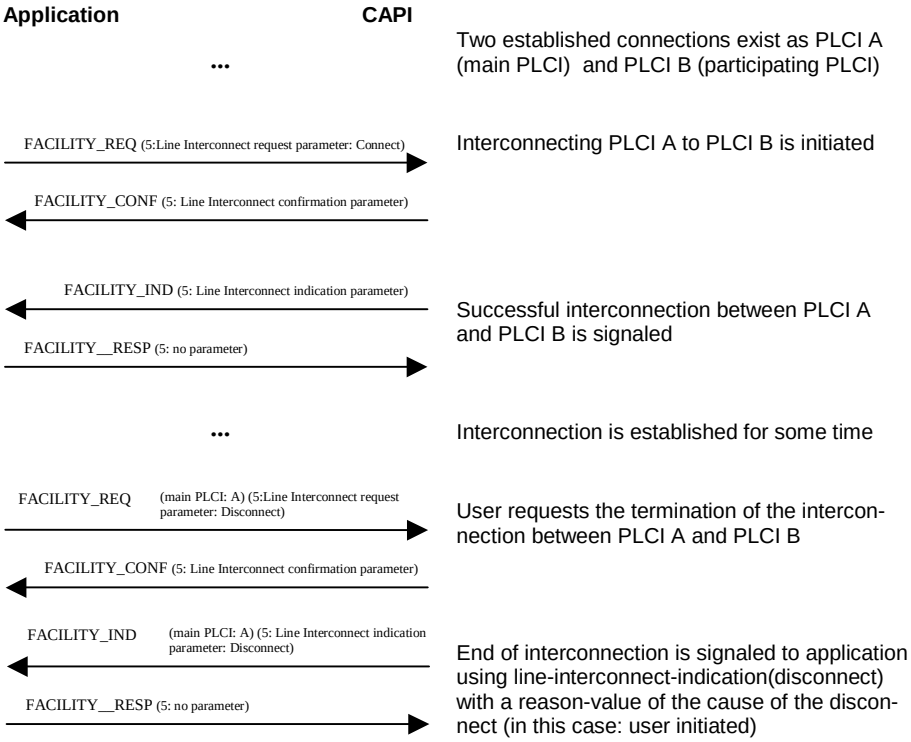


Two established connections exist as PLCI A (main PLCI) and PLCI B (participating PLCI)

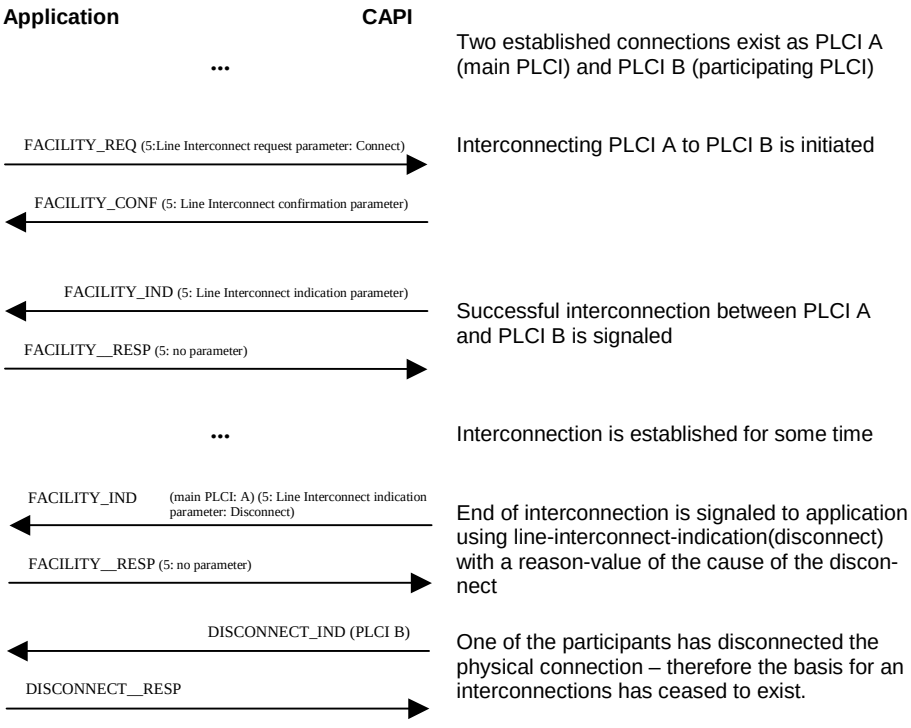
Interconnecting PLCI A to PLCI B is initiated

Unsuccessful initiation of the interconnection is signaled by issuing a line-interconnect-disconnect with a line-interconnect-reason specifying the cause of the failure.

A.9.6 Call Clearing



A.9.7 Call Clearing (Initiated by Remote Party)



ANNEX B (NORMATIVE): SFF FORMAT

B.1 Introduction

SFF (**S**tructured **F**ax **F**ile) is a representation especially for Group 3 fax documents. It consists of information concerning the page structure and the compressed line data of the fax document. An SFF-formatted document always starts with a header, which is valid for the complete document. Every page starts with a page header. This is followed by the pixel information, line by line. As the SFF format is a file format specification, some entries in header structures (e.g. double-link chaining of pages) may not be used or supported by **COMMON-ISDN-API**.

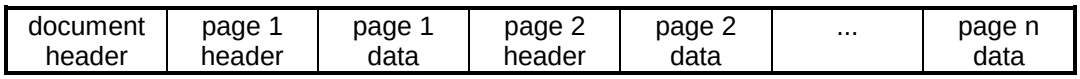


Figure 6: SFF format

B.2 SFF Coding Rules

The following data type conventions are used:

byte	8-bit unsigned
word	16-bit unsigned integer, least significant octet first
dword	32-bit unsigned integer, least significant word first

B.2.1 Document Header

Parameter	Type	Comment
SFF_Id	dword	Magic number (identification) of SFF Format: coded as **0x66666653** ("**SFFF**")
Version	byte	Version number of SFF document: coded **0x01**
reserved	byte	Reserved for future extensions; coded **0x00**
User Information	word	Manufacturer-specific user information (not used by **COMMON-ISDN-API**, coded as **0x0000**)
Page Count	word	Number of pages in the document. Must be coded **0x0000** if not known (as in the case of receiving a document).
OffsetFirstPageHeader	word	Byte offset of first page header from start of document header. This value is normally equal to the size of the document header (**0x14**), but there could be additional user-specific data between the document header and the first page header. **COMMON-ISDN-API** ignores and does not provide such additional data.
OffsetLastPageHeader	dword	Byte offset of last page header from start of document header. Must be coded **0x00000000** if not known (as in the case of receiving a document).
OffsetDocumentEnd	dword	Byte offset of document end from start of document header. Must be coded **0x00000000** if not known (as in the case of receiving a document).

B.2.2 Page Header

Parameter	Type	Comment
PageHeaderID	byte	**254** (Page data record type)
PageHeaderLen	byte	**0:** Document end **1...255**: byte offset of first page data from the *Resolution Vertical* field of the page header. This value is normally equal to the size of the remainder of the header (**0x10**), but there may be additional user-specific data between page header and page data. **COMMON-ISDN-API** ignores and not provide such additional data.
Resolution Vertical	byte	Definition of vertical resolution; different resolutions in one document may be ignored by **COMMON-ISDN-API** **0:** 98 lpi (standard) **1:** 196 lpi (high resolution) **2...254:** reserved **255:** End of document (should not be used: *PageHeaderLen* should be coded **0** instead)
Resolution Horizontal	byte	Definition of horizontal resolution **0:** 203 dpi (standard) **1...255:** reserved
Coding	byte	Definition of pixel line coding **0:** Modified Huffman coding **1...255**: reserved
reserved	byte	Coded as **0**
Line Length	word	Number of pixels per line **1728:** Standard G3 fax **2048:** B4 (optional) **2432:** A3 (optional) Support for additional values is optional for **COMMON-ISDN-API**.
Page Length	word	Number of pixel lines per page. If not known, coded as **0x0000**.
OffsetPreviousPage	dword	Byte offset to previous page header or **0x00000000**. Coded as **0x00000001** if first page.
OffsetNextPage	dword	Byte offset to next page header or **0x00000000**. Coded as **0x00000001** if last page.

B.2.3 Page Data

Page data is coded line by line: data describes each pixel row. Lines are coded as records of variable length; each line is coded according to the element *coding* in the page header. At present, only modified Huffman coding is supported. MH coding is bit-oriented: the pixel bits are stored in the bits of code words, least significant first. No EOL code words or fill bits are included. If the data includes EOL code words, **COMMON-ISDN-API** ignores these.

Each record is identified by the first byte:

- **1...216:** a pixel row with 1...216 MH-coded bytes follows immediately
- **0:** escape code for a pixel row with more than 216 MH-coded bytes. In this case, the following word in the range **217...32767** defines the number of MH-coded bytes which follow.

- **217...253**: white space, skip 1...37 empty lines
- **254**: start of page header (see above)
- **255:** if followed by a byte with value **0**, illegal line coding. Applications may choose whether to interpret this line as empty or as a copy of the previous line. If this byte is followed by a byte with a value **1...255**, then 1...255 bytes of additional user information follow (reserved for future extensions).

ANNEX C (NORMATIVE): SUPPLEMENTARY SERVICES

Certain supplementary services are supported by **COMMON-ISDN-API** Part I:

- **MSN (Multiple Subscriber Number, ETS 300 050)**
  see parameter *Called/Calling Party Number*
- **CW (Call Waiting, ETS 300 056)**
  see parameter *B Channel Information*
- **SUB (Subaddressing, ETS 300 059**)
  see parameters *Called/Calling Party Subaddress*, *Connected Subaddress*
- **DDI (Direct Dialing In, ETS 300 062**)
  see parameters *Called Party Number* and *Info Mask* (bit 7)
- **CLIP/CLIR (Calling Line Identification Presentation/Restriction, ETS 300 089/090**)
  see parameters *Calling Party Number/Subaddress*
- **COLP/COLR (Connected Line Identification Presentation/Restriction, ETS 300 094/095)**
  see parameters *Connected Party Number/Subaddress*
- **AOC (Advice of Charge, ETS 300 178-180**)
  see parameter *Info Mask* (bit 6)
- **UUS1 (User-User Signaling Stage 1, ETS 300 284)**
  see parameter *Additional Info*
- **Redirection Number (ETS 300 207)**
  see parameter *Info Mask* (bit 10)
- **Redirecting Number (ETS 300 207)**
  see parameter *Info Mask* (bit 10)

COMMON-ISDN-API Part III covers the following supplementary services:

- **HOLD (Call Hold, ETS 300 139)**
- **TP (Terminal Portability, ETS 300 053)**
- **CF (Call Forwarding, ETS 300 199-201)**
- **CD (Call Deflection, ETS 300 202)**
- **ECT (Explicit Call Transfer, ETS 300 367)**
- **3PTY (Three-Party Conference, ETS 300 186)**
- **MCID (Malicious Call Identification, ETS 300 128)**
- **CCBS (Completion of Calls to Busy Subscriber, ETS 300 359-1 excluding Section 10)**
- **MWI (Message Waiting Indication, ETS 300 650).**

See **COMMON-ISDN-API** Part III for details.

ANNEX D (NORMATIVE): IMPLEMENTATION DETAILS

D.1 V.42 bis Compression

D.1.1 ISO 7776 (X.75 SLP) with V.42 bis Compression

The contents of every X.75 I frame shall be compressed in accordance with V.42 bis. To provide for the frame structure of ISO7776 (X.75 SLP), a V.42 bis C_FLUSH primitive has to be processed (i.e. need not be sent) after compressing/decompressing the data contents of a single frame. The encoder has to send a C_FLUSH code word in compressed mode to achieve byte-alignment at the end of the compressed data block if necessary. The decoder implicitly assumes byte-alignment at the beginning of each block. The C_FLUSH code word has to be accepted at any time in compressed mode.

After processing the internal C_INIT primitive following successful V.42 bis negotiation, the V.42 bis encoder and decoder begin working in **compressed** mode. Subsequent C_INIT primitives ('reset' procedures) do not change the state of the encoder/decoder; i.e. they do **not** cause a switch to compressed mode.

The C_FLUSH primitive processed (implicitly or explicitly) at the end of a data block executes the “Exception Process Next Character” procedure, whether in transparent or compressed mode: see Figure C-1.

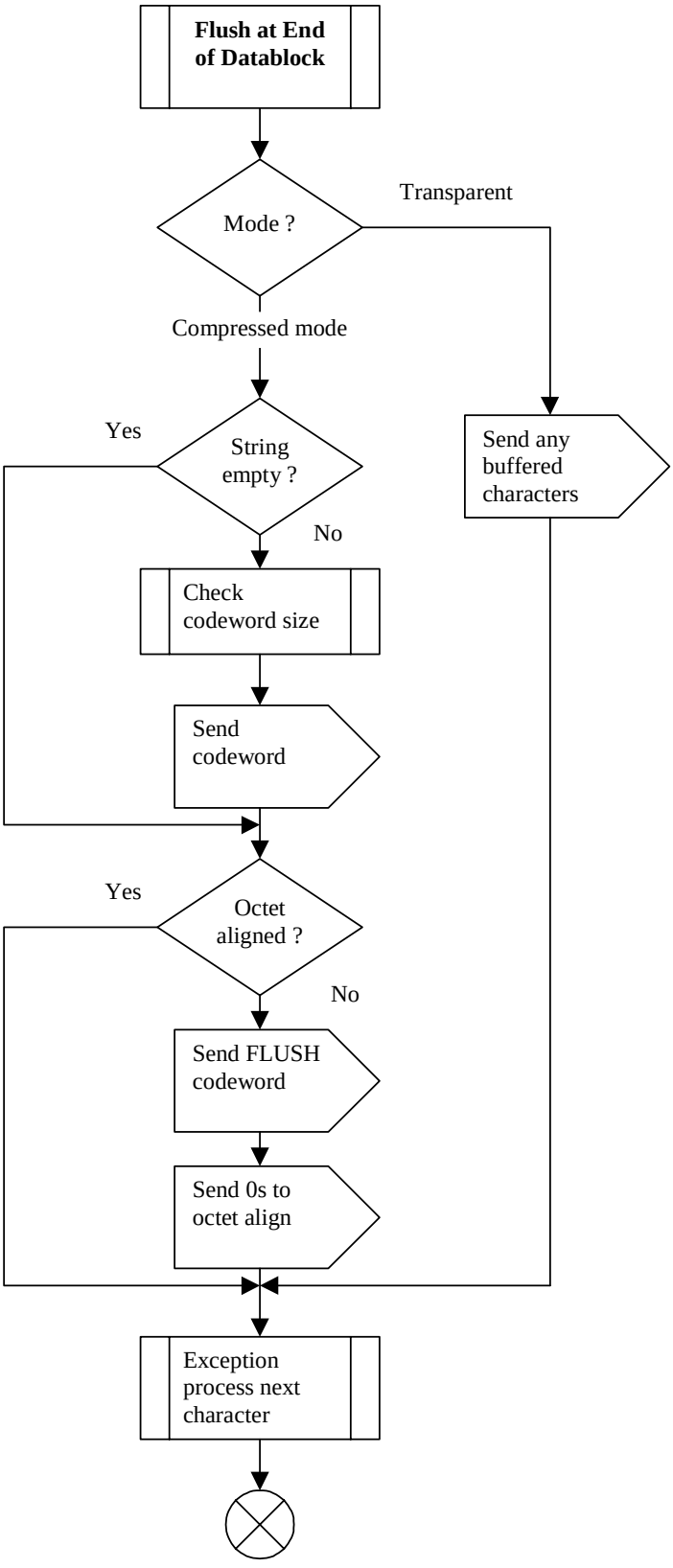


Figure C-1
Flush procedure at end of data block

The receiver must note that the length of transmitted frames can exceed the original data length. The transmitter shall try to avoid expansion of frames as a result of compression. The encoder should avoid such coding by switching to transparent mode, whereas the decoder has to support X.75 frames which in 'compressed' form exceed the maximum application block size. Switching from compressed to transparent mode and vice versa can be done at any time. Unnecessary switching shall be avoided, however.

Negotiation of compression parameters shall use the following XID mechanism, which guarantees compatibility with all existing ISO 7776 (X.75 SLP) implementations: The originator of the outgoing call shall send an XID frame (see coding below) before sending the X.75 SABM. A receiving station supporting ISO 7776 (X.75 SLP) without compression ignores the XID mechanism and responds with a UA frame as described in ISO 7776 (X.75 SLP). In this case, the originator shall not use compression. If the receiving station supports B2 protocol 8 (ISO 7776 (X.75 SLP) modified to support V.42 bis compression), then the XID frame contents are added to the UA frame. This is the indication that B2 protocol 8 (ISO 7776 (X.75 SLP) modified supporting V.42 bis compression) shall be used and the corresponding parameters negotiated.

Coding of the XID frame contents / UA frame extension (in accordance with Annex A of V.42 bis):

Bit	
7 0	
11110000	Private parameter set (ISO 8885, Addendum 3)
00000000	Length of parameter field (MSB)
00010110	Length of parameter field (LSB)
00000000	Parameter set identifier
00001010	Length of string
01000011	'C'
01000001	'A'
01010000	'P'
01001001	'I'
00110010	'2'
00110000	'0'
00101111	'/'
01011000	'X'
00110111	'7'
00110101	'5'
00000001	Rec. V.42 bis: Data compression request
00000001	Length of field
000000nn	Request for compression in: 00: Neither direction 01: Initiator to responder only 10: Responder to initiator only 11: Both directions
00000010	Rec. V.42 bis: Number of code words
00000010	16-bit integer
Nnnnnnnn	Value of parameter P1 (MSB)
Nnnnnnnn	Value of parameter P1 (LSB)
00000011	Rec. V.42 bis: Maximum string length
00000001	8-bit integer
Nnnnnnnn	Value of parameter P2

D.1.2 V.120 Asynchronous with V.42 bis Compression

The contents of every data block referenced by a DATA_B3_REQ shall be compressed in accordance with V.42 bis. To provide for the maximum Layer 2 blocksize of V.120 (259 bytes), **COMMON-ISDN-API** may send more than one V.120 I-frame.

In the receiving direction, an application may receive more than one DATA_B3_IND for one V.120 I-frame if the contents of the I-frame do not fit into the application's block size after decompression.

To negotiate the data compression mode, the XID negotiation procedure must be used (see §8.10 of Recommendation V.42). Coding of the XID frame contents (in accordance with Annex A of V.42 bis):

Bit	
7 0	
11110000	Private parameter set (ISO 8885, Addendum 3)
00000000	Length of parameter field (MSB)
00010000	Length of parameter field (LSB)
00000000	Parameter set identifier
00000100	Length of string
01010110	'V'
00110001	'1'
00110010	'2'
00110000	'0'
00000001	Rec. V.42 bis: Data compression request
00000001	Length of field
000000nn	Request for compression in: 00: Neither direction 01: Initiator to responder only 10: Responder to initiator only 11: Both directions
00000010	Rec. V.42 bis: Number of code words
00000010	16-bit integer
nnnnnnnn	Value of parameter P1 (MSB)
nnnnnnnn	Value of parameter P1 (LSB)
00000011	Rec. V.42 bis: Maximum string length
00000001	8-bit integer
nnnnnnnn	Value of parameter P2

D.2 CAPI Guard

CAPI Guard is an optional **COMMON-ISDN-API** mechanism for security purpose. It includes a *Call Control Supervision* for call control management, a *Supplementary Service Supervision* for supplementary service management as well as a general handling of other security items.

D.2.1 Call Control Supervision

D.2.1.1 Outgoing Call

Outgoing calls are cleared for security reason, if the combination of *called party number, called party subaddress* and *CIP Value* of the corresponding CONNECT_REQ is not allowed by Call Control Supervision. In case of overlap sending security clearing occurs after any INFO_REQ that builds a called party number which is not allowed.

Outgoing logical connections with B2 Protocol = 3 (“LAPD in accordance with Q.921 for D channel X.25”) are cleared for security reason, if the combination of X.25 Called DTE address (part of the X.25 Call Request coded in parameter *NCPI*) of the corresponding CONNECT_B3_REQ and TEI of the corresponding CONNECT_REQ is not allowed by Call Control Supervision.

D.2.1.2 Incoming Call

Incoming calls are not signaled for security reason if the combination of *calling party number, calling party subaddress* and *CIP Value* is not allowed by Call Control Supervision. In case of overlap receiving security clearing could occur after any INFO_IND that builds a calling party number which is not allowed.

Incoming logical connections with B2 Protocol = 3 (“LAPD in accordance with Q.921 for D channel X.25”) are not signaled for security reason if the combination of X.25 Calling DTE address (see X.25 Incoming Call in NCPI) and TEI in the corresponding CONNECT_REQ is not allowed by Call Control Supervision.

D.2.2 Supplementary Service Supervision

Only Call Forwarding and Call Deflection need extra security handling. All other supplementary services rely on *Call Control Supervision*.

Call Forwarding (CF Activate) is rejected for security reason, if parameters (Basic Service, Served User Number, Forwarded-to Number and Forwarded-to Subaddress) of the corresponding FACILITY_REQ are not allowed.

Call Deflection (CD) is rejected for security reason, if parameters (Deflected-to Number and Deflected-to Subaddress) and CONNECT_IND (CIP Value) are not allowed.

D.2.3 General Supervision

D.2.3.1 Keypad facility

The sub parameter *keypad* facility is a member of parameter *Additional Info* which occurs in different messages. It is used to send extra digits of dial information e.g. for overlap sending.

Normally overlap sending is handled by *Call Control Supervision*. To get additional security a **COMMON-ISDN-API** implementation should allow to disable the information element keypad facility. In this case each occurrence of keypad facility is ignored.

D.2.3.2 Facility data array

The sub parameter *facility data array* is a member of parameter *Additional Info* which occurs in different messages. It is used to carry additional network specific elements.

A **COMMON-ISDN-API** implementation should allow to disable the element *facility data array* to get maximum security. In this case each occurrence of *facility data array* is ignored.

D.2.3.3 LAPD in accordance with Q.921 including free SAPI selection

The B2 Protocol 12 (“LAPD in accordance with Q.921 including free SAPI selection”) is used to get direct access to the D channel layer 2.

This feature leaves great concerns about security because an application can get access to the network in a way that could not be controlled by the **COMMON-ISDN-API** implementation.

To secure this feature a **COMMON-ISDN-API** implementation should allow to disable the use of this protocol. In this case for an incoming or outgoing call where this protocol is selected the call is cleared by the message DISCONNECT_IND with *Reason* = 0x3305.