Dialogic® NaturalAccess™ ISDN Messaging API Developer's Manual

October 2010 64-0509-02 Rev A

www.dialogic.com

Table Of Contents

Copyright and legal notices

Revision history

Revision	Release date	Notes
9000-6272-25	November, 2001	SJC, for NACD 2002-1 Beta
9000-6272-26	May, 2002	LBG, NACD 2002-1
9000-6272-27	November, 2002	LBG, for Natural Access 2003-1 Beta
9000-6272-28	April, 2003	LBG, for Natural Access 2003-1
9000-6272-29	December, 2003	LBG, for Natural Access 2004-1 Beta
9000-6272-30	April, 2004	SRR, for Natural Access 2004-1
9000-6272-31	November, 2004	LBG, for Natural Access 2005-1 Beta
9000-6272-32	February, 2005	MCM, for Natural Access 2005-1
9000-6272-33	October, 2005	LBG, for Natural Access 2005-1, SP1
9000-6272-34	July, 2006	SRG, for Natural Access 2005-1, SP2
9000-6272-35	February, 2009	DEH, for Natural Access R8.1
64-0509-01	October 2009	LBG, NaturalAccess R9.0
64-0509-02 Rev A	October 2010	LBG, NaturalAccess R9.0.4
Last modified: 2010-09-20		

Refer to www.dialogic.com for product updates and for information about support policies, warranty information, and service offerings.

1. Introduction

The *Dialogic® NaturalAccess™ ISDN Messaging API Developer's Manual* provides:

- Background information about ISDN.
- A programming guide to NaturalAccess ISDN Messaging API applications.
- A summary of functions organized by task.
- An extensive reference section for functions, data structures, parameters, and codes.

The NaturalAccess ISDN Messaging product exposes ISDN protocols at three levels:

- The LAPD service access point (SAP) for layer 2 protocols, providing direct access to the D channel.
- The ACU SAP for access to a simplified abstraction of the Q.931 ISDN D channel layer 3 message protocols.
- An ISDN primary and basic rate trunk interface, which is channelized and integrated with the NaturalCallControl API.

This manual describes how to build applications that interface with the ISDN protocol stack at layers 2 and 3. For information on building applications that are integrated at higher layers, see the *Dialogic® NaturalAccess™ ISDN Software Developer's Manual*.

This document is for developers of ISDN applications in C who are programming at the messaging level. The developer should be experienced with ITU Q.931 ISDN call control messages and call states. The developer should also be familiar with Natural Access, basic telephony concepts, and the C programming language.

Terminology

Note: The product to which this document pertains is part of the NMS Communications Platforms business that was sold by NMS Communications Corporation (“NMS”) to Dialogic Corporation (“Dialogic”) on December 8, 2008. Accordingly, certain terminology relating to the product has been changed. Below is a table indicating both terminology that was formerly associated with the product, as well as the new terminology by which the product is now known. This document is being published during a transition period; therefore, it may be that some of the former terminology will appear within the document, in which case the former terminology should be equated to the new terminology, and vice versa.

Former terminology	Dialogic terminology
CG 6060 Board	Dialogic® CG 6060 PCI Media Board
CG 6060C Board	Dialogic® CG 6060C CompactPCI Media Board
CG 6565 Board	Dialogic® CG 6565 PCI Media Board
CG 6565C Board	Dialogic® CG 6565C CompactPCI Media Board
CG 6565e Board	Dialogic® CG 6565E PCI Express Media Board
CX 2000 Board	Dialogic® CX 2000 PCI Station Interface Board
CX 2000C Board	Dialogic® CX 2000C CompactPCI Station Interface Board
AG 2000 Board	Dialogic® AG 2000 PCI Media Board
AG 2000C Board	Dialogic® AG 2000C CompactPCI Media Board
AG 2000-BRI Board	Dialogic® AG 2000-BRI Media Board
NMS OAM Service	Dialogic® NaturalAccess™ OAM API
NMS OAM System	Dialogic® NaturalAccess™ OAM System
NMS SNMP	Dialogic® NaturalAccess™ SNMP API
Natural Access	Dialogic® NaturalAccess™ Software
Natural Access Service	Dialogic® NaturalAccess™ Service
Fusion	Dialogic® NaturalAccess™ Fusion™ VoIP API
ADI Service	Dialogic® NaturalAccess™ Alliance Device Interface API

Former terminology	Dialogic terminology
CDI Service	Dialogic® NaturalAccess™ CX Device Interface API
Digital Trunk Monitor Service	Dialogic® NaturalAccess™ Digital Trunk Monitoring API
MSPP Service	Dialogic® NaturalAccess™ Media Stream Protocol Processing API
Natural Call Control Service	Dialogic® NaturalAccess™ NaturalCallControl™ API
NMS GR303 and V5 Libraries	Dialogic® NaturalAccess™ GR303 and V5 Libraries
Point-to-Point Switching Service	Dialogic® NaturalAccess™ Point-to-Point Switching API
Switching Service	Dialogic® NaturalAccess™ Switching Interface API
Voice Message Service	Dialogic® NaturalAccess™ Voice Control Element API
NMS CAS for Natural Call Control	Dialogic® NaturalAccess™ CAS API
NMS ISDN	Dialogic® NaturalAccess™ ISDN API
NMS ISDN for Natural Call Control	Dialogic® NaturalAccess™ ISDN API
NMS ISDN Messaging API	Dialogic® NaturalAccess™ ISDN Messaging API
NMS ISDN Supplementary Services	Dialogic® NaturalAccess™ ISDN API Supplementary Services
NMS ISDN Management API	Dialogic® NaturalAccess™ ISDN Management API
NaturalConference Service	Dialogic® NaturalAccess™ NaturalConference™ API
NaturalFax	Dialogic® NaturalAccess™ NaturalFax™ API
SAI Service	Dialogic® NaturalAccess™ Universal Speech Access API
NMS SIP for Natural Call Control	Dialogic® NaturalAccess™ SIP API
NMS RJ-45 interface	Dialogic® MD1 RJ-45 interface
NMS RJ-21 interface	Dialogic® MD1 RJ-21 interface

Former terminology	**Dialogic terminology**
NMS Mini RJ-21 interface	Dialogic® MD1 Mini RJ-21 interface
NMS Mini RJ-21 to NMS RJ-21 cable	Dialogic® MD1 Mini RJ-21 to MD1 RJ-21 cable
NMS RJ-45 to two 75 ohm BNC splitter cable	Dialogic® MD1 RJ-45 to two 75 ohm BNC splitter cable
NMS signal entry panel	Dialogic® Signal Entry Panel
Video Access Utilities	Dialogic® NaturalAccess™ Video Access Toolkit Utilities
Video Mail Application Demonstration Program	Dialogic® NaturalAccess™ Video Access Toolkit Video Mail Application Demonstration Program
Video Messaging Server Interface	Dialogic® NaturalAccess™ Video Access Toolkit Video Messaging Server Interface
3G-324M Interface	Dialogic® NaturalAccess™ Video Access Toolkit 3G-324M Interface

2. NMS ISDN Messaging API overview

Integrated Services Digital Network (ISDN)

Integrated Services Digital Network (ISDN) is a continually evolving international standard for networking services, including voice and non-voice services. The network is completely digital from one end to the other. Voice information is digitized and sent in digital form. Signaling information is sent separately from voice information, using a method called common channel signaling (CCS).

This topic describes:

- ISDN protocols and protocol layering
- Information exchange between layers

ISDN protocols and protocol layering

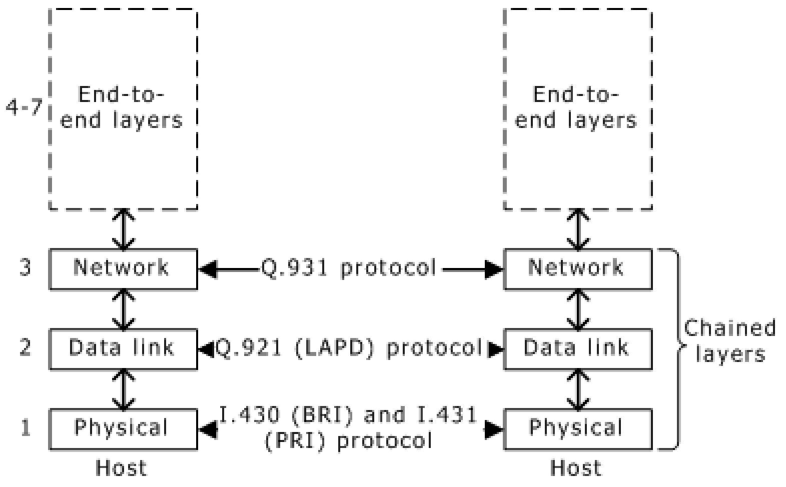
ISDN communications can be described at many levels, from the way bits are transferred from machine to machine to the sets of messages computers pass to one another. A scheme for communication at a certain level is called a protocol.

In the late 1970's, the International Standards Organization (ISO) established the Open Systems Interconnect (OSI) model for communication. ISDN is based on this model. In OSI, seven separate levels, or layers, of communication are defined. The first three layers, called the chained layers, are the lowest levels. The chained layers are:

Name	Number	Description
Physical layer	Layer 1	The electrical and mechanical layer. Protocols for this layer describe, on an electrical and mechanical basis, the methods used to transfer bits from one device to another. One protocol used at this layer is CCITT recommendation I.430/I.431
Data link layer	Layer 2	The layer above the physical layer. Protocols for this layer describe methods for error-free communication between devices across the physical link. One protocol used at this layer is CCITT recommendation Q.921, also known as Link Access Procedures on the D Channel (LAPD).
Network layer	Layer 3	The layer above the data link layer. Protocols for this layer describe methods for transferring information between computers. They also describe how data is routed within and between networks. One protocol used at this layer is CCITT recommendation Q.931.

Layers higher than these are end-to-end layers. They describe how information is exchanged and delivered end-to-end. They also define process-to-process communication, and describe application-independent user services, user interfaces, and applications.

The following illustration shows the hierarchy of layers:



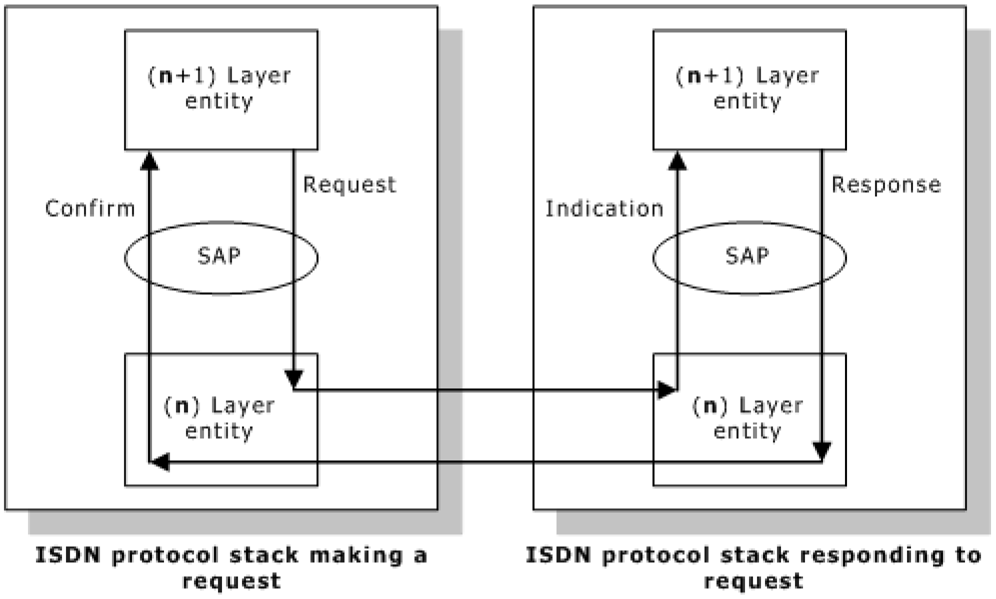
The functionality provided by a layer includes the services and functions of all of the layers below it. A service access point (SAP) is the point at which a layer provides services to the layer directly above it. A unique service access point identifier (SAPI) is associated with each SAP.

Information exchange between layers

Cooperation between entities on the same layer is governed by a peer-to-peer protocol specific to the layer and the entity. To exchange information between two or more layer entities, a connection must be established between the layer entities using the protocol of the layer directly below. Connections are provided by a layer between two or more SAPs.

Data message units are conveyed between peer-to-peer entities at the lowest layer by means of a physical connection. Layer (***n***+1) requests services from layer ***n*** through primitives. These primitives allow the logical exchange of information and control between two adjacent layers.

The following illustration shows the message primitives exchanged between layers:



Four types of primitives are exchanged between adjacent layers:

Primitive type	**Example**	**Description**
REQUEST (RQ)	ACU_CONN_RQ	A layer issues this type to request a service from the layer directly below it.
INDICATION (IN)	ACU_CONN_IN	A layer providing a service issues this primitive type to notify the layer above it of any specific activity that is related to the service. An INDICATION that a layer receives may be the result of an activity performed by the layer directly below it that is related to a REQUEST given by a peer entity.
RESPONSE (RS)	ACU_CONN_RS	A layer issues this primitive type to acknowledge the receipt of an INDICATION from a lower layer.
CONFIRM (CO)	ACU_CONN_CO	A layer providing a requested service issues this primitive type to confirm that the activity completed.

ISDN functional devices and reference points

ISDN equipment is classified into a number of categories by international and United States domestic standards as described in the following table:

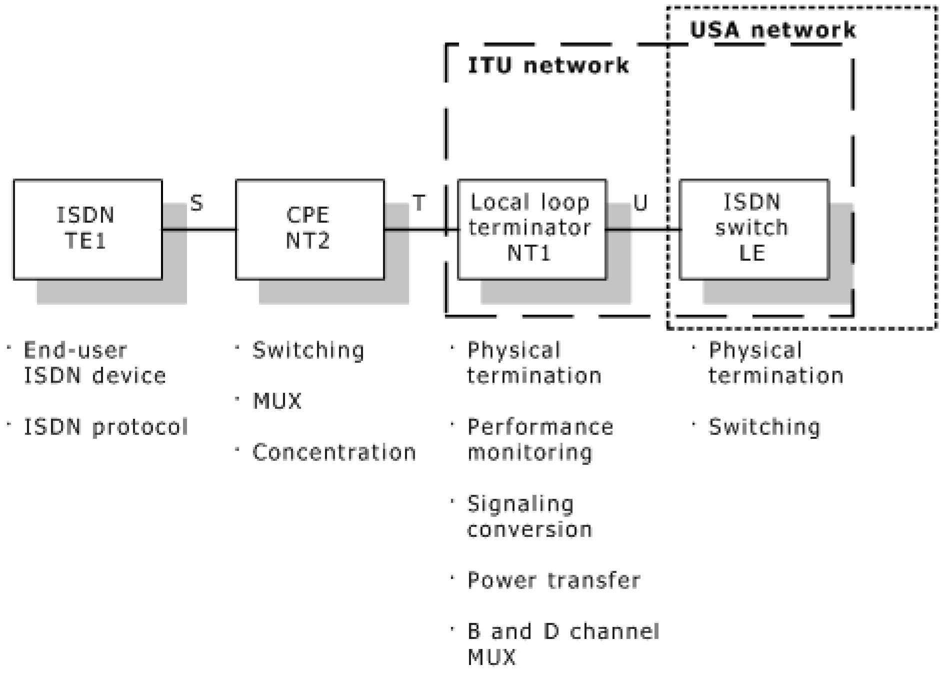
Category	**Description**
TE1	ISDN end-user terminating equipment, class 1, which terminates a single ISDN trunk.
NT2	Network terminating equipment, class 2. NT2 equipment can support more than one primary rate trunk, switch channels between different trunks, and take primary synchronization from the T1 or E1 network.
NT1	Physically terminates the local loop. In Europe, the NT1 equipment is the network in I.411-compliant installations. In the USA, NT1 is often referred to as the CSU, a separate device on the customer premises.
LE	Local exchange equipment.

Note: The International Telecommunications Union (ITU) document I.411 defines ISDN user-network interface reference configurations. In the USA, the definition is provided by a number of documents, including AT&T TR-41449 ISDN Primary Rate Interface Specification.

The interface between each category is called a point:

- The interface between TE1 and NT2 equipment is the S point.
- The interface between NT2 and NT1 equipment is the T point.
- The interface between NT1 and LE equipment is the U point.

The following illustration shows the component categories and associated reference points:



ISDN carriers

ISDN is transmitted over standard T1 and E1 carriers. T1 and E1 trunks are typically four-wire digital transmission links. T1 trunks are used mainly in the United States, Canada, Hong Kong, and Japan. E1 trunks are used in most of the rest of the world.

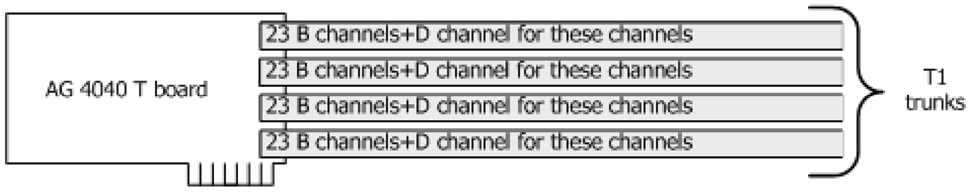
Data on a trunk is transmitted in channels. Each channel carries information digitized at 64000 bits per second (b/s).

Primary rate interface (PRI)

For primary rate ISDN, T1 trunks carry 24 channels. E1 trunks carry 32 channels. The channels are usually used as follows:

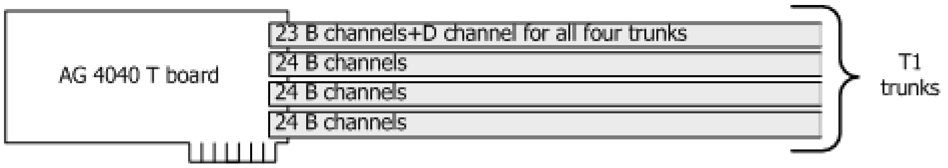
- On a T1 trunk, 23 of the 24 channels carry data: voice, audio, data and/or video signals. These channels are called bearer channels (B channels).
- On an E1 trunk, 30 of the 32 channels are B channels.
- On a T1 or E1 trunk, one channel carries signaling information for all B channels. This is called the D channel.

The following illustration shows a T1 trunk (standard configuration):


AG 4040 T board
23 B channels+D channel for these channels
23 B channels+D channel for these channels
23 B channels+D channel for these channels
23 B channels+D channel for these channels
T1 trunks

In setups with multiple T1 ISDN trunks, a non-facility associated signaling (NFAS) configuration is sometimes used. In this configuration, the D channel on one of the ISDN trunks carries signaling for all channels on several other trunks. This leaves channel 24 free on each of the other trunks to be used as another B channel.

The following illustration shows a sample NFAS configuration:



Note: NFAS configurations are supported only on T1 trunks. For more information about NFAS, see Non-facility associated signaling (NFAS).

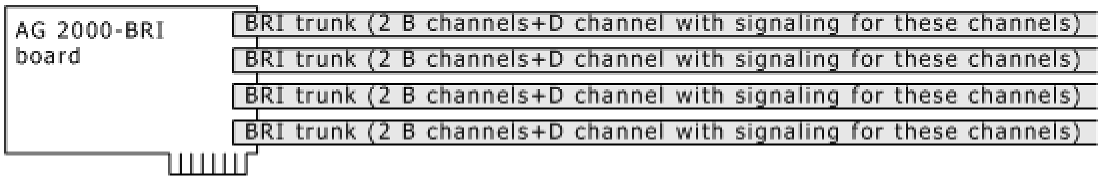
Basic rate interface (BRI)

ISDN is also transmitted over BRI trunks with four-wire digital transmission links. BRI trunks are used mainly in Europe and Asia and transmit data in 3 channels.

The three channels are usually used as follows:

- Two of the channels are B channels, carrying voice, audio, data and/or video signals at 64000 b/s.
- One of the channels is a D channel, carrying signaling information for the B channels at 16000 b/s.

The following illustration shows a BRI trunk (standard configuration):


AG 2000-BRI board
BRI trunk (2 B channels+D channel with signaling for these channels)
BRI trunk (2 B channels+D channel with signaling for these channels)
BRI trunk (2 B channels+D channel with signaling for these channels)
BRI trunk (2 B channels+D channel with signaling for these channels)

NMS ISDN software

NMS ISDN protocol software enables you to write Natural Access applications that communicate with T1, E1, or BRI trunks to perform voice processing functions and call control using ISDN common channel signaling (CCS) protocols.

NMS ISDN software is designed to use one or more NMS digital boards (such as the CG 6000C, AG 4000, or AG 2000-BRI board) as the physical interface to trunk lines. In addition to line interfaces, these boards also use powerful on-board digital signal processing (DSP) resources that can handle much of the call control and voice processing overhead.

This topic describes the:

- NMS ISDN product configurations
- NMS ISDN ACU configuration
- NMS ISDN LAPD configuration
- Reference points supported by NMS ISDN

NMS ISDN product configurations

You can use NMS ISDN software to access ISDN services in three ways:

- Configure the NMS ISDN software so an application can perform call control and other operations, using the Natural Call Control service.

  This NMS ISDN configuration is called the channelized configuration. For more information about this access method, see the *NMS ISDN for Natural Call Control Developer's Manual*.
- Access ISDN services at the ACU SAPI using the NMS ISDN Messaging API. An application can use the API to perform a wide range of Q.931 ISDN D channel functions.

  This NMS ISDN configuration is called the ACU configuration. This access method is discussed in this manual.
- Access ISDN services at the data link layer (layer 2) using the NMS ISDN Messaging API. An application can send and receive I-frame data in LAPD messages. This data typically consists of Q.931 messages.

  This NMS ISDN configuration is called the LAPD configuration. This access method is discussed in this manual.

Specify the configuration to use when initializing the ISDN protocol stack, as described in Initializing ISDN protocol stack instances.

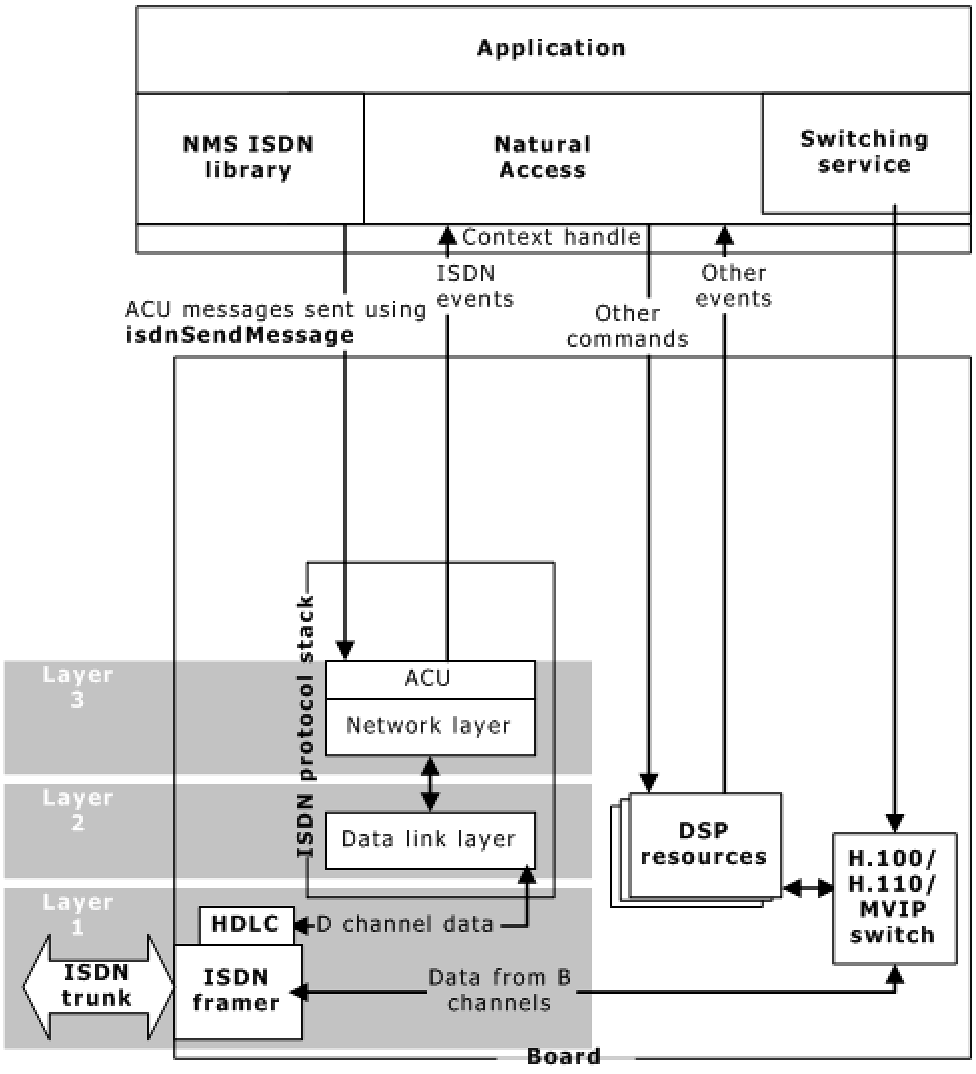
NMS ISDN ACU configuration

The NMS ISDN ACU configuration allows access to Q.931 (layer 3) call control, using the NMS ISDN Messaging API. The application can send and receive switch- and country-invariant D channel messages using this interface. Access at this level allows you direct control over D channel messages and greater control over the contents of these messages.

In the ACU configuration, one or more instances of the NMS ISDN protocol stack runs on the board, one for each D channel. The stack runs in ACU stack mode. In this mode, the protocol stack implements all ISDN layer 2 and layer 3 functionalities. The application uses the NMS ISDN Messaging API to command an entity in the stack called the ACU, which in turn commands the D channel through the lower ISDN layers. Events received by the stack from the D channel are placed in the same event queue as other Natural Access events, allowing the application to access ISDN events in the same way that other events are accessed.

B channel information is routed to the DSP resources through the board's H.100, H.110, or MVIP switch. The switch has certain default behavior, described in Making switch connections for NMS ISDN. The switch can also be controlled using the Natural Access Switching service.

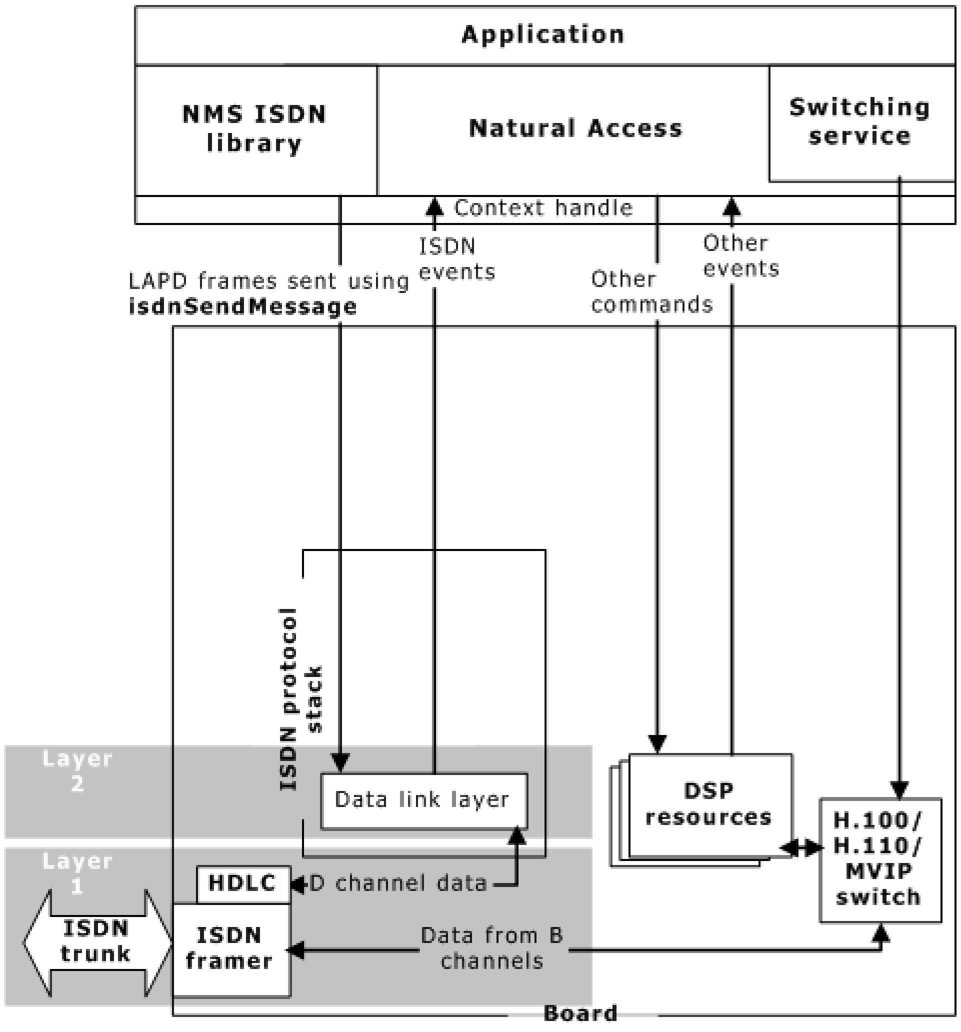
The following illustration shows the NMS ISDN application architecture (ACU configuration):



NMS ISDN LAPD configuration

Access at the data link layer is useful if an application must support a private data link protocol, or if the user wants to create a complete Q.931 protocol at the application level. At this level, the messages sent and received by the application constitute LAPD frames.

The NMS ISDN protocol stack runs in LAPD stack mode, as shown in the following illustration:



In this mode, the protocol stack implements ISDN layer 2 functionality. No ACU is present. Instead, the application uses the NMS ISDN Messaging API to send LAPD frames directly to the data link layer (layer 2).

Events coming from the data link layer are placed in the same event queue as other Natural Access events, allowing the user to access ISDN events in the same way that other events are accessed.

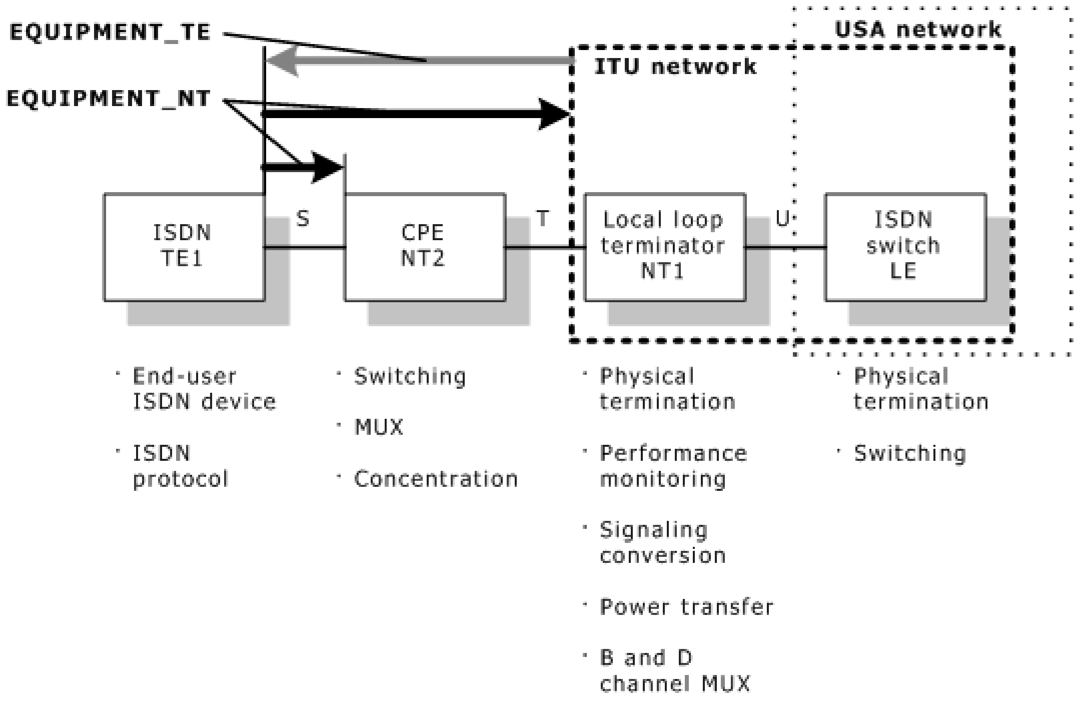
As in other configurations, B channel information is routed to the DSP resources through the board's H.100, H.110, or MVIP switch. The switch has certain default behavior, described in Making switch connections for NMS ISDN. Alternatively, the switch can be controlled using the Natural Access Switching service.

Reference points supported by NMS ISDN

NMS ISDN supports access across the S and the T reference points (as shown by the solid arrows on the upper left of the following illustration). At the ACU SAP, access to S/T is transparent.

You can configure an ISDN protocol stack to emulate the network by using the ***partner_equip*** field passed to **isdnStartProtocol**. See Function summary for more details on the parameters available to configure the ISDN protocol stack.

The following illustration shows the reference points supported by NMS ISDN:



NMS ISDN software components

ISDN is implemented differently around the world. For this reason, NMS provides versions of its ISDN software for different regions. Each package for a variant contains the software modules needed to allow a board to communicate on a T1, E1, or BRI trunk in one or more countries using that variant.

The NMS ISDN software package for a given region contains the following components that are briefly described in this topic:

- A *readme* file.
- NMS ISDN function libraries.
- Header files.
- Run modules containing the NMS ISDN protocol stack software and the NMS ISDN management software.
- Board keyword files.
- Demonstration programs and utilities, with their source code files and *makefiles*.

Other components, also included, are used only when the NMS ISDN protocol stack is running in channelized stack mode.

The channelized stack mode-specific components are:

- A trunk control program (TCP).
- Parameter files (*.pf* files) containing the complete set of NMS ISDN parameters. Some parameters in these files are set to values specific to a certain country or region.
- An *nccxisdn.par* file containing the subset of NMS ISDN parameters that can be changed, if desired. Two country-specific parameter files, *nccxadi**cty**.par* and

*nccstart***cty***.par*, contain parameters that should not be changed. These files are used when the protocol stack is in channelized configuration, using the NCC service.

For more information about the channelized stack mode, see the *NMS ISDN for Natural Call Control Developer's Manual*.

Readme file

The ASCII text file *readme_isdn.txt* contains release information that does not appear in other documentation. Consult this file to learn where the NMS ISDN software components are located after installation.

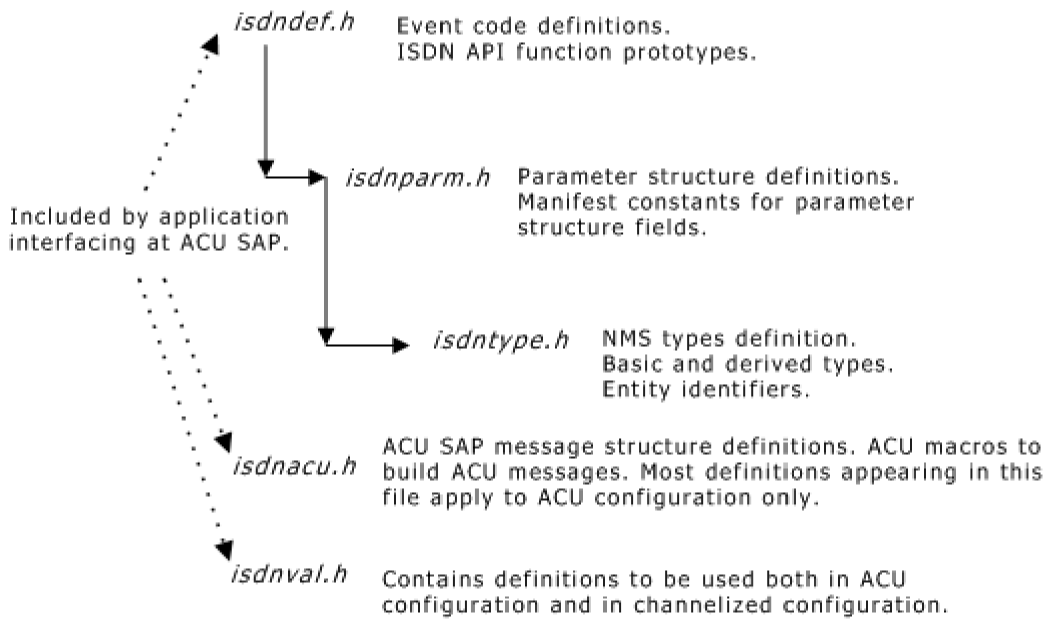
NMS ISDN function libraries

NMS ISDN function libraries run on the host computer. The application uses them to interact with ISDN protocol stacks running on a board and to communicate with the NCC service. These are dynamic-link libraries (DLL) under Windows, and are shared objects under UNIX. The libraries have different names under different operating systems:

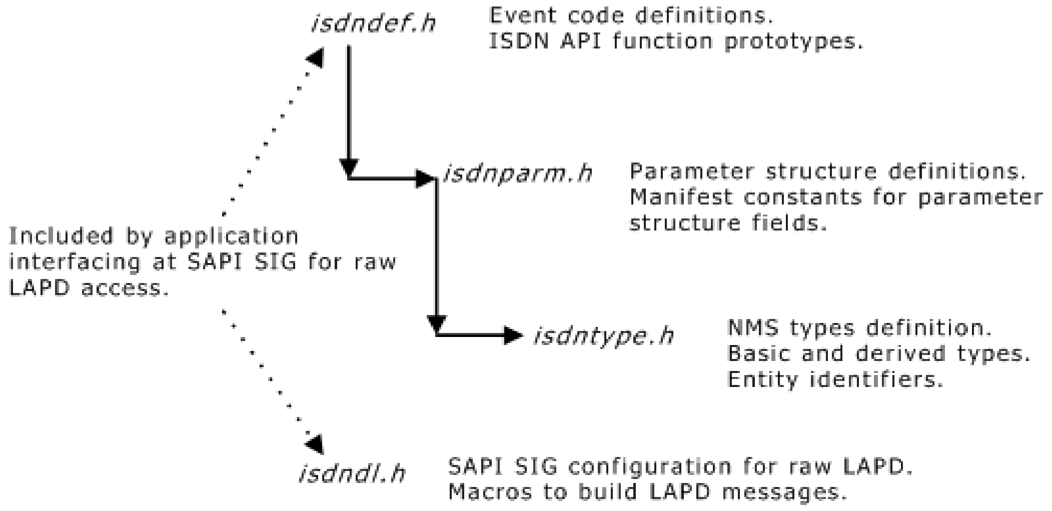
Operating system	**Natural Access library name(s)**
Windows	*isdnapi.lib*, *isdnapi.dll*, *nccisdn.lib*, *nccisdn.lib*, *imgtapi.lib*, *imgtapi.dll* **Note:** *nccisdn.lib, nccisdn.lib, imgtapi.lib,*and *imgtapi.dll* are not in use for layer 3.
UNIX	*libisdnapi.so*, *libnccisdn.so*, *libimgtapi.so* **Note:** *libnccisdn.so* and *libimgtapi.so* are not in use for layer 3.

Standard NMS ISDN header files, required by your applications to communicate with the ISDN protocol stack on the board, are also supplied.

The following illustration describes the include file structure used by an application using NMS ISDN in the ACU configuration:


isdndef.h
Event code definitions.
ISDN API function prototypes.
isdnparm.h
Parameter structure definitions.
Manifest constants for parameter structure fields.
Included by application interfacing at ACU SAP.
isdntype.h
NMS types definition.
Basic and derived types.
Entity identifiers.
isdnacu.h
ACU SAP message structure definitions. ACU macros to build ACU messages. Most definitions appearing in this file apply to ACU configuration only.
isdnval.h
Contains definitions to be used both in ACU configuration and in channelized configuration.

The following illustration describes the include file structure used by an application using NMS ISDN in the LAPD configuration:


isdndef.h
Event code definitions.
ISDN API function prototypes.
isdnparm.h
Parameter structure definitions.
Manifest constants for parameter structure fields.
isdntype.h
NMS types definition.
Basic and derived types.
Entity identifiers.
Included by application interfacing at SAPI SIG for raw LAPD access.
isdndl.h
SAPI SIG configuration for raw LAPD.
Macros to build LAPD messages.

Run modules

A run module contains the basic low-level software that a board requires to support ISDN. The module is transferred from the host into on-board memory when the board boots.

Different run modules are supplied for different configurations and are specific to the protocol variant and country. The module you use depends upon what board type you are using. For more information about run modules, see the *NMS ISDN Installation Manual*.

Board keyword files

Board keyword files contain information that determines how to set up your boards for use. These files also contain country-specific information, and define what trunks are assigned to which D channels.

Several example files are included, describing ISDN configurations for different boards. Use these files to create a file describing your hardware and software setup. For details, see the *NMS OAM System User's Manual*.

Demonstration programs and utilities

The following demonstration programs and utilities are included, with their source code files and makefiles:

Program	**Description**
isdndemo	Communicates with the ISDN stack in the ACU configuration and performs call control.
lapddemo	Illustrates establishing a LAPD data link on an ISDN trunk.
dectrace	Decodes and displays messages sent or received by the NMS ISDN protocol stack that were previously captured in a log file by the *oammon* utility.
itrace	Acts as a runtime filter for NMS ISDN messages from the stack being captured by the *oammon* utility.

Other components

In addition to the NMS ISDN software, you need the following components to build an NMS ISDN protocol application:

- One or more NMS T1, E1, or BRI trunk interface boards
- Natural Access

Warning: ⚠	NMS Communications obtains board-level approvals certificates for supported countries. Some countries require that you obtain system-level approvals before connecting a system to the public network. To learn what approvals you require, contact the appropriate regulatory authority in the target country.

Natural Access

Natural Access is a complete development environment for telephony applications. It provides a standard set of telephony functions grouped into logical services. Natural Access provides functions for telephony-related tasks such as call control, tone and DTMF tone generation and detection, and voice playing and recording.

Natural Access includes a service that controls switching on H.100, H.110, or MVIP-compliant devices. You can use this service to make or break connections, send patterns, and sample data. Alternatively, you can use the *swish* standalone utility to control switching interactively or in a batch mode.

For information about installing and using Natural Access, see the Natural Access documentation.

Developing an NMS ISDN application

Perform the following steps to create an NMS ISDN application:

Step	Action	Refer to...
1	Install digital trunk interface boards in a system, and any other boards you will need in your application.	The board installation manuals
2	Install Natural Access.	*Installing Natural Access*
3	Install the NMS ISDN software for each country or region where your application will be used.	*NMS ISDN Installation Manual*
4	Edit the system configuration file and the board keyword files so they describe all boards in the system.	The *NMS ISDN Installation Manual*, the installation manuals for your boards, and the *NMS OAM System User's Manual*
5	Test the hardware installation.	The board installation manuals
6	Write the application.	This manual and the Natural Access documentation set

3. Messaging API programming model

Natural Access environment

NMS ISDN applications are built primarily on the Natural Access platform. This topic provides background information about Natural Access and summarizes the main elements of the Natural Access environment. You must have Natural Access installed on your system to build applications using NMS ISDN.

For detailed information about Natural Access, see the *Natural Access Developer's Reference Manual*.

Natural Access components

Natural Access telephony functions are divided into groups of logically related functions called services.

A context organizes services and accompanying resources around a single process. A context usually represents an application instance controlling a single telephone call. A service can be opened only once on a context.

An event queue is the communication path from a service to an application. A service generates events indicating certain conditions or state changes. An application retrieves the events from the event queue.

Natural Access programming model

Natural Access employs an asynchronous programming model to take advantage of concurrent processing. When called, most functions return immediately indicating the operation was initiated. The application can then call other functions while Natural Access is processing the command.

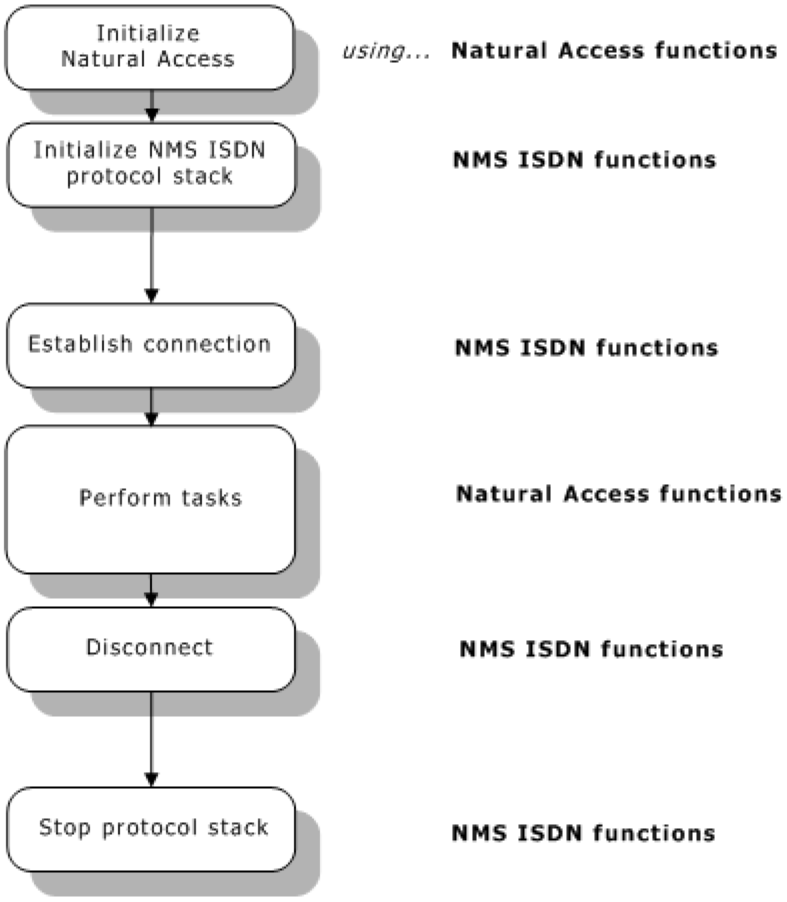
Asynchronous functions return SUCCESS if the function is successfully initiated. The execution result arrives later in an event. Asynchronous functions that return a non-zero value were never initiated. Therefore, no subsequent events are generated. If an asynchronous function fails after being initiated, Natural Access delivers a DONE event to the application with an error code in the event value field.

Synchronous functions indicate completion by sending a return value. The return value is either SUCCESS or an error code.

Refer to the Function summary for a list of all NMS ISDN functions and whether they are synchronous or asynchronous.

NMS ISDN Messaging API application overview

An NMS ISDN Messaging API application typically performs the following tasks:



The following table provides a summary of the tasks illustrated:

In this phase...	**The application...**
Initialize Natural Access	Makes CT bus switch connections to route D channel data to the HDLC controller, and to route B channel information to DSP resources (if necessary). Initializes Natural Access services, and creates one context for each B channel and D channel. Starts the *nocc* (no call control) trunk control program (TCP) on each B channel context.
Initialize NMS ISDN protocol stack	Calls **isdnStartProtocol** to start up an ISDN protocol stack instance on each D channel context. This function also determines whether the stack runs in LAPD, ACU, or channelized stack mode.
Establish connection	Uses **isdnSendMessage** to send ACU or LAPD messages to the stack to establish a connection.

In this phase...	The application...
Perform tasks	Uses functions from Natural Access or from other services to play or record voice, generate or detect DTMF tones, send and receive faxes, and other tasks.
Disconnect	Uses **isdnSendMessage** to send ACU or LAPD messages to the stack to terminate the connection.
Stop protocol stack	Calls **isdnStopProtocol** to stop the ISDN protocol stack.

Initializing boards

Before you can run an NMS ISDN application, you must initialize and load DSP files, trunk control programs (TCPs), and protocol stack runfiles to your board(s). The items to load to the boards are specified in board keyword files. To load the components to your boards, run *oamsys*, the board initialization and monitoring utility. To learn how to create a configuration file for your setup, see the *NMS ISDN Installation Manual*, the installation manual for your board, and the *NMS OAM System User's Manual*.

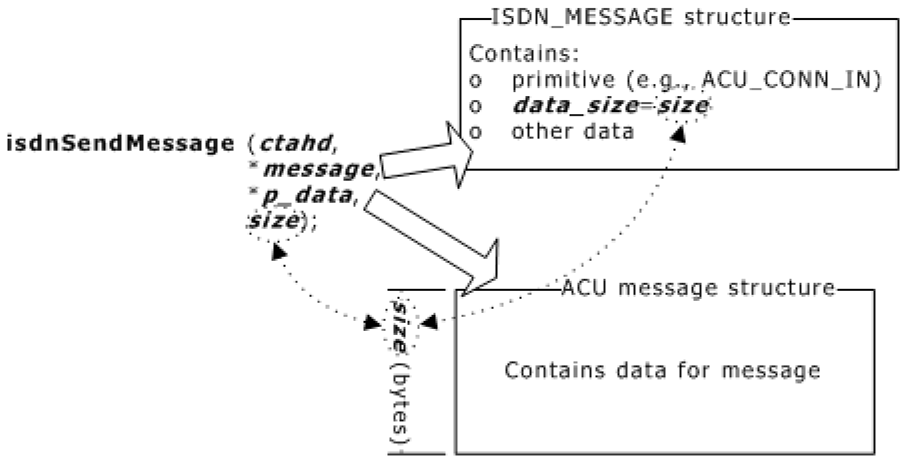
Sending ISDN messages to the stack

To send a message to the NMS ISDN protocol stack, the application builds two structures:

Structure	Description
ISDN_MESSAGE	In this structure, the application specifies the message to be sent, using one of the message primitives. For more information, refer to Overview of message primitives. The message primitive appears in the code field in this structure. For details about ISDN_MESSAGE, see ISDN_MESSAGE structure.
A message structure (optional).	For messages that require additional data, a message structure (containing the data) is sent. The data differs for each message type. For details on each message type, see Overview of message primitives.

The application then calls **isdnSendMessage** in the NMS ISDN library. The ***ctahd*** argument in the function call specifies the context of the trunk on which the call is taking place. ***message*** is a pointer to ISDN_MESSAGE. ***pdata*** is a pointer to the associated message structure (if any). ***size*** is the size of the message structure (if any).

The following illustration shows the content and meaning of each of the arguments sent in **isdnSendMessage**:



ISDN_MESSAGE is defined as follows:

```
typedef struct ISDN_MESSAGE
{
  nai_t nai;        /* Network access interface index                  */
  ent_id_t from_ent;/* Message source                                  */
  ent_id_t to_ent;  /* Message destination                             */
  sapi_t to_sapi;   /* Destination Service Access Point                */
union {
  add_t conn_id;    /* Connection identifier for the ACU layer         */
  add_t crv;        /* Call Reference value for the NS layer. Not used.  */
  add_t ces;        /* Connection Endpoint suffix (DL later upper half)  */
  add_t tei;        /* Terminal Endpoint ID (DL layer lower half). Not used.*/
  add_t chani;      /* Physical layer channel identifier. Not used.      */
} add;
  code_t code;      /* Primitive code unique only between two entities   */
  WORD inf0;        /* Information location 0                          */
  WORD inf1;        /* Information location 1                          */
  WORD inf2;        /* Information location 2                          */
  WORD inf3;        /* Information location 3                          */
  WORD inf4;        /* Information location 4                          */
  WORD data_size;   /* Size of data to follow                          */
  WORD nfas_group;  /* NFAS group number                               */
  DWORD userid;     /* User ID                                         */
} ISDN_MESSAGE;
```

Building ACU message structures

The NMS ISDN protocol stack must be in ACU stack mode to send ACU messages. To learn how to place the stack in this mode, see Initializing ISDN protocol stack instances.

Before sending an ACU message, the application creates a message structure containing the ACU message data. When this structure (and its substructures) reaches the ISDN protocol stack, the stack rearranges the data into several Q.931 information elements (IEs), builds a complete Q.931 message with the IEs, and sends it to the network.

The application uses macros to assign values to the fields in the message structure. Each macro corresponds to a field in the structure. The application uses these macros instead of accessing fields directly, because the fields do not correspond to actual C language structure fields and need some simple addressing computation to be reached. These macros are defined in *isdnacu.h*. See ACU primitives summary for more information about message primitives and macros.

The following sample code shows how to use macros to assign values to fields in an ACU message structure:

```
void build_cc_conn_rs(char *buffer, int *len, char ts)
{
    struct acu_conn_rs_args *p_data;
    p_data = (struct acu_conn_rs_args *)buffer;
    memset(p_data, OFF, ISDN_BUFFER_DATA_LGTH);
    /* Fill in two fields using macros*/
    Acu_conn_rs_data_chani = ts;             /* T1/E1 time-slot    */
    Acu_conn_rs_data_chani_nb = 1;           /* Only one B channel */
    /* User to user information could go here */
    *len = sizeof( struct acu_conn_rs_args );
    return;
}
```

Certain pointer and size macros must be called in a specific sequence, so that the corresponding IEs are filled out in order. For details, see ACU primitives summary.

The application also builds an ISDN_MESSAGE structure specifying the message primitive and other data, as follows:

ISDN_MESSAGE field	**Value**
nai	Specifies the network access identifier (NAI) of the destination trunk.
from_ent	ENT_APPLI Indicates that the message is sent from the application to the stack.
to_ent	ENT_CC Indicates that the message is sent to the call control layer (ACU).
to_sapi	ACU_SAPI Specifies the unique service access point identifier (SAPI) associated with the ACU service access point (SAP).
conn_id	Specifies the connection ID of the call that the message concerns.
code	Specifies the primitive of the ACU message (for example ACU_CONN_RQ).
inf0, inf1, inf2, inf3, inf4	Reserved for internal data relay.
data_size	Specifies the size of the message buffer containing data associated with message.
nfas_group	Specifies the NFAS group number for this network access identifier (NAI). Used only if duplicate NAI values are defined.
userid	Not used.

An application can specify custom information elements containing raw Q.931 data, using transparent IEs. For details, see Overview of Q.931 data.

Sending LAPD messages

The NMS ISDN stack must be in LAPD stack mode to send LAPD messages.

To send a LAPD message, the application builds an ISDN_MESSAGE structure specifying the message primitive and other data, as follows:

ISDN_MESSAGE field	Value
nai	Specifies the network access identifier (NAI) of the destination trunk.
from_ent	ENT_APPLI Indicates that the message is sent from the application to the stack.
to_ent	ENT_DL_D Indicates that the message is sent to the data link layer.
to_sapi	DL_SAPI_SIG
conn_id	1
code	Specifies the primitive of the LAPD message (for example DL_EST_RQ).
inf0, inf1, inf2, inf3, inf4	Reserved for internal data relay.
data_size	Specifies the size of the message buffer containing data associated with the message, if any.
nfas_group	Specifies the NFAS group number for this network access identifier (NAI). Used only if duplicate NAI values are defined.
userid	Not used.

If necessary, the application also builds a message buffer containing data to send.

Receiving events and ISDN protocol stack messages

All messages and events are returned to the application through the standard Natural Access event handling mechanism. The events returned can be standard Natural Access events, events sent by an ISDN protocol stack instance, or events specific to any Natural Access extensions. They arrive in the form of the standard event data structure:

```
typedef struct CTA_EVENT
{
    DWORD    id;              /* Event code (and source service ID)     */
    CTAHD    ctahd;           /* Context handle                         */
    DWORD    timestamp;       /* Timestamp                              */
    DWORD    userid;          /* User ID (defined by ctaCreateContext)  */
    DWORD    size;            /* Size of buffer if buffer != NULL       */
    void     *buffer;         /* Buffer pointer                         */
    DWORD    value;           /* Event status or event-specific data    */
    DWORD    objHd;           /* Service client side object handle      */
} CTA_EVENT;
```

The CTA_EVENT structure informs the application which event occurred on which context, and provides additional information specific to the event. The event's prefix relates the event to a specific NMS library of functions. The following table shows some of the available prefixes:

This prefix...	Indicates...
CTAEVN	A Natural Access event.
NCCEVN	An NCC service event.
ADIEVN	An ADI service event.
ISDNEVN	An NMS ISDN event.
OAMEVN	An OAM service event.
HSWEVN	An OAM Hot Swap event.
CLKEVN	An OAM clock management event.

The CTA_EVENT structure fields are used as follows:

Field	Description
id	Contains an event code defined in the library header file. All NMS ISDN events are prefixed with ISDNEVN_ (for example, ISDNEVN_SOMETHING_HAPPENED).
ctahd	Contains the context handle returned from **ctaCreateContext**.
timestamp	Contains the time when the event was created in milliseconds since midnight, January 1, 1970, modulo 49 days. The resolution for board events is 10 milliseconds.

Field	Description
userid	Contains the user-supplied ID. This field is unaltered by Natural Access and facilitates asynchronous programming. Its purpose is to correlate a context with an application object and context when events occur.
size	Indicates the size (bytes) of the area pointed to by buffer. If the buffer is NULL, this field can hold an event-specific value.
buffer	Points to data returned with the event. The field contains an application process address. The event's size field contains the actual size of the buffer.
value	Provides a reason code or an error code. This is an event-specific value.
objHd	Contains the call handle, if the event concerns a particular call. If the event concerns the line and not a particular call, objHd is ctaWaitEvent.

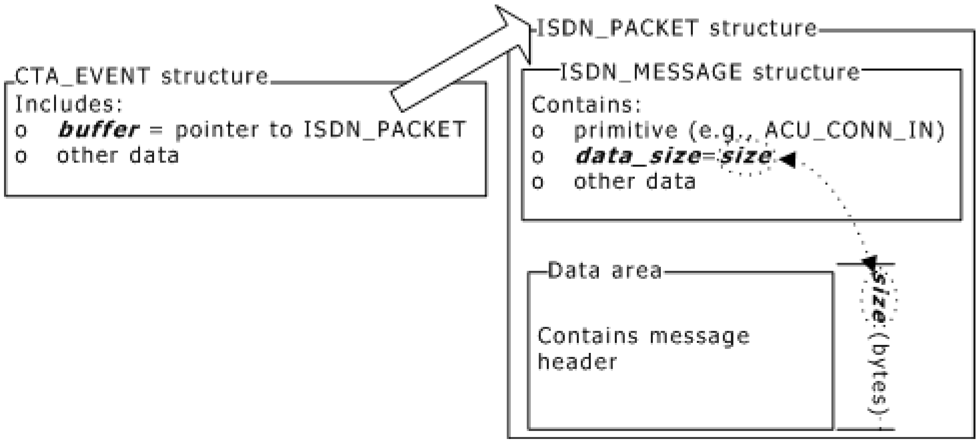
Receiving messages from the NMS ISDN protocol stack

When an NMS ISDN protocol stack message is received, an ISDNEVN_RCV_MESSAGE event occurs.

The ***buffer*** field in the CTA_EVENT structure is a pointer to an ISDN_PACKET structure. This structure contains:

- An ISDN_MESSAGE structure that contains the message, and other data.
- A data area that contains the message header.

The following illustration shows the structure of this message packet:



After receiving and processing the data within the CTA_EVENT event buffer, the application must use **isdnReleaseBuffer** to free the buffer as quickly as possible. Otherwise, the ISDN interface times out and stops passing events to the application.

The following sample code illustrates how the application processes an incoming ISDN event from the on-board protocol stack through the Natural Access event mechanism:

```
cta_ret = ctaWaitEvent(ctaqueuehd, &cta_event, CTA_WAIT_FOREVER);
if (cta_ret != SUCCESS)
   {
   printf("isdnproc: ctaWaitEvent failure %d %x\n",
          cta_ret, cta_ret);
   continue;
   }
/* We have an event -- switch on the type                  */
switch (cta_event.id)
   {
          .
          .
          .
   case ISDNEVN_RCV_MESSAGE:
/*      Pick up the ISDN_PACKET structure                  */

        isdnpkt = (ISDN_PACKET *)adi_event.buffer;
        isdnmsg = (ISDN_MESSAGE *)&isdnpkt->message;

/*      Extract information from the ISDN incoming packet  */

        message = isdnmsg->code;
        sender = isdnmsg->from_ent;
        recipient = isdnmsg->to_ent;
        CRN = isdnmsg->add.conn_id;
}
```

The fields in the received ISDN_MESSAGE structure contain the following information:

ISDN_MESSAGE field	Value in ACU stack mode...	Value in LAPD stack mode...
nai	Specifies the network access identifier (NAI) of the trunk that the message concerns.	Specifies the network access identifier (NAI) of the trunk that the message concerns.
from_ent	ENT_CC. Indicates that the message was sent from call control layer (ACU).	ENT_DL_D. Indicates that the message was sent from data link layer.
to_ent	ENT_APPLI. Indicates that the message was sent to an application.	ENT_APPLI. Indicates that the message was sent to an application.
to_sapi	ACU_SAPI	DL_SAPI_SIG
conn_id	Contains the connection ID of the call that the message concerns.	1
code	Specifies the primitive of the message, for example, ACU_CONN_IN.	Specifies the primitive of the message, for example, DL_DA_IN.
inf0, inf1, inf2, inf3, inf4	Reserved for internal data relay.	Reserved for internal data relay.

ISDN_MESSAGE field	Value in ACU stack mode...	Value in LAPD stack mode...
data_size	Specifies the size of the message buffer containing data associated with the event, if any.	Specifies the size of the message buffer containing data associated with the event, if any.
nfas_group	Specifies the NFAS group number for the network access identifier (NAI). Used only if duplicate NAI values are defined.	Specifies the NFAS group number for the network access identifier (NAI). Used only if duplicate NAI values are defined.
userid	Not used.	Not used.

4. Initializing a Messaging API application

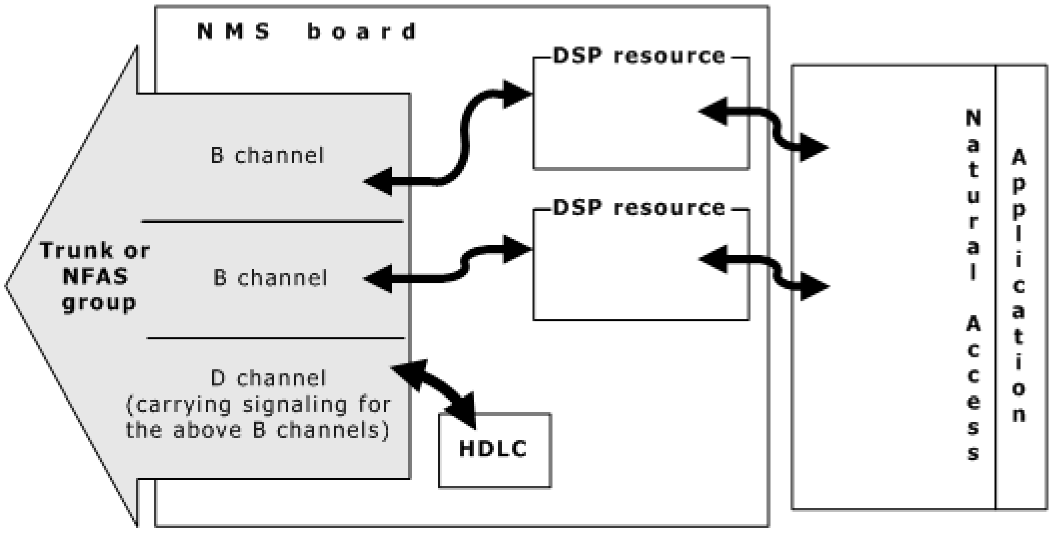
Initialization tasks

An NMS ISDN application performs the following initialization tasks:

- Task 1: Route channel data to on-board resources
- Task 2: Create contexts for channels
- Task 3: Call isdnStartProtocol
- Task 4: Starting NOCC TCPs on B channel contexts

Task 1: Route channel data to on-board resources

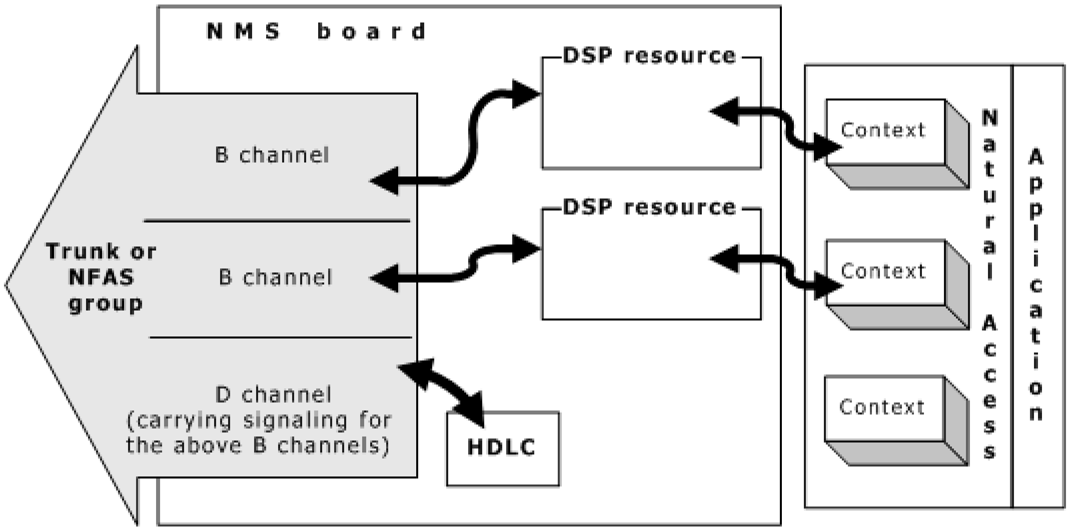
If necessary, the application makes switch connections to route D channel data to the HDLC controller, and to route B channel information to DSP resources. Certain default connections are made automatically if H.100, H.110, or MVIP switching is not enabled. The following illustration shows routing channel data to on-board resources:



For more information, see Making switch connections for NMS ISDN.

Task 2: Create contexts for channels

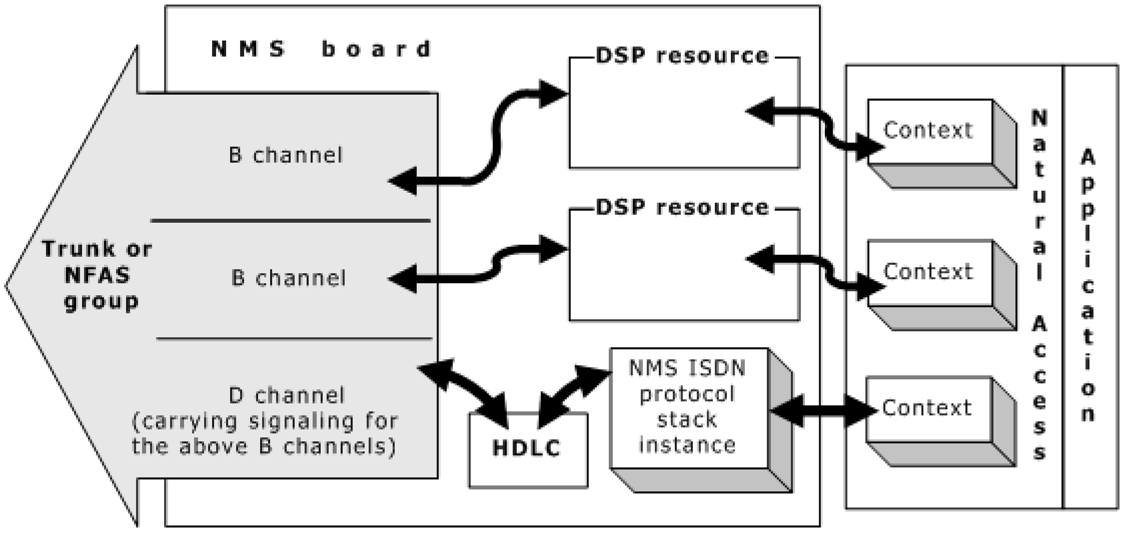
The application initializes Natural Access and creates a separate context for each B channel and D channel with which it will interact. The following illustration shows creating contexts for channels:



For more information, see Initializing Natural Access.

Task 3: Call isdnStartProtocol

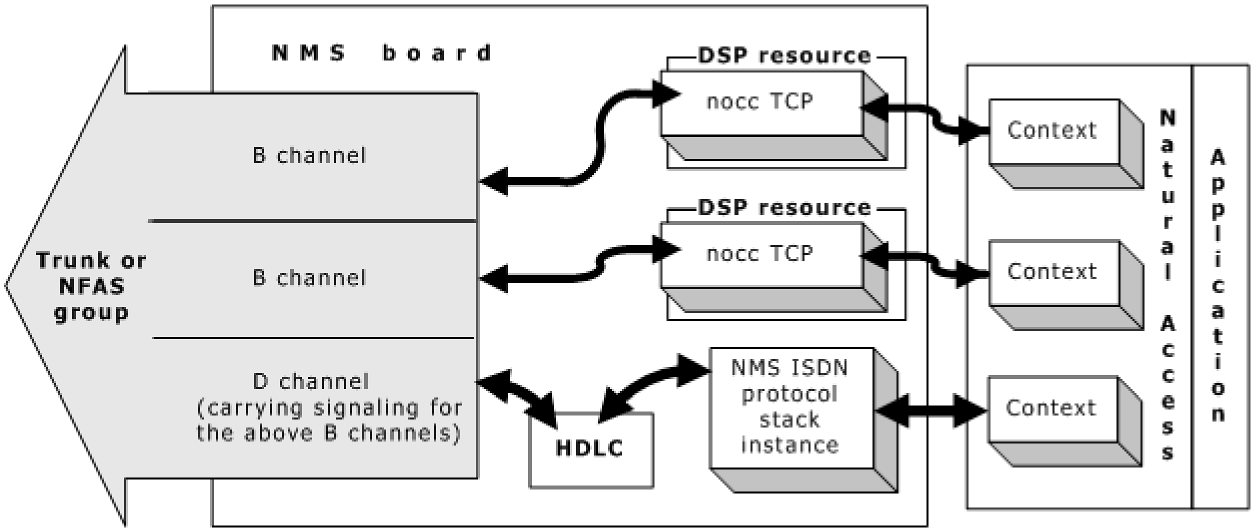
The application uses **isdnStartProtocol** to initialize ISDN protocol stack instances on each D channel context. This function starts up an ISDN protocol stack instance on the D channel context in ACU or LAPD stack mode. In the function call, the trunk is specified using its network access identifier (NAI) and, if duplicate NAI values are defined, the NFAS group number to which this NAI belongs. The following illustration shows calling **isdnStartProtocol**:



For more information, see Accessing D channels.

Task 4: Start NOCC TCPs on B channel contexts

The application starts a special no call control trunk control program (nocc TCP) on each B channel context. This program puts the context in a state where voice or media functions can be used without call control. The following illustration shows starting NOCC TCPs on B channel contexts:



For more information, see Starting the NOCC TCP.

Making switch connections for NMS ISDN

To enable an application access to ISDN channels, several switch connections must be made. In the board keyword file, if Clocking.HBus.ClockMode is set to StandAlone, these settings are automatically made when the board boots.

If Clocking.HBus.ClockMode is set to any other value, the application must set these values using the NMS Switching service or *swish* utility. The application must make the following connections:

- On each trunk, streams and timeslots carrying D channel information to and from the HDLC controller (if any) must be connected to streams and timeslots accessible by the ISDN protocol stack. These connections must be full duplex. They must be made before the ISDN protocol stack is initialized. (Initialization is described in Initializing ISDN protocol stack instances.)
- The streams and timeslots carrying voice information to and from the trunk must be connected to the streams and timeslots carrying voice information to and from the DSP resources. The connections must be full duplex.

The connections differ depending upon whether the NetworkInterface.T1E1.SignalingType keyword is set to PRI or RAW. Specifically, if NetworkInterface.T1E1.SignalingType=RAW, no connections are made between the HDLC controller and signaling streams. This setting is for trunks that are included in NFAS groups and do not have a D channel in operation. For more information about these keywords and about NFAS, see the *NMS ISDN Installation Manual*.

Connections are listed in MVIP-95 nomenclature unless otherwise specified.

Note: On CG boards, the framer signaling is hard wired to the HDLCs. Any attempt to switch framer signaling to the HDLCs will fail.

The following table shows the default connections. The exact settings depend upon the NetworkInterface.T1E1.SignalingType keyword setting. The connections are full duplex.

Board	Setting	Default connections
Four-trunk T1 board (for example, CG 6060)	PRI	Trunk 1: 0:0..22 => 65:0..22 64:0..22 => 1:0..22 Trunk 2: 4:0..22 => 65:24..46 64:24..46 => 5:0..22 Trunk 3: 8:0..22 => 65:48..70 64:48..70 => 9:0..22 Trunk 4: 12:0..22 => 65:72..94 64:72..94 => 13:0..22
Four trunk T1 board (for example, CG 6060)	RAW	Trunk 1: 0:0..23 => 65:0..23 64:0..23 => 1:0..23 Trunk 2: 4:0..23 => 65:24..47 64:24..47 => 5:0..23 Trunk 3: 8:0..23 => 65:48..71 64:48..71 => 9:0..23 Trunk 4: 12:0..23 => 65:72..95 64:72..95 => 13:0..23
Four trunk E1 board (for example, CG 6060)	PRI	Trunk 1: 0:0..29 => 65:0..29 64:0..29 => 1:0..29 Trunk 2: 4:0..29 => 65:30..59 64:30..59 => 5:0..29 Trunk 3: 8:0..29 => 65:60..89 64:60..89 => 9:0..29 Trunk 4: 12:0..29 => 65:90..119 64:90..119 => 13:0..29
Four trunk E1 board (for example, CG 6060)	RAW	Trunk 1: 0:0..30 => 65:0..30 64:0..30 => 1:0..30 Trunk 2: 4:0..30 => 65:31...61 64:31..61 => 5:0..30 Trunk 3: 8:0..30 => 65:62..92 64:62..92 => 9:0..30 Trunk 4: 12:0..30 => 65:93..123 64:93..123 => 13:0..30

Initializing Natural Access

To begin operations, the application performs the following steps:

Step	Action
1	Initializes Natural Access services (including the ISDN service) with **ctaInitialize**.
2	Creates one or more event queues with **ctaCreateQueue**. Each function call creates a queue and returns a handle. Make sure at least one queue is attached to the ADI service manager.
3	Creates one or more contexts with **ctaCreateContext**. Each call creates a context and returns a context handle.
4	Opens services (initialized in Step 1) on the contexts using **ctaOpenServices.**

For more information about initializing Natural Access, see the *Natural Access Developer's Reference Manual*.

Specifying B channel contexts

Create a separate context for each B channel your application interacts with. To open contexts under Natural Access, use **ctaCreateContext** followed by **ctaOpenServices**. In each call to **ctaOpenServices**, specify the following values:

Value	For B channel contexts, set to...
stream	The voice stream or streams for the on-board DSPs. These streams are:

Board	DSP resource stream
AG 2000-BRI	MVIP 95: Local stream 0
All other NMS digital trunk interface boards	MVIP 95: Local stream 16

timeslot	A base timeslot in ***stream***.
mode	ADI_VOICE_DUPLEX. This mode allows voice (inband) transmission and reception. Do not use modes involving signaling that are defined in the ADI documentation, since ISDN signaling is not carried in the B channels.

Under Natural Access, these values are included in the **ctaOpenServices**.

Specifying D channel contexts

The application must also create a separate context for each D channel with which it will interact. Four-digital trunk boards can each support up to four separate D channels. Two-digital trunk boards can each support up to two separate D channels.

To open D channel contexts under Natural Access, use **ctaCreateContext** followed by **ctaOpenServices**. In each call to **ctaOpenServices**, set ***stream***, ***timeslot*** and ***mode*** to 0, since no DSP processing resources are needed to control the D channel data stream.

Under Natural Access, these values are included in the **ctaOpenServices**.

Accessing D channels

Once one or more contexts are created for the D channels, initialize a separate ISDN protocol stack instance for each context and associate a specific D channel with the context. When this is done, a D channel is ready to send and receive messages.

Network access identifiers (NAIs)

A trunk is referred to by its network access identifier (NAI). When you initialize an ISDN protocol stack instance for a context (using **isdnStartProtocol**), specify the NAI of the trunk to associate with the context. If duplicate NAI values are defined, specify the NFAS group number. From then on, the application can communicate with the D channel on that trunk through the context handle.

For example, when an event is received, the context handle indicates the trunk on which the event occurred. Different board types support different numbers of D channels and provide different NAI default values. The following table shows what each board type supports:

Board type	**Number of D channels**	**NAI default values**
Four-trunk board	As many as 4	0 through 3
Two-trunk board	As many as 2	0 through 1

Initializing ISDN protocol stack instances

Use **isdnStartProtocol** to initialize NMS ISDN protocol stack instances.

Note: Once you reference the context in an **isdnStartProtocol** function call, do not reference that context in any other function call except **isdnStopProtocol** (to stop the stack).

To start up an NMS ISDN protocol stack instance for ISDN ACU call control or LAPD, set the ***protocol***, ***partner_equip***, and ***parms*** arguments as follows:

protocol

Set the ***protocol*** argument to:

Structure	Mode
ISDN_PROTOCOL_Q931CC	ACU stack mode
ISDN_PROTOCOL_LAPD	LAPD stack mode

partner_equip

partner_equip indicates the type of equipment connected to the board.

If the board is...	And NMS ISDN is to run in...	Set partner_equip to...
Connected to network	ACU stack mode	EQUIPMENT_NT
Connected to network	LAPD stack mode	EQUIPMENT_DCE
Acting as network	ACU stack mode	EQUIPMENT_TE
Acting as network	LAPD stack mode	EQUIPMENT_DTE

parms

The ***parms*** argument is a pointer to a parameter structure to configure the protocol stack. If the application needs to change a parameter, pass a pointer to one of the following structures in this call:

Structure	Mode
ISDN_PROTOCOL_PARMS_Q931CC	ACU stack mode
ISDN_PROTOCOL_PARMS_LAPD	LAPD stack mode

If the application does not need to modify parameters, pass NULL to accept the default settings. For ACU, all of the services are supported by default.

The ISDNEVN_START_PROTOCOL event contains the completion status of the start request. If the ISDN protocol stack instance starts successfully, the value field in this event contains SUCCESS. Otherwise, another value appears here.

See **isdnStartProtocol** for more information.

Starting the NOCC TCP

Once all ISDN protocol stack instances are created, start a NOCC TCP on each B channel context. This TCP puts the context in a state where voice or media functions can be used without call control.

Call **nccStartProtocol** for each B channel context, specifying NOCC in the ***protname*** argument.

When **nccStartProtocol** is called, NCCEVN_STARTPROTOCOL_DONE is returned. If the NOCC TCP is started successfully, the event value field contains CTA_REASON_FINISHED. Otherwise, the value field contains another reason code.

Stopping an ISDN protocol stack instance

Applications call **isdnStopProtocol** to stop an ISDN protocol stack instance. This function shuts down the ISDN protocol stack instance, and releases all on-board resources and buffers formerly used by the stack instance.

The ISDNEVN_STOP_PROTOCOL event returns the completion status of the stop request. If the stack instance stopped successfully, the value field in this event contains SUCCESS. Otherwise, the event returns another reason code.

5. Call control (ACU configuration)

Connection IDs

The connection ID is a handle to a call on a B channel. It is used to identify the call in all communications between the ACU and the application. When an incoming call arrives, the protocol stack assigns it a connection ID. When the application places a call, it assigns a connection ID to the call. When a call is disconnected, the connection ID is freed. The ID can then be assigned to a new call by the protocol stack or the application.

A different set of connection IDs is available for each D channel. Thus a call is identified both by its D channel context and by its connection ID.

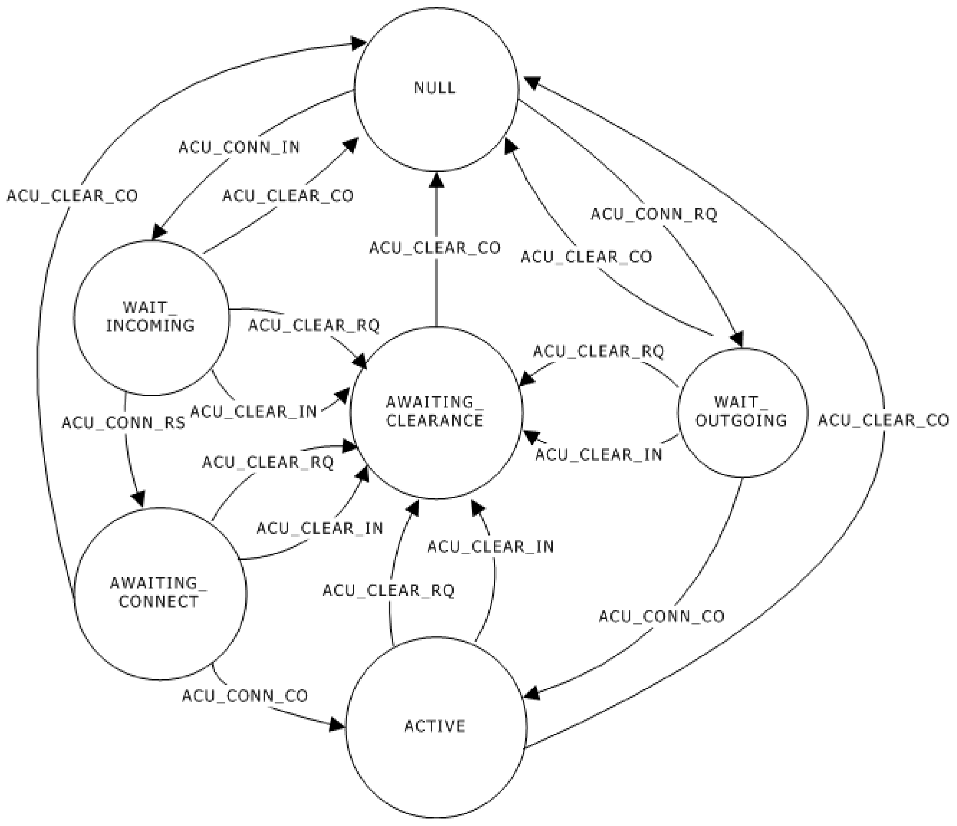
The range of connection IDs available for a trunk is between 0 and ACU_MX_CALLS (defined in *isdnacu.h*). In an NFAS group containing multiple trunks, there are ACU_MX_CALLS connection IDs for each NAI. The connection ID and NAI together identify a particular call.

The connection ID assigned to an incoming call by the protocol stack is the highest available unused value. For example, if 60 connection IDs are available for a trunk, and connection IDs 59 and 58 are already allocated to calls, the protocol stack assigns ID 57 to the next call.

To reduce the chance of collision, assign connection IDs beginning with 0 when placing outgoing calls. The application must also keep track of which connection IDs are in use and which are available.

ISDN call control state machine

The following illustration shows the NMS ISDN call control state machine:



Receiving inbound calls

When a call SETUP message is detected on the trunk, an ACU_CONN_IN is sent to the application from sender ENT_CC. Receipt of ACU_CONN_IN triggers a state transition from the NULL state to the WAIT_INCOMING state.

The application decodes the message and obtains the calling number and called number. The application then decides to accept or reject the call based on the value of the called number.

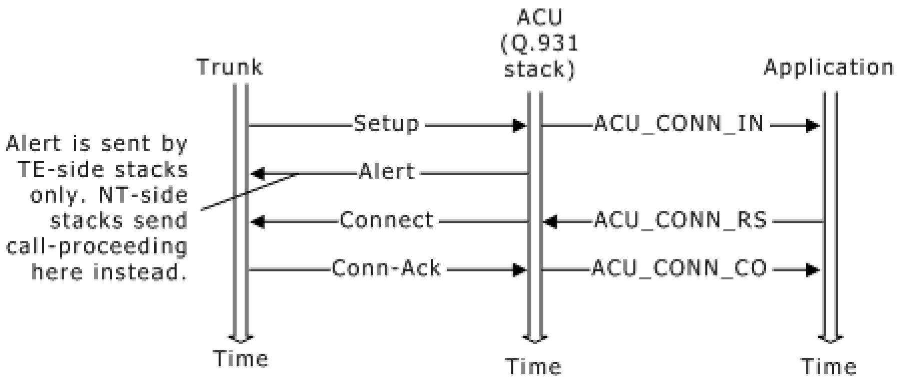
Accepting the call

To accept the call, the application builds a connect response message (ACU_CONN_RS) and sends it to the ISDN protocol stack. The event ACU_CONN_CO confirms that the call has been established. The stack switches to its ACTIVE state, where the application processing takes place.

Note: Applications with DSP resources that are not connected to a B channel connect calls when they receive connection indications.

The following illustration shows the default sequence of messages sent between the ACU and the application for an accepted inbound call. The actual messages sent back and forth in response to an incoming call may differ from this example depending upon the settings of the bits in the in_calls_behaviour substructure.

This substructure is referenced in the ISDN_PROTOCOL_PARMS structure passed to **isdnStartProtocol**. For details, see ISDN_PROTOCOL_PARMS_Q931CC parameters and ISDN_PROTOCOL_PARMS_LAPD parameters.



Incoming calls with overlap receiving mode enabled

If overlap receiving mode is enabled, when a call arrives, the ACU sends an ACU_CONN_IN message to the application even if the called number, the calling number, or both are not complete. The ACU then sends any additional incoming digits in ACU_DIGIT_IN messages.

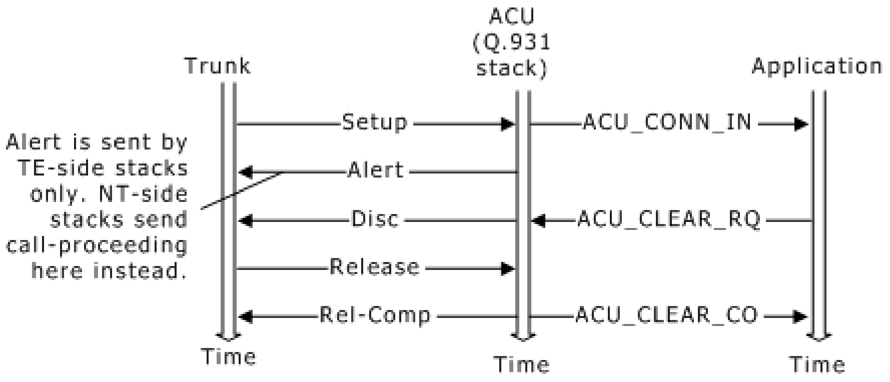
To enable this mode, set the CC_TRANSPARENT_OVERLAP_RCV bit in in_calls_behaviour. (This substructure is referenced in the ISDN_PROTOCOL_PARMS structure passed to **isdnStartProtocol**.) For details, see ISDN_PROTOCOL_PARMS_Q931CC parameters and ISDN_PROTOCOL_PARMS_LAPD parameters.

Rejecting the call

To reject the call, the application builds a clear request message (ACU_CLEAR_RQ) and sends it to the protocol stack. When the call is released, the network responds with ACU_CLEAR_CO.

The following illustration shows the default sequence of messages sent between the ACU and the application for a rejected inbound call. The actual messages may differ depending upon the settings of the bits in in_calls_behaviour.

This substructure is referenced in the ISDN_PROTOCOL_PARMS structure passed to **isdnStartProtocol**. For details, see ISDN_PROTOCOL_PARMS_Q931CC parameters and ISDN_PROTOCOL_PARMS_LAPD parameters.



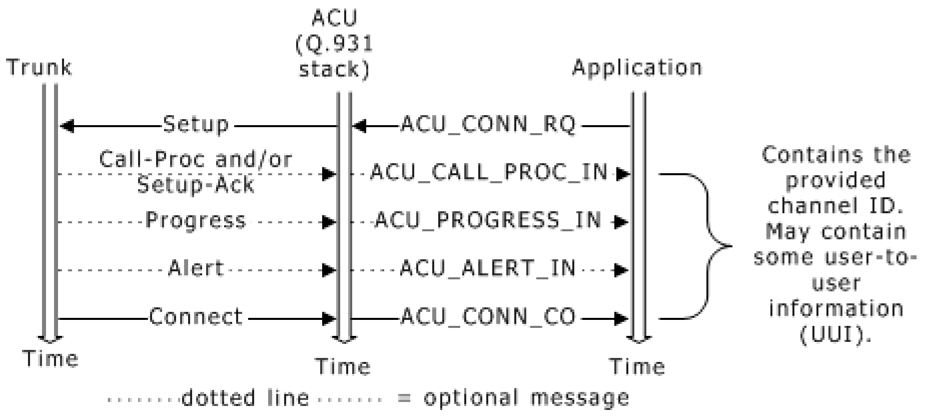
Placing outbound calls

The application initiates an outbound call by sending an ACU_CONN_RQ message to the protocol stack. The connection ID is assigned by the user. This ID must not currently be in use; otherwise the connection request is rejected by the ISDN protocol stack. The ACU_CONN_RQ message must contain the complete called number.

At this point, an ACU_CALL_PROC_IN event or an ACU_ALERT_IN event can be received from the network. These messages indicate that the call is in progress. If the call is successfully connected, the ACU_CONN_CO event is received. Otherwise a hang up indication, ACU_CLEAR_CO, is received.

Note: The bits in out_calls_behaviour (in the ISDN_PROTOCOL_PARMS structure passed to **isdnStartProtocol**) control stack behavior with outgoing calls.

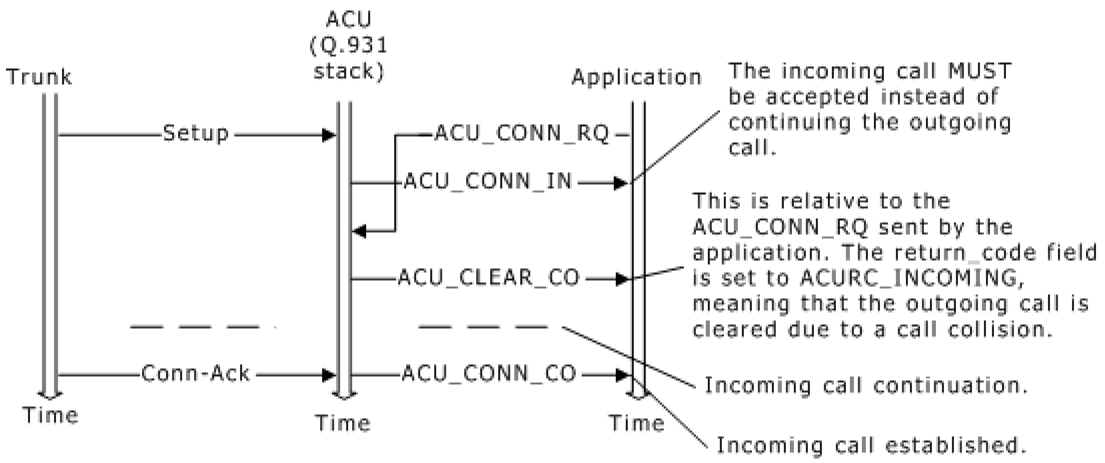
The following illustration shows the sequence of messages sent between the ACU and the application for an accepted outbound call:


ACU (Q.931 stack)
Trunk
Application
Setup
ACU_CONN_RQ
Call-Proc and/or Setup-Ack
ACU_CALL_PROC_IN
Progress
ACU_PROGRESS_IN
Alert
ACU_ALERT_IN
Connect
ACU_CONN_CO
Contains the provided channel ID. May contain some user-to-user information (UUI).
Time
Time
Time
dotted line = optional message

Call collision

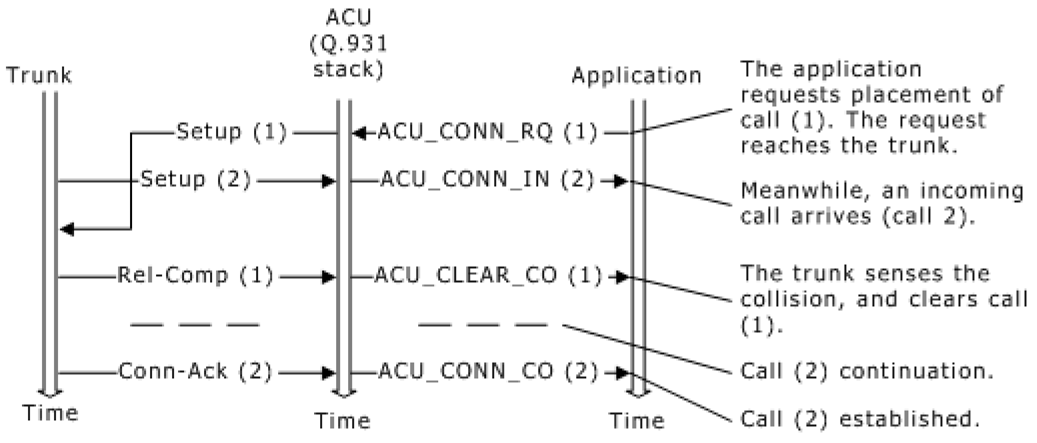
Call collision (also called glare) can occur when a call setup request and an incoming call occur on the same B channel at the same time. In this case, the terminal side must cancel its outgoing call and accept the incoming call.

Call collision can take place at the stack or at the network level. Call collision at the stack occurs when a SETUP message is received by the stack directly before it receives an ACU_CONN_RQ from the application. The following illustration shows the sequence of messages sent between the ACU and the application in this situation:

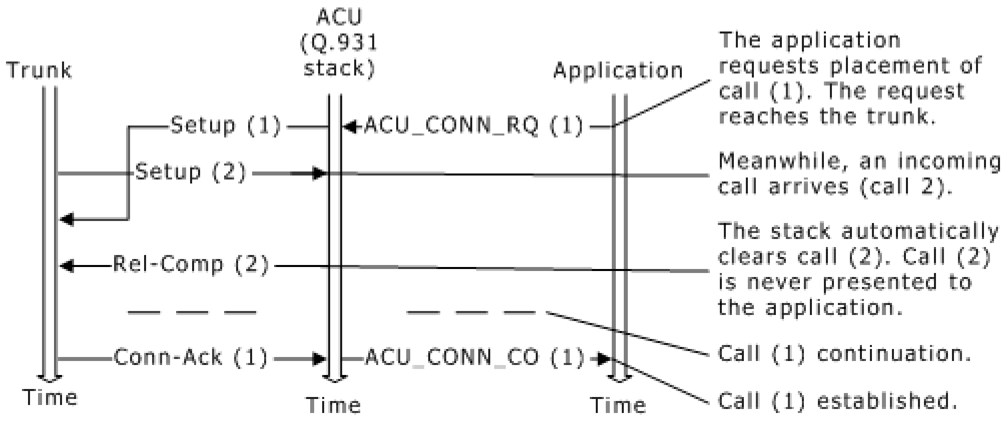

ACU (Q.931 stack)
Trunk
Application
The incoming call MUST be accepted instead of continuing the outgoing call.
Setup
ACU_CONN_RQ
ACU_CONN_IN
This is relative to the ACU_CONN_RQ sent by the application. The return_code field is set to ACURC_INCOMING, meaning that the outgoing call is cleared due to a call collision.
ACU_CLEAR_CO
Conn-Ack
ACU_CONN_CO
Incoming call continuation.
Time
Time
Time
Incoming call established.

The ACU_CLEAR_CO message is relative to the ACU_CONN_RQ sent by the application. The return_code field in the primitive's message data is set to ACURC_INCOMING, meaning that the outgoing call is cleared due to a call collision. No answer is sent by the application in response to this ACU_CLEAR_CO message.

Call collision at the network level occurs when the stack sends a SETUP message to the trunk directly before receiving a SETUP message from the trunk. If the stack is configured as terminal equipment, it abandons its call setup attempt and receives the incoming call. The following illustration shows the sequence of messages exchanged in this situation:



If the ISDN stack is configured as network equipment, it continues with its call setup attempt, signaling the trunk to release the incoming call. The following illustration shows the sequence of messages exchanged in this situation:

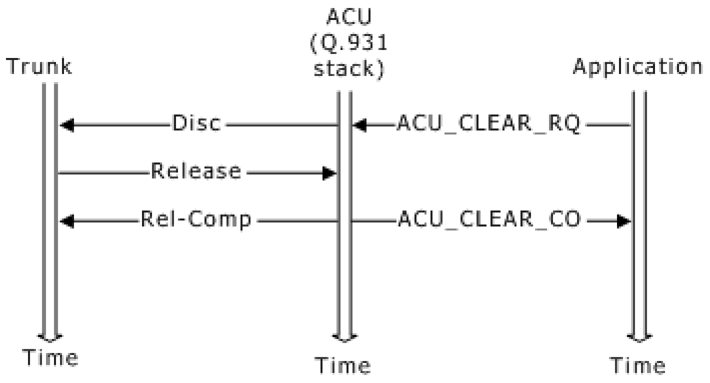

Trunk
ACU (Q.931 stack)
Application
The application requests placement of call (1). The request reaches the trunk.
Setup (1)
ACU_CONN_RQ (1)
Setup (2)
Meanwhile, an incoming call arrives (call 2).
Rel-Comp (2)
The stack automatically clears call (2). Call (2) is never presented to the application.
Call (1) continuation.
Conn-Ack (1)
ACU_CONN_CO (1)
Call (1) established.
Time
Time
Time

In any situation, if the application sends an ACU primitive to the stack while the stack is sending an ACU_CLEAR_CO message, the ISDN stack resends an ACU_CLEAR_CO message. In this case, the application receives two ACU_CLEAR_CO primitives.

Call clearing

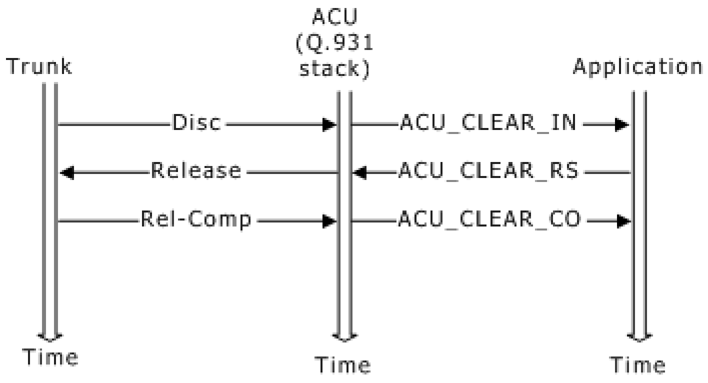
To hang up a call, the application builds an ACU_CLEAR_RQ message with the connection ID of the associated call. Receipt of an ACU_CLEAR_CO message confirms that the remote end has hung up. The stack returns to its IDLE state.

The following illustration shows the sequence of messages sent between the ACU and the application when the application initiates a hang up:



If the remote end hangs up first, the application receives an ACU_CLEAR_IN message. The application responds with an ACU_CLEAR_RS (clearing response) message. When the application receives an ACU_CLEAR_CO message, the clearing is confirmed. The stack returns to its IDLE state.

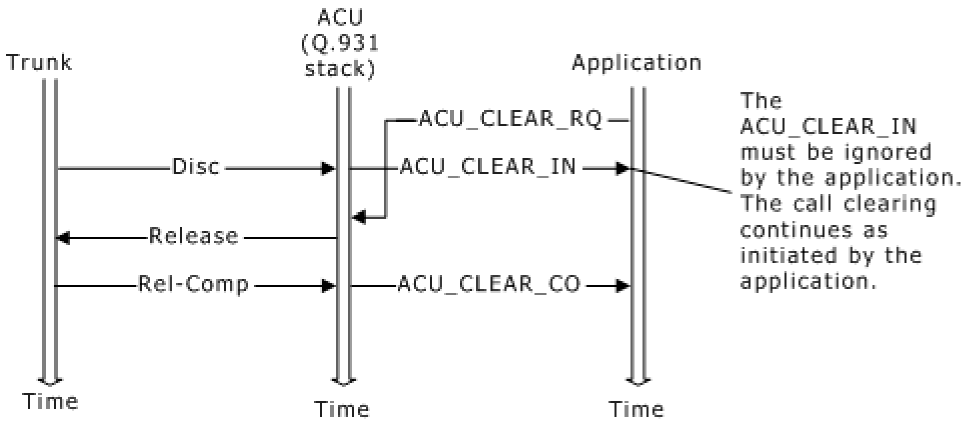
The following illustration shows the sequence of messages sent between the ACU and the application when a hang up is received by the application:



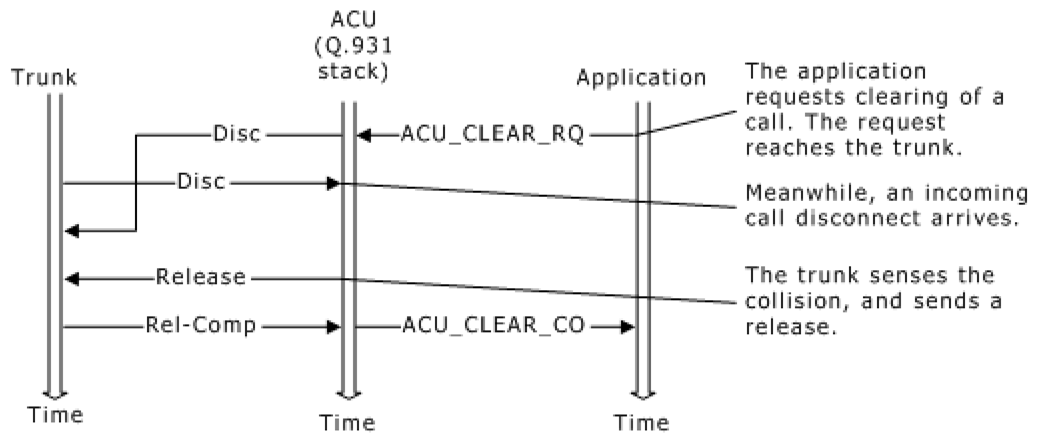
Clear collision

Clear collision can occur when a call clearing request (ACU_CLEAR_RQ) and a call clearing indication (ACU_CLEAR_IN) for the same call are sent at the same time. In this case, the application must ignore the ACU_CLEAR_IN message, and continue the call clearing as a normal outgoing call clearing (one issued by the application).

Clear collision can take place at the stack level or at the network level. Clear collision at the stack level takes place when a DISCONNECT message is received by the stack directly before it receives an ACU_CLEAR_RQ message from the application. The following illustration shows the sequence of messages exchanged in this situation:


Trunk
ACU (Q.931 stack)
Application
ACU_CLEAR_RQ
Disc
ACU_CLEAR_IN
Release
Rel-Comp
ACU_CLEAR_CO
The ACU_CLEAR_IN must be ignored by the application. The call clearing continues as initiated by the application.
Time
Time
Time

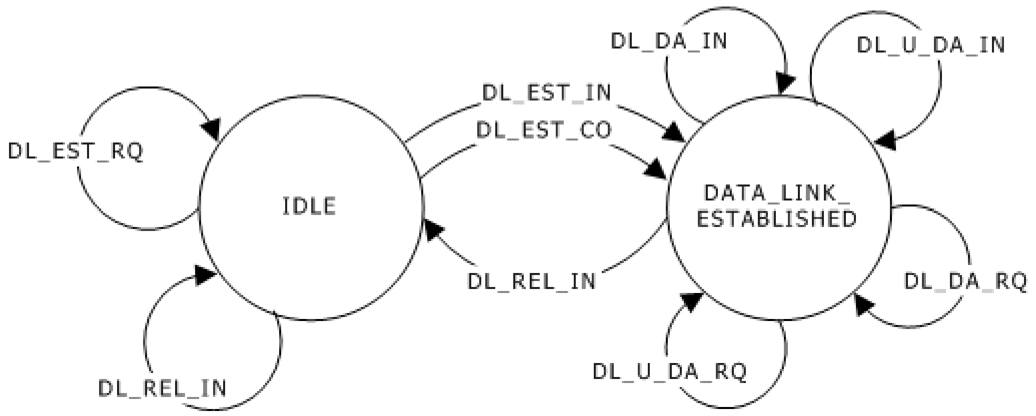
Clear collision at the network level takes place when a DISCONNECT message is received by the network directly after it sends a DISCONNECT message to the stack. The following illustration shows the sequence of messages exchanged in this situation:


Trunk
ACU (Q.931 stack)
Application
Disc
ACU_CLEAR_RQ
Disc
Release
Rel-Comp
ACU_CLEAR_CO
The application requests clearing of a call. The request reaches the trunk.
Meanwhile, an incoming call disconnect arrives.
The trunk senses the collision, and sends a release.
Time
Time
Time

6. Data link operations (LAPD)

NMS ISDN state machine (LAPD configuration)

An application can perform various operations at the data link layer with NMS ISDN software in an LAPD configuration. The following illustration shows the NMS ISDN software LAPD configuration state machine:



lapddemo, the LAPD demonstration program, demonstrates LAPD operations using NMS ISDN software.

NMS ISDN events are generated by the NMS ISDN protocol stack running on the board. When an NMS ISDN event occurs, the event ID ISDNEVN_RCV_MESSAGE is returned in the CTA_EVENT structure. When decoding the message, the following information is extracted:

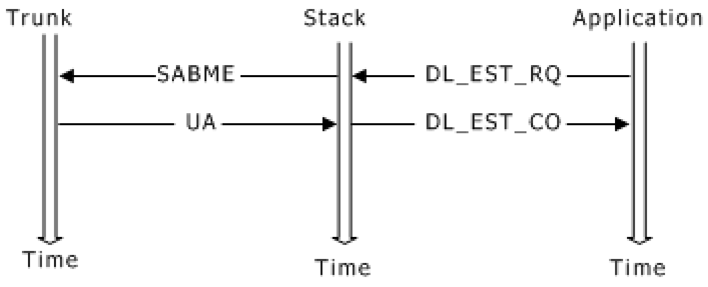
Item	Description
LAPD message	The message is a single byte code.
Sender	The sender is the ISDN entity that generated the message. For layer 2 ISDN call control messages (LAPD), the sender is ENT_DL_D (the data link layer).
Recipient	The recipient is ENT_APPLI (the application).

Initiating a data link

Before a data link is established, NMS ISDN is in the IDLE state. To establish a data link, the application sends a DL_EST_RQ message to the NMS ISDN protocol stack. This message requests the ISDN stack to transmit a SABME (set asynchronous balanced mode extended) message to the trunk. NMS ISDN remains in the IDLE state until the request is acknowledged by the remote side of the ISDN trunk.

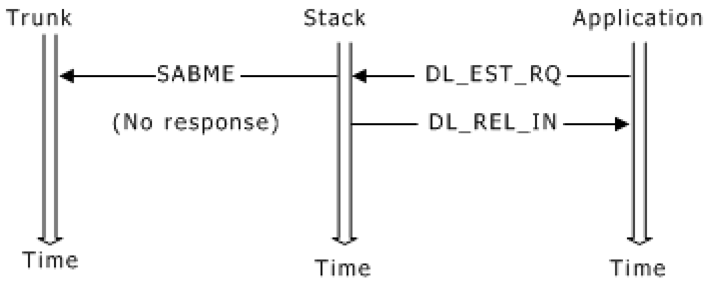
If the equipment on the remote side of the ISDN trunk is prepared to complete the data link, it acknowledges the SABME message with a UA (unnumbered acknowledgement) message. When the stack receives the UA message, it sends a DL_EST_CO message to the application. The data link is established, and NMS ISDN enters the DATA_LINK_ESTABLISHED state. Messages and other information can now be passed across the data link.

The following illustration shows the sequence of messages sent between the trunk, the stack, and the application when the application requests a data link:



If the SABME message is not acknowledged (the physical link is down, or the remote equipment is not prepared to complete the link), the stack sends the SABME message four more times, at one second intervals. If the message is not acknowledged within five tries, the stack sends a DL_REL_IN message to the application. NMS ISDN remains in the IDLE state.

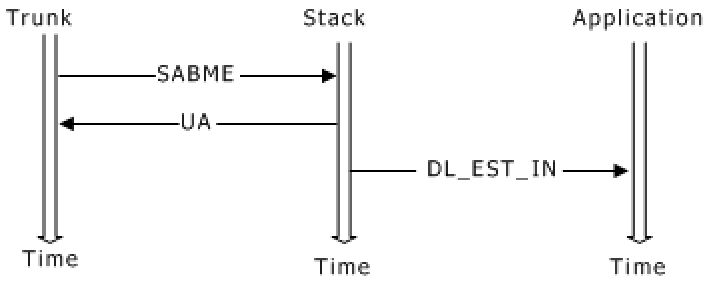
The following illustration shows the sequence of messages sent between the trunk, the stack, and the application when the remote party does not respond to a data link request:



Responding to a data link establishment request

Before a data link is established, NMS ISDN is in the IDLE state. If the NMS ISDN protocol stack receives a SABME message, it automatically responds with a UA message, and sends a DL_EST_IN message to the application. The data link is established, and NMS ISDN enters the DATA_LINK_ESTABLISHED state. Messages and other information can now be passed across the data link.

The following illustration shows the sequence of messages sent between the trunk, the stack, and the application when the stack responds to a data link establishment request:



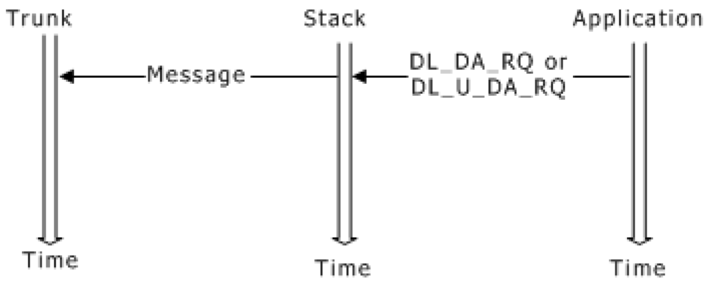
Sending messages across an established link

Once NMS ISDN is in the DATA_LINK_ESTABLISHED state, the application and the remote equipment can exchange messages. To send a packet of data, the application sends either of the following messages:

- If the packet is acknowledged (as defined in the Q.921 specification), send DL_DA_RQ to the stack with a pointer to the data.
- If the packet is unacknowledged, send DL_U_DA_RQ to the stack with a pointer to the data.

No confirmation is returned in response to either of these requests.

The following illustration shows the sequence of messages sent between the trunk, the stack, and the application when the application sends a message across an established data link:

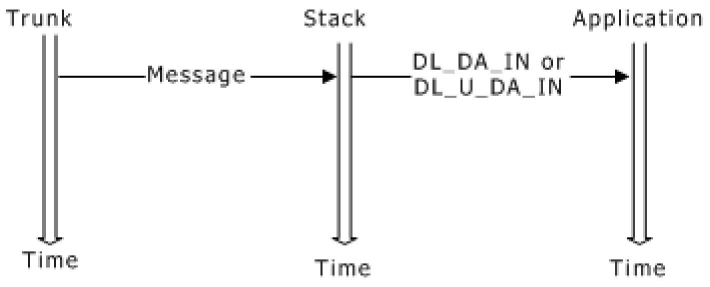


When a packet is received from the remote party, the stack sends one of the following messages to the application:

- DL_DA_IN if the packet is acknowledged.
- DL_U_DA_IN if the packet is unacknowledged.

The application does not confirm receipt of a packet.

The following illustration shows the sequence of messages sent between the trunk, the stack, and the application when the application receives a message across an established data link:



If the data link is broken while NMS ISDN is in DATA_LINK_ESTABLISHED state, the stack sends DL_REL_IN to the application. NMS ISDN returns to the IDLE state.

7. Messaging API data structures

Overview of API data structures

NMS ISDN uses data structures to configure the ISDN protocol stack, send messages to the protocol stack, and receive messages from the protocol stack. This topic describes the NMS ISDN data structures in detail. These data structures are defined in the *isdnparm.h* header file.

Messaging structures

The message structures used to convey messages and message data between the ISDN protocol stack and the application are:

- ISDN_MESSAGE
- ISDN_PACKET

For more information on sending and receiving NMS ISDN messages, refer to Sending ISDN messages to the stack and Receiving events and ISDN protocol stack messages.

Protocol parameter structures

Data structures are passed to **isdnStartProtocol**. The ***parms*** argument points to one of these parameter structures, depending upon the stack mode in which the ISDN protocol stack is running:

For...	Specify...
ACU stack mode	ISDN_PROTOCOL_PARMS_Q931CC
LAPD stack mode	ISDN_PROTOCOL_PARMS_LAPD
Channelized stack mode	ISDN_PROTOCOL_PARMS_CHANNELIZED

Refer to the Message primitives section for more information on the default values of these parameters.

ISDN_MESSAGE structure

A pointer to the ISDN_MESSAGE structure is passed to **isdnSendMessage** in the ***message*** argument. In this structure, the application specifies the NAI, the NFAS group number (if duplicate NAI values are configured), and the connection ID of the call that the message concerns. The message to be sent, expressed using one of the message primitives, is also specified. The message primitive appears in the code field in this structure. For more information, refer to the Message primitives section.

When the ISDN_PACKET structure is received by the application, it contains a pointer to an ISDN_MESSAGE structure containing message data. For more information, see ISDN_PACKET structure.

ISDN_MESSAGE is defined as follows:

```
typedef struct ISDN_MESSAGE
{
  nai_t nai;         /* Network access interface index                       */
  ent_id_t from_ent; /* Message source                                       */
  ent_id_t to_ent;   /* Message destination                                  */
  sapi_t to_sapi;    /* Destination Service Access Point                     */
union {
  add_t conn_id;     /* Connection identifier for the ACU layer              */
  add_t crv;         /* Call Reference value for the NS layer. Not used      */
  add_t ces;         /* Connection Endpoint suffix (DL later upper half)     */
  add_t tei;         /* Terminal Endpoint ID (DL layer lower half). Not used */
  add_t chani;       /* Physical layer channel identifier. Not used          */
} add;
  code_t code;       /* Primitive code unique only between two entities      */
  WORD inf0;         /* Information location 0                               */
  WORD inf1;         /* Information location 1                               */
  WORD inf2;         /* Information location 2                               */
  WORD inf3;         /* Information location 3                               */
  WORD inf4;         /* Information location 4                               */
  WORD data_size;    /* Size of data to follow                               */
  WORD nfas_group;   /* NFAS group number, if multiple NAI values are        */
                     /* configured                                           */
  DWORD userid;      /* User ID                                              */
} ISDN_MESSAGE;
```

ISDN_PACKET structure

When a message is sent from the protocol stack to the application, it is sent in an ISDN_PACKET structure. This ISDN_PACKET structure is contained in the buffer element of the data structure returned by the function **ctaWaitEvent**.

For more information about receiving ISDN messages, see Network access identifiers (NAIs).

ISDN_PACKET is defined as follows:

```
typedef struct ISDN_PACKET
{
    ISDN_MESSAGE message;   /* ISDN message identification information    */
    BYTE data[4];           /* Data included in packet (>=0)              */
} ISDN_PACKET;
```

ISDN_PROTOCOL_PARMS_LAPD structure

The ISDN_PROTOCOL_PARMS_LAPD data structure configures the protocol stack for LAPD. A pointer to this structure is passed as an argument to **isdnStartProtocol**.

Refer to ISDN_PROTOCOL_PARMS_LAPD parameters for more information on the default values of the parameters in this structure.

The structure is defined as:

```
typedef struct ISDN_PROTOCOL_PARMS_LAPD
{
    DWORD size;                /* Size of this structure */
    timer_val_t  t101;
    timer_val_t  t102;
    timer_val_t  t198;           /* Observation period for Frame error count */
    WORD rate;                   /* Data rate                                */
    WORD max_FEC_errors;         /* Maximum number of FEC during t198        */

/* When to assign/remove a TEI */
    WORD tei_time_assignment;    /* TEI time assignment at : CONFIGURATION,
                                    NA ACTIVATION or USAGE time (isdndl.h)   */
    WORD tei_time_removal;       /* TEI time removal at    : NA DEACTIVATION,
                                    POWER DOWN (dlint.h)                     */
    BYTE tei[3];                 /* TEI values:
                                   0     :      Broadcast
                                   1-63  :      Non automatic TEI assignment
                                   127   :      Automatic TEI assignment     */
    BYTE bpad[1];              /* For 8 bytes alignment                      */
                               /* Size is now 32 bytes                       */
    WORD nfas_group;           /* NFAS group number if duplicate NAI values */
    BYTE bpad1[2];             /* For 8 bytes alignment                      */
                               /* Size is now 32 bytes                       */
} ISDN_PROTOCOL_PARMS_LAPD;
```

ISDN_PROTOCOL_PARMS_Q931CC structure

The ISDN_PROTOCOL_PARMS_Q931CC data structure configures the protocol stack for Q.931 call control. A pointer to this structure is passed as an argument to **isdnStartProtocol**.

Refer to ISDN_PROTOCOL_PARMS_Q931CC parameters for more information on the default values of the parameters in this structure.

The structure is defined as:

```
typedef struct ISDN_PROTOCOL_PARMS_Q931CC
{
    DWORD size;                  /* Size of the structure                    */
                                 /* NAI when interfacing the physical layer
                                    should be the same as the 'nai'          */
    WORD rate;                   /* Data rate                                */
    WORD t309;                   /* T309 in use flag                         */
  union{                         /* These structures contain sets of values */
    timer_val_t at5[AT5_T_LAST]; /* for ISDN timers. The actual set used     */
    timer_val_t at9[AT9_T_LAST]; /* depends upon the country variant         */
    timer_val_t dms[DMS_T_LAST]; /* specified with the 'country' argument    */
    timer_val_t ni1[NI1_T_LAST]; /* in the isdnStartProtocol call. The       */
    timer_val_t ni2[NI2_T_LAST]; /* structures should be set to 0 in         */
    timer_val_t au1[AU1_T_LAST]; /* order to use the built-in defaults       */
    timer_val_t hkt[HKT_T_LAST]; /* defined by the network signaling         */
    timer_val_t ntt[NTT_T_LAST]; /* layers of the ISDN protocol stack.       */
    timer_val_t bt2[BT2_T_LAST]; /* Timers are specified in milliseconds.    */
    timer_val_t bv1[BV1_T_LAST]; /*CAUTION: The values of these timers       */
    timer_val_t ets[ETS_T_LAST]; /*         may be controlled by local       */
    timer_val_t qsi[QSI_T_LAST]; /*         regulatory authorities:          */
    timer_val_t swd[SWD_T_LAST]; /*         changing these values may        */
    timer_val_t tr6[TR6_T_LAST]; /*         invalidate regulatory            */
    timer_val_t vn2[VN2_T_LAST]; /*         approvals. Check with the        */
    timer_val_t vn3[VN3_T_LAST]; /*         local authority for more         */
    timer_val_t vn6[VN6_T_LAST]; /*         specific information on          */
                                 /*         any limitations.                 */
    timer_val_t foo[24];         /* 24 is more then the rest, will align
                                    the union on an 8 byte boundary.         */
    } timers;

/* Available services */
    BYTE services_list[CC_MX_SERVICES+1];
    BYTE bpad[2];                /* Padding for 8 bytes alignment           */
    timer_val_t  t101;
    timer_val_t  t102;
    timer_val_t  t198;           /* Observation period for Frame error count*/
    WORD max_FEC_errors;         /* Maximum number of FEC during t198       */

/* When to assign/remove a TEI */
    WORD tei_time_assignment;    /* TEI time assignment at : CONFIGURATION,
                                  NA ACTIVATION or USAGE time (isdndl.h) */
    WORD tei_time_removal;       /* TEI time removal at  : NA DEACTIVATION,
                                   POWER DOWN (isdndl.h)                 */
    WORD wpad[3];                /* Padding for 8 bytes alignment           */
    BYTE tei[3];                 /* TEI values:
                                  0     :      Broadcast
                                  1-63  :      Non automatic TEI assignment
                                  127   :      Automatic TEI assignment     */
    BYTE digitstoroute;          /* Number of digits needed to route when
                                   using overlap receiving                  */
    BYTE bpad1[4];               /* Padding for 8 bytes alignment           */

/* Call control behaviour        */
    WORD in_calls_behaviour;     /* Incoming calls behaviour                */
    WORD out_calls_behaviour;    /* Outgoing calls behaviour                */

/* Network signaling behavior    */
    WORD ns_behaviour;

/* Automatic call unit behaviour */
    WORD acu_behaviour;
    BYTE qsig_source_party_nb_type;             /* Used for network node
                                                   addressing in SS         */
    BYTE qsig_source_type_of_nb;                /* Used for network node
                                                   addressing in SS         */
    BYTE qsig_source_addr[CC_QSIG_MX_ADDR_SIZE+1];
    BYTE aoc_s_presubscribed;    /* On/Off, ON indicates na presubscribes
                                    to service                              */
    BYTE aoc_d_presubscribed;    /* On/Off, ON indicates na presubscribes
                                    to service                              */
    BYTE aoc_e_presubscribed;    /* On/Off, ON indicates na presubscribes
                                    to service                        */
    BYTE bpad2[1];               /* Padding for 8 bytes alignment           */
    WORD nfas_group;           /* NFAS group number if duplicate NAI values */
    BYTE bpad3[2];               /* Padding for 8 bytes alignment           */
    WORD rfu1;                   /* Reserved for future use  MUST BE 0      */
    WORD rfu2;                   /* Reserved for future use  MUST BE 0      */
    WORD w2pad[2];               /* Padding for 8 bytes alignment           */

} ISDN_PROTOCOL_PARMS_Q931CC;
```

ISDN_PROTOCOL_PARMS_CHANNELIZED structure

The ISDN_PROTOCOL_PARMS_CHANNELIZED data structure configures the protocol stack so it is accessible to the Natural Call Control (NCC) service. A pointer to this structure is passed as an argument to **isdnStartProtocol**.

The structure is identical to the ISDN_PROTOCOL_PARMS_Q931CC structure.

8. Function reference

Function summary

The following table summarizes the NMS ISDN Messaging API functions:

Function	Synchronous / Asynchronous	Description
isdnReleaseBuffer	Synchronous	Indicates that the application has completed processing of an ISDN event buffer.
isdnSendMessage	Asynchronous	Sends a message to an ISDN protocol stack instance.
isdnSetMsgCapture	Asynchronous	Enables or disables debugging trace information for the selected entity in the protocol stack.
isdnStartProtocol	Asynchronous	Initializes an ISDN protocol stack instance on a D channel context.
isdnStopProtocol	Asynchronous	Shuts down a previously started ISDN protocol stack instance on a context.

Using the function reference

A prototype of each function is shown with the function description and details of all arguments and return values. A typical function description includes:

Prototype	The prototype is shown followed by a listing of the function's arguments. NMS data types include: • WORD 16-bit unsigned • DWORD 32-bit unsigned • INT16 16-bit signed • INT32 32-bit signed • BYTE 8-bit unsigned If a function argument is a data structure, the complete data structure is defined.
Return values	The return value for a function is either SUCCESS or an error code. For asynchronous functions, a return value of SUCCESS indicates the function was initiated; subsequent events indicate the status of the operation. Refer to NMS ISDN errors for a listing of all errors returned by NMS ISDN functions.

Prototype	The prototype is shown followed by a listing of the function's arguments. NMS data types include: • WORD 16-bit unsigned • DWORD 32-bit unsigned • INT16 16-bit signed • INT32 32-bit signed • BYTE 8-bit unsigned If a function argument is a data structure, the complete data structure is defined.
Events	If events are listed, the function is asynchronous and is complete when the DONE event is returned. If there are no events listed, the function is synchronous. Additional information such as reason codes and return values can be provided in the value field of the event. Refer to NMS ISDN events and NMS ISDN reasons for information on all NMS ISDN events and reason codes.
Details	Information specific to the operation and use of a function.
See also	Functions related to the function being described.
Example	An example code fragment. The notation /* ... */ indicates additional code that is not shown.

isdnReleaseBuffer

Returns an event buffer to the NMS ISDN API.

Prototype

DWORD **isdnReleaseBuffer** (CTAHD ***ctahd***, void ****buffer***)

Argument	Description
ctahd	Context handle associated with a D channel, returned by **ctaCreateContext**.
buffer	Pointer to the event buffer.

Return values

Return value	Description
SUCCESS	
CTAERR_INVALID_CTAHD	The context handle is invalid.

Return value	Description
CTAERR_INVALID_STATE	An ISDN protocol stack instance is not started on the specified context, or an instance is starting or stopping on the context.
ISDNERR_INVALID_BUFFER	The buffer submitted is not valid.

Events

None.

Details

This function informs the stack that the application has finished processing an event buffer (described by the CTA_EVENT buffer and size fields) and is returning the buffer to the NMS ISDN API. The event ID for ISDN events is ISDNEVN_RCV_MESSAGE.

The application must return every event buffer to the NMS ISDN API as soon as possible, or the API times out and stops passing events to the application.

Example

```
DWORD sample_process_events (CTAHD ctahd)
{
CTA_EVENT     event;
ISDN_MESSAGE *imsg;
ISDN_PACKET  *ipkt;
BYTE         *data;
DWORD         ret;
char          errortext[40];

#define EVENT_CODE(from, code)       ((from<<8)|code)
...
/*
** Protocol already started...
** Application may have sent messages to stack
*/
...
/*
** Application waiting for events
*/
myWaitForEvent( ctahd, &event);

/*
** Got event ISDN_RCV_MESSAGE
** If the event value field is not SUCCESS, then
** the event was not received successfully.
*/
if( event.value != SUCCESS )
{
    ctaGetText(ctahd, event.value, errortext, 40);
    printf("RCV_FAIL: %s\n", errortext);
    return MY_ERROR_RCV_FAILED;
}

/*
** NOTE: all asynchronous events have the
** msg->userid field is ISDN_USERID_ASYNC
*/
ipkt = (ISDN_PACKET *) event->buffer;
imsg = &ipkt->message;
data = ipkt->data;

printf("from: %c  code=%c  to=%c   id=%d len=%d\n",
        imsg->from_ent,
        imsg->code,
        imsg->to_ent,
        imsg->add.conn_id,
        ipkt->data_len);

switch(EVENT_CODE(imsg->from_ent, imsg->code) )
{
    case EVENT_CODE(ENT_ACU, CONN_CO):
        /*
        ** Call is now connected
        */
        printf("Connected on conn_id: %d\n",imsg->add.conn_id);
        break;

    case EVENT_CODE(ENT_ACU,ACU_CLEAR_CO):
        /*
        ** Call is now cleared
        */
        printf("Cleared on conn_id: %d\n", imsg->add.conn_id);
        break;
...

    default:
        printf("Unprocessed message: %c %c\n", imsg->from_ent,
                imsg->code);
        break;
}
/*
** Processing is done, release the buffer as soon as possible
*/
ret = isdnReleaseBuffer( ctahd, event.buffer );
if( ret != SUCCESS )
{
    ctaGetText(ctahd, event.value, errortext, 40);
    printf("RELEASE_FAIL: %s\n", errortext);
    return  MY_ERROR_RELEASE_FAILED;
}
...
}
```

isdnSendMessage

Sends a message to the ISDN stack, with optional attached data.

Prototype

DWORD **isdnSendMessage** (CTAHD ***ctahd***, ISDN_MESSAGE ***message***, void ***pdata***, unsigned ***size***)

Argument	Description
ctahd	Context handle associated with a D channel, returned by **ctaCreateContext**.

Argument	Description

message	Pointer to ISDN_MESSAGE structure, as follows:
	`typedef struct ISDN_MESSAGE`
	`{`
	`  nai_t nai;         /* Network access interface index                     */`
	`  ent_id_t from_ent; /* Message source                                     */`
	`  ent_id_t to_ent;   /* Message destination                                */`
	`  sapi_t to_sapi;    /* Destination Service Access Point                   */`
	`union {`
	`      add_t conn_id; /* Connection identifier for the ACU layer            */`
	`      add_t crv;     /* Call Reference value for NS layer. Not used.       */`
	`      add_t ces;     /* Connection Endpoint suffix (DL layer upper 1/2)    */`
	`      add_t tei;     /* Terminal Endpoint ID, DL layer lower 1/2 Not used*/`
	`      add_t chani;   /* Physical layer channel identifier. Not used.       */`
	`      } add;`
	`  code_t code;       /* Primitive code unique only between 2 entities      */`
	`  WORD inf0;         /* Information location 0                             */`
	`  WORD inf1;         /* Information location 1                             */`
	`  WORD inf2;         /* Information location 2                             */`
	`  WORD inf3;         /* Information location 3                             */`
	`  WORD inf4;         /* Information location 4                             */`
	`  WORD data_size;    /* Size of data to follow                             */`
	`  WORD nfas_group;   /* NFAS group number, used for configurations         */`
	`                     /* with duplicate NAI values only                     */`
	`  DWORD userid;      /* User ID                                            */`
	`} ISDN_MESSAGE;`

pdata	Pointer to the message data (if any). The data is specific to the type of message specified in ISDN_MESSAGE.
size	Size of data block referenced by ***pdata***. ***size*** must match the data_size field in the ISDN_MESSAGE structure.

Return values

Return value	Description
SUCCESS	
CTAERR_BAD_ARGUMENT	This return value means any of the following: • ***message*** argument is NULL. • ***pdata*** is NULL but ***size*** is non-zero. • data_size in the message does not match the ***size*** parameter. • Size of the data exceeds MAX_ISDN_BUFFER_SIZE.
CTAERR_INVALID_CTAHD	Context handle is invalid.

Events

Event name	Description
ISDNEVN_SEND_MESSAGE	The event value field contains one of the following reasons or an error code: SUCCESS ISDNERR_BAD_NAI The network access identifier (NAI) in the message structure is not valid (the NAI must be less than MAX_NAI specified in *isdnparm.h*), or an ***nfas_group*** and NAI couple is not valid (if duplicate NAI values are defined). ISDNERR_BUFFER_TOO_BIG The size of the buffer is too large.

Details

This function sends a message with optional attached data to the ISDN subsystem. Any ISDN-specific command can be sent to any layer of the protocol stack using this function. The ISDN_MESSAGE structure contains the addressing information for the message.

The size field of the event contains the user ID for the message, as specified in the userid field in ISDN_MESSAGE. This value is sent to distinguish between multiple messages sent to the protocol stack. ISDN_USERID_ASYNC is reserved for events initiated by the protocol stack.

If multiple NAI values have not been defined, it is recommended to set ***nfas_group*** to 0.

See also

isdnReleaseBuffer

Example

```
DWORD sample_send_message (CTAHD ctahd, int mycode)
{
CTA_EVENT     event;
DWORD         ret;
ISDN_MESSAGE  imsg={0};
code_t        code;
unsigned char idata[MAX_ISDN_BUFFER_SIZE];
unsigned      datasize;

/*
** Protocol already started
*/
imsg.nai = 0;
imsg.nfas_group = 0
/*
** When using ACU, all messages should be directed
** to ENT_ACU and the ACU_SAPI within it.
** The from_ent field should always be ENT_APPLI.
*/
imsg.from_ent = ENT_APPLI;
imsg.to_ent   = ENT_CC;
imsg.to_sapi  = ACU_SAPI;

/*
** The connection ID is the logical connection number.
** For ACU, the lowest unused connection ID value must be used.
*/
imsg.add.conn_id = myGetLowestConnectionId();

/*
** Build the contents of the message for place call, release call, etc.
** This should fill the idata with the message contents and return the
** used part of the data buffer in the datasize argument and the code for the
** requested function (ACU_CONN_RQ, ACU_CLEAR_RQ, etc).
*/
myBuildMessage( mycode, idata, &datasize, &code);
imsg.datasize = datasize;
imsg.code     = code;

/*
** Add an ISDN-specific user ID to identify this message if it fails:
*/
imsg.userid = myGetNextMessageId();

ret = isdnSendMessage( ctahd, &imsg, idata, datasize);
if( ret != SUCCESS)
{
    ctaGetText(ctahd, ret, errortext, 40;
    printf("SEND_FAIL: %s mg id=%x\n", errortext, imsg.userid );
    return MY_ERROR_SEND_FAILED;
}
myWaitForEvent( ctahd, &event)
if( event.value != SUCCESS)
{
    ctaGetText(ctahd, event.value, errortext, 40);
    printf("SEND_FAIL: %s mg id=%x\n", errortext, imsg.userid );
    return MY_ERROR_SEND_FAILED;
}
...
}
```

isdnSetMsgCapture

Sends tracing messages to the *oammon* monitor screen from the selected ISDN entity on the board associated with the specified context.

Prototype

DWORD **isdnSetMsgCapture** (CTAHD ***ctahd***, DWORD ***enable***, DWORD ***nai***, char ****entity_id_string***, DWORD ***nfas_group***)

Argument	Description
ctahd	Context handle associated with a D channel, returned by **ctaCreateContext**.
enable	Toggle for tracing: 1 - ISDN protocol entity ***entity_id_string*** generates tracing messages. 0 - tracing disabled.
nai	Network access identifier (NAI) on which to enable or disable tracing.

Argument	Description
entity_id_string	Pointer to the NULL-terminated string of one-character names of ISDN protocol entities for which tracing is either enabled or disabled according to the ***enable*** parameter. The entity names are defined in the *isdntype.h* include file.
nfas_group	NFAS group number. This parameter is used only if duplicate NAI values are defined in the configuration. Otherwise, NMS recommends setting ***nfas_group*** to the NAI used to set the capture mask.

Return values

Return value	Description
SUCCESS	
CTAERR_BAD_ARGUMENT	The entity string is NULL.
CTAERR_INVALID_CTAHD	The context handle is invalid.
CTAERR_INVALID_STATE	An ISDN protocol stack instance: • Has not been started on the specified context handle, • Is being started by a previous call, • Is already started, or • Is stopping.

Events

Event name	Description
ISDNEVN_SET_MSG_CAPTURE	The event value field contains one of the following reasons or an error code: SUCCESS ISDNERR_BAD_NAI The network access identifier (NAI) is not valid. The NAI must be less than MAX_NAI specified in *isdnparm.h*. An NAI and NFAS group couple is invalid if the NAI value is not unique in the configuration.

Details

This function enables or disables tracing of messages generated by the ISDN protocol entities named by the ***entity_id_string***. The associated board is indicated by the specified ***ctahd***.

When the ***entity_id_string*** contains an asterisk (*), all entities are affected. By default, tracing is enabled for all entities.

In order for messages to be sent to the monitor screen, the flag passed to *oamtrace* (formerly *agtrace*) must be set to 0x801000.

For configurations where all NAI values are unique, NMS recommends setting ***nfas_group*** to 0.

Example

```
#define TRACE_ENABLE   1
#define TRACE_DISABLE  0

DWORD mySetTrace( CTAHD ctahd, DWORD enable, char *list )
{
DWORD ret;
CTA_EVENT event;
char errortext[40];

    ret = isdnSetMsgCapture(  ctahd,  enable, 0, trace_list, 0 );
    if( ret != SUCCESS)
    {
        ctaGetText(ctahd, ret, errortext, 40);
        printf("TRACE_FAIL: %s\n", errrortext);
        return MY_ERROR_TRACE_FAILED;
    }

    myWaitForEvent( ctahd, &event)
    if( event.value != SUCCESS)
    {
        ctaGetText(ctahd, event.value, errortext, 40);
        printf("TRACE_FAIL: %s\n", errortext);
        return MY_ERROR_TRACE_FAILED;
    }
}


void do_trace (CTAHD ctahd)
{
    char trace_list[20];

     /*
     **  Disable all tracing first
     */
    mySetTrace( ctahd, TRACE_DISABLE, "*" );

     /*
     ** Enable tracing for call control layer and the application
     */
    sprintf(trace_list, "%c%c", ENT_CC, ENT_APPLI );
    mySetTrace( porthd, TRACE_ENABLE,  trace_list );
}
```

isdnStartProtocol

Starts up an ISDN protocol stack instance on a specified context.

Prototype

DWORD **isdnStartProtocol** (CTAHD ***ctahd***, unsigned ***protocol***, unsigned ***netoperator***, unsigned ***country***, unsigned ***partner_equip***, unsigned ***nai***, void ****parms***)

Argument	Description
ctahd	Context handle associated with a D channel returned by **ctaCreateContext**.
protocol	Stack mode to start protocol instance in: • ISDN_PROTOCOL_Q931CC for ACU stack mode. • ISDN_PROTOCOL_LAPD for LAPD stack mode. • ISDN_PROTOCOL_CHANNELIZED for channelized stack mode.
netoperator	Network operator variant to start. Refer to Valid netoperator and country combinations.
country	Country mode in which the network operator variant starts. A variant's behavior can change depending on the country specified. Refer to Valid netoperator and country combinations. ***country*** must be the same for all NAIs on a single board.
partner_equip	Type of equipment connected to the board. Refer to partner_equip settings.
nai	Network access identifier (NAI) of the D channel to link to the protocol stack instance.
parms	Pointer to the parameter block/structure required by the protocol: • For access to the parameters for NMS ISDN in LAPD stack mode, use the ISDN_PROTOCOL_PARMS_LAPD structure, as described in ISDN_PROTOCOL_PARMS_LAPD structure. • For access to the parameters for NMS ISDN in ACU stack mode, use ISDN_PROTOCOL_PARMS_Q931CC, as described in ISDN_PROTOCOL_PARMS_Q931CC structure. • For access to the parameters for NMS ISDN in Channelized stack mode, use the ISDN_PROTOCOL_PARMS_CHANNELIZED structure, as described in ISDN_PROTOCOL_PARMS_CHANNELIZED structure. When parms is NULL, the default parameters for the protocol are used. The default parameters for each protocol enable the required service access points (SAPIs). For the ACU stack mode, all the services are supported by default. The size field of the structure must contain the size of the structure. Refer to Overview of API data structures for details on the contents of these parameter structures.

Valid netoperator and country combinations

The following table lists the valid netoperator and country combinations:

netoperator value	Country value	Country
ISDN_OPERATOR_ATT_4ESS ISDN_OPERATOR_ATT_5E10 ISDN_OPERATOR_NT_DMS ISDN_OPERATOR_NT_DMS250 ISDN_OPERATOR_NI2	COUNTRY_USA	USA
ISDN_OPERATOR_FT_VN6	COUNTRY_FRA	France
ISDN_OPERATOR_AUSTEL_1	COUNTRY_AUS	Australia
ISDN_OPERATOR_ETSI	COUNTRY_AUS, COUNTRY_BEL, COUNTRY_GER, COUNTRY_SWE, COUNTRY_SGP, COUNTRY_GBR, COUNTRY_CHINA, COUNTRY_EUR	Australia, Belgium, Germany, Sweden, Singapore, Great Britain, and China. COUNTRY_EUR includes the following countries: Austria, Belgium, Denmark, Finland, Germany, Greece, Iceland, Ireland, Italy, Liechtenstein, Luxembourg, Netherlands, Norway, Portugal, Russia, Spain, Sweden, Switzerland, and the UK.
ISDN_OPERATOR_HONG_KONG	COUNTRY_HONG_KONG	Hong Kong
ISDN_OPERATOR_KOREA	COUNTRY_KOR	Korea
ISDN_OPERATOR_NTT	COUNTRY_JPN	Japan
ISDN_OPERATOR_TAIWAN	COUNTRY_TWN	Taiwan
ISDN_OPERATOR_ECMA_QSIG	NA	country is ignored if variant is QSIG.
ISDN_OPERATOR_DPNSS	COUNTRY_EUR	Europe

The following table shows the ISDN and DPNSS run module files for NMS boards:

Board	ISDN	DPNSS
AG (except for AG 2000-BRI boards)	*isdngen.leo*	*dpnss.leo*
CG 6565/C	*c6565igen.dlm*	*c6565dpnss.dlm*
CG 6060/C	*c6060igen.dlm*	*c6060dpnss.dlm*

For AG 2000-BRI boards, the following table shows the valid netoperator values and the corresponding run module file:

netoperator value	Variant	Run module file
ISDN_OPERATOR_ETSI	ETSI	*brietsi.leo*
ISDN_OPERATOR_NTT	NTT	*brintt.leo*
ISDN_OPERATOR_VN6	VN6	*brivn6.leo*

partner_equip settings

Refer to the following table when setting the partner_equip argument:

If board is...	And NMS ISDN is to run in...	Set partner_equip to:
Connected to network	ACU stack mode	EQUIPMENT_NT
Connected to network	LAPD stack mode	EQUIPMENT_DCE
Acting as a network	ACU stack mode	EQUIPMENT_TE
Acting as a network	LAPD stack mode	EQUIPMENT_DTE

NMS ISDN also supports DPNSS. For DPNSS, the designations TE and NT do not apply. For the DPNSS implementation, TE maps to PBX A and NT maps to PBX B. To run PBX A, the partner equipment designation is EQUIPMENT_NT.

By default, PBX A is assigned to side X and PBX B is assigned to side Y. Use the NS behavior bits to modify this setting.

Return values

Return value	Description
SUCCESS	
CTAERR_BAD_ARGUMENT	protocol argument is invalid, or the size field of the parms data structure does not match the size of the structure corresponding to the protocol value.
CTAERR_INVALID_CTAHD	Context handle is invalid.
CTAERR_INVALID_STATE	One of the following conditions exists: • The context is not open. • An ISDN protocol stack instance is being started on the same context by a previous call. • An instance is already started on the context. • An instance on the context is in the process of stopping.
CTAERR_OUT_OF_MEMORY	Memory allocation failed on the host machine.
ISDNERR_CHANNELIZED_ON_MULTIPLE_BOARDS	An NFAS group is defined including multiple boards, and an attempt was made to start the stack in channelized stack mode.
ISDNERR_PROTOCOL_NS_FAILURE	NFAS groups are configured, and an attempt was made to start the stack using a variant that does not support NFAS.
ISDNERR_RACE_STARTING_PROTOCOL	An attempt was made to restart the stack directly after stopping it, before it was completely stopped. The process of stopping the stack can take several seconds.

Events

Event name	Description
ISDNEVN_START_PROTOCOL	Value field of the received event contains the completion status of the protocol starting operation, as follows: SUCCESS ISDNERR_BAD_NAI Network access identifier (NAI) in the protocol parameters structure is not valid. Or, if duplicate NAI values are configured, the NAI and NFAS group couple is not valid. ISDNERR_INCOMPATIBLE_LIB NMS ISDN library used is incompatible with the run module. ISDNERR_INVALID_COUNTRY Country specified is invalid for the network operator specified. ISDNERR_INVALID_OPERATOR Network operator specified is not supported by the run module. ISDNERR_INVALID_PARTNER partner_equip is not supported by the run module. ISDNERR_INVALID_PROTOCOL protocol argument is not supported by the run module. ISDNERR_NAI_IN_USE Another thread or process has already started a protocol for the same network access identifier. ISDNERR_PROTOCOL_CC_FAILURE Call control parameters are invalid. ISDNERR_PROTOCOL_DL_FAILURE Data link parameters are invalid. ISDNERR_PROTOCOL_NS_FAILURE Network signaling parameters are invalid. ISDNERR_PROTOCOL_PH_FAILURE Physical layer parameters are invalid.

Details

This function starts the specified protocol on the board that is associated with the specified ctahd.

The run module is specified in the board keyword file. For information, refer to the NMS ISDN Installation Manual and the NMS OAM System User's Manual.

If the parms pointer is NULL, the default values for the specified protocol are used. It is assumed that an HDLC data stream was connected to the specified HDLC controller during initialization or by explicit switching calls. The parameters for the particular selected protocol are found in the parms data structure defined in isdnparm.h.

See also

isdnStopProtocol

Example 1

```
DWORD mystartisdn (CTAHD ctahd)   /* Use defaults */
{
CTA_EVENT event;
DWORD     ret;
char      errortext[40];
unsigned  nai = 0;

nai = 0;
ret = isdnStartProtocol( ctahd, ISDN_PROTOCOL_Q931CC,
ISDN_OPERATOR_NI2, COUNTRY_USA, ISDN_PARTNER_NT, nai, NULL);

if( ret != SUCCESS)
{
    ctaGetText(ctahd, ret, errortext, 40);
    printf("START_FAIL: %s\n", errortext );
    return MY_ERROR_START_FAILED;
}
myWaitForEvent( ctahd, &event);
if( event.value != SUCCESS)
{
    ctaGetText(ctahd, event.value, errortext, 40);
    printf("START_FAIL: %s\n", errortext) );
    return MY_ERROR_START_FAILED;
}
return SUCCESS ;
}
```

Example 2

```
DWORD mystartisdn (CTAHD ctahd)   /* User-specified parms */
{
CTA_EVENT event;
DWORD     ret;
char      errortext[40];
unsigned  nai;
unsigned  j;
ISDN_PROTOCOL_PARMS_Q931CC parms;

memset( parms, 0, sizeof parms);
parms.size = sizeof(ISDN_PROTOCOL_PARMS_Q931CC);
nai = 0;
j   = 0;

parms.nfas_group = 0;
parms.services_list[j++] = ACU_FAX_SERVICE;
parms.services_list[j++] = ACU_VOICE_SERIVCE;
parms.services.list[j]   = ACU_NO_SERVICE;
/*
** NOTE: The last service MUST contain ACU_NO_SERVICE
*/


ret = isdnStartProtocol( ctahd, ISDN_PROTOCOL_Q931CC,
ISDN_OPERATOR_NI2, COUNTRY_USA, ISDN_PARTNER_NT, nai, &parms);
if( ret != SUCCESS)
{
    ctaGetText(ctahd, ret, errortext, 40);
    printf("START_FAIL: %s\n", errortext );
    return MY_ERROR_START_FAILED;
}
myWaitForEvent( ctahd, &event)
if( event.value != SUCCESS)
{
    ctaGetText(ctahd, event.value, errortext, 40);
    printf("START_FAIL: %s\n", errortext );
    return MY_ERROR_START_FAILED;
}
return SUCCESS;
}
```

isdnStopProtocol

Shuts down a previously started ISDN protocol stack instance and releases all on-board resources and buffers used by the protocol stack instance.

Prototype

DWORD **isdnStopProtocol** (CTAHD ***ctahd***)

Argument	Description
ctahd	Context handle associated with a D channel, returned by **ctaCreateContext**.

Return values

Return value	Description
SUCCESS	
CTAERR_INVALID_CTAHD	The context handle is invalid.
CTAERR_INVALID_STATE	An ISDN protocol stack instance is: • Not started on the specified context handle, • Started • Stopped, or • In the process of stopping.

Events

Event name	Description
ISDNEVN_STOP_PROTOCOL	The value field of this event contains the completion status of the protocol stopping operation. If the instance stopped successfully, the value field contains SUCCESS. Otherwise, an error appears here.

Details

This function shuts down a previously started ISDN protocol stack instance and releases all HDLC LAPD or Q.931 ACU resources and buffers associated with the instance.

When the ISDN protocol is stopped, the RESTART procedure is initiated. This RESTART procedure implies sending RESTART messages to the remote end (a single RESTART for the entire trunk or one RESTART for each B channel, depending on the variant) and waiting for RESTART ACKNOWLEDGE.

The ISDN stack waits for each RESTART ACKNOWLEDGE message for 20 seconds before declaring the RESTART procedure failed, clearing its part of the state machine, and continuing the stopping procedure. As a result, if the remote end does not respond (for example, the RESTART procedure is not implemented or the line is down), stopping the protocols may take more than 20 seconds (up to 8 minutes for variants that do not support RESTART messages for the entire interface).

A behavior bit, NS_DISABLE_RESTART, can be used to disable the RESTART procedure in the stack. If this bit is set, the stack does not send RESTART messages when it is stopped, but it also does not respond to incoming RESTART messages. Do not set this bit unless you are certain that the remote end does not implement the RESTART procedure.

See also

isdnStartProtocol

9. Message primitives

Overview of message primitives

This section describes the types and contents of the messages interchanged between the NMS ISDN protocol stack and the application.

To send a message to the stack, the application specifies the message primitive in the ISDN_MESSAGE structure passed to **isdnSendMessage**. For more information, refer to Sending ISDN messages to the stack.

Events received from the stack contain message primitives. For more information, refer to Network access identifiers (NAIs).

Messages with the prefix ACU_ are valid only when the NMS ISDN software is in its ACU configuration, and the protocol stack is in ACU stack mode. Messages with the prefix DL_ are valid only when the NMS ISDN software is in its LAPD configuration, and the protocol stack is in LAPD stack mode.

LAPD primitives

The following table summarizes the supported LAPD primitives:

Message type	Description
DL_DA_IN	Indicates that the trunk from the remote party has received a packet of acknowledged data.
DL_DA_RQ	Requests that the stack transmit a packet of acknowledged data. Sent by the application to the NMS ISDN protocol stack.
DL_EST_CO	Indicates that equipment on the remote side of the ISDN trunk has acknowledged a SABME message sent by the ISDN protocol stack, and has sent back a UA message indicating that the data link is established.
DL_EST_IN	Indicates that the NMS ISDN protocol stack has received a SABME message. The stack automatically acknowledges the message by sending back a UA message, and establishes the data link.
DL_EST_RQ	Requests that the stack establish a data link. The stack transmits a SABME message over the trunk. Sent by the application to the NMS ISDN protocol stack.
DL_REL_IN	If the NMS ISDN software is in the IDLE state, this message indicates that an establishment request sent by the NMS ISDN protocol stack was not successfully answered. If the software is in the DATA_LINK_ESTABLISHED state, this message indicates that the data link has been broken.
DL_U_DA_IN	Indicates that the trunk from the remote party has received a packet of unacknowledged data.

Message type	Description
DL_U_DA_RQ	Requests that the stack transmit a packet of unacknowledged data. Sent by the application to the NMS ISDN protocol stack

ACU primitives summary

NMS supports the generic ISDN protocol, but no longer supports the Omnitel stack. The following table summarizes the supported ACU message primitives and indicates which primitives are deprecated:

Message type	Description
ACU_ALERT_IN	Indicates an alert.
ACU_ALERT_RQ	Sends an alert message.
ACU_CALLID_IN	Delivers a ***callid*** or reports a failure to an application.
ACU_CALL_PROC_IN	Indicates incoming call proceeding information (receipt of SETUP ACK, CALL PROCEEDING messages).
ACU_CALL_PROC_RQ	Requests events indicating the progress of a call.
ACU_CLEAR_CO	Confirms that the call has been released.
ACU_CLEAR_IN	Indicates the release of a call.
ACU_CLEAR_RQ	Requests release of a call.
ACU_CLEAR_RS	Indicates a response to a release indication.
ACU_CONN_CO	Indicates a call connection confirmation.
ACU_CONN_IN	Signals an incoming call.
ACU_CONN_RQ	Requests establishment of an outgoing call.
ACU_CONN_RS	Answers an incoming call.
ACU_D_CHANNEL_STATUS_IN	Indicates status of D channel.
ACU_D_CHANNEL_STATUS_RQ	Requests status of the D channel.
ACU_DIGIT_IN	Receives called number digits in overlap receiving mode.

Message type	Description
ACU_DIGIT_RQ	Requests called number digits in overlap receiving mode.
ACU_ERR_IN	Indicates an error.
ACU_FACILITY_IN	Indicates a specific facility.
ACU_FACILITY_RQ	Requests a specific facility.
ACU_NOTIFY_IN	Indicates that a NOTIFY message was received.
ACU_NOTIFY_RQ	Requests that a NOTIFY message be sent.
ACU_PROGRESS_IN	Indicates outgoing call progress information (receipt of PROGRESS message).
ACU_PROGRESS_RQ	Indicates progress request.
ACU_RESTART_IN	Indicates that a RESTART ACKNOWLEDGE message has been received.
ACU_SERVICE_CO	Indicates that a SERVICE ACKNOWLEDGE message has been received.
ACU_SERVICE_IN	Indicates that a SERVICE message has been received.
ACU_SERVICE_RQ	Requests that a SERVICE message be sent.
ACU_SETUP_ACK_IN	Acknowledges incoming call proceeding information.
ACU_SETUP_REPORT_IN	Signals that the ACU rejected or ignored an incoming call because it was not compatible (address or service filtering). The application must not answer this incoming call indication. This message is only for information purposes.
ACU_TRANSFER_CO	Indicates whether or not the attempted transfer was successful.
ACU_TRANSFER_RQ	Requests the transfer of two calls.

Using the ACU message reference

The topics in this section describe each of the supported ACU message primitives and include the following information:

Information	Description
Purpose	A short description of the purpose of each macro.
Conn_id	Contains the connection ID of the call that the message concerns.
Macro	The macros you can use to specify values for the information elements (IEs) in outgoing Q.931 messages. When the structure (and associated substructures) containing this data reaches the ISDN protocol stack, the stack rearranges the data into several IEs, builds a complete Q.931 message with the IEs, and sends it to the network. Filling order: Some pointer and size macros must be assigned values in sequence so that the IEs in outgoing messages are ordered correctly. If macros must be filled out in order, the filling order is specified in the protocol variants table of the message primitive. For example, when creating an ACU_ALERT_RQ message, the application must call the Acu_alert_rq_a_uui and Acu_alert_rq_uui_size macros before calling the Acu_alert_rq_a_display and Acu_alert_rq_display_size macros. In the protocol variants table, the Acu_alert_rq_a_uui and Acu_alert_rq_uui_size macros are marked Filling order: 1. The Acu_alert_rq_a_display and Acu_alert_rq_display_size macros are marked Filling order: 2.
Protocol variants	The variants under which each macro is supported. **Note:** Not all macros or macro combinations are supported for every variant. Macros supported under E10 are also supported under AT&T 5E9.
Macro descriptions	A short description of the macro.
Q.931 IE	The Q.931 information element in which the assigned value appears. **Note:** Some macros do not map directly to any information element, and several macros are not currently used at all. They are included in the reference table for informational purposes only.

ACU_ALERT_IN

This topic describes:

- ACU_ALERT_IN protocol variants
- ACU_ALERT_IN macro descriptions and Q.931 IE

Purpose

Indicates an alert.

Conn_id

An allocated call.

ACU_ALERT_IN protocol variants

The following table lists the variants under which each macro is supported for this primitive. In this table, x indicates both NT and TE sides.

Macro	4ESS	E10	NI2	DMS	ETSI	VN6	HKG	AUSTEL1	NTT	KOREA	TAIWAN	QSIG	DPNSS	T1607
Acu_alert_in_a_display, Acu_alert_in_display_size			x		x	x		x	x	x	x			x
Acu_alert_in_a_display_list														
Acu_alert_in_a_ext_parms														
Acu_alert_in_a_facility, Acu_alert_in_facility_size					x					x	x	x		
Acu_alert_in_a_pcs_user, Acu_alert_in_pcs_user_size						x								
Acu_alert_in_a_q931, Acu_alert_in_q931_size	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Acu_alert_in_a_redir_nb, Acu_alert_in_redir_nb_size	x			x										
Acu_alert_in_a_ss_cnip_name, Acu_alert_in_ss_cnip_name_size												x		
Acu_alert_in_a_uui, Acu_alert_in_uui_size	x	x			x	x			x	x	x			
Acu_alert_in_alert														
Acu_alert_in_call_ref_length			x											
Acu_alert_in_call_ref_value			x											
Acu_alert_in_data_chani	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Acu_alert_in_data_chani_excl													x	
Acu_alert_in_data_chani_nai													x	

Macro	**4ESS**	**E10**	**NI2**	**DMS**	**ETSI**	**VN6**	**HKG**	**AUSTEL1**	**NTT**	**KOREA**	**TAIWAN**	**QSIG**	**DPNSS**	**T1607**
Acu_alert_in_data_chani_nb														
Acu_alert_in_data_chani_tab[***i***]														
Acu_alert_in_data_chani_tab_nai[***i***]														
Acu_alert_in_ext_parms_lgth														
Acu_alert_in_ext_parms_nb														
Acu_alert_in_pcs_user_protocol						x								
Acu_alert_in_progress_descr_x [***i***]														
Acu_alert_in_progress_description		x	x	x	x	x			x	x	x			x
Acu_alert_in_progress_ind_nb		x	x	x	x	x			x	x	x			x
Acu_alert_in_progress_loc_x [***i***]														
Acu_alert_in_progress_location		x	x	x	x	x			x	x	x			x
Acu_alert_in_redir_nb_plan														
Acu_alert_in_redir_nb_pres	x			x										
Acu_alert_in_redir_nb_reason	x			x										
Acu_alert_in_redir_nb_screen	x			x										
Acu_alert_in_redir_nb_type	x			x										
Acu_alert_in_signal_val			x		x									
Acu_alert_in_ss_cnip_name_active												x		
Acu_alert_in_ss_cnip_name_pres												x		
Acu_alert_in_x_display_nb			x		x	x		x	x	x	x			

Macro	4ESS	E10	NI2	DMS	ETSI	VN6	HKG	AUSTEL1	NTT	KOREA	TAIWAN	QSIG	DPNSS	T1607
Acu_alert_in_x_display_size[*i*]			x		x	x		x	x	x	x			
Acu_alert_in_x_display_total_size			x		x	x		x	x	x	x			
Acu_alert_in_x_display_type[*i*]			x		x	x		x	x	x	x			
Acu_alert_in_x_p_display[*i*]			x		x	x		x	x	x	x			

ACU_ALERT_IN macro descriptions and Q.931 IE

Macro	Description	Q.931 IE
Acu_alert_in_a_display, Acu_alert_in_display_size	Pointer to (and size of) buffer containing ISDN display information (optional).	Display
Acu_alert_in_a_display_list	Pointer to display structure.	Display
Acu_alert_in_a_ext_parms	Pointer to buffer containing extended parameters.	Not used.
Acu_alert_in_a_facility, Acu_alert_in_facility_size	Pointer to (and size of) buffer containing ISDN facility information (optional).	Facility
Acu_alert_in_a_pcs_user, Acu_alert_in_pcs_user_size	Pointer to (and size of) buffer containing ISDN PCS-to-user/user-to-PCS information. See PCS-user information elements for more information on this IE.	PCS-user
Acu_alert_in_a_q931, Acu_alert_in_q931_size	Pointer to (and size of) buffer containing raw data of incoming Q.931 message.	Does not map to an IE.
Acu_alert_in_a_redir_nb, Acu_alert_in_redir_nb_size	Pointer to (and size of) buffer containing redirecting number.	Redirecting number
Acu_alert_in_a_ss_cnip_name, Acu_alert_in_ss_cnip_name_size	Pointer to (and size of) buffer containing calling name identification presentation (CNIP) name.	CNIP

Macro	**Description**	**Q.931 IE**
Acu_alert_in_a_uui, Acu_alert_in_uui_size	Pointer to (and size of) buffer containing ISDN user-to-user information (optional).	User-user
Acu_alert_in_alert	Code indicating which alert was detected. Used when the behavior bit CC_SEND_ALERT_IN is used. Valid values include: ACUAC_RING: Ring detected ACUAC_REMOTE_ALERTED: Remote equipment alerted	Not used.
Acu_alert_in_call_ref_length	Length of call reference value.	Call reference
Acu_alert_in_call_ref_value	Call reference value and call reference flag.	Call reference
Acu_alert_in_data_chani	Data channel to use (B1, B2, ..., D).	Channel identification
Acu_alert_in_data_chani_excl	Channel IDs are preferred (OFF) or exclusive (ON).	Channel identification
Acu_alert_in_data_chani_nai	NAI.	Channel identification
Acu_alert_in_data_chani_nb	Number of channel IDs in the chani_list field. If no channel IDs, use 0.	Channel identification
Acu_alert_in_data_chani_tab[***i***]	Channel ID ***i***.	Channel identification
Acu_alert_in_data_chani_tab_nai[***i***]	NAI ***i***.	Channel identification
Acu_alert_in_ext_parms_lgth	Total length of buffer containing extended parameters.	Not used.
Acu_alert_in_ext_parms_nb	Number of parameters in buffer containing extended parameters.	Not used.

Macro	**Description**	**Q.931 IE**
Acu_alert_in_pcs_user_protocol	Protocol discriminator for the pcs_user information element. See PCS-user information elements for more information on this IE.	PCS-user
Acu_alert_in_progress_descr_x [*i*]	Description for information element *i*. See Progress description values for a list of valid values.	Progress indicator
Acu_alert_in_progress_description	Description for information element 0. See Progress description values for a list of valid values.	Progress indicator
Acu_alert_in_progress_ind_nb	Number of progress indication information elements.	Progress indicator
Acu_alert_in_progress_loc_x [*i*]	Location of information element *i*. See Location values for a list of valid values.	Progress indicator
Acu_alert_in_progress_location	Location of information element 0. See Location values for a list of valid values.	Progress indicator
Acu_alert_in_redir_nb_plan	Redirecting number plan. See Plan values for a list of valid values.	Redirecting number
Acu_alert_in_redir_nb_pres	Redirecting number presentation. Allowed values include: N_PRES_ALLOWED: Presentation allowed N_PRES_RESTRICTED: Presentation restricted N_PRES_NOT_AVAILABLE: Presentation not available	Redirecting number
Acu_alert_in_redir_nb_reason	Reason for redirection. See Redirecting reason values for a list of valid values.	Redirecting number
Acu_alert_in_redir_nb_screen	Redirecting number screening indicator. See Screening indicator values for a list of valid values.	Redirecting number
Acu_alert_in_redir_nb_type	Redirecting number type. See Number type values for a list of valid values.	Redirecting number

Macro	Description	Q.931 IE
Acu_alert_in_signal_val	Signal value. See Signal values for a list of valid values.	Does not map to an IE.
Acu_alert_in_ss_cnip_name_active	Indicates calling name identification presentation (CNIP) supplementary service should be invoked.	CNIP
Acu_alert_in_ss_cnip_name_pres	Calling name identification presentation (CNIP) mode. Allowed values include: N_PRES_ALLOWED: Presentation allowed N_PRES_RESTRICTED: Presentation restricted N_PRES_NOT_AVAILABLE: Presentation not available	CNIP
Acu_alert_in_x_display_nb	Number of present occurrences.	Display
Acu_alert_in_x_display_size[***i***]	Size of occurrence ***i*** (optional).	Display
Acu_alert_in_x_display_total_size	Total size of the stored strings.	Display
Acu_alert_in_x_display_type[***i***]	Type of display ***i***.	Display
Acu_alert_in_x_p_display[***i***]	Pointer to occurrence ***i***.	Display

ACU_ALERT_RQ

This topic describes:

- ACU_ALERT_RQ protocol variants
- ACU_ALERT_RQ macro descriptions and Q.931 IE

Note: Address parameters must be set according to the filling order. If one of them is unused, its size field must be set to 0.

Purpose

Sends an alert message.

Conn_id

An allocated call.

ACU_ALERT_RQ protocol variants

The following table lists the variants under which each macro is supported for this primitive. In this table, x indicates both NT and TE sides.

Macro	4ESS	E10	NI2	DMS	ETSI	VN6	HKG	AUSTEL1	NTT	KOREA	TAIWAN	QSIG	DPNSS	T1607
Acu_alert_rq_a_display, Acu_alert_rq_display_size Filling order: 2														
Acu_alert_rq_a_display_list										x	x			
Acu_alert_rq_a_ext_parms														
Acu_alert_rq_a_pcs_user, Acu_alert_rq_pcs_user_size Filling order: 3						x								
Acu_alert_rq_a_ss_cnip_name, Acu_alert_rq_ss_cnip_name_size Filling order: 4												x		
Acu_alert_rq_a_tsp_ie_list, Acu_alert_rq_tsp_ie_list_size Filling order: 5	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Acu_alert_rq_a_uui, Acu_alert_rq_uui_size Filling order: 1					x	x			x	x	x			
Acu_alert_rq_data_chani	x	x	x	x	x	x	x	x	x	x	x	x		x
Acu_alert_rq_data_chani_excl	x	x	x	x	x	x	x	x	x	x	x	x		x
Acu_alert_rq_data_chani_nai														
Acu_alert_rq_data_chani_nb	x	x	x											x
Acu_alert_rq_data_chani_tab	x	x	x											x
Acu_alert_rq_data_chani_tab_nai														
Acu_alert_rq_ext_parms_lgth														

Macro	**4ESS**	**E10**	**NI2**	**DMS**	**ETSI**	**VN6**	**HKG**	**AUSTEL1**	**NTT**	**KOREA**	**TAIWAN**	**QSIG**	**DPNSS**	**T1607**
Acu_alert_rq_ext_parms_nb														
Acu_alert_rq_pcs_user_protocol						x								
Acu_alert_rq_progress_descr_x [***i***]														x
Acu_alert_rq_progress_description														x
Acu_alert_rq_progress_ind_nb														x
Acu_alert_rq_progress_loc_x [***i***]														x
Acu_alert_rq_progress_location														x
Acu_alert_rq_signal_val														
Acu_alert_rq_ss_cnip_name_active												x		
Acu_alert_rq_ss_cnip_name_pres												x		
Acu_alert_rq_uui_protocol										x	x			
Acu_alert_rq_x_display_nb														
Acu_alert_rq_x_display_size[***i***]														
Acu_alert_rq_x_display_total_size														
Acu_alert_rq_x_display_type[***i***]														
Acu_alert_rq_x_p_display[***i***]														

ACU_ALERT_RQ macro descriptions and Q.931 IE

Macro	**Description**	**Q.931 IE**
Acu_alert_rq_a_display, Acu_alert_rq_display_size Filling order: 2	Pointer to (and size of) buffer containing ISDN display info (optional).	Display

Macro	Description	Q.931 IE
Acu_alert_rq_a_display_list	Pointer to display structure.	Display
Acu_alert_rq_a_ext_parms	Pointer to buffer containing extended parameters.	Not used.
Acu_alert_rq_a_pcs_user, Acu_alert_rq_pcs_user_size Filling order: 3	Pointer to (and size of) buffer containing ISDN PCS-to-user/user-to-PCS information. See PCS-user information elements for more information on this IE.	PCS-user
Acu_alert_rq_a_ss_cnip_name, Acu_alert_rq_ss_cnip_name_size Filling order: 4	Pointer to (and size of) buffer containing calling name identification presentation (CNIP) name.	CNIP
Acu_alert_rq_a_tsp_ie_list, Acu_alert_rq_tsp_ie_list_size Filling order: 5	Pointer to (and size of) transparent IE buffer.	Does not map to an IE.
Acu_alert_rq_a_uui, Acu_alert_rq_uui_size Filling order: 1	Pointer to (and size of) buffer containing ISDN user-to-user information (optional).	User-user
Acu_alert_rq_data_chani	Data channel to use (B1, B2, ... D).	Channel identification
Acu_alert_rq_data_chani_excl	Channel IDs are preferred (OFF) or exclusive (ON).	Channel identification
Acu_alert_rq_data_chani_nai	NAI.	Channel identification
Acu_alert_rq_data_chani_nb	Number of channel IDs in the chani_list field. If no channel IDs, use 0.	Channel identification
Acu_alert_rq_data_chani_tab	Channel ID.	Channel identification
Acu_alert_rq_data_chani_tab_nai	NAI.	Channel identification
Acu_alert_rq_ext_parms_lgth	Total length of buffer containing extended parameters.	Not used.
Acu_alert_rq_ext_parms_nb	Number of parameters in buffer containing extended parameters.	Not used.

Macro	Description	Q.931 IE
Acu_alert_rq_pcs_user_protocol	Protocol discriminator for the pcs_user information element. See PCS-user information elements for more information on this IE.	PCS-user
Acu_alert_rq_progress_descr_x [***i***]	Description for information element ***i***. See Progress description values for a list of valid values.	Progress indicator
Acu_alert_rq_progress_description	Description for information element 0. See Progress description values for a list of valid values.	Progress indicator
Acu_alert_rq_progress_ind_nb	Number of progress indication information elements.	Progress indicator
Acu_alert_rq_progress_loc_x [***i***]	Location of information element ***i***. See Location values for a list of valid values and default setting information.	Progress indicator
Acu_alert_rq_progress_location	Location of information element 0. See Location values for a list of valid values and default setting information.	Progress indicator
Acu_alert_rq_signal_val	Signal value. See Signal values for a list of valid values.	Not used.
Acu_alert_rq_ss_cnip_name_active	Indicates calling name identification presentation (CNIP) supplementary service should be invoked.	CNIP
Acu_alert_rq_ss_cnip_name_pres	Calling name identification presentation mode. Allowed values: N_PRES_ALLOWED: Presentation allowed N_PRES_RESTRICTED: Presentation restricted N_PRES_NOT_AVAILABLE: Presentation not available	CNIP
Acu_alert_rq_uui_protocol	UUI protocol discriminator value. See UUI protocol discriminator values for a list of valid values. Default: UUI_IA5 (UUI_USER_SPF for HKG variant).	User-user
Acu_alert_rq_x_display_nb	Number of present occurrences.	Display

Macro	Description	Q.931 IE
Acu_alert_rq_x_display_size[***i***]	Size of occurrence ***i*** (optional).	Display
Acu_alert_rq_x_display_total_size	Total size of the stored strings.	Display
Acu_alert_rq_x_display_type[***i***]	Type of display ***i***.	Display
Acu_alert_rq_x_p_display[***i***]	Pointer to occurrence ***i***.	Display

ACU_CALLID_IN

This topic describes:

- ACU_CALLID_IN protocol variants
- ACU_CALLID_IN macro descriptions and Q.931 IE

Purpose

Delivers a ***callid*** or reports a failure to an application.

Conn_id

An allocated call. TE side only.

ACU_CALLID_IN protocol variants

The following table lists the variants under which each macro is supported for this primitive. In this table, x indicates TE side only.

Macro	4ESS	E10	NI2	DMS	ETSI	VN6	HKG	AUSTEL1	NTT	KOREA	TAIWAN	QSIG	DPNSS	T1607
Acu_transfer_status			x	x	x							x	x	
Acu_transfer_callid_present			x	x	x							x	x	
Acu_transfer_callid			x	x	x							x	x	
Acu_transfer_size			x	x	x							x	x	

ACU_CALLID_IN macro descriptions and Q.931 IE

Macro	Description	Q.931 IE
Acu_transfer_status	Type of result. A value of 0 indicates success; any other value indicates an error.	NA

Macro	Description	Q.931 IE
Acu_transfer_callid_present	A value of 1 indicates that the ***callid*** field contains valid information.	NA
Acu_transfer_callid	Information used to identify a call.	NA
Acu_transfer_size	The size of the structure.	NA

ACU_CALL_PROC_IN

This topic describes:

- ACU_CALL_PROC_IN protocol variants
- ACU_CALL_PROC_IN macro descriptions and Q.931 IE

Purpose

Indicates incoming call proceeding information (receipt of SETUP ACK, CALL PROCEEDING messages).

Conn_id

An allocated call.

ACU_CALL_PROC_IN protocol variants

The following table lists the variants under which each macro is supported for this primitive. In this table, x indicates both NT and TE sides.

Macro	4ESS	E10	NI2	DMS	ETSI	VN6	HKG	AUSTEL1	NTT	KOREA	TAIWAN	QSIG	DPNSS	T1607
Acu_call_proc_in_a_display, Acu_call_proc_in_display_size			x		x	x		x	x	x	x			x
Acu_call_proc_in_a_display_list			x		x	x		x	x	x	x			x
Acu_call_proc_in_a_ext_parms														
Acu_call_proc_in_a_facility, Acu_call_proc_in_facility_size					x					x	x			
Acu_call_proc_in_a_pcs_user, Acu_call_proc_in_pcs_user_size														

Macro	**4ESS**	**E10**	**NI2**	**DMS**	**ETSI**	**VN6**	**HKG**	**AUSTEL1**	**NTT**	**KOREA**	**TAIWAN**	**QSIG**	**DPNSS**	**T1607**
Acu_call_proc_in_a_q931, Acu_call_proc_in_q931_size	x	x	x	x	x	x	x	x	x	x	x	x		x
Acu_call_proc_in_call_ref_length			x											
Acu_call_proc_in_call_ref_value			x											
Acu_call_proc_in_cause														
Acu_call_proc_in_data_chani	x	x	x	x	x	x	x	x	x	x	x	x		x
Acu_call_proc_in_data_chani_excl														
Acu_call_proc_in_data_chani_nai														
Acu_call_proc_in_data_chani_nb														
Acu_call_proc_in_data_chani_tab[***i***]														
Acu_call_proc_in_data_chani_tab_nai[***i***]														
Acu_call_proc_in_ext_parms_lgth														
Acu_call_proc_in_ext_parms_nb														
Acu_call_proc_in_progress_descr_x [***i***]														
Acu_call_proc_in_progress_description					x	x				x	x			x
Acu_call_proc_in_progress_ind_nb					x	x				x	x			x
Acu_call_proc_in_progress_loc_x [***i***]														
Acu_call_proc_in_progress_location					x	x				x	x			x
Acu_call_proc_in_signal_val			x											
Acu_call_proc_in_x_display_nb			x		x	x		x	x	x	x			
Acu_call_proc_in_x_display_size[***i***]			x		x	x		x	x	x	x			

Macro	4ESS	E10	NI2	DMS	ETSI	VN6	HKG	AUSTEL1	NTT	KOREA	TAIWAN	QSIG	DPNSS	T1607
Acu_call_proc_in_x_display_total_size			x		x	x		x	x	x	x			
Acu_call_proc_in_x_display_type[***i***]			x		x	x		x	x	x	x			
Acu_call_proc_in_x_p_display[***i***]			x		x	x		x	x	x	x			

ACU_CALL_PROC_IN macro descriptions and Q.931 IE

Macro	Description	Q.931 IE
Acu_call_proc_in_a_display, Acu_call_proc_in_display_size	Pointer to (and size of) buffer containing ISDN display information (optional).	Display
Acu_call_proc_in_a_display_list	Pointer to display structure.	Display
Acu_call_proc_in_a_ext_parms	Pointer to buffer containing extended parameters.	Not used.
Acu_call_proc_in_a_facility, Acu_call_proc_in_facility_size	Pointer to (and size of) buffer containing ISDN facility information (optional).	Facility
Acu_call_proc_in_a_pcs_user, Acu_call_proc_in_pcs_user_size	Pointer to (and size of) buffer containing ISDN PCS-to-user/user-to-PCS information. See PCS-user information elements for more information on this IE.	PCS-User
Acu_call_proc_in_a_q931, Acu_call_proc_in_q931_size	Pointer to (and size of) buffer containing raw data of incoming Q.931 message.	Does not map to an IE.

Macro	Description	Q.931 IE
Acu_call_proc_in_call_ref_length	Length of call reference value.	Call reference
Acu_call_proc_in_call_ref_value	Call reference value and call reference flag.	Call reference
Acu_call_proc_in_cause	Cause value.	Cause
Acu_call_proc_in_data_chani	Data channel to use (B1, B2, ... D).	Channel identification
Acu_call_proc_in_data_chani_excl	Channel IDs are preferred (OFF) or exclusive (ON).	Channel identification
Acu_call_proc_in_data_chani_nai	NAI.	Channel identification
Acu_call_proc_in_data_chani_nb	Number of channel IDs in the chani_list field. If no channel IDs, use 0.	Channel identification
Acu_call_proc_in_data_chani_tab[***i***]	Channel ID ***i***.	Channel identification
Acu_call_proc_in_data_chani_tab_nai[***i***]	NAI ***i***.	Channel identification
Acu_call_proc_in_ext_parms_lgth	Total length of buffer containing extended parameters.	Not used.
Acu_call_proc_in_ext_parms_nb	Number of parameters in buffer containing extended parameters.	Not used.
Acu_call_proc_in_progress_descr_x [***i***]	Description for information element ***i***. See Progress description values for a list of valid values.	Progress indicator
Acu_call_proc_in_progress_description	Description for information element 0. See Progress description values for a list of valid values.	Progress indicator

Macro	**Description**	**Q.931 IE**
Acu_call_proc_in_progress_ind_nb	Number of progress indication information elements.	Progress indicator
Acu_call_proc_in_progress_loc_x [***i***]	Location of information element ***i***. See Location values for a list of valid values.	Progress indicator
Acu_call_proc_in_progress_location	Location of information element 0. See Location values for a list of valid values.	Progress indicator
Acu_call_proc_in_signal_val	Signal value. See Signal values for a list of valid values.	Signal value
Acu_call_proc_in_x_display_nb	Number of present occurrences.	Display
Acu_call_proc_in_x_display_size[***i***]	Size of occurrence ***i*** (optional).	Display
Acu_call_proc_in_x_display_total_size	Total size of the stored strings.	Display
Acu_call_proc_in_x_display_type[***i***]	Type of display ***i***.	Display
Acu_call_proc_in_x_p_display[***i***]	Pointer to occurrence ***i***.	Display

ACU_CALL_PROC_RQ

This topic describes:

- ACU_CALL_PROC_RQ protocol variants
- ACU_CALL_PROC_RQ macro descriptions and Q.931 IE

Note: Address parameters must be set according to the filling order. If one of them is unused, its size field must be set to 0.

Purpose

Requests events indicating the progress of a call.

Conn_id

An allocated call.

ACU_CALL_PROC_RQ protocol variants

The following table lists the variants under which each macro is supported for this primitive. In this table, x indicates both NT and TE sides.

Macro	4ESS	E10	NI2	DMS	ETSI	VN6	HKG	AUSTEL1	NTT	KOREA	TAIWAN	QSIG	DPNSS	T1607
Acu_call_proc_rq_a_display, Acu_call_proc_rq_display_size Filling order: 1					x									
Acu_call_proc_rq_a_display_list					x					x	x			
Acu_call_proc_rq_a_ext_parms														
Acu_call_proc_rq_a_pcs_user, Acu_call_proc_rq_pcs_user_size Filling order: 2														
Acu_call_proc_rq_a_tsp_ie_list, Acu_call_proc_rq_tsp_ie_list_size Filling order: 3	x	x	x	x	x	x	x	x	x	x	x	x		x
Acu_call_proc_rq_data_chani	x	x	x	x	x	x	x	x	x	x	x	x		x
Acu_call_proc_rq_data_chani_excl										x	x	x		
Acu_call_proc_rq_data_chani_nai														
Acu_call_proc_rq_data_chani_nb	x	x	x	x	x	x	x	x	x					x
Acu_call_proc_rq_data_chani_tab[***i***]														
Acu_call_proc_rq_data_chani_tab_nai[***i***]														
Acu_call_proc_rq_ext_parms_lgth														
Acu_call_proc_rq_ext_parms_nb														
Acu_call_proc_rq_progress_description				x	x	x				x	x			x
Acu_call_proc_rq_progress_ind_nb				x	x	x				x	x			x

Macro	4ESS	E10	NI2	DMS	ETSI	VN6	HKG	AUSTEL1	NTT	KOREA	TAIWAN	QSIG	DPNSS	T1607
Acu_call_proc_rq_progress_location				x	x	x				x	x			x
Acu_call_proc_rq_x_display_nb					x									
Acu_call_proc_rq_x_display_size[***i***]					x									
Acu_call_proc_rq_x_display_total_size					x									
Acu_call_proc_rq_x_display_type[***i***]					x									
Acu_call_proc_rq_x_p_display[***i***]					x									

ACU_CALL_PROC_RQ macro descriptions and Q.931 IE

Macro	Description	Q.931 IE
Acu_call_proc_rq_a_display, Acu_call_proc_rq_display_size Filling order: 1	Pointer to (and size of) buffer containing ISDN display information (optional).	Display
Acu_call_proc_rq_a_display_list	Pointer to display structure.	Display
Acu_call_proc_rq_a_ext_parms	Pointer to buffer containing extended parameters.	Not used.
Acu_call_proc_rq_a_pcs_user, Acu_call_proc_rq_pcs_user_size Filling order: 2	Pointer to (and size of) buffer containing ISDN PCS-to-user/user-to-PCS information. See PCS-user information elements for more information on this IE.	PCS-user
Acu_call_proc_rq_a_tsp_ie_list, Acu_call_proc_rq_tsp_ie_list_size Filling order: 3	Pointer to (and size of) transparent IE buffer.	Does not map to an IE.
Acu_call_proc_rq_data_chani	Data channel to use (B1, B2, ... D).	Channel identification

Macro	Description	Q.931 IE
Acu_call_proc_rq_data_chani_excl	Channel IDs are preferred (OFF) or exclusive (ON).	Channel identification
Acu_call_proc_rq_data_chani_nai	NAI.	Channel identification
Acu_call_proc_rq_data_chani_nb	Number of channel IDs in the chani_list field. If no channel IDs, use 0.	Channel identification
Acu_call_proc_rq_data_chani_tab[***i***]	Channel ID ***i***.	Channel identification
Acu_call_proc_rq_data_chani_tab_nai[***i***]	NAI ***i***.	Channel identification
Acu_call_proc_rq_ext_parms_lgth	Total length of buffer containing extended parameters.	Not used.
Acu_call_proc_rq_ext_parms_nb	Number of parameters in buffer containing extended parameters.	Not used.
Acu_call_proc_rq_progress_description	Description for information element 0. See Progress description values for a list of valid values.	Progress indicator
Acu_call_proc_rq_progress_ind_nb	Number of progress indication information elements.	Progress indicator
Acu_call_proc_rq_progress_location	Location of information element 0. See Location values for a list of valid values and default setting information.	Progress indicator
Acu_call_proc_rq_x_display_nb	Number of present occurrences.	Display
Acu_call_proc_rq_x_display_size[***i***]	Size of occurrence ***i*** (optional).	Display
Acu_call_proc_rq_x_display_total_size	Total size of the stored strings.	Display
Acu_call_proc_rq_x_display_type[***i***]	Type of display ***i***.	Display
Acu_call_proc_rq_x_p_display[***i***]	Pointer to occurrence ***i***.	Display

ACU_CLEAR_CO

This topic describes:

- ACU_CLEAR_CO protocol variants
- ACU_CLEAR_CO macro descriptions and Q.931 IE

Purpose

Confirms release.

Conn_id

An allocated call.

ACU_CLEAR_CO protocol variants

The following table lists the variants under which each macro is supported for this primitive. In this table, TE indicates the TE side only, and x indicates both NT and TE sides.

Macro	4ESS	E10	NI2	DMS	ETSI	VN6	HKG	AUSTEL1	NTT	KOREA	TAIWAN	QSIG	DPNSS	T1607
Acu_clear_co_a_display, Acu_clear_co_display_size			x		x	x		x	x	x	x			x
Acu_clear_co_a_display_list			x		x	x		x	x	x	x			x
Acu_clear_co_a_ext_parms														
Acu_clear_co_a_facility, Acu_clear_co_facility_size					x	x				x	x	x		
Acu_clear_co_a_pcs_user, Acu_clear_co_pcs_user_size						x								
Acu_clear_co_a_q931, Acu_clear_co_q931_size	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Acu_clear_co_a_uui, Acu_clear_co_uui_size			x		x	x			x	x	x			
Acu_clear_co_charging_available						TE								
Acu_clear_co_charging_multi														
Acu_clear_co_charging_period														

Macro	**4ESS**	**E10**	**NI2**	**DMS**	**ETSI**	**VN6**	**HKG**	**AUSTEL1**	**NTT**	**KOREA**	**TAIWAN**	**QSIG**	**DPNSS**	**T1607**
Acu_clear_co_charging_type						TE								
Acu_clear_co_charging_value						TE								
Acu_clear_co_ext_parms_lgth														
Acu_clear_co_ext_parms_nb														
Acu_clear_co_network_cause	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Acu_clear_co_network_cause_loc	x	x	x	x	x	x	x	x	x	x	x	x		x
Acu_clear_co_pcs_user_protocol						x								
Acu_clear_co_ret_code	x	x	x	x	x	x	x	x	x			x		x
Acu_clear_co_total_cost			x		x	x			x					
Acu_clear_co_uui_protocol										x	x			
Acu_clear_co_x_display_nb			x		x	x		x	x	x	x			x
Acu_clear_co_x_display_size[***i***]			x		x	x		x	x	x	x			x
Acu_clear_co_x_display_total_size			x		x	x		x	x	x	x			x
Acu_clear_co_x_display_type[***i***]			x		x	x		x	x	x	x			x
Acu_clear_co_x_p_display[***i***]			x		x	x		x	x	x	x			x

ACU_CLEAR_CO macro descriptions and Q.931 IE

Macro	**Description**	**Q.931 IE**
Acu_clear_co_a_display, Au_clear_co_display_size	Pointer to (and size of) buffer containing ISDN display information (optional).	Display
Acu_clear_co_a_display_list	Pointer to display structure.	Display

Macro	Description	Q.931 IE
Acu_clear_co_a_ext_parms	Pointer to buffer containing extended parameters.	Not used.
Acu_clear_co_a_facility, Acu_clear_co_facility_size	Pointer to (and size of) buffer containing ISDN facility information (optional).	Facility
Acu_clear_co_a_pcs_user, Acu_clear_co_pcs_user_size	Pointer to (and size of) buffer containing ISDN PCS-to-user/user-to-PCS information. See PCS-user information elements for more information on this IE.	PCS-User
Acu_clear_co_a_q931, Acu_clear_co_q931_size	Pointer to (and size of) buffer containing raw data of incoming Q.931 message.	Does not map to an IE.
Acu_clear_co_a_uui, Acu_clear_co_uui_size	Pointer to (and size of) buffer containing ISDN user-to-user information (optional).	User-user
Acu_clear_co_charging_available	Indicates that charging information is available.	Does not map to an IE.
Acu_clear_co_charging_multi	Charging multiplier.	Not used.
Acu_clear_co_charging_period	Charging period.	Not used.
Acu_clear_co_charging_type	Charging type. See Charging type values for a list of valid values.	National facility
Acu_clear_co_charging_value	Charging value (number of units).	National facility
Acu_clear_co_ext_parms_lgth	Total length of buffer containing extended parameters.	Not used.
Acu_clear_co_ext_parms_nb	Number of parameters in buffer containing extended parameters.	Not used.
Acu_clear_co_network_cause	Network-provided clear cause value. See Network-provided clearing cause values for a list of valid values.	Cause
Acu_clear_co_network_cause_loc	Network-provided clear cause location.	Cause

Macro	Description	Q.931 IE
Acu_clear_co_pcs_user_protocol	Protocol discriminator for the pcs_user information element. See PCS-user information elements for more information on this IE.	PCS-user
Acu_clear_co_ret_code	Return code. See Clear code values for a list of valid values.	Does not map to an IE.
Acu_clear_co_total_cost	Complete charging information sent by the network (optional).	Not used.
Acu_clear_co_uui_protocol	UUI protocol discriminator value. See UUI protocol discriminator values for a list of valid values.	User-User
Acu_clear_co_x_display_nb	Number of present occurrences.	Display
Acu_clear_co_x_display_size[***i***]	Size of occurrence ***i*** (optional).	Display
Acu_clear_co_x_display_total_size	Total size of the stored strings.	Display
Acu_clear_co_x_display_type[***i***]	Type of display ***i***.	Display
Acu_clear_co_x_p_display[***i***]	Pointer to occurrence ***i***.	Display

ACU_CLEAR_IN

This topic describes:

- ACU_CLEAR_IN protocol variants
- ACU_CLEAR_IN macro descriptions and Q.931 IE

Purpose

Indicates call release.

Conn_id

An allocated call.

ACU_CLEAR_IN protocol variants

The following table lists the variants under which each macro is supported for this primitive. In this table, TE indicates the TE side only, and x indicates both NT and TE sides.

Macro	**4ESS**	**E10**	**NI2**	**DMS**	**ETSI**	**VN6**	**HKG**	**AUSTEL1**	**NTT**	**KOREA**	**TAIWAN**	**QSIG**	**DPNSS**	**T1607**
Acu_clear_in_a_display, Acu_clear_in_display_size			x		x	x	x	x	x	x	x			x
Acu_clear_in_a_display_list			x		x	x	x	x	x	x	x			x
Acu_clear_in_a_ext_parms														
Acu_clear_in_a_facility, Acu_clear_in_facility_size					x	x				x	x	x		
Acu_clear_in_a_pcs_user, Acu_clear_in_pcs_user_size						x								
Acu_clear_in_a_q931, Acu_clear_in_q931_size	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Acu_clear_in_a_ss_cnip_name, Acu_clear_in_ss_cnip_name_size												x		
Acu_clear_in_a_uui, Acu_clear_in_uui_size	x	x	x		x	x	x		x	x	x			
Acu_clear_in_charging						TE								
Acu_clear_in_charging_multi														
Acu_clear_in_charging_period														
Acu_clear_in_charging_type						TE								
Acu_clear_in_charging_value						TE								
Acu_clear_in_ext_parms_lgth														
Acu_clear_in_ext_parms_nb														
Acu_clear_in_network_cause	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Acu_clear_in_pcs_user_protocol						x								
Acu_clear_in_progress_description					x	x	x	x	x	x	x			

Macro	4ESS	E10	NI2	DMS	ETSI	VN6	HKG	AUSTEL1	NTT	KOREA	TAIWAN	QSIG	DPNSS	T1607
Acu_clear_in_progress_ind_nb					x	x	x	x	x	x	x			
Acu_clear_in_progress_location					x	x	x	x	x	x	x			
Acu_clear_in_ret_code														
Acu_clear_in_signal_val														
Acu_clear_in_ss_cnip_name_active												x		
Acu_clear_in_ss_cnip_name_pres										x	x	x		
Acu_clear_in_total_cost														
Acu_clear_in_x_display_nb			x		x	x	x	x	x	x	x			x
Acu_clear_in_x_display_size[***i***]			x		x	x	x	x	x	x	x			x
Acu_clear_in_x_display_total_size			x		x	x	x	x	x	x	x			x
Acu_clear_in_x_display_type[***i***]			x		x	x	x	x	x	x	x			x
Acu_clear_in_x_p_display[***i***]			x		x	x	x	x	x	x	x			x

ACU_CLEAR_IN macro descriptions and Q.931 IE

Macro	Description	Q.931 IE
Acu_clear_in_a_display, Acu_clear_in_display_size	Pointer to (and size of) buffer containing ISDN display information (optional).	Display
Acu_clear_in_a_display_list	Pointer to display structure.	Display
Acu_clear_in_a_ext_parms	Pointer to buffer containing extended parameters.	Not used.
Acu_clear_in_a_facility, Acu_clear_in_facility_size	Pointer to (and size of) buffer containing ISDN facility information (optional).	Facility

Macro	Description	Q.931 IE
Acu_clear_in_a_pcs_user, Acu_clear_in_pcs_user_size	Pointer to (and size of) buffer containing ISDN PCS-to-user/user-to-PCS information. See PCS-user information elements for more information on this IE.	PCS-user
Acu_clear_in_a_q931, Acu_clear_in_q931_size	Pointer to (and size of) buffer containing raw data of incoming Q.931 message.	Does not map to an IE.
Acu_clear_in_a_ss_cnip_name, Acu_clear_in_ss_cnip_name_size	Pointer to (and size of) buffer containing calling name identification presentation (CNIP) name.	CNIP
Acu_clear_in_a_uui, Acu_clear_in_uui_size	Pointer to (and size of) buffer containing ISDN user-to-user information (optional).	User-user
Acu_clear_in_charging	Charging value (number of units).	Not used.
Acu_clear_in_charging_multi	Charging multiplier.	Not used.
Acu_clear_in_charging_period	Charging period.	Not used.
Acu_clear_in_charging_type	Charging type. See Charging type values for a list of valid values.	Not used.
Acu_clear_in_charging_value	Charging value (number of units).	Not used.
Acu_clear_in_ext_parms_lgth	Total length of buffer containing extended parameters.	Not used.
Acu_clear_in_ext_parms_nb	Number of parameters in buffer containing extended parameters.	Not used.
Acu_clear_in_network_cause	Network-provided clear cause value. See Network-provided clearing cause values for a list of valid values.	Cause
Acu_clear_in_pcs_user_protocol	Protocol discriminator for the pcs_user information element. See PCS-user information elements for more information on this IE.	PCS-user
Acu_clear_in_progress_description	Description for information element 0. See Progress description values for a list of valid values.	Progress indicator

Macro	Description	Q.931 IE
Acu_clear_in_progress_ind_nb	Number of progress indication information elements.	Progress indicator
Acu_clear_in_progress_location	Location of information element 0. See Location values for a list of valid values.	Progress indicator
Acu_clear_in_ret_code	Return code. See Clear code values for a list of valid values.	Does not map to an IE.
Acu_clear_in_signal_val	Signal value. See Signal values for a list of valid values.	Not used.
Acu_clear_in_ss_cnip_name_active	Indicates calling name identification presentation (CNIP) supplementary service should be invoked.	CNIP
Acu_clear_in_ss_cnip_name_pres	Calling name identification presentation (CNIP)mode. Allowed values: N_PRES_ALLOWED: Presentation allowed N_PRES_RESTRICTED: Presentation restricted N_PRES_NOT_AVAILABLE: Presentation not available	CNIP
Acu_clear_in_total_cost	Alias for charging field.	Not used.
Acu_clear_in_x_display_nb	Number of present occurrences.	Display
Acu_clear_in_x_display_size[***i***]	Size of occurrence ***i*** (optional).	Display
Acu_clear_in_x_display_total_size	Total size of the stored strings.	Display
Acu_clear_in_x_display_type[***i***]	Type of display ***i***.	Display
Acu_clear_in_x_p_display[***i***]	Pointer to occurrence ***i***.	Display

ACU_CLEAR_RQ

This topic describes:

- ACU_CLEAR_RQ protocol variants
- ACU_CLEAR_RQ macro descriptions and Q.931 IE

Note: Address parameters must be set according to the filling order. If one of them is unused, its size field must be set to 0.

Purpose

Requests releasing of a call.

Conn_id

An allocated call.

ACU_CLEAR_RQ protocol variants

The following table lists the variants under which each macro is supported for this primitive. In this table, NT indicates the NT side only, and x indicates both NT and TE sides.

Macro	**4ESS**	**E10**	**NI2**	**DMS**	**ETSI**	**VN6**	**HKG**	**AUSTEL1**	**NTT**	**KOREA**	**TAIWAN**	**QSIG**	**DPNSS**	**T1607**
Acu_clear_rq_a_display, Acu_clear_rq_display_size Filling order: 3														
Acu_clear_rq_a_display_list										x	x			
Acu_clear_rq_a_ext_parms														
Acu_clear_rq_a_facility, Acu_clear_rq_facility_size Filling order: 2					x	x				x	x	x		
Acu_clear_rq_a_pcs_user, Acu_clear_rq_pcs_user_size Filling order: 4						x								
Acu_clear_rq_a_ss_cnip_name, Acu_clear_rq_ss_cnip_name_size Filling order: 5												x		
Acu_clear_rq_a_tsp_ie_list, Acu_clear_rq_tsp_ie_list_size Filling order: 6	x	x	x	x	x	x	x	x	x	x	x	x		x
Acu_clear_rq_a_uui, Acu_clear_rq_uui_size Filling order: 1	x	x			x	x				x	x			

Macro	**4ESS**	**E10**	**NI2**	**DMS**	**ETSI**	**VN6**	**HKG**	**AUSTEL1**	**NTT**	**KOREA**	**TAIWAN**	**QSIG**	**DPNSS**	**T1607**
Acu_clear_rq_cause	x	x	x	x	x	x	x	x	x	x	x	x		x
Acu_clear_rq_charging_available						NT								
Acu_clear_rq_charging_multi														
Acu_clear_rq_charging_period														
Acu_clear_rq_charging_type														
Acu_clear_rq_charging_value						NT								
Acu_clear_rq_ext_parms_lgth														
Acu_clear_rq_ext_parms_nb														
Acu_clear_rq_pcs_user_protocol						x								
Acu_clear_rq_priority														
Acu_clear_rq_signal_val														
Acu_clear_rq_ss_cnip_name_active												x		
Acu_clear_rq_ss_cnip_name_pres												x		
Acu_clear_rq_total_cost														
Acu_clear_rq_uui_protocol										x	x			
Acu_clear_rq_x_display_nb														
Acu_clear_rq_x_display_size[***i***]														
Acu_clear_rq_x_display_total_size														
Acu_clear_rq_x_display_type[***i***]														
Acu_clear_rq_x_p_display[***i***]														

ACU_CLEAR_RQ macro descriptions and Q.931 IE

Macro	Description	Q.931 IE
Acu_clear_rq_a_display, Acu_clear_rq_display_size Filling order: 3	Pointer to (and size of) buffer containing ISDN display information (optional).	Display
Acu_clear_rq_a_display_list	Pointer to display structure.	Display
Acu_clear_rq_a_ext_parms	Pointer to buffer containing extended parameters.	Not used.
Acu_clear_rq_a_facility, Acu_clear_rq_facility_size Filling order: 2	Pointer to (and size of) buffer containing ISDN facility information (optional).	Facility
Acu_clear_rq_a_pcs_user, Acu_clear_rq_pcs_user_size Filling order: 4	Pointer to (and size of) buffer containing ISDN PCS-to-user/user-to-PCS information. See PCS-user information elements for more information on this IE.	PCS-user
Acu_clear_rq_a_ss_cnip_name, Acu_clear_rq_ss_cnip_name_size Filling order: 5	Pointer to (and size of) buffer containing calling name identification presentation (CNIP) name.	CNIP
Acu_clear_rq_a_tsp_ie_list, Acu_clear_rq_tsp_ie_list_size Filling order: 6	Pointer to (and size of) transparent IE buffer.	Does not map to an IE.
Acu_clear_rq_a_uui, Acu_clear_rq_uui_size Filling order: 1	Pointer to (and size of) buffer containing ISDN user-to-user information (optional).	User-user
Acu_clear_rq_cause	Call clearing cause value to be sent. If 0, Normal Clearing (16) is sent.	Cause
Acu_clear_rq_charging_available	Charging information available indicator.	
Acu_clear_rq_charging_multi	Charging multiplier.	Not used.
Acu_clear_rq_charging_period	Charging period.	Not used.
Acu_clear_rq_charging_type	Charging type. See Charging type values for a list of valid values.	Not used.
Acu_clear_rq_charging_value	Charging value (number of units).	

Macro	Description	Q.931 IE
Acu_clear_rq_ext_parms_lgth	Total length of buffer containing extended parameters.	Not used.
Acu_clear_rq_ext_parms_nb	Number of parameters in buffer containing extended parameters.	Not used.
Acu_clear_rq_pcs_user_protocol	Protocol discriminator for the pcs_user information element. See PCS-user information elements for more information on this IE.	PCS-user
Acu_clear_rq_priority	Call clear priority. Allowed values include: ACU_PHIGH: normal ACU_PLOW: urgent	Low layer compatibility
Acu_clear_rq_signal_val	Signal value. See Signal values for a list of valid values.	Not used.
Acu_clear_rq_ss_cnip_name_active	Indicates calling name identification presentation (CNIP) supplementary service should be invoked.	CNIP
Acu_clear_rq_ss_cnip_name_pres	Calling name identification presentation (CNIP) mode. Allowed values include: N_PRES_ALLOWED: Presentation allowed N_PRES_RESTRICTED: Presentation restricted N_PRES_NOT_AVAILABLE: Presentation not available	CNIP
Acu_clear_rq_total_cost	Pointer to display structure.	Not used.
Acu_clear_rq_uui_protocol	UUI protocol discriminator value. See UUI protocol discriminator values for a list of valid values. Default: UUI_IA5 (UUI_USER_SPF for HKG variant)	User-user
Acu_clear_rq_x_display_nb	Number of present occurrences.	Display
Acu_clear_rq_x_display_size[*i*]	Size of occurrence ***i*** (optional).	Display
Acu_clear_rq_x_display_total_size	Total size of the stored strings.	Display

Macro	Description	Q.931 IE
Acu_clear_rq_x_display_type[***i***]	Type of display ***i***.	Display
Acu_clear_rq_x_p_display[***i***]	Pointer to occurrence ***i***.	Display

ACU_CLEAR_RS

This topic describes:

- ACU_CLEAR_RS protocol variants
- ACU_CLEAR_RS macro descriptions and Q.931 IE

Note: Address parameters must be set according to the filling order. If one of them is unused, its size field must be set to 0.

Purpose

Indicates a response to a release indication.

Conn_id

An allocated call.

ACU_CLEAR_RS protocol variants

The following table lists the variants under which each macro is supported for this primitive. In this table, NT indicates the NT side only, and x indicates both NT and TE sides.

Macro	4ESS	E10	NI2	DMS	ETSI	VN6	HKG	AUSTEL1	NTT	KOREA	TAIWAN	QSIG	DPNSS	T1607
Acu_clear_rs_charging_value														
Acu_clear_rs_a_display, Acu_clear_rs_display_size Filling order: 3														
Acu_clear_rs_a_display_list										x	x			
Acu_clear_rs_a_ext_parms														
Acu_clear_rs_a_facility, Acu_clear_rs_facility_size Filling order: 2					x	x				x	x	x		

Macro	4ESS	E10	NI2	DMS	ETSI	VN6	HKG	AUSTEL1	NTT	KOREA	TAIWAN	QSIG	DPNSS	T1607
Acu_clear_rs_a_pcs_user, Acu_clear_rs_pcs_user_size Filling order: 4						x								
Acu_clear_rs_a_tsp_ie_list, Acu_clear_rs_tsp_ie_list_size Filling order: 5	x	x	x	x	x	x	x	x	x	x	x	x		x
Acu_clear_rs_a_uui, Acu_clear_rs_uui_size Filling order: 1			x		x	x			x	x	x			
Acu_clear_rs_charging_available						NT								
Acu_clear_rs_charging_multi														
Acu_clear_rs_charging_period														
Acu_clear_rs_charging_type														
Acu_clear_rs_charging_value						NT								
Acu_clear_rs_ext_parms_lgth														
Acu_clear_rs_ext_parms_nb														
Acu_clear_rs_pcs_user_protocol						x								
Acu_clear_rs_priority										x	x			
Acu_clear_rs_uui_protocol														
Acu_clear_rs_x_display_nb														
Acu_clear_rs_x_display_size[***i***]														
Acu_clear_rs_x_display_total_size														
Acu_clear_rs_x_display_type[***i***]														

Macro	4ESS	E10	NI2	DMS	ETSI	VN6	HKG	AUSTEL1	NTT	KOREA	TAIWAN	QSIG	DPNSS	T1607
Acu_clear_rs_x_p_display[*i*]														

ACU_CLEAR_RS macro descriptions and Q.931 IE

Macro	Description	Q.931 IE
Acu_clear_rs_charging_value	Charging value (number of units).	Not used.
Acu_clear_rs_a_display, Acu_clear_rs_display_size Filling order: 3	Pointer to (and size of) buffer containing ISDN display information (optional).	Display
Acu_clear_rs_a_display_list	Pointer to display structure.	Display
Acu_clear_rs_a_ext_parms	Pointer to buffer containing extended parameters.	Not used.
Acu_clear_rs_a_facility, Acu_clear_rs_facility_size Filling order: 2	Pointer to (and size of) buffer containing ISDN facility information (optional).	Facility
Acu_clear_rs_a_pcs_user, Acu_clear_rs_pcs_user_size Filling order: 4	Pointer to (and size of) buffer containing ISDN PCS-to-user/user-to-PCS information. See PCS-user information elements for more information on this IE.	PCS-user
Acu_clear_rs_a_tsp_ie_list, Acu_clear_rs_tsp_ie_list_size Filling order: 5	Pointer to (and size of) transparent IE buffer.	Does not map to an IE.
Acu_clear_rs_a_uui, Acu_clear_rs_uui_size Filling order: 1	Pointer to (and size of) buffer containing ISDN user-to-user information (optional).	User-user
Acu_clear_rs_charging_available	Charging information available indicator.	Does not map to an IE.
Acu_clear_rs_charging_multi	Charging multiplier.	Not used.

Macro	Description	Q.931 IE
Acu_clear_rs_charging_period	Charging period.	Not used.
Acu_clear_rs_charging_type	Charging type. See Charging type values for a list of valid values.	Not used.
Acu_clear_rs_charging_value	Charging value (number of units).	
Acu_clear_rs_ext_parms_lgth	Total length of buffer containing extended parameters.	Not used.
Acu_clear_rs_ext_parms_nb	Number of parameters in buffer containing extended parameters.	Not used.
Acu_clear_rs_pcs_user_protocol	Protocol discriminator for the pcs_user information element. See PCS-user information elements for more information on this IE.	PCS-user
Acu_clear_rs_priority	Primitive priority. Values: • ACU_PHIGH: normal • ACU_PLOW: urgent	Bearer capability
Acu_clear_rs_uui_protocol	UUI protocol discriminator value. See UUI protocol discriminator values for a list of valid values. Default: UUI_IA5 (UUI_USER_SPF for HKG variant)	User-user
Acu_clear_rs_x_display_nb	Number of present occurrences.	Display
Acu_clear_rs_x_display_size[***i***]	Size of occurrence ***i*** (optional).	Display
Acu_clear_rs_x_display_total_size	Total size of the stored strings.	Display
Acu_clear_rs_x_display_type[***i***]	Type of display ***i***.	Display
Acu_clear_rs_x_p_display[***i***]	Pointer to occurrence ***i***.	Display

ACU_CONN_CO

This topic describes:

- ACU_CONN_CO protocol variants
- ACU_CONN_CO macro descriptions and Q.931 IE

Purpose

Indicates a call connection confirmation.

Conn_id

An allocated call.

ACU_CONN_CO protocol variants

The following table lists the variants under which each macro is supported for this primitive. In this table, x indicates both NT and TE sides.

Macro	4ESS	E10	NI2	DMS	ETSI	VN6	HKG	AUSTEL1	NTT	KOREA	TAIWAN	QSIG	DPNSS	T1607
Acu_conn_co_a_connected_nb, Acu_conn_co_connected_nb_size									x			x		
Acu_conn_co_a_connected_sub, Acu_conn_co_connected_sub_size									x			x		
Acu_conn_co_a_date_time					x	x				x	x			
Acu_conn_co_a_display, Acu_conn_co_display_size		x	x		x	x	x	x	x	x	x			x
Acu_conn_co_a_display_list		x	x		x	x	x	x	x	x	x			x
Acu_conn_co_a_ext_parms														
Acu_conn_co_a_facility, Acu_conn_co_facility_size					x	x				x	x	x		
Acu_conn_co_a_pcs_user, Acu_conn_co_pcs_user_size						x								
Acu_conn_co_a_q931, Acu_conn_co_q931_size	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Acu_conn_co_a_redir_nb, Acu_conn_co_redir_nb_size	x	x												
Acu_conn_co_a_ss_cnip_name, Acu_conn_co_ss_cnip_name_size												x		
Acu_conn_co_a_uui, Acu_conn_co_uui_size	x	x	x		x	x	x		x	x	x			

Macro	4ESS	E10	NI2	DMS	ETSI	VN6	HKG	AUSTEL1	NTT	KOREA	TAIWAN	QSIG	DPNSS	T1607
Acu_conn_co_call_ref_length If the network side initiates the call, it does not receive the call reference value with this primitive.			x											
Acu_conn_co_call_ref_value If the network side initiates the call, it does not receive the call reference value with this primitive.			x											
Acu_conn_co_charging														
Acu_conn_co_charging_available														
Acu_conn_co_charging_multi														
Acu_conn_co_charging_period														
Acu_conn_co_charging_type														
Acu_conn_co_charging_value														
Acu_conn_co_connected_nb_pres									x			x		x
Acu_conn_co_connected_nb_screen									x			x		x
Acu_conn_co_connected_nb_type									x			x		x
Acu_conn_co_connected_sub_odd_even									x			x		
Acu_conn_co_connected_sub_type									x			x		
Acu_conn_co_data_chani	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Acu_conn_co_data_chani_excl													x	
Acu_conn_co_data_chani_nai													x	
Acu_conn_co_data_chani_nb														

Macro	**4ESS**	**E10**	**NI2**	**DMS**	**ETSI**	**VN6**	**HKG**	**AUSTEL1**	**NTT**	**KOREA**	**TAIWAN**	**QSIG**	**DPNSS**	**T1607**
Acu_conn_co_data_chani_tab[***i***]														
Acu_conn_co_data_chani_tab_nai[***i***]														
Acu_conn_co_date_available										x	x			
Acu_conn_co_day					x	x				x	x			
Acu_conn_co_ext_parms_lgth														
Acu_conn_co_ext_parms_nb														
Acu_conn_co_hour					x	x				x	x			
Acu_conn_co_minute					x	x				x	x			
Acu_conn_co_month					x	x				x	x			
Acu_conn_co_pcs_user_protocol						x								
Acu_conn_co_redir_nb_pres	x	x												
Acu_conn_co_redir_nb_reason	x	x												
Acu_conn_co_redir_nb_screen	x	x												
Acu_conn_co_redir_nb_type	x	x												
Acu_conn_co_second														
Acu_conn_co_service														
Acu_conn_co_signal_val														
Acu_conn_co_ss_cnip_name_active												x		
Acu_conn_co_ss_cnip_name_pres												x		
Acu_conn_co_uui_protocol														

Macro	**4ESS**	**E10**	**NI2**	**DMS**	**ETSI**	**VN6**	**HKG**	**AUSTEL1**	**NTT**	**KOREA**	**TAIWAN**	**QSIG**	**DPNSS**	**T1607**
Acu_conn_co_x_display_nb		x	x		x	x	x	x	x	x	x			x
Acu_conn_co_x_display_size[***i***]		x	x		x	x	x	x	x	x	x			x
Acu_conn_co_x_display_total_size		x	x		x	x	x	x	x	x	x			x
Acu_conn_co_x_display_type[***i***]		x	x		x	x	x	x	x	x	x			x
Acu_conn_co_x_p_display[***i***]		x	x		x	x	x	x	x	x	x			x
Acu_conn_co_year					x	x				x	x			

ACU_CONN_CO macro descriptions and Q.931 IE

Macro	**Description**	**Q.931 IE**
Acu_conn_co_a_connected_nb, Acu_conn_co_connected_nb_size	Pointer to (and size of) buffer containing connected number.	Connected party address
Acu_conn_co_a_connected_sub, Acu_conn_co_connected_sub_size	Pointer to (and size of) buffer containing connected subaddress.	Connected party subaddress
Acu_conn_co_a_date_time	Pointer to date_time.	Date/time
Acu_conn_co_a_display, Acu_conn_co_display_size	Pointer to (and size of) buffer containing ISDN display information (optional).	Display
Acu_conn_co_a_display_list	Pointer to display structure.	Display
Acu_conn_co_a_ext_parms	Pointer to buffer containing extended parameters.	Not used.
Acu_conn_co_a_facility, Acu_conn_co_facility_size	Pointer to (and size of) buffer containing ISDN facility information (optional).	Facility

Macro	Description	Q.931 IE
Acu_conn_co_a_pcs_user, Acu_conn_co_pcs_user_size	Pointer to (and size of) buffer containing ISDN PCS-to-user and user-to-PCS information. See PCS-user information elements for more information on this IE.	PCS-user
Acu_conn_co_a_q931, Acu_conn_co_q931_size	Pointer to (and size of) buffer containing raw data of incoming Q.931 message.	Does not map to an IE.
Acu_conn_co_a_redir_nb, Acu_conn_co_redir_nb_size	Pointer to (and size of) buffer containing redirecting number.	Redirecting number
Acu_conn_co_a_ss_cnip_name, Acu_conn_co_ss_cnip_name_size	Pointer to (and size of) buffer containing calling name identification presentation (CNIP) name.	CNIP
Acu_conn_co_a_uui, Acu_conn_co_uui_size	Pointer to (and size of) buffer containing ISDN user-to-user information (optional).	User-user
Acu_conn_co_call_ref_length	Length of call reference value.	Call reference
Acu_conn_co_call_ref_value	Call reference value and call reference flag.	Call reference
Acu_conn_co_charging	Charging value (number of units).	Not used.
Acu_conn_co_charging_available	Charging information available indicator.	Not used.
Acu_conn_co_charging_multi	Charging multiplier.	Not used.
Acu_conn_co_charging_period	Charging period.	Not used.
Acu_conn_co_charging_type	Charging type. See Charging type values for a list of valid values.	Not used.
Acu_conn_co_charging_value	Charging value (number of units).	Not used.

Macro	Description	Q.931 IE
Acu_conn_co_connected_nb_pres	Connected number presentation. Allowed values include: N_PRES_ALLOWED: Presentation allowed N_PRES_RESTRICTED: Presentation restricted N_PRES_NOT_AVAILABLE: Presentation not available	Connected party address
Acu_conn_co_connected_nb_screen	Connected number screening indicator. See Screening indicator values for a list of valid values.	Connected party address
Acu_conn_co_connected_nb_type	Connected number type. See Number type values for a list of valid values.	Connected party address
Acu_conn_co_connected_sub_odd_even	Connected subaddress odd/even. Valid values are: SUBADDRESS_ODD: Odd number of address signals SUBADDRESS_EVEN: Even number of address signals	Connected party subaddress
Acu_conn_co_connected_sub_type	Connected subaddress type. See Number type values for a list of valid values.	Connected party subaddress
Acu_conn_co_data_chani	Data channel to use (B1, B2, ..., D).	Channel identification
Acu_conn_co_data_chani_excl	Channel IDs are preferred (OFF) or exclusive (ON).	Channel identification
Acu_conn_co_data_chani_nai	NAI.	Channel identification
Acu_conn_co_data_chani_nb	Number of channel IDs in the chani_list field. If no channel IDs, use 0.	Channel identification
Acu_conn_co_data_chani_tab[***i***]	Channel ID ***i***.	Channel identification
Acu_conn_co_data_chani_tab_nai[***i***]	NAI ***i***.	Channel identification

Macro	**Description**	**Q.931 IE**
Acu_conn_co_date_available	ON if the information is available, else OFF.	Date/time
Acu_conn_co_day	Day.	Date/time
Acu_conn_co_ext_parms_lgth	Total length of buffer containing extended parameters.	Not used.
Acu_conn_co_ext_parms_nb	Number of parameters in buffer containing extended parameters.	Not used.
Acu_conn_co_hour	Hour.	Date/time
Acu_conn_co_minute	Minute.	Date/time
Acu_conn_co_month	Month.	Date/time
Acu_conn_co_pcs_user_protocol	Protocol discriminator for the pcs_user information element. See PCS-user information elements for more information on this IE.	PCS-user
Acu_conn_co_redir_nb_pres	Redirecting number presentation. Allowed values include: N_PRES_ALLOWED: Presentation allowed N_PRES_RESTRICTED: Presentation restricted N_PRES_NOT_AVAILABLE: Presentation not available	Redirecting number
Acu_conn_co_redir_nb_reason	Reason for redirection. See Redirecting reason values for a list of valid values.	Redirecting number
Acu_conn_co_redir_nb_screen	Redirecting number screening indicator. See Screening indicator values for a list of valid values.	Redirecting number
Acu_conn_co_redir_nb_type	Redirecting number type. See Number type values for a list of valid values.	Redirecting number

Macro	Description	Q.931 IE
Acu_conn_co_second	Second.	Date/time
Acu_conn_co_service	Telephony service requested by the remote. See Service values for a list of valid values.	bc + llc + hlc
Acu_conn_co_signal_val	Signal value. See Signal values for a list of valid values.	Not used.
Acu_conn_co_ss_cnip_name_active	Indicates calling name identification presentation (CNIP) supplementary service should be invoked.	CNIP
Acu_conn_co_ss_cnip_name_pres	Calling name identification presentation mode. Allowed values include: N_PRES_ALLOWED: Presentation allowed N_PRES_RESTRICTED: Presentation restricted N_PRES_NOT_AVAILABLE: Presentation not available	CNIP
Acu_conn_co_uui_protocol	UUI protocol discriminator value. See UUI protocol discriminator values for a list of valid values.	User-user
Acu_conn_co_x_display_nb	Number of present occurrences.	Display
Acu_conn_co_x_display_size[***i***]	Size of occurrence ***i*** (optional).	Display
Acu_conn_co_x_display_total_size	Total size of stored strings.	Display
Acu_conn_co_x_display_type[***i***]	Type of display ***i***.	Display
Acu_conn_co_x_p_display[***i***]	Pointer to occurrence ***i***.	Display
Acu_conn_co_year	Year.	Date/time

ACU_CONN_IN

This topic describes:

- ACU_CONN_IN protocol variants
- ACU_CONN_IN macro descriptions and Q.931 IE

Purpose

Signals an incoming call.

Conn_id

A currently unused ID.

ACU_CONN_IN protocol variants

The following table lists the variants under which each macro is supported for this primitive. In this table, x indicates both NT and TE sides.

Macro	4ESS	E10	NI2	DMS	ETSI	VN6	HKG	AUSTEL1	NTT	KOREA	TAIWAN	QSIG	DPNSS	T1607
Acu_conn_in_a_called_nb, Acu_conn_in_called_nb_size	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Acu_conn_in_a_called_nb_sub, Acu_conn_in_called_nb_sub_size									x					x
Acu_conn_in_a_calling_nb, Acu_conn_in_calling_nb_size	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Acu_conn_in_a_calling_nb2, Acu_conn_in_calling_nb2_size						x								
Acu_conn_in_a_calling_nb_sub, Acu_conn_in_calling_nb_sub_size									x					x
Acu_conn_in_a_calling_name, Acu_conn_in_calling_name_size				x										
Acu_conn_in_a_display, Acu_conn_in_display_size			x	x	x	x		x	x	x	x			x
Acu_conn_in_a_display_list			x	x	x	x		x	x	x	x			x
Acu_conn_in_a_ext_parms														
Acu_conn_in_a_facility, Acu_conn_in_facility_size					x	x				x	x	x		
Acu_conn_in_a_layer_1_info														
Acu_conn_in_a_orig_called_nb				x	x									

Macro	4ESS	E10	NI2	DMS	ETSI	VN6	HKG	AUSTEL1	NTT	KOREA	TAIWAN	QSIG	DPNSS	T1607
Acu_conn_in_orig_called_nb_cfnr				x	x									
Acu_conn_in_orig_called_nb_count				x	x									
Acu_conn_in_orig_called_nb_plan				x	x									
Acu_conn_in_orig_called_nb_pres				x	x									
Acu_conn_in_orig_called_nb_reason				x	x									
Acu_conn_in_orig_called_nb_screen				x	x									
Acu_conn_in_orig_called_nb_size				x	x									
Acu_conn_in_orig_called_nb_type				x	x									
Acu_conn_in_orig_line_info	x			x	x									
Acu_conn_in_a_pcs_user, Acu_conn_in_pcs_user_size						x								
Acu_conn_in_a_ph_num														
Acu_conn_in_a_q931, Acu_conn_in_q931_size	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Acu_conn_in_a_redir_nb, Acu_conn_in_redir_nb_size	x	x	x	x	x	x								
Acu_conn_in_a_ss_cnip_name, Acu_conn_in_ss_cnip_name_size												x		
Acu_conn_in_a_uui, Acu_conn_in_uui_size	x	x	x		x	x	x		x	x	x			
Acu_conn_in_call_ref_length			x											
Acu_conn_in_call_ref_value			x											
Acu_conn_in_called_nb_plan	x	x	x	x	x	x				x	x			x

Macro	**4ESS**	**E10**	**NI2**	**DMS**	**ETSI**	**VN6**	**HKG**	**AUSTEL1**	**NTT**	**KOREA**	**TAIWAN**	**QSIG**	**DPNSS**	**T1607**
Acu_conn_in_called_nb_sub_odd_even									x					
Acu_conn_in_called_nb_sub_type									x					
Acu_conn_in_called_nb_type	x	x	x	x	x	x		x		x	x			x
Acu_conn_in_calling_nb2_pres						x								
Acu_conn_in_calling_nb2_screen						x								
Acu_conn_in_calling_nb2_type														
Acu_conn_in_calling_nb_plan			x	x	x		x							x
Acu_conn_in_calling_nb_pres	x	x	x	x	x	x	x	x	x	x	x	x		x
Acu_conn_in_calling_nb_screen	x	x	x	x	x	x	x	x	x	x	x	x		x
Acu_conn_in_calling_nb_sub_odd_even									x					
Acu_conn_in_calling_nb_sub_type									x	x	x			
Acu_conn_in_calling_nb_type	x	x	x	x	x	x	x	x		x	x	x		x
Acu_conn_in_chani_nai														
Acu_conn_in_data_bits														
Acu_conn_in_data_chani	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Acu_conn_in_data_chani_excl														
Acu_conn_in_data_chani_nb														
Acu_conn_in_data_chani_tab														
Acu_conn_in_data_chani_tab_nai														
Acu_conn_in_dest_call_appear														

Macro	**4ESS**	**E10**	**NI2**	**DMS**	**ETSI**	**VN6**	**HKG**	**AUSTEL1**	**NTT**	**KOREA**	**TAIWAN**	**QSIG**	**DPNSS**	**T1607**
Acu_conn_in_ext_parms_lgth														
Acu_conn_in_ext_parms_nb														
Acu_conn_in_interworking														
Acu_conn_in_parity														
Acu_conn_in_pcs_user_protocol						x								
Acu_conn_in_ph_rate														
Acu_conn_in_progress_descr_x[***i***]														
Acu_conn_in_progress_description					x	x				x	x			x
Acu_conn_in_progress_ind_nb										x	x			x
Acu_conn_in_progress_loc_x[***i***]														
Acu_conn_in_progress_location					x	x				x	x			x
Acu_conn_in_redir_nb_plan	x	x	x	x	x									
Acu_conn_in_redir_nb_pres	x	x	x	x	x	x								
Acu_conn_in_redir_nb_reason	x	x	x	x	x	x								
Acu_conn_in_redir_nb_screen	x	x	x	x	x	x								
Acu_conn_in_redir_nb_type	x	x	x	x	x	x								
Acu_conn_in_semi_permanent_circuit														
Acu_conn_in_sending_complete					x	x	x			x	x	x	x	x
Acu_conn_in_service													x	
Acu_conn_in_service_list_id														

Macro	**4ESS**	**E10**	**NI2**	**DMS**	**ETSI**	**VN6**	**HKG**	**AUSTEL1**	**NTT**	**KOREA**	**TAIWAN**	**QSIG**	**DPNSS**	**T1607**
Acu_conn_in_ss_cnip_name_active												x		
Acu_conn_in_ss_cnip_name_pres												x		
Acu_conn_in_stop_bits														
Acu_conn_in_syn_asyn														
Acu_conn_in_user_rate														
Acu_conn_in_uui_protocol														
Acu_conn_in_x_display_nb			x	x	x	x		x	x	x	x			x
Acu_conn_in_x_display_size[***i***]			x	x	x	x				x	x			x
Acu_conn_in_x_display_total_size			x	x	x	x				x	x			x
Acu_conn_in_x_display_type[***i***]			x	x	x	x				x	x			x
Acu_conn_in_x_p_display[***i***]			x	x	x	x				x	x			x

ACU_CONN_IN macro descriptions and Q.931 IE

Macro	**Description**	**Q.931 IE**
Acu_conn_in_a_called_nb, Acu_conn_in_called_nb_size	Pointer to the called number.	Called party number + called party subaddress
Acu_conn_in_a_called_nb_sub, Acu_conn_in_called_nb_sub_size	Pointer to (and size of) buffer containing called subaddress.	Calling party subaddress
Acu_conn_in_a_calling_nb, Acu_conn_in_calling_nb_size	Pointer to the calling number.	Calling party number + calling party subaddress

Macro	Description	Q.931 IE
Acu_conn_in_a_calling_nb2, Acu_conn_in_calling_nb2_size	Pointer to (and size of) buffer containing second calling number.	Calling party number + calling party subaddress
Acu_conn_in_a_calling_nb_sub, Acu_conn_in_calling_nb_sub_size	Pointer to (and size of) buffer containing calling subaddress.	Calling party subaddress
Acu_conn_in_a_calling_name, Acu_conn_in_calling_name_size	Pointer to (and size of) buffer containing calling name.	Calling name
Acu_conn_in_a_display, Acu_conn_in_display_size	Pointer to (and size of) buffer containing ISDN display information (optional).	Display
Acu_conn_in_a_display_list	Pointer to display structure.	Display
Acu_conn_in_a_ext_parms	Pointer to buffer containing extended parameters.	Network specific facility
Acu_conn_in_a_facility, Acu_conn_in_facility_size	Pointer to (and size of) buffer containing ISDN facility information (optional).	Facility
Acu_conn_in_a_layer_1_info	Pointer to structure containing layer 1 information.	Low layer information
Acu_conn_in_a_orig_called_nb	Pointer to the original called number.	Original called number
Acu_conn_in_orig_called_nb_cfnr	Original called number call forward no reply indicator (Boolean).	Original called number
Acu_conn_in_orig_called_nb_count	Original called number redirection count.	Original called number
Acu_conn_in_orig_called_nb_plan	Original called number plan. See Plan values for a list of valid values.	Original called number

Macro	Description	Q.931 IE
Acu_conn_in_orig_called_nb_pres	Original called number presentation indicator. Valid values include: N_PRES_ALLOWED N_PRES_RESTRICTED N_PRES_NOT_AVAILABLE	Original called number
Acu_conn_in_orig_called_nb_reason	Original called number redirection reason. See Redirecting reason values for a list of valid values.	Original called number
Acu_conn_in_orig_called_nb_screen	Original called number screening indicator. See Screening indicator values for a list of valid values.	Original called number
Acu_conn_in_orig_called_nb_size	Size of the original called number.	Original called number
Acu_conn_in_orig_called_nb_type	Original called number type. See Number type values for a list of valid values.	Original called number
Acu_conn_in_orig_line_info	Originating line information (0xFF means no information available).	Originating line information
Acu_conn_in_a_pcs_user, Acu_conn_in_pcs_user_size	Pointer to (and size of) buffer containing ISDN PCS-to-user and user-to-PCS information. See PCS-user information elements for more information on this IE.	PCS-user
Acu_conn_in_a_ph_num	Alias for Acu_conn_in_a_calling_nb.	Low layer information
Acu_conn_in_a_q931, Acu_conn_in_q931_size	Pointer to (and size of) buffer containing raw data of incoming Q.931 message.	Does not map to an IE.
Acu_conn_in_a_redir_nb, Acu_conn_in_redir_nb_size	Pointer to (and size of) buffer containing redirecting number.	Redirecting number

Macro	Description	Q.931 IE
Acu_conn_in_a_ss_cnip_name, Acu_conn_in_ss_cnip_name_size	Pointer to (and size of) buffer containing calling name identification presentation (CNIP) name.	CNIP
Acu_conn_in_a_uui, Acu_conn_in_uui_size	Pointer to (and size of) buffer containing ISDN user-to-user information (optional).	User-user
Acu_conn_in_call_ref_length	Length of call reference value.	Call reference
Acu_conn_in_call_ref_value	Call reference value and call reference flag.	Call reference
Acu_conn_in_called_nb_plan	Called number plan. See Plan values for a list of valid values.	Called party number
Acu_conn_in_called_nb_sub_odd_even	Called subaddress odd/even. Valid values include: SUBADDRESS_ODD: Odd number of address signals SUBADDRESS_EVEN: Even number of address signals	Called party subaddress
Acu_conn_in_called_nb_sub_type	Called subaddress number type. Valid values include: SUBADDRESS_TYPE_NSAP: NSAP SUBADDRESS_TYPE_USER: User specified	Called party subaddress
Acu_conn_in_called_nb_type	Called number type. See Number type values for a list of valid values.	Called party number
Acu_conn_in_calling_nb2_pres	Second calling number presentation. Allowed values include: N_PRES_ALLOWED: Presentation allowed N_PRES_RESTRICTED: Presentation restricted N_PRES_NOT_AVAILABLE: Presentation not available	Calling party number

Macro	Description	Q.931 IE
Acu_conn_in_calling_nb2_screen	Second calling number screening indicator. See Screening indicator values for a list of valid values.	Calling party number
Acu_conn_in_calling_nb2_type	Second calling number type. See Number type values for a list of valid values.	Calling party number
Acu_conn_in_calling_nb_plan	Calling number plan. See Plan values for a list of valid values.	Calling party number
Acu_conn_in_calling_nb_pres	Calling number presentation. Allowed values include: N_PRES_ALLOWED: Presentation allowed N_PRES_RESTRICTED: Presentation restricted N_PRES_NOT_AVAILABLE: Presentation not available	Calling party number
Acu_conn_in_calling_nb_screen	Calling number screening indicator. See Screening indicator values for a list of valid values.	Calling party number
Acu_conn_in_calling_nb_sub_odd_even	Called subaddress odd/even. Allowed values include: SUBADDRESS_ODD: Odd number of address signals SUBADDRESS_EVEN: Even number of address signals	Calling party subaddress
Acu_conn_in_calling_nb_sub_type	Calling subaddress number type. Valid values include: SUBADDRESS_TYPE_NSAP: NSAP SUBADDRESS_TYPE_USER: User specified	Calling party subaddress
Acu_conn_in_calling_nb_type	Calling number type. See Number type values for a list of valid values.	Calling party number

Macro	Description	Q.931 IE
Acu_conn_in_chani_nai	NAI.	Channel identification
Acu_conn_in_data_bits	Number of data bits for V.110 and V.120 services. Available values include: ACU_DATA_BIT_5: 5 data bits ACU_DATA_BIT_7: 7 data bits ACU_DATA_BIT_8: 8 data bits	Not used.
Acu_conn_in_data_chani	Data channel to use (B1, B2, ..., D).	Channel identification
Acu_conn_in_data_chani_excl	Channel IDs are preferred (OFF) or exclusive (ON).	Channel identification
Acu_conn_in_data_chani_nb	Number of channel IDs in the chani_list field. If no channel IDs, use 0.	Channel identification
Acu_conn_in_data_chani_tab	Channel ID.	Channel identification
Acu_conn_in_data_chani_tab_nai	NAI.	Channel identification
Acu_conn_in_dest_call_appear	Destination call appearance.	Not used.
Acu_conn_in_ext_parms_lgth	Total length of buffer containing extended parameters.	Not used.
Acu_conn_in_ext_parms_nb	Number of parameters in buffer containing extended parameters.	Not used.
Acu_conn_in_interworking	Interworking indication: ON: Interworking occurred OFF: No interworking	Not used.
Acu_conn_in_parity	Parity for V.110 and V.120 services. Available values include: ACU_ODD: odd parity ACU_EVEN: even parity ACU_NO_PARITY: no parity	Low layer information

Macro	Description	Q.931 IE
Acu_conn_in_pcs_user_protocol	Protocol discriminator for the pcs_user information element. See PCS-user information elements for more information on this IE.	PCS-user
Acu_conn_in_ph_rate	Physical rate (for all services).	Low layer information
Acu_conn_in_progress_descr_x[***i***]	Description for information element ***i***. See Progress description values for a list of valid values.	Progress indicator
Acu_conn_in_progress_description	Description for information element 0. See Progress description values for a list of valid values.	Progress indicator
Acu_conn_in_progress_ind_nb	Number of progress indication information elements.	Progress indicator
Acu_conn_in_progress_loc_x[***i***]	Location of information element ***i***. See Location values for a list of valid values.	Progress indicator
Acu_conn_in_progress_location	Location of information element 0. See Location values for a list of valid values.	Progress indicator
Acu_conn_in_redir_nb_plan	Redirecting number plan. See Plan values for a list of valid values.	Redirecting number
Acu_conn_in_redir_nb_pres	Redirecting number presentation. Allowed values include: N_PRES_ALLOWED: Presentation allowed N_PRES_RESTRICTED: Presentation restricted N_PRES_NOT_AVAILABLE: Presentation not available	Redirecting number
Acu_conn_in_redir_nb_reason	Reason for redirection. See Redirecting reason values for a list of valid values.	Redirecting number

Macro	**Description**	**Q.931 IE**
Acu_conn_in_redir_nb_screen	Redirecting number screening indicator. See Screening indicator values for a list of valid values.	Redirecting number
Acu_conn_in_redir_nb_type	Redirecting number type. See Number type values for a list of valid values.	Redirecting number
Acu_conn_in_semi_permanent_circuit	Semi-permanent circuit.	
Acu_conn_in_sending_complete	Indicates if the sending-complete information element has been received (ON/OFF).	Sending complete
Acu_conn_in_service	Service requested by the remote. See Service values for a list of valid values.)	bc + llc + hlc
Acu_conn_in_service_list_id	Service list ID associated with the selected services list.	bc + llc + hlc
Acu_conn_in_ss_cnip_name_active	Indicates calling name identification presentation (CNIP) supplementary service should be invoked.	CNIP
Acu_conn_in_ss_cnip_name_pres	Calling name identification presentation mode. Allowed values include: N_PRES_ALLOWED: Presentation allowed N_PRES_RESTRICTED: Presentation restricted N_PRES_NOT_AVAILABLE: Presentation not available	CNIP
Acu_conn_in_stop_bits	Number of stop bits for V.110 and V.120 services. Available values include: ACU_STOP_BIT_1: 1 stop bit ACU_STOP_BIT_1_5: 1.5 stop bits ACU_STOP_BIT_2: 2 stop bits	Low layer information

Macro	Description	Q.931 IE
Acu_conn_in_syn_asyn	Synchronous/asynchronous for V.110 and V.120 services. Available values include: ACU_SYN: Synchronous mode ACU_ASYN: Asynchronous mode	Low layer information
Acu_conn_in_user_rate	User rate for V.110 and V.120 services. See User rate values for a list of valid values.	Low layer information
Acu_conn_in_uui_protocol	UUI protocol discriminator value. See UUI protocol discriminator values for a list of valid values.	User-user
Acu_conn_in_x_display_nb	Number of present occurrences.	Display
Acu_conn_in_x_display_size[*i*]	Size of occurrence ***i*** (optional).	Display
Acu_conn_in_x_display_total_size	Total size of the stored strings.	Display
Acu_conn_in_x_display_type[*i*]	Type of display ***i***.	Display
Acu_conn_in_x_p_display[*i*]	Pointer to occurrence ***i***.	Display

ACU_CONN_RQ

This topic describes:

- ACU_CONN_RQ protocol variants
- ACU_CONN_RQ macro descriptions and Q.931 IE

Note: Address parameters must be set according to the filling order. If one of them is unused, its size field must be set to 0.

Purpose

Requests establishment of an outgoing call.

Conn_id

A currently unused ID.

ACU_CONN_RQ protocol variants

The following table lists the variants under which each macro is supported for this primitive. In this table, x indicates both NT and TE sides.

Macro	4ESS	E10	NI2	DMS	ETSI	VN6	HKG	AUSTEL1	NTT	KOREA	TAIWAN	QSIG	DPNSS	T1607
Acu_conn_rq_a_called_nb, Acu_conn_rq_called_nb_size Filling order: 1	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Acu_conn_rq_a_called_nb_sub, Acu_conn_rq_a_called_nb_sub_size Filling order: 9									x	x	x	x		x
Acu_conn_rq_a_calling_nb, Acu_conn_rq_calling_nb_size Filling order: 2	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Acu_conn_rq_a_calling_nb2, Acu_conn_rq_calling_nb2_size Filling order: 3						x								
Acu_conn_rq_a_calling_nb_sub, Acu_conn_rq_a_calling_nb_sub_size Filling order: 10									x	x	x	x		x
Acu_conn_rq_a_calling_name, Acu_conn_rq_calling_name_size				x										
Acu_conn_rq_a_display, Acu_conn_rq_display_size Filling order: 7										x	x			
Acu_conn_rq_a_display_list														
Acu_conn_rq_a_ext_parms	x													x
Acu_conn_rq_a_facility, Acu_conn_rq_facility_size Filling order: 6					x	x				x	x	x		
Acu_conn_rq_a_layer_1_info														

Macro	**4ESS**	**E10**	**NI2**	**DMS**	**ETSI**	**VN6**	**HKG**	**AUSTEL1**	**NTT**	**KOREA**	**TAIWAN**	**QSIG**	**DPNSS**	**T1607**
Acu_conn_rq_a_pcs_user, Acu_conn_rq_pcs_user_size Filling order: 8						x								
Acu_conn_rq_a_redir_nb, Acu_conn_rq_redir_nb_size Filling order: 4	x	x	x	x										
Acu_conn_rq_a_ss_cnip_name, Acu_conn_rq_ss_cnip_name_size Filling order: 11												x		
Acu_conn_rq_a_tsp_ie_list, Acu_conn_rq_tsp_ie_list_size Filling order: 12	x	x	x	x	x	x	x	x	x			x	x	x
Acu_conn_rq_a_uui, Acu_conn_rq_uui_size Filling order: 5			x		x	x			x	x	x			
Acu_conn_rq_auto_dial														
Acu_conn_rq_call_appear		x	x											
Acu_conn_rq_callid_rq			x	x										
Acu_conn_rq_called_nb_plan		x	x	x	x					x	x	x		x
Acu_conn_rq_called_nb_sub_odd_even									x					x
Acu_conn_rq_called_nb_sub_type									x	x	x			x
Acu_conn_rq_called_nb_type		x	x	x	x	x		x		x	x	x		x
Acu_conn_rq_calling_nb2_pres						x								
Acu_conn_rq_calling_nb2_screen						x								
Acu_conn_rq_calling_nb2_type						x								

Macro	4ESS	E10	NI2	DMS	ETSI	VN6	HKG	AUSTEL1	NTT	KOREA	TAIWAN	QSIG	DPNSS	T1607
Acu_conn_rq_calling_nb_plan	x	x	x	x	x		x			x	x	x		x
Acu_conn_rq_calling_nb_pres					x	x	x	x	x	x	x	x		x
Acu_conn_rq_calling_nb_screen					x	x	x	x	x	x	x	x		x
Acu_conn_rq_calling_nb_sub_odd_even					x				x	x	x			
Acu_conn_rq_calling_nb_sub_type					x				x					x
Acu_conn_rq_calling_nb_type		x	x	x	x	x	x	x		x	x	x		x
Acu_conn_rq_charging_rq														
Acu_conn_rq_data_bits														
Acu_conn_rq_data_chani	x	x	x	x	x	x	x	x	x	x	x	x		x
Acu_conn_rq_data_chani_excl	x	x	x	x	x	x	x	x	x	x	x	x		x
Acu_conn_rq_data_chani_nai	x	x	x	x										
Acu_conn_rq_data_chani_nb	x	x	x	x	x	x	x	x	x	x	x	x		x
Acu_conn_rq_data_chani_tab	x	x	x	x	x	x	x	x	x	x	x	x		x
Acu_conn_rq_data_chani_tab_nai	x	x	x	x										
Acu_conn_rq_ext_parms_lgth	x													x
Acu_conn_rq_ext_parms_nb	x													x
Acu_conn_rq_ident_denied_rq														
Acu_conn_rq_identification_denied_rq														
Acu_conn_rq_p_display[***i***]														
Acu_conn_rq_parity														

Macro	**4ESS**	**E10**	**NI2**	**DMS**	**ETSI**	**VN6**	**HKG**	**AUSTEL1**	**NTT**	**KOREA**	**TAIWAN**	**QSIG**	**DPNSS**	**T1607**
Acu_conn_rq_pcs_user_protocol						x								
Acu_conn_rq_ph_rate														
Acu_conn_rq_priority														
Acu_conn_rq_progress_description														x
Acu_conn_rq_progress_ind_nb														x
Acu_conn_rq_progress_location														x
Acu_conn_rq_redir_nb_plan	x	x	x	x										
Acu_conn_rq_redir_nb_pres	x	x	x	x										
Acu_conn_rq_redir_nb_reason	x	x	x	x										
Acu_conn_rq_redir_nb_screen	x	x	x	x										
Acu_conn_rq_redir_nb_type	x	x	x	x										
Acu_conn_rq_sending_complete					x	x	x			x	x	x	x	x
Acu_conn_rq_service	x	x	x	x	x	x	x	x	x					
Acu_conn_rq_ss_cnip_name_active												x		
Acu_conn_rq_ss_cnip_name_pres												x		
Acu_conn_rq_stop_bits														
Acu_conn_rq_syn_asyn														
Acu_conn_rq_transfer_conn_id														
Acu_conn_rq_transfer_rq														
Acu_conn_rq_user_rate														

Macro	4ESS	E10	NI2	DMS	ETSI	VN6	HKG	AUSTEL1	NTT	KOREA	TAIWAN	QSIG	DPNSS	T1607
Acu_conn_rq_uui_protocol														
Acu_conn_rq_x_display_nb														
Acu_conn_rq_x_display_size[*i*]														
Acu_conn_rq_x_display_total_size														
Acu_conn_rq_x_display_type[*i*]														

ACU_CONN_RQ macro descriptions and Q.931 IE

Macro	Description	Q.931 IE
Acu_conn_rq_a_called_nb, Acu_conn_rq_called_nb_size Filling order: 1	Pointer to the called number.	Called party number (+ called subaddress)
Acu_conn_rq_a_called_nb_sub, Acu_conn_rq_a_called_nb_sub_size Filling order: 9	Pointer to (and size of) buffer containing called subaddress.	Calling party subaddress
Acu_conn_rq_a_calling_nb, Acu_conn_rq_calling_nb_size Filling order: 2	Calling number.	Calling party number (+ calling subaddress)
Acu_conn_rq_a_calling_nb2, Acu_conn_rq_calling_nb2_size Filling order: 3	Second calling number.	Second calling party number (+ calling subaddress)
Acu_conn_rq_a_calling_nb_sub, Acu_conn_rq_a_calling_nb_sub_size Filling order: 10	Pointer to (and size of) buffer containing calling subaddress.	Calling party subaddress
Acu_conn_rq_a_calling_name, Acu_conn_rq_calling_name_size	Pointer to (and size of) buffer containing calling name.	Calling name

Macro	Description	Q.931 IE
Acu_conn_rq_a_display, Acu_conn_rq_display_size Filling order: 7	Pointer to (and size of) buffer containing ISDN display information (optional).	Display
Acu_conn_rq_a_display_list	Pointer to display structure.	Display
Acu_conn_rq_a_ext_parms	Pointer to buffer containing extended parameters.	Network specific facilities
Acu_conn_rq_a_facility, Acu_conn_rq_facility_size Filling order: 6	Pointer to (and size of) buffer containing ISDN facility information (optional).	Facility
Acu_conn_rq_a_layer_1_info	Layer 1 information.	Low layer compatibility
Acu_conn_rq_a_pcs_user, Acu_conn_rq_pcs_user_size Filling order: 8	Pointer to (and size of) buffer containing ISDN PCS-to-user/user-to-PCS information. See PCS-user information elements for more information on this IE.	PCS-user
Acu_conn_rq_a_redir_nb, Acu_conn_rq_redir_nb_size Filling order: 4	Redirecting number.	Redirecting number
Acu_conn_rq_a_ss_cnip_name, Acu_conn_rq_ss_cnip_name_size Filling order: 11	Pointer to (and size of) buffer containing calling name identification presentation (CNIP) name.	CNIP
Acu_conn_rq_a_tsp_ie_list, Acu_conn_rq_tsp_ie_list_size Filling order: 12	Pointer to (and size of) transparent IE buffer.	Does not map to an IE.
Acu_conn_rq_a_uui, Acu_conn_rq_uui_size Filling order: 5	Pointer to (and size of) buffer containing ISDN user-to-user information (optional).	User-user
Acu_conn_rq_auto_dial	OFF: Go off hook only. ON: Automatically dial number.	Not used.

Macro	Description	Q.931 IE
Acu_conn_rq_call_appear	Call appearance.	Call appearance
Acu_conn_rq_callid_rq	Set to 1 to request a ***callid***.	Does not map to an IE.
Acu_conn_rq_called_nb_plan	Called number plan. See Plan values for a list of valid values. Default: N_PLAN_UNKNOWN. Default for VN6 and AusTel variants: N_PLAN_ISDN	Called party number
Acu_conn_rq_called_nb_sub_odd_even	Called subaddress odd/even. Valid values include: SUBADDRESS_ODD: Odd number of address signals SUBADDRESS_EVEN: Even number of address signals	Called party subaddress
Acu_conn_rq_called_nb_sub_type	Called subaddress number type. Valid values include: SUBADDRESS_TYPE_NSAP: NSAP SUBADDRESS_TYPE_USER: User specified	Called party subaddress
Acu_conn_rq_called_nb_type	Called number type. See Number type values for a list of valid values. Default: N_TYPE_UNKNOWN. Default for the 4ESS and VN6 variants: N_TYPE_NATIONAL.	Called party number
Acu_conn_rq_calling_nb2_pres	Second calling number presentation. Allowed values include: N_PRES_ALLOWED: Presentation allowed (default) N_PRES_RESTRICTED: Presentation restricted N_PRES_NOT_AVAILABLE: Presentation not available	Calling party number

Macro	**Description**	**Q.931 IE**
Acu_conn_rq_calling_nb2_screen	Second calling number screening indicator. See Screening indicator values for a list of valid values. Default: N_SCREEN_USER_PROVIDED (N_SCREEN_USER_PASSED for AusTel).	Calling party number
Acu_conn_rq_calling_nb2_type	Second calling number type. See Number type values for a list of valid values. Default: N_TYPE_UNKNOWN Default for the 4ESS and VN6 variants: N_TYPE_NATIONAL	Calling party number
Acu_conn_rq_calling_nb_plan	Calling number plan. See Plan values for a list of valid values. Default: N_PLAN_UNKNOWN Default for VN6 and AusTel variants: N_PLAN_ISDN	Calling party number
Acu_conn_rq_calling_nb_pres	Calling number presentation. Allowed values include: N_PRES_ALLOWED: Presentation allowed (default) N_PRES_RESTRICTED: Presentation restricted N_PRES_NOT_AVAILABLE: Presentation not available	Calling party number
Acu_conn_rq_calling_nb_screen	Calling number screening indicator. See Screening indicator values for a list of valid values. Default: N_SCREEN_USER_PROVIDED (N_SCREEN_USER_PASSED for AusTel).	Calling party number
Acu_conn_rq_calling_nb_sub_odd_even	Called subaddress odd/even. Valid values include: SUBADDRESS_ODD: Odd number of address signals SUBADDRESS_EVEN: Even number of address signals	Calling party subaddress

Macro	Description	Q.931 IE
Acu_conn_rq_calling_nb_sub_type	Calling subaddress number type. Valid values include: SUBADDRESS_TYPE_NSAP: NSAP SUBADDRESS_TYPE_USER: User specified	Calling party subaddress
Acu_conn_rq_calling_nb_type	Calling number type. See Number type values for a list of valid values. Default: N_TYPE_UNKNOWN. Default for the 4ESS and VN6 variants: N_TYPE_NATIONAL.	Calling party number
Acu_conn_rq_charging_rq	Request charging (ON/OFF).	Not used.
Acu_conn_rq_data_bits	Number of data bits for V.110 and V.120 services. Available values include: ACU_DATA_BIT_5: 5 data bits ACU_DATA_BIT_7: 7 data bits ACU_DATA_BIT_8: 8 data bits	Channel identification
Acu_conn_rq_data_chani	Data channel to use (B1, B2, ... D, or 0 for any). This is used as an alias for Acu_conn_rq_data_chani_tab.	Channel identification
Acu_conn_rq_data_chani_excl	Channel IDs are preferred (OFF) or exclusive (ON).	Channel identification
Acu_conn_rq_data_chani_nai	NAI.	Channel identification
Acu_conn_rq_data_chani_nb	Number of channel IDs in the chani_list field. If no channel IDs, use 0.	Channel identification
Acu_conn_rq_data_chani_tab	Channel ID.	Channel identification
Acu_conn_rq_data_chani_tab_nai	NAI.	Channel identification
Acu_conn_rq_ext_parms_lgth	Total length of buffer containing extended parameters.	Network specific facilities

Macro	**Description**	**Q.931 IE**
Acu_conn_rq_ext_parms_nb	Number of parameters in buffer containing extended parameters.	Network specific facilities
Acu_conn_rq_ident_denied_rq	Request identity denied (ON/OFF).	Facility
Acu_conn_rq_identification_denied_rq	Alias macro for Acu_conn_rq_ident_denied_rq.	Facility
Acu_conn_rq_p_display[*i*]	Pointer to occurrence ***i***.	Display
Acu_conn_rq_parity	Parity for V.110 and V.120 services. Available values: ACU_ODD: odd parity ACU_EVEN: even parity ACU_NO_PARITY: no parity	Bearer capability
Acu_conn_rq_pcs_user_protocol	Protocol discriminator for the pcs_user information element. See PCS-user information elements for more information on this IE.	PCS-user
Acu_conn_rq_ph_rate	Physical rate (for all services).	Low layer compatibility
Acu_conn_rq_priority	Phone call priority normal/urgent.	Low layer compatibility
Acu_conn_rq_progress_description	Description for information element 0. See Progress description values for a list of valid values.	Progress indication
Acu_conn_rq_progress_ind_nb	Number of progress indication information elements.	Progress indication
Acu_conn_rq_progress_location	Location of information element 0. See Location values for a list of valid values and default setting information.	Progress indication

Macro	**Description**	**Q.931 IE**
Acu_conn_rq_redir_nb_plan	Redirecting number plan. See Plan values for a list of valid values. Default: N_PLAN_UNKNOWN Default for VN6 and AusTel variants: N_PLAN_ISDN	Redirecting number
Acu_conn_rq_redir_nb_pres	Redirecting number presentation. Allowed values: N_PRES_ALLOWED: Presentation allowed (default) N_PRES_RESTRICTED: Presentation restricted N_PRES_NOT_AVAILABLE: Presentation not available	Redirecting number
Acu_conn_rq_redir_nb_reason	Reason for redirection. See Redirecting reason values for a list of valid values. Default: REASON_UNKNOWN.	Redirecting number
Acu_conn_rq_redir_nb_screen	Redirecting number screening indicator. See Screening indicator values for a list of valid values. Default: N_SCREEN_USER_PROVIDED (N_SCREEN_USER_PASSED for AusTel)	Redirecting number
Acu_conn_rq_redir_nb_type	Redirecting number type. See Number type values for a list of valid values. Default: N_TYPE_UNKNOWN Default for the 4ESS and VN6 variants: N_TYPE_NATIONAL	Redirecting number
Acu_conn_rq_sending_complete	Indicates if sending-complete information element is generated (ON/OFF).	Sending complete
Acu_conn_rq_service	Requested service. See Service values for a list of valid values.	bc + hlc + llc

Macro	Description	Q.931 IE
Acu_conn_rq_ss_cnip_name_active	Indicates calling name identification presentation (CNIP) supplementary service should be invoked.	CNIP
Acu_conn_rq_ss_cnip_name_pres	Calling name identification presentation mode. Allowed values include: N_PRES_ALLOWED: Presentation allowed N_PRES_RESTRICTED: Presentation restricted N_PRES_NOT_AVAILABLE: Presentation not available	CNIP
Acu_conn_rq_stop_bits	Number of stop bits for V.110 and V.120 services only). Available values include: ACU_STOP_BIT_1: 1 stop bit ACU_STOP_BIT_1_5: 1. 5 stop bits ACU_STOP_BIT_2: 2 stop bits	Low layer compatibility
Acu_conn_rq_syn_asyn	Synchronous/asynchronous for V.110 and V.120 services. Available values include: ACU_SYN: Synchronous mode ACU_ASYN: Asynchronous mode	Low layer compatibility
Acu_conn_rq_transfer_conn_id	Connection ID of the call to be transferred.	Facility
Acu_conn_rq_transfer_rq	Request transfer (ON/OFF).	Facility
Acu_conn_rq_user_rate	Requested user rate for V.110 and V.120 services. See User rate values for a list of valid values.	Low layer compatibility
Acu_conn_rq_uui_protocol	UUI protocol discriminator value. See UUI protocol discriminator values for a list of valid values. Default: UUI_IA5 (UUI_USER_SPF for HKG variant).	User-to-user information
Acu_conn_rq_x_display_nb	Number of present occurrences.	Display

Macro	**Description**	**Q.931 IE**
Acu_conn_rq_x_display_size[***i***]	Size of occurrence ***i*** (optional).	Display
Acu_conn_rq_x_display_total_size	Total size of the stored strings.	Display
Acu_conn_rq_x_display_type[***i***]	Type of display ***i***.	Display

ACU_CONN_RS

This topic describes:

- ACU_CONN_RS protocol variants
- ACU_CONN_RS macro descriptions and Q.931 IE

Note: Address parameters must be set according to the filling order. If one of them is unused, its size field must be set to 0.

Purpose

Answers an incoming call.

Conn_id

An allocated call.

ACU_CONN_RS protocol variants

The following table lists the variants under which each macro is supported for this primitive. In this table, x indicates both NT and TE sides.

Macro	**4ESS**	**E10**	**NI2**	**DMS**	**ETSI**	**VN6**	**HKG**	**AUSTEL1**	**NTT**	**KOREA**	**TAIWAN**	**QSIG**	**DPNSS**	**T1607**
Acu_conn_rs_a_connected_nb, Acu_conn_rs_connected_nb_size Filling order: 4									x			x		x
Acu_conn_rs_a_connected_sub, Acu_conn_rs_connected_sub_size Filling order: 5									x			x		x
Acu_conn_rs_a_date_time					x	x								

Macro	4ESS	E10	NI2	DMS	ETSI	VN6	HKG	AUSTEL1	NTT	KOREA	TAIWAN	QSIG	DPNSS	T1607
Acu_conn_rs_a_display, Acu_conn_rs_display_size Filling order: 3														
Acu_conn_rs_a_display_list										x	x			
Acu_conn_rs_a_ext_parms														
Acu_conn_rs_a_facility, Acu_conn_rs_facility_size Filling order: 2					x	x				x	x	x		
Acu_conn_rs_a_layer_1_info														
Acu_conn_rs_a_pcs_user, Acu_conn_rs_pcs_user_size Filling order: 6						x								
Acu_conn_rs_a_ss_cnip_name, Acu_conn_rs_ss_cnip_name_size Filling order: 7												x		
Acu_conn_rs_a_tsp_ie_list, Acu_conn_rs_tsp_ie_list_size Filling order: 8	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Acu_conn_rs_a_uui, Acu_conn_rs_uui_size Filling order: 1			x		x	x			x	x	x			
Acu_conn_rs_charging														
Acu_conn_rs_charging_available														
Acu_conn_rs_charging_multi														
Acu_conn_rs_charging_period														
Acu_conn_rs_charging_type														

Macro	**4ESS**	**E10**	**NI2**	**DMS**	**ETSI**	**VN6**	**HKG**	**AUSTEL1**	**NTT**	**KOREA**	**TAIWAN**	**QSIG**	**DPNSS**	**T1607**
Acu_conn_rs_charging_value														
Acu_conn_rs_connected_nb_plan									x			x		x
Acu_conn_rs_connected_nb_pres									x			x		x
Acu_conn_rs_connected_nb_screen									x			x		x
Acu_conn_rs_connected_nb_type									x			x		x
Acu_conn_rs_connected_sub_odd_even									x			x		x
Acu_conn_rs_connected_sub_type									x			x		x
Acu_conn_rs_data_bits														
Acu_conn_rs_data_chani	x	x	x	x	x	x	x	x	x	x	x	x		x
Acu_conn_rs_data_chani_excl	x	x	x	x	x	x	x	x	x	x	x	x		x
Acu_conn_rs_data_chani_nai														
Acu_conn_rs_data_chani_nb	x	x	x											
Acu_conn_rs_data_chani_tab[***i***]	x	x	x											
Acu_conn_rs_data_chani_tab_nai[***i***]														
Acu_conn_rs_date_available					x	x								
Acu_conn_rs_day					x	x								
Acu_conn_rs_ext_parms_length														
Acu_conn_rs_ext_parms_nb														
Acu_conn_rs_hour					x	x				x	x			
Acu_conn_rs_minute					x	x				x	x			

Macro	**4ESS**	**E10**	**NI2**	**DMS**	**ETSI**	**VN6**	**HKG**	**AUSTEL1**	**NTT**	**KOREA**	**TAIWAN**	**QSIG**	**DPNSS**	**T1607**
Acu_conn_rs_month					x	x				x	x			
Acu_conn_rs_p_display[***i***]														
Acu_conn_rs_parity														
Acu_conn_rs_pcs_user_protocol						x								
Acu_conn_rs_ph_rate														
Acu_conn_rs_priority														
Acu_conn_rs_second														
Acu_conn_rs_service	x	x	x	x	x	x	x	x						
Acu_conn_rs_ss_cnip_name_active												x		
Acu_conn_rs_ss_cnip_name_pres												x		
Acu_conn_rs_stop_bits														
Acu_conn_rs_syn_asyn														
Acu_conn_rs_user_rate														
Acu_conn_rs_uui_protocol										x	x			
Acu_conn_rs_x_display_nb														
Acu_conn_rs_x_display_size[***i***]														
Acu_conn_rs_x_display_total_size														
Acu_conn_rs_x_display_type[***i***]														
Acu_conn_rs_year					x	x				x	x			

ACU_CONN_RS macro descriptions and Q.931 IE

Macro	Description	Q.931 IE
Acu_conn_rs_a_connected_nb, Acu_conn_rs_connected_nb_size Filling order: 4	Pointer to (and size of) buffer containing connected number.	Connected party address
Acu_conn_rs_a_connected_sub, Acu_conn_rs_connected_sub_size Filling order: 5	Pointer to (and size of) buffer containing connected subaddress.	Connected party subaddress
Acu_conn_rs_a_date_time	Pointer to date_time.	Date/time
Acu_conn_rs_a_display, Acu_conn_rs_display_size Filling order: 3	Pointer to (and size of) buffer containing ISDN display information (optional).	Display
Acu_conn_rs_a_display_list	Pointer to display structure.	Display
Acu_conn_rs_a_ext_parms	Pointer to buffer containing extended parameters.	Not used.
Acu_conn_rs_a_facility, Acu_conn_rs_facility_size Filling order: 2	Pointer to (and size of) buffer containing ISDN facility information (optional).	Facility
Acu_conn_rs_a_layer_1_info	Pointer to structure containing layer 1 information.	Low layer compatibility
Acu_conn_rs_a_pcs_user, Acu_conn_rs_pcs_user_size Filling order: 6	Pointer to (and size of) buffer containing ISDN PCS-to-user/user-to-PCS information. See PCS-user information elements for more information on this IE.	PCS-user
Acu_conn_rs_a_ss_cnip_name, Acu_conn_rs_ss_cnip_name_size Filling order: 7	Pointer to (and size of) buffer containing calling name identification presentation (CNIP) name.	CNIP
Acu_conn_rs_a_tsp_ie_list, Acu_conn_rs_tsp_ie_list_size Filling order: 8	Pointer to (and size of) transparent IE buffer.	Does not map to an IE.

Macro	**Description**	**Q.931 IE**
Acu_conn_rs_a_uui, Acu_conn_rs_uui_size Filling order: 1	Pointer to (and size of) buffer containing ISDN user-to-user information (optional).	User-user
Acu_conn_rs_charging	Charging value (number of units).	Not used.
Acu_conn_rs_charging_available	Charging information available indicator.	Not used.
Acu_conn_rs_charging_multi	Charging multiplier.	Not used.
Acu_conn_rs_charging_period	Charging period.	Not used.
Acu_conn_rs_charging_type	Charging type. See Charging type values for a list of valid values.	Not used.
Acu_conn_rs_charging_value	Charging value (number of units).	Not used.
Acu_conn_rs_connected_nb_plan	Connected number plan. See Plan values for a list of valid values.	Connected party address
Acu_conn_rs_connected_nb_pres	Connected number presentation. Allowed values: N_PRES_ALLOWED: Presentation allowed N_PRES_RESTRICTED: Presentation restricted N_PRES_NOT_AVAILABLE: Presentation not available	Connected party address
Acu_conn_rs_connected_nb_screen	Connected number screening indicator. See Screening indicator values for a list of valid values.	Connected party address
Acu_conn_rs_connected_nb_type	Connected number type. See Number type values for a list of valid values.	Connected party address

Macro	Description	Q.931 IE
Acu_conn_rs_connected_sub_odd_even	Connected subaddress odd/even. Valid values: SUBADDRESS_ODD: Odd number of address signals SUBADDRESS_EVEN:Even number of address signals	Connected party subaddress
Acu_conn_rs_connected_sub_type	Connected subaddress type. See Number type values for a list of valid values.	Connected party subaddress
Acu_conn_rs_data_bits	Number of data bits for V.110 and V.120 services. Available values include: ACU_DATA_BIT_5: 5 data bits ACU_DATA_BIT_7: 7 data bits ACU_DATA_BIT_8: 8 data bits.	Not used.
Acu_conn_rs_data_chani	Data channel to use (B1, B2, ... D, or 0 for any).	Channel identification
Acu_conn_rs_data_chani_excl	Channel IDs are preferred (OFF) or exclusive (ON).	Channel identification
Acu_conn_rs_data_chani_nai	NAI.	Channel identification
Acu_conn_rs_data_chani_nb	Number of channel IDs in the chani_list field. If no channel IDs, use 0. (For X25_PACKET services only; otherwise unused.)	Channel identification
Acu_conn_rs_data_chani_tab[***i***]	Channel ID ***i***.	Channel identification
Acu_conn_rs_data_chani_tab_nai[***i***]	NAI ***i***.	Channel identification
Acu_conn_rs_date_available	ON if the information is available, else OFF.	Date/time
Acu_conn_rs_day	Day.	Date/time

Macro	**Description**	**Q.931 IE**
Acu_conn_rs_ext_parms_length	Total length of buffer containing extended parameters.	Not used.
Acu_conn_rs_ext_parms_nb	Number of parameters in buffer containing extended parameters.	Not used.
Acu_conn_rs_hour	Hour.	Date/time
Acu_conn_rs_minute	Minute.	Date/time
Acu_conn_rs_month	Month.	Date/time
Acu_conn_rs_p_display[***i***]	Pointer to occurrence ***i***.	Display
Acu_conn_rs_parity	Parity for V.110 and V.120 services. Available values include: ACU_ODD: odd parity ACU_EVEN: even parity ACU_NO_PARITY: no parity	Bearer capability
Acu_conn_rs_pcs_user_protocol	Protocol discriminator for the pcs_user information element. See PCS-user information elements for more information on this IE.	PCS-user
Acu_conn_rs_ph_rate	Physical rate (for all services).	Low layer compatibility
Acu_conn_rs_priority	Answer priority. Valid values include: ACU_PHIGH, ACU_PLOW	Low layer compatibility
Acu_conn_rs_second	Second.	Date/time
Acu_conn_rs_service	Agreed service. Can be different than the one stored in ACU_CONN_IN. See Service values for a list of valid values.	bc + hlc + llc

Macro	**Description**	**Q.931 IE**
Acu_conn_rs_ss_cnip_name_active	Indicates calling name identification presentation (CNIP) supplementary service should be invoked.	CNIP
Acu_conn_rs_ss_cnip_name_pres	Calling name identification presentation (CNIP) mode. Allowed values include: N_PRES_ALLOWED: Presentation allowed N_PRES_RESTRICTED: Presentation restricted N_PRES_NOT_AVAILABLE: Presentation not available	CNIP
Acu_conn_rs_stop_bits	Number of stop bits for V.110 and V.120 services. Available values include: ACU_STOP_BIT_1: 1 stop bit ACU_STOP_BIT_1_5: 1.5 stop bits ACU_STOP_BIT_2: 2 stop bits	Low layer compatibility
Acu_conn_rs_syn_asyn	Synchronous/asynchronous for V.110 and V.120 services. Available values include: ACU_SYN: Synchronous mode ACU_ASYN: Asynchronous mode	Low layer compatibility
Acu_conn_rs_user_rate	User rate for V.110 and V.120 services. See User rate values for a list of valid values.	Low layer compatibility
Acu_conn_rs_uui_protocol	UUI protocol discriminator value. See UUI protocol discriminator values for a list of valid values. Default: UUI_IA5 (UUI_USER_SPF for HKG variant)	User-user
Acu_conn_rs_x_display_nb	Number of present occurrences.	Display
Acu_conn_rs_x_display_size[*i*]	Size of occurrence ***i*** (optional).	Display

Macro	**Description**	**Q.931 IE**
Acu_conn_rs_x_display_total_size	Total size of the stored strings.	Display
Acu_conn_rs_x_display_type[***i***]	Type of display ***i***.	Display
Acu_conn_rs_year	Year.	Date/time

ACU_D_CHANNEL_STATUS_IN

This topic describes:

- ACU_D_CHANNEL_STATUS_IN protocol variants
- ACU_D_CHANNEL_STATUS_IN macro descriptions and Q.931 IE

This message primitive can be returned when:

- An ACU_D_CHANNEL_STATUS_RQ primitive is sent to the stack.
- The Acu_send_d_channel_status_change bit is set in the Acu_behaviour substructure included in ISDN_PROTOCOL_PARMS_Q931CC.

This message primitive must be sent to the ACU_SAPI_MGT SAPI, rather than the ACU_SAPI. To do so, specify ACU_SAPI_MGT in the to_sapi field in the outgoing ACU_MESSAGE structure.

Purpose

Indicates the status of the D channel.

Conn_id

An allocated call.

ACU_D_CHANNEL_STATUS_IN protocol variants

The following table lists the variants under which each macro is supported for this primitive. In this table, x indicates both NT and TE sides.

Macro	**4ESS**	**EI0**	**N12**	**DMS**	**ETSI**	**VN6**	**HKG**	**AUSTEL1**	**NTT**	**KOREA**	**TAIWAN**	**QSIG**	**DPNSS**	**T1607**
Acu_d_channel_state	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Acu_d_channel_nb													x	

ACU_D_CHANNEL_STATUS_IN macro descriptions and Q.931 IE

Macro	Description	Q.931 IE
Acu_d_channel_state	State of D channel: 0: OFF 1: ON	Does not map to an IE.
Acu_d_channel_nb	Equivalent to B channel number (for DPNSS only).	Does not map to an IE.

ACU_D_CHANNEL_STATUS_RQ

This topic describes:

- ACU_D_CHANNEL_STATUS_RQ protocol variants
- ACU_D_CHANNEL_STATUS_RQ macro descriptions and Q.931 IE

This primitive must be sent to the ACU_SAPI_MGT SAPI, rather than the ACU_SAPI. To do so, specify ACU_SAPI_MGT in the to_sapi field in the outgoing ACU_MESSAGE structure.

Purpose

Inquires the status of the D channel.

Conn_id

Any connection ID.

ACU_D_CHANNEL_STATUS_RQ protocol variants

The following table lists the variants under which each macro is supported for this primitive. In this table, x indicates both NT and TE sides.

Macro	4ESS	E10	NI2	DMS	ETSI	VN6	HKG	AUSTEL1	NTT	KOREA	TAIWAN	QSIG	DPNSS	T1607
Acu_d_channel_state	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Acu_d_channel_nb													x	

ACU_D_CHANNEL_STATUS_RQ macro descriptions and Q.931 IE

Macro	Description	Q.931 IE
Acu_d_channel_state	State of D channel: 0: OFF 1: ON	Does not map to an IE.
Acu_d_channel_nb	Equivalent to B channel number (for DPNSS only).	Does not map to an IE.

ACU_DIGIT_IN

This topic describes:

- ACU_DIGIT_IN protocol variants
- ACU_DIGIT_IN macro descriptions and Q.931 IE

Purpose

Receives called number digits in overlap receiving mode.

Conn_id

An allocated call.

ACU_DIGIT_IN protocol variants

The following table lists the variants under which each macro is supported for this primitive. In this table, NT = NT side only, TE = TE side only, and x = both NT and TE sides.

Macro	4ESS	E10	NI2	DMS	ETSI	VN6	HKG	AUSTEL1	NTT	KOREA	TAIWAN	QSIG	DPNSS	T1607
Acu_digit_in_a_digit, Acu_digit_in_digit_size					x	x	TE		NT	x	x	x	x	
Acu_digit_in_a_display, Acu_digit_in_display_size					x	x	TE		NT	x	x	x		
Acu_digit_in_a_display_list					x	x			NT	x	x	x		
Acu_digit_in_a_q931, Acu_digit_in_q931_size					x	x	TE		NT	x	x	x	x	
Acu_digit_in_digit_type					x	x				x	x	x		

Macro	4ESS	E10	NI2	DMS	ETSI	VN6	HKG	AUSTEL1	NTT	KOREA	TAIWAN	QSIG	DPNSS	T1607
Acu_digit_in_digit_plan					x	x				x	x	x		
Acu_digit_in_sending_complete					x	x	TE		NT	x	x	x	x	
Acu_digit_in_x_display_nb					x	x	TE		NT	x	x	x		
Acu_digit_in_x_display_size[***i***]					x	x	TE		NT	x	x	x		
Acu_digit_in_x_display_total_size					x	x	TE		NT	x	x	x		
Acu_digit_in_x_display_type[***i***]					x	x	TE		NT	x	x	x		
Acu_digit_in_x_p_display[***i***]					x	x	TE		NT	x	x	x		

ACU_DIGIT_IN macro descriptions and Q.931 IE

Macro	Description	Q.931 IE
Acu_digit_in_a_digit, Acu_digit_in_digit_size	Address and size of digit string.	Called party number
Acu_digit_in_a_display, Acu_digit_in_display_size	Pointer to (and size of) buffer containing ISDN display information (optional).	Display
Acu_digit_in_a_display_list	Pointer to display structure.	Display
Acu_digit_in_a_q931, Acu_digit_in_q931_size	Pointer to (and size of) buffer containing raw data of incoming Q.931 message.	Does not map to an IE.
Acu_digit_in_digit_type	Called number type.	Called party number
Acu_digit_in_digit_plan	Called number plan.	Called party number
Acu_digit_in_sending_complete	Indicates if the sending-complete information element was received (ON/OFF).	Sending complete

Macro	Description	Q.931 IE
Acu_digit_in_x_display_nb	Number of present occurrences.	Display
Acu_digit_in_x_display_size[***i***]	Size of occurrence ***i*** (optional).	Display
Acu_digit_in_x_display_total_size	Total size of the stored strings.	Display
Acu_digit_in_x_display_type[***i***]	Type of display ***i***.	Display
Acu_digit_in_x_p_display[***i***]	Pointer to occurrence ***i***.	Display

ACU_DIGIT_RQ

This topic describes:

- ACU_DIGIT_RQ protocol variants
- ACU_DIGIT_RQ macro descriptions and Q.931 IE

Note: Address parameters must be set according to the filling order. If one of them is unused, its ***size*** field must be set to 0.

Purpose

Requests that called digits be sent in overlap receiving mode.

Conn_id

An allocated call.

ACU_DIGIT_RQ protocol variants

The following table lists the variants under which each macro is supported for this primitive. In this table, NT = NT side only, TE = TE side only, and x = both NT and TE sides.

Macro	4ESS	E10	NI2	DMS	ETSI	VN6	HKG	AUSTEL1	NTT	KOREA	TAIWAN	QSIG	DPNSS	T1607
Acu_digit_rq_a_digit, Acu_digit_rq_digit_size Filling order: 1					x	x	TE		NT	x	x	TE	x	
Acu_digit_rq_a_display, Acu_digit_rq_display_size Filling order: 2							TE		NT			TE		

Macro	4ESS	E10	NI2	DMS	ETSI	VN6	HKG	AUSTEL1	NTT	KOREA	TAIWAN	QSIG	DPNSS	T1607
Acu_digit_rq_a_tsp_ie_list, Acu_digit_rq_tsp_ie_list_size Filling order: 3					x	x		x	x	x	x			x
Acu_digit_rq_type					x	x				x	x	TE		
Acu_digit_rq_digit_plan					x	x				x	x	TE		
Acu_digit_rq_sending_complete					x	x	x		x	x	x	x	x	
Acu_digit_rq_x_display_nb														
Acu_digit_rq_x_display_size[***i***]														
Acu_digit_rq_x_display_total_size														
Acu_digit_rq_x_display_type[***i***]														
Acu_digit_rq_x_p_display[***i***]														

ACU_DIGIT_RQ macro descriptions and Q.931 IE

Macro	Description	Q.931 IE
Acu_digit_rq_a_digit, Acu_digit_rq_digit_size Filling order: 1	Address and size of digit string.	Called party number
Acu_digit_rq_a_display, Acu_digit_rq_display_size Filling order: 2	Pointer to (and size of) buffer containing ISDN display information (optional).	Display
Acu_digit_rq_a_tsp_ie_list, Acu_digit_rq_tsp_ie_list_size Filling order: 3	Pointer to (and size of) transparent IE buffer.	Does not map to an IE.
Acu_digit_rq_type	Called number type.	Called party number

Macro	Description	Q.931 IE
Acu_digit_rq_digit_plan	Called number plan.	Called party number
Acu_digit_rq_sending_complete	Requests that sending-complete information element be sent.	Sending complete
Acu_digit_rq_x_display_nb	Number of present occurrences.	Display
Acu_digit_rq_x_display_size[***i***]	Size of occurrence ***i*** (optional).	Display
Acu_digit_rq_x_display_total_size	Total size of the stored strings.	Display
Acu_digit_rq_x_display_type[***i***]	Type of display ***i***.	Display
Acu_digit_rq_x_p_display[***i***]	Pointer to occurrence ***i***.	Display

ACU_ERR_IN

This topic describes:

- ACU_ERR_IN protocol variants
- ACU_ERR_IN macro descriptions and Q.931 IE

Purpose

Indicates an error.

Conn_id

An allocated call.

ACU_ERR_IN protocol variants

The following table lists the variants under which each macro is supported for this primitive. In this table, x indicates both NT and TE sides.

Macro	4ESS	E10	NI2	DMS	ETSI	VN6	HKG	AUSTEL1	NTT	KOREA	TAIWAN	QSIG	DPNSS	T1607
Acu_err_in_cause	x	x	x	x	x	x	x	x	x	x	x	x		x
Acu_err_in_diagnostics	x	x	x	x	x	x	x	x	x	x	x	x		x

ACU_ERR_IN macro descriptions and Q.931 IE

Macro	Description	Q.931 IE
Acu_err_in_cause	Error cause. See Error cause values for a list of valid values.	Does not map to an IE.
Acu_err_in_diagnostics	Diagnostic.	Does not map to an IE.

ACU_FACILITY_IN

This topic describes:

- ACU_FACILITY_IN protocol variants
- ACU_FACILITY_IN macro descriptions and Q.931 IE

Purpose

Indicates a FACILITY message.

Conn_id

An allocated call.

ACU_FACILITY_IN protocol variants

The following table lists the variants under which each macro is supported for this primitive. In this table, TE indicates the TE side only, and x indicates both NT and TE sides.

Macro	4ESS	E10	NI2	DMS	ETSI	VN6	HKG	AUSTEL1	NTT	KOREA	TAIWAN	QSIG	DPNSS	T1607
Acu_facility_a_called_address														
Acu_facility_a_calling_nb, Acu_facility_calling_nb_size														
Acu_facility_a_calling_nb2, Acu_facility_calling_nb2_size														
Acu_facility_a_display, Acu_facility_display_size														
Acu_facility_a_display_list														
Acu_facility_a_ext_parms														

Macro	4ESS	E10	NI2	DMS	ETSI	VN6	HKG	AUSTEL1	NTT	KOREA	TAIWAN	QSIG	DPNSS	T1607
Acu_facility_a_facility, Acu_facility_facility_size														
Acu_facility_a_pcs_user, Acu_facility_pcs_user_size						x								
Acu_facility_a_q931, Acu_facility_q931_size			TE		x							x	x	x
Acu_facility_action														
Acu_facility_button_nb														
Acu_facility_button_type														
Acu_facility_call_appear														
Acu_facility_code			TE		x							x		x
Acu_facility_ext_parms_lgth														
Acu_facility_ext_parms_nb														
Acu_facility_module_nb														
Acu_facility_pcs_user_protocol						x								
Acu_facility_service														
Acu_facility_switchhook														
Acu_facility_x_display_nb														
Acu_facility_x_display_size[***i***]														
Acu_facility_x_display_total_size														
Acu_facility_x_display_type[***i***]														
Acu_facility_p_display[***i***]														

ACU_FACILITY_IN macro descriptions and Q.931 IE

Macro	Description	Q.931 IE
Acu_facility_a_called_address	Pointer to buffer containing called address.	Not used.
Acu_facility_a_calling_nb, Acu_facility_calling_nb_size	Pointer to (and size of) buffer containing the calling number.	Not used.
Acu_facility_a_calling_nb2, Acu_facility_calling_nb2_size	Pointer to (and size of) buffer containing second calling number.	Not used.
Acu_facility_a_display, Acu_facility_display_size	Pointer to (and size of) buffer containing ISDN display information (optional).	Not used.
Acu_facility_a_display_list	Pointer to display structure.	Not used.
Acu_facility_a_ext_parms	Pointer to buffer containing extended parameters.	Not used.
Acu_facility_a_facility, Acu_facility_facility_size	Pointer to (and size of) buffer containing ISDN facility information (optional).	Facility
Acu_facility_a_pcs_user, Acu_facility_pcs_user_size	Pointer to (and size of) buffer containing ISDN PCS-to-user/user-to-PCS information. See PCS-user information elements for more information on this IE.	PCS-user
Acu_facility_a_q931, Acu_facility_q931_size	Pointer to (and size of) buffer containing raw data of incoming Q.931 message.	Does not map to an IE.
Acu_facility_action	Action. See Action code values for a list of valid values.	Not used.
Acu_facility_button_nb	Voice button number.	Not used.
Acu_facility_button_type	Voice button type.	Not used.
Acu_facility_call_appear	Call appearance.	Not used.
Acu_facility_code	Facility code. See Facility code values for a list of valid values.	Facility
Acu_facility_ext_parms_lgth	Total length of buffer containing extended parameters.	Not used.

Macro	**Description**	**Q.931 IE**
Acu_facility_ext_parms_nb	Number of parameters in buffer containing extended parameters.	Not used.
Acu_facility_module_nb	Voice module number.	Not used.
Acu_facility_pcs_user_protocol	Protocol discriminator for the pcs_user information element. See PCS-user information elements for more information on this IE.	PCS-user
Acu_facility_service	Requested service. See Service values for a list of valid values.	Not used.
Acu_facility_switchhook	Voice switchhook.	Not used.
Acu_facility_x_display_nb	Number of present occurrences.	Not used.
Acu_facility_x_display_size[***i***]	Size of occurrence ***i*** (optional).	Not used.
Acu_facility_x_display_total_size	Total size of the stored strings.	Not used.
Acu_facility_x_display_type[***i***]	Type of display ***i***.	Not used.
Acu_facility_p_display[***i***]	Pointer to occurrence ***i***.	Not used.

ACU_FACILITY_RQ

This topic describes:

- ACU_FACILITY_RQ protocol variants
- ACU_FACILITY_RQ macro descriptions and Q.931 IE

Note: Address parameters must be set according to the filling order. If one of them is unused, its size field must be set to 0.

Purpose

Requests that a FACILITY message be sent.

Conn_id

An allocated call.

ACU_FACILITY_RQ protocol variants

The following table lists the variants under which each macro is supported for this primitive. In this table, TE indicates the TE side only, and x indicates both NT and TE sides.

Macro	4ESS	E10	NI2	DMS	ETSI	VN6	HKG	AUSTEL1	NTT	KOREA	TAIWAN	QSIG	DPNSS	T1607
Acu_facility_a_called_address														
Acu_facility_a_calling_nb, Acu_facility_calling_nb_size Filling order: 2														
Acu_facility_a_calling_nb2, Acu_facility_calling_nb2_size Filling order: 3														
Acu_facility_a_display, Acu_facility_display_size Filling order: 4														
Acu_facility_a_ext_parms														
Acu_facility_a_facility, Acu_facility_facility_size Filling order: 1														
Acu_facility_a_pcs_user, Acu_facility_pcs_user_size Filling order: 5						x								
Acu_facility_a_tsp_ie_list, Acu_facility_tsp_ie_list_size Filling order: 6			TE		x							x		x
Acu_facility_button_nb														
Acu_facility_button_type														
Acu_facility_call_appear														
Acu_facility_code			TE		x							x		x
Acu_facility_conn_id														
Acu_facility_ext_parms_lgth														

Macro	4ESS	E10	NI2	DMS	ETSI	VN6	HKG	AUSTEL1	NTT	KOREA	TAIWAN	QSIG	DPNSS	T1607
Acu_facility_ext_parms_nb														
Acu_facility_module_nb														
Acu_facility_pcs_user_protocol						x								
Acu_facility_service														
Acu_facility_switchhook														
Acu_facility_x_display_nb														
Acu_facility_x_display_size[*i*]														
Acu_facility_x_display_total_size														
Acu_facility_x_display_type[*i*]														
Acu_facility_x_p_display[*i*]														

ACU_FACILITY_RQ macro descriptions and Q.931 IE

Macro	Description	Q.931 IE
Acu_facility_a_called_address	Pointer to buffer containing called address.	Not used.
Acu_facility_a_calling_nb, Acu_facility_calling_nb_size Filling order: 2	Pointer to (and size of) buffer containing the calling number.	Not used.
Acu_facility_a_calling_nb2, Acu_facility_calling_nb2_size Filling order: 3	Pointer to (and size of) buffer containing second calling number.	Not used.
Acu_facility_a_display, Acu_facility_display_size Filling order: 4	Pointer to (and size of) buffer containing ISDN display information (optional).	Not used.

Macro	**Description**	**Q.931 IE**
Acu_facility_a_ext_parms	Pointer to buffer containing extended parameters.	Not used.
Acu_facility_a_facility, Acu_facility_facility_size Filling order: 1	Pointer to (and size of) buffer containing ISDN facility information (optional).	Facility
Acu_facility_a_pcs_user, Acu_facility_pcs_user_size Filling order: 5	Pointer to (and size of) buffer containing ISDN PCS-to-user/user-to-PCS information. See PCS-user information elements for more information on this IE.	PCS-user
Acu_facility_a_tsp_ie_list, Acu_facility_tsp_ie_list_size Filling order: 6	Pointer to (and size of) transparent IE buffer.	Does not map to an IE.
Acu_facility_button_nb	Voice button number.	Not used.
Acu_facility_button_type	Voice button type.	Not used.
Acu_facility_call_appear	Call appearance.	Not used.
Acu_facility_code	Facility code. See Facility code values for a list of valid values.	Facility
Acu_facility_conn_id	Connection ID.	Not used.
Acu_facility_ext_parms_lgth	Total length of buffer containing extended parameters.	Not used.
Acu_facility_ext_parms_nb	Number of parameters in buffer containing extended parameters.	Not used.
Acu_facility_module_nb	Voice module number.	Not used.
Acu_facility_pcs_user_protocol	Protocol discriminator for the pcs_user information element. See PCS-user information elements for more information on this IE.	PCS-user
Acu_facility_service	Requested service. See Service values for a list of valid values.	Not used.
Acu_facility_switchhook	Voice switchhook.	Not used.

Macro	Description	Q.931 IE
Acu_facility_x_display_nb	Number of present occurrences.	Not used.
Acu_facility_x_display_size[***i***]	Size of occurrence ***i*** (optional).	Not used.
Acu_facility_x_display_total_size	Total size of the stored strings.	Not used.
Acu_facility_x_display_type (I)	Type of display ***i***.	Not used.
Acu_facility_x_p_display (I)	Pointer to occurrence ***i***.	Not used.

ACU_NOTIFY_IN

This topic describes:

- ACU_NOTIFY_IN protocol variants
- ACU_NOTIFY_IN macro descriptions and Q.931 IE

Purpose

Indicates that a NOTIFY message was received.

Conn_id

An allocated call.

ACU_NOTIFY_IN protocol variants

The following table lists the variants under which each macro is supported for this primitive. In this table, x indicates both NT and TE sides.

Macro	4ESS	E10	NI2	DMS	ETSI	VN6	HKG	AUSTEL1	NTT	KOREA	TAIWAN	QSIG	DPNSS	T1607
Acu_notify_in_a_connected_name, Acu_notify_in_connected_name_size				x										
Acu_notify_in_a_ext_parms														
Acu_notify_in_ext_parms_nb														
Acu_notify_in_ext_parms_lgth														
Acu_notify_in_a_q931, Acu_notify_in_q931_size			x		x							x		x

ACU_NOTIFY_IN macro descriptions and Q.931 IE

Macro	Description	Q.931 IE
Acu_notify_in_a_connected_name, Acu_notify_in_connected_name_size	Pointer to (and size of) buffer containing calling name.	Connected name
Acu_notify_in_a_ext_parms	Pointer to buffer containing extended parameters.	Not used.
Acu_notify_in_ext_parms_nb	Number of parameters in buffer containing extended parameters.	Not used.
Acu_notify_in_ext_parms_lgth	Total length of buffer containing extended parameters.	Not used.
Acu_notify_in_a_q931, Acu_notify_in_q931_size	Pointer to (and size of) buffer containing raw data of incoming Q.931 message.	Does not map to an IE.

ACU_NOTIFY_RQ

This topic describes:

- ACU_NOTIFY_RQ protocol variants
- ACU_NOTIFY_RQ macro descriptions and Q.931 IE

Purpose

Requests that a NOTIFY message be sent.

Conn_id

An allocated call.

ACU_NOTIFY_RQ protocol variants

The following table lists the variants under which each macro is supported for this primitive. In this table, x indicates both NT and TE sides.

Macro	4ESS	E10	NI2	DMS	ETSI	VN6	HKG	AUSTEL1	NTT	KOREA	TAIWAN	QSIG	DPNSS	T1607
Acu_notify_rq_a_tsp_ie_list, Acu_notify_rq_tsp_ie_list_size					x							x		x

ACU_NOTIFY_RQ macro descriptions and Q.931 IE

Macro	Description	Q.931 IE
Acu_notify_rq_a_tsp_ie_list, Acu_notify_rq_tsp_ie_list_size	Pointer to (and size of) transparent IE buffer.	Does not map to an IE.

ACU_PROGRESS_IN

This topic describes:

- ACU_PROGRESS_IN protocol variants
- ACU_PROGRESS_IN macro descriptions and Q.931 IE

Purpose

Indicates outgoing call progress information (receipt of PROGRESS message).

Conn_id

An allocated call.

ACU_PROGRESS_IN protocol variants

The following table lists the variants under which each macro is supported for this primitive. In this table, x indicates both NT and TE sides.

Macro	4ESS	E10	NI2	DMS	ETSI	VN6	HKG	AUSTEL1	NTT	KOREA	TAIWAN	QSIG	DPNSS	T1607
Acu_progress_in_a_display, Acu_progress_in_display_size			x		x	x		x	x	x	x			x
Acu_progress_in_a_display_list			x		x	x		x	x	x	x			x
Acu_progress_in_a_ext_parms														
Acu_progress_in_a_facility, Acu_progress_in_facility_size					x							x		
Acu_progress_in_a_pcs_user, Acu_progress_in_pcs_user_size						x								
Acu_progress_in_a_q931, Acu_progress_in_q931_size	x	x	x		x	x	x	x	x				x	x

Macro	**4ESS**	**E10**	**NI2**	**DMS**	**ETSI**	**VN6**	**HKG**	**AUSTEL1**	**NTT**	**KOREA**	**TAIWAN**	**QSIG**	**DPNSS**	**T1607**
Acu_progress_in_call_ref_length			x											
Acu_progress_in_call_ref_value			x											
Acu_progress_in_cause	x	x	x	x	x	x	x	x	x	x	x	x		x
Acu_progress_in_data_chani													x	
Acu_progress_in_data_chani_excl													x	
Acu_progress_in_data_chani_nai													x	
Acu_progress_in_data_chani_nb														
Acu_progress_in_data_chani_tab[***i***]														
Acu_progress_in_data_chani_tab_nai[***i***]														
Acu_progress_in_ext_parms_lgth														
Acu_progress_in_ext_parms_nb														
Acu_progress_in_pcs_user_protocol						x								
Acu_progress_in_progress_descr_x[***i***]														
Acu_progress_in_progress_description	x	x	x	x	x	x	x	x	x	x	x	x		x
Acu_progress_in_progress_ind_nb			x	x	x	x				x	x	x		x
Acu_progress_in_progress_loc_x[***i***]														
Acu_progress_in_progress_location	x	x	x	x	x	x	x	x	x	x	x	x		x
Acu_progress_in_signal_val			x		x	x		x	x					
Acu_progress_in_x_display_nb			x		x	x		x	x	x	x			x
Acu_progress_in_x_display_size[***i***]			x		x	x		x	x	x	x			x

Macro	4ESS	E10	NI2	DMS	ETSI	VN6	HKG	AUSTEL1	NTT	KOREA	TAIWAN	QSIG	DPNSS	T1607
Acu_progress_in_x_display_total_size			x		x	x		x	x	x	x			x
Acu_progress_in_x_display_type[*i*]			x		x	x		x	x	x	x			x
Acu_progress_in_x_p_display[*i*]			x		x	x		x	x	x	x			x

ACU_PROGRESS_IN macro descriptions and Q.931 IE

Macro	Description	Q.931 IE
Acu_progress_in_a_display Acu_progress_in_display_size	Pointer to (and size of) buffer containing ISDN display information (optional).	Display
Acu_progress_in_a_display_list	Pointer to display structure.	Display
Acu_progress_in_a_ext_parms	Pointer to buffer containing extended parameters.	Not used.
Acu_progress_in_a_facility Acu_progress_in_facility_size	Pointer to (and size of) buffer containing ISDN facility information (optional).	Facility
Acu_progress_in_a_pcs_user Acu_progress_in_pcs_user_size	Pointer to (and size of) buffer containing ISDN PCS-to-user/user-to-PCS information. See PCS-user information elements for more information on this IE.	PCS-user
Acu_progress_in_a_q931 Acu_progress_in_q931_size	Pointer to (and size of) buffer containing raw data of incoming Q.931 message.	Does not map to an IE.
Acu_progress_in_call_ref_length	Length of call reference value.	Call reference
Acu_progress_in_call_ref_value	Call reference value and call reference flag.	Call reference
Acu_progress_in_cause	Cause value.	Cause

Macro	Description	Q.931 IE
Acu_progress_in_data_chani	Data channel to use (B1, B2, ... D).	Channel identification
Acu_progress_in_data_chani_excl	Channel IDs are preferred (OFF) or exclusive (ON).	Channel identification
Acu_progress_in_data_chani_nai	NAI.	Channel identification
Acu_progress_in_data_chani_nb	Number of channel IDs in the chani_list field. If no channel IDs, use 0.	Channel identification
Acu_progress_in_data_chani_tab[***i***]	Channel ID ***i***.	Channel identification
Acu_progress_in_data_chani_tab_nai[***i***]	NAI ***i***.	Channel identification
Acu_progress_in_ext_parms_lgth	Total length of buffer containing extended parameters.	Not used.
Acu_progress_in_ext_parms_nb	Number of parameters in buffer containing extended parameters.	Not used.
Acu_progress_in_pcs_user_protocol	Protocol discriminator for the pcs_user information element. See PCS-user information elements for more information on this IE.	PCS-user
Acu_progress_in_progress_descr_x[***i***]	Description for information element ***i***. See Progress description values for a list of valid values.	Progress indicator
Acu_progress_in_progress_description	Description for information element 0. See Progress description values for a list of valid values.	Progress indicator
Acu_progress_in_progress_ind_nb	Number of progress indication information elements.	Progress indicator
Acu_progress_in_progress_loc_x[***i***]	Location of information element ***i***. See Location values for a list of valid values.	Progress indicator

Macro	Description	Q.931 IE
Acu_progress_in_progress_location	Location of information element 0. See Location values for a list of valid values.	Progress indicator
Acu_progress_in_signal_val	Signal value. See Signal values for a list of valid values.	Not used.
Acu_progress_in_x_display_nb	Number of present occurrences.	Display
Acu_progress_in_x_display_size[***i***]	Size of occurrence ***i*** (optional).	Display
Acu_progress_in_x_display_total_size	Total size of the stored strings.	Display
Acu_progress_in_x_display_type[***i***]	Type of display ***i***.	Display
Acu_progress_in_x_p_display[***i***]	Pointer to occurrence ***i***.	Display

ACU_PROGRESS_RQ

This topic describes:

- ACU_PROGRESS_RQ protocol variants
- ACU_PROGRESS_RQ macro descriptions and Q.931 IE

Purpose

Indicates progress request.

Conn_id

An allocated call.

ACU_PROGRESS_RQ protocol variants

The following table lists the variants under which each macro is supported for this primitive. In this table, x indicates both NT and TE sides.

Macro	4ESS	E10	NI2	DMS	ETSI	VN6	HKG	AUSTEL1	NTT	KOREA	TAIWAN	QSIG	DPNSS	T1607
Acu_progress_rq_a_tsp_ie_list, Acu_progress_rq_tsp_ie_list_size	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Acu_progress_rq_cause	x	x	x	x	x	x	x	x	x	x	x	x		x

Macro	4ESS	E10	NI2	DMS	ETSI	VN6	HKG	AUSTEL1	NTT	KOREA	TAIWAN	QSIG	DPNSS	T1607
Acu_progress_rq_cause_location	x	x	x	x	x	x	x	x	x	x	x	x		x
Acu_progress_rq_progress_description	x	x	x	x	x	x	x	x	x	x	x	x		x
Acu_progress_rq_progress_location	x	x	x	x	x	x	x	x	x	x	x	x		x

ACU_PROGRESS_RQ macro descriptions and Q.931 IE

Macro	Description	Q.931 IE
Acu_progress_rq_a_tsp_ie_list, Acu_progress_rq_tsp_ie_list_size	Pointer to (and size of) transparent IE buffer.	Does not map to an IE.
Acu_progress_rq_cause	Cause value.	Cause
Acu_progress_rq_cause_location	Cause location.	Cause
Acu_progress_rq_progress_description	Description for information element 0. See Progress description values for a list of valid values.	Progress indicator
Acu_progress_rq_progress_location	Location of information element 0. See Location values for a list of valid values and default setting information.	Progress indicator

ACU_RESTART_IN

This topic describes:

- ACU_RESTART_IN protocol variants
- ACU_RESTART_IN macro descriptions and Q.931 IE

Purpose

Indicates that a RESTART ACKNOWLEDGE message has been received.

Conn_id

Unused.

ACU_RESTART_IN protocol variants

The following table lists the variants under which each macro is supported for this primitive. In this table, x indicates both NT and TE sides.

Macro	4ESS	E10	NI2	DMS	ETSI	VN6	HKG	AUSTEL1	NTT	KOREA	TAIWAN	QSIG	DPNSS	T1607
Acu_restart_pref (RESTART for the B channel)	x	x		x										
Acu_restart_pref (RESTART for the interface)		x												
Acu_restart_int_id	x	x	x	x	x	x	x	x	x	x	x	x		x
Acu_restart_b_chan	x	x	x	x	x	x	x	x	x	x	x	x		x
Acu_restart_q931_size														
Acu_restart_size														
Acu_restart_a_q931														

ACU_RESTART_IN macro descriptions and Q.931 IE

Macro	Description	Q.931 IE
Acu_restart_pref	RESTART for the: • B channel • Interface	Channel identification
Acu_restart_int_id	NAI.	Channel identification
Acu_restart_b_chan	B channel.	Channel identification
Acu_restart_q931_size	Reserved for future use.	N/A
Acu_restart_size	Size of structure.	N/A
Acu_restart_a_q931	Reserved for future use.	N/A

ACU_SERVICE_CO

This topic describes:

- ACU_SERVICE_CO protocol variants
- ACU_SERVICE_CO macro descriptions and Q.931 IE

Purpose

Indicates that a SERVICE ACKNOWLEDGE message has been received.

Conn_id

Unused.

ACU_SERVICE_CO protocol variants

The following table lists the variants under which each macro is supported for this primitive. In this table, x indicates both NT and TE sides.

Macro	4ESS	E10	NI2	DMS	ETSI	VN6	HKG	AUSTEL1	NTT	KOREA	TAIWAN	QSIG	DPNSS	T1607
Acu_service_pref (interface)		x												
Acu_service_pref (B channel)	x	x	x	x										x
Acu_service_int_id	x		x	x										x
Acu_service_b_chan	x		x	x										x
Acu_service_status	x		x	x										x
Acu_service_action_type	x		x	x										x
Acu_service_q931_size														
Acu_service_size														
Acu_service_a_q931														

ACU_SERVICE_CO macro descriptions and Q.931 IE

Macro	Description	Q.931 IE
Acu_service_pref	Preference. Valid values include: I_PREF_INTERFACE I_PREF_B_CHANNEL	Channel identification
Acu_service_int_id	Interface ID.	Channel identification
Acu_service_b_chan	B channel (if Acu_service_pref = I_PREF_B_CHANNEL).	Channel identification
Acu_service_status	New status. Valid values include: I_B_CHAN_IN_SERVICE I_B_CHAN_OUT_OF_SERVICE	Change status. This IE is not defined in Q.931.
Acu_service_action_type	Action to be performed (maintenance only).	Not applicable.
Acu_service_q931_size	Reserved for future use.	Not applicable.
Acu_service_size	Size of structure.	Not applicable.
Acu_service_a_q931	Reserved for future use.	Not applicable.

ACU_SERVICE_IN

This topic describes:

- ACU_SERVICE_IN protocol variants
- ACU_SERVICE_IN macro descriptions and Q.931 IE

Purpose

Indicates that a SERVICE message has been received.

Conn_id

Unused.

ACU_SERVICE_IN protocol variants

The following table lists the variants under which each macro is supported for this primitive. In this table, x indicates both NT and TE sides.

Macro	4ESS	E10	NI2	DMS	ETSI	VN6	HKG	AUSTEL1	NTT	KOREA	TAIWAN	QSIG	DPNSS	T1607
Acu_service_pref (interface)		x												
Acu_service_pref (B channel)	x	x	x	x										x
Acu_service_int_id	x		x	x										x
Acu_service_b_chan	x		x	x										x
Acu_service_status	x		x	x										x
Acu_service_action_type	x		x	x										x
Acu_service_q931_size														
Acu_service_size														
Acu_service_a_q931														

ACU_SERVICE_IN macro descriptions and Q.931 IE

Macro	Description	Q.931 IE
Acu_service_pref	Preference. Valid values include: I_PREF_INTERFACE I_PREF_B_CHANNEL	Channel identification
Acu_service_int_id	Interface ID.	Channel identification
Acu_service_b_chan	B channel (if Acu_service_pref = I_PREF_B_CHANNEL).	Channel identification
Acu_service_status	New status. Valid values include: I_B_CHAN_IN_SERVICE I_B_CHAN_OUT_OF_SERVICE	Change status. This IE is not defined in Q.931.
Acu_service_action_type	Action to be performed (maintenance only).	Not applicable.

Macro	Description	Q.931 IE
Acu_service_q931_size	Reserved for future use.	Not applicable.
Acu_service_size	Size of structure.	Not applicable.
Acu_service_a_q931	Reserved for future use.	Not applicable.

ACU_SERVICE_RQ

This topic describes:

- ACU_SERVICE_RQ protocol variants
- ACU_SERVICE_RQ macro descriptions and Q.931 IE

Purpose

Requests that a SERVICE message be sent.

Conn_id

Unused.

ACU_SERVICE_RQ protocol variants

The following table lists the variants under which each macro is supported for this primitive. In this table, x indicates both NT and TE sides.

Macro	4ESS	E10	NI2	DMS	ETSI	VN6	HKG	AUSTEL1	NTT	KOREA	TAIWAN	QSIG	DPNSS	T1607
Acu_service_pref (interface)		x												
Acu_service_pref (B channel)	x	x	x	x										x
Acu_service_int_id	x		x	x										x
Acu_service_b_chan	x		x	x										x
Acu_service_status	x		x	x										x
Acu_service_action_type	x		x	x										x
Acu_service_q931_size														

Macro	4ESS	E10	NI2	DMS	ETSI	VN6	HKG	AUSTEL1	NTT	KOREA	TAIWAN	QSIG	DPNSS	T1607
Acu_service_size														
Acu_service_a_q931														

ACU_SERVICE_RQ macro descriptions and Q.931 IE

Macro	Description	Q.931 IE
Acu_service_pref	Preference. Valid values include: I_PREF_INTERFACE I_PREF_B_CHANNEL	Channel identification
Acu_service_int_id	Interface ID.	Channel identification
Acu_service_b_chan	B channel (if Acu_service_pref = I_PREF_B_CHANNEL).	Channel identification
Acu_service_status	New status. Valid values include: I_B_CHAN_IN_SERVICE I_B_CHAN_OUT_OF_SERVICE	Change status. This IE is not defined in Q.931.
Acu_service_action_type	Action to be performed (maintenance only).	Not applicable.
Acu_service_q931_size	Reserved for future use.	Not applicable.
Acu_service_size	Size of structure.	Not applicable.
Acu_service_a_q931	Reserved for future use.	Not applicable.

ACU_SETUP_ACK_IN

This topic describes:

- ACU_SETUP_ACK_IN protocol variants
- ACU_SETUP_ACK_IN macro descriptions and Q.931 IE

Purpose

Acknowledges incoming call proceeding information.

Conn_id

An allocated call.

ACU_SETUP_ACK_IN protocol variants

The following table lists the variants under which each macro is supported for this primitive. In this table, x indicates both NT and TE sides.

Macro	**4ESS**	**E10**	**NI2**	**DMS**	**ETSI**	**VN6**	**HKG**	**AUSTEL1**	**NTT**	**KOREA**	**TAIWAN**	**QSIG**	**DPNSS**	**T1607**
Acu_setup_ack_in_a_display, Acu_setup_ack_in_display_size			x		x	x		x	x	x	x			x
Acu_setup_ack_in_a_display_list			x		x	x		x	x	x	x			x
Acu_setup_ack_in_a_ext_parms														
Acu_setup_ack_in_a_facility, Acu_setup_ack_in_facility_size					x					x	x			
Acu_setup_ack_in_a_pcs_user, Acu_setup_ack_in_pcs_user_size														
Acu_setup_ack_in_a_q931, Acu_setup_ack_in_q931_size	x	x	x	x	x	x	x	x	x	x	x	x		x
Acu_setup_ack_in_cause														
Acu_setup_ack_in_data_chani	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Acu_setup_ack_in_data_chani_excl														
Acu_setup_ack_in_data_chani_nai													x	
Acu_setup_ack_in_data_chani_nb													x	
Acu_setup_ack_in_data_chani_tab[***i***]														
Acu_setup_ack_in_data_chani_tab_nai[***i***]														
Acu_setup_ack_in_ext_parms_lgth														

Macro	4ESS	E10	NI2	DMS	ETSI	VN6	HKG	AUSTEL1	NTT	KOREA	TAIWAN	QSIG	DPNSS	T1607
Acu_setup_ack_in_ext_parms_nb														
Acu_setup_ack_in_progress_descr_x[***i***]														
Acu_setup_ack_in_progress_description			x	x	x	x				x	x			x
Acu_setup_ack_in_progress_ind_nb			x	x	x	x				x	x			x
Acu_setup_ack_in_progress_loc_x[***i***]														
Acu_setup_ack_in_progress_location			x	x	x	x				x	x			x
Acu_setup_ack_in_signal_val			x											
Acu_setup_ack_in_x_display_nb			x		x	x		x	x	x	x			x
Acu_setup_ack_in_x_display_size[***i***]			x		x	x		x	x	x	x			x
Acu_setup_ack_in_x_display_total_size			x		x	x		x	x	x	x			x
Acu_setup_ack_in_x_display_type[***i***]			x		x	x		x	x	x	x			x
Acu_setup_ack_in_x_p_display[***i***]			x		x	x		x	x	x	x			x

ACU_SETUP_ACK_IN macro descriptions and Q.931 IE

Macro	Description	Q.931 IE
Acu_setup_ack_in_a_display Acu_setup_ack_in_display_size	Pointer to (and size of) buffer containing ISDN display information (optional).	Display
Acu_setup_ack_in_a_display_list	Pointer to display structure.	Display
Acu_setup_ack_in_a_ext_parms	Pointer to buffer containing extended parameters.	Not used.
Acu_setup_ack_in_a_facility Acu_setup_ack_in_facility_size	Pointer to (and size of) buffer containing ISDN facility information (optional).	Facility

Macro	Description	Q.931 IE
Acu_setup_ack_in_a_pcs_user Acu_setup_ack_in_pcs_user_size	Pointer to (and size of) buffer containing ISDN PCS-to-user/user-to-PCS information. See PCS-user information elements for more information on this IE.	PCS-user
Acu_setup_ack_in_a_q931 Acu_setup_ack_in_q931_size	Pointer to (and size of) buffer containing raw data of incoming Q.931 message.	Does not map to an IE.
Acu_setup_ack_in_cause	Cause value.	Cause
Acu_setup_ack_in_data_chani	Data channel to use (B1, B2, ... D).	Channel identification
Acu_setup_ack_in_data_chani_excl	Channel IDs are preferred (OFF) or exclusive (ON).	Channel identification
Acu_setup_ack_in_data_chani_nai	NAI.	Channel identification
Acu_setup_ack_in_data_chani_nb	Number of channel IDs in the chani_list field. If no channel IDs, use 0.	Channel identification
Acu_setup_ack_in_data_chani_tab[***i***]	Channel ID ***i***.	Channel identification
Acu_setup_ack_in_data_chani_tab_nai[***i***]	NAI [***i***].	Channel identification
Acu_setup_ack_in_ext_parms_lgth	Total length of buffer containing extended parameters.	Not used.
Acu_setup_ack_in_ext_parms_nb	Number of parameters in buffer containing extended parameters.	Not used.
Acu_setup_ack_in_progress_descr_x[***i***]	Description for information element ***i***. See Progress description values for a list of valid values.	Progress indicator
Acu_setup_ack_in_progress_description	Description for information element 0. See Progress description values for a list of valid values.	Progress indicator

Macro	Description	Q.931 IE
Acu_setup_ack_in_progress_ind_nb	Number of progress indication information elements.	Progress indicator
Acu_setup_ack_in_progress_loc_x[*i*]	Location of information element *i*. See Location values for a list of valid values.	Progress indicator
Acu_setup_ack_in_progress_location	Location of information element 0. See Location values for a list of valid values.	Progress indicator
Acu_setup_ack_in_signal_val	Signal value. See Signal values for a list of valid values.	Signal value
Acu_setup_ack_in_x_display_nb	Number of present occurrences.	Display
Acu_setup_ack_in_x_display_size[*i*]	Size of occurrence *i* (optional).	Display
Acu_setup_ack_in_x_display_total_size	Total size of the stored strings.	Display
Acu_setup_ack_in_x_display_type[*i*]	Type of display *i*.	Display
Acu_setup_ack_in_x_p_display[*i*]	Pointer to occurrence *i*.	Display

ACU_SETUP_REPORT_IN

Signals an incoming call that has been rejected or ignored by the ACU because it was not compatible (address or service filtering).

Use the macros associated with ACU_CONN_IN to access information for this message.

Conn_id

A currently unused ID.

Note: The application must not answer this incoming call indication. This message is only for informational purposes.

ACU_TRANSFER_CO

This topic describes:

- ACU_TRANSFER_CO protocol variants
- ACU_TRANSFER_CO macro descriptions and Q.931 IE

Purpose

Indicates whether the attempted transfer was successful or not.

Conn_id

An allocated call.

ACU_TRANSFER_CO protocol variants

The following table lists the variants under which each macro is supported for this primitive. In this table, TE indicates the TE side only, and x indicates both NT and TE sides.

Macro	4ESS	E10	NI2	DMS	ETSI	VN6	HKG	AUSTEL1	NTT	KOREA	TAIWAN	QSIG	DPNSS	T1607
Acu_transfer_status			TE	TE	TE							x	x	
Acu_transfer_callid_present														
Acu_transfer_callid														
Acu_transfer_size			TE	TE	TE							x	x	

ACU_TRANSFER_CO macro descriptions and Q.931 IE

Macro	Description	Q.931 IE
Acu_transfer_status	Type of result. A value of 0 indicates success, and any other value indicates an error. If successful, an ACU_CLEAR_IN message may follow, depending on the variant.	Facility
Acu_transfer_callid_present	A value of 1 indicates that the ***callid*** field contains valid information.	Does not map to an IE.
Acu_transfer_callid	Information used to identify a call.	Does not map to an IE.
Acu_transfer_size	The size of the structure.	Does not map to an IE.

ACU_TRANSFER_RQ

This topic describes:

- ACU_TRANSFER_RQ protocol variants
- ACU_TRANSFER_RQ macro descriptions and Q.931 IE

Purpose

Initiates the transfer of two calls. The first call is the call for which this message is sent, and the second call is identified by ***callid***.

Conn_id

An allocated call.

ACU_TRANSFER_RQ protocol variants

The following table lists the variants under which each macro is supported for this primitive. In this table, TE indicates the TE side only, and x indicates both NT and TE sides.

Macro	4ESS	E10	NI2	DMS	ETSI	VN6	HKG	AUSTEL1	NTT	KOREA	TAIWAN	QSIG	DPNSS	T1607
Acu_transfer_status														
Acu_transfer_callid_present			TE	TE	TE							x	x	
Acu_transfer_callid			TE	TE	TE							x	x	
Acu_transfer_size			TE	TE	TE							x	x	

ACU_TRANSFER_RQ macro descriptions and Q.931 IE

Macro	Description	Q.931 IE
Acu_transfer_status	This field is ignored for this message.	Does not map to an IE.
Acu_transfer_callid_present	A value of 1 indicates that the ***callid*** field contains valid information. If this is true, the ***callid*** will represent the second call. If the ***callid*** field does not contain valid information, the second call will be implicitly chosen (if allowed by the protocol variant).	Does not map to an IE.
Acu_transfer_callid	Information used to identify a call.	Facility
Acu_transfer_size	The size of the structure.	Does not map to an IE.

10. Data types and constants in primitives

User rate values

The following table lists the valid user rate values for ACU_CONN_IN, ACU_CONN_RQ, ACU_CONN_RS, and ACU_SETUP_REPORT_IN messages.

These values are defined in *isdnacu.h*.

Rate	Description
ACU_RA_50	0.050 kbit/s CCITT V.6 and X.1
ACU_RA_75	0.075 kbit/s CCITT V.6 and X.1
ACU_RA_75_1200	0.075/1.2 kbit/s CCITT V.6 and x.1
ACU_RA_100	0.100 kbit/s CCITT V.6 and X.1
ACU_RA_110	0.110 kbit/s CCITT V.6 and X.1
ACU_RA_134	0.1345 kbit/s CCITT X.1
ACU_RA_150	0.150 kbit/s CCITT V.6 and X.1
ACU_RA_200	0.200 kbit/s CCITT V.6 and X.1
ACU_RA_300	0.300 kbit/s CCITT V.6 and X.1
ACU_RA_600	0.6 kbit/s CCITT V.6 and x.1
ACU_RA_1200	1.2 kbit/s CCITT V.6
ACU_RA_1200_75	1.2/0.075 kbit/s CCITT V.6 and x.1
ACU_RA_2400	2.4 kbit/s CCITT V.6 and X.1
ACU_RA_3600	3.6 kbit/s CCITT V.6
ACU_RA_4800	4.8 kbit/s CCITT V.6 and x.1
ACU_RA_7200	7.2 kbit/s CCITT V.6
ACU_RA_8000	8 kbit/s CCITT I.460
ACU_RA_9600	9.6 kbit/s CCITT V.6 and x.1
ACU_RA_12000	12 kbit/s CCITT V.6

Rate	Description
ACU_RA_14400	14.4 kbit/s CCITT V.6
ACU_RA_16000	16 kbit/s CCITT I.460
ACU_RA_19200	19.2 kbit/s CCITT V.6
ACU_RA_32000	32 kbit/s CCITT I.460
ACU_RA_38400	38.4 kbit/s extended V.14
ACU_RA_38400_NO_ETSI	38.4 kbit/s extended V.14 (VN3) NON ETSI
ACU_RA_48000	48 kbit/s CCITT V.6 and X.1
ACU_RA_56000	56 kbit/s CCITT V.6
ACU_RA_57600	57.6 kbit/s extended V.14 (VN3) NON ETSI NON CCITT
ACU_RA_64000	64 kbit/s CCITT I.460

Service values

The following table lists the valid service values for ACU_CONN_CO, ACU_CONN_IN, ACU_CONN_RQ, ACU_CONN_RS, ACU_FACILITY_IN, ACU_FACILITY_RQ, and ACU_SETUP_REPORT_IN messages.

These values are defined in *isdnval.h*.

Service	Description
FAX_SERVICE	G3 facsimile service.
FAX_4_SERVICE	G4 facsimile service.
DATA_SERVICE	Data service.
DATA_GCI_SERVICE	Data service on GCI bus.
DATA_56KBS_SERVICE	Data at 56 kbits/s service.
RAW_DATA_SERVICE	Raw data service on GCI bus: no MPH_B_INIT_RQ is generated (no B channel driver is associated).
DATA_TRANS_SERVICE	Transparent data service.
MODEM_SERVICE	Modem data service.

Service	Description
AUDIO_7_SERVICE	7 kHz audio service.
X25_SERVICE	X.25 circuit-mode service.
X25_PACKET_SERVICE	X.25 packet-mode service.
VOICE_SERVICE	Voice service.
VOICE_GCI_SERVICE	Voice service on GCI bus.
RAW_TELEPHONY_SERVICE	Raw telephony service on GCI bus: no MPH_B_INIT_RQ generated (no B channel driver is associated).
VOICE_TRANS_SERVICE	Transparent voice service.
V110_SERVICE	V.110 service.
V120_SERVICE	V.120 service.
VIDEO_SERVICE	Video service.
TDD_SERVICE	TDD service.
DATA_H0_SERVICE	Data using H0 (384 kbits/s) channel service (PRI only).
DATA_H11_SERVICE	Data using H11 (1536 kbits/s) channel service.
DATA_H12_SERVICE	Data using H12 (1536 kbits/s) channel service.
DATA_MULTIRATE_SERVICE	Data using multirate (2..30x64 kbits/s) channel service.
DATA_128KBS_SERVICE	Data using 2x64 kbits/s channel service (BRI only).
NO_B_CHAN_SERVICE	No B channel service (bearer-independent calls - QSIG only).
FAX_RELAY_SERVICE	G3 facsimile service (for use with physical relay process).
DATA_RELAY_SERVICE	Data service (for use with physical relay process).
DATA_56KBS_RELAY_SERVICE	Data at 56 kbit/s service (for use with physical relay process).

Service	Description
DATA_TRANS_RELAY_SERVICE	Data transparent service (for use with physical relay process).
MODEM_RELAY_SERVICE	Modem data service (for use with physical relay process).
X25_RELAY_SERVICE	X.25 circuit-mode service (for use with physical relay process).
VOICE_RELAY_SERVICE	Voice service (for use with physical relay process).
VOICE_GCI_RELAY_SERVICE	Voice service on GCI bus (for use with physical relay process).
NO_SERVICE	Undefined service.

Number type values

The following table lists the valid calling, called, and redirecting number type values for ACU_CONN_IN, ACU_SETUP_REPORT_IN, and ACU_ALERT_IN messages.

These values are defined in *isdnval.h*.

Type	Description
N_TYPE_UNKNOWN (Default for all variants except 4ESS, VN6)	Unknown.
N_TYPE_INTERNATIONAL	International number.
N_TYPE_NATIONAL (Default for 4ESS and VN6 variants)	National number.
N_TYPE_NET_SPF	Network specific number.
N_TYPE_SUBSCRIBER	Subscriber number.
N_TYPE_LOCAL	AT5 local (directory) number.
N_TYPE_ABBREVIATED	Abbreviated number.

Plan values

The following table lists the valid calling, called, and redirecting number plan values for ACU_ALERT_IN, ACU_CONN_IN, ACU_CONN_RQ, and ACU_SETUP_REPORT_IN messages.

These values are defined in *isdnval.h*.

Plan	Description
N_PLAN_UNKNOWN (Default for all variants except VN6)	Unknown.
N_PLAN_ISDN (Default for VN6 and AusTel variants)	ISDN/telephony numbering plan (CCITT E.164/E.163).
N_PLAN_TELEPHONE	Telephony - not in CEPT.
N_PLAN_DATA	Data numbering plan (CCITT x.121).
N_PLAN_TELEX	Telex numbering plan (CCITT f.69).
N_PLAN_NATIONAL	National standard numbering plan.
N_PLAN_PRIVATE	Private numbering plan.

Screening indicator values

The following table lists the valid calling and redirecting number screening indicator values for ACU_ALERT_IN, ACU_ALERT_RQ, ACU_CONN_IN, ACU_CONN_RQ, and ACU_SETUP_REPORT_IN messages.

These values are defined in *isdnval.h*.

Screening indicator	Description
N_SCREEN_USER_PROVIDED (Default)	User-provided, not screened.
N_SCREEN_USER_PASSED (Default for AusTel)	User-provided, verified, and passed.
N_SCREEN_USER_FAILED	User-provided, verified, and failed.
N_SCREEN_NETWORK_PROVIDED	Network provided.

Redirecting reason values

The following table lists the valid redirecting reason values for ACU_ALERT_IN, ACU_CONN_CO, ACU_CONN_IN, ACU_CONN_RQ, and ACU_SETUP_REPORT_IN messages.

These values are defined in *isdnval.h*.

Redirecting reason	Description
REASON_UNKNOWN (Default)	Unknown.
REASON_CALL_FORWARDING_BUSY	Call forwarding busy.

Redirecting reason	Description
REASON_CALL_FORWARDING_NO_REPLY	Call forwarding no reply.
REASON_CALL_FORWARDING_DTE_OUT	Call forwarding DTE out of order.
REASON_CALL_FORWARDING_BY_CALLED	Call forwarding by called equipment.
REASON_CALL_TRANSFER	Call transfer.
REASON_CALL_PICKUP	Call pickup.
REASON_CALL_FORWARDING_UNCONDITIONAL	Call forwarding unconditional.

Location values

The following table lists the valid location values for ACU_ALERT_IN, ACU_ALERT_RQ, ACU_CALL_PROC_IN, ACU_CALL_PROC_RQ, ACU_CLEAR_IN, ACU_CONN_IN, ACU_CONN_RQ, ACU_PROGRESS_IN, ACU_PROGRESS_RQ, and ACU_SETUP_REPORT_IN messages.

These values are defined in *isdnval.h*.

Location	Description
LOC_USER (Default if ***partner_equip*** argument passed to **isdnStartProtocol** is EQUIPMENT_NT)	User.
LOC_PR_LOCAL_USER	Private network serving the local user.
LOC_NETWORK (Default if ***partner_equip*** argument passed to **isdnStartProtocol** is EQUIPMENT_TE)	Location network (SWD).
LOC_TRANSIT_NET	Transit network.
LOC_PU_REMOTE_USER	Public network serving the remote user.
LOC_PR_REMOTE_USER	Private network serving the remote user.
BEYOND_IWP (Default for DMS variant)	Network beyond interworking point.

UUI protocol discriminator values

The following table lists the valid UUI protocol discriminator values for ACU_ALERT_RQ, ACU_CLEAR_CO, ACU_CLEAR_RQ, ACU_CLEAR_RS, ACU_CONN_CO, ACU_CONN_IN, ACU_CONN_RQ, ACU_CONN_RS, ACU_SETUP_REPORT_IN, ACU_USER_INFO_IN, and ACU_USER_INFO_RQ messages.

These values are defined in *isdnval.h*.

UUI protocol discriminator	Description
UUI_USER_SPF (default for HKG variant)	User specific coding.
UUI_OSI	OSI.
UUI_X244	Rec. X.244.
UUI_SYSTEM_MANAGEMENT	Reserved for system management convergence function.
UUI_IA5 (default)	IA5 characters (ASCII).
UUI_V120	Rec. V.120.
UUI_Q931	Rec. Q.931.

Signal values

The following table lists the valid signal values in ACU_ALERT_IN, ACU_ALERT_RQ, ACU_CALL_PROC_IN, ACU_CLEAR_IN, ACU_CLEAR_RQ, ACU_CONN_CO, and ACU_PROGRESS_IN messages.

These values are defined in *isdnval.h*.

Signal value	Description
ACUDIAL_ON	Dial tone on.
ACURING_BACK_ON	Ring back tone on.
ACUNET_CONGEST_ON	Network congestion tone on.
ACUBUSY_ON	Busy tone on.
ACUTONES_OFF	Tones off.
ACUALERTING_OFF	Alerting off.
ACUCREDIT_CARD_TONE_ON	Credit card tone on.
ACUOUT_OF_RANGE_TONE_ON	Out of range tone on.
ACUQUEUING_TONE_ON	Queuing tone on.

Signal value	Description
ACUAUDIBLE_TONE_ON	Audible tone on.
ACUVISUAL_ALERT_1_ON	Visual alert #1 on.
ACUVISUAL_ALERT_2_ON	Visual alert #2 on.
ACUSATCOM_CALL_NOTIFY	SATCOM call notify.

Network-provided clearing cause values

The following table lists the valid network-provided clearing cause values for ACU_CLEAR_CO, ACU_CLEAR_IN, and ACU_SUSPEND_CO messages.

These values depend upon the variant. A list of values, derived from various specifications, is defined in *decisdn.h.*

Network-provided clearing cause	Description
CAU_UNALL	Unallocated number.
CAU_NOR_STN	No route to transit network.
CAU_NOR_D	No route to destination.
CAU_CH_UNACC	Channel unacceptable.
CAU_AWARD	Call awarded and delivered in an established channel.
CAU_NORMAL_CC	Normal call clearing.
CAU_BUSY	User busy.
CAU_NO_USER_RES	No user responding.
CAU_NO_ANSW	No answer from user.
CAU_REJ	Call rejected.
CAU_NUM_CHANGED	Number changed.
CAU_NON_SEL	Non-selected user clearing.
CAU_DEST_OOF	Destination out of order.
CAU_INV	Invalid number format.

Network-provided clearing cause	**Description**
CAU_FAC_REJ	Facility rejected.
CAU_RES_TO_SE	Response to STATUS ENQUIRY.
CAU_NORMAL_UNSPEC	Normal, unspecified.
CAU_NO_CIRC_CHAN	No circuit or channel available.
CAU_NET_OOF	Network out of order.
CAU_TEMP_FAIL	Temporary failure.
CAU_CONG	Switching equipment congestion.
CAU_ACC_INF_DISC	Access information discarded.
CAU_NOT_AVAIL	Requested circuit or channel not available.
CAU_RES_UNAVAIL	Resource unavailable.
CAU_QOF_UNAVAIL	Quality of service unavailable.
CAU_FAC_NOT_SUB	Requested facility not subscribed.
CAU_BC_NOT_AUT	Bearer capability not authorized.
CAU_BC_NOT_AVAIL	Bearer capability not presently available.
CAU_SERV_NOTAVAIL	Service or option not available.
CAU_BC_NOT_IMP	Bearer capability not implemented.
CAU_CHT_NOT_IMP	Channel type not implemented.
CAU_FAC_NOT_IMP	Requested facility not implemented.
CAU_RESTR_BC	Only restricted digital bearer capability is available.
CAU_SERV_NOT_IMP	Service or option not implemented.
CAU_INV_CRV	Invalid call reference value.
CAU_CH_NOT_EX	Identified channel does not exist.
CAU_CALL_ID_NOT_EX	Call identity does not exist.

Network-provided clearing cause	**Description**
CAU_CALL_ID_IN_USE	Call identity in use.
CAU_NO_CALL_SUSP	No call suspended.
CAU_CALL_CLEARED	The call has been cleared.
CAU_INCOMP_DEST	Incompatible destination.
CAU_INV_TN	Invalid transit network selection.
CAU_INV_MSG	Invalid message.
CAU_MAND_IE_MISSING	Mandatory information element is missing.
CAU_MSGT_NOT_EX	Message type nonexistent or not implemented.
CAU_MSG_NOT_COMP	Message not compatible with call state or message type not existent.
CAU_IE_NOT_EX	Information element nonexistent or not implemented.
CAU_INV_IE_CONTENTS	Invalid information element contents.
CAU_MSG_NOT_COMP_CS	Message not compatible with call state.
CAU_RECOVERY	Recovery on timer expiration.
CAU_PROTO_ERR	Protocol error.
CAU_INTERW	Interworking.
CAU_PREEMPTION	Preemption (5ESS).
CAU_PREEMPTION_CRR	Preemption, circuit reserved for reuse (5ESS).
CAU_PREC_CALL_BLK	Precedence call blocked (5ESS).
CAU_BC_INCOMP_SERV	Bearer capability incompatible with service request (5ESS).
CAU_OUT_CALLS_BARRED	Outgoing calls barred (4ESS, 5ESS).
CAU_SERV_VIOLATED	Service operation violated (5ESS).
CAU_IN_CALLS_BARRED	Incoming calls barred (DMS, 4ESS, 5ESS).

Network-provided clearing cause	Description
CAU_DEST_ADD_MISSING	Destination address missing and direct call not subscribed (DMS).
CAU_RESTART	Call clearing due to restart procedure.
CAU_TIMER_300	Call clearing due to internal timer 300 expiration.
CAU_TIMER_302	Call clearing due to internal timer 302 expiration.
CAU_TIMER_303	Call clearing due to internal timer 303 expiration.
CAU_TIMER_304	Call clearing due to internal timer 304 expiration.
CAU_TIMER_308	Call clearing due to internal timer 308 expiration.
CAU_TIMER_309	Call clearing due to internal timer 309 expiration.
CAU_TIMER_310	Call clearing due to internal timer 310 expiration.
CAU_TIMER_316	Call clearing due to internal timer 316 expiration.
CAU_TIMER_317	Call clearing due to internal timer 317 expiration.
CAU_TIMER_318	Call clearing due to internal timer 318 expiration.
CAU_TIMER_319	Call clearing due to internal timer 319 expiration.
CAU_TIMER_399	Call clearing due to internal timer 399 expiration.

Clear code values

The following table lists the valid clear code values for ACU_CLEAR_CO and ACU_CLEAR_IN messages.

These values are defined in *isdnacu.h*.

Clear code	Description
ACURC_BUSY	Busy.
ACURC_NOPROCEED	No proceed indication (dial tone).
ACURC_NOANSWER	No answer.
ACURC_NOAUTOANSWER	No auto-answer tone detected.

Clear code	Description
ACURC_CONGESTED	GSTN or system is congested.
ACURC_INCOMING	Incoming call detected while trying to dial.
ACURC_NOLINE	Wrong addressing information or context already used.
ACURC_ERRNUM	Errored number.
ACURC_INHNUM	Inhibited number.
ACURC_2MNUM	Too many errored or inhibited numbers.
ACURC_HUNGUP	Remote has hung up or incident on connection.
ACURC_NETWORK_ERROR	Network has disconnected.
ACURC_TIMEOUT	Timeout error.
ACURC_BAD_SERVICE	Bad service ID in ACU_CONN_RQ/RS.
ACURC_INTERNAL	Other internal error.

Facility code values

The following table lists the valid facility code values for ACU_FACILITY_RQ, ACU_FACILITY_IN, and ACU_FACILITY_RQ messages.

These values are defined in *isdnacu.h*.

Facility code	Description
ACU_FAC_CALL_FORWARDING	Premise transfer.
ACU_FAC_CALL_DEFLECTION	Terminal transfer.
ACU_FAC_CHARGING	Charging.
ACU_FAC_CHARGING_TOTAL	Charging total cost.
ACU_FAC_HOLD	Call hold.
ACU_FAC_RETRIEVE	Call retrieve.
ACU_FAC_ALTERNATE	To and from facility.
ACU_FAC_TRF	Transfer.

Facility code	Description
ACU_FAC_THREE_PARTY	Conferencing.
ACU_FAC_MALICIOUS_CALL_ID	Malicious call identification.
ACU_FAC_RECALL	Recall facility (ETSI).
ACU_FAC_TRANSPARENT	Transparent facility for CC&NS.
ACU_FAC_CALL_APPEAR	Call appearance request.
ACU_FAC_FEATURE_ACT	Feature activation request.
ACU_FAC_DROP	Drop call.
ACU_FAC_FEATURE_IND	Feature indication (NT->TE only).
ACU_FAC_PCS_USER_ONLY	Send PCS-user information only, with no other facility request. Only for VN6, within ACU_FACILITY_RQ. See PCS-user information elements for more information on this IE.

Action code values

The following table lists the valid action code values for ACU_FACILITY_IN, ACU_SET_MODE_CO, ACU_SET_MODE_RQ, ACU_TEST_CO, and ACU_TEST_RQ messages.

These values are defined in *isdnacu.h*.

Action code	Description
ACU_RQ_ACTIVATE	Activate or register action.
ACU_RQ_CLEAR	Deactivate or clear action.
ACU_RQ_ENQUIRY	Inquiry action.
ACU_IN_TX	Transmission.
ACU_CO_ACK	Acknowledgement.
ACU_CO_REJ	Reject.

Mode code values

The following table lists the valid mode code values for the Acu_set_mode_code macro in ACU_SET_MODE_RQ messages.

These values are defined in *isdnacu.h*.

Mode code	Description
ACU_MODE_NA_OUT_OF_ORDER	Allow or forbid calls on NAI given by R_msg_nai.
ACU_MODE_CALL_OUT_OF_ORDER	Allow or forbid calls on NAI/CONN_ID given by R_msg_nai/R_msg_conn_id.
ACU_MODE_CHANI_OUT_OF_ORDER	Allow or forbid usage of the B channel given by Acu_set_mode_data_chani.
ACU_MODE_ALL_NA_OUT_OF_ORDER	Allow or forbid calls on all NAIs.

Error cause values

The following table lists the valid error code values for the Acu_err_in_cause macro in ACU_ERR_IN messages. These values are defined in *isdnacu.h*.

Cause code	Description
ACUER_PRIMITIVE_CODE	Unknown primitive code.
ACUER_PARAM_VAL	Invalid parameter.
ACUER_MANDATORY_PARAM_MISSING	Mandatory parameter missing.
ACUER_PARAM_TYPE	Incorrect parameter type.
ACUER_PARAM_LGTH	Incorrect parameter length.
ACUER_UNEXPECTED_PRIMITIVE	Unexpected primitive.
ACUER_PRIMITIVE_NOT_IMPLEMENTED	SSDU primitive not implemented.
ACUER_NO_TIMER_AVAILABLE	No more space to allocate new timer cells.
ACUER_CONGESTION	Resource congestion.

Progress description values

The following table lists the valid progress description values for ACU_ALERT_IN, ACU_ALERT_RQ, ACU_CALL_PROC_IN, ACU_CALL_PROC_RQ, ACU_CLEAR_IN, ACU_CONN_IN, ACU_CONN_RQ, ACU_PROGRESS_IN, ACU_PROGRESS_RQ, and ACU_SETUP_REPORT_IN messages.

These values are defined in *isdnval.h*.

Progress description	Description
NO_PROGRESS_DESCR	No progress information to be stored.
PROGRESS_DESCR_NON_END_TO_END_ISDN	Call is not end-to-end ISDN; further call progress information may be available.
PROGRESS_DESCR_CALL_RETURNED_ISDN	Call has returned to the ISDN.
PROGRESS_DESCR_IN_BAND_NOW	In-band information or appropriate pattern now available.

Charging type values

The following table lists the valid charging type values for ACU_CLEAR_CO, ACU_CLEAR_IN, ACU_CLEAR_RQ, ACU_CLEAR_RS, ACU_CONN_CO, ACU_CONN_RS, ACU_INFORMATION_IN, and ACU_INFORMATION_RQ messages.

These values are defined in *isdnacu.h*.

Charging type	Description
ACUCHARG_TYPE_FREE_OF_CHARGE	Free of charge (N/A in VN6/FRANCE).
ACUCHARG_TYPE_CHARGE_ON_DURATION	Charge on duration (N/A in VN6/FRANCE).
ACUCHARG_TYPE_CHARGE_ON_ANSWER	Charge on answer (N/A in VN6/FRANCE).
ACUCHARG_TYPE_CHARGE_INCREMENT	Charge increment (N/A in VN6/FRANCE).
ACUCHARG_TYPE_SUB_TOTAL_CHARGE	Sub-total charge.
ACUCHARG_TYPE_TOTAL_CHARGE	Total charge.

Endpoint ID values

These values are defined in *isdnacu.h*.

Endpoint ID	Description
ACU_ACTION_NO_ENDPOINT_ID	Do not send any endpoint ID (ACU_CONN_RQ).
ACU_ACTION_SND_USID	SPID was OK; send the endpoint ID information.

Endpoint ID	Description
ACU_ACTION_NO_SPID_NEGOTIATION	SPID negotiation is not supported.
ACU_ACTION_INVALID_SPID	The received SPID is invalid.
ACU_ACTION_PROMPT_INFO	Need a SPID from the terminal.
ACU_INTERPRETER_0	Terminal is selected if it is assigned the indicated TID and USID value.
ACU_INTERPRETER_1	Terminal is selected if it is not assigned the indicated TID but is assigned the indicated USID value.

11. isdndemo: Layer 3 call control

isdndemo overview

isdndemo uses the NMS ISDN Messaging API to place and receive calls on an ISDN trunk. It demonstrates:

- A digital trunk application that uses the layer 3 ISDN signaling interface.
- How to use the NMS ISDN Messaging API to perform call control on a primary rate ISDN trunk.
- An application that uses non-facility associated signaling (NFAS) group configurations.

The program supports multiple T1, E1, or BRI trunks or NFAS groups. You can specify the ISDN country and signaling system variant to use with a command line option.

The default behavior of the program is to accept inbound calls. Outbound calls can be generated by specifying the appropriate command line option.

Featured functions

isdnReleaseBuffer, **isdnSendMessage**, **isdnStartProtocol**

Requirements

- One or more digital trunk interface boards
- Natural Access
- *nocc.tcp* file

Usage

```
isdndemo [options]
```

where ***options*** are one or more of the following:

Option	Meaning	Defaults
-0	Disables the OAM service.	OAM service is enabled.
-?	Displays the Help screen and terminates.	N/A
-a	Specifies the network access identifier (NAI) of the trunk to use.	0
-b ***boardno***	Specifies the board number.	0
-d ***dialstring***	Specifies the digit string to dial.	1234567
-D	Specifies configurations with two or more D channels per program. Any options following -D on the command line apply to the next D channel. You can use this option several times.	One D channel is assumed.

Option	Meaning	Defaults
-g ***group_number***	Specifies the NFAS group number. Use this option instead of the -a option to run in NFAS configuration mode.	Non-NFAS configuration
-I ***time***	Specifies the time (in milliseconds) to wait after releasing an outbound call before placing the next one.	1000
-l	Displays log statistics. Statistics are logged to either: • *isdndemo_b**B**a**A**.log*, where ***B*** is the board number, and ***A*** is the NAI number, or • *isdndemo_b**B**g**G**.log*, where ***B*** is the board number and ***G*** is the NFAS group number. The log file used depends on whether the trunk is part of an NFAS group. If the trunk is part of an NFAS group, *isdndemo_b**B**g**G**.log* is used. If the trunk is not part of an NFAS group, *isdndemo_b**B**a**A**.log* is used.	No logging.
-L ***time***	Specifies statistics logging time in seconds.	60
-n	Runs ISDN protocol stack as NT, not as TE.	TE
-o ***outlines***	Specifies the number of lines on which to place outbound calls.	0
-O ***channel***	Specifies the first B channel on which outbound calls are placed. (For example, if -O is set to 10 and -o is set to 5, outbound calls are placed on channels 10 through 14.)	0
-p ***protocol***	Specifies the protocol variant to run. See Protocol option allowed values for a list of valid values.	T1 boards: 24 (AT&T 4ESS) E1 and BRI boards: 11 (EuroISDN)
-Q	Receives a buffer with a message containing raw Q.931 version of the message.	No extra buffer received.
-S	Runs the demonstration program in localhost.	inproc.
-t	Demonstrates the transparent IE sending feature. Builds a codeset 7 IE (user-specific IE) and attaches it to the next ACU_CLEAR_CO (Connect message).	No indicator.

Option	Meaning	Defaults
-T	Specifies the timer for the duration of an outbound call (in milliseconds), once the call has reached the CONNECTED state.	15000
-v ***hex_mask***	Controls which information will be printed by *isdndemo*. Possible values: 0x01: Program configuration 0x02: Call status messages 0x04: ACU messages 0x08: Call statistics to the screen	3
-V	Plays voice files in CONNECTED state.	Does not play voice files.

For example, the following command line specifies USA National ISDN 2 on the second board in a system and places calls on the first 10 channels:

```
isdndemo -b 1 -p 20 -o 10
```

When *isdndemo* is run with no command line options, the program defaults to board 0. (The board number for each board is specified in the system configuration file.) The program automatically determines the board type, and runs one of the following protocols:

Board type	Default protocol
T1	AT&T 4ESS
E1	EuroISDN
BRI	EuroISDN

When no command-line options are specified, the program assumes that it is not placing any outbound calls but is only accepting inbound calls.

Protocol option allowed values

Value	Protocol
3	France Telecom VN6
8	Northern Telecom DMS 100
9	INS-1500 NTT
11	EuroISDN
15	Australian Telecom 1

Value	Protocol
16	QSIG
17	Hong Kong Telephone
20	US National ISDN 2
23	AT&T 5ESS10
24	AT&T 4ESS
25	Korea
50	Taiwan
51	DPNSS
52	ANSI T1.607
88	Northern Telecom DMS250

Note: Use EuroISDN for the following countries: Austria, China, Denmark, Finland, Greece, Iceland, Ireland, Italy, Liechtenstein, Luxembourg, Netherlands, Norway, Portugal, Russia, Singapore, Spain, and Switzerland.

Using isdndemo

Perform the following steps to use *isdndemo*:

Step	Action
1	Set up the boards in a configuration that allows one trunk to talk with another. For example, install two boards and connect a cable between their trunk connectors. Do not link the boards together over the CT bus. For NFAS configurations, you can also install a single CG 6000C and connect a cable between two trunk connectors.
2	Start *oamsys* to configure and boot the boards.

Step	Action

3	Use *isdndemo* by either invoking an instance of the demonstration program with two active D channels or by invoking two separate instance of the demonstration program to call each other.
	To invoke one instance of the demonstration program with two active D channels (using the -D option), enter:
	`isdndemo -b 0 -n -p 20 -D -b 1 -o 10`
	If you specify a protocol variant for one D channel, the other uses the same variant unless otherwise specified. All other parameters (such as board, NT/TE, and call parameters) are unique for each D channel and must be configured separately.
	To invoke two separate instances of the demonstration program to call each other, invoke one instance for each trunk. Set up one to place calls, and the other to receive calls. Make sure to include the -n (lowercase) option to configure one instance as the NT side. For example:
	`isdndemo -b 0 -n -p 20`
	`isdndemo -b 1 -p 20 -o 10`

isdndemo compilation

isdndemo is supplied in executable form, as well as source code. To recompile *isdndemo*, enter one of the following commands:

Operating system	Directory	Command
Windows	*\nms\ctaccess\demos\isdndemo\*	nmake
UNIX	*/opt/nms/ctaccess/demos/isdndemo/*	make

For more information, see the *readme* file that came with the NMS ISDN software package.

isdndemo files

isdndemo consists of the following files:

File	Description
isdndemo.cpp	Program initialization and some utility functions.
isdndemo.h	Header file for the *isdndemo* program.

File	Description
BChannel.cpp *Call.cpp* *Context.cpp* *DChannel.cpp* *NAI.cpp* *Timer.cpp*	Implementation of appropriate classes.

Using NFAS with isdndemo

You can use *isdndemo* to demonstrate placing and receiving calls on trunks in non-facility associated signaling (NFAS) mode. To do this, use the following option to specify the NFAS group number:

```
-g group_number
```

Use this option in place of the -a option, which specifies the NAI number and is used only with non-NFAS configurations.

isdndemo reads the NFAS configuration for a given group from the OAM database. The group number given must correspond to the board where the D channel for the NFAS group is located.

isdndemo program structure and coding

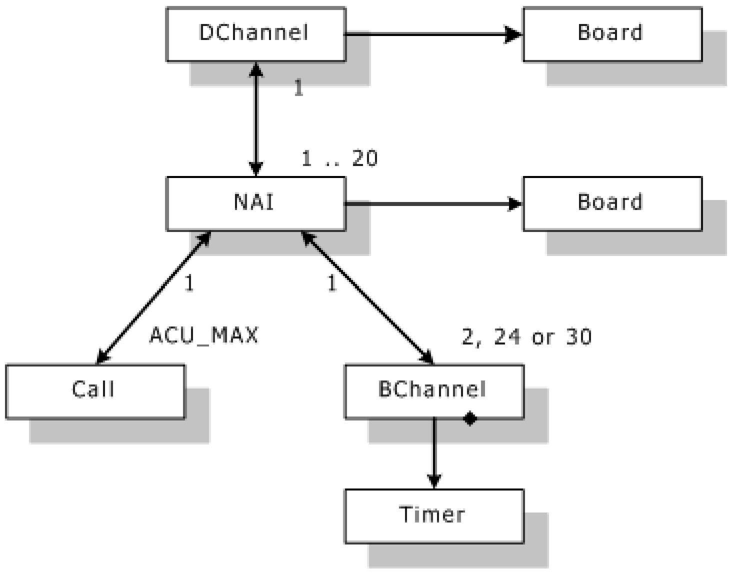
isdndemo is a single-threaded C++ application consisting of global functions and classes. The following table lists the global functions found in the *isdndemo.cpp* file:

Function	Description
cta_error_handler	Processes Natural Access errors.
dfprintf, **dvprintf**	Provides conditional program output.
error	Provides critical application error notification.
error_cta	Provides output of Natural Access codes in text form.
getACU, **getACUERR**	Converts some constants to text form.
main	Provides a program entry point, performs processing of program-wide command line arguments, and initializes Natural Access and program classes.
readConfig	Creates a program object hierarchy based on command line arguments.

The global classes used with *isdndemo* are found in several files. The following table lists each class along with its associated file and description:

Class	File	Description
BChannel	*BChannel.cpp*	Specifies a context class that represents the ISDN B channel.
Board	*Board.cpp*	Provides access to information about NMS boards.
Call	*Call.cpp*	Specifies an object that represents a single NMS ISDN call.
Context	*Context.cpp*	Implements a single-threaded program model and a generic mechanism for processing Natural Access events.
DChannel	*DChannel.cpp*	Specifies a context class that represents the ISDN D channel.
NAI	*NAI.cpp*	Specifies an object that represents board trunks (or NAI in NFAS terminology) associated with BChannel and Call objects.
Timer	*Timer.cpp*	Implements a synchronous timer based on the asynchronous ADI timer.

The following illustration shows how *isdndemo* functions and classes are related:



isdndemo program initialization

The *isdndemo* program initializes in the following manner:

Step	Action
1	Processes program-wide command line arguments (such as -h and -v).
2	Sets handler for Natural Access errors to **ctaSetErrorHandler.**
3	Initializes Natural Access using **ctaInitialize.**
4	Initializes any classes that require it.
5	Creates DChannel and other objects based on command line arguments.
6	Starts main event processing loop.

All event processing and other activities are performed by the interaction of created objects.

isdndemo classes

This topic describes the following major global classes used in the *isdndemo* program:

- Context
- DChannel
- NAI
- Call
- BContext
- Timer

Context

This abstract class provides a generic mechanism for event processing. Protected constructor Context::Context() creates a unique Natural Access context for each Context class object and its children. Every context is then bound to a single event processing queue (Context::qid).

Each child class overrides two functions of Context to implement some processing logic:

- **processEvent.** Process a single event that is passed as an argument.
- **start.** Invoked for each Context object just before entering the main processing loop in the Context::eventLoop function.

Each Context object has a unique index (Context::index) that enables the main processing loop to direct an event to the correct context.

DChannel

The DChannel is a key class for all of *isdndemo*. One object of this class is created for each D channel the program uses. During object construction, one NAI object is created for each trunk that a given D channel supports. For a non-NFAS configuration, only one NAI is created. Thus, a single call of the DChannel::DChannel constructor creates a tree of related objects.

DChannel objects perform the following functions in *isdndemo*:

- Process incoming ISDN call control messages with the **processEvent** function. Messages are passed to the appropriate Call object for processing.
- Send ISDN call control messages to the board with **sendIsdnMessage**. DChannel never creates ISDN messages by itself. This function is used only by Call objects.
- Print ACU messages in a form readable to users.
- Print program operation statistics to the screen and to the log file.

NAI

NAIs tie together DChannel, Call, and BChannel objects. An NAI object supports a list of Call objects, which presents calls associated with a given NAI. A Call object can be obtained by its connection ID (from **getCall**). The free call with the lowest connection ID can be obtained by invoking the **getFreeCall** function.

NAI objects also support a list of BChannels associated with a given NAI. These objects can be obtained by their B channel number by invoking **getBChannel**.

All Call and BChannel objects are created during the creation of an NAI object.

Call

Call objects implement a reduced version of the NMS ISDN messaging API state machine. A Call object can be in one of the following four states:

State	Description
ST_NULL	Free start. No real call exists.
ST_AWAITING_CONNECT	Objects wait for the connection of the call (ACU_CONN_CO).
ST_ACTIVE	Call is in the connected state.
ST_AWAITING_CLEARANCE	Objects wait for the call to be cleared (ACU_CLEAR_CO).

The following three functions affect the Call object state:

Function	Description
processIsdnMessage	Processes incoming ACU messages and sends the ACU messages in response. A DChannel object associated with the Call object calls this function.
makeCall	Initiates a new outbound call. BChannel calls this function.
hangUp	Hangs up a call in the connected state. BChannel calls this function.

A Call object can have a BChannel object associated with it. In this case, it uses some of BChannel's functions to indicate a Call change of state to a BChannel object.

BContext

BContext objects provide voice playing and ADI timers associated with a physical B channel. This object can have a single Call object associated with it.

A BContext object can be in one of the following three states:

State	Description
ST_FREE	Not associated with any Call object.
ST_USED	Associated with a Call object.
ST_CALL	Associated with a Call object in the connected state.

Two different modes for BContext objects determine its behavior:

Mode	Description
Outbound	Initiates an outbound ISDN call upon entering the ST_FREE state. It also tries to hang up an existing call after a time interval when in the ST_CALL state.
Inbound	Initiates nothing. This is passive mode.

A Call object can use the following BChannel functions to indicate a change of state:

Function	Description
get	Associate BChannel with a Call object.
free	Disassociate BChannel with a Call object.
startCall	Play voice. Call entered connected state.
stopCall	Stop playing voice. Call left connected state.

Timer

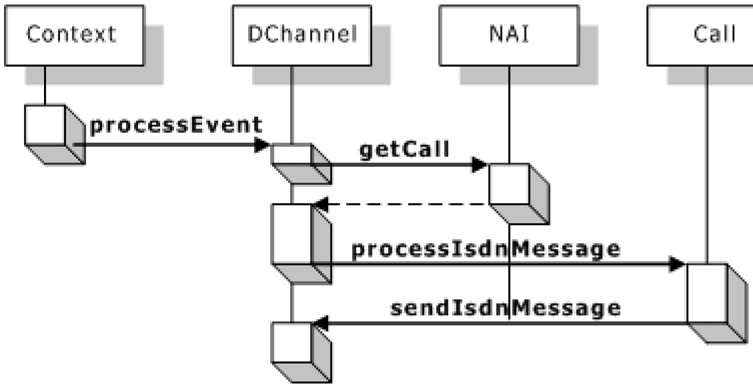
BContext uses a Timer object to wait for a specific time interval. Timer is a utility class that implements a synchronous timer based on the asynchronous ADI timer. It has three functions:

Function	Description
start	Starts the timer for a given time interval, or restarts a running timer.
stop	Stops running the timer, or does nothing if the timer is not active.
event	Passes an ADI timer event to the Timer object. **event** returns true if the Timer object has expired. Otherwise, it returns false.

Processing ISDN call control events

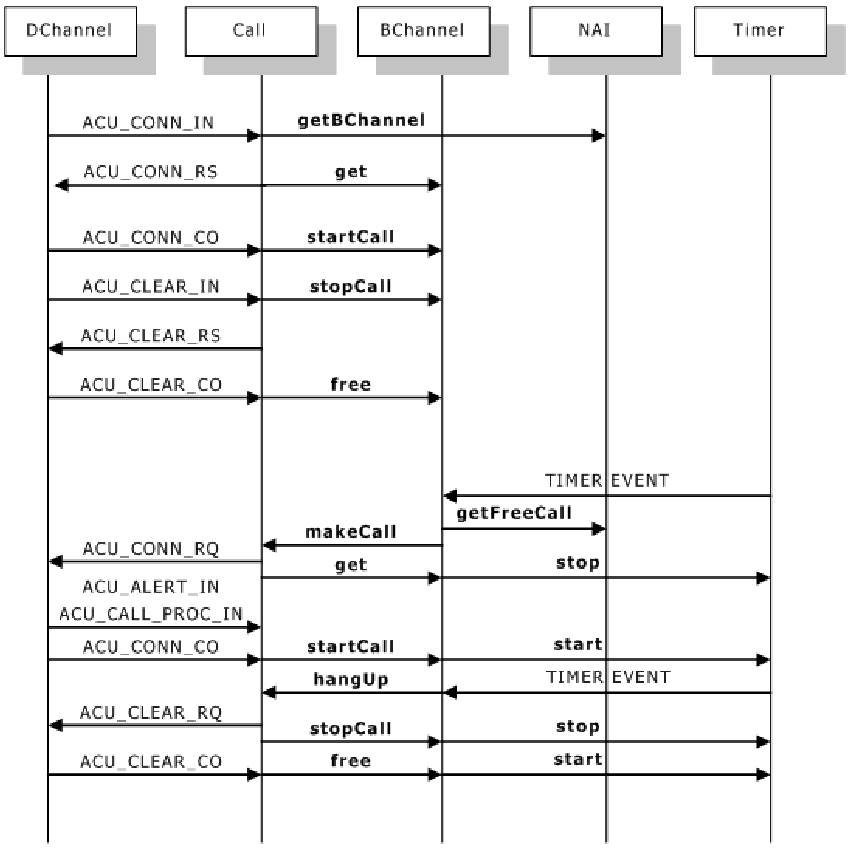
An incoming ISDN event is received by the **processEvent** function of a DChannel object. NAI objects are selected based on NAI number. The **getCall** function is then called to get a Call object for the connection ID. After a Call object is obtained, the message is passed for processing by the **processIsdnMessage** function.

Each Call has one NAI object associated with it, and each NAI object has a DChannel object associated with it. When a Call object needs to send an ISDN message to the stack, it uses the **sendIsdnMessage** function of a DChannel object. This process is shown in the following illustration:



Placing and receiving calls

The following illustration shows the object interaction while placing and receiving calls:

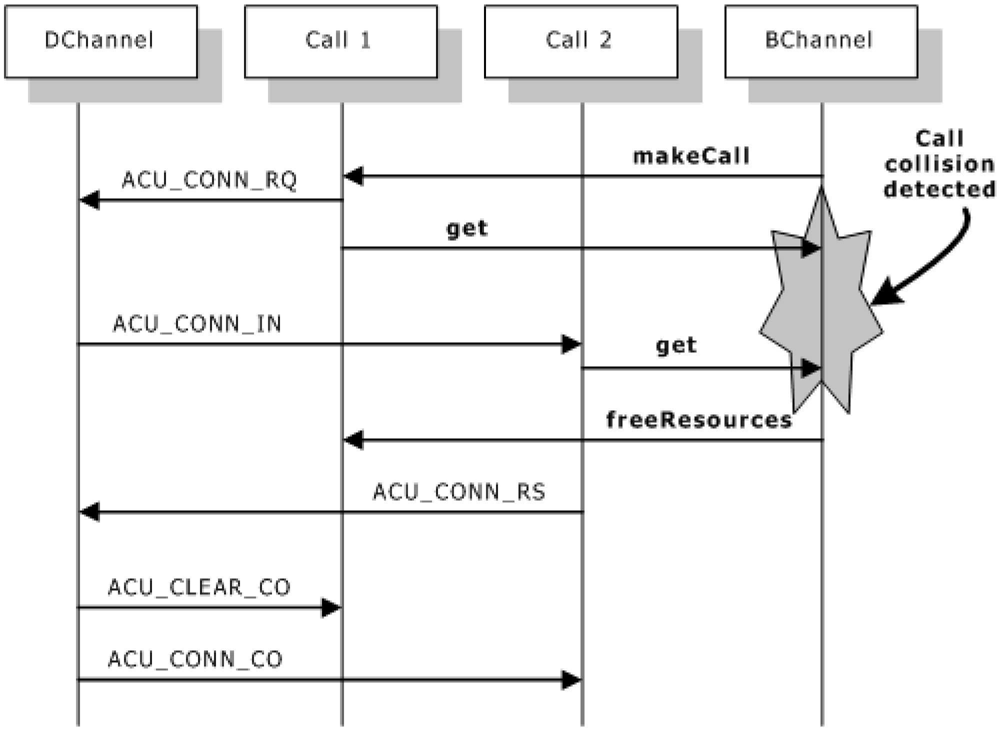


At the initial stage of the call, the Call object associates itself with the BChannel object. The B channel number comes when the startCall function is called. When a call leaves the connected state, the stopCall function is called. Then the ISDN call ceases to exist. The Call object invokes the free function to free the BChannel object. Once free, that BChannel object may be used by another call.

When a BChannel object is in outbound mode, it initiates a new call by getting a free Call object (**getFreeCall**) and invoking **makeCall**. When the call reaches the connected state, a Timer object is started. After the timer expires, BChannel invokes **hangUp** to terminate the call.

In a call collision situation, there are two Call objects (one inbound, one outbound) that have invoked **get** without invoking **free**. In this situation, the **freeResources** function is invoked for the first Call object to disassociate it with the BChannel object.

The following illustration shows call collision processing:



12. lapddemo: LAPD demonstration program

lapddemo overview

lapddemo establishes a data link and sends and receives Q.931 messages. It demonstrates how:

- An application accessing the stack at the LAPD interface can establish a data link on an ISDN trunk.
- To build raw Q.931 messages and send them to the trunk.
- To access the stack if duplicate NAI values are configured.

Messages built and sent with this demonstration program are based on standard Q.931 specifications (Blue Book) and may not be accepted or allowed in some switch variants.

Featured functions

isdnReleaseBuffer, **isdnSendMessage**, **isdnStartProtocol**, **isdnStopProtocol**

Requirements

- One or more digital trunk interface boards
- Natural Access
- *nocc.tcp* file

Usage

```
lapddemo [options]
```

where ***options*** are one or more of the following:

Option	Meaning	Defaults
-b ***boardno***	Number of the board to use. (Board numbers are specified in the board keyword file.)	0
-n	*lapddemo* configured for the network terminator (NT) side.	Not specified. *lapddemo* is configured for terminal equipment (TE).
-a ***nai***	Network access identifier (NAI).	None.
-g ***nfas_group***	NFAS group number, for duplicate NAI values.	None.
-v ***verboselevel***	Verbosity level. Valid values are: 0: Show no messages 1: Show SETUP messages only 2: Show all ISDN messages	2

Option	Meaning	Defaults
-f ***filename***	Name of the configuration file to be read. This file describes the behavior of the program in terms of what Q.931 messages are sent or received. Specifications for outgoing calls are not given in the default configuration file. You must make additions to this file in order for the demonstration program to achieve the CONNECTED state for outbound calls.	*lapddemo.cfg*

Functional overview

When *lapddemo* is run, it performs the following tasks:

Task	Description
1	It parses command-line arguments and makes sure all arguments are valid and do not conflict.
2	It initializes Natural Access.
3	It starts the ISDN protocol stack on the board specified on the command-line, either as terminal equipment (TE) or as network equipment (NT), with the specified network operator variant and country variant.
4	It sends a SABME message on the trunk to establish the data link.
5	If you type **S**, the program sends a SETUP message to the trunk. Otherwise, it waits for messages from the line.
6	If a Q.931 message is received, the program decodes information from the message and responds with an appropriate message, specified in the configuration file. For example, after receiving a SETUP message, it may send an ALERTING message.

Using lapddemo

Perform the following steps to run *lapddemo*:

Step	Action
1	Set up the board keyword file to describe the board and software configuration. For more information, see the *NMS ISDN Installation Manual* and the *NMS OAM System User's Manual*.
2	Run *oamsys* to initialize your hardware and make your configuration file changes effective.
3	If necessary, modify the demonstration configuration file *lapddemo.cfg*. For more information, see lapddemo configuration file.
4	Start the demonstration by entering: `lapddemo [options]` where ***options*** are one or more of the command line options described in the lapddemo overview. *lapddemo* sends a SABME message on the trunk to establish the data link.
5	Press **S**. The program sends a SETUP message with a called number value, a calling number value, a B channel number, and an encoding flag. These values are specified in *lapddemo.cfg*. The default values are: Called number value: 12345 Calling number value: 678 B channel number: 12 Encoding flag: A law

If the data link is established, the program exchanges Q.931 messages with the remote party, as described in the configuration file. By default, the following exchanges are made:

If...	lapddemo...
SETUP is received	Sends an ALERTING message, and then a CONNECT message.
CONNECT ACK is received	Starts timer T.
Timer T expires	Sends a DISCONNECT message.
DISCONNECT is received	Sends a RELEASE message.

If...	**lapddemo...**
RELEASE is received	Sends a RELEASE COMP message.

lapddemo compilation

lapddemo is supplied in executable form, as well as source code.

If you need to recompile *lapddemo*, enter one of the following commands:

Operating system	**Directory**	**Command**
Windows	*\nms\ctaccess\demos\lapddemo\*	nmake
UNIX	*/opt/nms/ctaccess/demos/lapddemo/*	make

For more information, see the *readme* file that came with the NMS ISDN software package.

lapddemo files

lapddemo consists of the following files:

File	**Description**
lapddemo.c	The main application program code.
isdnlib.c	A library of functions used by the main program to build correct Q.931 messages.
lapdlib.h	The prototypes for the functions contained in *isdnlib.c* plus the definitions of the structures used by those functions.
decisdn.h	The definitions used to build Q.931 information elements and messages. Can be found in:
	Operating system — **Directory**
	Windows — *\nms\ctaccess\demos\isdndemo\*
	UNIX — */opt/nms/ctaccess/demos/isdndemo/*
lapddemo.cfg	An example of a configuration file for this demonstration program.

lapddemo configuration file

lapddemo reads a configuration file to determine what Q.931 messages to send and how to respond to incoming messages. A sample configuration file, *lapddemo.cfg*, is supplied with the demonstration. Specify the configuration file to use with the -f ***filename*** option.

Note: Specifications for outgoing calls are not given in the default configuration file. You must make additions to this file for the demonstration program to achieve the CONNECTED state for outbound calls.

Two types of information appear in the file:

- Parameters and associated values
- Message exchange rules

Parameters and associated values

Each parameter appears on a separate line, followed by a space and then the assigned value. The following parameters can be set in the configuration file:

Parameter	Meaning	Allowed values	Default
!D	Called number (applies to the SETUP message).	One or more digits	12345
!A	Calling number (applies to the SETUP message).	One or more digits	678
!B	B channel (applies to the SETUP message).	Decimal value	12
!E	Encoding (applies to the SETUP message).	2: mu-law 3: A-law	3
!C	Cause (applies to the DISCONNECT message).	Decimal value	16

Message exchange rules

Each rule specifies a command for *lapddemo* to send or a task for *lapddemo* to perform when certain events happen (such as when a message is received or a timer expires). Each rule appears on a separate line. A rule is formatted as follows:

cause direction command

where ***cause*** is a message or event code from the following table:

Code	Message or event
S	SETUP
A	ALERTING

Code	Message or event
K	CALL PROCEEDING
P	PROGRESS
C	CONNECT
c	CONNECT ACK
D	DISCONNECT
R	RELEASE
r	RELEASE COMP
T	TIMER T (2 s)
t	TIMER T (1 s)

direction specifies whether *lapddemo* performs the response when a message is received, a message is sent, or a timer expires:

Indicator	Meaning
i	Incoming message or an expiring timer. *lapddemo* takes the action if message ***cause*** is received or timer ***cause*** expires.
o	Outgoing message. *lapddemo* takes the action when it sends message ***cause***.

command is the message for *lapddemo* to send or the timer for *lapddemo* to set when ***cause*** happens according to ***direction***. ***command*** can be any value from the Code column on the previous table.

The following syntax rules apply in the file:

- Text between a number sign (#) and the end of a line is ignored by the application.
- The exclamation point (!) indicates that the following symbol is a parameter.
- Any line not beginning with a number sign (#) or an exclamation point (!) is a message exchange rule, formatted as previously described.
- ***cause*** and ***command*** values are case sensitive.

The following code sample shows the *lapddemo.cfg* file included with *lapddemo*:

```
# File: lapddemo.cfg
# Configuration file for lapddemo

###################################################################
# Parameters
#
# !D = Called number                        (Applies to the SETUP msg)
# !A = Calling number                       (Applies to the SETUP msg)
# !B = B channel (decimal value)            (Applies to the SETUP msg)
# !E = Encoding ( 2 = mu Law, 3 = A Law)    (Applies to the SETUP msg)
# !C = Cause (decimal value)                (Applies to the DISCONNECT msg)
#
###################################################################
# Parameter      value

    !D         12345
    !A         678
    !B         12
    !C         16
    !E         3
###################################################################
# Command list
#
# S = SETUP
# A = ALERTING
# K = CALL PROCEEDING
# P = PROGRESS
# C = CONNECT
# c = CONNECT ACK
# D = DISCONNECT
# R = RELEASE
# r = RELEASE COMP
# T = TIMER T (2 s)
# t = TIMER T (1 s)
#
###################################################################

# Cause   Dir   Command

    S      i      K      # Send CALL PROCEEDING after receiving a SETUP
    K      o      C      # Send CONNECT after sending CALL PROCEEDING
    C      o      T      # Start timer T after receiving a CONNECT
    T      i      D      # Send a DISCONNECT when timer T expires
    D      i      R      # Send a RELEASE after receiving DISCONNECT
    R      i      r      # Send a RELEASE COMP after receiving RELEASE
```

lapddemo structure and coding features

The **main** function (in *lapddemo.c*) accepts and parses the command line options, reads the configuration file, and establishes the data link. At this point, the program can either send a SETUP message to the trunk (if you type **S**) or wait for messages from the line.

Note: The SETUP message sent by *lapddemo* contains the following information elements only: bearer capability, channel ID, calling number, and called number. Some switch variants may require additional information elements.

If a Q.931 message is received, the program decodes information from the message, such as the protocol discriminator (for example, Q.931 CC), the call reference value, and the message type (for example, SETUP). It responds with an appropriate message, specified in the configuration file. For example, after receiving a SETUP message, it sends an ALERTING message.

The state machine takes into consideration only the state of the data link. There are only two states: idle and data link established. The program has no knowledge of the state of a call, so it can be configured to break Q.931 message protocol rules (for example, by sending a SETUP message after receiving a SETUP message). It is the user's responsibility to configure the program properly.

Q.931 messages are built using functions from *isdnlib.c*, for example, **BuildSetup** and **BuildAlerting**. *lapdlib.h* contains the prototypes for these functions and the definitions of the structures used by the functions. *decisdn.h* contains the defines used to build Q.931 information elements and messages.

13. dectrace: Stack traffic analysis tool

dectrace overview

dectrace decodes and displays messages sent or received by the NMS ISDN protocol stack that were previously captured in a log file by the board monitoring utility (*oammon*).

dectrace decodes:

- Q.931/Q.921 messages sent or received by the stack when the stack is running in any mode.
- ACU primitives exchanged by the application and the stack when the stack is running in ACU stack mode.
- **Note:** By default, the log file does not contain stack message data. To provide data in this file that *dectrace* can process, the *agtrace, itrace,* or both utilities must be used. For details, see Creating a log file for dectrace.

Usage

```
dectrace [-f srcfile] [-d tmask] [-b brdno] [-a nai] [-g grp] [-c clrf]
```

where:

Option	Meaning	Default
-a ***nai***	Decodes only the messages on the specified NAI.	All NAIs.
-b ***brdno***	Decodes only the messages from or to board ***brdno***.	All boards.
-c ***clrf***	Decodes only the messages with call reference ***clrf***.	All call references.
-d ***tmask***	Provides a decoding mask for Q.931 decoding. See tmask values.	TIME STAMP + MESSAGE + CALL_REF + IE + IE CONT + ISDN_MSG
-f ***srcfile***	Specifies the file to be decoded.	*agerror.log*
-g ***grp***	Decodes only the messages for the specified NFAS group.	All NFAS groups.

The following table lists the valid ***tmask*** values:

Mask	Mnemonic	Description
8000	BUFFER	Prints the whole hexadecimal buffer.
2000	PROTOCOL DISC	Decodes the protocol discriminator.

Mask	Mnemonic	Description
1000	CALL REF	Decodes the call reference.
0800	MESSAGE	Decodes the message type.
0400	INFO ELEM	Decodes the information element ID.
0200	INFO ELEM CONT	Decodes the data contents of the IE.
0100	DATA LINK	Decodes the commands or events at the DL interface.
0080	TIME STAMP	Includes a time stamp before the messages.
0008	ISDN_MSG	Decodes the ISDN messages.
0004	ACU_MSG	Decodes the ACU messages.
0002	TRACE_PH	Decodes PH primitives.

Using dectrace

Perform the following steps to run *dectrace*:

Step	Action
1	Run *agtrace* or *itrace* to instruct the AG driver to send NMS ISDN protocol stack trace messages to *oammon*.
2	For CG boards, run *oammon* to configure your boards as described in the board keyword file and to enable error logging. For AG boards, this is done automatically.
3	Run the NMS ISDN protocol stack.
4	Run *dectrace.* For CG boards, use the *oammon* -f command to specify the log file on the command line. For AG boards, the default name of this file is *agpierror.log*.

dectrace compilation

dectrace is supplied in executable form, as well as source code. To recompile *dectrace*, enter one of the following commands:

Operating system	**Directory**	**Command**
Windows	*\nms\ctaccess\demos\dectrace\*	nmake
UNIX	*/opt/nms/ctaccess/demos/dectrace/*	make

For more information, see the *readme* file that came with the NMS ISDN software package.

dectrace files

dectrace consists of the following files:

- *dectrace.c*
- *dectrace.h*
- *decisdn.c*
- *decisdn.h*

Creating a log file for dectrace

dectrace reads the log file created by *oammon* (*agpierror.log*). By default, this file does not contain NMS ISDN protocol stack messages. To enable *oammon* to log messages from the stack, use either (or both) of the following utilities:

Utility	**Description**
agtrace	*agtrace* takes, as parameters: • A bit mask • The board number (default 0) • The channel number (default 0) To trace the NMS ISDN protocol stack, set the 0x80000 bit. To enable *oammon* to display and log the trace, use the -f option.

Utility	Description

itrace	The ISDN stack includes many entities (for example, layer 1 entities, layer 2 entities, layer 3 entities, management entities, timer entities) that send debug messages to *oammon*. *itrace* limits the number of ISDN entities that send debug messages to *oammon* (by default all the entities are enabled).
	itrace takes as arguments a flag (on or off), board number, NAI number, and a list of entities (from the file *isdntype.h*). An exclamation point (!) before the entity list enables the sending of buffers along with the messages.
	For example, to prepare an *agpierror.log* for board 0, NAI 1, containing Q.931 buffers, enter the following information:
	`agtrace 801000 0`
	`itrace off 0 1 *`
	`itrace on 0 1 !Dd`
	and run the application.
	To prepare an *agpierror.log* for board 0, NAI 1, containing only ACU messages, enter the following information:
	`agtrace 801000 0`
	`itrace off 0 1*`
	`itrace on 0 1 AC`
	and run the application.

The log file is located in the following directory:

Operating system	Directory
Windows	*nms\oam\log*
UNIX	*/var/opt/nms/ag*

Tracing is not recommended when performing heavy load tests. The high number of accesses to the log file caused by tracing can dramatically decrease the coprocessor's capability to handle the messages.

Embedded errors in *agpierror.log* may not be handled correctly by *dectrace* during load tests, or if you do not follow the indications about limiting the number of messages or buffers sent to the board monitoring utility.

When reporting a potential problem in the NMS ISDN stack, include the entire *agpierror.log* file. Do not use *itrace* in this case, and do not include the output of *dectrace*.

Messages decoded by dectrace

dectrace decodes ISDN layer 3 messages in accordance with Q.931/Q.932 specifications, except where noted.

Q.931/Q.932 message types

dectrace decodes all the messages types defined in the Q.931/Q.932 specifications. The following table lists the messages types:

Message	Hexadecimal value	Notes
Escape to nationally specific message type:		
MSG_ESCAPE	0x00	Q.931
Call establishing messages:		
MSG_ALERTING	0x01	Q.931
MSG_CALL_PROC	0x02	Q.931
MSG_PROGRESS	0x03	Q.931
MSG_SETUP	0x05	Q.931
MSG_CONNECT	0x07	Q.931
MSG_SETUP_ACK	0x0D	Q.931
MSG_CONNECT_ACK	0x0F	Q.931
Call information phase messages:		
MSG_USER_INFO	0x20	Q.931
MSG_SUSPEND_REJ	0x21	Q.931
MSG_RESUME_REJ	0x22	Q.931
MSG_HOLD	0x24	Q.932
MSG_SUSPEND	0x25	Q.931
MSG_RESUME	0x26	Q.931
MSG_HOLD_ACK	0x28	Q.932
MSG_SUSPEND_ACK	0x2D	Q.931

Message	**Hexadecimal value**	**Notes**
MSG_RESUME_ACK	0x2E	Q.931
MSG_HOLD_REJ	0x30	Q.932
MSG_RETRIEVE	0x31	Q.932
MSG_RETRIEVE_ACK	0x33	Q.932
MSG_RETRIEVE_REJ	0x37	Q.932
Call clearing messages:		
MSG_DISCONNECT	0x45	Q.931
MSG_RESTART	0x46	Q.931
MSG_RELEASE	0x4D	Q.931
MSG_RESTART_ACK	0x4E	Q.931
MSG_RELEASE_COMP	0x5A	Q.931
Miscellaneous messages:		
MSG_SEGMENT	0x60	Q.931
MSG_FACILITY	0x62	Q.931
MSG_REGISTER	0x34	Q.932
MSG_NOTIFY	0x6E	Q.931
MSG_STATUS_ENQ	0x75	Q.931
MSG_CONGESTION_CTRL	0x79	Q.931
MSG_INFO	0x7B	Q.931
MSG_STATUS	0x7D	Q.931
Messages not defined in Q.931/Q.932:		
MSG_SERVICE	0x0F	According to 235-900-342 (5ESS)
MSG_SERVICE_ACK	0x07	According to 235-900-342 (5ESS)

Information elements

dectrace decodes all the information element identifiers defined in the Q.931/Q.932 specifications. The following table lists the information element identifiers:

Message	Hexadecimal value	Notes
Single octet information elements:		
IE_SHIFT	0x90	All values from 0x90 to 0x9F indicate shift
IE_MORE_DATA	0xA0	More data
IE_SENDING_COMPL	0xA1	Sending complete
IE_CONG_LEVEL	0xB0	All values from 0xB0 to 0xBF indicate congestion level
IE_REPEAT_IND	0xD0	All values from 0xD0 to 0xDF indicate repeat indicator
Variable length information elements:		
IE_SEGMENTED	0x00	Segmented message
IE_BC	0x04	Bearer capability
IE_CAUSE	0x08	Cause
IE_CONNECTED	0xC	Connected party number
IE_CALL_ID	0x10	Call identity
IE_CALL_STATE	0x14	Call state
IE_CHANNEL_ID	0x18	Channel identification
IE_FACILITY	0x1C	Facility
IE_PROGRESS_IND	0x1E	Progress indicator
IE_NSF	0x20	Network-specific facilities
IE_NOTIFY_IND	0x27	Notification indicator
IE_DISPLAY	0x28	Display

Message	Hexadecimal value	Notes
IE_DATE_TIME	0x29	Date/time
IE_KEYPAD	0x2C	Keypad facility
IE_INFO_RQ	0x32	Information request
IE_SIGNAL	0x34	Signal
IE_SWITCHHOOK	0x36	Switchhook
IE_FEATURE_ACK	0x38	Feature activation
IE_FEATURE_IND	0x39	Feature indication
IE_SERVICE_PROF	0x3A	Service profile identification
IE_ENDPOINT_ID	0x3B	Endpoint identifier
IE_INFO_RATE	0x40	Information rate
IE_END_TO_END_DELAY	0x42	End to end transit delay
IE_TDSI	0x43	Transit delay selection and indication
IE_PKT_BIN_PAR	0x44	Packet layer binary parameters
IE_PKT_WIN_SIZE	0x45	Packet layer window size
IE_PKT_SIZE	0x46	Packet size
IE_MIN_THR_CLASS	0x47	Minimum throughput class
IE_CONNECTED_QSIG	0x4C	Connected party number (QSIG)
IE_CALLING	0x6C	Calling party number
IE_CALLING_SUB	0x6D	Calling party subaddress
IE_CALLED	0x70	Called party number
IE_CALLED_SUB	0x71	Called party subaddress
IE_ORIG_CALLED	0x73	Original called number
IE_REDIRECTING	0x74	Redirecting number

Message	**Hexadecimal value**	**Notes**
IE_REDIRECTION	0x76	Redirection number
IE_TRANSIT_SEL	0x78	Transit network selection
IE_REST_IND	0x79	Restart indicator
IE_LLC	0x7C	Low layer compatibility
IE_HLC	0x7D	High layer compatibility
IE_UUI	0x7E	User-to-user information
IE_ESCAPE	0xFF	Escape for extension
Information elements not defined in Q.931/Q.932:		
IE_CHANGE_STATUS	0x01	Change status according to 235-900-342 (5ESS)

If the program does not decode the contents of an information element, it displays:

```
IE ie_name: NOT DECODED
```

If the program does not decode the contents of one octet inside one information element, it displays:

```
octet octet_number: NOT DECODED
```

If *dectrace* encounters a value in the message type, the information element ID, or the information element contents that:

- Is not in accordance with the Q.931/Q.932 specifications (except where noted differently), it displays RESERVED.
- It does not recognize, it displays UNKNOWN. Unknown represents a legal value for several fields.

Note: The presence of RESERVED or UNKNOWN fields in the program output does not necessarily indicate a failure. Values that are illegal in the Q.931/Q.932 specification may be legal depending on the variant you are running.

Sample dectrace output

In the following sample of *dectrace* output text, the arrow after the board number shows if the message was received (<--) or sent (-->).

```
    43.61
          protocol discriminator = Q.931 Call Control
          call reference = 01 00  flag = 0
00000101  message type = SETUP   board 00  nai 00  group 00  -->
00000100    IE bearer capability
00000011        length = 0x03
              octet 3
1-------        extension bit
-00-----        coding standard = CCITT
---00000        information transfer capability = speech
              octet 4
```

```
1-------        extension bit
-00-----        transfer mode = circuit mode
---10000        information transfer rate = 64 Kbit/s
              octet 5
1-------        extension bit
-01-----        layer 1 id = layer 1 id
---00010        user info layer 1 = mu law
00011000    IE channel identification
00000011        length = 0x03
              octet 3
1-------        extension bit
-0------        interface identifier = implicitly identified
--1-----        interface type = PRI
---0----        spare bits = spare bits
----1---        pref/excl = exclusive
-----0--        D-channel ind = not
------01        info channel selection = B1 channel
              octet 3.2
1-------        extension bit
-00-----        coding standard = CCITT
---0----        number/map = number
----0011        channel type = B channel
              octet 3.3
1-------        extension bit
-0000001        channel number = 0x01
01110000    IE called party number
00000100        length = 0x04
              octet 3
1-------        extension bit
-000----        type of number = unknown
----0000        numbering plan = unknown
              octet 4 etc
                number = 12.

channel number = 0x01

    43.67
          protocol discriminator = Q.931 Call Control
          call reference = 01 00  flag = 1
00000010  message type = CALL_PROCEEDING   board 00  nai 00  group 00  <--
00011000    IE channel identification
00000011        length = 0x03
              octet 3
1-------        extension bit
-0------        interface identifier = implicitly identified
--1-----        interface type = PRI
---0----        spare bits = spare bits
----1---        pref/excl = exclusive
-----0--        D-channel ind = not
------01        info channel selection = B1 channel
              octet 3.2
1-------        extension bit
-00-----        coding standard = CCITT
---0----        number/map = number
----0011        channel type = B channel
              octet 3.3
1-------        extension bit
-0000001        channel number = 0x01


    45.84
          protocol discriminator = Q.931 Call Control
          call reference = 01 00  flag = 1
00000001  message type = ALERTING   board 00  nai 00  group 00  <--
```

14. itrace: Stack traffic analysis tool

itrace overview

itrace limits the number of debug ISDN messages sent to *oammon*. The ISDN stack includes many entities (for example, layer 1 entities, layer 2 entities, layer 3 entities, management entities, and timer entities) that exchange messages.

Debug information is sent to *oammon* when the *agtrace* bit 0x800000 is set. It can be filtered by this utility based on board number, NAI number, NFAS group number, and entity. By default, all the entities are enabled.

Usage

```
itrace [on|off] [board] [nai] [entity_list|*] [nfas_group]
```

where:

Option	Meaning
on|off	Logging enabled or disabled.
board	Board number.
nai	Network access identifier (NAI).
entity_list	String of entity IDs (as defined in *isdntype.h*). If the string starts with an exclamation point (!), the buffers associated with the messages are sent to the monitoring utility.
*	All entities.
nfas_group	NFAS group number for configurations with duplicate NAI values.

Using itrace

Perform the following steps to run the *itrace* utility:

Step	Action
1	Run *agtrace* with mask 801000 to enable the ISDN protocol stack messages to be passed to *oammon*.
2	Run *itrace* to enable or disable messages to and from entities in the ***entity_list*** string. **Note:** *itrace* can be run at any time of execution. Message logging is changed dynamically.

itrace file

itrace is supplied in executable format and is located in one of the following directories:

Operating system	Directory
Windows	*\nms\bin\itrace.exe*
UNIX	*/opt/nms/bin/itrace*

itrace examples

Example 1

Enable messages between ENT_APPLI and ENT_CC on board 0, nai 1:

```
agtrace 801000 0   Start tracing
itrace off 0 1 *   Disable all ISDN stack traffic on board 0, nai 1.
itrace on 0 1 AC   Enable logging messages between 'A' and 'C'
                   entities on board 0, nai 1
('A' = ENT_APPLI = application entity and
'C' = ENT_CC = Call Control Entity.)
```

Example 2

Enable messages and buffers between ENT_DL_D and ENT_PH_D on board 0, nai 1, NFAS group 0:

```
agtrace 801000 0    Start tracing
itrace off 0 1 * 0  Disable all ISDN stack traffic on board 0,
                    nai 1, NFAS group 0.
itrace on 0 1 !Dd 0 Enable logging messages between 'D' and 'd'
                    entities on board 0, nai 1, NFAS group 0
```

15. Events, reasons, and errors

NMS ISDN events

The following events are specific to NMS ISDN. Examine the reason code or error code stored in the value field of the event for more information about the event or about the result of the function's execution. For more information about event handling, refer to Receiving messages from the NMS ISDN protocol stack.

Event	Hexadecimal	Decimal	Description
ISDNEVN_ERROR	0x00072083	467075	A trunk error occurred.
ISDNEVN_RCV_MESSAGE	0x00072020	466976	A message was received from an ISDN trunk.
ISDNEVN_SEND_MESSAGE	0x00072004	466948	**isdnSendMessage** completed. The value field of this event contains the result of the function call.
ISDNEVN_SET_MSG_CAPTURE	0x00072082	467074	**isdnSetMsgCapture** completed. The value field of this event contains the result of the function call.
ISDNEVN_START_PROTOCOL	0x00072001	466945	**isdnStartProtocol** completed. The value field of this event contains the result of the function call.
ISDNEVN_STOP_PROTOCOL	0x00070002	466946	**isdnStopProtocol** completed. The value field of this event contains the result of the function call.

NMS ISDN reasons

The NMS ISDN reason codes in the following table can appear in the value fields of the events listed in the NMS ISDN events table.

Reason	Hexadecimal	Decimal	Description
ISDNERR_BAD_NAI	0x00071001	462849	The network access identifier (NAI) in the message structure is not valid. The NAI must be less than MAX_NAI.

Reason	**Hexadecimal**	**Decimal**	**Description**
ISDNERR_BUFFER_TOO_BIG	0x00071022	462882	A message buffer is too large.
ISDNERR_INCOMPATIBLE_LIB	0x00071017	462871	The ISDN library used is incompatible with the run file.
ISDNERR_INVALID_COUNTRY	0x00071014	462868	The country specified is invalid for the network operator specified.
ISDNERR_INVALID_HDLC_CHAN	0x00071016	462870	The HDLC controller number specified is invalid.
ISDNERR_INVALID_OPERATOR	0x00071012	462866	The network operator specified is not supported by the run file.
ISDNERR_INVALID_PARTNER	0x00071013	462867	***partner_equip*** specified in **isdnStartProtocol** is not supported by the run file.
ISDNERR_INVALID_PROTOCOL	0x00071011	462865	The protocol parameter is not supported by the run file.
ISDNERR_NAI_IN_USE	0x00071015	462869	Another thread or process has already started a protocol for the same network access identifier.
ISDNERR_PROTOCOL_CC_FAILURE	0x0007101D	462877	The call control parameters are invalid.
ISDNERR_PROTOCOL_DL_FAILURE	0x0007101B	462876	The data link parameters are invalid.
ISDNERR_PROTOCOL_NS_FAILURE	0x0007101C	462875	The network signaling parameters are invalid.
ISDNERR_PROTOCOL_PH_FAILURE	0x0007101A	462874	The physical layer parameters are invalid.

NMS ISDN errors

NMS ISDN functions can return the following error code:

Error	Hexadecimal	Decimal	Description
ISDNERR_INVALID_BUFFER	0x00070303	459523	The buffer submitted is not a valid buffer.

16. Parameters

ISDN_PROTOCOL_PARMS_LAPD parameters

The ISDN_PROTOCOL_PARMS_LAPD structure contains parameters that configure the ISDN protocol stack for LAPD. Either this structure or ISDN_PROTOCOL_PARMS_Q931CC is passed to **isdnStartProtocol**, depending upon how the ISDN protocol stack is configured.

For more information, refer to Initializing ISDN protocol stack instances. For a definition of the data structure, refer to ISDN_PROTOCOL_PARMS_LAPD structure.

Type	Parameter	Description	Default value	Range of values
DWORD	size	Size of the structure.	None.	
WORD	rate	Data rate.	ISDN_RATE_64K	ISDN_RATE_64K, ISDN_RATE_56K
WORD	max_FEC_errors	Maximum number of framing errors allowed during the T198 interval.	20	0 or more ms
timer_val_t	t101	Milliseconds of bad framing before disabling sending of D channel packets.	750	0 or more ms
timer_val_t	t102	Milliseconds of good framing before enabling sending D channel packets.	50	0 or more ms
timer_val_t	t198	Observation period for frame error count.	5	0 or more seconds
WORD	tei_time_assignment	TEI time assignment.	0	(reserved)

Type	Parameter	Description	Default value	Range of values
WORD	tei_time_removal	TEI time removal.	0	(reserved)
BYTE	bpad1[2]	Padding for 8 byte alignment.	0	0
WORD	nfas_group	NFAS group number if duplicate NAI values.	0	ON or OFF

ISDN_PROTOCOL_PARMS_Q931CC parameters

The ISDN_PROTOCOL_PARMS_Q931CC structure contains parameters that configure the ISDN protocol stack for Q.931 call control through the ACU. Either this structure or ISDN_PROTOCOL_PARMS_LAPD is passed to **isdnStartProtocol**, depending on how the ISDN protocol stack is configured.

For more information, refer to Initializing ISDN protocol stack instances. For a definition of the data structure, see ISDN_PROTOCOL_PARMS_Q931CC structure.

Type	Parameter	Description	Default value	Range of values
DWORD	size	Size of the structure.	None.	
WORD	rate	Data rate.	ISDN_RATE_64K	ISDN_RATE_64K, ISDN_RATE_56K
WORD	t309	T309 in use flag indicates if data link release and establish timers are in effect (used by D channel backup protocol). See D channel backup.	0	0, 1
timer_val_t	***xxx***	Timer values by country or operator.	See Timer overview.	See Timer overview.
BYTE	services_list[ ]	Services supported on incoming calls.	All services. See services_list field.	All services.

Type	Parameter	Description	Default value	Range of values
WORD	max_FEC_errors	Maximum number of framing errors allowed during the T198 interval.	20	0 or more ms
timer_val_t	t101	Milliseconds of bad framing before disabling sending of D channel packets.	750	0 or more ms
timer_val_t	t102	Milliseconds of good framing before enabling sending of D channel packets.	50	0 or more ms
timer_val_t	t198	Observation period for frame error count.	5	0 or more seconds
WORD	tei_time_assignment	TEI time assignment.	0	(reserved)
WORD	tei_time_removal	TEI time removal.	0	(reserved)
BYTE	tei[3]	TEI values.	0	(reserved)
BYTE	digitstoroute	Number of digits needed to route when using overlap receiving.	0	0 or more
WORD	in_calls_behaviour	Incoming calls behavior.	0	See in_calls_behaviour field.
WORD	out_calls_behaviour	Outgoing calls behavior.	0	See out_calls_behaviour field.

Type	Parameter	Description	Default value	Range of values
WORD	ns_behaviour	Bits controlling NS automatic responses.	0	See ns_behaviour field.
WORD	acu_behaviour	Bits controlling ACU automatic responses.	0	See acu_behaviour field.
BYTE	qsig_source_party_nb_type	Type of PINX node address. Used for network node addressing in supplementary services.	None.	See qsig_source_party_nb_type field.
BYTE	qsig_source_type_of_nb	Type of public PINX number. Used for network node addressing in supplementary services.	None.	See qsig_source_type_of_nb field.
BYTE	qsig_source_addr	Node address.	None.	None.
BYTE	aoc_s_presubscribed	Availability of Advice-of-Charge (Start of Call) supplementary service.	OFF	ON, OFF
BYTE	aoc_d_presubscribed	Availability of Advice-of-Charge (Start of Call) supplementary service.	OFF	ON, OFF
BYTE	aoc_e_resubscribed	Availability of Advice-of-Charge (Start of Call) supplementary service.	OFF	ON, OFF
BYTE	bpad2[1]	Padding for 8 byte alignment.	0	0

Type	Parameter	Description	Default value	Range of values
WORD	nfas_group	NFAS group number if duplicate NAI values.	0	0 - 255
BYTE	bpad3[2]	Padding for 8 byte alignment.	0	0
WORD	rfu1	Reserved for future use.	0	0
WORD	rfu2	Reserved for future use.	0	0

For details on the BYTE field, see the *NMS ISDN Supplementary Services Developer's Reference Manual*.

Protocol parameter settings

The following code segment illustrates the typical settings of the protocol parameter to **isdnStartProtocol** for application access to ACU SAP.

```
memset(&cc_parms, 0, sizeof(ISDN_PROTOCOL_PARMS_Q931CC));
cc_parms.rate = ISDN_RATE_64K;
cc_parms.services_list[0] = VOICE_SERVICE;
cc_parms.services_list[1] = NO_SERVICE;
```

services_list field

The services_list data field in ISDN_PROTOCOL_PARMS_Q931CC consists of a set of up to CC_MX_SERVICES elements, which together define the set of ACU services required by an instance of the ISDN protocol stack.

By default, all services are accepted. If you specify only certain services, and an incoming call requests a different service, the protocol stack automatically rejects the call.

The following available services are defined in *isdnval.h*:

Service	Description
FAX_SERVICE	G3 facsimile service.
FAX_4_SERVICE	G4 facsimile service.
DATA_SERVICE	Data service.
DATA_GCI_SERVICE	Data service on GCI bus.
DATA_56KBS_SERVICE	Data at 56 kbit/s service.
RAW_DATA_SERVICE	Raw data service on GCI bus: no MPH_B_INIT_RQ is generated (no B channel driver is associated).

Service	**Description**
DATA_TRANS_SERVICE	Transparent data service.
MODEM_SERVICE	Modem data service.
AUDIO_7_SERVICE	7 kHz audio service.
X25_SERVICE	X.25 circuit-mode service.
X25_PACKET_SERVICE	X.25 packet-mode service.
VOICE_SERVICE	Voice service.
VOICE_GCI_SERVICE	Voice service on GCI bus.
RAW_TELEPHONY_SERVICE	Raw telephony service on GCI bus: no MPH_B_INIT_RQ generated (no B channel driver is associated).
VOICE_TRANS_SERVICE	Transparent voice service.
V110_SERVICE	V.110 service.
V120_SERVICE	V.120 service.
VIDEO_SERVICE	Video service.
TDD_SERVICE	TDD service.
DATA_H0_SERVICE	Data using H0 (384 kbit/s) channel service (PRI only).
DATA_H11_SERVICE	Data using H11 (1536 kbit/s) channel service.
DATA_H12_SERVICE	Data using H12 (1536 kbit/s) channel service.
DATA_MULTIRATE_SERVICE	Data using multirate (2..30*64 kbit/s) channel service.
DATA_128KBS_SERVICE	Data using 2*64 kbit/s channel service (BRI only).
NO_B_CHAN_SERVICE	No B channel service (bearer-independent calls - QSIG only).
FAX_RELAY_SERVICE	G3 facsimile service (for use with physical relay process).

Service	Description
DATA_RELAY_SERVICE	Data service (for use with physical relay process).
DATA_56KBS_RELAY_SERVICE	Data at 56 kbit/s service (for use with physical relay process).
DATA_TRANS_RELAY_SERVICE	Data transparent service (for use with physical relay process).
MODEM_RELAY_SERVICE	Modem data service (for use with physical relay process).
X25_RELAY_SERVICE	X.25 circuit-mode service (for use with physical relay process).
VOICE_RELAY_SERVICE	Voice service (for use with physical relay process).
VOICE_GCI_RELAY_SERVICE	Voice service on GCI bus (for use with physical relay process).
NO_SERVICE	Undefined service.

The services array must be terminated by NO_SERVICE. Thus you can specify at most CC_MX_SERVICES minus one.

Note: The delivery of some of these services may be regulated by local authorities. You may be responsible for formally certifying these services in some countries. Check with the local authority for more specific information on these limitations.

in_calls_behaviour field

The in_calls_behaviour field in ISDN_PROTOCOL_PARMS_Q931CC determines how an incoming call is handled by the ISDN protocol stack. Refer to *isdnparm.h* for more details.

The bit settings in this field regarding call control actions (CC_SEND_ALERT_IN, CC_SEND_CALL_PROC_RQ, CC_DATA_ALERT_RQ, CC_VOICE_ALERT_RQ, CC_DATA_CONN_RS, and CC_VOICE_CONN_RS) must not be set when the protocol stack is started in channelized stack mode.

Each bit set by the value in this field determines a particular element of the ISDN protocol stack's behavior. The values are defined as follows:

Value	Description
--	Reserved for compatibility with former auto_answer field values (ON/OFF).
CC_SEND_ALERT_IN	Determines if ACU_ALERT_IN is automatically sent after ACU_CONN_IN. If this bit is set, the stack generates an ACU_ALERT_IN after sending an ACU_CONN_IN.

Value	**Description**
CC_SEND_CALL_PROC_RQ	Determines if CALL PROCEEDING is automatically sent on incoming calls. If this bit is set, the ACU sends CALL PROCEEDING on an incoming call. Otherwise, the application must send ACU_CALL_PROC_RQ for CALL PROCEEDING. (This message is optional.)
CC_DATA_ALERT_RQ	Determines if ALERTING is automatically sent when a NOT TELEPHONY call arrives. If this bit is set, the stack sends ALERTING on a NOT TELEPHONY incoming call.
CC_VOICE_ALERT_RQ	Determines if ALERTING is automatically sent when a TELEPHONY call arrives. If this bit is set, the stack sends ALERTING on a TELEPHONY incoming call.
CC_DATA_CONN_RS	Determines if NOT TELEPHONY incoming calls are automatically answered. If this bit is set, the stack sends CONNECT on NOT TELEPHONY incoming calls.
CC_VOICE_CONN_RS	Determines if TELEPHONY incoming calls are automatically answered. If this bit is set, the stack sends CONNECT on TELEPHONY incoming calls.
CC_TRANSPARENT_OVERLAP_RCV	If this bit is set, the stack sends an ACU_CONN_IN to the application even if not all digits have arrived (for example, the sending complete IE is not present). Additional incoming digits arrive in ACU_DIGIT_IN messages. If this bit is not set, then the ACU waits for at least nb_digits_to_route digits to arrive before sending an ACU_CONN_IN message to the application. Additional digits coming in INFORMATION messages are ignored.
CC_TRUNCATE_NB	Meaningful only in buffered Overlap Receiving mode. If this bit is set, the called number is truncated to the number of digits specified by the nb_digits_to_route configuration field.
CC_CALLED_NB_ABSENT_MATCH	If this bit is set, a received incoming call with no called number will match a list with a programmed address.
CC_CALL_WAITING	If this bit is set, the stack supports the call-waiting supplementary service. When this service is active, an incoming call received with a channel ID indicating No-channel will be accepted by the stack. If this bit is not set, then the stack rejects these calls with the appropriate cause value (#34 in most cases).
CC_SEND_NO_CALLED_NB	Not used.
CC_DISABLE_SUPPLEMENTARY_SERVICES	If this bit is set, supplementary services are disabled. Supplementary service extended data structures in the extended data area are ignored.

Value	Description
CC_SET_CHAN_ID	If this bit is set, the stack forces the channel ID information element to be present in the first message in response to a SETUP even if the specifications do not consider this mandatory.
CC_BEHAVIOUR_NIL	If this bit is set to 1 and all other bits in in_calls_behaviour are set to 0, no messages are automatically sent by the stack in response to an incoming call. The application must build and send all messages. If any other bits are set, this bit is ignored. The stack sends all specified messages.

If in_calls_behaviour is set to 0 or not set at all, the ISDN protocol stack behaves in one of the following ways, depending upon how the ***partner_equipment*** argument is set in the call to **isdnStartProtocol**:

If partner_equipment is set to:	The stack behaves as if:
EQUIPMENT_TE	CC_SEND_CALL_PROC_RQ is set; all other bits cleared.
EQUIPMENT_NT	CC_VOICE_ALERT_RQ and CC_DATA_ALERT_RQ are set; all other bits cleared.

When using the 4ESS variant, all three bits are set, both on the TE side and NT side.

out_calls_behaviour field

The out_calls_behaviour field in ISDN_PROTOCOL_PARMS_Q931CC determines how an outgoing call is handled by the ISDN protocol stack. Refer to *isdnparm.h* for more details.

Each bit set by the value in this field determines a particular element of the ISDN protocol stack's behavior. The values are defined as follows:

Value	Description
--	Reserved for compatibility with former auto_answer field values (ON/OFF).
CC_USER_SENDING_COMPLETE	If this bit is set, the stack does not automatically generate the Sending-complete IE. Instead, the application must request it in ACU_CONN_RQ. If the bit is not set, the stack generates the sending complete IE automatically in ACU_CONN_RQ only. This bit does not affect ACU_DIGIT_RQ. The application must specify the Acu_digit_rq_sending_complete field value explicitly.

Value	Description
CC_SEND_CONN_CO_ON_PROGRESS	If this bit is set, the stack sends ACU_CONN_CO when it receives a PROGRESS message for a VOICE call. Otherwise, it sends ACU_CONN_CO when it receives a CONNECT message.
CC_SEND_DIGIT_CO_ON_PROGRESS	If this bit is set and the stack receives NS_CALL_PROC_IN before NS_SETUP_ACK_IN, it sends ACU_DIGIT_CO and then ACU_CALL_PROC_IN. Otherwise, it only sends ACU_CALL_PROC_IN (default behavior).
CC_USE_MU_LAW	For Korean operators only. If this bit is set, the stack sends G711-Mu-Law in outgoing voice calls. If this bit is cleared, the stack sends G711-A-Law in outgoing voice calls. The value of this bit is used by the stack in all stack modes.
CC_USE_A_LAW	Forces A law specification in the Bearer Capability IE.
CC_E1_CONTINUOUS_CHANNELS	Makes internal assignment in the stack of the D channel to timeslot 31.
CC_SET_CALL_ID_TO_CRV	If this bit is set, the call ID returned to the application is the call reference value.
CC_USE_PATH_REPLACEMENT	QSIG only. If this bit is set, then instead of using explicit call transfer when **nccTransferCall** is called, path replacement is invoked.
CC_NETWORK_TEST_FACILITY	Reserved for test purposes only.
CC_E1_CONTINUOUS_CHANNELS_LOGICAL	Forces B-channel numbering to be 1 through 30 inclusive.
CC_USE_SINGLE_STEP_TRANSFER	QSIG only. If this bit is set, then calling **nccAutomaticTransfer** invokes the single step transfer supplementary service.
CC_BEHAVIOUR_NIL	If this bit is set to 1 and all other bits in out_calls_behaviour are set to 0, no messages are sent by the stack during an outgoing call. The application must build and send all messages. If any of the other bits are set, this bit is ignored. The stack sends all messages you specify.

If out_calls_behaviour is set to 0 or not set at all, the ISDN protocol stack behaves as if all out_calls_behaviour bits are cleared.

acu_behaviour field

The acu_behaviour field in ISDN_PROTOCOL_PARMS_Q931CC determines what automatic responses the ACU layer makes. Refer to *isdnparm.h* for more details.

The bit settings in this field are ignored when the protocol stack is started in channelized stack mode. Each bit set by the value in this field determines a particular element of the ACU layer's behavior. The values are defined as follows:

Value	Description
ACU_SEND_Q931_BUFFER	If this bit is set, the stack sends a whole received Q.931 message to the application, along with the ACU primitive.
ACU_SEND_D_CHANNEL_STATUS_CHANGE	If this bit is set, when the status of the D channel changes, an ACU_D_CHANNEL_STATUS_IN primitive is automatically sent to the application, indicating the change. The Acu_d_channel_state in this primitive indicates the status of the channel: 0 = OFF 1 = ON
ACU_SEND_UNKNOWN_FACILITY	If this bit is set, the stack sends the application an ACU_FACILITY_IN message containing the whole Q.931 buffer with the unknown facility IE. Setting this flag automatically enables the NS_ACCEPT_UNKNOWN_FAC_IE behavior bit.
ACU_BEHAVIOUR_NIL	This is the default value corresponding to the behavior of the ACU entity if neither of the previous bits are set.

If acu_behaviour is set to 0 or not set at all, the ISDN protocol stack behaves as if the ACU_BEHAVIOUR_NIL bit is set.

ns_behaviour field

The ns_behaviour field in ISDN_PROTOCOL_PARMS_Q931CC determines which automatic responses the NS layer makes. Refer to *isdnparm.h* for more details.

The bit settings in this field are ignored when the protocol stack is started in channelized stack mode. Each bit set by the value in this field determines a particular element of the NS layer's behavior. The values are defined as follows:

Value	Description
NS_NO_STATUS_ON_UNKNOWN_IE	If this bit is set, the stack does not generate a STATUS message when it receives a message containing one or more unknown/unrecognized IE(s). This bit applies only to network variants for which the sending of STATUS under these circumstances is optional.

Value	Description
NS_NO_STATUS_ON_INV_OP_IE	If this bit is set, the stack does not generate a STATUS message when it receives a message containing one or more optional IE(s) with invalid content. This bit applies only to network variants for which the sending of STATUS under such circumstances is optional.
NS_ACCEPT_UNKNOWN_FAC_IE	If this bit is set, the stack accepts incoming messages containing facility IEs that it does not recognize. The stack does not check the IE: it operates as if the facility IE is correct. If this bit is not set, the stack rejects messages containing unknown facility IEs. This bit must be set for the bridge calls and notify transfer supplementary services to operate. For more information, see the *NMS ISDN Supplementary Services Developer's Reference Manual*.
NS_IE_RELAY_BEHAVIOUR	This bit applies only when the NS_RELAY compile-time option is set to ON. The stack passes IEs received from the line transparently (as in incoming transparent mode). Layer 3 procedures are still in use. You can send IEs transparently to the line using a second buffer, possibly including unknown or unexpected IEs. The NS layer does not consider unknown or unexpected IEs received from the line as erroneous.
NS_SEND_USER_CONNECT_ACK	Setting this bit applies when the configuration is ETS, EUROPE, TE-side, outgoing call. If this bit is set, the ISDN stack sends a CONNECT_ACK message in response to a received CONNECT message.
NS_EXPLICIT_INTERFACE_ID	If this bit is set, outbound call control messages (for example, SETUP or PROCEEDING) contain the NAI in the channel ID IE, as defined in the configuration file. Use this bit only for US variants.
NS_PRESERVE_EXT_BIT_IN_CHAN_ID	This bit applies when the configuration is DMS and USA, and for an incoming call. When it is set, the extension bit in the channel ID's octets 3.3 is set to the value used in the SETUP message, for use inside PROCEEDING or ALERT messages.
NS_NO_B_CHANNEL_MANAGEMENT	Reserved for test purposes only.
NS_DISABLE_RESTART	This bit is used to disable the RESTART procedure in the stack. If this bit is set, the stack does not send RESTART messages when it is stopped, but it also does not respond to incoming RESTART messages. Do not set this bit unless you are certain that the remote end does not implement the RESTART procedure.

Value	Description

NS_PBX_XY	DPNSS only. This bit reverses the default X and Y assignments.
	
	The default assignments (NS_PBX_XY = 0) are:
	TE => PBX A, side X
	NT => PBX B, side Y
	
	Setting this bit changes the assignments to:
	TE => PBX A, side Y
	NT => PBX B, side X

NS_PBX_XY_ALTERNATE	DPNSS only. This bit supports the following channel configuration:

Channel	Configuration 1	Configuration 2
1	X	Y
2	Y	X
...		
14	Y	X
15	X	Y
16	none	(signaling)
17	X	Y
18	Y	X
...		
30	Y	X
31	X	Y

Value	Description
	If this bit is not set, all channels have the same configuration as channel 1. If this bit is set, all odd channels (1, 3 ... 15, 17 ... 31) have the same configuration as channel 1. All even channels (2, 4, 6 ... 14, 16 ... 30) have a configuration opposite to that of channel 1 (for example, Y if channel 1 is X and vice versa).
NS_BEHAVIOUR_NIL	If this bit is set, the ISDN stack behaves as if none of the other ns_behaviour bits are set.

If ns_behaviour is set to 0 or not set at all, the ISDN protocol stack behaves as if the NS_IE_RELAY_BEHAVIOUR bit is set and all other bits are cleared.

qsig_source_party_nb_type field

The qsig_source_party_nb_type field in ISDN_PROTOCOL_PARMS_Q931CC is used when the application is designed for a Q.SIG private ISDN exchange (PINX). Each node in a Q.SIG network has an address. This field is used with qsig_source_type_of_nb and qsig_source_addr to specify the address of the node.

The qsig_source_party_nb_type field specifies the type of PINX node address. The following table lists possible values:

Value	Description
ACU_QSIG_PINX_NB_UNKNOWN	Unknown party number (type_of_nb field is meaningless)
ACU_QSIG_PINX_NB_PUBLIC	Public party number.
ACU_QSIG_PINX_NB_PRIVATE	Private party number.

qsig_source_type_of_nb field

The qsig_source_type_of_nb field in ISDN_PROTOCOL_PARMS_Q931CC is used when the application is designed for a Q.SIG private ISDN exchange (PINX). Each node in a Q.SIG network has an address. This field is used with qsig_source_party_nb_type and qsig_source_addr to specify the address of the node.

The qsig_source_type_of_nb field specifies the type of public or private PINX number.

If qsig_source_party_nb_type is set to ACU_QSIG_PINX_NB_PUBLIC, the following values are valid for qsig_source_type_of_nb:

Value	Description
ACU_QSIG_PINX_PUB_NB_UNKNOWN	Public or private: unknown number.
ACU_QSIG_PINX_PUB_NB_INTERNATIONAL	Public: international number.
ACU_QSIG_PINX_PUB_NB_NATIONAL	Public: national number.
ACU_QSIG_PINX_PUB_NB_NETWORK_SPFC	Public: network specific number.
ACU_QSIG_PINX_PUB_NB_SUBSCRIBER	Public: subscriber number.
ACU_QSIG_PINX_PUB_NB_ABBREVIATED	Public or private: abbreviated number.

If qsig_source_party_nb_type is set to ACU_QSIG_PINX_NB_PRIVATE, the following values are valid for qsig_source_type_of_nb:

Value	**Description**
ACU_QSIG_PINX_PRIV_NB_UNKNOWN	Public or private: unknown number.
ACU_QSIG_PINX_PRIV_NB_LEVEL2_REGIONAL	Private: level 2 regional number.
ACU_QSIG_PINX_PRIV_NB_LEVEL1_REGIONAL	Private: level 1 regional number.
ACU_QSIG_PINX_PRIV_NB_PTN_SPECIFIC	Private: PTN specific number.
ACU_QSIG_PINX_PRIV_NB_LOCAL	Private: local number.
ACU_QSIG_PINX_PRIV_NB_ABBREVIATED	Public or private: abbreviated number.

17. Sending and receiving raw Q.931 data

Overview of Q.931 data

NMS ISDN allows an application to include raw Q.931 data in one or more completely custom-built information elements (IEs) in messages sent to the stack. These information elements, called transparent IEs, are inserted verbatim in the Q.931 message generated by the stack. This specification method allows an application access to IEs and fields in IEs that cannot be accessed using the macros associated with ACU messages. This method can be used to specify both standard IEs (codeset 0) and extensions (such as codeset 6 and 7).

An application can also access and read the raw data in an incoming Q.931 message, rather than reading returned values for specific fields in data structures.

Creating transparent IEs

The application supplies the data (in hexadecimal format) for the transparent IEs in a buffer referenced in one of the standard ACU messages. The following macros refer to the transparent IE buffer:

Macro	**Description**
acumessage_tsp_ie_list_size	Size of transparent IE buffer, in bytes. ***acumessage*** is a standard ACU message (for example ACU_CONN_RQ). For example, Acu_conn_rq_tsp_ie_list_size. Do not include a null terminator when calculating this value.
acumessage_a_tsp_ie_list	Pointer to transparent IE buffer. Do not include a null terminator in this string.

To use transparent IEs, the application must disable the stack's syntax checking mechanism. To do this, set the NS_IE_RELAY_BEHAVIOUR bit in the ns_behaviour substructure referenced in the ISDN_PROTOCOL_PARMS_Q931CC structure passed to **isdnStartProtocol**. By default, this bit is 0. The following code fragment shows how to set this bit prior to calling **isdnStartProtocol**:

```
myStartProtocol()
{
  struct ISDN_PROTOCOL_PARMS_Q931CC parms;
...
  memset(&parms,0,sizeof(ISDN_PROTOCOL_PARMS_Q931CC));
  parms.size = sizeof(ISDN_PROTOCOL_PARMS_Q931CC);
  parms.services_list[0] = ACU_VOICE_SERVICE;
                /* Other services may be added here */
  parms.services_list[1] = ACU_NO_SERVICE;
  parms.ns_behaviour = NS_IE_RELAY_BEHAVIOUR;
...
  isdnStartProtocol (   ctahd,
            ISDN_PROTOCOL_Q931CC,
            networkoperator,
            country,
            partner,
            SM->nai,
            &parms );          /* Instead of NULL */
...
}
```

Transparent IE formatting rules

The following rules apply to transparent IE formatting:

- If the buffer contains more than one IE, the IEs must appear in the same order as they appear in the Q.931 message. For example, the channel ID (IE ID = 0x18) cannot be followed by bearer capability (IE ID = 0x04).
- When the stack's syntax checking mechanism is disabled, the stack does not perform syntax checking on transparent IEs. It is the application's responsibility to build the transparent IEs according to the specifications for the variant the application is using.
- The application should not use the transparent IE method and the macro method to access the same IE in the same ACU message. For example, if the application sets the macros for the called party number and includes the called party number IE (IE ID = 0x70) in the transparent IE buffer in the same ACU message, both IEs are present in the final SETUP message.

  Certain information elements are automatically generated by the stack in Q.931 messages. The stack checks to see if any of the transparent IEs sent to it correspond to these information elements. If it finds a transparent IE equivalent for an automatically generated information element, it uses the transparent IE instead. See the following table for a list of IEs automatically generated by the stack.
- The transparent IE buffer must not be null-terminated.
- When calculating the size of a transparent IE buffer, do not include a null terminator. The following table lists IEs automatically generated by the stack:

Name	Applies to variant
bc	All variants
llc	All variants except HKT, NTT, QSI, SWD
hlc	All variants except E10
cause	All variants
chan_id	All variants

- The transparent IE must be formatted correctly, as shown in the following table. The multiple octet IEs in a Q.931 message follow an ascending numerical order by IE ID (for example, octet #1). Single octet IEs can appear at any point in the message. In the table, ***u*** and ***v*** represent bits that can be changed by the user:

IE type	Information element id: octet #1	Information element id: octet #2	Information element id: octet #3
Single octet IE	0x1***uuuuuuu***	Not present	Not present

IE type	**Information element id: octet #1**	**Information element id: octet #2**	**Information element id: octet #3**
Multiple octet IE	0x0***uuuuuuu***	0x***vvvvvvvv***	Follow as many octets as specified in octet #2

Accessing and reading a Q.931 buffer

To read the raw data in an incoming Q.931 message, the application uses the following macros to access the buffer containing the message. ***acumessage*** is a standard ACU message (for example ACU_CONN_IN). For example, Acu_conn_in_a_q931.

Macro	**Description**
acumessage_q931_size1	Size of Q.931 buffer, in bytes
acumessage_a_q931	Pointer to Q.931 buffer

For this data to be available, the NS_BEHAVIOUR_NIL bit must be set in the acu_behaviour substructure referenced in the ISDN_PROTOCOL_PARMS_Q931CC structure passed to **isdnStartProtocol**. By default, this bit is 0.

The following code fragment shows how to set this bit prior to calling **isdnStartProtocol**:

```
myStartProtocol()
{
  struct ISDN_PROTOCOL_PARMS_Q931 parms;
...
  memset(&parms,0,sizeof(ISDN_PROTOCOL_PARMS_Q931CC));
  parms.size = sizeof(ISDN_PROTOCOL_PARMS_Q931CC);
  parms.services_list[0] = ACU_VOICE_SERVICE;
                /* Other services may be added here */
  parms.services_list[1] = ACU_NO_SERVICE;
  parms.acu_behaviour = ACU_SEND_Q931_BUFFER;
...
  isdnStartProtocol (   ctahd,
            ISDN_PROTOCOL_Q931CC,
            networkoperator,
            country,
            partner,
            SM->nai,
            &parms );          /* Instead of NULL */
...
}
```

The buffer is not null terminated. It cannot be read by functions that expect null termination (such as *strcpy*).

The total amount of data (all buffers, including the raw Q.931 data buffer) that can be received in an ISDN message is MAX_ISDN_BUFFER_SIZE. (This value is defined in *isdnparm.h*.) If the size of incoming data is greater than MAX_ISDN_BUFFER_SIZE, the raw Q.931 data buffer is omitted.

18. Sending and receiving PCS-user information

PCS-user information elements

The user-to-PCS and PCS-to-user information elements are network-specific information elements coded using codeset 6, used to transmit information between the user and a PCS (Point de Commande de Service). They are used in the French ISDN variant (VN6). These information elements can be included in several ISDN message types, in the call setup, call connected, and call disconnection phases.

Structure of user-PCS IEs

Each user-to-PCS and PCS-to-user information element includes a protocol discriminator field and an information area.

This information area contains differing amounts of information, depending on the associated message type:

For these message types...	The information area of the PCS IEs can consist of...
FACILITY	As many as 128 octets
ALERTING DISCONNECT PROGRESS RELEASE RELEASE COMPLETE SETUP	As many as 64 octets

Sending and receiving PCS information elements

An application using the ISDN Messaging API interface can send and receive user-to-PCS and PCS-to-user information elements, for the VN6 variant only, in the following primitives:

- ACU_ALERT_IN
- ACU_ALERT_RQ
- ACU_CLEAR_IN
- ACU_CLEAR_CO
- ACU_CLEAR_RQ
- ACU_CLEAR_RS
- ACU_CONN_CO
- ACU_CONN_IN
- ACU_CONN_RQ
- ACU_CONN_RS
- ACU_FACILITY_IN
- ACU_FACILITY_RQ
- ACU_PROGRESS_IN

The NMS ISDN stack provides three macros for each primitive, for accessing these information elements. (***xxx*** represents the primitive name, such as conn_rq):

Macro	**Description**
Acu_***xxx***_pcs_user_size	Size of the pcs_user information.
Acu_***xxx***_pcs_user_protocol	Protocol discriminator for the pcs_user information element (can assume the values: ACUPCS_USER_TRANSGROUP, ACUPCS_USER_PUBLIPHONE, ACUPCS_USER_CALL_ROUTING, ACUPCS_USER_DIALOGUE).
Acu_***xxx***_a_pcs_user	Address of the pcs_user information.

The application can send pcs_user macros in the connected state, using the ACU_FACILITY_RQ primitive. In this case, in addition to the pcs_user macros previously described, the following macros must be set:

Macro	**Must be set to...**
Acu_facility_code	ACU_FAC_PCS_USER_ONLY
Acu_facility_action	ACU_RQ_ACTIVATE

The user-to-PCS information element can be sent only in the user-to-network direction. The PCS-to-user information element can be sent only in the network-to-user direction.

Therefore, the two information elements are never present at the same time in an ISDN message. For this reason, the same macros are used for both information elements.

There are no restrictions on the contents of the information area in the user-to-PCS or PCS-to-user information elements. Non-printable characters, including \0, can be sent and received. However, for consistency with other macros and for ease of use, the stack automatically adds a \0 character at the end of any incoming pcs_user information. For example, if the incoming pcs_user information is 0x31 0x32 0x33 (representing the number 123), the stack sets the Acu_***xxx***_pcs_user_size macro to 4 and adds a 0x00 byte.

For outbound pcs_user information elements, the stack sends the exact number of octets specified by the application in Acu_***xxx***_pcs_user_size. For example, to send 0x31 0x32 0x33, the application sets Acu_***xxx***_pcs_user_size to 3.

Example code

The following sample code illustrates how to build a structure containing PCS information, prior to sending it:

```
void build_facility_with_pcs(char *buffer, int *len)
{
struct acu_facility *p_data;
/* For simplicity let's use a string. We could also have used non-printable
characters here and used memcpy instead of strcpy */
char pcs_string[] = "pcs_string";

p_data = (struct acu_facility *)buffer;

memset(p_data, OFF, ISDN_BUFFER_DATA_LGTH);

Acu_facility_code = ACU_FAC_PCS_USER_ONLY;
Acu_facility_action = ACU_RQ_ACTIVATE;
Acu_facility_pcs_user_protocol = ACUPCS_USER_TRANSGROUP;
strcpy(Acu_facility_a_pcs_user,pcs_string);
Acu_facility_pcs_user_size = strlen(pcs_string);

*len = Acu_facility_total_size;

return;
}
```

19. Timers

Timer overview

Values are assigned to various ISDN timers for different country variants. These timers control the behavior of network signaling layer 3 in the ISDN protocol stack.

These values are included as timer_val_t structures in the ISDN_PROTOCOL_PARMS_Q931CC and ISDN_PROTOCOL_PARMS_LAPD structures referenced by **isdnStartProtocol**.

The following table summarizes the meanings of the timers. The items shown in all capital letters are the names of Q.931 messages. For example, ALERT refers to the alert message indicating that an incoming call has arrived.

Timer	Description	Started...	Reset...	On expiration...
T300	Internal timer. Determines the time the stack has to respond to an incoming SETUP or DISCONNECT. For SETUP, if none of the in_calls_behaviour bits are set, the application must initiate this response.	After a SETUP or DISCONNECT is received.	When the first message in response to the SETUP or DISCONNECT is sent.	The call is cleared.
T301	ALERT message timing.	After an ALERT is received.	When a CONN is received.	The call is cleared.
T302	Overlap receiving timer.	After a SETUP_ACK is sent.	When an INFO is received.	The call is cleared.
T303	Setup message timing.	After a SETUP is sent.	When a CALL_PROC, ALERT, SETUP_ACK, or REL_COM is received.	The call is cleared with a REL_COM.
T304	Control of overlapped sending state.	After an INFO is sent.	When an INFO is received.	A DISC is sent.

Timer	Description	Started...	Reset...	On expiration...
T305	Disconnection control.	After a DISC is sent.	When a REL or REL_COM is received.	The link is placed in maintenance state and CRVs are released.
T306	Call is in disconnect indication.	After a DISC with progress indicator is received.	When a REL or DISC is received.	A REL is sent on the line (ACU_CLEAR_CO is sent to the application).
T307	Internal timer.	After SUSPEND_ACK is sent.	When a RESUME_ACK is received.	The call is cleared (ACU_CLEAR_CO is sent to the application).
T308	Release message control.	After a REL is sent.	When a REL or REL_COM is received.	The link is placed in maintenance state and CRVs are released.
T309	Allows the data link to be dropped without losing calls.	After a data link release message is sent.	When a data link establish message is received.	All calls are cleared locally.
T310	Used to govern the behavior of CALL_PROC.	After a CALL_PROC is received.	When an ALERT, CONN, DISC, or PROG is received.	A DISC is sent.
T313	Controls the behavior of CONN message.	After a CONN is sent.	When CONN_ACK is received.	A DISC is sent.
T314	Future usage for segmented messages.	After a message segment is received.	After last message segment is received.	The incomplete message is discarded.
T316	RESTART procedure timing.	After a RESTART is sent.	----	A RESTART is sent and the timer is restarted.
T317	Controls the internal clearing of CRVs after a RESTART.	After a RESTART is received.	----	A maintenance indication sent to the application.

Timer	Description	Started...	Reset...	On expiration...
T318	Used when the state of the call is suspend request.	When a RESUME message is sent.	When a RESUME_ACK or RESUME_REJ is received.	The call is cleared (ACU_CLEAR_CO is sent to the application).
T319	Used when the state of the call is suspend request.	When a SUSPEND message is sent.	When a SUSPEND_ACK, SUSPEND_REJ is received.	ACU_CLEAR_CO is sent to the application.
T320	Internal timer.	When a DL establishment indication or confirmation is received.	When a call request packet or DL_RELEASE is received.	DL_RELEASE request is sent to the line.
T321	Controls the internal timing of backup D channels.	----	----	----
T322	Controls the behavior of STATUS_ENQUIRY.	After a STATUS_ENQUIRY is sent.	When a STAT, DISC, REL, or REL_COM is received.	STATUS_ENQUIRY is submitted.

France VN6 layer 3 timer defaults

```
timer_val_t vn6[VN6_T_LAST]
```

Timer index	Value (network side, in seconds)	Value (user side, in seconds)
VN6_T300	1	1
VN6_T301	180	0
VN6_T302	15	15
VN6_T303	4	10
VN6_T304	20	30
VN6_T305	30	30
VN6_T306	30	0
VN6_T307	180	0

Timer index	Value (network side, in seconds)	Value (user side, in seconds)
VN6_T308	4	4
VN6_T309	90	90
VN6_T310	20	60
VN6_T312	4	0
VN6_T313	0	4
VN6_T3141	4	4
VN6_T316 (not exposed to the API)	120	120
VN6_T317 (not exposed to the API)	100	100
VN6_T318	0	4
VN6_T319	0	4
VN6_T320	30	0
VN6_T321	30	0
VN6_T322	4	4
VN6_T399	0	10

EuroISDN layer 3 timer defaults

```
timer_val_t ets[ETS_T_LAST]
```

Timer index	Value (network side, in seconds)	Value (user side, in seconds)
ETS_T300	1	1
ETS_T301	180	0
ETS_T302	15	15
ETS_T303	4	4
ETS_T304	20	30

Timer index	Value (network side, in seconds)	Value (user side, in seconds)
ETS_T305	30	30
ETS_T306	30	0
ETS_T307	180	0
ETS_T308	4	4
ETS_T309	90	90
ETS_T310	40	45
ETS_T312	6	0
ETS_T313	0	4
ETS_T316 (not exposed to the API)	120	120
ETS_T317 (not exposed to the API)	100	100
ETS_T318	0	4
ETS_T319	0	4
ETS_T320	30	0
ETS_T321	30	0
ETS_T322	4	4

AT&T E10 layer 3 timer defaults

```
timer_val_t e10[E10_T_LAST]
```

Timer index	Value (network side, in seconds)	Value (user side, in seconds)
E10_T300	2	2
E10_T303	4	4
E10_T305	4	4
E10_T306	4	0

Timer index	Value (network side, in seconds)	Value (user side, in seconds)
E10_T308	4	4
E10_T309	90	90
E10_T310	10	45
E10_T313	N/A	4
E10_T316 (not exposed to the API)	30	30
E10_T317 (not exposed to the API)	120	120
E10_T321 (not exposed to the API)	5	5
E10_TSRV (not exposed to the API)	0	60

Nortel DMS 100 layer 3 timer defaults

```
timer_val_t dms[DMS_T_LAST]
```

Timer index	Value (network side, in seconds)	Value (user side, in seconds)
DMS_T300	1	1
DMS_T301	180	0
DMS_T303	4	4
DMS_T305	30	30
DMS_T308	4	4
DMS_T309	90	90
DMS_T310	10	45
DMS_T312	6	4
DMS_T313	N/A	4
DMS_T316 (not exposed to the API)	120	120
DMS_T317 (not exposed to the API)	0	100

Timer index	**Value (network side, in seconds)**	**Value (user side, in seconds)**
DMS_T321 (not exposed to the API)	40	40
DMS_T322	4	0
DMS_T3DW (not exposed to the API)	5	5
DMS_T3M1 (not exposed to the API)	120	120
DMS_T3MB (not exposed to the API)	5	5
DMS_TSPID (not exposed to the API)	0	20

USA National ISDN 2 layer 3 timer defaults

```
timer_val_t ni2[NI2_T_LAST]
```

Timer index	**Value (network side, in seconds)**	**Value (user side, in seconds)**
NI2_T300	1	1
NI2_T301	180	0
NI2_T302	15	0
NI2_T303	5	4
NI2_T305	30	30
NI2_T306	60	0
NI2_T308	4	4
NI2_T309	30	30
NI2_T310	40	0
NI2_T313	4	4
NI2_T316 (not exposed to the API)	30	30

Timer index	Value (network side, in seconds)	Value (user side, in seconds)
NI2_T317 (not exposed to the API)	20	20
NI2_T321 (not exposed to the API)	5	40
NI2_TSPID (not exposed to the API)	0	20

Australian Telecom 1 layer 3 timer defaults

```
timer_val_t au1[AU1_T_LAST]
```

Timer Index	Value (network side, in seconds)	Value (user side, in seconds)
AU1_T300	1	1
AU1_T302	15	0
AU1_T303	4	15
AU1_T305	30	30
AU1_T306	60	0
AU1_T308	4	4
AU1_T309	2	90
AU1_T310	10	0
AU1_T312	6	4
AU1_T316 (not exposed to the API)	30	30
AU1_T317 (not exposed to the API)	25	25
AU1_T322	4	4

Hong Kong Telephone layer 3 timer defaults

```
timer_val_t hkt[HKT_T_LAST]
```

Timer index	**Value (network side, in seconds)**	**Value (user side, in seconds)**
HKT_T300	1	1
HKT_T301	180	180
HKT_T302	10	15
HKT_T303	4	4
HKT_T304	20	15
HKT_T305	30	30
HKT_T306	30	0
HKT_T307	180	0
HKT_T308	4	4
HKT_T309	90	90
HKT_T310	10	10
HKT_T312	6	4
HKT_T313	4	4
HKT_T314 (not exposed to the API)	4	4
HKT_T316 (not exposed to the API)	120	120
HKT_T317 (not exposed to the API)	90	100
HKT_T318	0	4
HKT_T319	0	4
HKT_T320	30	0
HKT_T321	30	30
HKT_T322	4	4

INS-1500 NTT layer 3 timer defaults

```
timer_val_t ntt[NTT_T_LAST]
```

Timer index	**Value (network side, in seconds)**	**Value (user side, in seconds)**
NTT_T300	3	3
NTT_T301	180	0
NTT_T303	5	5
NTT_T305	30	30
NTT_T306	30	0
NTT_T307	180	0
NTT_T308	4	4
NTT_T309	90	90
NTT_T310	10	45
NTT_T312	6	4
NTT_T313	4	4
NTT_T314 (not exposed to the API)	4	4
NTT_T316 (not exposed to the API)	120	120
NTT_T317 (not exposed to the API)	100	100
NTT_T318	0	4
NTT_T319	0	4
NTT_T322	4	4
NTT_T3JA	0	50

AT&T 4ESS layer 3 timer defaults

```
timer_val_t at4[AT4_T_LAST]
```

Timer index	Value (network side, in seconds)	Value (user side, in seconds)
AT4_T300	1	1
AT4_T303	4	4
AT4_T305	4	4
AT4_T306	60	4
AT4_T308	4	90
AT4_T309	90	30
AT4_T310	10	4
AT4_T313	N/A	4
AT4_T316 (not exposed to the API)	120	120
AT4_T317 (not exposed to the API)	0	0
AT4_T321 (not exposed to the API)	40	40
AT4_T3M1 (not exposed to the API)	120	120

Korean (and Taiwan) layer 3 timer defaults

```
timer_val_t kor[KOR_T_LAST]
```

Timer index	Value (network side, in seconds)	Value (user side, in seconds)
T300	1	1
T301	180	N/A
T302	15	15
T303	4	4
T304	20	30
T305	30	30

Timer index	**Value (network side, in seconds)**	**Value (user side, in seconds)**
T306	30	N/A
T307	180	N/A
T308	4	4
T309	90	90
T310	40	45
T312	6	0
T313	N/A	4
T314	4	4
T316	120	120
T317	100	100
T318	0	4
T319	0	4
T320	30	0
T321	30	0
T322	4	4
T-HOLD	4	4
T_RETRIEVE	4	4

QSIG layer 3 timer defaults

```
timer_val_t qsi[QSI_T_LAST]
```

Timer index	Value (network side, in seconds)	Value (user side, in seconds)
QSI_T300	1	1
QSI_T301	180	180
QSI_T302	15	15
QSI_T303	4	4
QSI_T304	20	20
QSI_T305	30	30
QSI_T308	4	4
QSI_T309	90	90
QSI_T310	30	30
QSI_T313	4	4
QSI_T314 (not exposed to the API)	4	4
QSI_T316 (not exposed to the API)	120	120
QSI_T322	4	4

20. NFAS and D channel backup

Non-facility associated signaling (NFAS)

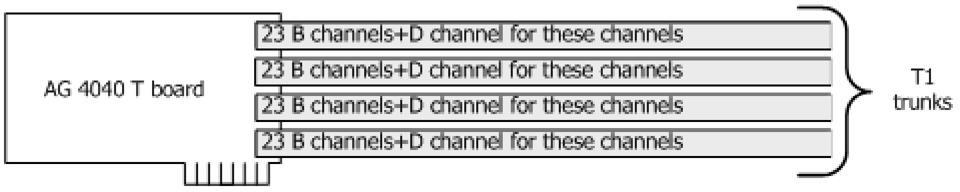
In setups with multiple T1 ISDN trunks, you can set up a single D channel to serve all trunks. This configuration is called non-facility associated signaling (NFAS).

ISDN is transmitted over standard T1 and E1 carriers. T1 and E1 trunks are typically four-wire digital transmission links.

Data on a T1 or E1 trunk is transmitted in channels. For primary rate ISDN, a T1 trunk carries 24 channels. An E1 trunk carries 32 channels. With primary rate ISDN, the channels are usually used as follows:

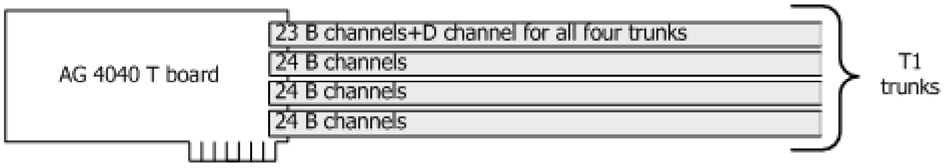
- On a T1 trunk, 23 of the 24 channels carry data: voice, audio, data and/or video signals. These channels are called bearer channels (B channels). On an E1 trunk, 30 of the 32 channels are bearer channels.
- On a T1 or E1 trunk, one channel carries signaling for all B channels. This is called the D channel. On T1 trunks, the D channel is typically carried in channel 24. On E1 trunks, channel 16 is used as the D channel.

The following illustration shows an AG 4040 T (standard configuration):

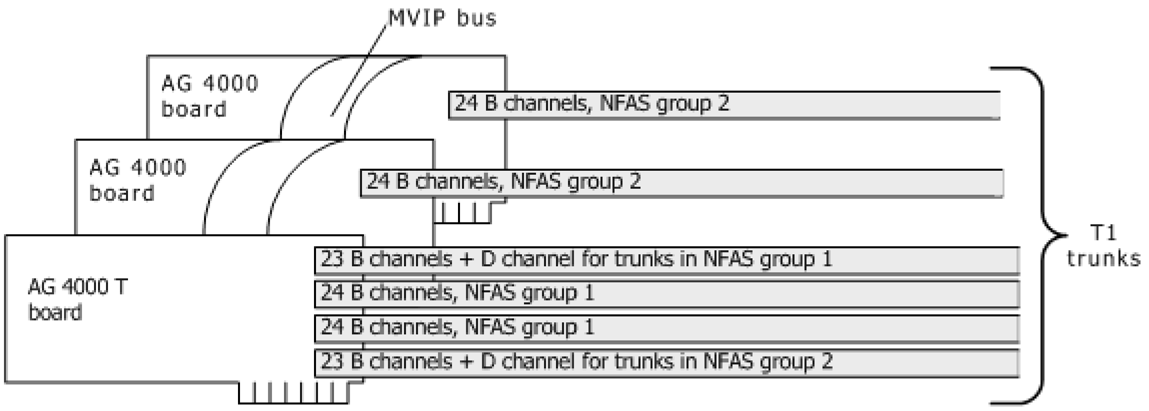

AG 4040 T board
23 B channels+D channel for these channels
23 B channels+D channel for these channels
23 B channels+D channel for these channels
23 B channels+D channel for these channels
T1 trunks

NFAS configurations are supported on T1 trunks only. In an NFAS configuration, trunks are grouped into one or more NFAS groups. One of the trunks in each group has a D channel carrying the signaling for all of the B channels on all of the trunks in the group. This leaves channel 24 free on all other trunks in the NFAS group. This extra channel can be used as another B channel.

The following illustration shows a sample NFAS configuration:



A single NFAS group can contain trunks from multiple boards as shown in the following illustration:


MVIP bus
AG 4000 board
AG 4000 board
AG 4000 T board
24 B channels, NFAS group 2
24 B channels, NFAS group 2
23 B channels + D channel for trunks in NFAS group 1
24 B channels, NFAS group 1
24 B channels, NFAS group 1
23 B channels + D channel for trunks in NFAS group 2
T1 trunks

If the application uses NMS ISDN in a channelized configuration, all trunks in an NFAS group must be on the same board. An NFAS group cannot contain trunks from multiple boards. This restriction does not apply to the ACU configuration or the LAPD configuration.

Setting up and using NFAS

NFAS groups are specified in the board keyword file. In the file, you specify which trunks on which boards belong to which groups. You also specify which trunk in the group will carry the D channel. All other trunks are set to NetworkInterface.T1E1[x].SignalingType=RAW so all 24 channels on these trunks can be used as D channels. For more information, see the *NMS ISDN Installation Manual*.

Your Messaging API application initializes just as described in Initialization tasks. The only difference is that when the application calls **isdnStartProtocol**, the NAI specified in the call can be greater than 3.

The fact that a given trunk is a part of an NFAS group and the D channel resides elsewhere is largely transparent to the application. There are two slight behavioral differences:

- A maximum of four groups on one board can be defined in the configuration file.
- The range of connection IDs available for the D channel is higher. In an NFAS group containing multiple trunks, there will be ACU_MX_CALLS connection IDs for each NAI.

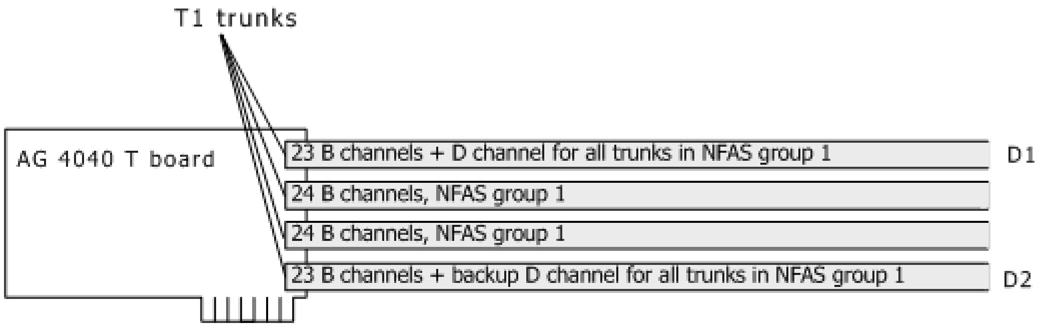
NFAS groups and Hot Swap configurations

If an NFAS group spans multiple boards in a Hot Swap system, and you remove or insert a board while the system is running, the NMS ISDN protocol stacks on the other boards are not affected. If you remove a board containing only B channels, the stack on the D channel board does not detect that the board is missing. It is the application's responsibility to detect this change and take appropriate action (for example, not accessing B channels on that board).

D channel backup

When NFAS is employed, the reliability of the signaling performance for the ISDN interfaces controlled by the D channel can be improved by employing a standby D channel: the D channel backup. The D channel backup feature allows a customer continued access to the ISDN network if one of the D channels fails by transferring most of the signaling information to the backup D channel.

The designated primary D channel (labeled D1) is always present on one trunk. A backup or standby D channel (labeled D2) is present on a different trunk:


T1 trunks
AG 4040 T board
23 B channels + D channel for all trunks in NFAS group 1
D1
24 B channels, NFAS group 1
24 B channels, NFAS group 1
23 B channels + backup D channel for all trunks in NFAS group 1
D2

At any point in time, only one of the D channels, D1 or D2, conveys B channel signaling information. The other D channel remains in a standby role and is active at the LAPD layer (layer 2) only. While the backup D channel is on standby, any layer 3 messages received on it are ignored.

Neither D1 nor D2 can serve as a B channel while designated as a backup D channel. Also, each D1/D2 pair provides signaling only for the set of B channels assigned to it, and cannot backup any other D channel(s) on a different interface.

When both D channels are out of service, D1 has priority as the channel to carry call control signaling. If D1 cannot be established, then D2 is chosen.

Setting up D channel backup

The D channel to use as the backup is specified in the configuration file. For more information, see the *NMS ISDN Installation Manual*.

Note: Both primary and backup D channels must be defined on the same board and belong to the same NFAS group.

In order to preserve active call signaling information in the event of the D channel failure, the application can enable the t309 parameter defined in ISDN_PROTOCOL_PARMS_Q931CC or ISDN_PROTOCOL_PARMS_CHANNELIZED. This parameter enables timer T309 (described in the NSF IE structure).

Handling D channel failure

When the primary D channel fails, B channel signaling information carried by the channel is transferred to the backup D channel. When a transition occurs, most stable calls (those calls in the active or connected states) can be preserved, although message-associated user-to-user information (MA-UUI) and both call-associated and non-call-associated temporary signaling connections may be lost. There is a small interval (controlled by timer T309) after a failure of the LAPD link before the B channels are removed from service.

If the primary D channel fails and timer T309 is enabled (the t309 parameter in the ISDN_PROTOCOL_PARMS_Q931CC structure is set to 1), any calls that are in the connected state at the time of the failure are preserved. Any calls that are initiated but have not entered the connected state are cleared.

Each cleared call receives:

- ACU_CLEAR_CO (ACU stack mode), or
- NCCEVN_CALL_DISCONNECTED (channelized stack mode).

If the primary D channel fails and timer T309 is disabled (the t309 parameter is set to 0), all initiated calls on both terminal and network sides are cleared with ACU_CLEAR_CO or NCCEVN_CALL_DISCONNECTED messages.

After the data link is reestablished on the backup D channel, the application can start placing and receiving new calls. If the data link on the backup D channel cannot be established, the stack keeps trying to establish the link until one of the D channel's connections is reestablished.

A situation may arise when the primary D channel fails, and one side of a call in the connected state initiates the disconnect process (leaves the connected state) immediately before the data link failure is detected in the system. At the moment of failure recognition, one side is in the connected state, while the side that started the disconnect sequence is not in the connected state. In this case, the state of the first side is preserved. The second side receives an ACU_CLEAR_CO or NCCEVN_CALL_DISCONNECTED message. When the data link is reestablished, the first side remains in the connected state. The application disconnects and releases this call.

A similar situation can arise if the primary D channel fails and both sides are in the connected state (and their states are preserved), but then one side initiates the disconnect process and gets cleared before the data link is reestablished. As a result, when the data link is reestablished, one side remains in the connected state and does not know that the other side has disconnected. The application detects this situation, disconnects, and releases the call.

Since neither D1 nor D2 can serve as a B channel while designated as a backup D channel, an application is not allowed to place a call on a timeslot on a primary or backup channel. Both D1 and D2 channels are assigned to the slots number 24 on the corresponding trunks, so that the maximum B channel slot number for a trunk with a primary or a backup D channel is 23. If the application attempts to place a call on a D channel timeslot, the application receives ACU_CLEAR_CO or NCCEVN_CALL_DISCONNECTED with the cause Acu_clear_co_network_cause = f0: ACU_CAUSE_ACU_BAD_ADDRESS.

21. Modifying the NSF IE

NSF IE structure

The network specific facilities information element (NSF IE) indicates which network facilities are being invoked. This information element is supported only for the 4ESS variant.

The following structure defines the NSF IE:

```
struct acu_ext_spf_fac_ie {
     struct u4_acu_ext_hdr hdr;                         /* Extension header */
          pad4
          uchar          net_id[ACU_MX_SZ_NET_ID];  /* Network identification array    */
          uchar          net_id_lgth;               /* Length of network identification*/
          uchar          net_id_type;               /* Type of ID                      */
          uchar          action;                    /* Parameterized/binary indicator  */
          uchar          serv_feature;              /* Service/feature indicator       */
          uchar          facility_coding;           /* Facility coding                 */
          uchar          param_fld;                 /* Parameterized field             */
          pad5
};
```

Use the following macro to access the fields of the data structure:

```
Acu_ext_spf_fac_ie ( field )
```

NSF IE fields

The following tables show the valid settings for the fields of the NSF IE structure. The fields included are:

- net_id_type
- action
- serv_feature
- param_fld

net_id_type

Setting	Value	Description
ACU_USER_SPECIFIED_TYPE	0	Net ID type is user specified.
ACU_NATIONAL_NETWORK_ID_TYPE	2	Use the national network net ID type.

action

Setting	Value	Description
ACU_SPF_FAC_PARAMETER	0	Provided parameters are associated with the facility.
ACU_SPF_FAC_BINARY	1	Use the binary facility.

serv_feature

Setting	Value	Description
ACU_FAC_SERVICE	1	Service
ACU_FAC_FEATURE	0	Feature

Facility Coding

If serv_feature is set to ACU_FAC_SERVICE, then:

Setting	Value	Description
ACU_FAC_CPN_SID_PREFERRED	1	CPN (SID) preferred
ACU_FAC_BN_ANI_PREFERRED	2	BN (ANI) preferred
ACU_FAC_CPN_SID_ONLY	3	CPN (SID) only
ACU_FAC_BN_ANI_ONLY	4	BN (ANI) only
ACU_FAC_CALL_ASSOC_TSC	9	Call associated TSC
ACU_FAC_TSC_CLEAR_RU	10	Notification of call associated TSC clearing or resource unavailable
ACU_FAC_OPERATOR	5	Operator
ACU_FAC_PCCO	6	Pre-subscribed common carrier operator

If serv_feature is set to ACU_FAC_FEATURE, then:

Setting	Value	Description
ACU_FAC_SDN	1	ISDN including GSDN
ACU_FAC_MEGACOM_800	2	Access to MEGACOM 800
ACU_FAC_MEGACOM	3	Access to MEGACOM
ACU_FAC_ACCUNET	6	ACCUNET
ACU_FAC_LONG_DISTANCE	7	International long distance
ACU_FAC_INTERNATIONAL_800	8	International 800
ACU_FAC_ATT_MULTIQUEST	16	AT&T MultiQuest

param_fld

Setting	Value	Description
ACU_FAC_VARI_A_BILL	6	Vari-A-Bill (flexible billing)

The NSF IE structure is treated as an extended data structure. The acu_ext_descr structure is placed in the ACU message primitive. Refer to the Example to learn how to fill in the structures.

Example

The following code shows how to modify the NSF IE using the defined macro:

```
unsigned char *p_ext_data;
char nsf;

if (nsf)
{
     p_ext_data = Acu_conn_rq_a_ext_parms;
     Acu_conn_rq_ext_parms_nb ++;
     Acu_conn_rq_ext_parms_lgth = Acu_ext_spf_fac_ie_size;
     Acu_ext_id   = ACU_EXT_SPF_FAC_IE;
     Acu_ext_lgth = Acu_ext_spf_fac_ie_size;
     Acu_ext_spf_fac_ie(net_id_lgth)     = strlen("288");
     Acu_ext_spf_fac_ie(net_id_type)     = ACU_NATIONAL_NETWORK_ID_TYPE;
     memcpy (Acu_ext_spf_fac_ie (net_id), "288", strlen("288"));
     Acu_ext_spf_fac_ie (action) = ACU_SPF_FAC_BINARY;
     Acu_ext_spf_fac_ie (param_fld) =  0xFF;
     Acu_ext_spf_fac_ie (serv_feature) = ACU_FAC_FEATURE;
     Acu_ext_spf_fac_ie (facility_coding) =  ACU_FAC_BN_ANI_PREFERRED;
}
```

22. Encoding and decoding the Precedence Level IE

Precedence Level IE structure

The precedence level information element (Precedence Level IE) selects a precedence level and reservation in the Digital Subscriber Signaling System No.1 (DSS1). This IE is supported only for the ANSI T1.607 ISDN variant.

The application can encode the Precedence Level IE when placing calls or decode the Precedence Level IE when receiving calls. This enables the application to use the algorithm described in ANSI T1.619 to preempt the calls.

The following structure defines the Precedence Level IE:

```
struct acu_ext_precedence_level
{
    struct u4_acu_ext_hdr hdr;         /* Extension header                     */
    pad4
    uchar          level;              /* precedence level, 0 - 4              */
    uchar          lfb;                /* Look Forward Busy, MLPP_LFB_xxx      */
    uchar          change;             /* change value 0 or 1                  */
    uchar          coding_std;         /* coding STD. 0- ITU-T,2- national     */
    DWORD          domain;             /* bits 0-23 are MLPP service domain    */
    WORD           net_id;             /* network id = 4 decimal digit integer */
    pad2
};
```

Precedence Level IE fields

The following tables show the valid settings for the fields of the Precedence Level IE structure:

- level
- lfb
- change
- coding_std
- domain
- net_id

level

Value	Description
0	FLASH OVERRIDE: Highest precedence level.
1	FLASH
2	IMMEDIATE
3	PRIORITY
4	ROUTINE: Lowest precedence level.

lfb

Value	Description
0	MLPP_LFB_ALLOWED: Look-ahead for busy (LFB) allowed.
1	MLPP_LFB_NOT_ALLOWED: LFB not allowed.
2	IMLPP_LFB_PATH_RESERVED: Path reserved.

change

Value	Description
0	Precedence level coding privilege can be changed at the network boundaries.
1	Precedence level coding privilege cannot be changed at the network boundaries.

coding_std

Value	Description
0	CCITT standardized coding.
1	National standard.

domain

MLPP service domain. 24-bit pure binary expressing the number that uniquely identifies a customer domain across multiple ISDN networks.

net-id

Network identity. Each digit is in binary coded decimal representation from 0 to 9.

Precedence Level IE examples

The following examples show how to encode and decode the Precedence Level IE.

Encoding the Precedence Level IE

```
void *p_data = msg_buffer;
int send_mlpp = 1;

Acu_ext_descr_nb        = 0;
Acu_ext_descr_lgth      = 0;
Acu_ext_descr_offset    = Acu_conn_rq_start_ext_data;
uchar *p_ext_data       = Acu_ext_descr_first_address;

//
// Generate  MLPP Precedence Level IE
//
if ( send_mlpp )
{
    acu_ext_precedence_level *p = ( acu_ext_precedence_level * ) p_ext_data;
    Acu_ext_descr_nb   += 1;
    Acu_ext_descr_lgth += sizeof( acu_ext_precedence_level );
    Acu_ext_lgth        = sizeof( acu_ext_precedence_level );
    Acu_ext_id          = ACU_EXT_PRECEDENCE_LEVEL;
    p_ext_data         += sizeof( acu_ext_precedence_level );

    p->level            = 3;                        // PRIORITY level
    p->lfb              = MLPP_LFB_PATH_RESERVED;   // Look Forward Busy - Path reserved.
    p->change           = 0;      // level privilege may be changed at network boundaries
    p->coding_std       = 0;                        // CCITT standard coding
    p->domain           = 0x123456;                 // Domain is 123456
    p->net_id           = 0x0789;                   // Network identity is 789
}
...
```

Decoding the Precedence Level IE

```
void printExtParameters( void * p_data )
{
    uchar *p_ext_data = Acu_ext_descr_first_address;

    for ( int i = 0; i < Acu_ext_descr_nb; i++ )
    {
        // Process according to parameter type
        switch ( Acu_ext_id )
        {

        //---------------------------------------------------------------------------
        //          MLPP Precedence Level
        //---------------------------------------------------------------------------
        case ACU_EXT_PRECEDENCE_LEVEL:
            {
                acu_ext_precedence_level *p = ( acu_ext_precedence_level * ) p_ext_data;

                printf("\t<EXT>: PRECEDENCE: level=%d, lfb=%d, chg=%d "
                    "NI='%X%X%X%X', domain=0x%06X\n",
                    p->level,
                    p->lfb,
                    p->change,
                    ( p->net_id >> 12 ) & 0x0F,
                    ( p->net_id >> 8  ) & 0x0F,
                    ( p->net_id >> 4  ) & 0x0F,
                    ( p->net_id >> 0  ) & 0x0F,
                    p->domain
                );
            }
            break;

        ...

        default:
            printf("\t<EXT>: UNKNOWN: id=0x%04X\n", Acu_ext_id );
            break;
        }
        p_ext_data += Acu_ext_lgth; // Move to the next parameter
    }
}
```

23. Index