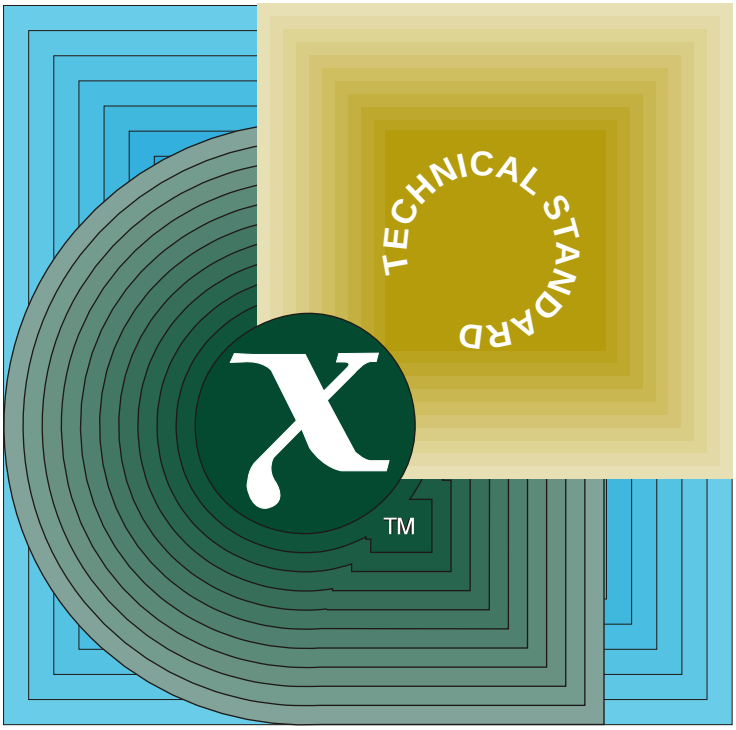
Technical Standard

X.25 Programming Interface using XTI (XX25)



THE Open GROUP

[This page intentionally left blank]

X/Open CAE Specification

X.25 Programming Interface using XTI (XX25)

X/Open Company Ltd.

X/Open CAE Specification

X.25 Programming Interface using XTI (XX25)

ISBN: 1-85912-136-5

X/Open Document Number: C411

Published by X/Open Company Ltd., U.K.

Any comments relating to the material contained in this document may be submitted to X/Open at:

X/Open Company Limited
Apex Plaza
Forbury Road
Reading
Berkshire, RG1 1AX
United Kingdom

or by Electronic Mail to:

XoSpecs@xopen.org

Contents

List of Tables

Preface

X/Open

X/Open is an independent, worldwide, open systems organisation supported by most of the world's largest information systems suppliers, user organisations and software companies. Its mission is to bring to users greater value from computing, through the practical implementation of open systems.

X/Open's strategy for achieving this goal is to combine existing and emerging standards into a comprehensive, integrated, high-value and usable open system environment, called the Common Applications Environment (CAE). This environment covers the standards, above the hardware level, that are needed to support open systems. It provides for portability and interoperability of applications, and so protects investment in existing software while enabling additions and enhancements. It also allows users to move between systems with a minimum of retraining.

X/Open defines this CAE in a set of specifications which include an evolving portfolio of application programming interfaces (APIs) which significantly enhance portability of application programs at the source code level, along with definitions of and references to protocols and protocol profiles which significantly enhance the interoperability of applications and systems.

The X/Open CAE is implemented in real products and recognised by a distinctive trade mark — the X/Open brand — that is licensed by X/Open and may be used on products which have demonstrated their conformance.

X/Open Technical Publications

X/Open publishes a wide range of technical literature, the main part of which is focussed on specification development, but which also includes Guides, Snapshots, Technical Studies, Branding/Testing documents, industry surveys, and business titles.

There are two types of X/Open specification:

- *CAE Specifications*

  CAE (Common Applications Environment) specifications are the stable specifications that form the basis for X/Open-branded products. These specifications are intended to be used widely within the industry for product development and procurement purposes.

  Anyone developing products that implement an X/Open CAE specification can enjoy the benefits of a single, widely supported standard. In addition, they can demonstrate compliance with the majority of X/Open CAE specifications once these specifications are referenced in an X/Open component or profile definition and included in the X/Open branding programme.

  CAE specifications are published as soon as they are developed, not published to coincide with the launch of a particular X/Open brand. By making its specifications available in this way, X/Open makes it possible for conformant products to be developed as soon as is practicable, so enhancing the value of the X/Open brand as a procurement aid to users.

- *Preliminary Specifications*

  These specifications, which often address an emerging area of technology and consequently are not yet supported by multiple sources of stable conformant implementations, are released in a controlled manner for the purpose of validation through implementation of products. A Preliminary specification is not a draft specification. In fact, it is as stable as X/Open can make it, and on publication has gone through the same rigorous X/Open development and review procedures as a CAE specification.

  Preliminary specifications are analogous to the *trial-use* standards issued by formal standards organisations, and product development teams are encouraged to develop products on the basis of them. However, because of the nature of the technology that a Preliminary specification is addressing, it may be untried in multiple independent implementations, and may therefore change before being published as a CAE specification. There is always the intent to progress to a corresponding CAE specification, but the ability to do so depends on consensus among X/Open members. In all cases, any resulting CAE specification is made as upwards-compatible as possible. However, complete upwards-compatibility from the Preliminary to the CAE specification cannot be guaranteed.

In addition, X/Open publishes:

- *Guides*

  These provide information that X/Open believes is useful in the evaluation, procurement, development or management of open systems, particularly those that are X/Open-compliant. X/Open Guides are advisory, not normative, and should not be referenced for purposes of specifying or claiming X/Open conformance.

- *Technical Studies*

  X/Open Technical Studies present results of analyses performed by X/Open on subjects of interest in areas relevant to X/Open's Technical Programme. They are intended to communicate the findings to the outside world and, where appropriate, stimulate discussion and actions by other bodies and the industry in general.

- *Snapshots*

  These provide a mechanism for X/Open to disseminate information on its current direction and thinking, in advance of possible development of a Specification, Guide or Technical Study. The intention is to stimulate industry debate and prototyping, and solicit feedback. A Snapshot represents the interim results of an X/Open technical activity. Although at the time of its publication, there may be an intention to progress the activity towards publication of a Specification, Guide or Technical Study, X/Open is a consensus organisation, and makes no commitment regarding future development and further publication. Similarly, a Snapshot does not represent any commitment by X/Open members to develop any specific products.

Versions and Issues of Specifications

As with all *live* documents, CAE Specifications require revision, in this case as the subject technology develops and to align with emerging associated international standards. X/Open makes a distinction between revised specifications which are fully backward compatible and those which are not:

- a new *Version* indicates that this publication includes all the same (unchanged) definitive information from the previous publication of that title, but also includes extensions or additional information. As such, it *replaces* the previous publication.

- a new *Issue* does include changes to the definitive information contained in the previous publication of that title (and may also include extensions or additional information). As such, X/Open maintains *both* the previous and new issue as current publications.

Corrigenda

Most X/Open publications deal with technology at the leading edge of open systems development. Feedback from implementation experience gained from using these publications occasionally uncovers errors or inconsistencies. Significant errors or recommended solutions to reported problems are communicated by means of Corrigenda.

The reader of this document is advised to check periodically if any Corrigenda apply to this publication. This may be done either by email to the X/Open info-server or by checking the Corrigenda list in the latest X/Open Publications Price List.

To request Corrigenda information by email, send a message to info-server@xopen.co.uk with the following in the Subject line:

```
request corrigenda; topic index
```

This will return the index of publications for which Corrigenda exist.

This Document

This document is an X/Open CAE Specification. It defines an Application Programming Interface (API) to support X.25 working under the established X/Open independent transport-service interface (XTI).

The X.25 networking service is an accepted standard which is in widespread use throughout the world, both as a public tariffed service provided by many national networks, and in private switches as a component in corporate networks. Its pervasive presence in the marketplace, and the consequential huge industry-wide investment in X.25 applications, provides a good business case for supporting X.25 networking through any complementary X/Open APIs. Such is the case with the X/Open XTI API. This XX25 API defines an X.25 service interface that is independent of any X.25 provider. This is achieved by using the existing XTI functions.

Structure

- Chapter 1 on page 1 gives a brief introduction to X.25 and how the X/Open XTI API to any transport provider offers significant benefits to X.25 application writers and users. It also describes the requirements that a conforming implementation of this XX25 API must satisfy. Finally, it indicates possible future development directions that may impact this API.
- Chapter 2 on page 5 outlines the key elements involved in providing a connection-oriented service for X.25.
- Chapter 3 on page 11 indicates that the XX25 API is fully compatible with the states, the incoming and outgoing events and the sequence of function calls defined for XTI.
- Chapter 4 on page 13 describes the specific extensions to existing XTI functions for X.25 requirements.
- Chapter 5 on page 19 identifies the options available in X.25, and explains how these are handled by this XX25 API.
- Appendix A on page 31 presents the additional header file information needed for XX25, to be added to the existing XTI header file information provided in the XTI specification.
- Appendix B on page 35 summarises relevant X.25 protocol information for the convenience of users of this XX25 API.

Intended Audience

This API is aimed at X.25 users who wish to ensure the portability and interoperability of their X.25 applications by making them independent of the X.25 provider. X/Open's XTI API provides an established interface to achieve this independence. Implementors who already know XTI will additionally benefit from their familiarity with the XTI programming environment.

Typographical Conventions

The following typographical conventions are used throughout this document:

- **Bold** font is used in text for options to commands, filenames, keywords, type names, data structures and their members, language-independent names, and symbols that are specific to the XX25 API.
- *Italic* strings are used for emphasis or to identify the first instance of a word requiring definition. Italics in text also denote:
  - — function parameters, command operands, command option-arguments or variable names, for example, substitutable argument prototypes
  - — environment variables, which are also shown in capitals
  - — utility names
  - — external variables, such as *errno*
  - — functions; these are shown as follows: *name*( ). Names without parentheses are C external variables, C function family names, utility names, command operands or command option-arguments.
- Normal font is used for the names of constants and literals.
- The notation **<file.h>** indicates a header file.
- Names surrounded by braces, for example, {ARG_MAX}, represent symbolic limits or configuration values which may be declared in appropriate headers by means of the C **#define** construct.
- The notation [ABCD] is used to identify a return value ABCD, including if this is an error value.
- Syntax, code examples and user input in interactive examples are shown in `fixed width` font. Brackets shown in this font, `[ ]`, are part of the syntax and do *not* indicate optional items. In syntax the `|` symbol is used to separate alternatives, and ellipses (`. . .`) are used to show that additional arguments are optional.

Trade Marks

X/Open® is a registered trade mark, and the ‘‘X’’ device is a trade mark, of X/Open Company Limited.

Referenced Documents

The following documents are referenced in this Preliminary Specification:

ISO / IEC 8208

ISO / IEC 8208:1990(E): Information Technology — Data Communications — X.25 Packet Layer Protocol for Data Terminal Equipment.

ISO / IEC 8348

ISO / IEC 8348:1987(F): Information Processing Systems — Data Communications — Network Service Definition.

ITU-T X.3

ITU-T, 1993, Data Communication Networks: Services, Facilities and Interfaces, Series X Recommendations (X.1 to X.32), Recommendation X.3 — Packet Assembly Disassembly Facility (PAD) in a Public Data Network.

ITU-T X.25

ITU-T, 1993, Data Communication Networks: Services, Facilities and Interfaces, Series X Recommendations (X.1 to X.32), Recommendation X.25 — Interface Between Data Terminal Equipment (DTE) and Data Circuit-terminating Equipment (DCE) for Terminals Operating in the Packet Mode and Connected to Public Data Networks by Dedicated Circuit.

ITU-T X.28

ITU-T, 1993, Data Communication Networks: Services, Facilities and Interfaces, Series X Recommendations (X.1 to X.32), Recommendation X.28 — DTE / DCE Interface for a Start-Stop Mode Data Terminal Equipment Accessing the Packet Assembly / Disassembly Facility (PAD) in a Public Data Network Situated in the Same Country.

ITU-T X.29

ITU-T, 1993, Data Communication Networks: Services, Facilities and Interfaces, Series X Recommendations (X.1 to X.32), Recommendation X.29 — Procedures for the Exchange of Control Information and User Data Between a Packet Assembly / Disassembly (PAD) Facility and a Packet Mode DTE or Another PAD.

XNS (Networking Services, Issue 4)

X / Open CAE Specification, August 1994, Networking Services, Issue 4 (ISBN: 1-85912-049-0, C438).

The XNS Specification includes the XTI specification.

Chapter 1

Introduction

1.1 Scope of the Specification

The purpose of the XX25 API is to provide the X.25 Network Service under the X/Open Transport Interface.

Well known and understood around the world, X.25 is an accepted standard, provided by many national networks as a tariffed service, and is readily available in private switches for building corporate networks.

X.25 is ideally suited for users requiring access on a global scale to computing services, or even on a national scale to far-flung locations. The protocol is reliable and takes responsibility for networks integrity, always ensuring that transmitted packets are received correctly at their destination address. The service also applies flow control and buffering mechanisms, acting as a store-and-forward for packets.

Since X.25 is a readily available service spanning the globe, there will always be applications that simply cannot be handled cost-effectively by any other means. In addition, many companies already have investments in private X.25 networks, and it therefore makes sense to continue to use the X.25 fabric for the organisation's networking requirements.

Examples of applications which may be interested in the extension of XTI for X.25 include:

- terminal and host PAD applications based on ITU-T Recommendation X.3, ITU-T Recommendation X.28 and ITU-T Recommendation X.29
- VIDEOTEX — referring to the ETS 300 080 document from ETSI (ISDN lower-layer protocols for telematics terminals), the VIDEOTEX lower layers have the OSI transport layer not applicable, so the application directly accesses the X.25 packet service.

The functions offer full access to all X.25 level 3 services (for example, Q bit, negotiation of the facilities) for applications such as videotext servers or other servers with direct X.25 access.

The basic X.25 services (connection establishment and release, data transfer and reset mechanism) are provided either by the existing XTI functions or by a compatible extension of these functions.

This XX25 API provides for portability and interoperability of X.25 applications, so encouraging the development of X.25 applications in open systems. It defines an X.25 service interface that is independent of any underlying X.25 provider. Programmers with prior knowledge of XTI benefit further from familiarity with the XTI programming environment.

1.2 Relationship to the XTI Specification

XX25 provides access to X.25 providers through the XTI interface.

This document details the X.25-specific options and X.25-specific behaviours of the XTI interface when used to access an X.25 provider. The XX25 specification has been designed in such a way as to allow existing generic XTI applications to run unchanged over X.25.

This specification must be used in conjunction with the **XTI** specification.

1.3 Modes of the X.25 Service

The X.25 service interface supports a connection-oriented service with two modes:

- the switched-connection mode
- the permanent-connection mode.

The switched-connection mode enables data to be transferred over an established connection (called a Switched Virtual Circuit (SVC)) in a flow-controlled, sequenced manner. This circuit-oriented mode is attractive to applications that require relatively long-lived, datastream-oriented interactions.

The permanent-connection mode has the same features as the switched-connection mode for the transfer of data. This service requires a pre-existing association (called a Permanent Virtual Circuit (PVC)) between the peer users involved, which determines the characteristics of the data to be transmitted. No dynamic negotiation of options is supported by this mode.

A single X.25 endpoint does not support different modes of connections simultaneously.

The use of the interface - that is, the functions called and the order in which they are called - is identical for SVCs or for PVCs.

1.4 Terminology

Definition of Terms

The terminology used in this specification is that of the ISO standards which define the X.25 service to which XX25 provides access. For convenience, the abbreviations have been defined in the Glossary for this specification.

In addition, terms particular to the ISO X.25 protocol are described in Appendix B.

Use of Naming Prefixes

In order to preserve uniqueness of the additions to XTI support the X.25 service, the constants and the flags defined by this specification have names that have the format **T_X25_xxx.**

The new definitions are presented in the XX25 header file summary, see Appendix A.

1.5 Conformance Requirements

The XX25 conformance has two facets:

- the XX25 API
- the underlying X.25 Service Provider.

1.5.1 API Implementation

An XX25 conformant implementation provides the functions defined in Chapter 4.

The following points should be noted:

- When the XX25 user requests a feature which is supported by the XX25 API but not supported by the underlying X.25 Service provider, the XX25 API returns a [TNOTSUPPORT] error.
- An implementation of the XX25 API must only return the error codes defined by this specification and the **XTI** specification.

1.5.2 Underlying X.25 Service Provider

An implementation of the underlying X.25 Service Provider which complies with this XX25 specification will also comply with the requirements specified in ISO/IEC 8208 or ITU-T Recommendation X.25. The version supported by the provider (X.25-1980, X.25-1984 X.25-1988, X.25-1993, and so on) is indicated in an XX25 option (named **T_X25_VERSION**), which is defined in Chapter 5.

1.6 Future Directions

Appendix for Security

X.25-specific interworking security issues will be addressed in the XTI security appendix.

Extensions to use XX25 over ISDN Networks

ETSI has defined in the specification ETS 300 325 (Programming Communication Interface (PCI) for Euro-ISDN) a programming interface to provide Programming Users Facilities (PUFs) with interfaces to ISDN Network Access Facilities. XX25 could be a candidate as an X.25 user plane PUF, when X.25 is used as a protocol over ISDN networks. However, some extensions will be necessary to manage the control plane for signalling and the administration plane for resource management, which are specific to ISDN networks.

Chapter 2

Overview of the Connection-Oriented Service

The connection-oriented service for X.25 consists of five phases of communication. Like the transport service, it includes:

- Initialisation / De-initialisation
- Connection Establishment
- Data Transfer
- Connection Release.

In addition, it includes:

- Reset.

Notes:

1. The underlying X.25 provider manages NSDUs (Network Service Data Units) in order to carry data packets, whereas a transport provider deals with TSDUs (Transport Service Data Units). In the following sections, we consider NSDU when the word TSDU is used.
2. **Bold** font is used in text for symbols that are XX25 API-specific.

2.1 Overview of Initialisation/De-initialisation

There are no special considerations involved in the local management of an X.25 endpoint.

When the packet layer is restarted, or reset, all open user XX25 connections (including PVCs) are given a disconnect indication.

2.2 Overview of Connection Establishment

For an SVC, the connection establishment functions map onto the X.25 call setup procedures, so that *t_connect*( ) and *t_accept*( ) cause call request and call accepted packets to be generated, and incoming call and call confirmed packets cause **T_LISTEN** and **T_CONNECT** events to be generated.

There is no equivalent protocol exchange on PVCs. Function *t_connect*( ) causes a **T_CONNECT** event to be generated immediately, if the PVC is operable, and a **T_DISCONNECT** event to be generated immediately if it is inoperable. When listening, a data or interrupt packet arriving from a remote DTE on an idle PVC causes the packet to be queued and a **T_LISTEN** event to be generated. The user can invoke *t_accept*( ) as usual, after which a **T_DATA** or **T_EXDATA** event is generated for the queued packet.

2.3 Overview of Data Transfer

Once an X.25 connection has been established between two users, data may be transferred back and forth over the connection in full duplex mode.

Two existing XTI functions support data transfer in connection-oriented service:

- *t_snd*( )
- *t_rcv*( ).

These two functions enable X.25 users to send or receive over an X.25 connection:

- normal data
- expedited data (interrupt packets)
- qualified (Q-bit) data
- data with the Delivery Confirmation bit (D-bit) set
- explicit acknowledgements of D-bit data (see Section 2.3.3 on page 7)
- explicit acknowledgements of Expedited data (see Section 2.3.3 on page 7)
- connection resets.

2.3.1 Receiving Data

X.25 users are able to receive:

- normal data
- expedited data (if the **T_EXPEDITED** flag is set on return from the call)
- data sent with Delivery Confirmation bit (if the **T_X25_D** flag is set)
- qualified data (if the **T_X25_Q** flag is set)
- an acknowledgement of data previously sent with the D bit (if the **T_X25_DACK** flag is set)
- an acknowledgement of expedited data (if the **T_X25_EACK** flag is set)
- a reset indication (if the **T_X25_RST** flag is set).

Each element of this list generates an event that unblocks a synchronous call.

If a reset indication or an explicit acknowledgement arrives during receipt of a data TSDU and some data has already been transferred to the users buffer, then *t_rcv*( ) returns, indicating the number of bytes of data in the users buffer and with **T_MORE** set in flags. The reset indication or explicit acknowledgement is returned by the next *t_rcv*( ) call.

In order to allow unmodified programs to run over XX25, reset indications and D-bit notifications are only given to the application if the appropriate option has been negotiated on. By default, D-bit acknowledgement is silently handled by the provider (**T_X25_DACK** option defaults to **T_NO**) and receipt of a reset causes the connection to be released (**T_X25_RST_OPT** option defaults to **T_NO**).

2.3.2 Sending Data

X.25 users are able to send over an X.25 connection:

- normal data
- expedited data (by setting the **T_EXPEDITED** flag)
- normal data sent with the Delivery Confirmation bit (by setting the **T_X25_D** flag)
- qualified data (by setting the **T_X25_Q** flag)
- an explicit acknowledgement of data previously sent with the D bit (by setting the **T_X25_DACK** flag)
- an explicit acknowledgement of expedited data (by setting the **T_X25_EACK** flag)
- a reset request or a confirmation (by setting the **T_X25_RST** flag).

Note: Setting both the **T_EXPEDITED** flag and the **T_X25_D** flag results in a [TBADFLAG] error.

2.3.3 Acknowledgements of Data (D bit and Expedited Data)

There are two modes for acknowledging receipt of data with the D bit and expedited data:

- implicit acknowledgement
- explicit acknowledgement.

In implicit mode (the default), the X.25 service provider automatically generates an acknowledgement when it passes the data to the user. In this mode the user is not notified that the D-bit is set.

In explicit mode, the XX25 user specifically acknowledges the data by calling *t_snd*( ) with one or more of the following flags:

- **T_X25_DACK** to acknowledge data with the D bit
- **T_X25_EACK** to acknowledge expedited data.

Note: When used in this manner to acknowledge received data, a *t_snd*( ) call can not contain any user data and can not have any other flags set.

Explicit acknowledgement of D-bit data is selected by setting **T_X25_DACK** option to **T_YES**. Explicit acknowledgement of expedited data is selected by setting **T_X25_EACK** option to **T_YES**.

2.3.4 Data with the D bit

Note: As the D-bit procedures are not supported by all networks, their use may impact application portability.

The level of support for the D-bit procedures by the X.25 provider is indicated in the **T_X25_D_OPT**.

- When **T_X25_D_OPT** is set to **T_NO**, an attempt to send data with the D-bit set generates a [TBADDATA] error; receipt of a data packet with the D-bit set will cause the X.25 provider to reset the connection (as specified in Section 6.3 of ISO / IEC 8208).

- When **T_X25_D_OPT** is set to **T_YES**, the D-bit can be set on sent data packets and receipt of data packets with the D-bit set is permitted. The user may select either implicit acknowledgement or notification and explicit acknowledgement by setting the **T_X25_USER_DACK** option as described above.

The **T_X25_D_OPT** option is a read-only option.

The optional D-bit negotiation mechanism (as described in Section 6.3 of ISO/IEC 8208) can be invoked at connection establishment time by use of the **T_X25_CONN_DBIT**. However, as this mechanism is optional, its failure does not necessarily mean that the D-bit procedures cannot be used — it may be that the remote provider does not implement the mechanism. Where an X.25 provider allows access to D-bit procedures, the user can issue *t_snd*( ) calls with the **T_X25_D** flag, regardless of the outcome of the D-bit negotiation at connection establishment.

2.3.5 Connection Resets

A reset may be invoked from the data transfer phase. During this phase, either the user or the network may reinitialise a virtual circuit at any time.

The reset is supported by using an additional flag **T_X25_RST** in the functions *t_snd*( ) and *t_rcv*( ).

1. Sending a reset

   A reset request is sent by setting the **T_X25_RST** flag in a *t_snd*( ) call with the cause and the diagnostic of the reset in the two first octets of the *buf* argument. The cause is in the first octet and the diagnostic in the second octet.

   The *t_snd*( ) function returns immediately. If further *t_snd*( ) calls are accepted while the reset is being performed, the send data will remain pending until the X.25 provider receives the confirmation of reset. This confirmation of reset is not returned to the user. The normal flow control mechanism may result in a subsequent *t_snd*( ) in synchronous mode blocking, or a *t_snd*( ) call in asynchronous mode returning the [TFLOW] error.

2. Receiving a reset

   When a reset indication is received and the option T_X25_RST_OPT is set to T_YES, the X.25 provider notifies the user with a **T_DATA** event. The user consumes this event with the *t_rcv*( ) call. On return from *t_rcv*( ), **T_X25_RST** is set in the *flags* field, and the *buf* argument contains the cause of the reset in the first octet and the diagnostic in the second octet. If the users buffer is less than two bytes long then the diagnostic value is discarded, and if the length is zero the cause is also discarded. All further data presented on *t_snd*( ) calls is discarded by the X.25 provider until the user acknowledges the reset by issuing a *t_snd*( ) call with the **T_X25_RST** flag set. Any cause and diagnostic passed with the *t_snd*( ) call is ignored by the X.25 provider. If the X.25 user attempts to send data while a reset indication is pending, the *t_snd*( ) call returns with a [TLOOK] error. A subsequent *t_look*( ) call will return a **T_DATA** event. The user must consume the event before sending further data.

Notes:

1. Data sent prior to a reset may be discarded. Data sent after a reset will be delivered to the peer entity after it has been notified of the reset. Data received after a reset will have been sent by the peer entity after the reset. No explicit acknowledgements will be received after a reset for data sent prior to the reset.
2. Reset collisions are managed by the X.25 provider and have no effect for the user.

2.4 Overview of Connection Release

The X.25 switched-connection mode only supports the abortive release.

The functions that support connection release in all cases are already defined:

- *t_snddis*( ), to send a request of a connection release
- *t_rcvdis*( ), to receive an indication of a connection release.

Outstanding sent and received data may be discarded.

In addition, the user can indicate the reason of the connection release (the cause and the diagnostic) by setting the **T_X25_DISCON_REASON** option by calling *t_optmgmt*( ) prior to calling *t_snddis*( ).

Optionally, after receiving an indication of a connection release, the address of the user that released the connection can be retrieved in the **T_X25_DISCON_ADD** option, by calling the function *t_optmgmt*( ). In the same way, the X.25 facilities associated with the connection release can be retrieved in specific XTI options defined to support the X.25 facilities (see Chapter 5). The information may be overwritten if an incoming call is queued on the endpoint.

When *t_snddis*( ) is invoked on a PVC, the circuit is reset with the cause and diagnostic from **T_X25_DISCON_REASON**, unless **T_X25_DISCON_REASON** is set to **T_UNSPEC**, in which case the PVC is not reset. The default value of **T_X25_DISCON_REASON** for a PVC gives a cause of 0x80 and a diagnostic of 0xf1. These values may be changed by using *t_optmgmt*( ).

Note: The **T_X25_DISCON_REASON** option is not permitted to take the value **T_UNSPEC** on an endpoint representing an SVC.

Receipt of a reset indication with certain cause and diagnostic values generates a disconnect indication. These include the reset causes 0x01 (out of order), 0x11 (incompatible destination) and 0x1d (network out of order), their private network equivalents (0x81, 0x91 and 0x9d) and the cause/diagnostic pair 0x80/0xf1 which are generated by *t_snddis*( ).

For further details of the functions referred to in this chapter, see their descriptions in Chapter 4.

Chapter 3

States and Events

To avoid major changes to the XTI state definitions and state tables, resets are handled within the data transfer state **T_DATAXFER**.

The only difference is that a reset indication causes a *t_snd*( ) call to return with a [TLOOK] error. The event is reported as a **T_DATA** event by a subsequent call to *t_look*( ).

Chapter 4

Functions

This chapter describes extensions to the existing XTI functions (as defined in the **XTI** specification), required for XTI support of X.25.

The relevant description for X.25 use of existing XTI functions are presented below in alphabetical order.

To ensure clarity of presentation, the options that contain X.25 facilities are specified separately, in Chapter 5.

t_accept( ) No special consideration.

t_bind( ) The address field of the *t_bind*( ) structure contains the matching requirements for routing incoming calls to the endpoint. This may include (but is not limited to) representations of one or more of the following:

- a local SNPA identifier
- a local X.25 address
- a local X.25 subaddress
- a local NSAP
- a call user data matching requirement
- a PVC number.

Where an incoming call can be routed to multiple endpoints on the basis of their matching requirements, the actual endpoint selected will be implementation dependent.

If the application likes to initiate a connection, it can either bind itself to a NULL address (by setting *req* to NULL or *req→addr.len* to zero) or use any of the matching requirements defined above. If a NULL address is used, the application is free to use both SVCs and PVCs and any X.25 line. If matching requirements have been defined, connections may be restricted to SVCs, a certain PVC, or a certain X.25 line, depending on the matching criteria.

It is not possible to receive connection indications on a NULL address.

Note: An implementation may choose to provide support for a wildcard mechanism for address information, for example to route incoming calls whose call user data starts with a particular pattern.

t_connect( ) The *sndcall→addr* is used to select either an SVC or a PVC.

For an SVC the *sndcall→addr* structure contains a representation of the addressing information necessary to reach the destination, it may contain (but is not limited to) one or more of the following:

- SNPA identifier
- destination X.25 address
- destination NSAP.

When the connection has been established, the *rcvcall→addr* structure represents the address on which the call has been accepted.

For a PVC, the *sndcall→addr* structure represents the PVC to be used. If it is already in use, the error [TADDRBUSY] is returned. On successful return:

- In synchronous mode, the PVC will be in state **T_DATAXFER**.
- In asynchronous mode, the PVC will be in state **T_OUTCON** and a **T_CONNECT** event will be outstanding.

When the connection has been established, the *rcvcall→addr* structure represents the actual PVC allocated.

t_getinfo( )

The information returned by *t_getinfo*( ) reflects the characteristics of the X.25 connection or, if no connection is established, the maximum characteristics an X.25 connection could take on using the underlying X.25 provider.

The parameters of the *t_getinfo*( ) function, for the different versions of the X.25 protocol (X.25-1980, X.25-1984, X.25-1988, X.25-1993, and so on) are presented in the table below.

Parameters	**Before Call**	**After Call**	
		X.25-1988 X.25-1984 X.25-1993	X.25-1980
fd	x	/	/
info→addr		x	x
info→options	/	x	x
info→tsdu	/	x (1)	x (1)
info→etsdu	/	–2 / 32 (2)	–2 / 1 (3)
info→connect	/	16 / 128 (4)	16 / 128 (4)
info→discon	/	0 / 128 (5)	0 / 128 (5)
info→servtype	/	T_COTS	T_COTS
info→flags	/	T_SENDZERO	T_SENDZERO

1. –1 or an integral number greater than zero.
2. –2 if no expedited data transfer can be exchanged, and 32 otherwise.
3. –2 if no expedited data transfer can be exchanged, and 1 otherwise.
4. 16 in basic format or 128 in extended format (if the X.25 facility *Fast Select* has been negotiated).
5. 0 in basic format or 128 in extended format (if the X.25 facility *Fast Select* has been negotiated).

t_look( )

No special consideration.

t_listen( )

No special consideration.

t_open( )

The function *t_open*( ) is called at the first step in the initialisation of an X.25 endpoint. This function returns various default characteristics associated with the different versions of X.25 that are supported. If, for example an X.25 provider supports X.25-1984 and X.25-1988, the characteristics returned are those of X.25-1988. If the X.25 provider is limited to X.25-1980, the characteristics returned are those of X.25-1980.

The parameters of the *t_open*( ) function, for the different versions of the X.25 protocol (X.25-1980, X.25-1984, X.25-1988, X.25-1993, and so on) are presented in the table below.

Parameters	**Before Call**	**After Call**	
		X.25-1988 X.25-1984 X.25-1993	X.25-1980
fd	x	/	/
info→addr		x	x
info→options	/	x	x
info→tsdu	/	x (1)	x (1)
info→etsdu	/	32	1
info→connect	/	128	128
info→discon	/	128	128
info→servtype	/	T_COTS	T_COTS
info→flags	/	T_SENDZERO	T_SENDZERO

(1) –1 or an integral number greater than zero.

t_optmgmt( ) The function *t_optmgmt*( ) uses specific options to support the X.25 service. The options are described in Chapter 5 on page 19.

t_rcv( ) The behaviour of the function *t_rcv*( ) remains unchanged. The function can operate in synchronous and asynchronous modes. It follows the current flow control rules.

The default behaviour is to acknowledge, in an automatic way, data sent with the Delivery Confirmation bit and expedited data.

The optional explicit acknowledgement is selected in the functions *t_optmgmt*( ), *t_connect*( ), *t_accept*( ) either with the **T_X25_USER_DACK** option, for the acknowledgement of data sent with the D bit, or with the **T_X25_USER_EACK** option, for the acknowledgement of expedited data.

If expedited data arrives after part of a TSDU has been retrieved, receipt of the remainder of the TSDU is suspended until the ETSDU has been processed. Only after the full ETSDU has been retrieved (the **T_MORE** flag not set), the remainder of the TSDU is made available to the user.

In addition to the **T_EXPEDITED** and **T_MORE** flags, the following flags can be set in the argument *flags:*

- On return from the call, if **T_X25_D** is set in *flags*, this indicates that the data returned was sent with the D bit, and the **T_X25_USER_DACK** option is set. This data has to be acknowledged explicitly by the receiver.
- On return from the call, if **T_X25_DACK** is set in *flags*, data previously sent with the D bit has been acknowledged.
- On return from the call, if **T_X25_EACK** is set in *flags*, the previously sent expedited data has been acknowledged.

  Note: If either **T_X25_DACK** or **T_X25_EACK** is set in *flags*, then no other flags are set and no user data is returned to the user.
- On return from the call, if **T_X25_Q** is set in *flags*, the data returned are qualified.

- On return from the call, if **T_X25_RST** is set in *flags*, this indicates that a reset indication occurred.

When **T_X25_RST** is returned, the argument *buf* contains the cause and diagnostic of the reset. Each one is coded into one octet. The cause is encoded in the first octet, and the diagnostic in the second octet. If the user's buffer is less than two bytes long then the diagnostic value is discarded, and if the length is zero the cause is also discarded.

t_rcvconnect( ) No special consideration.

t_rcvdis( ) This function is used to retrieve an indication of a connection release.

The field *discon→reason* contains the X.25 cause and diagnostic of the connection release. The cause and the diagnostic are both encoded in an octet and can be retrieved by using respectively the **T_X25_GET_CAUSE** macro and the **T_X25_GET_DIAG** macro.

This function allows operations in accordance with XTI, but cannot be used to retrieve charging information or the address of the user that released the connection. For these purposes, the user has to call the function *t_optmgmt*( ) and retrieve the meaningful options.

For further details about the management of options, see Chapter 5.

t_snd( ) The behaviour of the function *t_snd*( ) remains unchanged. The function can operate in synchronous and asynchronous modes.

In addition to the **T_EXPEDITED** and **T_MORE** flags, the following flags can be set in the argument *flags:*

- **T_X25_D**
  If set in *flags*, the data is sent with the D bit set. This data has to be acknowledged by the peer.

  As with normal *t_snd*( ) requests, the user may issue multiple *t_snd*( ) requests with the D-bit set which will be queued by the provider. A separate acknowledgement is generated for each one. The normal flow control mechanism applies: if a *t_snd*( ) cannot be accepted, the [TFLOW] code is returned.

  Note: As a D-bit send requires end-to-end acknowledgement, it can considerably delay the transmission of further packets.

- **T_X25_Q**
  If set in *flags*, the data is sent as normal qualified data.

- **T_X25_RST**
  If set in *flags*, this indicates to the underlying provider that a request or a confirmation of reset is required.

  The *t_snd*( ) function returns immediately. If further *t_snd*( ) calls are accepted while the reset request is being performed the send data will remain pending until the X.25 provider receives the confirmation of reset. This confirmation of reset is not returned to the user. The normal flow control mechanism may result in a subsequent *t_snd*( ) in synchronous mode blocking, or a *t_snd*( ) call in asynchronous mode returning the [TFLOW] error.

The cause and diagnostic of a reset request are encoded in the two first octets of the *buf* argument. The cause is in the first octet and the diagnostic in the second octet. If the *buf* argument is NULL or the *nbytes* is 0, then a cause of 0 and a diagnostic of 0xFA (that means user resynchronisation) are used. If *nbytes* is 1, the diagnostic is set to 0.

Any cause and diagnostic passed in the *t_snd*( ) call are ignored by the X.25 provider when sending a reset confirmation.

Data received after a successful *t_snd*( ) call requesting a reset is data transmitted by the peer after completion of the reset.

- **T_X25_DACK**
  If set in *flags*, this indicates that an explicit acknowledgement of data with the D bit is sent.
- **T_X25_EACK**
  If set in *flags*, this indicates that an explicit acknowledgement of expedited data is sent.

Note: When either the **T_X25_DACK** or the **T_X25_EACK** flag is set, no other flags can be set, and there must be no user data present on the *t_snd*( ) call.

If T_X25_DACK or T_X25_EACK is set, although no (expedited) data need be acknowledged, the *t_snd*( ) call either fails with t_errno set to [TBADDATA], or a subsequent call fails with t_errno set to [TSYSERR] or [TPROTO].

t_snddis( ) The function is used to send a request of a connection release.

The function induces a state transfer to **T_IDLE** and returns at the receipt of the confirmation of the connection release.

In case of PVC-connection mode, *t_snddis*( ) dissociates the user from the PVC and normally resets the PVC.

Chapter 5

Options

The functions *t_optmgmt*( ), *t_connect*( ), *t_listen*( ), *t_accept*( ), and *t_rcvconnect*( ), all contain an *opt* argument of type **struct netbuf** as an input or output parameter. This argument is used to convey options, and in particular X.25 facilities, between the X.25 user and the X.25 provider.

Each option is formatted according to the structure *t_opthdr*, possibly followed by the option value.

In the structure *t_opthdr*, the *name* field specifies the mnemonic of the option.

The *level* field specifies the protocol affected (here **T_X25_NP** for all options of the XX25 API).

The *len* field specifies the total length of the option (that is, the length of the option header *t_opthdr* plus the length of the option value, without the possible alignment characters).

The *status* field of the returned options contains information about the success or failure of a negotiation.

The structure *t_opthdr* is followed by the option value.

The XX25 user has to ensure that each option starts at a long-word boundary.

5.1 Description of the XX25 Options

The table presented in this section, describes the options available in the XX25 API. An XTI implementation supports none, all or any subset of them.

All the names of the options are defined in the ‘‘New Options to Support X.25 Service’’ part of the **<xti.h>** header file.

While the ITU-T X.25 standard mentions both *normal* and *extended* facilities, the facilities described below combine both into one facility which allows values into the *extended* limits. The provider will resolve the issue of which facility (*normal* or *extended*) to use, transparently to the user.

The **legal option values** shown in the X.25 facilities table are those limits defined in X.25 1993. They are given as an example only.

The actual *legal option values* in force for X.25 facilities are dependent on the version of the X.25 protocol in use and the network subscription options in force on a particular SNPA.

For each facility, if the user gives a value that is not one of the allowed values, the provider will negotiate it to a valid value.

The normal XTI option handling is employed for XX25. Chapter 5 of the **XTI** specification describes this in detail.

XX25 options relevant only to a version of the X.25 protocol specification later than that in use are treated as ‘‘unknown’’ or ‘‘not supported’’ by the XX25 provider.

Negotiable numeric XX25 options which contain a value outside those defined for the particular version of X.25 in use are *not* considered to be ‘‘illegal options’’, but are negotiated to the nearest legal option value by the XX25 provider. (This allows for forward compatibility with newer versions of X.25.)

Other XX25 options are considered to be illegal if they contain a value which cannot be represented in the protocol or which cannot be interpreted by the provider.

Note: Examples of the option values which cannot be represented are a CUG number greater than that allowed for basic format when X.25-1980 is in use, and a CUG number greater than that allowed for extended format when X.25-1984 is in use. An example of an option value which cannot be interpreted by the provider is T_X25_REVCHG being set to a value other than T_YES or T_NO.

The handling of ‘‘unknown’’, ‘‘not supported’’ and ‘‘illegal’’ options is clearly described in Section 5.3.2, Illegal Options of the **XTI** specification.

X.25 facilities are conveyed by the options described under ‘‘X.25 facilities’’ in the following table.

The option values associated with the endpoint are used by the X.25 provider to build a call packet. Before user modification, the effective option values are the implementation defined default values (for example a combination of subscription time options and local provider configuration data). The effective values can be modified at any time by the functions *t_optmgmt*( ), *t_connect*( ), *t_listen*( ), *t_accept*( ) and *t_rcvconnect*( ). Function *t_optmgmt*( ) can be used to retrieve the implementation defined default values or to reset the effective values to these defaults. If an option value is set to **T_UNSPEC**, the associated X.25 facility will not be specified in the call packet.

Options which are not user-settable are marked as *read-only* in the table below.

Table 5-1 X25_NP-level Options

Option Name	Type of Option Value	Legal Option Value	Meaning
T_X25_RST_OPT	unsigned long	T_YES / T_NO The default value is T_NO.	User supports resets.
T_X25_D_OPT **read-only**	unsigned long	T_YES / T_NO The value is implementation defined.	Support of the D bit.
T_X25_USER_DACK	unsigned long	T_YES / T_NO The default value is T_NO.	Explicit acknowledgement of data with delivery bit.
T_X25_USER_EACK	unsigned long	T_YES / T_NO The default value is T_NO.	Explicit acknowledgement of expedited data.
T_X25_VERSION **read-only**	unsigned long	T_X25_yyyy with ‘‘yyyy’’ representing the year of the X.25 Recommendation. The value is implementation defined.	Version of the ITU-T Recommendation X.25 or of ISO / IEC 8208 supported by the provider (X.25-1980, X.25-1984, X.25-1988, and so on).
T_X25_DISCON_REASON	unsigned long	See meaning. The default value is 0xf1.	Reason for a connection release that includes the cause and the diagnostic. This reason can be encoded by using the **T_X25_SET_CAUSE_DIAG** macro.
T_X25_DISCON_ADD **read-only**	struct t_x25facaddr	See text.	Address of the user that released the connection.
T_X25_CONN_DBIT	unsigned long	T_YES / T_NO The default value is T_NO.	Setting of the D-bit during the connection phase in order to negotiate the support of the D-bit during data transfer.

X.25 Facilities

Legal option values are those specified in the ITU-T Recommendation X.25 or ISO / IEC 8208 (see **T_X25_VERSION**).

Option Name	Type of Option Value	Legal Option Value	Meaning
T_X25_PKTSIZE	struct t_x25facval	Size in octets from 16 to 4096, **T_UNSPEC**. See text.	Packet Size
T_X25_WINDOWSIZE	struct t_x25facval	Size from 1 to 7 or from 1 to 127 (in extended format), **T_UNSPEC**. See text.	Window Size
T_X25_TCN	struct t_x25facval	Throughput in bits / s from 75 to 192000, **T_UNSPEC**. See text.	Throughput Class Negotiation
T_X25_CUG	unsigned long	Index from 0 to 9999, **T_UNSPEC**.	CUG (Closed User Group)
T_X25_CUGOUT	unsigned long	Index from 0 to 9999, **T_UNSPEC**.	CUG with Outgoing Access

Option Name	Type of Option Value	Legal Option Value	Meaning
T_X25_BCUG	unsigned long	Index from 0 to 9999, **T_UNSPEC**.	Bilateral CUG
T_X25_FASTSELECT	unsigned long	See text.	Fast Select
T_X25_REVCHG	unsigned long	T_YES / T_NO	Reverse Charging
T_X25_NUI	string	Identifier. See text.	NUI (Network User Identification)
T_X25_CHGINFO_REQ	unsigned long	T_YES / T_NO	Charging Information - Service Request
T_X25_CHGINFO_MU read-only	string	Unit. See text.	Charging Information - Monetary Unit
T_X25_CHGINFO_SC read-only	struct t_x25facval	Number of octets See text.	Charging Information - Segment Count
T_X25_CHGINFO_CD read-only	struct t_x25facinfocd	See text.	Charging Information - Call Duration
T_X25_RPOA	Array of unsigned longs	Indexes of each RPOA from 0 to 9999.	RPOA (Recognised Private Operating Agency)
T_X25_CALLDEF	struct t_x25facaddr	See text.	Call Deflection Selection
T_X25_CALLRED read-only	struct t_x25facaddr	See text.	Call Redirection or Deflection Notification
T_X25_CALLADDMOD	unsigned long	See text.	Called Line Address Modified Notification
T_X25_TDSAI	unsigned long	Transit delay in milliseconds from 0 to 65534, **T_UNSPEC**.	Transit Delay Selection and Indication
T_X25_CALLING_ADDEXT	struct t_x25addext	See text.	Calling Address Extension
T_X25_CALLED_ADDEXT	struct t_x25addext	See text.	Called Address Extension
T_X25_MTCN	struct t_x25facval	Throughput in bits/s from 75 to 64000, **T_UNSPEC**. See text.	Minimum Throughput Class Negotiation
T_X25_EETDN	struct t_x25faceetdn	Transit delay in milliseconds from 0 to 65534, **T_UNSPEC**. See text.	End-to-End Transit Delay Negotiation
T_X25_PRIORITY	struct t_x25facpr	**T_UNSPEC**. See text.	Priority
T_X25_PROTECTION	struct t_x25facpr	**T_UNSPEC**. See text.	Protection
T_X25_EDN	unsigned long	T_YES / T_NO, **T_UNSPEC**	Expedited Data Negotiation
T_X25_LOC_NONX25	string	See text.	Non-X.25 facilities provided by the local network.
T_X25_REM_NONX25	string	See text.	Non-X.25 facilities provided by the remote network.

A detailed description of the X.25 facilities can be found in ISO/IEC 8208 and ITU-T Recommendation X.25. Most of the fields elements of the structures specified above in the table are self-explanatory.

The following details provide further relevant information:

- **T_X25_PACKETSIZE**
  T_X25_WINDOWSIZE
  T_X25_TCN
  T_X25_MTCN

  The option value is in form of *struct t_x25facval.*

```
struct t_x25facval {
   unsigned long remote;   /* value for the direction of data    */
                           /* transmission from the called DTE.  */
   unsigned long local;    /* value for the direction of data    */
                           /* transmission from the calling DTE. */
}
```

- **T_X25_NUI**
  T_X25_CHGINFO_MU

  For these options of type string, the length of the option value is the total length of the option (specified in the *len* field of the **t_opthdr** structure) minus the length of the option header.

- **T_X25_FASTSELECT**

  The option value is in form of *unsigned long*. The different values are:

  — **T_NO**
  Fast Select not requested.

  — **T_X25_FASTSEL_NOREST**
  Fast Select requested with no restriction on response.

  — **T_X25_FASTSEL_REST**
  Fast Select requested with restriction on response.

- **T_X25_CHGINFO_SC**

  The option value is in form of one or many *struct t_x25facval*. There is one structure per tariff period managed by the network.

- **T_X25_CHGINFO_CD**

  The option value is in form of *struct t_x25facinfocd*. There is one structure per tariff period managed by the network.

```
struct t_x25facinfocd {
   unsigned char day;    /* number of days for the call    */
   unsigned char hour;   /* number of hours for the call   */
   unsigned char min;    /* number of minutes for the call */
   unsigned char sec;    /* number of seconds for the call */
}
```

- **T_X25_RPOA**

  The length of the option value determines how many RPOA transit networks are defined. This length is the total length of the option (specified in the *len* field of the **t_opthdr** structure) minus the length of the option header. As there is one entry in the array of unsigned longs per RPOA transit network, the number of defined RPOA transit networks is the total number of entries in the array.

- **T_X25_CALLDEF**

  The option value is in form of *struct t_x25facaddr*.

```
struct t_x25facaddr {
   unsigned char code;     /* reason code           */
   unsigned char len;      /* length in semi-octets */
   unsigned char addr[8];  /* DTE address           */
}
```

  The *code* field representing the reason for the call deflection or redirection is defined as below:

  — **T_X25_CLDEF1**
  Call-deflection by the originally-called DTE.

  — **T_X25_CLDEF2**
  Call-deflection by gateway as a result of call redirection due to originally-called DTE busy.

  — **T_X25_CLDEF3**
  Call-deflection by gateway as a result of call redirection due to originally-called DTE out-of-order.

  — **T_X25_CLDEF4**
  Call-deflection by gateway as a result of call redirection due to prior request from originally-called DTE for systematic call redirection.

- **T_X25_CALLRED**

  The option value is in form of *struct t_x25facaddr* described above. The *code* field represents the reason for the call deflection or redirection.

  In addition to **T_X25_CLDEF1 T_X25_CLDEF2 T_X25_CLDEF3** and **T_X25_CLDEF4**, the *code* field can have the following values:

  — **T_X25_CLRED1**
  Call-redirection due to originally-called DTE busy.

  — **T_X25_CLRED2**
  Call-distribution within a hunt group.

  — **T_X25_CLRED3**
  Call-redirection due to originally-called DTE out-of-order.

  — **T_X25_CLRED4**
  Call-redirection due to prior request from originally-called DTE for systematic call redirection.

- **T_X25_CALLADDMOD**

  The value of the option **T_X25_CALLADDMOD** can take one of the following values: **T_X25_CLDEF1 T_X25_CLDEF2 T_X25_CLDEF3 T_X25_CLDEF4 T_X25_CLRED1 T_X25_CLRED2 T_X25_CLRED3** or **T_X25_CLRED4**.

- **T_X25_CALLING_ADDEXT**
  T_X25_CALLED_ADDEXT

  The option value is in form of *struct t_x25addext*.

```
struct t_x25addext {
   unsigned char addr_type;
   unsigned char len;         /* length in bytes */
   unsigned char addr[20];
```

The *addr_type* field represents the type of the address extension:

— **T_X25_NSAPADDR**
For an address defined according to ISO / IEC 8348.

— **T_X25_OTHERADDR**
For an address defined in another format.

- **T_X25_EETDN**

The option value is in form of *struct t_x25faceetdn*.

```
struct t_x25faceetdn {
   unsigned short cumuldel;    /* cumulative end-to-end transit
                                       delay */
   unsigned short targetdel;   /* target end-to-end transit
                                       delay */
   unsigned short maxdel;      /* maximum end-to-end transit
                                       delay */
}
```

The values of the fields are used as follows:

— On an outgoing call:

cumuldel	If set **T_UNSPEC** the facility is omitted, otherwise value of the field is placed in outgoing packet.
targetdel	Ignored if *cumuldel* is **T_UNSPEC**. If set **T_UNSPEC** the target and maximum values are omitted from the outgoing packet, otherwise the target value is placed in the outgoing packet.
maxdel	Ignored if *cumuldel* or *targetdel* is **T_UNSPEC**. If set **T_UNSPEC** the maximum value is omitted from the outgoing packet, otherwise the maximum value is placed in the outgoing packet.

— On an incoming call:

cumuldel	Set **T_UNSPEC** if the field is not present in the received packet, otherwise set to the value present in the received packet.
targetdel	Set **T_UNSPEC** if the field is not present in the received packet, otherwise set to the value present in the received packet.
maxdel	Set **T_UNSPEC** if the field is not present in the received packet, otherwise set to the value present in the received packet.

— On an outgoing call accept:

cumuldel	Ignored if the facility was not present in the incoming call, otherwise the value of the field is placed in the outgoing packet.
targetdel	Ignored.
maxdel	Ignored.

Note: When the *cumuldel* field has been set by receipt of a call packet, options negotiation will not permit the value to be negotiated to a lower value or to be set to **T_UNSPEC**.

— On an incoming call accept:

cumuldel	Set **T_UNSPEC** if the field is not present in the received packet. Otherwise set to the value present in the received packet.
targetdel	Unchanged.
maxdel	Unchanged.

- **T_X25_PRIORITY**

The option value is in form of *struct t_x25facpr*.

```
struct t_x25facpr {
   unsigned char typeval;     /* type of the value       */
   unsigned char targetval;   /* target value            */
   unsigned char lowval;      /* lowest-acceptable value */
}
```

The *typeval* field can take the following values:

— **T_X25_PRIDATA**
Priority on data on a connection.

— **T_X25_PRIGAIN**
Priority to gain a connection.

— **T_X25_PRIKEEP**
Priority to keep a connection.

The *targetval* and the *lowval* fields have the following values: **T_PRITOP**, **T_PRIHIGH**, **T_PRIMID**, **T_PRILOW** or **T_PRIDFLT**, **T_UNSPEC**.

The length of the option value allows to determine how many types of priority are defined. The length of the option value is the total length of the option (specified in the *len* field of the **t_opthdr** structure) minus the length of the option header. For each define type, the target and the lowest acceptable values have to be given. The value **T_UNSPEC** allows not specifying a value.

- **T_X25_PROTECTION**

The option value is in form of *struct t_x25facpr* described above. The *typeval* field can take the following values:

— **T_X25_SRCPROTECT**
Source-address specific protection.

— **T_X25_DESTPROTECT**
Destination-address specific protection.

— **T_X25_GLBPROTECT**
Global protection.

The protection levels defined for the *targetval* and the *lowval* fields are :

— **T_NOPROTECT**
No protection (default value).

— **T_PASSIVEPROTECT**

— **T_ACTIVEPROTECT**

- **T_X25_LOC_NONX25**

  The option contains all local non-X.25 facilities in raw form as encoded in the local non-X25 facilities part of the facilities field. It may contain multiple local non-X.25 facilities.

  Note: As with all facilities markers, the marker itself is not present in the option buffer.

- **T_X25_REM_NONX25**

  The option contains all remote non-X.25 facilities in raw form as encoded in the remote non-X25 facilities part of the facilities field. It may contain multiple remote non-X.25 facilities.

  Note: As with all facilities markers, the marker itself is not present in the option buffer.

5.2 Use of XX25 Options

The following tables identify the *name* field of the structure *t_opthdr* with the mnemonic of the option, and the functions with which the option is meaningful.

X.25 Service

Option Name	t_optmgmt	t_connect	t_listen	t_accept	t_rcvconnect	t_optmgmt before a call of t_snddis	t_optmgmt after a call of t_rcvdis
T_X25_RST_OPT	X	X		X			
T_X25_D_OPT	X	X	X	X			
T_X25_USER_DACK	X	X	X	X	X		
T_X25_USER_EACK	X	X	X	X	X		
T_X25_VERSION	X	X		X	X	X	X
T_X25_DISCON_REASON	X					X	X
T_X25_DISCON_ADD	X						X
T_X25_CONN_DBIT	X	X	X	X			

X.25 Facilities

Option Name	t_optmgmt	t_connect	t_listen	t_accept	t_rcvconnect	t_optmgmt before a call of t_snddis	t_optmgmt after a call of t_rcvdis
T_X25_PKTSIZE	X	X	X	X	X		
T_X25_WINDOWSIZE	X	X	X	X	X		
T_X25_TCN	X	X	X	X	X		
T_X25_CUG	X	X	X				
T_X25_CUGOUT	X	X	X				
T_X25_BCUG	X	X	X				
T_X25_FASTSELECT	X	X	X				
T_X25_REVCHG	X	X	X				
T_X25_NUI	X	X		X			
T_X25_CHGINFO_REQ	X	X		X			
T_X25_CHGINFO_MU	X						X
T_X25_CHGINFO_SC	X						X
T_X25_CHGINFO_CD	X						X
T_X25_RPOA	X	X					
T_X25_CALLDEF						X	
T_X25_CALLRED	X		X				
T_X25_CALLADDMOD	X			X	X	X	X
T_X25_TDSAI	X	X	X		X		
T_X25_CALLING_ADDEXT	X	X	X			X	
T_X25_CALLED_ADDEXT	X	X	X	X	X	X	X
T_X25_MTCN	X	X	X			X	
T_X25_EETDN	X	X	X	X	X	X	
T_X25_PRIORITY	X	X	X	X	X	X	
T_X25_PROTECTION	X	X	X	X	X	X	
T_X25_EDN	X	X	X	X	X	X	

Appendix A

XX25 Header File

This appendix presents the additional header file information for XX25. Implementations supporting XX25 will provide equivalent definitions in **<xti_xx25.h>**. XX25 programs should include **<xti_xx25.h>** as well as **<xti.h>**.

Values specified for some of the symbolic constants in this X25 header definitions are designated as not mandatory for conformance purposes. These are indentified by the comment accompanying the constant definition.

```
/*
 *  New Error Codes to Support X.25 Service
 */
#define TX25NOTOACK   41          /* no data to acknowledge            */

/*
 * New Flags, defined to support X.25 Service
 */
#define T_X25_D         0x0800    /* Data with D bit set               */
#define T_X25_Q         0x1000    /* Qualified Data                    */
#define T_X25_RST       0x2000    /* Request or Indication of reset    */
#define T_X25_DACK      0x4000    /* Acknowledgement of data sent      */
                                  /* with D bit.                       */
#define T_X25_EACK      0x8000    /* Acknowledgement of expedited data */

/*
 * New Macros to Set and Retrieve Cause and Diagnostic
 * of a Connection Release
 */
#define T_X25_SET_CAUSE_DIAG(x,y)    (((x) << 8) + (y))
#define T_X25_GET_CAUSE(x)           (((x) >> 8) & 0xff)
#define T_X25_GET_DIAG(x)            ((x) & 0xff)

/*
 * X.25 Level
 */
#define T_X25_NP          0x101     /* X.25 Level; value is recommended    */
                                    /* only, not mandatory                 */

/*
 * New Options to Support X.25 Service.
 * These values are recommended only, not mandatory.
 */
#define T_X25_USER_DACK      0x0001  /* Explicit Acknowledgement of data   */
#define T_X25_USER_EACK      0x0002  /* Explicit Acknowledgement           */
                                     /* of expedited data                  */
#define T_X25_RST_OPT        0x0003  /* Reset is known to the appli.       */
#define T_X25_VERSION        0x0004  /* Version of ITU-T Recommendation    */
                                     /* X.25 or ISO/IEC X.25               */
#define T_X25_DISCON_REASON  0x0005  /* Reason of a Connection release     */
#define T_X25_DISCON_ADD     0x0006  /* Address of the user                */
                                     /* that released the connection       */
#define T_X25_D_OPT          0x0007  /* Support of the D bit               */
#define T_X25_CONN_DBIT      0x0008  /* Setting of the D-bit               */
                                     /* at the connection phase            */
```

```
/*
 * Options to support X.25 facilities.
 * These values are recommended only, not mandatory.
 */
#define T_X25_PKTSIZE         0x0009  /* Packet Size                          */
#define T_X25_WINDOWSIZE      0x000A  /* Window Size                          */
#define T_X25_TCN             0x000B  /* Througput Class Negotiation          */
                                      /* (basic format)                       */
#define T_X25_CUG             0x000C  /* CUG (basic format)                   */
#define T_X25_CUGOUT          0x000D  /* CUG with Outgoing Access             */
                                      /* (basic format)                       */
#define T_X25_BCUG            0x000E  /* Bilateral CUG                        */
#define T_X25_FASTSELECT      0x000F  /* Fast Select                          */
#define T_X25_REVCHG          0x0010  /* Reverse Charging                     */
#define T_X25_NUI             0x0011  /* NUI - Network User Identification    */
#define T_X25_CHGINFO_REQ     0x0012  /* Charging Information Req. Service    */
#define T_X25_CHGINFO_MU      0x0013  /* Charging Information Monetary Unit   */
#define T_X25_CHGINFO_SC      0x0014  /* Charging Information Segment Count   */
#define T_X25_CHGINFO_CD      0x0015  /* Charging Information Call Duration   */
#define T_X25_RPOA            0x0016  /* RPOA-Recognised Private              */
                                      /* Operating Agency-(basic format)      */
#define T_X25_CALLDEF         0x0017  /* Call Deflection Selection            */
#define T_X25_CALLRED         0x0018  /* Call Redirection or                  */
                                      /* Deflection Notification              */
#define T_X25_CALLADDMOD      0x0019  /* Called Line Address                  */
                                      /* Modified Notification                */
#define T_X25_TDSAI           0x001A  /* Transit Delay Selection              */
                                      /* and Indication                       */
#define T_X25_CALLING_ADDEXT  0x001B  /* Calling Address Extension            */
#define T_X25_CALLED_ADDEXT   0x001C  /* Called Address Extension             */
#define T_X25_MTCN            0x001D  /* Minimum Throughput Class Neg.        */
#define T_X25_EETDN           0x001E  /* End-to-End Transit Delay Neg.        */
#define T_X25_PRIORITY        0x001F  /* Priority                             */
#define T_X25_PROTECTION      0x0020  /* Protection                           */
#define T_X25_EDN             0x0021  /* Expedited Data Negotiation           */
#define T_X25_LOC_NONX25      0x0022  /* Non-X25 local facilities             */
#define T_X25_REM_NONX25      0x0023  /* Non-X25 remote facilities            */

/*
 * New Values for the XX25 Options
 */

/*
 * New Values for the T_X25_VERSION Option
 *   "Version of ITU-T Recommendation X.25 or ISO/IEC X.25"
 */
#define T_X25_1980    1980    /* X.25 1980 version */
#define T_X25_1984    1984    /* X.25 1984 version */
#define T_X25_1988    1988    /* X.25 1988 version */
#define T_X25_1993    1993    /* X.25 1993 version */

/*
 * New Values for the T_X25_FASTSELECT Option
 *   "Fast Select X.25 facility"
 */
#define T_X25_FASTSEL_NOREST   0x0002   /* Fast Select requested with no  */
                                        /* restriction on response        */
#define T_X25_FASTSEL_REST     0x0003   /* Fast Select requested with     */
                                        /* restriction on response        */
```

```
/*
 * New Defines for the reason code for the DTE deflecting the call
 * for the Options:
 *   T_X25_CALLDEF    "Call Deflection Selection facility"
 *   T_X25_CALLRED    "Call Redirection or Deflection Notification facility"
 *   T_X25_CALLADDMOD "Called Line Address Modified Notification facility"
 */
#define T_X25_CLDEF1   0x0001   /* Call-deflection by the                */
                                /* originally-called DTE                 */
#define T_X25_CLDEF2   0x0002   /* Call-deflection by gateway as a result */
                                /* of call redirection due to            */
                                /* originally-called DTE busy            */
#define T_X25_CLDEF3   0x0003   /* Call-deflection by gateway as a result */
                                /* of call redirection due to            */
                                /* originally-called DTE out-of-order    */
#define T_X25_CLDEF4   0x0004   /* Call-deflection by gateway as a result */
                                /* of call redirection due to prior      */
                                /* request from originally-called DTE    */
                                /* systematic call redirection           */
#define T_X25_CLRED1   0x0005   /* Call-redirection due to               */
                                /* originally-called DTE busy            */
#define T_X25_CLRED2   0x0006   /* Call-distribution within a hunt group  */
#define T_X25_CLRED3   0x0007   /* Call-redirection due to               */
                                /* originally-called DTE out-of-order    */
#define T_X25_CLRED4   0x0008   /* Call-redirection due to prior request  */
                                /* from originally-called DTE for        */
                                /* systematic call redirection           */

/*
 * New Defines for the type of the address extension for the Options:
 *   T_X25_CALLING_ADDEXT  "Calling Address Extension facility"
 *   T_X25_CALLED_ADDEXT   "Called Address Extension facility"
 */
#define T_X25_NSAPADDR    0x0001   /* address defined according to        */
                                   /* ISO/IEC 8348                        */
#define T_X25_OTHERADDR   0x0002   /* address defined in another format   */

/*
 * New Defines for the type of the priority for the T_X25_PRIORITY
 * option "Priority facility"
 */
#define T_X25_PRIDATA     0x0001   /* priority on data on a connection    */
#define T_X25_PRIGAIN     0x0002   /* priority to gain a connection       */
#define T_X25_PRIKEEP     0x0003   /* priority to keep a connection       */

/*
 * New Defines for the Type of the Protection for the T_X25_PROTECTION
 * option "Protection Facility"
 */
#define T_X25_SRCPROTECT   0x0001   /* Source-address specific protection */
#define T_X25_DESTPROTECT  0x0002   /* destination-address specific       */
                                    /* protection                         */
#define T_X25_GLBPROTECT   0x0003   /* global protection                  */
```

```
/*
 * New Structures for the XX25 Options (X.25 facilities)
 */

struct t_x25addext {
    unsigned char addr_type;
    unsigned char len;
    unsigned char addr[20];
};

struct t_x25facval {
    unsigned long remote;       /* value for the direction of data   */
                                /* transmission from the called DTE  */
    unsigned long local;        /* value for the direction of data   */
                                /* transmission from the calling DTE */
}

struct t_x25facinfocd {
    unsigned char day;          /* number of days for the call       */
    unsigned char hour;         /* number of hours for the call      */
    unsigned char min;          /* number of minutes for the call    */
    unsigned char sec;          /* number of seconds for the call    */
}

struct t_x25facaddr {
    unsigned char code;         /* reason code                       */
    unsigned char len;          /* length in semi-octets             */
    unsigned char addr[8];      /* DTE address                       */
}

struct t_x25faceetdn {
    unsigned short cumuldel;    /* cumulative transit delay          */
    unsigned short targetdel;   /* target end-to-end transit delay   */
    unsigned short maxdel;      /* maximum end-to-end transit delay  */
}

struct t_x25facpr {
    unsigned char typeval;      /* type of the value       */
    unsigned char targetval;    /* target value            */
    unsigned char lowval;       /* lowest-acceptable value */
}
```

Appendix B

ISO X.25 Protocol Terminology

Protocol Address

The protocol address is the X.25 address.

Sending Data of Zero Octets

The X.25 service definition, both in connection-oriented mode and in permanent-connection mode, permits sending of a TSDU of zero octets.

Expedited Data Negotiation

Expedited Data Negotiation is an optional ITU-T-specified DTE facility which may be used for a given Virtual Call. The calling DTE uses the Expedited Data Negotiation Facility during the connection phase to indicate whether it wishes to use the expedited data-transfer procedures. This indication is provided by a higher layer entity in the calling DTE. This facility is conveyed transparently by public data networks but may be set to non-use of the expedited data-transfer procedures by gateways and private networks that do not support them (see ISO / IEC 8208 and ITU-T Recommendation X.25).

Fast Select

Fast Select is an optional user facility which may be requested by a DTE for a given Virtual Call. The Fast Select facility allows the DTE to transmit call setup and clearing packets with an User Data Field of up to 128 octets.

Segmentation and Concatenation

The X.25 provider manages the segmentation and the concatenation of the TSDU (if the X.25 user did not already make it). At the XTI interface, when a TSDU is broken up into more than one sub-unit, the **T_MORE** flag is set on each sub-unit passed across the interface, except the last.

User Data

Since, at most, 128 octets of User Data may be sent in X.25 call and clear packets, the field *udata.len* in the functions *t_connect*( ), *t_listen*( ), *t_rcvdis*( ) and *t_snddis*( ) should have a value up to 128.

Glossary

API
Application Programming Interface

CUD
Call User Data

DCE
Data Communication Equipment

DTE
Data Terminal Equipment

DXE
DTE or DCE

EM
Event Management

ENSDU
Expedited Network Service Data Unit

ETSI
European Telecommunications Standards Institute

ISO
International Organization for Standardization

ITU-T
International Telecommunications Union - Telecommunications. Previously called CCITT.

NSAP
Network Service Access Point

NSDU
Network Service Data Unit

OSI
Open System Interconnection

PVC
Permanent Virtual Circuit

SVC
Switched Virtual Circuit

VC
Virtual Circuit

X.25
ITU-T Recommendation X.25: interface between a Data Terminal Equipment (DTE) and a Data Circuit terminating Equipment (DCE) for terminals operating in the packet mode and connected to public data networks by dedicated circuits [ITU-T Blue Book, 1988]

XX25
API for X.25 under X/Open Transport Interface (XTI)

Index