INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

T.180

(06/98)

SERIES T: TERMINALS FOR TELEMATIC SERVICES

Homogeneous access mechanism to communication services

ITU-T Recommendation T.180

(Previously CCITT Recommendation)

ITU-T T-SERIES RECOMMENDATIONS

TERMINALS FOR TELEMATIC SERVICES

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

ITU-T RECOMMENDATION T.180

HOMOGENEOUS ACCESS MECHANISM TO COMMUNICATION SERVICES

Summary

This Recommendation specifies a homogenous access mechanism to communication services (called XAPI). The XAPI, which is an abbreviation for eXtensive Application Programming Interface, is an operating system and language-independent programming interface to general communication services.

This Recommendation specifies a set of functions which allow XAPI users to have access to the services of an underlying provider. These functions apply to all providers which are specified in this Recommendation. A model of communication is introduced which defines the semantics of those XAPI functions which are communication related.

Making available appropriate providers, the communication system can be tailored to specific requirements, and all communication services are accessible via one homogenous access mechanism.

Source

ITU-T Recommendation T.180 was prepared by ITU-T Study Group 8 (1997-2000) and was approved under the WTSC Resolution No. 1 procedure on the 18th of June 1998.

FOREWORD

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

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

The approval of Recommendations by the Members of the ITU-T is covered by the procedure laid down in WTSC Resolution No. 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation the term *recognized operating agency (ROA)* includes any individual, company, corporation or governmental organization that operates a public correspondence service. The terms *Administration, ROA* and *public correspondence* are defined in the *Constitution of the ITU (Geneva, 1992)*.

INTELLECTUAL PROPERTY RIGHTS

The ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. The ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, the ITU had not received notice of intellectual property, protected by patents, which may be required to implement this Recommendation. However, implementors are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database.

CONTENTS

Recommendation T.180

HOMOGENEOUS ACCESS MECHANISM TO COMMUNICATION SERVICES

(Geneva, 1998)

1 Scope

The XAPI, which is an abbreviation for eXtensive Application Programming Interface, is an operating system and language-independent homogenous access mechanism to general communication services. It is not dedicated to a certain layer, but allows access to all layers of the OSI reference model and other layered communication models (e.g. conferencing). The XAPI provides a framework of functions for the use in communication applications. All communication services are accessible through this set of functions. The XAPI does not impose any restrictions on the service interface of the underlying communication platform.

Which services are made available via the XAPI depends on the installed service providers, and not on the XAPI, which only provides the access mechanism. New service providers can be added in the XAPI configuration. Thus, the communication system can be tailored to specific requirements and all communication services are accessible via one homogeneous access mechanism.

2 References

The following ITU-T Recommendations and other references contain provision which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; all users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published.

– ITU-T Recommendation F.581 (1993), *Guidelines for Programming Communication Interfaces (PCIs) definition: Service Recommendation.*

– ITU-T Recommendation H.320 (1997), *Narrow-band visual telephone systems and terminal equipment.*

– ITU-T Recommendation T.30 (1996), *Procedures for document facsimile transmission in the general switched telephone network.*

– ITU-T Recommendation T.120 (1996), *Data protocols for multimedia conferencing.*

– ITU-T Recommendation T.121 (1996), *Generic application template.*

– ITU-T Recommendation T.122 (1998), *Multipoint communication service – Service definition.*

– ITU-T Recommendation T.123 (1996), *Network specific data protocol stacks for multimedia conferencing.*

– ITU-T Recommendation T.124 (1998), *Generic Conference Control.*

– ITU-T Recommendation T.125 (1994), *Multipoint communication service protocol specification.*

– ITU-T Recommendation T.127 (1995), *Multipoint binary file transfer protocol.*

– ITU-T Recommendation T.434 (1996), *Binary file transfer format for the telematic services.*

– ITU-T Recommendation X.200 (1994) | ISO/IEC 7498-1:1994, *Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model.*

3 Definitions

3.1 For the purposes of this Recommendation, the following terms as defined in Recommendation X.200 apply:

– **(N)-connection**;

– **(N)-entity**;

– **(N)-layer**;

– **(N)-service**;

– **(N)-Service Access Point (SAP)**.

3.2 For the purposes of this Recommendation, the following terms as defined in Recommendation T.124 apply:

– **conference**;

– **Multipoint Control Unit**;

– **node**;

– **terminal**.

3.3 This Recommendation defines the following terms:

3.3.1 application connection: Serves for communication between application entities.

3.3.2 application entity: A service user, or some other entity, which may participate in a connection.

3.3.3 application system: A protocol stack comprising some or all of the OSI layers 5 (Session) to 7 (Application).

3.3.4 communication endpoint: Synonymously used for "Service Endpoint".

3.3.5 communication platform: Consists in a number of communication service providers, in a homogeneous access mechanism, by means of which service users, which are distributed in space, may establish communication between them.

3.3.6 conference control: Encompasses functions such as conference establishment and termination, information about each node which is participating in the same conference, information about each application entity in the conference, coordination of conference conductorship, as well as other miscellaneous functions.

3.3.7 connection: A logical association between two or more entities, enabling communication between them.

3.3.8 control entity: A specific service user which handles conference control and which may participate in a connection.

3.3.9 control connection: Serves for communication between control entities.

3.3.10 multipeer relationship: In a multipeer relationship, the users may negotiate the characteristics of their interaction and, afterwards, communicate with each other obeying the rules they have negotiated: all users (an entity and its peers) have potential equal rights.

3.3.11 multipoint aware application: An application entity which is able to participate in a multipoint connection.

3.3.12 multipoint connection: A connection between more than two entities.

3.3.13 Peer-to-peer relationship: In a peer-to-peer relationship, the users may negotiate the characteristics of their interaction and, afterwards, communicate obeying the rules they have negotiated: both users (an entity and its peer entity) have potential equal rights.

3.3.14 point-to-point connection: A connection between two entities.

3.3.15 protocol module: The implementation of a communication protocol whose services can be accessed via the XAPI. Usually a protocol module implements a single layer in the OSI sense.

3.3.16 service access point: The point at which services are provided to a user. Associated with each SAP is an address.

3.3.17 service endpoint: Specifies the local link between a service user and a service provider. It consists of two parts: a SAP address and an additional identifier (optional), which is unique within the scope of the SAP.

3.3.18 service provider: A communication protocol stack that provides a certain service to the user at the upper interface of its topmost protocol module.

3.3.19 session: A peer-to-peer or multipeer relationship between application entities which are communicating via an application connection. In general, a session brings into focus a specific topic of discussion.

3.3.20 transport system: A protocol stack comprising some or all of the OSI layers 1 (Physic) to 4 (Transport).

4 Abbreviations

This Recommendation uses the following abbreviations:

AAL	ATM Adaptation Layer
ACSE	Association Control Service Element
ATM	Asynchronous Transfer Mode
AVC	Audiovisual Control
BFT	Binary File Transfer
CLS	Connectionless Service
COS	Connection oriented Service
CSCW	Computer-Supported Cooperative Work
Fd	File Descriptor
GCC	Generic Conference Control
HDLC	High-Level Data-Link Control
IDL	Interface Definition Language
ISDN	Integrated Services Digital Network
MBFT	Multipoint Binary File Transfer
MCS	Multipoint Communication Service

MCU	Multipoint Control Unit
MRPC	Multipoint Remote Procedure Call
NSAP	Network Service Access Point
NSM	Network-Specific Mappings
OSI	Open Systems Interconnection
PDId	Parallel Data Identifier
PDU	Protocol Data Unit
PSAP	Presentation Service Access Point
PSTN	Public Switched Telephone Network
QoS	Quality of Service
RDC	Remote Device Control
ROSE	Remote Operations Service Element
RTSE	Reliable Transfer Service Element
SAP	Service Access Point
SP	Service Primitive
SSAP	Session Service Access Point
TCP	Transmission Control Protocol
TSAP	Transport Service Access Point
XAPI	eXtensive Application Programming Interface

5 Structure of this Recommendation

The description of the XAPI is divided into three steps, each step representing a specific level of abstraction. The main part of this Recommendation contains steps 1 and 2 which are common for all providers. Step 3 is defined in Appendix I containing a list of examples of service providers. Appendix I contains seven parts (I.1 to I.7).

Step 1:

Subclauses 6 and 7 introduce the XAPI and a model of communication which is supported by the XAPI. The model represents appropriate connection-oriented communication service primitives which apply not only to peer-to-peer but also to multipeer communication. The service primitives and their sequences possible at a service access point describe a generic communication structure which is applicable to data services as well as to multimedia services.

Step 2:

Subclauses 8 and 9 introduce the XAPI functions in detail. The relationship of the functions with respect to the model of communication (step 1) is pointed out.

Step 3:

Steps 1 and 2 specify those aspects of the XAPI which apply to all providers and which are used by all types of applications. Steps 1 and 2 do not contain any protocol-specific information as option or service primitive definitions. All definitions that may be carried by XAPI services, but related to the specific underlying protocol stacks, are contained as examples in the following subclauses of Appendix I:

- Subclause I.1: XAPI access to the service provider for the ISDN B-channel
- Subclause I.2: XAPI access to the service provider for BFT over T.30
- Subclause I.3: XAPI access to the service provider for FAX4 and BFT
- Subclause I.4: XAPI access to the service provider for ACSE and ROSE
- Subclause I.5: XAPI access to the service provider for audio and video control
- Subclause I.6: XAPI access to the service provider for T.120 conference control
- Subclause I.7: XAPI access to the service provider for T.127 MBFT

An IDL description of the XAPI can be found in Annex A.

A list of error codes that may be returned by the XAPI functions is given in Annex B.

Appendix I describes the access of the XAPI user to specific providers.

A tutorial is presented in Appendix II.

6 Introduction to the XAPI

The XAPI, which is an abbreviation for eXtensive Application Programming Interface, is an operating system and language-independent homogenous access mechanism to general communication services. It is not dedicated to a certain layer, but allows access to all layers of the OSI reference model and other layered communication models (e.g. conferencing). The XAPI provides a framework of functions for the use in communication applications. All communication services are accessible through this set of functions. The XAPI imposes no restrictions on the service interface of the underlying communication platform.

The services made available via XAPI depend on the installed service providers, and not on XAPI which only provides the access mechanism. New service providers can be added in the XAPI configuration. Thus the communication system can be tailored to specific requirements, and all communication features are accessible via one homogeneous interface.

This Recommendation specifies the general aspects of the XAPI, as they apply to all providers and as they are used by all types of applications. It does not contain any protocol-specific information as option or service primitive definitions. All definitions that are carried by XAPI services, but related to the specific underlying protocols, are contained as examples in Appendix I: e.g. XAPI access to the service provider for BFT over T.30, XAPI access to the service provider for ACSE and ROSE, XAPI access to the service provider for audio and video control, XAPI access to the service provider for T.120 MBFT, etc.

6.1 Location of the XAPI

The XAPI is a communication interface inside a terminal equipment. It allows a service user to communicate with its respective service provider. Therefore, two entities can be distinguished around the XAPI. These are the service user (XAPI user, or user for short) and the service provider (or provider for short).

The XAPI defines a number of functions. First, the XAPI allows for the binding between a user and a provider (Initialization phase). After having performed successfully this process, a connection between the user and its peer(s) may be established, maintained and released (Communication establishment phase, Data transfer phase, Communication release phase). Finally, the interaction between user and provider is terminated (De-initialization phase).

There are several XAPI utility functions which can be used in all phases of communication. These functions are used to inform the XAPI user about current events on a service endpoint or indicate that no event is available, to change the execution mode (i.e. synchronous or asynchronous mode) of the XAPI functions for the service endpoint, to inform the user about the current characteristics of the underlying service providers, to negotiate protocol options, to provide error handling, etc.

A service endpoint specifies a local link between a specific XAPI user and a specific provider. During the initialization phase a service endpoint is created and activated for use in communication. Figure 1 gives a picture of how the XAPI relates to user and provider after the creation of a communication path between them. Fd is the name of the accessory service endpoint.

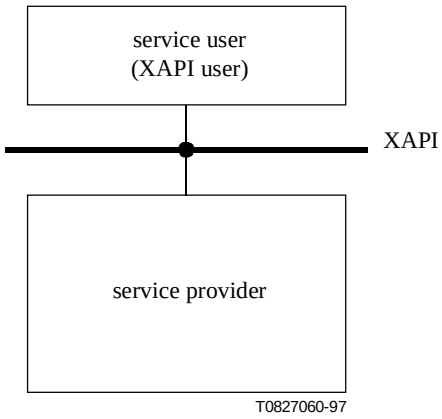


Figure 1/T.180 – Local link between user and provider, identified by the service endpoint Fd

The XAPI supports connection-mode operations defined in the framework of OSI (e.g. ACSE/ROSE) and non-OSI (e.g. Video Codec Controller) as well as communication functions for multipoint environments (e.g. conferencing).

The location of the XAPI depends on the choice of the underlying provider. Figure 2 shows the general location of the XAPI within an OSI-like structured platform: the XAPI is located within the application layer (layer 7) of the OSI Reference Model.

NOTE – The location also reflects the fact that the XAPI may be realized as a driver interface which is designed for use in different system architectures. Therefore, task-oriented functions such as file handling shall not be performed below the XAPI.

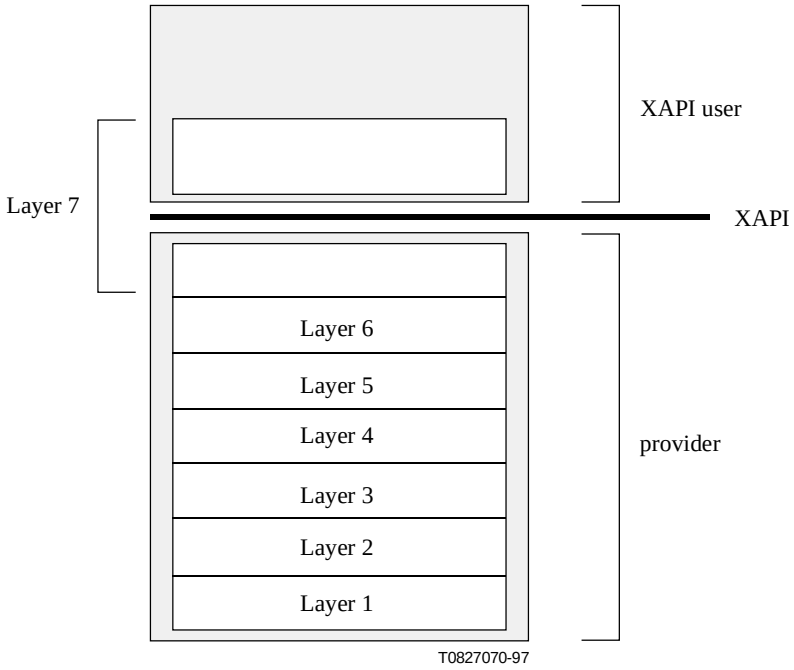


Figure 2/T.180 – Location of the XAPI within an OSI-like structured platform

6.2 Phases of communication

The XAPI supports connection-oriented communication between users. The four communication phases provided by XAPI functions are:

- Initialization/de-initialization,
- Connection establishment,
- Data transfer,
- Connection release.

Communication-related additional functions which are not mapped to a specific communication phase are listed in this subclause too.

In the following, these phases as well as the additional functions are briefly described. A detailed definition of the functions is contained in clauses 8 and 9 respectively. Clause 7 introduces the semantic model for the communication-relevant functions of the XAPI.

– **Initialization/de-Initialization phase**

During the initialization phase, a service endpoint for the required service provider is created and activated for use in communication. In the following, this service endpoint is called Fd or Fd-i. When communication has finished, the endpoint is first deactivated and finally destructed in the de-initialization phase. There are four XAPI functions, supporting initialization and de-initialization. They are listed in the sequential order, in which they are used:

Open: x_open

Bind: x_bind

Unbind: x_unbind

Close: x_close

– **Connection establishment phase**

During the connection establishment phase, users of the same service establish a connection between them (e.g. a Telefax terminal is connected to the remote Telefax terminal), or each of them establishes access to some appropriate multipoint connection by means of which they may communicate in a multipoint fashion with each other (e.g. a multipoint aware user application is connected to a multipoint session which offers communication facilities to matching user applications).

These functions are supported by the XAPI in the connection establishment phase:

Connect request: x_conreq

Connect confirm: x_conconf

Connect indication: x_conind

Connect response: x_conrsp

– **Data transfer phase**

Once communication has been established, data may be transferred. There are functions to initiate the transfer of data, and additional functions (i.e. x_sndsp and x_rcvsp) that may be invoked during the data transfer phase. The latter serve for different purposes; e.g. the functions are used to change the set of capabilities which characterize the data transfer phase. These functions are supported by the XAPI in the data transfer phase:

Send data: x_snddata

Receive data: x_rcvdata

Send service primitive: x_sndsp

Receive service primitive: x_rcvsp

– **Connection release phase**

The XAPI provides two ways to leave the communication: abortive release (disconnection) and orderly release. The abortive release is a mandatory feature and is to be supported by all service providers. The orderly release is an optional feature. These functions are supported by the XAPI in the connection release phase:

Send disconnect: x_snddis

Receive disconnect: x_rcvdis

Release request: x_relreq

Release confirmation: x_relconf

Release indication: x_relind

Release response: x_relrsp

– **Additional communication-related functions**

During connection establishment, data transfer, and connection release, general information services (e.g. information on conferences that are currently in existence) may be invoked by the user. In general, these services are not related to some specific connection. These functions are supported by the XAPI:

Send information: x_sndinfo

Receive information: x_rcvinfo

The function x_rcvend indicates to the user that the local resource management has finished all operations concerning the release phase of a connection and that a new connection may now be established:

Receive end: x_rcvend

6.3 Applications that are supported by XAPI

Various kinds of applications are supported by XAPI. Figure 3 shows examples of XAPI users and sketches underlying providers.

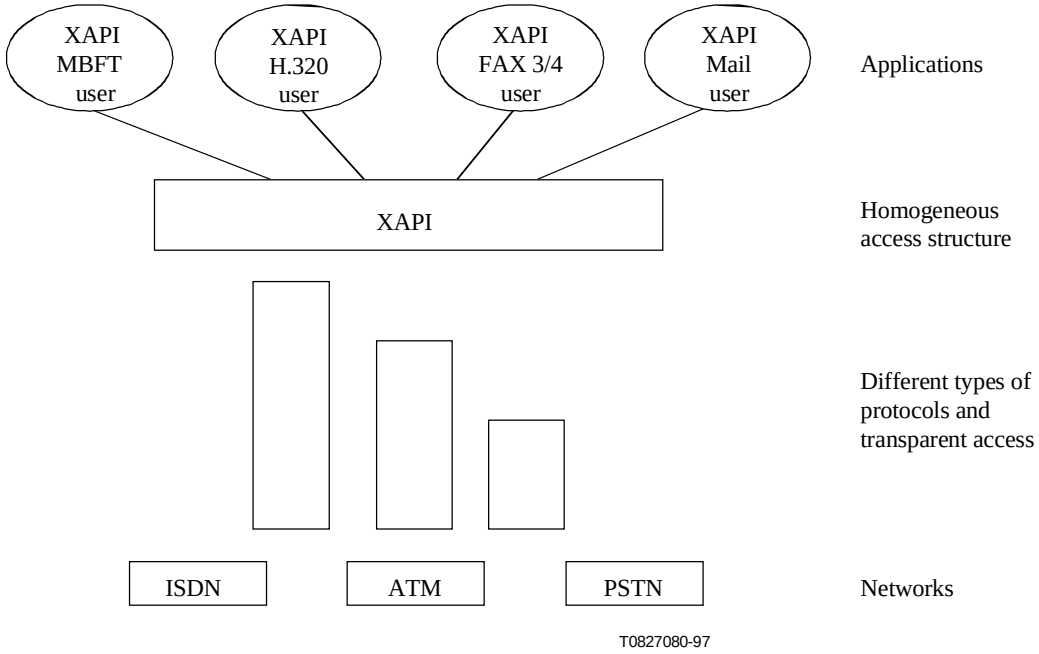


Figure 3/T.180 – Example: Applications, protocols and networks around the XAPI

Different types of applications are recognized that make use of XAPI. This implies that the use of XAPI functions may have different interpretations in different application environments.

There are point-to-point applications, where two users are directly connected during the communication phase and "multipoint-aware" applications, where many users may communicate with each other. Typical examples for point-to-point applications are Binary File Transfer (according to Recommendation T.434) or Telefax applications. As XAPI is the communication interface, store-and-forward applications also fall into this type of point-to-point communication.

There are also applications where the user may be connected to a service (e.g. offered by the network operator). The service itself may connect other providers on behalf of the requesting user and provide the requested information. In this case, no direct connection exists between end-users for a specific time period during the communication phase. (This may be the case when an application download is necessary from a level-1 gateway, before the user can be connected to the service provider.)

Additionally, the XAPI may be used in cases where nesting of applications is supported: a first "general" application is established and "specific" applications may be invoked afterwards (see Figure 15). Conferences are an example of such a configuration, in which several users can agree to establish a conference as the general application and subsequently select specific multipoint-aware applications that are used in the conference (e.g. Multipoint Binary File Transfer, Still Image, etc.).

The different types of applications may have an impact on the use of the XAPI functions in the four communication phases. For example, for the point-to-point applications, only one initialization phase is necessary (i.e. only one connection is needed). In turn, for conference configurations, more than one connection may be needed. Details are described in clause 7.

The XAPI divides XAPI user processes from service providers. So the XAPI is that part of the functionality of a platform by means of which an arbitrary user may have access to appropriate services to meet the user requirements.

Many types of applications may have access to different service providers. Figure 4 illustrates the access to the communication platform.

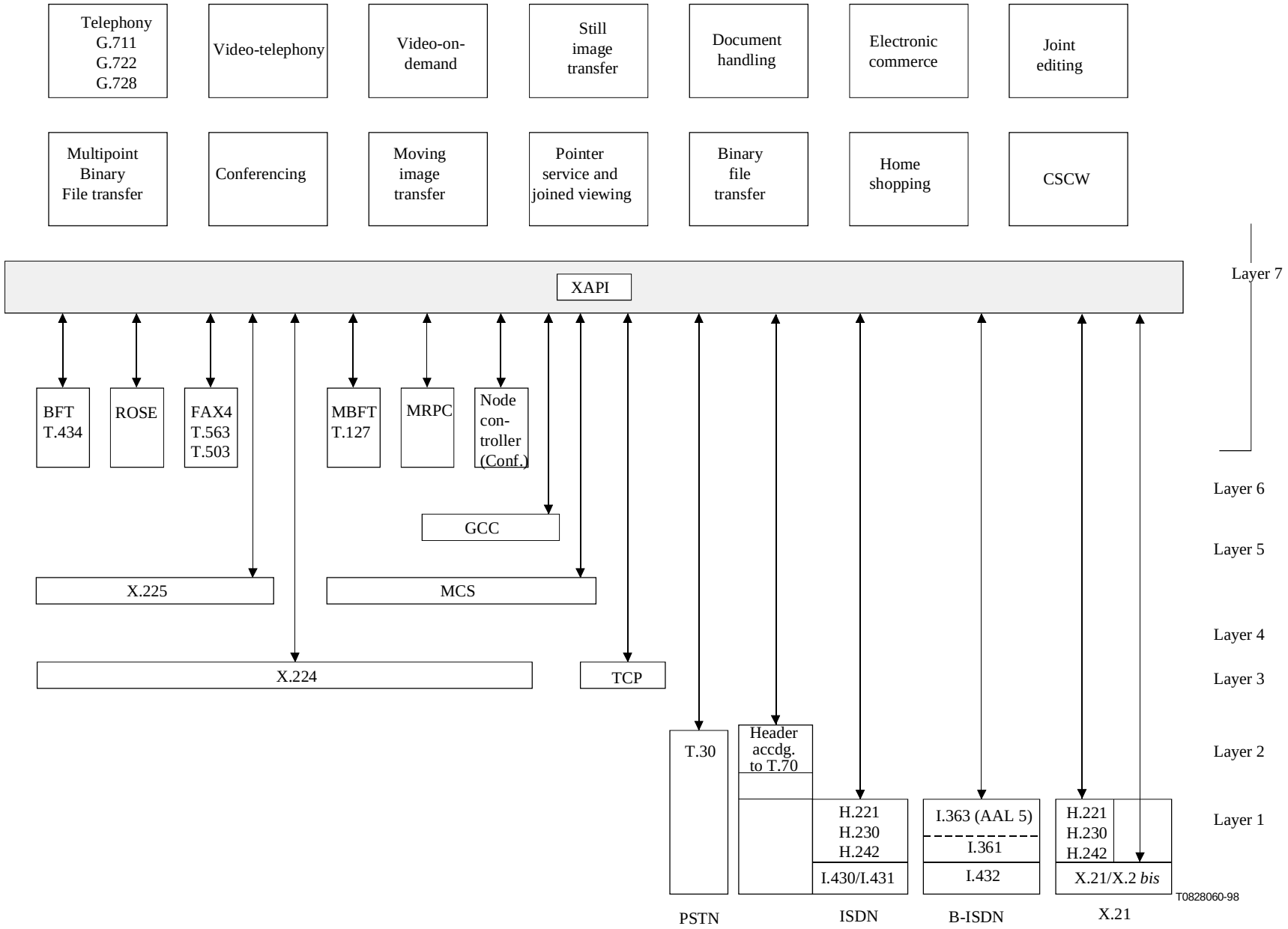


Figure 4/T.180 – Example: Access to the communication platform

Using this platform, single medium as well as multimedia applications and/or point-to-point as well as multipoint applications are supported by appropriate providers.

Figure 4 *bis* shows the relationship between the XAPI, the T.611 and the T.200 interfaces.

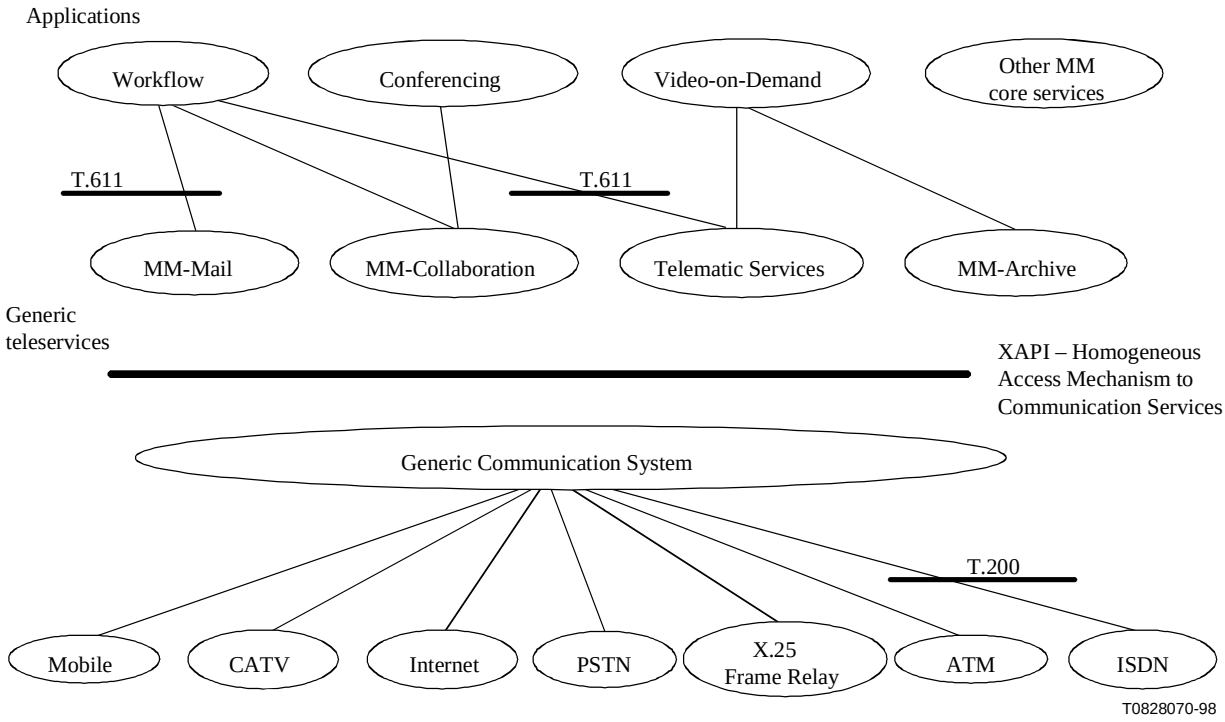


Figure 4 *bis*/T.180 – Relationship between XAPI, T.611, and T.200 interfaces

7 A Model of communication

This clause describes a model of communication which is supported by the XAPI. The model represents appropriate connection-oriented communication service primitives, which apply not only to peer-to-peer but also to multipeer communication. The service primitives and their sequences possible at service access points describe a generic communication structure, which is applicable to data services as well as to multimedia services.

7.1 Classes of communication

7.1.1 Peer-to-peer communication

The OSI reference model describes cooperation between computers independent of implementation aspects.

The *layer* is one of the basic concepts of the OSI reference model. When referring to a single layer, it is conventional to call it the (N)-layer. In the same way, the layer above is the (N + 1)-layer and the layer beneath the (N – 1)-layer.

The (N)-layer protocol provides the functionality of the (N)-layer: every layer is defined by a precise set of functions and associated messages [Protocol Data Units (PDUs)]. A "local part" of an (N)-layer protocol is called (N)-entity.

The purpose of a protocol is to provide a service to users (entities) residing above the respective layer boundary.

(N)-service users have access to the (N)-service only at (N)-Service Access Points [(N)-SAPs]. Associated with each SAP is an address. The address of that SAP can be used to identify the entity. A service endpoint consists of a SAP address and an additional identifier, which is unique within the scope of the SAP.

Most OSI relationships among communicating users (applications) are peer-to-peer. In a peer-to-peer relationship, the users may negotiate the characteristics of their interaction: both users (an entity and its peer entity) have potential equal rights.

In the framework of the OSI reference model, two modes of operations are defined: connection oriented (CO) operation and connectionless (CL) operation.

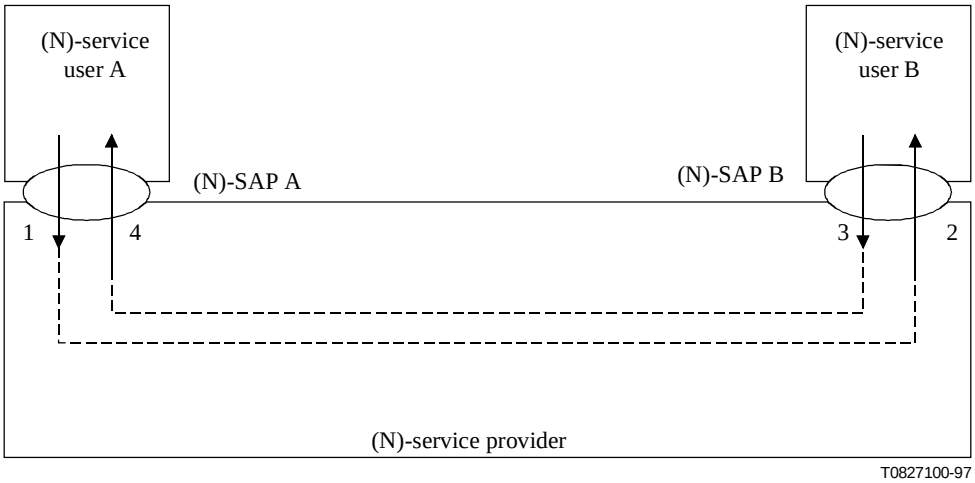
For connection mode operation, an (N)-connection is defined as "an association established by the (N)-layer between two or more (N + 1)-entities for the transfer of data". Three phases of operation are characterizing this mode, namely connection establishment, data transfer, and connection release.

The definition of the OSI services for connections between two entities (point-to-point connections) reflects these modes: for CO operation dedicated services such as (N)-CONNECT for connection establishment, (N)-DATA in the data transfer phase, and (N)-DISCONNECT or (N)-RELEASE for connection release are specified. The elements of these services are called service primitives. They apply at specific service endpoints.

These primitives and their sequences specify services which may be accessed at (almost) any layer boundary: the layer specific information is contained in accessory parameters. Thus the primitives and their sequences give rise to call the model also a generic communication structure.

There are OSI services in the data transfer phase, which apply only to a few layers. The OSI Session Service S-TOKEN-PLEASE is an example. In the model of communication described in this subclause these (and other) services are subsumed under the primitives X-SNDSP and X-RCVSP (see below).

Figure 5 shows the (N)-CONNECT service. User A is the initiator of an (N)-connection (A, B). Correct (N)-connection establishment between entity A (user A) and its peer entity (user B) is performed by communicating the service primitives 1 to 4 in the order stated.



1 (N)-CONNECT request
2 (N)-CONNECT indication
3 (N)-CONNECT response
4 (N)-CONNECT confirmation

Figure 5/T.180 – Sequence of service primitives of the (N)-CONNECT service

7.1.2 Multipeer communication

The primitives described in 7.1.1 will now be used to access services offered in point-to-point environments as well as in multipoint environments. To differentiate between the "pure" OSI semantics of the services and the semantics described in this subclause, the services and the primitives are prefixed by the character X- (e.g. X-CONNECT service).

The X-primitives allow entities to access connection-oriented services at specific service endpoints. These services are:

- connection establishment service (X-CONNECT service);
- data transfer service (X-DATA service);
- primitive transfer service (X-SP service);
- disconnect service (X-DISCONNECT service);
- orderly release service (X-RELEASE service);
- information service (X-INFO service).

The semantics of these services are provider dependent. The X-primitives and the sequences of service primitives at a service endpoint are defined in 7.2 (see Figure 10). The introduction of connectionless services is out of the scope of this Recommendation.

A connection is a logical association between two or more entities, enabling communication between them. As an example, and from the viewpoint of a service endpoint, a connection is established by the usage of an allowed sequence of primitives of the X-CONNECT service. Which primitives apply depends on the provider.

A connection between two entities is called a *point-to-point* connection. A connection between more than two entities is called a *multipoint* connection.

In peer-to-peer relationships, two users (applications) may exchange information via a point-to-point connection.

In multipeer-to-multipeer relationships, multiple users (applications) may exchange information via a multipoint connection.

Point-to-point and multipoint environments offer the communication facilities of an underlying provider to users rather than describe the relationship among users.

In point-to-point environments, the negotiation of the user requirements and the provider facilities is usually performed by a single service, the X-CONNECT service. Afterwards, both users may send and receive data using the X-DATA service.

In multipoint environments, negotiating the provider facilities is more complex than in point-to-point environments. Therefore, it may be advantageous to separate the negotiation of the provider facilities from that of the user requirements.

Conference environments may be looked at as multipoint environments, where specific conference providing services are added. Separating conference users from conference providers,

conference users may access:

- conference control services (e.g. getting information on conferences, creating a conference, joining a conference, leaving a conference, etc.); as well as
- conference application services (e.g. multipoint file transfer, whiteboard, audio, video, device control, etc.); and

conference providers may support:

- conference control (providing the services required from users, controlling the conference resources, ...); as well as
- application protocols (e.g. protocols providing file transfer, protocols providing whiteboards, protocols providing audio, ...); and
- multipoint communication.

Application protocols are point-to-point protocols using point-to-point communication functions, or multipoint protocols using multipoint communication functions. Conference control may comprise specific conference management functions (e.g. conference conductorship).

Figure 6 shows an example of a conference, consisting in three conference users A, B, and C, and a conference providing service S, and a conference provider. The service S consists in a user part and a provider part (the latter being part of the conference provider).

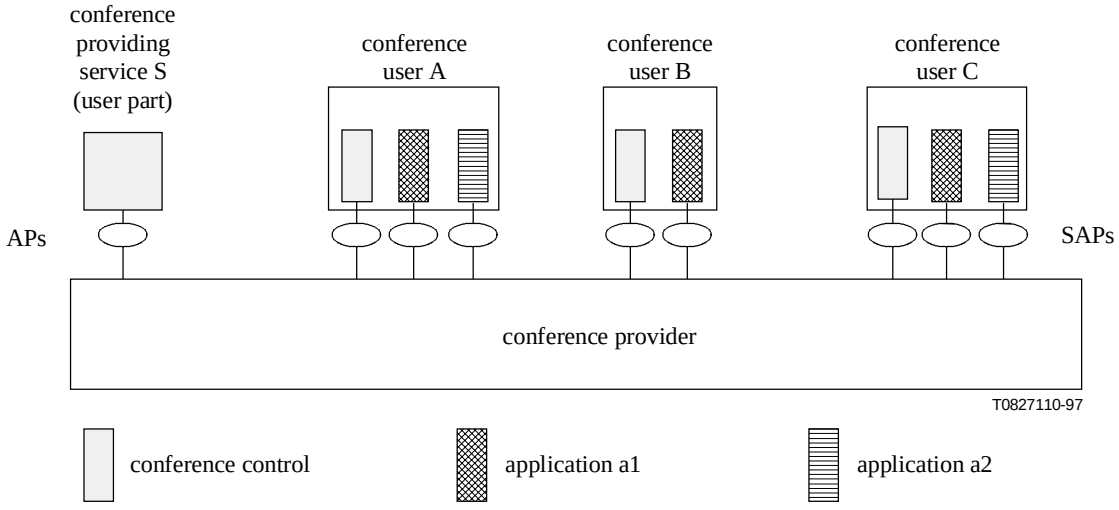


Figure 6/T.180 – Users and provider of an example conference

The service S (user part and/or provider part) supports conference control services as described above whereas the remainder of the conference provider is concerned with functions of the multipoint environment.

Figure 6 describes the separation of conference users from the conference provider rather than the mapping of user and provider functions on real systems. Figure 7 shows a possible configuration for the example conference, consisting in three terminals and a Multipoint Control Unit (MCU). The multipoint connection (cK, cA, cB, cC) handles the communication between the control entities (e.g. the T.120 Node Controller) of the terminals and the MCU. cX is the name of the control-SAP (or endpoint of the SAP) identifying the control entity residing in system X.

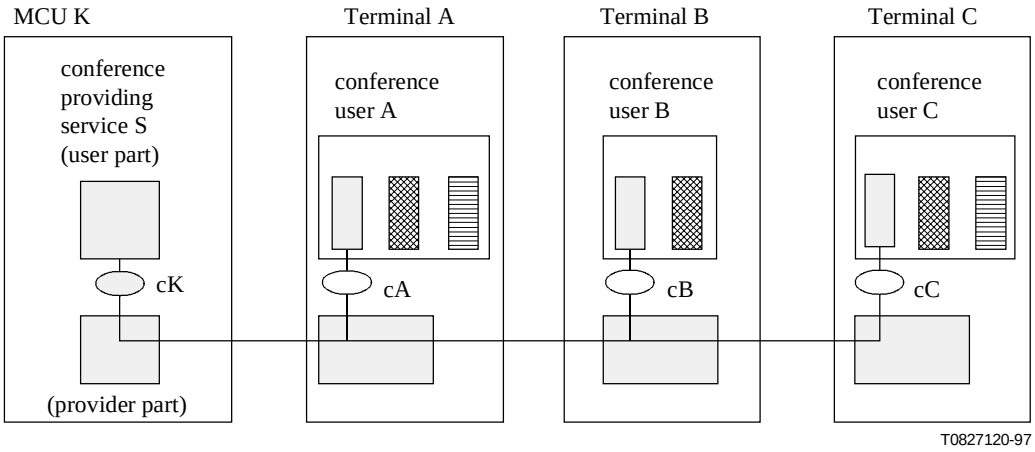


Figure 7/T.180 – Example configuration: Multipoint connection (cK, cA, cB, cC)

The address space of the control connection will comprise addresses of control entities of terminals and of other nodes (e.g. MCUs) participating in a conference. The connection may be identified by a list of addresses, or by the name of a list (e.g. a number), or by some other mechanism.

If the conference providing service S is not subdivided in a user part and a provider part (i.e. there is no service access point cK in the example above), then the services of S may be requested by identifying the node or the service in the control connection: (K, cA, cB, cC) or (S, cA, cB, cC) in the example above.

In general, a *control connection* serves for communication between control entities (residing in terminals or in other nodes) and between control entities and provider parts of conference providing services. As an example, creating a conference means establishing a control connection.

Figure 8 shows the multipoint connection (A-a1, B-a1, C-a1), which handles the communication between the entities of the application a1: entities of application a1, residing at different terminals, may communicate in a multipoint fashion with each other. X-ai is the name of the application-SAP (or endpoint of the SAP) identifying the application entity ai residing in system X.

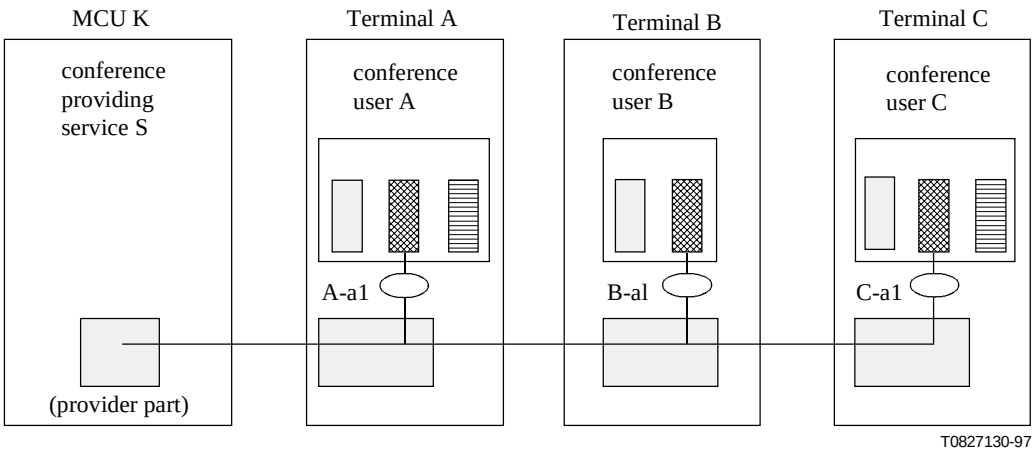


Figure 8/T.180 – Example configuration: Multipoint connection (A-a1, B-a1, C-a1)

In the same way, the multipoint connection (A-a2, C-a2) will handle the communication between the entities of the application a2.

Therefore, these connections are called application connections. Application connections rely on the conference creation process performed earlier.

The address space of an application connection will comprise addresses of application entities. The connection may be identified by a list of addresses, or by the name of a list (e.g. a number), or by some other mechanism.

In general, an *application connection* serves for communication between application entities.

An application connection provides the activities of a *session*. As an example, the members of a session may cooperate via the exchange of files interactively between them using the services of the data transfer phase of an application connection. If supported by the connection, a session may comprise the transfer of more than one data flow (e.g. file transfer) in parallel.

The identification of a connection should remain unchanged even in cases where the members participating in a session will change. Therefore, it might be convenient to identify a connection by a number rather than by listing the addresses of the actual participants.

Figure 9 shows the example configuration together with the three multipoint connections discussed above.

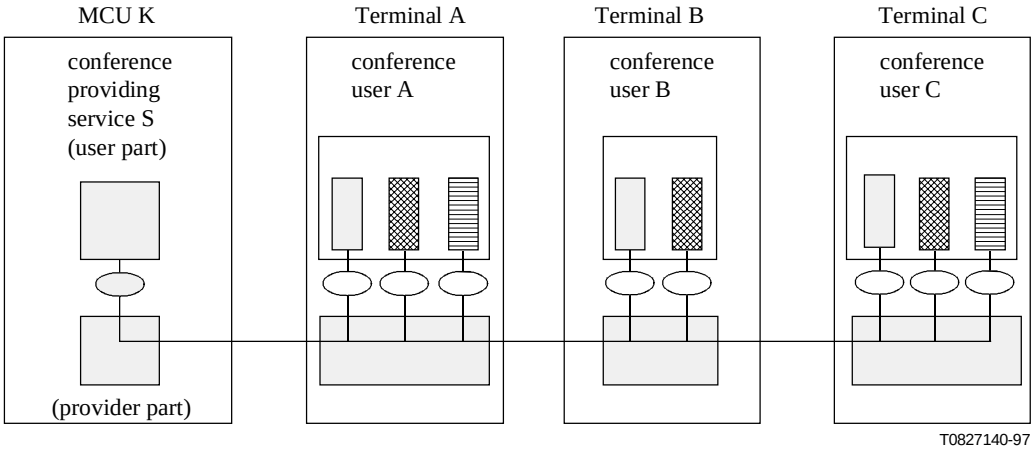


Figure 9/T.180 – Example configuration: Sketching all multipoint connections

As pointed out above, a conference may handle more than one connection. Conference control may be supplied by a separate connection (i.e. control connection). In this case,

– the X-CONNECT service at a control-SAP, identifying the control connection, will be used to establish or join a conference or to invite to a conference; and

– for each multipoint aware application, the X-CONNECT service at an application-SAP, identifying the specific application connection, will attach the application to the provider and user requirements will be negotiated.

Roughly spoken, the control connection is concerned with the management of the conference environment, and each application connection is concerned with the usage of the conference environment by a specific application.

If no control connection is used, conference control functions are added to application connections.

In any case, each multipoint connection is established by applying the X-CONNECT service. Having applied successfully this service, the connection is in the data transfer phase. The specific semantics of the X-CONNECT service and the services of the data transfer phase depend on the provider.

7.2 The state transition diagram as part of the model

The access to each connection (point-to-point or multipoint) is handled by the usage of the service primitives at appropriate service endpoints. This subclause specifies the service primitives and their sequences possible at some service endpoint (local view). The specification of the interdependencies between service primitives at different service endpoints accessing the same service (global view) is outside the scope of this Recommendation.

Figure 10 shows a state transition diagram, which presents the sequences of service primitives possible at a service endpoint. With this generic type of service any kind of CO OSI service may be accessed as well as CO-oriented non-OSI services such as video (video codecs) or other communication services (e.g. multipoint aware applications or conference systems).

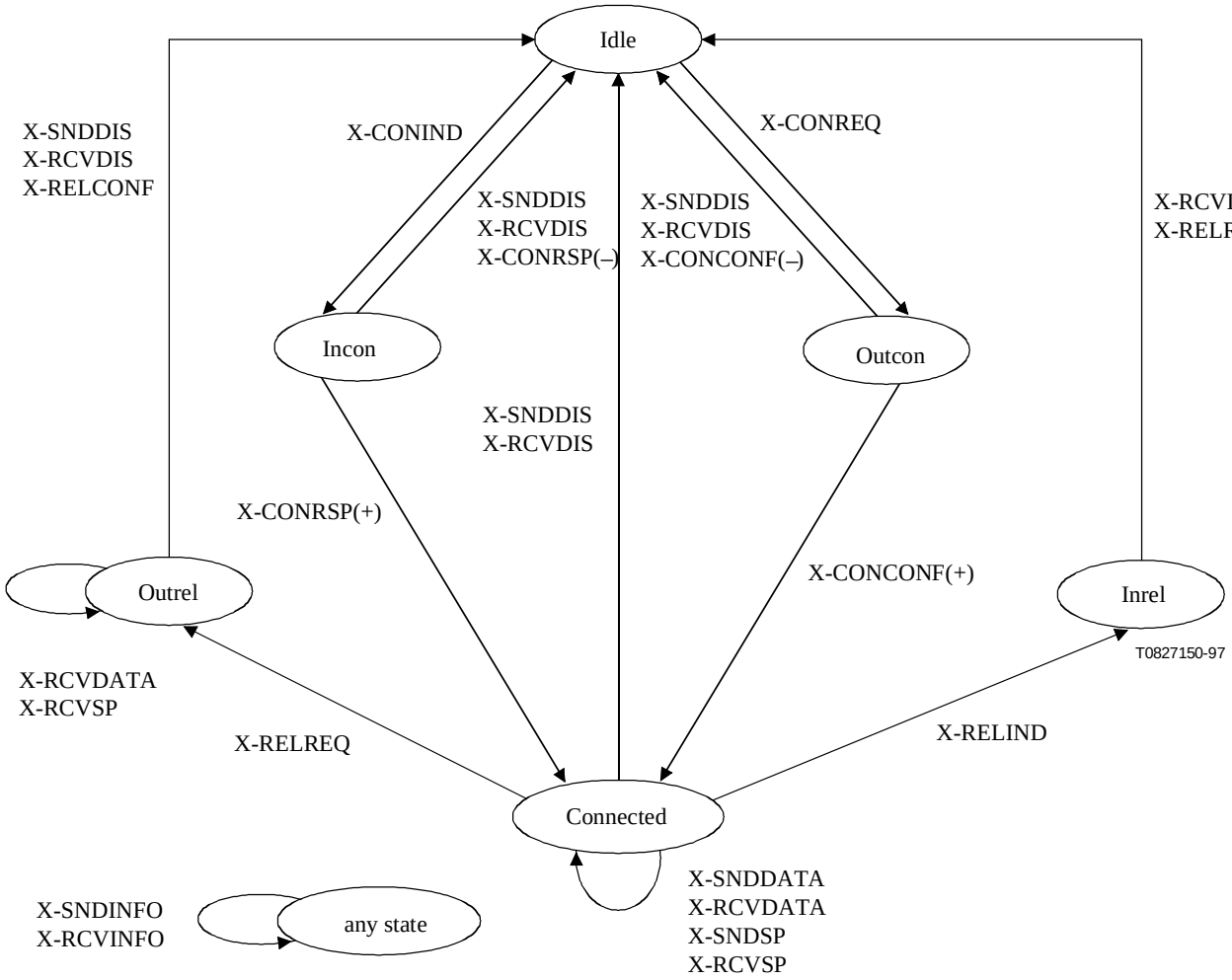


Figure 10/T.180 – Sequence of service primitives at a service endpoint

The semantics of the X-primitives are described below.

This diagram covers the connection establishment phase, the data transfer phase, and the connection release phase. Which services are made available to users depend on the underlying provider.

In the following, the service primitives are described which are supporting these three phases.

The connect service may comprise end-to-end confirmation using the request, the indication, the response, and the confirmation primitives in a peer-to-peer like manner (see Figure 5 for example), or it may comprise some other functionality (i.e. no peer-to-peer confirmation).

In general, applying the connect service will force the local user as well as the provider to switch to the data transfer phase of a connection. This service will address the remote user(s) and/or the provider.

During connection establishment phase (X-CONNECT service), either:

- e1 two users of the same service (i.e. a user and its peer) establish a connection between them, (one user is considered *active* and initiates the connection, while the other user is *passive* and waits for any other service user to request a connection); or
- e2 a user establishes a control connection to communicate in a control service defined fashion (e.g. to negotiate provider facilities); or
- e3 a user establishes an application connection (i.e. an application is connected to the provider), which then offers multipoint services to the application; or

- e4 a user establishes a connection combining e2 and e3 services.

The service primitives that are supporting the connection establishment phase are:

- X-CONREQ requests a connection to a remote service user (e1), or it requests access to some appropriate server to communicate in a server-defined fashion (e2), or it requests an application to be connected to the provider (e3), or it requests a combined service, consisting in e2 and e3 services. The called user's protocol address(es) or a service-related address is passed as an argument. Additional parameters may be passed as arguments too.
- X-CONCONF is used in conjunction with X-CONREQ to establish a connection or an access to the requested service. X-CONREQ does not wait for an answer from the called user or service. On receipt of the X-CONCONF primitive, the calling user may determine whether or not the request has been accepted by the remote user or by the server.
- X-CONIND indicates a connection request from a remote service user. It provides the calling user's protocol address, maybe some protocol-specific (or service-specific) parameters proposed by the caller, and the user data from the connect indication protocol data unit.
- X-CONRSP is initialized by the passive user to accept a connection or a service after an X-CONIND has been received. Protocol-specific (service-specific) parameters and user data may be submitted as arguments of the X-CONRSP. A negative response may or may not be provided.

Once, a connection has been established, data may be transferred (X-DATA service). There are service primitives to transfer data, and additional service primitives (X-SP service) that serve for different purposes, e.g. they may be used to change the set of capabilities which characterize the communication (e.g. change of QoS). Depending on the provider, transfer of parallel data flow may be supported by the connection. Parallel data may be transferred by the provider due to differing priorities. There are four primitives defined for use in the data transfer phase:

- X-SNDDATA enables the user to play the role of a source of user-relevant information. Playing this role data of any type and of any priority (if supported by the service provider) are sent over the connection to the communication partner(s). The data transfer between the peers is controlled by protocol-specific parameters that, usually, have been negotiated in the connection establishment phase. Additional protocol-specific parameters may be specified in the X-SNDDATA.
- X-RCVDATA enables the user to play the role of a sink of user-relevant information. Playing this role data of any type and of any priority are received over the connection which have been sent by the source. Beside the user data, X-RCVDATA may deliver protocol-specific parameters as additional information concerning the user data.
- X-SNDSP enables the user to transfer a protocol-specific service primitive, or other control information, or some other connection-related information to the provider and/or other user(s). Examples are the transfer of information between the applications (e.g. for the OSI Session Provider, the S-TOKEN-PLEASE rq. is transferred to the provider to request a data token from the active side), or the change of the set of capabilities which characterize the communication (e.g. for the H.320 service provider, the mode switch request primitive is passed to the provider to initiate a change of the current transmission mode). In the latter case, it depends on the

provider whether a service primitive is handled completely local (e.g. changing only parameter values of a local codec) or leads to a data transfer. The available service primitives are defined in the descriptions of the appropriate service providers.

- X-RCVSP enables the user to receive a protocol-specific service primitive, or other control information, or some other connection related information. The primitive may indicate the receipt of a protocol data unit or may be generated as result of some internal state changes or events in the local service provider. The primitives that may be returned are defined in the descriptions of the appropriate service providers.

Two ways are provided to release a connection: abortive release (X-DISCONNECT service) and orderly release (X-RELEASE service). The abortive release is a mandatory feature and to be supported by all XAPI service providers. The orderly release is an optional feature.

The abortive release may be invoked in the connection establishment phase or the data transfer phase. The abortive release takes effect immediately on request. Once an abortive release has been initiated there is no guarantee that data on the way between a user and its peer(s) will be delivered correctly. They may be lost. Service primitives supporting abortive release are:

- X-SNDDIS initiates an abortive release during the data transfer phase or rejects an incoming call during connection establishment. X-SNDDIS takes user data and service primitive parameters as arguments.
- X-RCVDIS may be used during data transfer and connection establishment. The parameters of the X-RCVDIS indicate the reason for the abortive release.

An orderly release may be invoked by either user in the data transfer phase only. The orderly release procedure allows a user and its peer(s) to gracefully release a connection and thus prevents the loss of data that may occur during an abortive release. Orderly release is a confirmed service. In an orderly release, protocol-specific parameters may be negotiated between the user and its peer(s). Conflicts are assumed to be solved by the provider. In the case of multipoint communication, the orderly release causes specific synchronization. The usage of the orderly release service in multipoint environments is for further study. Primitives supporting orderly release are:

- X-RELREQ initiates an orderly release during the data transfer phase. The function takes user data and service primitive parameters as arguments, but it is protocol dependent if they are supported by the service provider and will be transferred to the remote user(s) or not. The connection is not terminated before the release confirmation has arrived at the service endpoint and data or service primitives can be received from the endpoint while waiting for the release confirmation. Sending data is not possible in this state.
- X-RELCONF receives a release confirmation from the service endpoint. The primitive may contain user data and service primitive parameters. If a parameter negotiation has been initiated with the X-RELREQ, the negotiation results are contained as service primitive parameters of the release confirmation. The communication is finished with the X-RELCONF. A negative confirmation is not provided by the model.
- X-RELIND receives a release indication from the service endpoint. The primitive may contain the user data and service primitive parameters submitted with the X-RELREQ. The application has to respond to the indication with an X-RELRSP immediately. After reception of the release indication, no more data may be sent.

- X-RELRSP responds to a previously received orderly release indication. The primitive takes user data and service primitive parameters as arguments, but it is protocol dependent if they are supported by the service provider or not. If the remote user initiated a parameter negotiation with the release request, the application may respond to the proposals returned by the X-RELIND primitive. The final values for the negotiated parameters have to be specified as service primitive parameters in the X-RELRSP. The communication is finished with the X-RELRSP. Refusing the release with X-RELRSP(−) is not provided by the model.

General information services (e.g. information on conferences that are currently in existence) may be invoked by the user in any state (X-INFO service). In general, these services are not related to some specific connection.

- X-SNDINFO initiates an information request.
- X-RCVINFO may be used to receive information.

In Figure 10, there are six possible states for a service endpoint representing the connection establishment phase, the data transfer phase, and the connection release phase:

- Idle The service endpoint is active and the connection establishment phase may performed.
- Outcon The endpoint is engaged in active connection establishment. An outgoing call has been initiated and now a connect confirmation is awaited.
- Incon The endpoint is engaged in passive connection establishment. An incoming connect indication has been received and now a connect response is awaited from the user.
- Connected A connection has been established on this service endpoint which is now in the data transfer phase.
- Inrel An orderly release indication has been received on this endpoint and now a release response is awaited from the user.
- Outrel An orderly release has been initiated on this endpoint by the user and now an orderly release confirmation is awaited.

The states Inrel and Outrel are significant only for services that support orderly release.

8 Description of the XAPI

This clause introduces the architecture of the XAPI, the XAPI functions, and the relationship of the functions with respect to the model of communication, which was introduced in clause 7. XAPI offers point-to-point applications as well as multipoint-aware applications a homogeneous access mechanism to communication services.

8.1 XAPI in point-to-point and in multipoint environments

As stated in clause 6, the XAPI is a communication interface inside a terminal equipment. It allows a service user to communicate with its respective service provider. The location of the XAPI depends on the choice of the underlying provider.

Looking from the OSI Reference Model to audio or video applications (or to specific data applications), the functionality of some OSI layers is not needed.

These layers are empty in Figure 11.

The left part of Figure 11 shows an ATM service provider offering the "OSI Network Service" (data, or audio, or video) to the XAPI user. The right part of the figure shows an X.25 service provider together with an X.224 service provider offering the OSI transport service to the XAPI user.

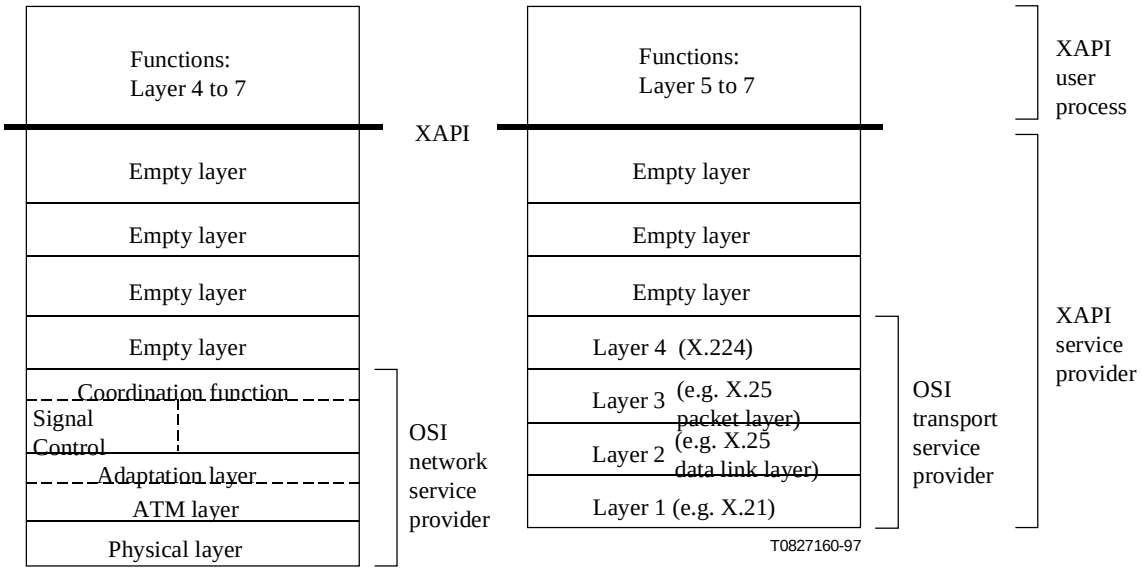


Figure 11/T.180 – XAPI Access to the OSI transport service provider and the OSI network service provider

In a conference environment, conference users and conference providers can be distinguished around the XAPI.

Figure 12 shows a Generic Model of a Conference platform together with the concept of separating conference control (using a control connection) and application functions (using application connections) from each other, as described in clause 7.

The conference provider shown in Figure 12 may be looked at as the local part of the conference provider (i.e. the restriction of the provider functionality to a system), as shown in Figure 4.

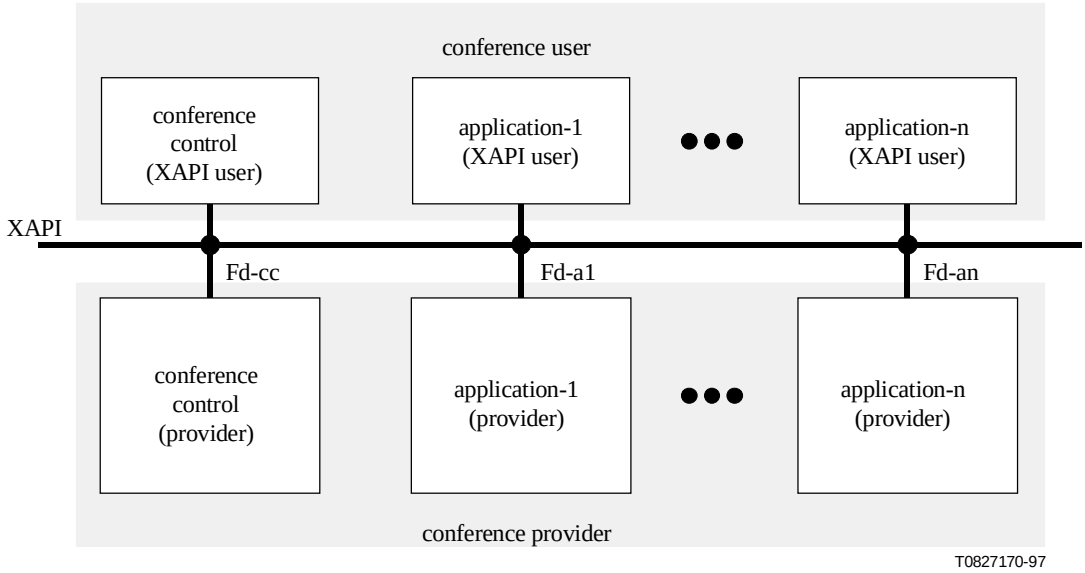


Figure 12/T.180 – Generic Model of a Conference platform: Separating conference control and application functions

The functionality of the providers shown in Figure 12 depends on the specific local part of the conference provider.

From the XAPI point of view this separation is supported by the creation of:

- a (conference) service endpoint (named Fd-cc in Figure 12) specifying a local link between the XAPI user part of the conference control and its provider part; and
- a (application) service endpoint for each application process (named Fd-a1 to Fd-an in Figure 12) specifying a local link between the user part of an application process and its application protocol instance (provider part).

The interaction of an XAPI user and the corresponding provider at a specific Fd is described by the sequences of XAPI service primitives at that specific service endpoint. For peer-to-peer as well as for multipeer communication (e.g. for conferencing), the same XAPI functions apply for each service endpoint:

– A service endpoint is initialized using the x_open() and the x_bind() functions.

– Connection establishment is supported by the x_conreq() and the x_conconf() functions (or by the x_conind() and the x_conrsp() functions).

– The data transfer phase is supported by the x_snddata() and x_rcvdata(), and/or by the x_sndsp() and x_rcvsp() functions.

– Connection release is supported by the x_snddis(), or by the x_rcvdis() functions (or by the functions of the orderly release).

– The service endpoint is de-initialized using the x_unbind() and the x_close() functions.

– The functions x_sndinfo() and x_rcvinfo() may be used for transferring appropriate information between user and provider via the XAPI.

NOTE – For conference control as well as for other applications, the connect service may comprise end-to-end confirmation [using the request, the indication, the response, and the confirmation functions in a peer-to-peer like manner (see Figure 3 for example)], or it may comprise minor functionality (i.e. no peer-to-peer confirmation).

According to the Generic Model of a Conference platform (see Figure 12), there is a separate user instance (accessing the provider at the service endpoint Fd-cc) that controls the conference, thus unburdening these functions from all other conference applications. If such an instance does not exist, conference control (if any) has to be handled by some (or by all) multipoint-aware applications. This does not affect the usability of the XAPI: as for other cases, the XAPI user access to some appropriate provider has to be specified in a separate part of Appendix I.

8.2 XAPI functions and the corresponding state transition diagram

The XAPI functions may be divided in two classes: the class of communication-related functions and the class of not communication-related functions. Communication-related functions get their specific semantics from the service primitives described in clause 7.

The communication-related XAPI functions together with the corresponding service primitives are listed in Table 1.

Table 1/T.180 – Communication-related XAPI functions

XAPI function	X-primitive
x_conreq()	X-CONREQ
x_conind()	X-CONIND
x_conrsp()	X-CONRSP
x_conconf()	X-CONCONF
x_snddata()	X-SNDDATA
x_rcvdata()	X-RCVDATA
x_sndsp()	X-SNDSP
x_rcvsp()	X-RCVSP
x_snddis()	X-SNDDIS
x_rcvdis()	X-RCVDIS
x_relreq()	X-RELREQ
x_relind()	X-RELIND
x_relrsp()	X-RELRSP
x_relconf()	X-RELCONF
x_sndinfo()	X-SNDINFO
x_rcvinfo()	X-RCVINFO

The XAPI functions that are not related to communication may be subdivided in two classes: the class of functions supporting the initialization and de-initialization phase, and the class of utility functions.

Table 2 shows the functions supporting initialization and de-initialization and a specific function, indicating explicitly that the local resource management has finished all operations concerning the connection release phase of a connection.

Table 2/T.180 – XAPI functions supporting initialization/de-initialization

x_open()	create and open a service endpoint
x_bind()	activate a service endpoint
x_unbind()	deactivate a service endpoint
x_close()	close a service endpoint
x_rcvend()	receive end indication from service provider

XAPI utility functions do not change the state of a service endpoint. Table 3 shows a list of the XAPI utility functions (sorted ascending in alphabetical order).

Table 3/T.180 – List of utility functions

x_b2c()	select a value of type character string out of a buffer
x_b2l()	select a value of type long out of a buffer
x_c2b()	write a value of type character string into a buffer
x_chexmod()	change execution mode
x_error()	produce an error message
x_getinfo()	get protocol-specific information from the service provider
x_look()	look for the current event on a service endpoint
x_l2b()	write a value of type long into a buffer
x_optmgmt()	manage options for a service endpoint
x_rcverror()	retrieve error indication from a service provider
x_strerror()	produce an error message string
x_sync()	synchronize data structures of the XAPI library with the information from the underlying service provider

Figure 13 shows a state transition diagram that presents the sequences of XAPI functions possible at a service endpoint. XAPI utility functions are not shown. Except for state X_UNINIT, they may be called in any state. Which functions are made available via the XAPI depends on the specific providers.

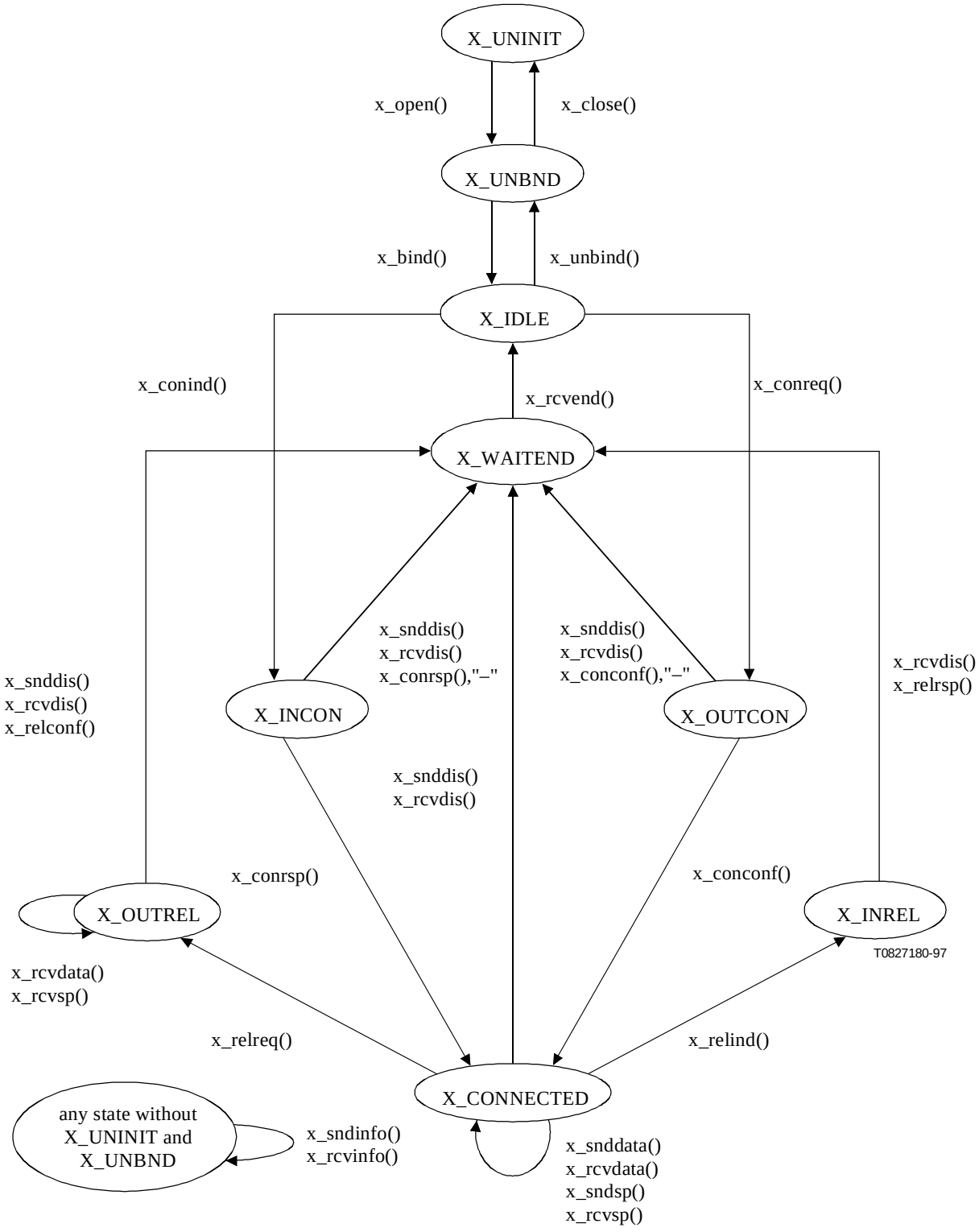


Figure 13/T.180 – Sequence of XAPI function calls (fragmentation of user data is not shown)

From the viewpoint of the mapping defined in Table 1, the structure shown in Figure 13 may be looked at as a refinement of the structure shown in Figure 8. The refinement preserves all the sequences of service primitives which are defined in Figure 8.

For each service endpoint, the XAPI has nine possible states:

– X_UNINIT The service endpoint is not initialized. This is the initial and the final state.

– X_UNBND The service endpoint is initialized but not activated.

– X_IDLE The service endpoint is active and the connection establishment phase may performed.

– X_OUTCON The endpoint is engaged in active connection establishment. An outgoing call has been initiated and now a connect confirmation is awaited.

– X_INCON The endpoint is engaged in passive connection establishment. An incoming connect indication has been received and now a connect response is awaited from the user.

– X_CONNECTED A connection has been established on this service endpoint which is now in the data transfer phase.

– X_INREL An orderly release indication has been received on this endpoint and now a release response is awaited from the user.

– X_OUTREL An orderly release has been initiated on this endpoint by the user and now an orderly release confirmation is awaited.

– X_WAITEND The connection established on this endpoint has been released. Now the endpoint is waiting for an end indication from the service provider, which indicates that the service provider is ready to establish a new connection.

The states X_INREL and X_OUTREL are significant only for services that support orderly release.

Figure 14 shows an example sequence for the communication between two or more users. Fd is the name of the service endpoint, which is created and activated by the usage of the x_open() and x_bind() function calls. After having performed this phase successfully, a connection between the user and its peer(s) is established [x_conreq() and x_conconf() functions]. In the Data transfer phase data are exchanged between the users [x_snddata(), x_rcvdata()]. Afterwards, the connection is released [x_snddis()], the x_rcvend() function is performed, and the service endpoint Fd is deactivated [x_unbind()] and finally destructed [x_close()].

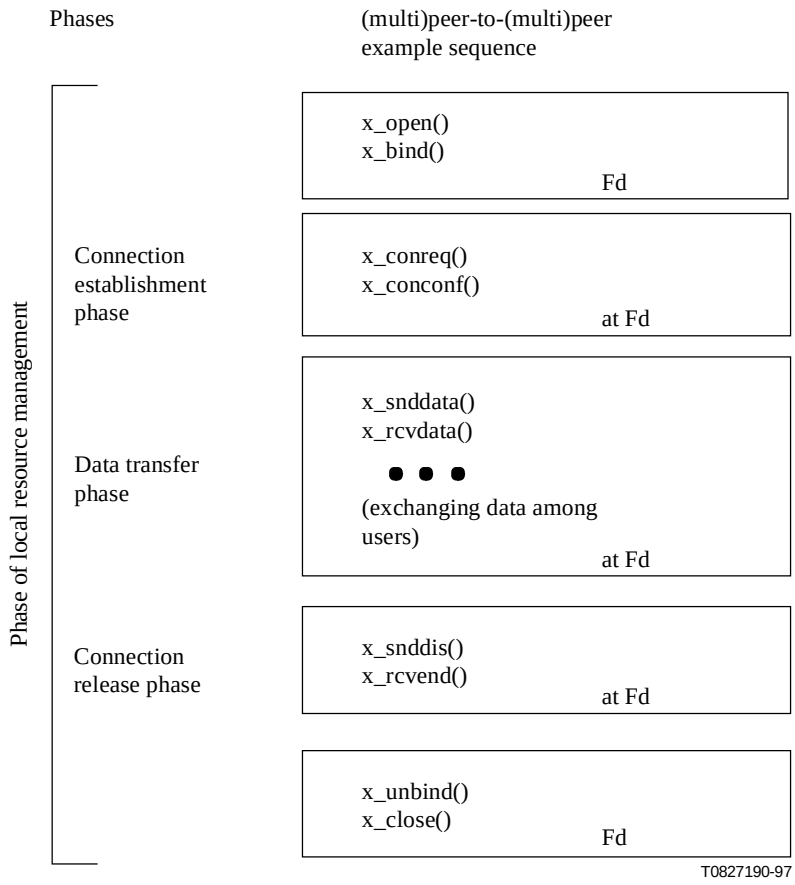


Figure 14/T.180 – Example sequence of XAPI function calls

Figure 15 shows an example sequence of XAPI function calls for conference purposes together with an example sequence for peer-to-peer communication.

The sequence at the right side consists of two parts of activities, namely an activity which handles participation in a conference (service endpoint Fd-cc and accessory primitives) and an activity which handles applications running in a conference (service endpoint Fd-a1 and accessory primitives). This sequence also applies even to the case of a conference with only two users being involved (point-to-point configuration).

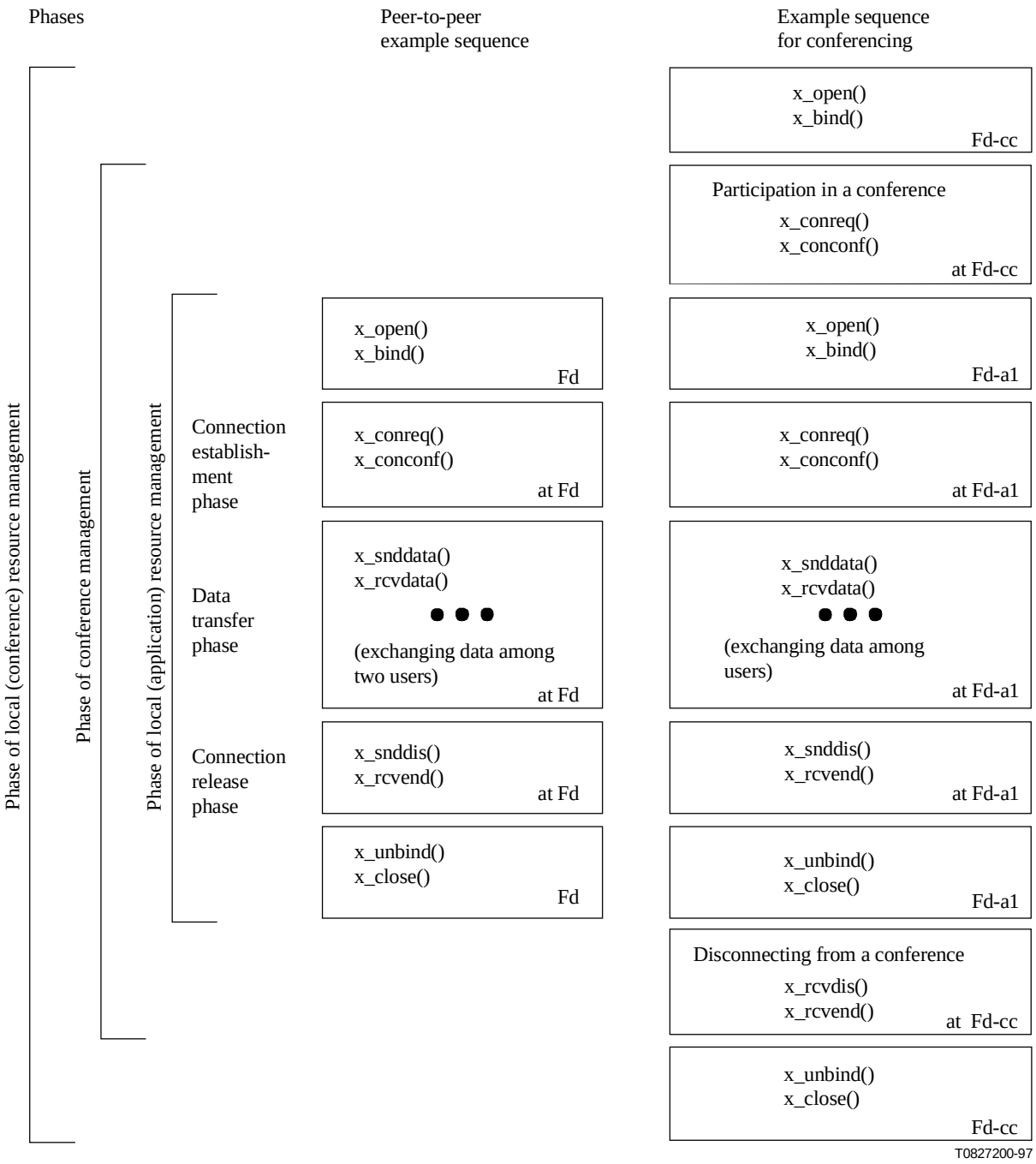


Figure 15/T.180 – Two example sequences of XAPI function calls for peer-to-peer and for conferencing

As can be seen from the example, the latter sequences of function calls may coincide with those defined in the peer-to-peer case.

Conference management services (see Figure 15) may be required at any instant of time after having created a conference (e.g. adding an additional node to an existing conference). So the phase of conference management will encompass the inner phases:

– local application resource management;
– connection establishment;
– data transfer; and
– connection release.

Users may join or leave a running conference and applications may be added to or removed from a conference. These operations do not only affect the phases connection establishment, data transfer, and connection release, but will require operations of the local application resource management too. Last, but not least, the phase of local conference resource management may be influenced by these operations and, therefore, it may encompass all other phases.

9 XAPI functions

The XAPI functions may be divided in two classes: the class of communication-related functions and the class of not communication-related functions. The communication-related functions are described in 9.2. The XAPI functions that are not related to communication may be subdivided in two classes: the class of functions supporting the initialization and de-initialization phase, described in 9.3.1, and the class of utility functions, described in 9.3.2.

A list of the XAPI functions is given in 8.2. Figure 13 shows the state transition diagram.

Annex A contains an IDL description of the XAPI. A list of error codes that may be returned by the XAPI functions is given in Annex B.

General buffer format

Figure 16 shows the general buffer format.

Several entries can be concatenated in one buffer. Three types of buffers are used: Option, service primitive parameter, or address buffers. The value of an entry immediately follows the entry header with no padding or alignment characters in between. The header consists of three members: *len*, *name*, and *status*.

The *len* field gives the total length of the entry. This comprises the entry header and the entry value.

The *name* field identifies the entry.

The *status* field is used to indicate success or failure of an option negotiation (see function *x_optmgmt*); is it not used for service primitive parameter and address buffers.

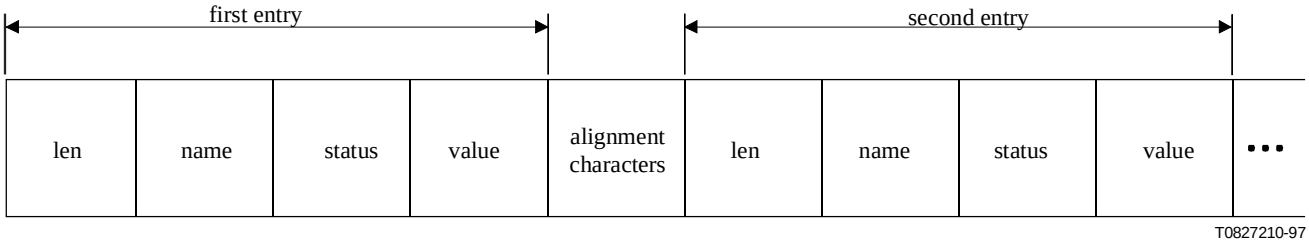


Figure 16/T.180 – General buffer format

For reading and writing buffer entries, the functions *x_b2c()*, *x_b2l()*, *x_c2b()*, and *x_l2b()* are helpful. The reading functions *x_b2c()* and *x_b2l()* will return the *len*, *name*, *status*, and *value* of the entry and an index to the next entry. The writing functions *x_c2b()* and *x_l2b()* take *len*, *name*, and *value* as arguments, write them into the buffer and return an index to the next entry.

Protocol options

The protocol options are introduced in XAPI as a flexible mechanism to exchange protocol information between the user and the service provider. It enables the user to express special wishes about the settings of some protocol parameters and it allows the service provider to pass information about the protocol parameters used to the user. Protocol options are used to control the general behaviour of a protocol module or to temporarily override some configuration parameters. Each option is specific for a certain protocol module. They must not be confounded with service primitive parameters.

The XAPI function *x_optmgmt()* is the only one that conveys options between the service user and the service provider. Except for the XAPI-level options described below, all options are passed transparently between provider and user. The XAPI is the carrier only. The options are interpreted, processed, managed, and stored by the provider.

The service provider has a default value for each option it supports. These defaults are sufficient for the majority of communication relations. Hence, a user should only request options actually needed to perform the task and leave all others at their default value.

Each option is characterized by several attributes.

Option attributes

Connection-related options are intimately related to a particular connection in a connection-mode service. Connection-related options are usually negotiated between the two peer entities during connection setup. Therefore, some ancillary information is transferred from the calling entity to the called entity and back in most cases. The interpretation and further processing of this information is protocol-dependent.

An option can have purely local relevance. *Local options* are negotiated solely between the service user and the local service provider. The remote entity is neither informed about the settings of local options nor influenced by these options.

Options that are not connection-related do not contain information destined for the remote service user. Some can have only local relevance; others have external relevance as they influence the transmission of data.

An option can either or not be an *absolute requirement*. This is explicitly defined for each option. An absolute requirement must not be diminished or changed by the service provider. In some cases an option is partly absolute.

An option can be restricted to *read-only access* in some, or all, states of the service endpoint.

The *scope* of an option defines the period of time during which the option is in effect. The lifetime of an option is the time the option has a value assigned and can be accessed by the user. The lifetime is restricted to the time the service provider is accessible through a service endpoint. After the return of *x_open()*, all options are set to their default values. They will lose their values when the last endpoint accessing the service provider is closed. Do not confound the scope of an option with its lifetime. The scope is limited to the time the option is in effect.

There are two classes of scope. Each option belongs to one and only one of these classes. The widest scope possible is called *permanent scope*. It comprises the whole lifetime of the option. The scope of connection-related options is the time the connection exists. Some connection-related options have an even smaller scope as they are restricted to the data transfer phase of the connection. They come out of scope when connection release is initiated, i.e. the service endpoint leaves the state X_CONNECTED. Options retain their values while they are out of scope. The next time they come into scope, they will start with the old value, which may be different from the default value.

Option negotiation

A service user can initiate an option negotiation with a call to *x_optmgmt()* with the NEGOTIATE flag set in the *flags* field of the input parameter *req*.

The negotiation rules depend on whether an option is an absolute requirement or not. If the proposed option value is legal and an absolute requirement, the following two outcomes are possible:

The negotiation succeeds and the option is set to the required value; *x_optmgmt()* successfully returns and reports SUCCESS in the *status* field of the option header.

The negotiation is rejected if the option is supported but the proposed value cannot be accepted. *x_optmgmt()* will successfully return in this case, but report FAILURE in the *status* field of the option header.

If the proposed option value is legal but not an absolute requirement:

The negotiated value is of equal or less quality than the proposed value. *x_optmgmt()* will indicate this in the *status* field of the output option. SUCCESS indicates that the proposed value could be negotiated; PARTSUCCESS indicates that the proposed value was diminished.

If the *name* field of the header is unknown to the protocol module selected by *level*, or if the option is not supported at all, the function *x_optmgmt()* succeeds, but reports NOTSUPPORT in the *status* field of this option on return.

Not all options are independent from one another. A requested option might conflict with the value of another option. Therefore, options that depend on one another must not be negotiated independently, but have to be submitted all together in one *x_optmgmt()* call to avoid temporary inconsistencies or conflicts. Thus conflicts can be detected at negotiation time. If only one of two or more dependent options shall be set to a new value, all the dependent options have to be negotiated in one *x_optmgmt()* call, even if their values need not to be changed. If a conflict between dependent options is detected by the service provider, *x_optmgmt()* will successfully return, but report FAILURE in the *status* field of the concerned options.

If the service user submits multiple options in one call and one option is not supported by the service provider, all other options within the buffer are not influenced by this and the legal options are negotiated as usual.

A violation of access rights occurs when the user tries to negotiate an option which is restricted to read-only access. The function *x_optmgmt()* will return successfully in this case, but the option is not changed and READONLY is returned in the *status* field of this option.

If the service user submits multiple options in one call and one of them causes a violation of access rights, all other options within the buffer are not influenced by this and the legal options are negotiated as usual.

As *x_optmgmt()* handles options for one module only, multi-module conflicts cannot be detected at negotiation time. It is left for the user to avoid conflicts between options of different protocol modules.

XAPI-level options

There is only one XAPI-level option: the trace option O_TRACE. It switches on/off traces in XAPI functions and protocol modules. The *level* argument in the *x_optmgmt()* call selects the target module. The XAPI functions are selected with *level* XAPI_LEVEL. For protocol modules, the protocol module identifier has to be used as *level*.

The option value specifies the trace level which controls the amount of traces generated.

The following trace levels are defined:

– OV_NOTRACE switches all traces off (default value);

– OV_BUFFTRACE enables the buffer trace for received and sent PDUs; all trace messages are disabled;

– OV_ERR enables error trace messages and disables buffer traces;

– OV_ERR_BUF enables error trace messages plus buffer traces;

– OV_WRN enables error and warning trace messages; buffer traces are disabled;

– OV_WRN_BUF enables error and warning trace messages plus buffer traces;

– OV_INF enables error, warning, and information trace messages; buffer traces are disabled;

– OV_INF_BUF enables error, warning, and information trace messages plus buffer traces.

Buffer traces and trace messages cannot be switched independently. A change in the O_TRACE option value always effects both categories. The combinations of the different levels of message traces and buffer traces may be calculated by the user with the bitwise-OR operator.

Option attributes

The option O_TRACE

– is an absolute requirement;

– is not connection related;

– has only local meaning;

– has permanent scope.

The way to access the generated trace data depends on the base operating system.

Note that the error messages generated by the service provider are not affected by the trace level. These error messages are generated whenever an error condition is detected. Usually, they will be written to a system-specific error log file.

9.1 Conventions

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

For each XAPI function, a list of parameters is specified.

The meaning of the functions and parameters is described for two types of providers:

- point-to-point (abbreviation p-p);
- multipoint (mp) with the two types of endpoints being supported: the conference control (abbreviation mp-c) and the application endpoint (mp-a).

If there is a different meaning for the function or parameters of the supported endpoints of the provider types, this is noticed. If there is only one description for a function or parameter, it is meaningful for all types of the supported endpoints.

In a multipoint environment, all communication-related functions at application endpoints are only available if the conference control endpoint is in the active state.

Different providers may cause the specification of provider-specific parameters and/or sets of values for these parameters.

For each specific provider, a separate part of Appendix I describes the XAPI functions, their parameters, and values.

The description of the XAPI functions in the following subclauses is in an alphabetical order.

9.2 Communication-related functions

For each XAPI function, the relation between the function and the accessory service primitive is defined. Afterwards, a description of the parameters of the XAPI function is given.

9.2.1 X-CONCONF/x_conconf

Description: This primitive enables the XAPI user to determine the status of a previously sent connect request and is used in conjunction with X-CONREQ. In the case of a positive connect confirm, the service endpoint changes to state "Connected". If the service provider supports the capability of negative connect responses (confirm), then an appropriate information may be returned to indicate that the communication was not accepted. In this case, the service endpoint will be in state "Idle".

The meaning of a positive X-CONCONF is for:
p-p: a connection between peer entities is established
mp-c: a control connection to a conference service provider and to other conference members is established
mp-a: the access of an application to a session in an active conference is established

Function in XAPI:

Name: x_conconf(fd, call, xerror)

Parameters:

fd: Identification of the local service endpoint where the active connection establishment has been initiated with *x_conreq*

call: The parameter *call* contains information related to the newly established connection. It is structured in the following parameters:

- *address*: contains address information
  p-p: specifies the responding protocol address, i.e. the address of the responding service user
  mp-c: specifies the address of the conference service provider and may contain a list of active conference members
  mp-a: not used
- *parameter*: contains the protocol specific parameters
  p-p: protocol-specific parameters of the peer-to-peer connection establishment
  mp-c: Conference-specific parameters
  mp-a: session and application-specific parameters
- *user_data*: contains optional data
  p-p: contains any user data that are received from the responding user
  mp-c: additional conference-specific information received from the conference service provider (e.g. information about authentication, billing) and other conference members
  mp-a: additional information of the local application service provider
- *sequence*: no meaning
- *flags*: to indicate a local segmentation of information at the interface, or a negative connect confirmation

xerror: Error codes

Return value: indicates successful completion or failure

9.2.2 X-CONIND/x_conind

Description: X-CONIND enables a passive user to initiate connection establishment. This service primitive may be diverted from an X-CONREQ. After the receipt of this primitive, the endpoint changes to state "Incon".

The meaning for:
p-p: indicates the passive user to establish a connection between the active user and itself
mp-c: invitation to conference
mp-a: indicates an application that it may use an application connection belonging to a conference to facilitate communication between all or a group of conference members

Function in XAPI:

Name: x_conind(fd, call, xerror)

Parameters:

fd: Identification of the local service endpoint where the passive connection establishment is initiated with *x_conind*

call: The parameter *call* contains information related to the new connection which will be established. It is structured in the following parameters:

- *called_addr*: contains address information of the called user
- *calling_addr*: contains address information of the calling user
- *parameter*: contains the protocol-specific parameters
  p-p: protocol specific parameters transferred in the connect indication primitive
  mp-c: Conference specific parameters
  mp-a: session- and application-specific parameters
- *user_data*: contains optional data
  p-p: contains any user data transferred from the calling to the called user
  mp-c: additional conference-specific information sent to the conference service provider or other conference members (e.g. information about authentication, billing)
  (mp-a): additional information to the application
- *sequence*: local unique identification number of the connect indication
- *flags*: to indicate a local segmentation at the interface or a negative connect confirmation

xerror: Error codes

Return value: indicates successful completion or failure

9.2.3 X-CONREQ/x_conreq

Description: This primitive enables a user to initiate active communication establishment. The communication establishment is initialized and the endpoint changes to state "Outcon".

The meaning for:
p-p: to establish a connection to another peer entity
mp-c: to establish a control connection to a conference service provider and to other conference members

mp-a: the application wants to use an application connection belonging to a conference to facilitate communication between all or a group of conference members

Function in XAPI:

Name: x_conreq(fd, sndcall, xerror)

Parameters:

fd: The identification of the local service endpoint where the communication will be established

sndcall: The parameter *sndcall* specifies information needed by the service provider to establish the communication. It is structured in the following parameters:

- *address*: contains address information
  p-p: specifies the destination protocol address, i.e. the address of the called service user
  mp-c: specifies the address of the conference service provider and may contain a list of conference members to be invited
  mp-a: not used
- *parameter*: contains the protocol-specific parameters that should be used in connection establishment
  p-p: protocol-specific parameters needed for peer-to-peer connection establishment
  mp-c: Conference-specific parameters
  mp-a: session- and application-specific parameters
- *user_data*: contains optional data
  p-p: contains any user data that shall be sent to the called user with the connect request
  mp-c: additional information for the conference service provider or other conference members
  mp-a: additional information to the local session service provider
- *sequence*: no meaning
- *flags*: to indicate a local segmentation at the interface

xerror: Error codes

Return value: indicates successful completion or failure

9.2.4 X-CONRSP/x_conrsp

Description: This primitive is issued by a service user to respond to a previously received connect indication. The communication establishment is now complete for the passive user and the endpoint changes to state "Connected". The service provider may or may not support the capability to send negative connect responses. In the case of a negative response, no communication is established and the service endpoint will be in state "Idle".

The meaning is for:
p-p: the passive user responds to the indication to establish a connection
mp-c: the member responds to an invitation to a conference
mp-a: the application responds to the invitation that it may use an application connection belonging to a conference to facilitate communication between all or a group of conference members

Function in XAPI:

Name: x_conrsp(fd, resfd, call, xerror)

Parameters:

fd: The parameter *fd* identifies the service endpoint where the connect indication arrived

resfd: *Resfd* specifies the local service endpoint where the connection is to be established. A service user may accept a connection on either the same, or on a different, local service endpoint than the one on which the connect indication arrived. Before the connection can be accepted on the same endpoint (*resfd = fd*), the user must have responded to any previous connect indications received on that service endpoint. If a different service endpoint is specified (*resfd ≠ fd*), then both endpoints must refer to the same service provider.

call: The parameter *call* contains information related to the new connection which will be established. It is structured in the following parameters:

- *address*: contains address information
  p-p: specifies the responding protocol address, i.e. the address of the responding service user
  mp-c: specifies the address of the conference service provider and may contain a list of active conference members
  mp-a: specifies all or a group of the actual in the session active conference members (session-members)
- *parameter*: contains the protocol-specific parameters
  p-p: protocol-specific parameters of the peer-to-peer connection establishment
  mp-c: Conference-specific parameters
  mp-a: session- and application-specific parameters
- *user_data*: contains optional data
  p-p: any user data that are received from the responding user
  mp-c: additional conference-specific information received from the conference service provider (e.g. information about authentication, billing) and other conference members
  mp-a: additional information to be passed to the local session service provider
- *sequence*: local unique identification number which corresponds to the connect indication
- *flags*: to indicate a local segmentation at the interface or a negative connect confirmation

xerror: Error codes

Return value: indicates successful completion or failure

9.2.5 X-RCVDATA/x_rcvdata

Description: This primitive offers the user either normal or expedited data or data with a specific priority. This service primitive does not change the state of the service endpoint.

The meaning is for:
p-p: the user receives data from the peer user
mp-c: the conference member receives conference-specific information from another conference member
mp-a: the session member receives data or application-specific information from another session member

Function in XAPI:

Name: x_rcvdata(fd, data, flags, xerror)

Parameters:

fd: Identification of the service endpoint

data: The parameter *data* contains the received data and related protocol-specific information. It is structured in the following parameters:

– *parameter*: additional information about the data
p-p: service primitive parameters which provide additional information about the user data
mp-c: additional information about the conference data (e.g. identification of the sending conference member)
mp-a: additional information about the session data (e.g. identification of a dataflow, if more than one dataflow is provided in the session; the identification of the sending session member)

– *data*: the received data

flags: indication of a local segmentation at the interface and the priority of the data
p-p: indication of expedited data
mp-c: priority of the data
mp-a: priority of the data

xerror: Error codes

Return value: indicates successful completion or failure

9.2.6 X-RCVDIS/x_rcvdis

Description: This primitive informs the user that a disconnect condition has occurred. A disconnect is an abortive release. The service primitive X-RCVDIS indicates the end of the data transfer phase. The endpoint changes to state "Idle".

The meaning is for:
p-p: the end of the connection
mp-c: no longer member in the conference
mp-a: no longer member in the session

Function in XAPI:

Name: x_rcvdis(fd, disc, xerror)

Parameters:

fd: Identification of the service endpoint

disc: The parameter *disc* contains disconnect related information. It is structured in the following parameters:

– *user_data*:
p-p: contains user data of the other user
mp-c: contains user data of the conference providing service perhaps created by another conference member
mp-a: contains user data of another session member

– *parameter*: contains service primitive parameters about the reason for disconnecting

– *origination*: specifies the origin of the disconnect

– *sequence*: local unique identification number which corresponds to a connect indication. It is only meaningful in the case where the indication is immediately followed by the disconnect.

xerror: Error codes

Return value: indicates successful completion or failure

9.2.7 X-RCVINFO/x_rcvinfo

Description: The X-RCVINFO primitive is used to receive appropriate information from the provider. This service primitive does not change the state of the service endpoint. It is allowed in all states except the states "Uninit" and "Unbnd".

Function in XAPI:

Name: x_rcvinfo(fd, level, info, par, flags, xerror)

Parameters:

fd: Identification of the service endpoint

level: Identification of the protocol module which generated the service primitive. This is one of the protocol module IDs which are available

info: Identification of the name of the protocol-specific service primitive
p-p: not used
mp-c: e.g. Information services for the conference
mp-a: e.g. Information services for the session

par: contains the parameters and user data of the service primitive

– *parameter*: contains the parameters of the service primitive
p-p: not used
mp-c: e.g. parameters of the conference-specific information service
mp-a: e.g. parameters of the session-specific information service

– *data*: contains the user data of the service primitive

flags: indication of a local segmentation at the interface

xerror: Error codes

Return value: indicates successful completion or failure

9.2.8 X-RCVSP/x_rcvsp

Description: The X-RCVSP primitive is used to receive a protocol-specific service primitive. This service primitive does not change the state of the service endpoint. It is only allowed in the states "Connected" and "Outrel".

Function in XAPI:

Name: x_rcvsp(fd, level, spname, sp, flags, xerror)

Parameters:

fd: Identification of the service endpoint

level: Identification of protocol module which generated the service primitive. This is one of the protocol module IDs which are available

spname: Identification of the name of the protocol-specific service primitive
p-p: *(* e.g. in the OSI-session protocol S-TOKEN-PLEASE service)
mp-c: Conference-specific services
mp-a: Session-specific services

sp: contains the parameters and user data of the service primitive

- *parameter*: contains the parameters of the service primitive
  p-p: contains the parameters of the service primitive
  mp-c: e.g. identification of the sending conference member
  mp-a: e.g. identification of a dataflow; if more than one dataflow is provided in the session, the identification of the sending session member
- *data*: contains the user data of the service primitive

flags: indication of a local segmentation at the interface

xerror: Error codes

Return value: indicates successful completion or failure

9.2.9 X-RELCONF/x_relconf

Description: The X-RELCONF primitive is used to receive the confirmation for a previously requested orderly release. The endpoint changes to state "Idle". This primitive is only used in a point-to-point configuration. The use in a multipoint environment is for further study.

Function in XAPI:

Name: x_relconf(fd, rel, xerror)

Parameters:

fd: Identification of the service endpoint

rel: contains protocol-specific parameters and the user data

- *parameter*: contains the release confirmation service primitive
- *user data*: contains user data of the remote peer entity
- *flags*: indication of a local segmentation at the interface

xerror: Error codes

Return value: indicates successful completion or failure

9.2.10 X-RELIND/x_relind

Description: The X-RELIND primitive is used to receive a release indication. This indicates the end of the data transfer on this connection to the user. After reception of a release indication, the user may not send any more data over the connection. The endpoint changes to state "Inrel". This primitive is only used in a point-to-point configuration. The use in a multipoint environment is for further study.

Function in XAPI:

Name: x_relind(fd, rel, xerror)

Parameters:

fd: Identification of the service endpoint

rel: contains protocol-specific parameters and the user data

- *parameter*: contains the release indication service primitive
- *user data*: contains user data of the remote peer entity
- *flags*: indication of a local segmentation at the interface

xerror: Error codes

Return value: indicates successful completion or failure

9.2.11 X-RELREQ/x_relreq

Description: The X-RELREQ primitive enables a user to initiate an orderly release of an existing connection and indicates to the service provider that there is no more data to send. The endpoint changes to state "Outrel". However, a user may continue to receive data until the confirmation for the orderly release request has arrived at the service endpoint. This primitive is only used in a point-to-point configuration. The use in a multipoint environment is for further study.

Function in XAPI:

Name: x_relreq(fd, rel, xerror)

Parameters:

fd: Identification of the service endpoint

rel: contains protocol-specific parameters and the user data
- *parameter*: contains the release request service primitive
- *user data*: contains user data
- *flags*: indication of a local segmentation at the interface

xerror: Error codes

Return value: indicates successful completion or failure

9.2.12 X-RELRSP/x_relrsp

Description: The X-RELRSP primitive responds to a previously received release indication. The endpoint changes to state "Idle". This primitive is only used in a point-to-point configuration. The use in a multipoint environment is for further study.

Function in XAPI:

Name: x_relrsp(fd, rel, xerror)

Parameters:

fd: Identification of the service endpoint

rel: contains protocol-specific parameters and the user data
- *parameter*: contains the release response service primitive
- *user data*: contains user data
- *flags*: indication of a local segmentation at the interface

xerror: Error codes

Return value: indicates successful completion or failure

9.2.13 X-SNDDATA/x_snddata

Description: This primitive is used to send either normal or expedited data or data with a specific priority. This service primitive does not change the state of the service endpoint.

The meaning is for:
p-p: the user sends data to the peer user
mp-c: the conference member sends conference-specific information to one, a group or all other conference members
mp-a: the session member sends application-specific information (data) to a group or all other session members

Function in XAPI:

Name: x_snddata(fd, data, flags, xerror)

Parameters:

fd: Identification of the service endpoint

data: The parameter *data* contains the data to be sent and related protocol-specific information. It is structured in the following parameters:

– *parameter*: additional information about the data
p-p: service primitive parameters which provide additional information about the user data
mp-c: additional information about the conference data (e.g. identification of the receiving conference member)
mp-a: additional information about the session data (e.g. identification of a dataflow; if more than one dataflow is provided in the session, the identification of the sending session member)

– *data*: the data to be sent

flags: indication of a local segmentation at the interface and the priority of the data
p-p: indication of normal or expedited data
mp-c: priority of the data
mp-a: priority of the data

xerror: Error codes

Return value: indicates successful completion or failure

9.2.14 X-SNDDIS/x_snddis

Description: This primitive is used to initiate an abortive release. Upon successful return, the endpoint changes to state "Idle".

The meaning is for:
p-p: the end of the connection
mp-c: the member leaves the conference
mp-a: the member leaves the session

Function in XAPI:

Name: x_snddis(fd, disc, xerror)

Parameters:

fd: Identification of the service endpoint

disc: The parameter *disc* contains disconnect-related information. It is structured in the following parameters:

– *user_data*: contains user data
p-p: the user may send disconnect-related user data to the peer user
mp-c: the conference member may send disconnect-related user data to the conference providing service and so perhaps to other conference members
mp-a: the session member may send disconnect-related user data to other session members

– *parameter*: contains service primitive parameters about the reason of disconnecting

– *origination*: no meaning

– *sequence*: local unique identification number which corresponds to a connect indication. It is only meaningful when X-SNDDIS is an answer to a connect indication

xerror: Error codes

Return value: indicates successful completion or failure

9.2.15 X-SNDINFO/x_sndinfo

Description: The X-SNDINFO primitive is used to send appropriate information to the provider. This service primitive does not change the state of the service endpoint. It is allowed in all states except the states "Uninit" and "Unbnd".

Function in XAPI:

Name: x_sndinfo(fd, level, info, par, flags, xerror)

Parameters:

fd: Identification of the service endpoint

level: Identification of the protocol module which shall get the info. This is one of the protocol module IDs which are available

info: Identification of the name of the protocol-specific service primitive
p-p: not used
mp-c: e.g. Information services for the conference
mp-a: e.g. Information services for the session

par: contains the parameters and user data of the service primitive

– *parameter*: contains the parameters of the service primitive
p-p: not used
mp-c: e.g. parameters of the conference-specific information service
mp-a: e.g. parameters of the session-specific information service

– *data*: contains the user data of the service primitive

flags: indication of a local segmentation at the interface

xerror: Error codes

Return value: indicates successful completion or failure

9.2.16 X-SNDSP/x_sndsp

Description: The X-SNDSP primitive is used to pass a protocol-specific service primitive to the service provider. The primitive does not change the state of the service endpoint. It is only allowed in the state "Connected".

Function in XAPI:

Name: x_sndsp(fd, level, spname, sp, flags, xerror)

Parameters:

fd: Identification of the service endpoint

level: Identification of protocol module which shall get the service primitive. This is one of the protocol module IDs which are available

spname: Identification of the name of the protocol-specific service primitive
p-p: e.g. in the OSI-session protocol S-TOKEN-PLEASE service
mp-c: Conference-specific services
mp-a: Session-specific services

sp: contains the parameters and user data of the service primitive

- *parameter*: contains the parameters of the service primitive
  p-p: contains the parameters of the service primitive
  mp-c: e.g. identification of the receiving conference member
  mp-a: e.g. identification of a dataflow; if more than one dataflow is provided in the session, the identification of the sending session member
- *data*: contains the user data of the service primitive

flags: indication of a local segmentation at the interface

xerror: Error codes

Return value: indicates successful completion or failure

9.3 Not communication-related functions

9.3.1 Functions for the initialization and de-initialization phase

A description of the functions for the initialization and de-initialization phase is given.

9.3.1.1 x_bind

Description: This function executes in two steps: first, link a transport system; second, associate a protocol address to the service endpoint identified by *fd*. On successful completion the service endpoint is activated and changes to state "Idle". The service provider may now begin to enqueue incoming connect indications, or the application may start an active connection establishment. If the service provider, which is accessed through the endpoint identified by *fd*, comprises an application system only, a transport system has to be linked below this application system in order to complete the protocol stack. This first step can be omitted if the service provider's protocol stack is already complete; this means that it comprises a built-in transport system.

Name: x_bind(fd, trans_name, req, ret, info, xerror)

Parameters:

fd: Identification of the service endpoint

trans_name: specifies the name of the transport system

req: Identification of the name of the protocol-specific service primitive

- *own_address*: contains the own protocol address
- *qlen*: the number of incoming calls that may be queued by the service provider for this endpoint

ret: Identification of the name of the protocol-specific service primitive

- *own_address*: contains the protocol address which has been bound to the service endpoint
- *qlen*: negotiation of the number of incoming calls that may be queued by the service provider for this endpoint

info: contains the characteristics of the service provider

xerror: Error codes

Return value: indicates successful completion or failure

9.3.1.2 x_close

Description: Informs the service provider that the user is finished with the service endpoint identified by *fd*, and frees any local XAPI library resources associated with the endpoint. In addition, *x_close()* closes the file descriptor associated with the service endpoint. The function *x_close()* should be called from the "Unbnd" state. However, this function does not check state information, so it may be called from any state to close a service endpoint. If this occurs, the local library resources associated with the endpoint will be freed automatically, and the operating system specific call for closing files, e.g. *close()*, will be issued for that file descriptor. This will break any connection that may be associated with the service endpoint. An *x_close()* issued on a service endpoint engaged in data transfer may cause data previously sent, or data not yet received, to be lost. It is not recommended to simply close a file descriptor identifying an XAPI service endpoint with the operating system-specific function call. Instead, the *x_close()* function should always be used to close a service endpoint. This enables the XAPI library to free the local resources allocated for this endpoint.

Name: x_close(fd, xerror)

Parameters:

fd: Identification of the service endpoint

xerror: Error codes

Return value: indicates successful completion or failure

9.3.1.3 x_open

Description: This function must be called as the first step to create a local service endpoint. A service provider is to be accessed through the newly created endpoint.

Name: x_open (prov_name, mode, info, xerror)

Parameters:

prov_name: Name of the service provider

mode: specifies the initial execution mode for the service endpoint

info: contains the characteristics of the service provider to be accessed

xerror: Error codes

Return value: generates a valid file descriptor or indicates failure

9.3.1.4 x_rcvend

Description: Indicates that the service provider is ready to establish a new connection on this endpoint. If charging units are supported, they will be returned.

Name: x_open(fd, end, xerror)

Parameters:

fd: Identification of the service endpoint

end: contains parameters

xerror: Error codes

Return value: indicates successful completion or failure

9.3.1.5 x_unbind

Description: Deactivates the service endpoint identified by *fd* which was previously activated by an *x_bind()* call. On completion of *x_unbind()*, no further data or events destined for this service endpoint will be accepted by the service provider. If the service provider accessed through the endpoint has a dynamically linked transport system below its application system, the transport system is unlinked from the application system. Built-in transport systems are not affected by *x_unbind()*. In both cases, the protocol address that has been associated with the endpoint on *x_bind()* is removed from the service endpoint. If this was a passive endpoint (one with a value of *qlen* greater than zero) the protocol address is now free again and may be used to activate another passive endpoint. The deactivated service endpoint can be reactivated by a subsequent call to *x_bind()*.

Name: x_unbind(fd, xerror)

Parameters:

fd: Identification of the service endpoint

xerror: Error codes

Return value: indicates successful completion or failure

9.3.2 Utility functions

9.3.2.1 x_b2c

Description: Selects a value of type character string out of a buffer.

Use this function to read options, service primitive parameters, and address buffers.

Name: x_b2c(nb, index, len, name, status, xerror)

Parameters:

nb: identifies a buffer

index: next entry in the chosen buffer

len: length of the entry value

name: name of the entry

status: indicates success or failure of an option negotiation; unused for address and parameter buffers

xerror: Error codes

Return value: a value of type character string or indication of failure

9.3.2.2 x_b2l

Description: Selects a value of type long out of a buffer.

Use this function to read option, service primitive parameter, and address buffers.

Name: x_b2l(nb, index, len, name, status, xerror)

Parameters:

nb: identifies a buffer

index: next entry in the chosen buffer

len: length of the entry value

name: name of the entry

status: indicates success or failure of an option negotiation; unused for address and parameter buffers

xerror: Error codes

Return value: a value of type long or indication of failure

9.3.2.3 x_c2b

Description: Writes a value of type character string into a buffer.

Use this function to write option, service primitive parameter, and address buffers.

Name: x_c2b(nb, index, len, name, value, xerror)

Parameters:

nb: identifies a buffer

index: next entry in the chosen buffer

len: length of the entry value

name: name of the entry

value: value of the entry

xerror: Error codes

Return value: indicates successful completion or failure

9.3.2.4 x_chexmod

Description: Changes the execution mode for the service endpoint identified by *fd*. The *mode* argument specifies the new execution mode:

SYNCHRON to switch to synchronous execution mode
ASYNCHRON to switch to asynchronous execution mode

The initial execution mode for a service endpoint is assigned when the endpoint is created with the *x_open()* function. The user may change the execution mode at any time.

Depending on the operating system, maybe not all execution modes are supported for all service providers.

Name: x_chexmod(fd, mode, xerror)

Parameters:

fd: identifies the service endpoint

mode: specifies the new execution mode

xerror: Error codes

Return value: indicates successful completion or failure

9.3.2.5 x_error

Description: Writes a message to the standard error output.

The whole message is written as follows: First, the string pointed to by *errmsg* followed by a colon and a space is written. Then, the error message string corresponding to the error indicated by *xerror* is written, followed by a newline character.

The messages are the same as returned by *x_strerror()*. If an error code is unknown, *x_error()* writes the string

```
"<error>: error unknown"
```

where <error> is the (decimal) error number supplied as value of *xerror*.

Name: x_error(errmsg, xerror)

Parameters:

errmsg: user-supplied error message

xerror: error code returned by an XAPI function call

Return value: indicates completion

9.3.2.6 x_getinfo

Description: Returns by *info* the current characteristics of the underlying service provider and/or connection associated with the service endpoint identified by *fd*. This is the same information as returned by *x_open()* and *x_bind()*, although not necessarily precisely the same values as some of them might have been changed in a peer-to-peer negotiation during connection establishment. The function *x_getinfo()* enables a service user to access this information during any phase of communication.

Name: x_getinfo(fd, info, xerror)

Parameters:

fd: identifies the service endpoint

info: shows the current characteristics of the underlying service provider

xerror: Error codes

Return value: indicates successful completion or failure

9.3.2.7 x_look

Description: Returns the current event on the service endpoint identified by *fd*. The user should then call *x_look()* to see which event occurred. The function *x_look()* acts independent of the current execution mode.

The function may also be used to poll a service endpoint periodically for asynchronous events.

Name: x_look(fd, xerror)

Parameters:

fd: identifies the service endpoint

xerror: Error codes

Return value: indicates the current event or failure

9.3.2.8 x_l2b

Description: Writes a value of type long into a buffer.

Use this function to write options, service primitive parameters, and address buffers.

Name: x_l2b(nb, index, len, name, value, xerror)

Parameters:

nb: identifies a buffer

index: next entry in the chosen buffer

len: length of the entry value

name: name of the entry

value: value of the entry

xerror: Error codes

Return value: indicates successful completion or failure

9.3.2.9 x_optmgmt

Description: Enables a user to retrieve, verify or negotiate protocol options with the service provider or the XAPI library. It can be used to negotiate non-connection-related options or preset connection-related options for future connections. The function can process and return all options related to the specified service endpoint. The function may be called in all states of the endpoint except X_UNINIT.

A call of *x_optmgmt()* is an atomic operation, i.e. it always acts independent of the current execution mode and the caller is blocked until the return of *x_optmgmt()*.

The protocol module which is the object of the *x_optmgmt()* call is denoted by the *level* parameter. To handle XAPI-level options, XAPI_LEVEL has to be specified as *level*.

The parameters *req* and *ret* both reference an option management structure which contains the members *options* and *flags.*

On input, the *options* buffer of *req* contains the requested options and the *flags* specify the action that shall be taken with these options. On return, the result of the option management call is available in *ret*. What is returned in the *options* buffer of *ret* and in *flags* depends on the action which has been specified in *req.* The *flags* field of *req* must specify one of the following actions: NEGOTIATE, CHECK, DEFAULT, CURRENT. The *status* field in the header of each returned option is set to indicate the result of the action. The value is SUCCESS, PARTSUCCESS, READONLY, NOTSUPPORT, or FAILURE. The overall result of the action is returned in *ret->flags*. It contains the worst single result, whereby the rating is done according to the order (from worst to best) FAILURE, NOTSUPPORT, READONLY, PARTSUCCESS, SUCCESS.

Name: x_optmgmt(fd, level, req, ret, xerror)

Parameters:

fd: identifies the service endpoint

level: a protocol module identifier that selects the target protocol module

req: contains the option specifications which shall be passed to the protocol module and the action to take with the options:
- *options*
- *flags*

ret: contains the result of the option management call:
- *options*
- *flags*

xerror: Error codes

Return value: indicates successful completion or failure

9.3.2.10 x_rcverror

Description: Is used to retrieve an error indication that has been generated by the service provider. It is the consuming function for the XAPI event ERROR and must be issued, if this event occurred at a service endpoint or a previous function failed with ER_ERROR.

The function behaves independent of the execution mode.

Name: x_rcverror(fd, err, xerror)

Parameters:

fd: identifies the service endpoint

err: gives some basic information about the location where the error has been detected, and the reason for failure:

- *level*: identifies the protocol module which generated the error indication
- *service*: specifies the service that caused the error indication
- *cause*: cause code
- *diagnostic*: conditional, some additional information about the error

xerror: Error codes

Return value: indicates successful completion or failure

9.3.2.11 x_strerror

Description: Maps the XAPI error number *errnum* to an error message string and returns the string. If an error code is unknown, *x_strerror()* returns the string

```
"<error>: error unknown"
```

where <error> is the (decimal) error number supplied as input.

Name: x_strerror(errnum)

Parameters:

errnum: specifies an XAPI-level error code

Return value: error message string

9.3.2.12 x_sync

Description: Synchronizes the data structures managed by the XAPI library with the information from the underlying service provider. In doing so it can convert an uninitialized file descriptor (not obtained from an *x_open()* call) to an initialized service endpoint by updating and allocating the necessary library data structures. Usually they are allocated and initialized by *x_open()*.

The function *x_sync()* also allows two cooperating processes which both access the same service endpoint, to synchronize their interaction with the service provider. It is important to remember that the service provider treats all users of a service endpoint as a single user. If multiple processes are using the same endpoint, they should coordinate their activities so as not to violate the state of the service endpoint. The function *x_sync()* returns the current state of the service endpoint to the caller, thereby enabling the user to verify the state before taking further action. This coordination is only valid among cooperating processes; it is possible that a process or an incoming event could change the endpoint's state **after** an *x_sync()* is issued.

Name: x_sync(fd, xerror)

Parameters:

fd: identifies the service endpoint

xerror: Error codes

Return value: indicates the current state or failure

ANNEX A

Interface definition language description

This annex provides additional information concerning the XAPI functions (see clause 9). It contains an IDL description of the XAPI.

IDL defines the types of objects according to the operations that may be performed on them and the parameters to those operations.

```
module x
{

/* ------------------------------------------------------------------------------------------------*/
/* Asynchronous events returned by x_look()                                                        */
/* ------------------------------------------------------------------------------------------------*/

const short NOEVENT       = 0x0000;   /* no event present at the endpoint          */
const short ERROR         = 0x0001;   /* error condition detected by the           */
                                      /* service provider                          */
const short CONIND        = 0x0002;   /* connection indication received            */
const short CONCONF       = 0x0004;   /* connect confirmation received             */
const short DATA          = 0x0008;   /* normal data received                      */
const short EXDATA        = 0x0010;   /* expedited data received                   */
const short SP            = 0x0020;   /* service primitive received                */
const short DISCONNECT    = 0x0040;   /* disconnect received                       */
const short RELIND= 0x0080;           /* orderly release indication received       */
const short RELCONF       = 0x0100;   /* orderly release confirmation received     */
const short GODATA        = 0x0200;   /* normal data may be sent again             */
const short GOEXDATA      = 0x0400;   /* expedited data may be sent again          */
const short END           = 0x0800;   /* end indication received                   */
const short EVENTS        = 0x0FFF;   /* event mask                                */

/* ------------------------------------------------------------------------------------------------*/
/* Bit-Flag Definitions                                                                            */
/* The bit-masks are used in the flags field of call_struct, conind_struct,                        */
/* optmgmt_struct, release_struct and in the flags argument of x_snddata(),                        */
/* x_rcvdata(), x_snddis(), x_rcvdis, x_sndsp(), x_rcvsp()                                         */
/* ------------------------------------------------------------------------------------------------*/

const unsigned long NOFLAG     = 0x0000;   /* no flag is set                       */
const unsigned long MORE       = 0x0001;   /* more data                            */
                                           /* used in struct x_call, x_conind,     */
                                           /* x_snddata(), x_rcvdata()             */
```

```
const unsigned long EXPEDITED  = 0x0002;   /* expedited data                  */
                                           /* used in x_snddata(), x_rcvdata()*/
const unsigned long NEGATIVE   = 0x0004;   /* negative confirmation/response  */
                                           /* used in struct x_call           */

/* ---------------------------------------------------------------------------------------------*/
/* Option Management Bit-Flag Definitions                                                       */
/* The bit-masks are used in the flags field of optmgmt_struct.                                 */
/* Input bits are marked with (i) and output bits with (o).                                     */
/* ---------------------------------------------------------------------------------------------*/

const unsigned long NEGOTIATE      = 0x00000100;  /* (i) negotiate (set) options */
const unsigned long CHECK          = 0x00000200;  /* (i) check options           */
const unsigned long DEFAULT        = 0x00000400;  /* (i) get default values of   */
                                                  /* options                     */
const unsigned long CURRENT        = 0x00000800;  /* (i) get current values of   */
                                                  /* options                     */
const unsigned long SUCCESS        = 0x00001000;  /* (o) successful option       */
                                                  /* negotiation                 */
const unsigned long PARTSUCCESS    = 0x00002000;  /* (o) partially successful    */
                                                  /* option negotiation          */
const unsigned long READONLY       = 0x00004000;  /* (o) option is readonly      */
const unsigned long NOTSUPPORT     = 0x00008000;  /* (o) option is not supported */
const unsigned long FAILURE        = 0x00010000;  /* (o) failure in option       */
                                                  /* negotiation                 */

/* ---------------------------------------------------------------------------------------------*/
/* Execution modes                                                                              */
/* Used in x_open() and x_chexmod() calls                                                       */
/* ---------------------------------------------------------------------------------------------*/

const short SYNCHRON     = 1;          /* synchronous execution mode              */
const short ASYNCHRON    = 2;          /* asynchronous execution mode             */


/* ---------------------------------------------------------------------------------------------*/
/* Protocol Module Identifiers                                                                  */
/* ---------------------------------------------------------------------------------------------*/

const unsigned long NO_MODULE    = 0x7FFFFFFF;  /* no protocol module present */
const unsigned long XAPI_LEVEL   = 20;          /* denotes the XAPI library;  */
                                                /* to access XAPI-level       */
                                                /* options with x_optmgmt()   */
const unsigned long TOP_LEVEL    = 2;           /* Topmost protocol module    */
                                                /* the special value is defined */
                                                /* in the provider info       */
```

```
/* ------------------------------------------------------------------------------------------*/
/* General purpose service parameter values                                                  */
/* ------------------------------------------------------------------------------------------*/

const unsigned long PV_TRUE         = 1;        /* parameter value boolean true          */
const unsigned long PV_FALSE        = 0;        /* parameter value boolean false         */


/* ------------------------------------------------------------------------------------------*/
/* Names of address components                                                               */
/* ------------------------------------------------------------------------------------------*/

const unsigned long A_OUTBAND_ADR   = 1;        /* outband address                       */
const unsigned long A_INBAND_ADR    = 3;        /* inband address                        */
const unsigned long A_INB_SUBADR    = 4;        /* inband subaddress                     */
const unsigned long A_SERVICE       = 5;        /* service indicator                     */
const unsigned long A_P_SELECTOR    = 1000;     /* presentation selector                 */
const unsigned long A_S_SELECTOR    = 1001;     /* session selector                      */
const unsigned long A_T_SELECTOR    = 1002;     /* transport selector                    */


/* ------------------------------------------------------------------------------------------*/
/* Constants defining the maximum length for address components                              */
/* ------------------------------------------------------------------------------------------*/

const unsigned long C_MAX_IBADR     = 32;   /* max. length of inband address         */
const unsigned long C_MAX_IBSADR    = 32;   /* max. length of inband subaddress      */
const unsigned long C_MAX_OBADR     = 32;   /* max. length of outband address        */
const unsigned long C_MAX_SERVICE   = 6;    /* max. length of service indicator      */
const unsigned long C_MAX_PSEL      = 32;   /* max. length of presentation selector*/
const unsigned long C_MAX_SSEL      = 16;   /* max. length of session selector       */
const unsigned long C_MAX_TSEL      = 32;   /* max. length of transport selector     */


/* ------------------------------------------------------------------------------------------*/
/* Constants defining the maximum length for session components                              */
/* ------------------------------------------------------------------------------------------*/

const unsigned long C_MAX_REF       = 64;   /* max. length of service user reference  */
                                            /* and common reference                   */
const unsigned long C_MAX_ARI       = 4;    /* max. length of additional information  */

/* ------------------------------------------------------------------------------------------*/
/* XAPI-level Options                                                                        */
/* ------------------------------------------------------------------------------------------*/

const unsigned long O_TRACE         = 1;        /* XAPI trace option                     */
```

```
/* ------------------------------------------------------------------------------------------*/
/* Defined values for option O_TRACE                                                         */
/* ------------------------------------------------------------------------------------------*/

const unsigned long OV_NOTRACE     = 0x00;  /* switches all traces off (default)    */
const unsigned long OV_BUFFTRACE   = 0x01;  /* enables buffer trace for             */
                                            /* received and sent PDUs;              */
                                            /* all trace messages are disabled      */
const unsigned long OV_ERR         = 0x10;  /* enables error trace messages         */
                                            /* and disables buffer traces           */
const unsigned long OV_ERR_BUF     = 0x11;  /* enables error trace messages         */
                                            /* plus buffer traces                   */
const unsigned long OV_WRN         = 0x30;  /* enables error and warning            */
                                            /* trace messages;                      */
                                            /* buffer traces are disabled           */
const unsigned long OV_WRN_BUF     = 0x31;  /* enables error and warning trace      */
                                            /* messages plus buffer traces          */
const unsigned long OV_INF         = 0x70;  /* enables error, warning, and          */
                                            /* info trace messages;                 */
                                            /* buffer traces are disabled           */
const unsigned long OV_INF_BUF     = 0x71;  /* enables error, warning, and          */
                                            /* info trace messages plus             */
                                            /* buffer traces                        */

/* ------------------------------------------------------------------------------------------*/
/* General Definitions                                                                       */
/* ------------------------------------------------------------------------------------------*/

const unsigned long INFINITE       = ~0;    /*0xFFFFFFFF;                           */
                                            /* BBB: value needs to be checked in    */
                                            /* language mappings infinite size      */
                                            /* suitable for unsigned long           */
const unsigned long UNSPECIFIED    = ~1;    /* 0xFFFFFFFE;                          */
                                            /* BBB: value needs to be checked in
                                            /* language mappings unspecified size   */
                                            /* suitable for unsigned long           */

/* ------------------------------------------------------------------------------------------*/
/* Service Provider Capabilities                                                             */
/* ------------------------------------------------------------------------------------------*/

const unsigned long SPC_COS          = 0x00000001; /* connection-mode service   */
const unsigned long SPC_CLS          = 0x00000002; /* connectionless service    */
const unsigned long SPC_ORD_REL      = 0x00000004; /* orderly release           */
const unsigned long SPC_0DATA        = 0x00000008; /* zero-length data units    */
const unsigned long SPC_DATA         = 0x00000010; /* (normal) data             */
const unsigned long SPC_EXPDATA      = 0x00000020; /* expedited data            */
const unsigned long SPC_NEGCONRSP    = 0x00000040; /* negative connect response */
const unsigned long SPC_NEGCONCNF    = 0x00000080; /* negative connect confirm. */
const unsigned long SPC_MORECONRQ    = 0x00000100; /* MORE in x_conreq()        */
```

```
const unsigned long SPC_MORECONCF  = 0x00000200;  /* MORE in x_conconf()      */
const unsigned long SPC_MORECONIN  = 0x00000400;  /* MORE in x_conind()       */
const unsigned long SPC_MORECONRP  = 0x00000800;  /* MORE in x_conrsp()       */
const unsigned long SPC_MORERELRQ  = 0x00001000;  /* MORE in x_relreq()       */
const unsigned long SPC_MORERELCF  = 0x00002000;  /* MORE in x_relconf()      */
const unsigned long SPC_MORERELIN  = 0x00004000;  /* MORE in x_relind()       */
const unsigned long SPC_MORERELRP  = 0x00008000;  /* MORE in x_relrsp()       */
const unsigned long SPC_SP         = 0x00010000;  /* send service primitive   */
const unsigned long SPC_MORESNDSP  = 0x00020000;  /* MORE in x_sndsp()        */
const unsigned long SPC_MORERCVSP  = 0x00030000;  /* MORE in x_rcvsp()        */
const unsigned long SPC_MORESNDDIS = 0x00040000;  /* MORE in x_snddis()       */
const unsigned long SPC_MORERCVDIS = 0x00050000;  /* MORE in x_rcvdis()       */
const unsigned long SPC_MASK       = 0x000FFFFF;  /* all defined capabilities */

/* ------------------------------------------------------------------------------------------*/
/* Disconnect Origination                                                                    */
/* ------------------------------------------------------------------------------------------*/

const unsigned long PROVIDER_ABORT  = 0x00000001;   /* disconnect generated   */
                                                    /* by the service provider */
const unsigned long ABORT           = 0x00000002;   /* disconnect generated   */
                                                    /* by the user            */

/* ------------------------------------------------------------------------------------------*/
/* Services indicated in error_struct                                                        */
/* ------------------------------------------------------------------------------------------*/

const unsigned long CONNECT_REQ    = 0x01;    /* connect request; x_conreq()     */
const unsigned long CONNECT_RES    = 0x02;    /* connect response; x_conrsp()    */
const unsigned long DATA_REQ       = 0x03;    /* data request; x_snddata()       */
const unsigned long EXPDATA_REQ    = 0x04;    /* expedited data request;         */
                                              /* x_snddata()                     */
const unsigned long DISC_REQ       = 0x05;    /* disconnect request; x_snddis()  */
const unsigned long RELEASE_REQ    = 0x06;    /* release request; x_relreq()     */
const unsigned long RELEASE_RES    = 0x07;    /* release response; x_relrsp()    */
const unsigned long SND_SP         = 0x08;    /* send service primitive;         */
                                              /* x_sndsp()                       */

/* ------------------------------------------------------------------------------------------*/
/* Cause Codes indicated in error_struct                                                     */
/* ------------------------------------------------------------------------------------------*/

const unsigned long CC_BADEVENT    = 0x01;    /* requested service unknown       */
                                              /* to the service provider         */
const unsigned long CC_UNEXPECT    = 0x02;    /* requested service not allowed   */
                                              /* in current state of             */
                                              /* service provider                */
const unsigned long CC_NOTSUPPORT  = 0x03;    /* requested service not           */
                                              /* supported by the service        */
                                              /* provider                        */
```

```
const unsigned long CC_BADVALUE     = 0x04;    /* illegal value specified for a    */
                                               /* a service parameter              */
const unsigned long CC_MANDMISS     = 0x05;    /* mandatory service parameter      */
                                               /* missing                          */
const unsigned long CC_OTHER        = 0x06;    /* any other error                  */
const unsigned long CC_NOSPACE      = 0x07;    /* no space                         */
const unsigned long CC_SPNAME       = 0x08;    /* SP_name incorrect                */
const unsigned long CC_ADDCOMP      = 0x09;    /* additional component             */
                                               /* incorrect                        */
const unsigned long CC_BADLENGTH    = 0x0A;    /* string or value too long         */
const unsigned long CC_SEQ          = 0x0B;    /* SEQ incorrect                    */

/* ------------------------------------------------------------------------------------------*/
/* Service parameters for x_rcvend()                                                         */
/* ------------------------------------------------------------------------------------------*/

const unsigned long P_CONN_TIME     = 0x01;    /* Connection time                  */
const unsigned long P_DISC_TIME     = 0x02;    /* Disconnection time               */
const unsigned long P_CHARGE        = 0x03;    /* Charging units                   */
const unsigned long P_DISC_REASON   = 0x04;    /* Disconnect reason                */

/* ------------------------------------------------------------------------------------------*/
/* Possible states of a service endpoint                                                     */
/* ------------------------------------------------------------------------------------------*/

const short UNINIT          = 0;        /* uninitialized                    */
const short UNBND           = 1;        /* unbound                          */
const short IDLE            = 2;        /* idle                             */
const short OUTCON          = 3;        /* outgoing connection pending      */
const short INCON           = 4;        /* incoming connection pending      */
const short CONNECT         = 5;        /* connected / data transfer        */
const short OUTREL          = 6;        /* outgoing release pending         */
const short INREL           = 7;        /* incoming release pending         */
const short WAITEND         = 8;        /* waiting for end indication       */

typedef sequence < octet > octet_sequence;

/* ------------------------------------------------------------------------------------------*/
/* provinfo_struct                                                                           */
/* Service provider information structure; returned by x_open(),                             */
/* x_bind() and x_getinfo().                                                                 */
/* ------------------------------------------------------------------------------------------*/

typedef unsigned long ten_unsigned_longs[10];

struct provinfo_struct
{
  /* Protocol module identifiers of the modules in the service
   * provider's protocol stack. If no protocol module is present on
   * a certain layer in the protocol stack, the corresponding ln_pmid
```

```
 * field(s) is(are) set to NO_MODULE
 */
ten_unsigned_longs    l7_pmid;          /* layer 7 protocol module IDs          */
unsigned long    l6_pmid;               /* layer 6 protocol module ID           */
unsigned long    l5_pmid;               /* layer 5 protocol module ID           */
unsigned long    l4_pmid;               /* layer 4 protocol module ID           */
unsigned long    l3_pmid;               /* layer 3 protocol module ID           */
unsigned long    l2_pmid;               /* layer 2 protocol module ID           */
unsigned long    l1_pmid;               /* layer 1 protocol module ID           */
unsigned long    Monitor_pmid;          /* Monitor protocol module ID           */

unsigned long    max_qlen;              /* max. supported value for qlen        */

/* Buffer sizes recommended for the service provider. Output buffers
 * with these sizes will be sufficent in all cases.
 */
unsigned long    addr_buff_size;        /* address buffer size                  */
unsigned long    optn_buff_size;        /* option buffer size                   */
unsigned long    parm_buff_size;        /* SP parameter buffer size             */
unsigned long    data_frgmt_size;       /* maximum data fragment size           */
unsigned long    end_buff_size;         /* end indication buffer size           */

/* Maximum data unit sizes supported by the service provider.
 */
unsigned long    max_sdu_size;          /* max. size of a (normal)              */
                                        /* service data unit                    */
unsigned long    max_esdu_size;         /* max. size of an expedited            */
                                        /* service data unit                    */
unsigned long    max_conn_user_data;    /* max. user data size for              */
                                        /* connection primitives                */
unsigned long    max_disc_user_data;    /* max. user data size for              */
                                        /* disconnect primitives                */
unsigned long    max_rels_user_data;    /* max. user data size for              */
                                        /* release primitives                   */

/* The capabilities of the service provider are represented as
 * bit-field. A capability is supported by the application system,
 * if the corresponding bit is set in prov_capabilities; it is not
 * supported, if the bit is cleared.
 */
unsigned long    prov_capabilities;     /* service provider capabilities        */
};
```

```
/* ------------------------------------------------------------------------------------------*/
/* bind_struct                                                                               */
/* argument structure for x_bind()                                                           */
/* ------------------------------------------------------------------------------------------*/

struct bind_struct
{
  octet_sequenceown_address;          /* own NSAP address                                    */
  unsigned long  qlen;                /* number of incoming calls                            */
                                      /* to be queued for the endpoint                       */
};

/* ------------------------------------------------------------------------------------------*/
/* call_struct                                                                               */
/* argument structure for connection establishment                                           */
/* ------------------------------------------------------------------------------------------*/

struct call_struct
{
  octet_sequenceaddress;              /* address buffer                                      */
  octet_sequenceparameter;            /* SP parameter buffer                                 */
  octet_sequenceuser_data;            /* user data buffer                                    */
  unsigned long  sequence_nr;         /* sequence number                                     */
  unsigned long  flags;               /* flags; MORE / NEGATIVE                              */
};

/* ------------------------------------------------------------------------------------------*/
/* conind_struct                                                                             */
/* argument structure for connect indication                                                 */
/* ------------------------------------------------------------------------------------------*/

struct conind_struct
{
  octet_sequencecalled_addr;          /* address buffer called address                       */
  octet_sequencecalling_addr;         /* address buffer calling address                      */
  octet_sequenceparameter;            /* SP parameter buffer                                 */
  octet_sequenceuser_data;            /* user data buffer                                    */
  unsigned long sequence_nr;          /* sequence number                                     */
  unsigned long flags;                /* flags; MORE                                         */
};
```

```
/* ------------------------------------------------------------------------------------------*/
/* data_struct                                                                               */
/* argument structure for x_snddata() and x_rcvdata()                                        */
/* ------------------------------------------------------------------------------------------*/

struct data_struct
{
  octet_sequenceparameter;              /* SP parameter buffer                               */
  octet_sequencedata;                   /* data buffer                                       */
};

/* ------------------------------------------------------------------------------------------*/
/* discon_struct                                                                             */
/* argument structure for x_snddis() and x_rcvdis()                                          */
/* ------------------------------------------------------------------------------------------*/

struct discon_struct
{
  octet_sequenceuser_data;              /* user data buffer                                  */
  octet_sequenceparameter;              /* SP parameter buffer                               */
  unsigned long  origination;           /* origination of disconnect;                        */
                                        /* used on output only                               */
  unsigned long  sequence_nr;           /* sequence number; used only                        */
                                        /* when an incoming call is rejected                 */
};

/* ------------------------------------------------------------------------------------------*/
/* error_struct                                                                              */
/* argument structure for x_rcverrors()                                                      */
/* ------------------------------------------------------------------------------------------*/

struct error_struct
{
  unsigned long  level;             /* protocol module ID of the originator module         */
  unsigned long  service;           /* requested service that caused the error             */
  unsigned long  cause;             /* cause code specifying the reason of failure         */
  unsigned long  diagnostic;        /* additional protocol-specific diagnostic information */
};

/* ------------------------------------------------------------------------------------------*/
/* end_struct                                                                                */
/* argument structure for x_rcvend()                                                         */
/* ------------------------------------------------------------------------------------------*/

struct end_struct
{
 octet_sequence parameter;          /* SP parameter buffer                                 */
};
```

```
/* ------------------------------------------------------------------------------------------*/
/* optmgmt_struct                                                                            */
/* argument structure for x_optmgmt()                                                        */
/* ------------------------------------------------------------------------------------------*/

struct optmgmt_struct
{
  octet_sequenceoptions;          /* option definitions                                      */
  unsigned long  flags;           /* what to do with options                                 */
};

/* ------------------------------------------------------------------------------------------*/
/* release_struct                                                                            */
/* argument structure for orderly release                                                    */
/* ------------------------------------------------------------------------------------------*/

struct release_struct
{
  octet_sequenceparameter;        /* SP parameter buffer                                     */
  octet_sequenceuser_data;        /* user data buffer                                        */
  unsigned long  flags;           /* flags; MORE                                             */
};

/* ------------------------------------------------------------------------------------------*/
/* sp_struct                                                                                 */
/* argument structure for x_sndsp() and x_rcvsp()                                            */
/* ------------------------------------------------------------------------------------------*/

struct sp_struct
{
  octet_sequenceparameter;        /* SP parameter buffer                                     */
  octet_sequencedata;             /* data buffer                                             */
};

exception XERROR
{
        short error_code;
};

interface ep
{

        short bind(             in      short fd,
                                in      string trans_name,
                                in      bind_struct req,
                                out     bind_struct ret,
                                out     provinfo_struct info)
                        raises ( XERROR );
```

```
short chexmod(          in      short fd,
                        in      short mode)
            raises ( XERROR );

short close(            in      short fd
                        )
            raises ( XERROR );

short conconf(          in      short fd,
                        out     call_struct call)
            raises ( XERROR );

short conind(           in      short fd,
                        out     conind_struct call)
            raises ( XERROR );

short conreq(           in      short fd,
                        in      call_struct sndcall)
            raises ( XERROR );

short conrsp(           in      short fd,
                        in      short resfd,
                        in      call_struct call)
            raises ( XERROR );

short getinfo(          in      short fd,
                        out     provinfo_struct info)
            raises ( XERROR );

short look(             in      short fd
                        )
            raises ( XERROR );

    short optmgmt(      in      short fd,
                        in      unsigned long level,
                        in      optmgmt_struct req,
                        out     optmgmt_struct ret)
            raises ( XERROR );

short rcvdata(          in      short fd,
                        out     data_struct data,
                        out     unsigned long flags)
            raises ( XERROR );

short rcvdis(           in      short fd,
                        out     discon_struct disc,
                        out     unsigned long flags)
            raises ( XERROR );
```

```
short rcvend(           in      short fd,
                        out     end_struct end)
            raises ( XERROR );

short rcverror(         in      short fd,
                        out     error_struct err)
            raises ( XERROR );

short rcvinfo(          in      short fd,
                        out     unsigned long level,
                        out     unsigned long info,
                        out     sp_struct par,
                        out     unsigned long flags)
            raises ( XERROR );

short rcvsp(            in      short fd,
                        out     unsigned long level,
                        out     unsigned long spname,
                        out     sp_struct sp,
                        out     unsigned long flags)
            raises ( XERROR );

short relconf(          in      short fd,
                        out     release_struct rel)
            raises ( XERROR );

short relind(           in      short fd,
                        out     release_struct rel)
            raises ( XERROR );

short relreq(           in      short fd,
                        in      release_struct rel)
            raises ( XERROR );

short relrsp(           in      short fd,
                        in      release_struct rel)
            raises ( XERROR );

short snddata(          in      short fd,
                        in      data_struct data,
                        in      unsigned long flags)
            raises ( XERROR );

short snddis(           in      short fd,
                        in      discon_struct disc,
                        in      unsigned long flags)
            raises ( XERROR );
```

```
        short sndinfo(          in      short fd,
                                in      unsigned long level,
                                in      unsigned long info,
                                in      sp_struct par,
                                in      unsigned long flags)
                        raises ( XERROR );

        short sndsp(            in      short fd,
                                in      unsigned long level,
                                in      unsigned long spname,
                                in      sp_struct sp,
                                in      unsigned long flags)
                        raises ( XERROR );

        short sync(             in      short fd
                                )
                        raises ( XERROR );

        short unbind(           in      short fd
                                )
                        raises ( XERROR );

};

interface api
{
        string b2c (            in      octet_sequence nb,
                                inout   unsigned long index,
                                out     unsigned long len,
                                out     unsigned long name,
                                out     unsigned long status)
                        raises ( XERROR );

        long b2l(               in      octet_sequence nb,
                                inout   unsigned long index,
                                out     unsigned long len,
                                out     unsigned long name,
                                out     unsigned long status)
                        raises ( XERROR );

        short c2b(              in      octet_sequence nb,
                                inout   unsigned long index,
                                in      unsigned long len,
                                in      unsigned long name,
                                in      string value)
                        raises ( XERROR );

        short error(            in      string errmsg,
                                in      short xerror);
```

```
        short l2b(          in      octet_sequence nb,
                            inout   unsigned long index,
                            in      unsigned long len,
                            in      unsigned long name,
                            in      long value)
                raises ( XERROR );

        short open(         in      string prov_name,
                            in      short mode,
                            out     provinfo_struct info)
                raises ( XERROR );

        string strerror(    in      short errnum);
};

};
```

ANNEX B

Error codes

An application program using the XAPI has to deal with errors on three levels. The first is the **XAPI error level**. Each function returns −1 to indicate failure and 0 (zero) to indicate success. In case of failure, the integer output parameter *xerror* is set to one of the error codes defined below. The error code provides some more detailed information about the reason of failure. In case of success, *xerror* is set to NOERROR.

The second level is the **operating system error level**. When an operating system service routine called by the application fails, the error is reported in an operating-system-dependent way. The XAPI functions use operating system calls, too, but these calls are hidden from the application. When an XAPI function fails due to an error in an operating system call, the error code reported by the system is mapped to the corresponding XAPI-level error code, which is returned to the user as described above. In some rare cases, if there is no XAPI error code corresponding to the system error, the XAPI function will return failure and indicate the special XAPI-level error code ER_SYSERR. The application may then access and check the operating system error code. For the meaning of the system error codes, refer to the operating system documentation. Note that the system error codes may (will) be different for the various base operating systems. This affects the portability of an application that checks system error codes. The XAPI-level error codes, conversely, are identical for all base operating systems.

Two utility functions can be used to map XAPI-level error codes to short text error messages. Both utilities take the error code as argument; *x_error* prints the corresponding error message to the standard error output *stderr*, while *x_strerror* returns the corresponding error message character string. For system level errors there may be similar functions available among the operating system services.

As result of a severe error, a major protocol error, for example, or some local resource problems, a service endpoint and the corresponding file descriptor may be unusable for all XAPI and system calls except *x_close*. To continue in this case, all users of the *fd* must close it, i.e. call *x_close*. Then a new service endpoint may be created and initialized to continue with communication.

The third level is the **service provider error level**. A service provider may generate an asynchronous event ERROR whenever the application program shall be informed about an error condition. This could be

– the reception of an incorrect or incomplete service primitive from the local application; or

– a request that is not allowed in the current protocol-specific state of the service provider.

The reception of an erroneous or out-of-order PDU from the peer entity is not indicated to the local application but handled by the service provider in a protocol-specific way.

The ERROR event is indicated to the application as are all other asynchronous events and has to be received from the service endpoint with the consuming function *x_rcverror*. On return, this function yields a service-specific error code and possibly further information about the error as output. The concrete error conditions and the corresponding error codes are defined in the service descriptions.

Error codes at the service provider error level are listed in the appropriate providers (see I.1 to I.7). All other error codes that may be returned by the XAPI functions are listed below.

For each error code a short explanation of the possible reason is given. The error codes are sorted in ascending numerical order.

NOERROR	8001	No error occurred; all in perfect order. This "error code" is returned in *xerror* if the function successfully completed.
ER_BADFD	8002	The file descriptor *fd* passed as argument does not identify a service endpoint. Maybe *fd* was not obtained as result of an *x_open()* call.
ER_OUTSTATE	8003	The function was issued in the wrong state of the service endpoint.
ER_LOOK	8004	An asynchronous event has occurred on this service endpoint and requires immediate attention. Call *x_look()*, please.
ER_NODATA	8005	The XAPI consuming function was called in asynchronous mode, but the requested event and no other asynchronous event are currently available at the service endpoint. The function *x_rcvdata()*, for instance, looks for a DATA rsp. EXDATA event; if none is present at the time of calling, this error code is returned in asynchronous mode.
ER_BADFLAG	8010	An invalid flag is specified in the *flags* field. This error is returned if either an unknown flag is specified or the flag is not allowed for this XAPI function. Note that for some flags it depends on the service provider capabilities whether a certain flag may be used with the function or not. Check the *prov_capabilities* field in the service provider information structure (*provinfo_struct*) which is returned as output of *x_getinfo()*.
ER_BADNEG	8011	The negative flag is not the same (set or cleared) as in the previous function calls. If multiple calls to *x_conrsp()* are used to respond to one connect indication, the value of the NEGATIVE flag must be the same in all of these calls.
ER_NOINFOBUF	8012	No buffer has been specified to keep the service provider information returned by *x_getinfo()*.
ER_BADNAME	8013	An invalid service provider identifier, or an invalid transport system identifier has been specified.

ER_BADADDR	8014	The specified protocol address is in an incorrect format or contained illegal information.
ER_DATASIZE	8015	The amount of user data specified in the call exceeds the limits allowed by the service provider. Check the values specified in the service provider information structure (*provinfo_struct*) which is returned as output of *x_getinfo()*.
ER_NOADDR	8016	No own protocol address was specified in the *x_bind()* call and the service provider could not allocate one.
ER_ADDRBUSY	8017	There is already a service endpoint with *qlen* greater than 0 bound to the requested address.
ER_ACCESS	8018	The application does not have permission to use the specified address.
ER_NOSYCH	8019	The synchronous execution mode is not supported for this service endpoint.
ER_NOASYCH	8020	The asynchronous execution mode is not supported for this service endpoint.
ER_BADSEQ	8021	An invalid sequence number is specified in an *x_conrsp()* or *x_snddis()* call. If multiple calls to *x_conrsp()* are used to respond to one connect indication, the value of the sequence number must be the same in all of these calls.
ER_QUEUEFULL	8022	The maximum number of outstanding (unresponded) connect indications has been reached. The user has to accept or reject some of these connections before more connect indications can be received with *x_conind()*.
ER_BADQLEN	8023	At the time of activation, a queue length of zero has been specified for the endpoint referenced by *fd*. Thus, the endpoint must not be used with *x_conind()* calls to wait for incoming calls.
ER_WAITCONF	8024	The function *x_conreq()* was called in asynchronous mode and successfully initiated the connection establishment procedure, but did not wait for a response from the remote user. Now the local user has to arrange to wait for a connect confirmation, e.g. by calling *x_conconf()* in synchronous mode.
ER_MOREDATA	8025	The function *x_conreq()* was called in asynchronous mode with the MORE flag set. To complete the connect request, more user data is required; the local user has to make another *x_conreq()* call.
ER_INDOUTST	8026	The function *x_conrsp()* was called with *fd* equal to *resfd*, but there are outstanding connection indications on the endpoint. They must be handled, either by rejecting or accepting them on a different endpoint, before the endpoint identified by *fd* may be used in a connection.
ER_PROVMISM	8027	The file descriptors *fd* and *resfd* specified in the *x_conrsp()* call do not refer to the same service provider.
ER_RESQLEN	8028	The function *x_conrsp()* was called with *fd* not equal to *resfd*, and the endpoint referenced by *resfd* has been bound with a *qlen* that is greater than 0.

ER_RESTRANS	8029	The function *x_conrsp()* was called with *fd* not equal to *resfd*, and the endpoint referenced by *resfd* has used a different transport system beneath the application system.
ER_BADACTN	8030	An invalid action is specified in the *flags* field as input of *x_optmgmt()*.
ER_BADOPT	8031	At least one of the options submitted in the *x_optmgmt()* call specifies an illegal value.
ER_BADLEVEL	8032	The value specified as *level* in an *x_optmgmt()* or *x_sndsp()* call does not select any protocol module of the service provider. The identifiers of the protocol modules in the service provider‘s protocol stack are available in the *provinfo_struct* which is returned as output of *x_open()*, *x_bind()* and *x_getinfo()*.
ER_NOERRBUFF	8033	No *error_struct* has been specified for the output of *x_rcverror()*.
ER_NODIS	8034	No disconnect indication currently exists at the specified service endpoint.
ER_FLOW	8035	The sending function is called in asynchronous mode and the flow control mechanism prevents the service provider from accepting the request at this time.
ER_NOTSUPP	8036	This function is not supported by the underlying service provider.
ER_STATECHNG	8037	The service endpoint is just undergoing a state change and therefore the function *x_sync()* could not determine the current state.
ER_SYSERR	8041	An operating system call returned an error during execution of this XAPI function. More information about the failure might be gathered in an operating-system-dependent way.
ER_INTR	8042	The execution of this XAPI function was interrupted by a signal.
ER_PROTO	8043	This error indicates that a communication problem has been detected between XAPI and the service provider for which there is no other suitable XAPI error code defined.
ER_NOPARAM	8044	This error code may be returned by the functions *x_b2c()* and *x_b2l()* if there is no parameter value found in the buffer.
ER_NOSND	8045	A sending function was called but no further sending is allowed because the connection is destroyed. The events already received from the service provider may be consumed by the corresponding consuming functions (e.g. *x_rcvdis()*).
ER_ERROR	8046	An error indication has arrived. Call *x_error()*, please.
ER_LIMITS	8048	An system limit has been reached. When returned by *x_open()*, this indicates that the number of open file descriptors in the process or the system has been reached; when returned by a sending function, this indicates that a buffer was too long.
ER_CONFIG	8049	An error in accessing the service providers or transport systems configuration file has been detected.

ER_BADMODE	8050	An invalid mode has been specified.
ER_TRANSLINK	8051	The function *x_bind()* failed to link the specified transport system below the service providers application system. This error is returned for two reasons: The service provider has a built-in transport system which cannot be replaced by a dynamically linked transport system, or a system error occurred while dynamically linking the transport system. In this case more information about the reason of failure may be available on operating system level.
ER_NOEND	8052	The function *x_rcvend()* has been called, but no END event is available at the service endpoint.
ER_NOERROR	8054	The function *x_rcverror()* has been called, but no ERROR event is available at the service endpoint.
ER_NOCONREQ	8055	The function *x_conreq()* has been called on a passive service endpoint (i.e. when binding the endpoint with *x_bind()* the *qlen* field of *bind_struct* was greater than zero). Initiating a connection is only allowed on active service endpoints (*qlen* = 0).
ER_REM_GENRPC	8056	A general Remote Procedure Call (RPC) failure has occurred in the remote XAPI.
ER_REM_RPC_PROC	8057	The remote procedure call failed.
ER_REM_CALLBACK	8058	The remote XAPI failed to set up the callback channel.
ER_REM_NOSERVER	8059	The server process of the remote XAPI is not started or is terminated.

APPENDIX I

Examples of XAPI access to service providers

The access to the service providers specified in this appendix serve as examples. They do not claim completeness (e.g. the access to a multipoint transport service provider may be specified). The list of specifications presented in this appendix may be extended if required.

An application which requests access to a service has to select an appropriate service provider prior to the usage of the service. Thus the communication system can be tailored to specific requirements, and all communication services are accessible via one homogenous access mechanism.

Available providers

This appendix contains the following service providers:

I.1 XAPI access to the service provider for the ISDN B-channel;

I.2 XAPI access to the service provider for BFT over T.30;

I.3 XAPI access to the service provider for FAX4 and BFT;

I.4 XAPI access to the service provider for ACSE and ROSE;

I.5 XAPI access to the service provider for audio and video control;

I.6 XAPI access to the service provider for T.120 conference control;

I.7 XAPI access to the service provider for T.127 MBFT.

I.1 XAPI access to the service provider for the ISDN B-channel

This part of Appendix I describes an example of how the service provider can be implemented, if an application needs the access to the specified service.

I.1.1 Scope

The XAPI, which is an abbreviation for eXtensive Application Programming Interface, is an operating system and language-independent programming interface to general communication services. Detailed information about the XAPI is given in the main part of this Recommendation, which is very important for understanding this appendix.

Which services are made available via the XAPI depends on the installed service providers and not on the XAPI, which only provides the access mechanism.

This part describes the XAPI access to the ISDN B-channel.

Figure I.1-1 shows the structure of the protocol stack that is accessible via the XAPI when selecting the ISDN B-channel service:

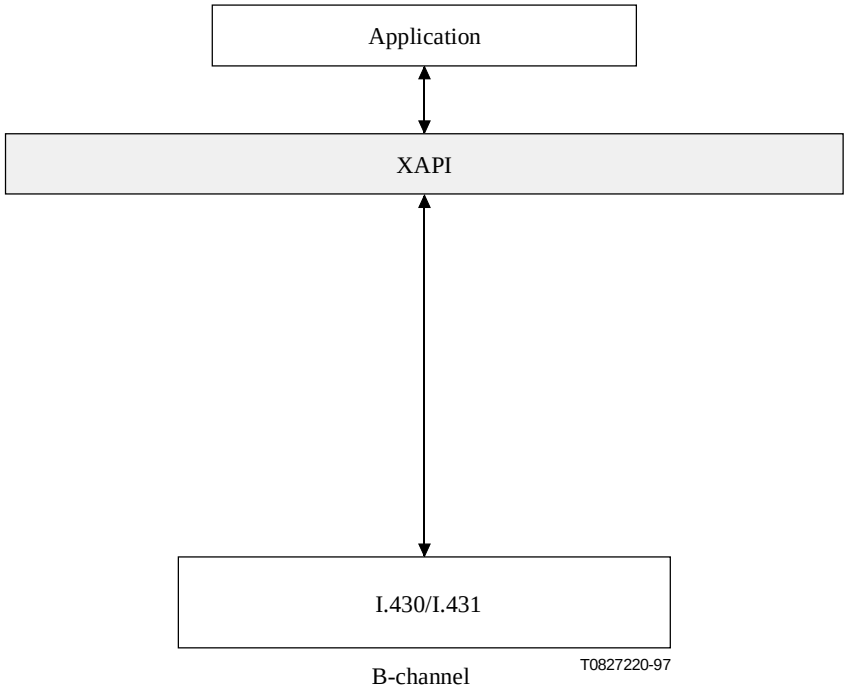


Figure I.1-1/T.180 – Structure of the ISDN B-channel service provider

I.1.2 References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; all users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published.

[I.430] ITU-T Recommendation I.430 (1995), *Basic user-network interface – Layer 1 specification.*

[I.431] ITU-T Recommendation I.431 (1993), *Primary rate user-network interface – Layer 1 specification.*

[Q.921] ITU-T Recommendation Q.921 (1997), *ISDN User-network interface – Data link layer specification.*

[ETS 300 129:1991-09] *Integrated Services Digital Network (ISDN); User-network interface data link layer specifications. Application of CCITT Recommendations Q.920/I.440 and Q.921/I.441.*

[Q.931] CCITT Recommendation Q.931 (1988), *ISDN user-network interface layer 3 specification for basic call control.*

[ETS 300 102-1:1990-12] *Integrated Services Digital Network (ISDN); User-network interface layer 3; Specifications for basic call control.*

I.1.3 Definitions

I.1.4 Abbreviations

This part uses the following abbreviations:

DSS 1 Digital Subscriber Signalling System No. 1

ISDN Integrated Services Digital Network

ISO International Organization for Standardization

NSAP Network Service Access Point

OSI Open Systems Interconnection

PHT abbreviation string specifying service specific definitions of the B-channel (PHysical) service provider, Transparent access (e.g. parameter names, service primitive names)

XAPI eXtensive Application Programming Interface

I.1.5 Conventions

Each service is described via three kinds of tables.

In the first kind of table the service and its service elements are described. It contains a row for each service element of the service (where service element means either request, indication, response, or confirmation) with its corresponding XAPI function and a short description.

In the second kind of table the use of the service primitive parameters within the service elements are described. It contains a row for each service primitive parameter of the service. The service primitive parameter name is stored within the first column, the succeeding columns contain the use of the parameter in the service elements:

Blank The service parameter is absent.

C Presence of the service parameter is conditional. Firstly, there may be a condition in the service provider to provide a parameter in an indication or confirmation. Secondly, there may be interdependencies between parameters of the same or the preceding service primitive.

M Presence of the service parameter is mandatory.

U Presence of the service parameter is a user option. If the user does not specify a value for such a parameter and there is no default value for that parameter, nothing is passed to the service provider. If the user does not specify a value for such a parameter and there is a default value, the default value is passed to the service provider.

(=) The value of the service parameter is identical to the value of the corresponding service parameter in the preceding service element. In the special case of a parameter, whose presence in the preceding service is a user option, for which a default value is defined, and the parameter was not specified in the preceding service element, the symbol (=) indicates that the parameter value is identical to the default value.

The third kind of table is used to describe the service primitive parameter containing the type of value and its scope, possible default values and a detailed description. For each service primitive parameter, one description table is given.

A **naming convention** is used throughout the XAPI in order to distinguish between the common functions, identifiers and values, and service specific identifiers and values. All service-specific settings of the XAPI ISDN B-channel service access are defined within the present part of this appendix and start with **X_PHT_** or **x_pht_**.

Extended naming conventions:

– SP_ Service Primitive;

– P_ Parameter;

– PV_ Parameter Value;

– O_ Option;

– OV_ Option Value.

I.1.6 Introduction to the ISDN physical service provider access

The ISDN B-channel service handles connection establishment, access to the B-channel for the data transfer and disconnection.

The XAPI gives a uniform access method; it is hidden from the XAPI user whether the D-channel or the B-channel is the destination for a special service request.

I.1.7 Description of the access to the ISDN physical service provider

I.1.7.1 Service initialization

I.1.7.1.1 Creation of an ISDN B-channel service access point with x_open()

A communication endpoint accessing the ISDN physical service provider is created when calling the *x_open()* function with the service provider identification string "**X_PHT_ISDN**".

The service provider comprises the D-channel protocols and the B-channel protocol module on the physical layer. Of course, no transport system has to be linked below it with the *x_bind()* function, as the protocol stack is already complete.

I.1.7.1.2 Activation of an ISDN B-channel service access point with x_bind()

x_bind() is to be called to activate the ISDN B-channel service endpoint. The function has the task to bind an address to the service endpoint.

No transport system has to be specified as argument of the *x_bind()* function, as the physical layer is the only layer which is needed in the B-channel.

I.1.7.1.3 Protocol addresses

The protocol address to be used is the NSAP address. Selectors are meaningless for the ISDN B-channel service.

I.1.7.1.3.1 The Application's own Address

The own address may be specified in the *own_address* buffer of the *bind_struct* passed as argument to the *x_bind()* function. For a passive application, it is returned in the *called_addr* buffer of the *x_conind()* function.

For a passive application, it is not supported to specify the own responding NSAP address in the *address* buffer of the *call_struct* in the *x_conrsp()* function, as this value is not transferred by the network.

NOTE – The specification of the application's own protocol address is completely optional. If no address information is specified, the own address is derived from system configuration information and the bound value is returned as output parameter of the *x_bind()* function.

The own address consists of the NSAP address only. The NSAP address has to comprise the local ISDN outband address, i.e. the address information used in the D-channel. The own ISDN inband address and subaddress parameters as well as the protocol selectors are meaningless in the ISDN B-channel service. If specified, they will be ignored.

Table I.1-1 shows the address component which has to be specified in the *x_bind()* call.

Table I.1-1/T.180 – Address component specified in the *x_bind()* call

ISDN network	Address component	Value
ISDN/DSS 1	A_OUTBAND_ADR	A decimal digit, which is locally mapped on the Multiple Subscriber Number (MSN)

I.1.7.1.3.2 The address of the communication partner

On the sending side, the address of the communication partner has to be specified in the *address* buffer of the *call_struct* passed as argument to the *x_conreq()* function. On the receiving side, the address of the communication partner is returned in the *calling_addr* buffer of the *x_conind()* function.

The address of the communication partner comprises at least the peer's ISDN outband address. The peer's ISDN inband address and subaddress as well as the protocol selectors are meaningless, as there is a data transfer in the B-channel without further protocols.

Table I.1-2 shows the address component to be used in the *address* buffer specifying the called NSAP address in an *x_conreq()* call.

Table I.1-2/T.180 – Address component specifying the called NSAP address in an *x_conreq()* call

ISDN network	Address component	Value
ISDN/DSS 1	A_OUTBAND_ADR	Optional the country code, optional the area code and the Multiple Subscriber Number (MSN)

I.1.7.1.4 Configuration of the service provider

The protocol module of the service provider behaves according to configured protocol options. Protocol options are used to control the general behaviour of a protocol module (they must not be confounded with service primitive parameters). The preconfigured values of the protocol options are sufficient for the majority of communication relations.

Currently, there are no options for the protocol modules of the access to the ISDN B-channel, which could be set by the XAPI function *x_optmgmt()*.

I.1.7.2 Connection Establishment service

I.1.7.2.1 Service description

During the connection establishment phase, two users of the same service establish a connection between them. The XAPI user must already have prepared a service endpoint before the connection establishment phase can start.

The service elements and their corresponding XAPI functions needed for Connection Establishment are described in Table I.1-3.

Table I.1-3/T.180 – Service elements and their corresponding XAPI functions for Connection Establishment

Service element	XAPI function	Description
Connect Request	x_conreq()	The Connect Request is passed to the provider to request the establishment of a connection.
Connect Indication	x_conind()	The Connect Indication is generated by the provider to indicate the request from a remote terminal to establish a connection.
Connect Response	x_conrsp()	The Connect Response is passed to the provider as reaction to a previously received connect indication as positive or negative response.
Connect Confirmation	x_conconf()	The Connect Confirmation is generated by the provider as positive or negative confirmation of a local establishment.

I.1.7.2.2 Service parameters

There are no service parameters to be specified during connection establishment.

I.1.7.3 Services in the connected state

I.1.7.3.1 Data Transfer service

I.1.7.3.1.1 Service description

The Data Transfer service can be used to transfer transparent data in the B-channel.

Flow control, i.e. the ability to read received data fast enough during the data transfer, is up to the XAPI user. Also, the segmentation of larger portions which shall be transmitted is up to the XAPI user.

The service elements and their corresponding XAPI functions needed for Data Transfer are described in Table I.1-4.

Table I.1-4/T.180 – Service elements and their corresponding XAPI functions for Data Transfer

Service elements	XAPI function	Description
Data Request	x_snddata()	The Data Request is passed to the provider to transmit data.
Data Indication	x_rcvdata()	The Data Indication is generated by the provider to indicate the received data.

I.1.7.3.1.2 Service parameters

There are no service parameters defined for the Data Transfer service.

I.1.7.4 Disconnect service

I.1.7.4.1 Service description

The Disconnect service allows either service user to disconnect the connection.

The service elements and their corresponding XAPI functions needed for Disconnection are described in Table I.1-5.

Table I.1-5/T.180 – Service elements and their corresponding XAPI functions for Disconnect

Service elements	XAPI function	Description
Disconnect Request	x_snddis()	The Disconnect Request is passed to the provider to request a disconnection.
Disconnect Indication	x_rcvdis()	The Disconnect Indication is generated by the provider to indicate the release of a connection due to peer or to provider.
End Indication	x_rcvend()	The End Indication is generated by the provider to indicate that the service provider is ready to establish a new connection. In addition, it contains some information about the released connection.

I.1.7.4.2 Service parameters

Table I.1-6 specifies the parameters needed for disconnection.

Table I.1-6/T.180 – Parameters of the Disconnect service

Parameter	Disconnect service		
	Request	Indication	End Indication
X_PHT_P_REASON		M	
X_P__CONN_TIME			M
X_P_DISC_TIME			M
X_P_CHARGE			C
X_P_DISC_REASON			C

I.1.7.4.3 Service parameter descriptions

Tables I.1-7 to I.1-11 define the parameters for the Disconnect service.

Table I.1-7/T.180

Parameter name	X_PHT_P_REASON
Type of value	unsigned long
Legal values	The values are directly mapped from the underlying hardware. See the hardware documentation to get further information.
Default value	none
Description	The parameter indicates the cause of the disconnection.

Table I.1-8/T.180

Parameter name	X_P_CONN_TIME
Type of value	unsigned long
Legal values	the time in seconds since 01.01.1970 00:00:00 GMT
Default value	none
Description	This parameter contains the time of physical connection.

Table I.1-9/T.180

Parameter name	X_P_DISC_TIME
Type of value	unsigned long
Legal values	the time in seconds since 01.01.1970 00:00:00 GMT
Default value	none
Description	This parameter contains the time of physical disconnection. If X_P_CONN_TIME and X_P_DISCON_TIME are both set to zero, no physical connection could be established.

Table I.1-10/T.180

Parameter name	X_P_CHARGE
Type of value	unsigned long
Legal values	any number
Default value	zero
Description	This parameter contains, if available, the charging unit of the connection. It is only set if both network and network connection provide this facility.

Table I.1-11/T.180

Parameter name	X_P_DISC_REASON
Type of value	unsigned long
Legal values	The values are directly mapped from the underlying hardware. See the hardware documentation to get further information.
Default value	zero
Description	This parameter contains, if available, the disconnection reason.

I.1.7.5 Usage of XAPI functions

This subclause provides some protocol-specific remarks to the use of the XAPI functions. The functions are mentioned in alphabetical order.

- x_conconf The user_data buffer in the *call_struct* is empty, as transfer of user data is not available in the connection establishment phase.
- x_conind The user_data buffer in the *conind_struct* is empty, as transfer of user data is not available in the connection establishment phase.
- x_conreq The user_data buffer in the *call_struct* is empty, as transfer of user data is not available in the connection establishment phase. Usage of the MORE flag is not supported.
- x_conrsp The user_data buffer in the *call_struct* is empty, as transfer of user data is not available in the connection establishment phase. Usage of the MORE flag is not supported.
- x_rcvdata Expedited data are not available.
- x_rcvdis The user_data buffer in the *discon_struct* is empty, as transfer of user data is not available in the disconnection.
- x_snddata Expedited data are not available. Usage of the MORE flag is not supported.
- x_snddis The user_data buffer in the *discon_struct* may not be used, as transfer of user data is not available in the disconnection. Usage of the MORE flag is not supported.

I.1.7.6 Tables of error codes

The XAPI error-level error codes are defined in Annex B.

I.1.7.6.1 CC_BADVALUE

An invalid mandatory parameter is given: *diagnostic* contains the corresponding parameter identifier.

I.1.7.6.2 CC_MANDMISS

A mandatory parameter is missing: *diagnostic* contains the number of missing mandatory parameters.

I.1.7.6.3 CC_BADEVENT

An invalid event is specified: *diagnostic* contains the bad event identifier.

I.1.7.6.4 CC_SEQ

An incorrect sequence number is given: *diagnostic* contains the bad sequence number.

I.1.7.6.5 CC_SPNAME

An invalid service primitive name is given: *diagnostic* contains the bad service primitive name.

I.1.7.6.6 CC_ADDCOMP

An additional parameter (neither address parameter nor service primitive parameter) is incorrect: no *diagnostic* is given.

I.1.7.6.7 CC_BADLENGTH

An address or parameter buffer contains an illegal length value: *diagnostic* contains the length.

I.1.7.6.8 CC_UNEXPECT

If the cause code indicates a unexpected event, the value of *diagnostic* will contain the actual state identifier in which the unexpected event caused the error indication.

I.1.7.6.9 CC_NOTSUPPORT

An event is given which is not supported: the value of *diagnostic* contains the identifier of the unsupported event which has been submitted with the XAPI function call that caused the error indication.

I.1.7.6.10 CC_OTHER

If the cause code indicates the CC_OTHER error code, the value of *diagnostic* contains the identifier which caused the error indication.

I.2 XAPI access to the service provider for BFT over T.30

This part of Appendix I describes as an example how the service provider can be implemented, if an application needs the access to the specified service.

I.2.1 Scope

The XAPI, which is an abbreviation for eXtensive Application Programming Interface, is an operating system and language-independent programming interface to general communication services. Detailed information about the XAPI is given in the main part of this Recommendation, which is very important for understanding this appendix.

Which services are made available via the XAPI, depends on the installed service providers, and not on the XAPI which only provides the access mechanism.

This part describes the XAPI access to the BFT(T.30) service provider.

Figure I.2-1 shows the structure of the protocol stack that is accessible via the XAPI when selecting the BFT(T.30) service provider:

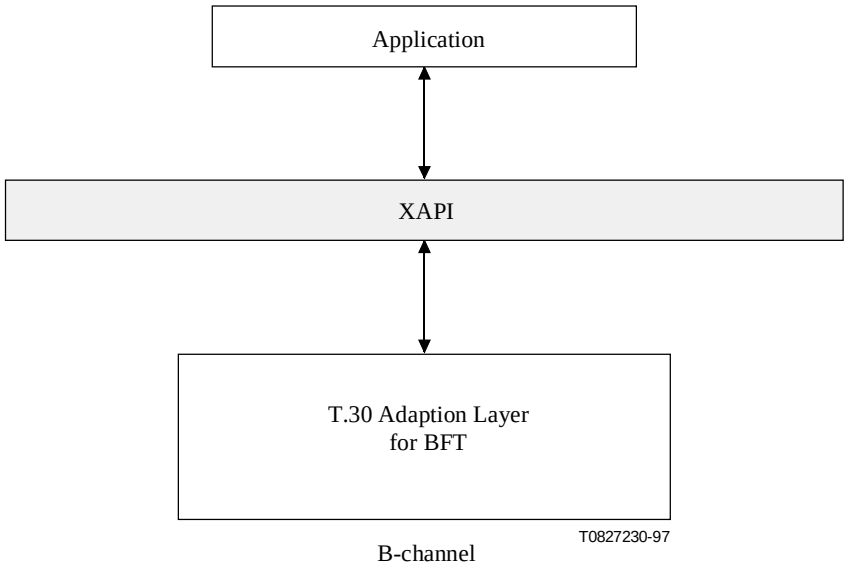


Figure I.2-1/T.180 – Structure of the BFT(T.30) service provider

I.2.2 References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; all users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published.

[T.4] ITU-T Recommendation T.4 (1996), *Standardization of Group 3 facsimile terminals for document transmission.*

[T.30] ITU-T Recommendation T.30 (1996), *Procedures for document facsimile transmission in the general switched telephone network.*

[T.434] ITU-T Recommendation T.434 (1996), *Binary file transfer format for the telematic services.*

I.2.3 Definitions

This part defines the following terms:

I.2.3.1 service provider: A communication system that provides a certain service to user.

I.2.3.2 transport system: A protocol stack comprising some or all the OSI layer 1 (physical layer) to 4 (transport layer)

I.2.4 Abbreviations

This part uses the following abbreviations:

BF3 acronym string specifying service specific definitions of the BFT(T.30) provider (e.g. parameter names, service primitive names)

BFT Binary File Transfer

DSS 1	Digital Subscriber Signalling System No. 1
ISDN	Integrated Services Digital Network
ISO	International Organization for Standardization
NSAP	Network Service Access Point
OSI	Open Systems Interconnection
TWA	Two-Way Alternate
XAPI	eXtensive Application Programming Interface

I.2.5 Conventions

Each service is described via three kinds of tables.

In the first table, the service and its service elements are described. It contains a row for each service element of the service (where service element means either request, indication, response, or confirmation) with its corresponding XAPI function and a short description.

In the second table, the use of the service primitive parameters within the service elements are described. It contains a row for each service primitive parameter of the service. The service primitive parameter name is stored within the first column; the following columns contain the use of the parameter in the service elements:

Blank The service parameter is absent.

C Presence of the service parameter is conditional. Firstly, there may be a condition in the service provider to provide a parameter in an indication or confirmation. Secondly, there may be interdependencies between parameters of the same or the preceding service primitive.

M Presence of the service parameter is mandatory.

U Presence of the service parameter is a user option. If the user does not specify a value for such a parameter and there is no default value for that parameter, nothing is passed to the service provider. If the user does not specify a value for such a parameter and there is a default value, the default value is passed to the service provider.

(=) The value of the service parameter is identical to the value of the corresponding service parameter in the preceding service element. In the special case of a parameter, whose presence in the preceding service is a user option, for which a default value is defined, and the parameter was not specified in the preceding service element, the symbol (=) indicates that the parameter value is identical to the default value.

The third kind of table is used to describe the service primitive parameter containing the type of value and its scope, possible default values and a detailed description. For each service primitive parameter, one description table is given.

A **naming convention** is used throughout the XAPI in order to distinguish between the common functions, identifiers and values, and service specific identifiers and values. All service-specific settings of the XAPI BFT(T.30) service access are defined within the present part of this appendix and start with **X_BF3_** or **x_bf3_**.

Extended naming conventions:

– SP_ Service Primitive;

– P_ Parameter;

– PV_ Parameter Value;

– O_ Option;

– OV_ Option Value.

I.2.6 Introduction to the BFT(T.30) provider access

The BFT(T.30) service is provided by a T.30 adaption module, which transmits BFT [T.434] encoded files according to Recommendation [T.30].

It gives the means of synchronized Binary File Transfer between cooperating Telematic applications: An application can:

- establish a connection with another BFT user;
- exchange binary files in a synchronized manner; and
- release a connection.

There are no restrictions which are made for implementation.

This implementation expects data encoded according to Binary File Transfer format [T.434] in each *x_snddata()* and delivers with each *x_rcvdata()* encoded data in the same format.

I.2.7 Description of the access to the BFT(T.30) provider

I.2.7.1 Service initialization

I.2.7.1.1 Creation of a BFT(T.30) Service Access Point with x_open()

A communication endpoint accessing the BFT(T.30) service provider is created when calling the *x_open()* function with the service provider identification string "**X_BFT_T.30_ISDN**".

The service provider comprises the D-channel protocols and the B-channel protocol module (T.30 adaptation) on the physical layer. Of course, no transport system has to be linked below it with the *x_bind()* function, as the protocol stack is already complete.

I.2.7.1.2 Activation of BFT(T.30) Service Access Point with x_bind()

x_bind() is to be called to activate the BFT(T.30) service endpoint. The function has the task to bind an address to the service endpoint.

No transport system has to be specified as argument of the *x_bind()* function, as the T.30 adaptation layer is the only layer which is needed in the B-channel.

I.2.7.1.3 Protocol addresses

The protocol address to be used is the NSAP address. Selectors are meaningless for the BFT(T.30) service.

I.2.7.1.3.1 The Application's own Address

The own address may be specified in the *own_address* buffer of the *bind_struct* passed as argument to the *x_bind()* function. It is returned in the *called_addr* buffer of the *x_conind()* function.

For a passive application it is not supported to specify the own responding NSAP address in the *address* buffer of the *call_struct* in the *x_conrsp()* function, as this value is not transferred by the network.

NOTE – The specification of the application's own protocol address is completely optional. If no address information is specified, the own address is derived from system configuration information and the bound value is returned as output parameter of the *x_bind()* function.

The own address consists of the NSAP address only. The NSAP address has to comprise the local ISDN outband address, i.e. the address information used in the D-channel. The own ISDN inband address and subaddress parameters as well as the protocol selectors are meaningless in the BFT(T.30) service. If specified, they will be ignored.

Table I.2-1 shows the address component which has to be specified in the *x_bind()* call.

Table I.2-1/T.180 – Address component specified in the *x_bind()* call

ISDN network	Address component	Value
ISDN/DSS 1	A_OUTBAND_ADR	A decimal digit, which is locally mapped on the MSN (multiple subscriber number)

I.2.7.1.3.2 The address of the communication partner

On the sending side the address of the communication partner has to be specified in the *address* buffer of the *call_struct* passed as argument to the *x_conreq()* function. On the receiving side the address of the communication partner is returned in the *calling_addr* buffer of the *x_conind()* function.

The address of the communication partner comprises at least the peer's ISDN outband address. The peer's ISDN inband address and subaddress as well as the protocol selectors are meaningless, as there is a transparent data transfer in the B-channel without further protocols.

Table I.2-2 shows the address component to be used in the *address* buffer specifying the called NSAP address in an *x_conreq()* call.

Table I.2-2/T.180 – Address component specifying the called NSAP address in an *x_conreq()* call

ISDN network	Address component	Value
ISDN/DSS 1	A_OUTBAND_ADR	Optional the country code, optional the area code, and the Multiple Subscriber Number (MSN)

I.2.7.1.4 Configuration of the service provider

The protocol module of the service provider behaves according to configured protocol options. Protocol options are used to control the general behaviour of a protocol module (they must not be confounded with service primitive parameters). The preconfigured values of the protocol options are sufficient for the majority of communication relations.

Currently, there are no options for the protocol modules of the BFT(T.30) service provider which could be set by the XAPI function *x_optmgmt()*.

I.2.7.2 Connection Establishment service

I.2.7.2.1 Service description

During connection establishment phase two users of the same service establish a connection between each other. Both users must have already prepared an active service endpoint before they can enter the connection establishment phase.

The service elements and their corresponding XAPI functions needed for BFT(T.30) Connection Establishment are described in Table I.2-3.

Table I.2-3/T.180 – Service elements and their corresponding XAPI functions for Connection Establishment

Service element	XAPI function	Description
Connect Request	x_conreq()	The Connect Request service is passed to the provider to request the establishment of a BFT(T.30) connection.
Connect Indication	x_conind()	The Connect Indication service is generated by the provider to indicate the request from a remote terminal to establish a BFT(T.30) connection.
Connect Response	x_conrsp()	The Connect Response service is passed to the provider as reaction to a previously received connect indication as positive or negative response.
Connect Confirmation	x_conconf()	The Connect Confirmation service is generated by the provider as positive or negative confirmation of a local BFT(T.30) establishment.

I.2.7.2.2 Service parameters

There are no service parameters defined for the Connection Establishment service.

I.2.7.3 Services in the connected state

The T.30 adaption layer for BFT provides several services in the connected state. While the service endpoint used to access the provider is in state X_CONNECTED these service primitives can be passed to the provider rsp. retrieved from the provider with calls of *x_snddata()* and *x_sndsp()* rsp. *x_rcvdata()* and *x_rcvsp()*.

The data transfer works in a so called TWA (Two-Way Alternate) mode. This means the initiator and responder act in a sink/source relationship, in which only one of them may send data at a time. The owner of the Data Token, which is the initiator of the connection when entering the X_CONNECTED state, may then give control to the responder by issuing the X_BF3_SP_GIVE_TOKEN_Q service primitive.

I.2.7.3.1 Data Transfer service

I.2.7.3.1.1 Service description

The Data Transfer service allows the BFT(T.30) service users to transfer files alternately.

The service elements and their corresponding XAPI functions needed for Data Transfer are described in Table I.2-4.

Table I.2-4/T.180 – Service elements and their corresponding XAPI functions for Data Transfer

Service element	XAPI function	Description
Data Request	x_snddata()	The Data Request service is passed to the provider to transmit data.
Data Indication	x_rcvdata()	The Data Indication service is generated by the provider to indicate the received data.

I.2.7.3.1.2 Service parameters

There are no service parameters defined for the Data Transfer service.

I.2.7.3.2 Checkpoint service

I.2.7.3.2.1 Service description

The BFT(T.30) service uses a checkpoint mechanism in the file transfer. This means each checkpoint in BFT mode is confirmed by the receiver.

The service elements and their corresponding XAPI functions needed for the Checkpoint service are described in Table I.2-5.

Table I.2-5/T.180 – Service elements and their corresponding XAPI functions for the Checkpoint service

Service element	XAPI function	Service element identifier	Description
Checkpoint Request	x_sndsp()	X_BF3_SP_CHK_Q	The Checkpoint Request is passed to the provider to request the checkpointing from the active side.
Checkpoint Indication	x_rcvsp()	X_BF3_SP_CHK_I	The Checkpoint Indication is generated by the provider at the passive side to indicate the reception of a Checkpoint Request.
Checkpoint Response	x_sndsp()	X_BF3_SP_CHK_P	The Checkpoint Response is passed to the provider to confirm the checkpoint at the passive side.
Checkpoint Confirmation	x_rcvsp()	X_BF3_SP_CHK_C	The Checkpoint Confirmation is given to the sender of the Checkpoint Request as acknowledgement of the command.

I.2.7.3.2.2 Service parameter

Table I.2-6 specifies the parameters of the Checkpoint service.

Table I.2-6/T.180 – Parameters of the Checkpoint service

Parameter	Checkpoint service			
	Request	Indication	Response	Confirmation
X_BF3_P_CHK_NR	M	M (=)	M (=)	M (=)

I.2.7.3.2.3 Service parameter description

Table I.2-7 describes the parameter for the Checkpoint service.

Table I.2-7/T.180

Parameter name	X_BF3_P_CHK_NR
Type of value	unsigned long
Legal values	any number greater than zero
Default value	none
Description	This parameter identifies the current checkpoint.

I.2.7.3.3 End of File service

I.2.7.3.3.1 Service description

The BFT(T.30) service uses a checkpoint mechanism in the file transfer. If the last data was transmitted, the End of File service is used.

The service elements and their corresponding XAPI functions needed for the End of File service are described in Table I.2-8.

Table I.2-8/T.180 – Service elements and their corresponding XAPI functions for the End of File service

Service element	XAPI function	Service element identifier	Description
End of File Request	x_sndsp()	X_BF3_SP_EOF_Q	The End of File Request is passed to the provider to request the final checkpoint from the active side.
End of File Indication	x_rcvsp()	X_BF3_SP_EOF_I	The End of File Indication is generated by the provider at the passive side to indicate the reception of the final checkpoint.
End of File Response	x_sndsp()	X_BF3_SP_EOF_P	The End of File Response is passed to the provider to confirm the final checkpoint at the passive side.
End of File Confirmation	x_rcvsp()	X_BF3_SP_EOF_C	The End of File Confirmation is sent to the originator of the End of File Request as acknowledgement of the command.

I.2.7.3.3.2 Service parameter

Table I.2-9 specifies the parameter of the End of File service.

Table I.2-9/T.180 – Parameter of the End of File service

Parameter	Checkpoint service			
	Request	Indication	Response	Confirmation
X_BF3_P_CHK_NR	M	M (=)	M (=)	M (=)

I.2.7.3.3.3 Service parameter description

Table I.2-10 describes the parameter for the Checkpoint service.

Table I.2-10/T.180

Parameter name	X_BF3_P_CHK_NR
Type of value	unsigned long
Legal values	any number greater than zero
Default value	none
Description	This parameter identifies the current checkpoint.

I.2.7.3.4 Data Token service

I.2.7.3.4.1 Service description

As stated above, data transfer works in a so-called TWA (Two-Way Alternate) mode. This means, the initiator and the responder act in a sink/source relationship, in which only one of them may send data at a time. The owner of the Data Token may give control to the responder by issuing the X_BF3_SP_GIVE_TOKEN_Q service primitive.

The service elements and their corresponding XAPI functions needed for the Data Token service are described in Table I.2-11.

Table I.2-11/T.180 – Service elements and their corresponding XAPI functions for the Data Token service

Service element	XAPI function	Service element identifier	Description
Data Token Request	x_sndsp()	X_BF3_SP_DATA_TOKEN_Q	The Data Token Request is passed to the provider to hand over the right for transmitting data from the active side to the passive side. The active side becomes now the passive side and the passive side becomes now the active side.
DataToken Indication	x_rcvsp()	X_BF3_SP_DATA_TOKEN_I	The Data Token Indication is generated by the provider to indicate that the remote side has handed over the right for transmitting data. The active side becomes now the passive side and the passive side becomes now the active side.

I.2.7.3.4.2 Service parameters

There are no service parameters defined for the Data Token service.

I.2.7.3.5 States in the connected state

In the BFT(T.30) protocol, the data transfer is managed in eight states, which may be entered by the service provider while the endpoint used for access is in state X_CONNECTED:

States which are entered, if the service provider is the initiator (active side) (see Figure I.2-2):

state 10 This is the initial state. It is entered when the service endpoint enters state X_CONNECTED. The service user is now able to transmit data with x_snddata(). This is the only state from which an X_BF3_SP_DATA_TOKEN_Q can be issued. In all data transfer states abortive disconnection has to be used to end the communication.

state 11 This state is entered after sending data with x_snddata(). In this state, the user may send more data (x_snddata()), send X_BF3_SP_CHK_Q to indicate a checkpoint, or X_BF3_SP_EOF_Q to indicate the end of the file. This state is also entered when an X_BF3_SP_CHK_C was received, which indicates that the checkpoint is confirmed.

state 12 This state is entered when an X_BF3_SP_CHK_Q was sent and an X_BF3_SP_CHK_C is pending.

state 13 This state is entered, when an X_BF3_SP_EOF_Q was sent and an X_BF3_SP_EOF_C is pending.

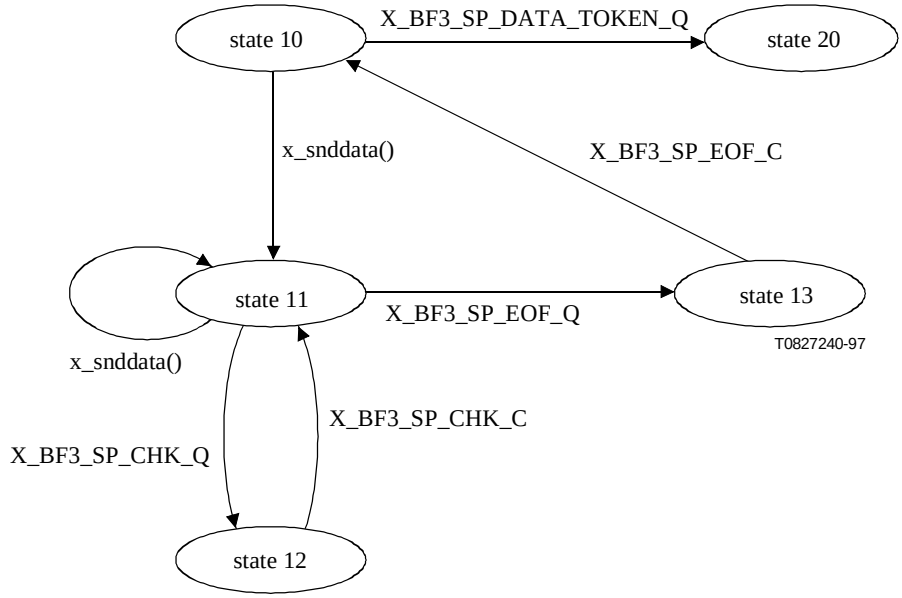


Figure I.2-2/T.180 – Data transfer states in active mode

States which are entered, if the service provider is the responder (receiving side) (see Figure I.2-3):

state 20 This is the initial state. It is entered when the service endpoint enters state X_CONNECTED. The service user may now receive Data with the function x_rcvdata(). If an X_BF3_SP_GIVE_TOKEN_I was received, the automaton enters the state10. In all data transfer states abortive disconnection has to be used to end the communication.

state 21 This state is entered after receiving data with x_rcvdata(). In this state, the user may receive more data (x_rcvdata()), receive X_BF3_SP_CHK_I to indicate a checkpoint, or X_BF3_SP_EOF_I to indicate the end of the file. This state is also entered, when an X_BF3_SP_CHK_P was sent, which indicates that the checkpoint is confirmed.

state 22 This state is entered when an X_BF3_SP_CHK_I was received, which indicates a checkpoint. The service user has now to confirm this checkpoint by sending an X_BF3_SP_CHK_P.

state 23 This state is entered when an X_BF3_SP_EOF_I was received, which indicates the successfully transmission of the whole file. The service user has now to confirm the end of the file with X_BF3_SP_EOF_P.

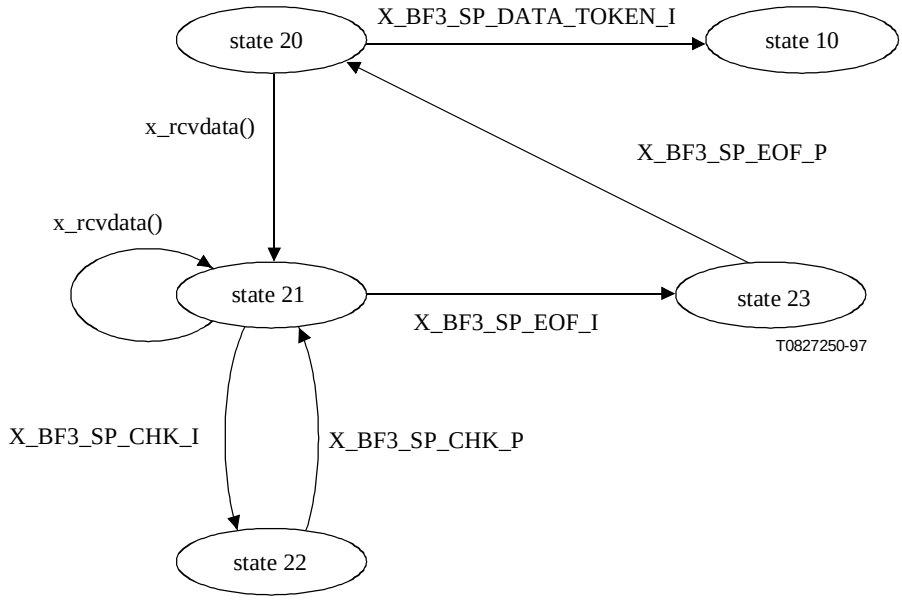


Figure I.2-3/T.180 – Data transfer states in passive mode

I.2.7.4 Connection Abort service

I.2.7.4.1 Service description

The abort service provides the means by which either BFT(T.30) service user or provider can instantaneously release the connection and have the other BFT(T.30) service user informed of the release. Use of this service will cause loss of undelivered data.

The service elements and their corresponding XAPI functions needed for abort of BFT(T.30) connection are described in Table I.2-12.

Table I.2-12/T.180 – Service elements and their corresponding XAPI functions for the Connection Abort service

Service element	XAPI function	Description
Abort Request	x_snddis()	The Abort Request is passed to the provider to request an abnormal BFT(T.30) connection release.
Abort Indication	x_rcvdis()	The Abort Indication is generated by the provider to indicate the abnormal release of a BFT(T.30) connection.
End Indication	x_rcvend()	The End Indication is generated by the provider to indicate that the service provider is ready to establish a new connection.

I.2.7.4.2 Service parameter

Table I.2-13 specifies the parameter needed for a abortive BFT(T.30) release.

Table I.2-13/T.180 – Parameters of the Connection Abort service

Parameter	Abort service		
	Request	Indication	End Indication
X_BF3_P_REASON		U	

I.2.7.4.3 Service parameter description

Table I.2-14 defines the parameter for the abort service.

Table I.2-14/T.180

Parameter name	X_FX3_P_REASON	
Type of value	unsigned long	
Legal values	X_FX3_PV_R_NOREASON	/* no reason */
	X_FX3_PV_R_NOBFT	/* connection establishment not possible, no */ /* facsimile terminal reached */
	X_FX3_PV_R_SLOW	/* error during file transfer, delivered */ /* data too slow */
	X_FX3_PV_R_NOANS	/* abort because of incorrect behaviour of remote terminal */
	X_FX3_PV_R_REMDISC	/* remote disconnect */
	X_FX3_PV_R_NOCMD	/* unexpected disconnect during file reception */
	X_FX3_PV_R_INCOMPAT	/* Transfer mode is not supported by remote terminal, e.g. */ /* high resolution, file transfer */
	X_FX3_PV_R_BADDATA	/* error during file transfer, wrong encoded BFT */
	X_FX3_PV_R_PROTO	/* protocol error, the remote terminal does not */ /* conform to T.30 */
Default value	X_FX3_PV_R_NOREASON	
Description	The parameter indicates the cause of the disconnection.	

I.2.7.5 Usage of XAPI functions

This subclause provides some protocol-specific remarks to XAPI functions. The functions are mentioned in alphabetical order. If a function is not listed, there are no special remarks.

- x_conconf The user_data buffer in the *call_struct* is empty, as transfer of user data is not available in the connection establishment phase.
- x_conind The user_data buffer in the *conind_struct* is empty, as transfer of user data is not available in the connection establishment phase.
- x_conreq The user_data buffer in the *call_struct* is empty, as transfer of user data is not available in the connection establishment phase. Usage of the MORE flag is not supported.
- x_conrsp The user_data buffer in the *call_struct* is empty, as transfer of user data is not available in the connection establishment phase. Usage of the MORE flag is not supported.

- x_rcvdis The user_data buffer in the *discon_struct* is empty, as transfer of user data is not available in the disconnection.
- x_snddata This function can only be called if BFT(T.30) service provider is in state 10 or state 11. Expedited data is not available at the XAPI BFT(T.30) service access point. Usage of the MORE flag is not supported.
- x_snddis The user_data buffer in the *discon_struct* may not be used, as transfer of user data is not available in the disconnection. Usage of the MORE flag is not supported.
- x_sndsp Usage of the MORE flag is not supported.

I.2.7.6 Tables of error codes

The XAPI error-level error codes are defined in Annex B.

I.2.7.6.1 CC_BADVALUE

An invalid mandatory parameter is given: *diagnostic* contains the corresponding parameter identifier.

I.2.7.6.2 CC_MANDMISS

A mandatory parameter is missing: *diagnostic* contains the number of missing mandatory parameters.

I.2.7.6.3 CC_BADEVENT

An invalid event is specified: *diagnostic* contains the bad event identifier.

I.2.7.6.4 CC_SEQ

An incorrect sequence number is given: *diagnostic* contains the bad sequence number.

I.2.7.6.5 CC_SPNAME

An invalid service primitive name is given: *diagnostic* contains the bad service primitive name.

I.2.7.6.6 CC_ADDCOMP

An additional parameter (neither address parameter nor service primitive parameter) is incorrect: no *diagnostic* is given.

I.2.7.6.7 CC_BADLENGTH

An address or parameter buffer contains an illegal length value: *diagnostic* contains the length.

I.2.7.6.8 CC_UNEXPECT

If the cause code indicates an unexpected event: the value of *diagnostic* will contain the actual state identifier in which the unexpected event caused the error indication. Table I.2-15 contains the values defined for the *diagnostic* parameter.

Table I.2-15/T.180

Name	Description
X_BF3_STATE_10	A service is requested, which is not expected in this state. Only the following services are expected: • X_BF3_SP_DATA_TOKEN_Q • x_snddata() • x_snddis()
X_BF3_STATE_11	A service is requested, which is not expected in this state. Only the following services are expected: • x_snddata() • X_BF3_SP_CHK_Q • X_BF3_SP_EOF_Q • x_snddis()
X_BF3_STATE_12	A service is requested, which is not expected in this state. Only the following service is expected: • x_snddis()
X_BF3_STATE_13	A service is requested, which is not expected in this state. Only the following service is expected: • x_snddis()
X_BF3_STATE_20	A service is requested, which is not expected in this state. Only the following service is expected: • x_snddis()
X_BF3_STATE_21	A service is requested, which is not expected in this state. Only the following service is expected: • x_snddis()
X_BF3_STATE_22	A service is requested, which is not expected in this state. Only the following services are expected: • X_BF3_SP_CHK_P • x_snddis()
X_BF3_STATE_23	A service is requested, which is not expected in this state. Only the following services are expected: • X_BF3_SP_EOF_P • x_snddis()

I.2.7.6.9 CC_NOTSUPPORT

An event is given which is not supported: the value of *diagnostic* contains the identifier of the unsupported event which has been submitted with the XAPI function call that caused the error indication.

I.2.7.6.10 CC_OTHER

If the cause code indicates the CC_OTHER error code, the value of *diagnostic* contains the identifier which caused the error indication.

I.2.7.7 Table of service primitives

Table I.2-16/T.180

Name	Description
X_BF3_SP_DATA_TOKEN_Q	Data token request
X_BF3_SP_DATA_TOKEN_I	Data token indication
X_BF3_SP_CHK_Q	Checkpoint request
X_BF3_SP_CHK_I	Checkpoint indication
X_BF3_SP_CHK_P	Checkpoint response
X_BF3_SP_CHK_C	Checkpoint confirmation
X_BF3_SP_EOF_Q	End of file request
X_BF3_SP_EOF_I	End of file indication
X_BF3_SP_EOF_P	End of file response
X_BF3_SP_EOF_C	End of file confirmation

I.2.7.8 Table of service primitive parameters

Table I.2-17/T.180

Name	Legal values
X_BF3_P_CHK_NR	number greater than 0
X_BF3_P_REASON	X_FX3_PV_R_NOREASON X_FX3_PV_R_NOBFT X_FX3_PV_R_SLOW X_FX3_PV_R_NOANS X_FX3_PV_R_REMDISC X_FX3_PV_R_NOCMD X_FX3_PV_R_INCOMPAT X_FX3_PV_R_BADDATA X_FX3_PV_R_PROTO

I.3 XAPI access to the service provider for FAX4 and BFT

This part of Appendix I describes an example of how the service provider can be implemented, if an application needs the access to the specified service.

I.3.1 Scope

The XAPI, which is an abbreviation for eXtensive Application Programming Interface, is an operating system and language-independent programming interface to general communication services. Detailed information about the XAPI is given in the main part of this Recommendation, which is very important for understanding this appendix.

Which services are made available via the XAPI depends on the installed service providers and not on the XAPI, which only provides the access mechanism.

This part describes the XAPI access to the FAX4/BFT service provider.

Figure I.3-1 shows the structure of the protocol stack that is accessible via the XAPI when selecting the FAX4/BFT service provider:

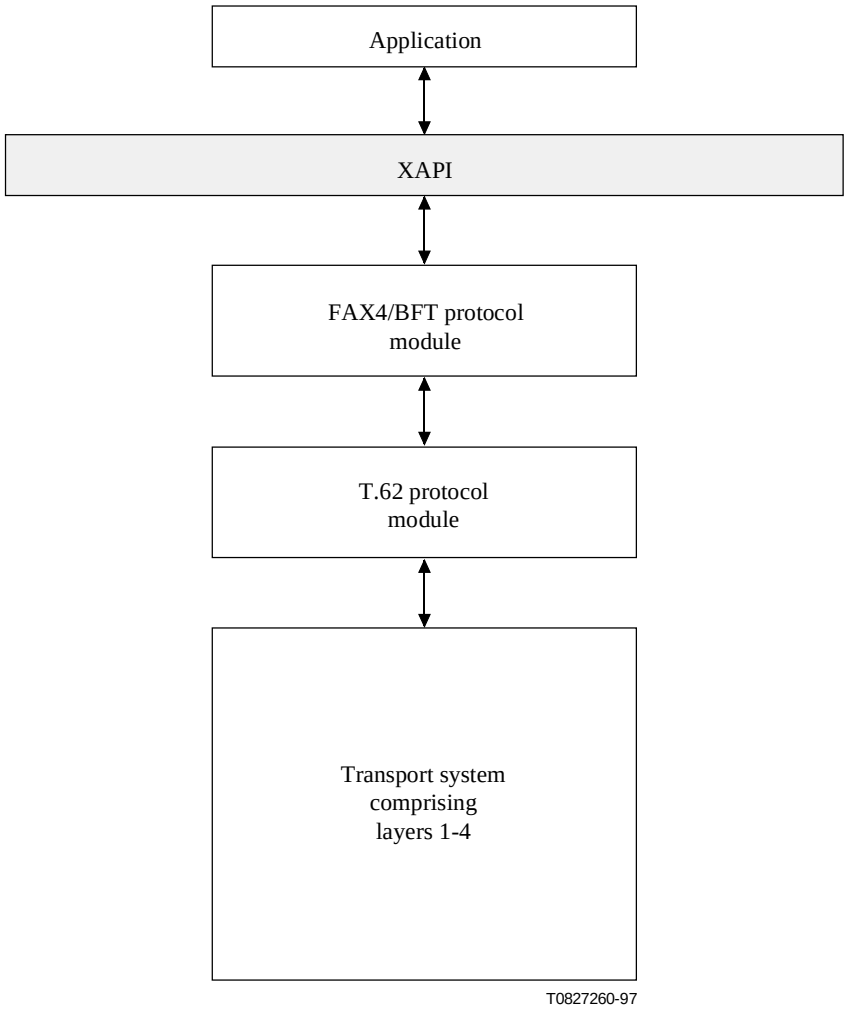


Figure I.3-1/T.180 – Structure of the FAX4/BFT service provider

The XAPI user is able to select one transport system (comprising layers 1 to 4) among the set of transport systems available in the XAPI communication platform to act as the underlying transport service provider.

I.3.2 References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations are subject to revision; all users of this Recommendation and other references are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published.

[T.62] ITU-T Recommendation T.62 (1993), *Control procedures for teletex and Group 4 facsimile services.*

[T.434] ITU-T Recommendation T.434 (1996), *Binary file transfer format for the telematic services.*

[T.503] CCITT Recommendation T.503 (1991), *A document application profile for the interchange of Group 4 facsimile documents.*

[T.563] ITU-T Recommendation T.563 (1996), *Terminal characteristics for Group 4 facsimile apparatus.*

[T.571] CCITT Recommendation T.571 (1992*), Terminal characteristics for the telematic file transfer within the teletex service.*

I.3.3 Definitions

I.3.4 Abbreviations

This part uses the following abbreviations:

BFT Binary File Transfer

DSS 1 Digital Subscriber Signalling System No. 1

FX4 acronym string specifying service specific definitions of the FAX4/BFT provider (e.g. parameter names, service primitive names)

HDLC High-Level Data-Link Control

ISDN Integrated Services Digital Network

ISO International Organization for Standardization

NSAP Network Service Access Point

OSI Open Systems Interconnection

XAPI eXtensive Application Programming Interface

I.3.5 Conventions

Each service is described via three kinds of tables.

In the first kind of table the service and its service elements are described. It contains a row for each service element of the service (where service element means either request, indication, response, or confirmation) with its corresponding XAPI function and a short description.

In the second kind of table the use of the service primitive parameters within the service elements are described. It contains a row for each service primitive parameter of the service. The service primitive parameter name is stored within the first column; the following columns contain the use of the parameter in the service elements:

Blank The service parameter is absent.

C Presence of the service parameter is conditional. Firstly, there may be a condition in the service provider to provide a parameter in an indication or confirmation. Secondly, there may be interdependencies between parameters of the same or the preceding service primitive.

M Presence of the service parameter is mandatory.

U Presence of the service parameter is a user option. If the user does not specify a value for such a parameter and there is no default value for that parameter, nothing is passed to the service provider. If the user does not specify a value for such a parameter and there is a default value, the default value is passed to the service provider.

(=) The value of the service parameter is identical to the value of the corresponding service parameter in the preceding service element. In the special case of a parameter, whose presence in the preceding service is a user option, for which a default value is defined, and the parameter was not specified in the preceding service element, the symbol (=) indicates that the parameter value is identical to the default value.

The third kind of table is used to describe the service primitive parameter containing the type of value and its scope, possible default values and a detailed description. For each service primitive parameter, one description table is given.

A **naming convention** is used throughout the XAPI in order to distinguish between the common functions, identifiers and values and service-specific identifiers and values. All service-specific settings of the XAPI FAX4/BFT service access are defined within the present part of this appendix and start with **X_FX4_** or **x_fx4_**.

Extended naming conventions:

– SP_ Service Primitive;

– P_ Parameter;

– PV_ Parameter Value;

– O_ Option;

– OV_ Option Value.

I.3.6 Introduction to the FAX4/BFT service provider access

The FAX4/BFT service is provided by the combination of a transport system selected by the application and the FAX4/BFT protocol module. The FAX4/BFT protocol module is implemented according to the Recommendations [T.503], [T.563], [T.434] and [T.571] and resides on top of [T.62].

It gives the means of synchronized FAX4/Binary File Transfer between cooperating Telematic applications: An application can:

- establish a connection with another FAX4/BFT user;
- exchange facsimile/binary files in a synchronized manner; and
- release connection in an orderly manner.

Each protocol element is directly mapped to the corresponding session protocol element of T.62. The only restriction is that resuming of interrupted documents is currently not accessible to the user of the FAX4/BFT service.

The implementation expects T.6 coded Data in case of Facsimile Group 4 and in case of BFT any binary data in each *x_snddata()* call. All ASN.1 coding is done within the service provider. The checkpointing (page boundary) must be initiated by the service user. In case of BFT a checkpoint should be inserted every 2 kilo-octets (according to Recommendation T.571) but other values may be used to improve transmission speed.

I.3.7 Description of the access to the FAX4/BFT service provider

I.3.7.1 Service initialization

I.3.7.1.1 Selection of the FAX4/BFT service provider with x_open()

A communication endpoint accessing the FAX4/BFT service provider is created when calling the *x_open()* function with an appropriate service provider identification string. The available identifiers

depend on the actual system configuration. In the standard configuration, **"X_FAX4_BFT_ISDN"** identifies the service provider which comprises the FAX4/BFT and T.62 session protocol modules as well as a transport system based on the ISDN network. On the protocol levels two to four, there are implementations of HDLC LAP B: ISO/IEC 8208 and Recommendation T.70.

If this service provider is selected, it is not necessary to link a transport system below it (using the *x_bind()* function) as the protocol stack is already complete.

I.3.7.1.2 Selection of the underlying transport system with x_bind()

x_bind() is to be called to activate the FAX4/BFT service endpoint. The function has the following tasks:

- if the service provider, which has been selected with *x_open()*, does not comprise a transport system, link a transport system below the available protocol modules;
- bind an address to the service endpoint.

If the service provider X_FAX4_BFT_ISDN has been selected in the *x_open()* function, no transport system has to be specified in the *x_bind()* function.

The list of all service providers and transport systems available in the standard configuration (and the protocol modules they comprise) is contained in Annex B.

I.3.7.1.3 Protocol addresses

The reader should be familiar with those general concepts and the terminology, as the present part of this appendix relies on the explanations given there and just informs about the service specifics.

The protocol address to be used for the BFT/FAX4 service is the NSAP address. Selectors are meaningless for the FAX4/BFT service.

I.3.7.1.3.1 The application's own address

The own address may be specified in the *own_address* buffer of the *bind_struct* passed as argument to the *x_bind()* function. For a passive application it is returned in the *called_addr* buffer of the *x_conind()* function.

For a passive application it is not supported to specify the own responding NSAP address in the *address* buffer of the *call_struct* in the *x_conrsp()* function, as this value is not transferred by the network.

Note that specification of the application's own protocol address is completely optional. If no address information is specified, the own address is derived from system configuration information and the bound value is returned as output parameter of the *x_bind()* function.

The own address consists of the NSAP address only. The NSAP address has to comprise the local ISDN outband address, i.e. the address information used in the D-channel. The own ISDN inband address and subaddress parameters as well as the protocol selectors are meaningless in the FAX4/BFT service. If specified, they will be ignored.

Table I.3-1 shows the address component which has to be specified in the *x_bind*() call.

Table I.3-1/T.180 – Address component specified in the *x_bind()* call

ISDN network	Address component	Value
ISDN/DSS 1	A_OUTBAND_ADR	A decimal digit, which is locally mapped on the Multiple Subscriber Number (MSN)

I.3.7.1.3.2 The address of the communication partner

On the sending side, the address of the communication partner has to be specified in the *address* buffer of the *call_struct* passed as argument to the *x_conreq()* function. On the receiving side, the address of the communication partner is returned in the *calling_addr* buffer of the *x_conind()* function.

The address of the communication partner comprises at least the peer's ISDN outband address. The peer's ISDN inband address and subaddress as well as the protocol selectors are meaningless for the BFT/FAX4 service.

Table I.3-2 shows the address component to be used in the *address* buffer specifying the called NSAP address in an *x_conreq()* call.

Table I.3-2/T.180 – Address component specifying the called NSAP address in an *x_conreq()* call

ISDN network	Address component	Value
ISDN/DSS 1	A_OUTBAND_ADR	Optional the country code, optional the area code and the Multiple Subscriber Number (MSN)

I.3.7.1.4 Configuration of the service provider

The protocol modules of the service provider behave according to configured protocol options. Protocol options are used to control the general behaviour of a protocol module (they must not be confounded with service primitive parameters). The preconfigured values of the protocol options are sufficient for the majority of communication relations.

Currently, there are no options for the protocol modules of the FAX4/BFT service provider, which could be set by the XAPI function *x_optmgmt()*.

I.3.7.2 Connection Establishment service

I.3.7.2.1 Service description

During the connection establishment phase, two users of the same service establish a connection between each other. Both users must have already prepared an active service endpoint before they can enter the connection establishment phase.

The service elements and their corresponding XAPI functions needed for FAX4/BFT Connection Establishment are described in Table I.3-3.

Table I.3-3/T.180 – Service elements and their corresponding XAPI functions for Connection Establishment

Service element	XAPI function	Description
Connect Request	x_conreq()	The Connect Request service is passed to the provider to request the establishment of a FAX4/BFT connection.
Connect Indication	x_conind()	The Connect Indication service is generated by the provider to indicate the request from a remote terminal to establish a FAX4/BFT connection.
Connect Response	x_conrsp()	The Connect Response service is passed to the provider as reaction to a previously received connect indication as positive or negative response.
Connect Confirmation	x_conconf()	The Connect Confirmation service is generated by the provider as positive or negative confirmation of a local FAX4/BFT establishment.

I.3.7.2.2 Service parameters

Table I.3-4 specifies the parameters of the FAX4/BFT Connection Establishment service.

Table I.3-4/T.180 – Parameters of the Connection Establishment service

Parameter	Connect Service			
	Request	Indication	Response	Confirmation
X_FX4_P_OWN_TERMID		M		M
X_FX4_P_FAR_TERMID	U	C (=)		U
X_FX4_P_REASON				C
X_FX4_P_PLEASE_TOKEN			U	C (=)
X_FX4_P_DATE_TIME		M		M

I.3.7.2.3 Service parameter descriptions

Tables I.3-5 to I.3-9 describe the parameters for the Connection Establishment service.

Table I.3-5/T.180

Parameter name	X_FX4_P_OWN_TERMID
Type of value	char [X_FX4_C_MAX_TERMID]
Legal values	A sequence of characters not '\0' terminated and no longer as X_FX4_C_MAX_TERMID characters.
Default value	none
Description	The parameter indicates the own terminal identifier as described in Recommendation F.200[1] with a maximum length of X_FX4_C_MAX_TERMID characters. The terminal identifier consists of up to 6 parts, namely: – the DNIC (Data Network Identification Code) (maximum 4 characters); – separator "-"; – the international phone number (maximum 12 characters); – optional an extension (maximum 4 characters); – separator "-"; – mnemonic part (minimum 3 characters); The sum of all parts should not exceed 24 characters.

Table I.3-6/T.180

Parameter name	X_FX4_P_FAR_TERMID
Type of value	char [X_FX4_C_MAX_TERMID]
Legal values	A sequence of characters not '\0' terminated and no longer as X_FX4_C_MAX_TERMID characters.
Default value	empty string
Description	The parameter indicates the remote terminal identifier as described in recommendation F.200[1] with a maximum length of X_FX4_C_MAX_TERMID characters. The terminal identifier consist of up to 6 parts, namely: – the DNIC (Data Network Identification Code) (maximum 4 characters) – separator "-"; – the international phone number (maximum 12 characters); – optional an extension (maximum 4 characters); – separator "-"; – mnemonic part (minimum 3 characters); The sum of all parts should not exceed 24 characters.

[1] Recommendation F.200, *Teletex service*: This Recommendation has been withdrawn.

Table I.3-7/T.180

Parameter name	X_FX4_P_REASON
Type of value	long
Legal values	X_FX4_PV_R_NO_REASON /* no error reason given */ X_FX4_PV_R_NO_CTX /* no more contexts of this type */ X_FX4_PV_R_LOCERR /* local terminal error */ X_FX4_PV_R_PROCERR /* procedure error */ X_FX4_PV_R_LOCPROCERR /* local terminal and procedure error */ X_FX4_PV_R_SSNAC /* session can not be accepted */ X_FX4_PV_R_ERRMSG /* error message exists (text) */ X_FX4_PV_R_NOMEM /* no memory available */ X_FX4_PV_R_TRANSERR /* transmission error */ X_FX4_PV_R_PNOERR /* wrong page number */ X_FX4_PV_R_FMTERR /* format error */ X_FX4_PV_R_RJCAP /* capabilities not supported */ X_FX4_PV_R_SVOK /* return code positive */ X_FX4_PV_R_SVNAV /* service not available */
Default value	X_FX4_PV_R_NO_REASON
Description	This parameter indicates the reason for a negative response or for an error request.

Table I.3-8/T.180

Parameter name	X_FX4_P_PLEASE_TOKEN
Type of value	unsigned long
Legal values	PV_TRUE PV_FALSE
Default value	PV_FALSE
Description	This parameter indicates a request control, which is used, if the transmission token is requested. The current owner of the transmission token may ignore this request or may give the transmission right to the passive side by issuing the X_FX4_SP_GIVE_TOKEN_Q service primitive. Initially, the initiator of a connection is the owner of the transmission token.

Table I.3-9/T.180

Parameter name	X_FX4_P_DATE_TIME
Type of value	char[X_FX4_C_MAX_DATE]
Legal values	A sequence of chars, not '\0' terminated, up to X_FX4_C_MAX_DATE characters long.
Default value	none
Description	The parameter indicates the connect time and has the following format: YY-MM-DD-HH:mm with YY = year MM = month DD = day HH = hour mm = minutes

I.3.7.3 Services in the connected state

The FAX4/BFT protocol module provides several services in the connected state. While the service endpoint used to access the provider is in state X_CONNECTED these service primitives can be passed to the provider rsp. retrieved from the provider with calls of *x_snddata()*, *x_sndsp()* rsp. *x_rcvdata()*, *x_rcvsp()*.

The data transfer works in a so called TWA (Two-Way Alternate) mode. This means the initiator and responder act in a sink/source relationship, in which only one of them may send data at a time. The responder may signal in some responses [x_conrsp(), X_FX4_SP_EOP_P, X_FX4_SP_EOD_P] a request control to indicate that the responder has information to transmit. The owner of the Data Token, which is the initiator of the connection when entering the X_CONNECTED state, may then give control to the responder by issuing the X_FX4_SP_GIVE_TOKEN_Q service primitive. If the token was exchanged, it must be exchanged again before an orderly release can be initiated. This means in general:

- the initiator of the connection owns the Data Token;
- the responder may ask for the token (X_FX4_PV_PLEASE_TOKEN);
- the initiator may hand over the Data Token by the service primitive X_FX4_SP_GIVE_TOKEN_Q;
- if the responder has got the Data Token, he must return the Data Token to the initiator by issuing the service primitive X_FX4_SP_GIVE_TOKEN_Q before releasing the connection;
- only the initiator can initiate an orderly release request.

I.3.7.3.1 Start of Document service

I.3.7.3.1.1 Service description

The Start of Document service indicates the start of the document to the receiver of the document. It also indicates the start of the first page.

The service elements and their corresponding XAPI functions needed for the Start of Document are described in Table I.3-10.

Table I.3-10/T.180 – Service elements and their corresponding XAPI functions for Start of Document

Service element	XAPI function	Service element identifier	Description
Start of Document Request	x_sndsp()	X_FX4_SP_SOD_Q	The Start of Document Request is passed to the provider to request the start of the document transmission from the active side.
Start of Document Indication	x_rcvsp()	X_FX4_SP_SOD_I	The Start of Document Indication is generated by the provider to indicate that the reception of a document has started by the passive side.

Table I.3-10/T.180 – Service elements and their corresponding XAPI functions for Start of Document *(concluded)*

Service element	XAPI function	Service element identifier	Description
Start of Document Confirmation	x_rcvsp()	X_FX4_SP_SOD_C	The Start of Document confirmation is sent to the sender of the Start of Document Request as acknowledgement of the command. The user should examine the parameters to check if the capabilities were accepted.

I.3.7.3.1.2 Service parameters

Table I.3-11 specifies the parameters of the Start of Document service.

Table I.3-11/T.180 – Parameters of the Start of Document service

Parameter	Start of Document service		
	Request	Indication	Confirmation
X_FX4_P_DOC_TYPE	M	M (=)	
X_FX4_P_BFT_FNAME	MC[a)]	C (=)	
X_FX4_P_BFT_PATH	MC[a)]	C (=)	
X_FX4_P_BFT_COMPR	MC[a)]	C (=)	
X_FX4_P_FAX_DENSITY	UC[b)]	C (=)	C (=)
X_FX4_P_FAX_COMPRESSION	UC[b)]	C (=)	C (=)
X_FX4_P_FAX_DIMENSION	UC[b)]	C (=)	C (=)
X_FX4_P_DOCREF	M	M (=)	
X_FX4_P_ACCEPT			M
X_FX4_P_REASON			M

a) Presence depends on the value of the parameter X_FX4_P_DOC_TYPE. If the value of X_FX4_P_DOC_TYPE is X_FX4_PV_FAX, this parameter is mandatory.

b) Presence depends on the value of the parameter X_FX4_P_DOC_TYPE. If set to X_FX4_PV_BFT, presence of this parameter is a user option.

I.3.7.3.1.3 Service parameter descriptions

Tables I.3-12 to I.3-21 describe the parameters for the Start of Document service.

Table I.3-12/T.180

Parameter name	X_FX4_P_DOC_TYPE
Type of value	long
Legal values	X_FX4_PV_FAX X_FX4_PV_BFT
Default value	none
Description	This parameter indicates the type of document which should be transmitted. The value could be either X_FX4_PV_FAX or X_FX4_PV_BFT. In case of X_FX4_PV_FAX, the document content has to contain T.6 encoded data, or any other type of binary data.

Table I.3-13/T.180

Parameter name	X_FX4_P_BFT_FNAME
Type of value	char [X_FX4_C_MAX_NAME]
Legal values	Any sequence of characters not containing '\0'. The length is restricted to X_FX4_C_MAX_NAME.
Default value	none
Description	This parameter indicates the file name of the binary file which should be transmitted. This parameter may only be used, if the X_FX4_P_DOC_TYPE is X_FX4_PV_BFT.

Table I.3-14/T.180

Parameter name	X_FX4_P_BFT_PATH
Type of value	char [X_FX4_C_MAX_PATH]
Legal values	Any sequence of characters not containing '\0'. The length is restricted to X_FX4_C_MAX_PATH.
Default value	none
Description	This parameter indicates the path name of the binary file which should be transmitted. This parameter may only be used, if the X_FX4_P_DOC_TYPE is X_FX4_PV_BFT.

Table I.3-15/T.180

Parameter name	X_FX4_P_BFT_COMPR
Type of value	char [X_FX4_C_MAX_NAME]
Legal values	Any sequence of characters not containing '\0'. The length is restricted to X_FX4_C_MAX_NAME.
Default value	none
Description	This parameter indicates the compression mode in which the binary file is transmitted. This parameter may only be used, if the X_FX4_P_DOC_TYPE is X_FX4_PV_BFT. This parameter should be omitted if the file is not compressed. Since Recommendation T.434 does not make any restriction on the contents of this attribute, the sender and receiver agree upon a string for the used compression algorithm.

Table I.3-16/T.180

Parameter name	X_FX4_P_FAX_DENSITY
Type of value	unsigned long
Legal values	One of the following values: X_FX4_PV_DENS_200; X_FX4_PV_DENS_240; X_FX4_PV_DENS_300; X_FX4_PV_DENS_400.
Default value	X_FX4_PV_DENS_200
Description	This parameter indicates the density of the FAX4 document which should be transmitted. This parameter may only be used if the X_FX4_P_DOC_TYPE = X_FX4_PV_FAX.

Table I.3-17/T.180

Parameter name	X_FX4_P_FAX_DIMENSION
Type of value	unsigned long
Legal values	X_FX4_PV_DIM_A4 X_FX4_PV_DIM_B4 X_FX4_PV_DIM_A3 X_FX4_PV_DIM_AM_LETTER X_FX4_PV_DIM_AM_LEGAL X_FX4_PV_DIM_AM_LEDGER X_FX4_PV_DIM_JAP_LETTER X_FX4_PV_DIM_JAP_LEGAL
Default value	X_FX4_PV_DIM_A4.
Description	This parameter indicates the density of the FAX4 document which should be transmitted. This parameter may only be used if the X_FX4_P_DOC_TYPE is X_FX4_PV_FAX.

Table I.3-18/T.180

Parameter name	X_FAX4_P_FAX_COMPRESSION
Type of value	unsigned long
Legal values	PV_TRUE PV_FALSE
Default value	PV_TRUE
Description	This parameter indicates the T.6 compression mode within the FAX4 document which should be transmitted. This parameter may only be used if the X_FX4_P_DOC_TYPE is X_FX4_PV_FAX.

Table I.3-19/T.180

Parameter name	X_FX4_P_DOCREF
Type of value	char [X_FX4_C_MAX_DOC_REF]
Legal values	Any sequence of the characters '0', '1', '2', ..., '9' not containing '\0'. The length is restricted to X_FX4_C_MAX_DOC_REF.
Default value	none
Description	This parameter indicates the document reference number. This number should start by 1 and should be incremented for every document, which is transmitted in the same connection. Leading zeros will be ignored.

Table I.3-20/T.180

Parameter name	X_FX4_P_ACCEPT
Type of value	unsigned long
Legal values	PV_TRUE PV_FALSE
Default value	PV_TRUE
Description	This parameter indicates if a Connect Request or Start of Document Request was accepted (= PV_TRUE) or rejected (= PV_FALSE).

Table I.3-21/T.180

Parameter name	X_FX4_P_REASON		
Type of value	long		
Legal values	X_FX4_PV_R_NO_REASON X_FX4_PV_R_NO_CTX X_FX4_PV_R_LOCERR X_FX4_PV_R_PROCERR X_FX4_PV_R_LOCPROCERR X_FX4_PV_R_SSNAC X_FX4_PV_R_ERRMSG X_FX4_PV_R_NOMEM X_FX4_PV_R_TRANSERR X_FX4_PV_R_PNOERR X_FX4_PV_R_FMTERR X_FX4_PV_R_RJCAP X_FX4_PV_R_SVOK X_FX4_PV_R_SVNAV	/* no error reason given /* no more contexts of this type /* local terminal error /* procedure error /* local terminal and procedure error /* session can not be accepted /* error message exists (text) /* no memory available /* transmission error /* wrong page number /* format error /* capabilities not supported /* return code positive /* service not available	*/ */ */ */ */ */ */ */ */ */ */ */ */ */
Default value	X_FX4_PV_R_NO_REASON		
Description	This parameter indicates the reason for a negative response or for an error request.		

I.3.7.3.2 Data Transfer service

I.3.7.3.2.1 Service description

The Data Transfer service allows both FAX4/BFT service users to transfer documents alternately.

The service elements and their corresponding XAPI functions needed for Data Transfer are described in Table I.3-22.

Table I.3-22/T.180 – Service elements and their corresponding XAPI functions for Data Transfer

Service element	XAPI function	Description
Data Request	x_snddata()	The Data Request service is passed to the provider to transmit real data (T.6 data or binary data).
Data Indication	x_rcvdata()	The Data Indication service is generated by the provider to indicate the received data (T.6 data or binary data).

I.3.7.3.2.2 Service parameter

Table I.3-23 specifies the parameter of the Data Transfer service.

Table I.3-23/T.180 – Parameters of the Data Transfer service

Parameter	Data Transfer service	
	Request	Indication
X_FX4_P_BLOCK_NR	M	M

I.3.7.3.2.3 Service parameter description

Table I.3-24 describes the parameter for the Data Transfer service.

Table I.3-24/T.180

Parameter name	X_FX4_P_BLOCK_NR
Type of value	unsigned long
Legal values	any number
Default value	none
Description	The parameter indicates the current block number in the current page.

I.3.7.3.3 Page Boundary service

I.3.7.3.3.1 Service description

The FAX4/BFT service uses a checkpoint mechanism in the document transfer. This means each page (or checkpoint in BFT mode) is confirmed by the receiver.

The service elements and their corresponding XAPI functions needed for the Page Boundary service are described in Table I.3-25.

Table I.3-25/T.180 – Service elements and their corresponding XAPI functions for Page Boundary

Service element	XAPI function	Service element identifier	Description
End of Page Request	x_sndsp()	X_FX4_SP_EOP_Q	The End of Page Request is passed to the provider to request the checkpointing from the active side.
End of Page Indication	x_rcvsp()	X_FX4_SP_EOP_I	The End of Page Indication is generated by the provider to indicate the reception of a checkpoint by the passive side.
End of Page Response	x_sndsp()	X_FX4_SP_EOP_P	The End of Page Response is passed to the provider to confirm the checkpoint from the active side.
End of Page Confirmation	x_rcvsp()	X_FX4_SP_EOP_C	The End of Page Confirmation is given to the sender of the End of Page Request as acknowledgement of the command. The user should examine the parameters to check if the page was accepted.

I.3.7.3.3.2 Service parameters

Table I.3-26 specifies the parameters of the Page Boundary service.

Table I.3-26/T.180 – Parameters of the Page Boundary service

Parameter	Page Boundary service			
	Request	Indication	Response	Confirmation
X_FX4_P_PAGE_NR	M	M (=)	M (=)	M (=)
X_FX4_P_REC_JEOP			U	C (=)
X_FX4_P_ACCEPT			U	C (=)
X_FX4_P_REASON			U	C (=)
X_FX4_P_PLEASE_TOKEN			U	C (=)

I.3.7.3.3.3 Service parameter descriptions

Tables I.3-27 to I.3-31 describe the parameters for the Page Boundary service.

Table I.3-27/T.180

Parameter name	X_FX4_P_PAGE_NR
Type of value	unsigned long
Legal values	any number greater than zero
Default value	none
Description	This parameter indicates the current page number.

Table I.3-28/T.180

Parameter name	X_FX4_P_REC_JEOP
Type of value	unsigned long
Legal values	PV_TRUE PV_FALSE
Default value	PV_FALSE
Description	This parameter indicates if the receiver can not receive more pages. If this parameter is set to PV_TRUE, the sender should abort the transmission.

Table I.3-29/T.180

Parameter name	X_FX4_P_ACCEPT
Type of value	unsigned long
Legal values	PV_TRUE PV_FALSE
Default value	PV_TRUE
Description	This parameter indicates if a Connect Request, Start of Document Request or End of Page Request was accepted (= PV_TRUE) or rejected (= PV_FALSE).

Table I.3-30/T.180

Parameter name	X_FX4_P_REASON
Type of value	long
Legal values	X_FX4_PV_R_NO_REASON /* no error reason given */ X_FX4_PV_R_LOCERR /* local terminal error */ X_FX4_PV_R_PROCERR /* procedure error */ X_FX4_PV_R_NOMEM /* no memory available */ X_FX4_PV_R_TRANSERR /* transmission error */ X_FX4_PV_R_PNOERR /* wrong page number */ X_FX4_PV_R_FMTERR /* format error */
Default value	X_FX4_PV_R_NO_REASON
Description	This parameter indicates the reason of a negative response or error request.

Table I.3-31/T.180

Parameter name	X_FX4_P_PLEASE_TOKEN
Type of value	unsigned long
Legal values	PV_TRUE PV_FALSE
Default value	PV_FALSE
Description	This parameter indicates a request control which is used if the transmission token is requested. The current owner of the transmission token may ignore this request or may give the transmission request to the passive side by issuing the X_FX4_SP_GIVE_TOKEN_Q service primitive.

I.3.7.3.4 End of document service

I.3.7.3.4.1 Service description

The FAX4/BFT service uses a checkpoint mechanism in the document transfer. This means each page (or checkpoint in BFT mode) is confirmed from the receiver. If the last page was transmitted, the End of Document service is used instead of End of Page.

The service elements and their corresponding XAPI functions needed for the End of Document service are described in Table I.3-32.

Table I.3-32/T.180 – Service elements and their corresponding XAPI functions for End of Document

Service element	XAPI function	Service element identifier	Description
End of Document Request	x_sndsp()	X_FX4_SP_EOD_Q	The End of Document Request is passed to the provider to request the final checkpoint from the active side.
End of Document Indication	x_rcvsp()	X_FX4_SP_EOD_I	The End of Document Indication is generated by the provider to indicate the reception of the final checkpoint by the passive side.
End of Document Response	x_sndsp()	X_FX4_SP_EOD_P	The End of Document Response is passed to the provider to confirm the final checkpoint from the active side.
End of Document Confirmation	x_rcvsp()	X_FX4_SP_EOD_C	The End of Document Confirmation is sent to the originator of the Start of Document Request as acknowledgement of the command.

I.3.7.3.4.2 Service parameters

Table I.3-33 specifies the parameters of the End of Document service.

Table I.3-33/T.180 – Parameters of the End of Document service

Parameter	End of Document service			
	Request	Indication	Response	Confirmation
X_FX4_P_PAGE_NR	M	M (=)	M (=)	M (=)
X_FX4_P_PLEASE_TOKEN			U	C (=)

I.3.7.3.4.3 Service parameter descriptions

Tables I.3-34 and I.3-35 describe the parameters for the End of Document service.

Table I.3-34/T.180

Parameter name	X_FX4_P_PAGE_NR
Type of value	unsigned long
Legal values	any number greater than zero
Default value	none
Description	This parameter indicates the current page number.

Table I.3-35/T.180

Parameter name	X_FX4_P_PLEASE_TOKEN
Type of value	unsigned long
Legal values	PV_TRUE PV_FALSE
Default value	PV_FALSE
Description	This parameter indicates a request control which is used if the transmission token is requested. The current owner of the transmission token may ignore this request or may give the transmission request to the passive side by issuing the X_FX4_SP_GIVE_TOKEN_Q service primitive.

I.3.7.3.5 Resynchronize document service

I.3.7.3.5.1 Service description

The FAX4/BFT service uses a resynchronization mechanism for an abnormal end of a document transfer. This means the initiator of the document transfer indicates the passive side the abnormal end of the document transfer phase; this indication is passed to the XAPI user.

The service elements and their corresponding XAPI functions needed for the Resynchronize Document service are described in Table I.3-36.

Table I.3-36/T.180 – Service elements and their corresponding XAPI functions for Resynchronize Document

Service element	XAPI function	Service element identifier	Description
Resynchronize Document Indication	x_rcvsp()	X_FX4_SP_RSYN_I	The Resynchronize Document Indication is an indication of the abnormal end of the document transfer.

I.3.7.3.5.2 Service parameter

Table I.3-37 specifies the parameter of the Resynchronize Document service.

Table I.3-37/T.180 – Parameters of the Resynchronize Document service

Parameter	Resynchronize Document service
	Indication
X_FX4_P_REASON	C

I.3.7.3.5.3 Service parameter description

Table I.3-38 describes the parameter for the Resynchronize Document service.

Table I.3-38/T.180

Parameter name	X_FX4_P_REASON		
Type of value	long		
Legal values	X_FX4_PV_R_NO_REASON X_FX4_PV_R_LOCERR X_FX4_PV_R_PROCERR X_FX4_PV_R_NOMEM X_FX4_PV_R_TRANSERR X_FX4_PV_R_PNOERR X_FX4_PV_R_FMTERR	/* no error reason given */ /* local terminal error */ /* procedure error */ /* no memory available */ /* transmission error */ /* wrong page number */ /* format error */	
Default value	X_FX4_PV_R_NO_REASON		
Description	This parameter indicates the reason for an abnormal ending of a document.		

I.3.7.3.6 Give Token service

I.3.7.3.6.1 Service description

The data transfer works in a so called TWA (Two-Way Alternate) mode. This means the initiator and responder acts in a sink/source relationship, in which only one of them may send data at a time. The responder may signal in some responses [x_conrsp(), X_FX4_SP_EOP_P, X_FX4_SP_EOD_P] a request control to indicate that the responder has information to transmit. The owner of the Data Token may then give control to the responder by issuing the X_FX4_SP_GIVE_TOKEN_Q service primitive. If the token was exchanged, it must be exchanged again before an orderly release can be initiated. This means in general:

- the initiator of the connection owns the Data Token;
- the responder may ask for the token (X_FX4_PV_PLEASE_TOKEN);
- the initiator may hand over the Data Token by the service primitive X_FX4_SP_GIVE_TOKEN_Q;
- if the responder has received the Data Token, he must return the Data Token to the initiator by issuing the service primitive X_FX4_SP_GIVE_TOKEN_Q before releasing the connection;
- only the initiator can initiate an orderly release request.

The service elements and their corresponding XAPI functions needed for the Give Token service are described in Table I.3-39.

Table I.3-39/T.180 – Service elements and their corresponding XAPI functions for Give Token

Service element	XAPI function	Service element identifier	Description
Give Token Request	x_sndsp()	X_FX4_SP_GIVE_TOKEN_Q	The Give Token Request is passed to the provider to hand over the right for transmitting data from the active side to the passive side. The active side now becomes the passive side and the passive side now becomes the active side.
Give Token Indication	x_rcvsp()	X_FX4_SP_GIVE_TOKEN_I	The Give Token Indication is generated by the provider to indicate that the remote side has handed over the right for transmitting data. The active side now becomes the passive side and the passive side now becomes the active side.

I.3.7.3.6.2 Service parameters

There are no service parameters defined for the Give Token service.

I.3.7.3.7 States in the connected state

In the FAX4/BFT protocol the data transfer is managed in ten states, which may be entered by the service provider while the endpoint used for access is in state X_CONNECTED:

States which are entered, if the service provider is the initiator (active side) (see Figure I.3-2):

state 10 This is the initial state. It is entered when the service endpoint enters state X_CONNECTED. Orderly release may be started from this state only. In all other data transfer states abortive disconnection has to be used to end the communication. This is also the only state from which an X_FX4_SP_GIVE_TOKEN_Q can be issued.

state 11 This state is entered when an X_FX4_SP_SOD_Q was sent and an X_FX4_SP_SOD_C is pending.

state 12 This state is entered when a positive X_FX4_SP_SOD_C was received. The service user is now able to transmit data with *x_snddata()*.

state 13 This state is entered after sending data with *x_snddata()*. In this state, the user may send more data (*x_snddata()*), send X_FX4_SP_EOP_Q to indicate a page boundary, or X_FX4_SP_EOD_Q to indicate the end of the document.
This state is entered, when an X_FX4_SP_EOP_C was received, which indicates that the checkpoint is confirmed.

state 14 This state is entered when an X_FX4_SP_EOP_Q was sent and an X_FX4_SP_EOP_C is pending.

state 15 This state is entered when an X_FX4_SP_EOD_Q was received and an X_FX4_SP_EOD_C is pending.

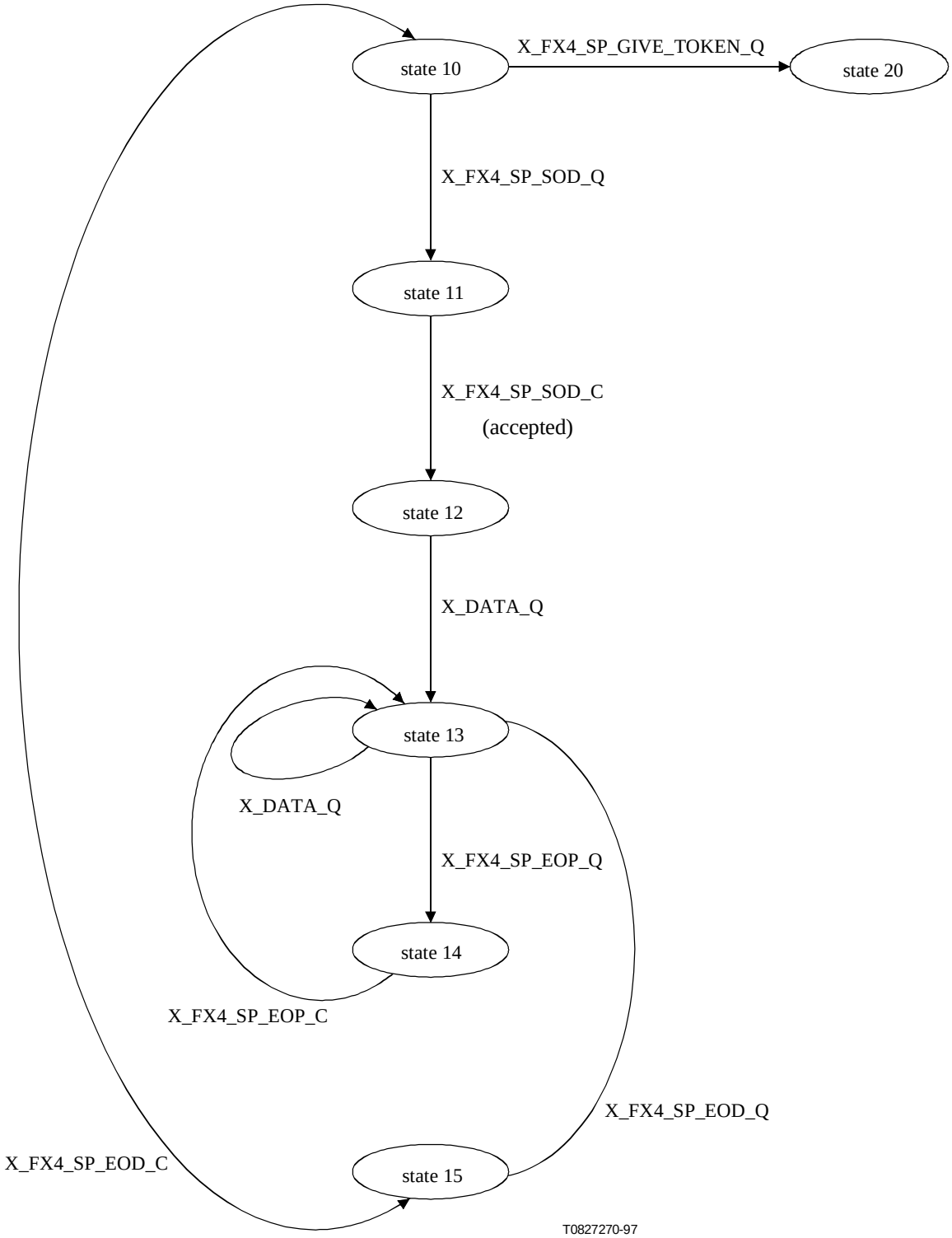


Figure I.3-2/T.180 – Data transfer states in active mode

States which are entered, if the service provider is the responder (receiving side) (see Figure I.3-3):

state 20 This is the initial state. It is entered when the service endpoint enters state X_CONNECTED. Orderly release may be started from this state only. In all other data transfer states, abortive disconnection has to be used to end the communication. If a X_FX4_SP_GIVE_TOKEN_I was received, the automaton enters the state 10. Before the connection can be orderly released, the token must be returned by

X_FX4_GIVE_TOKEN_Q to the initiator side. If an X_FX4_SP_RSYN_I was received, the automaton will stay in this state.

state 21 This state is entered when an X_FX4_SP_SOD_I was received. The service user may now receive Date with the function *x_rcvdata()*. If an X_FX4_SP_RSYN_I was received, the automaton enters the state 20.

state 22 This state is entered when an X_FX4_SP_EOP_I was received, which indicates the end of the current page (checkpoint). The service user has now to confirm this page by sending an X_FX4_SP_EOP_P.

state 23 This state is entered when an X_FX4_SP_EOD_I was received, which indicates the successfully transmission of the whole document. The service user has now to confirm the end of the document with X_FX4_SP_EOD_P.

The user may ask for the transmission token (please token request), by setting the parameter X_FX4_P_PLEASE_TOKEN equal PV_TRUE in the service primitives *x_conrsp()*, X_FX4_SP_EOP_P or X_FX4_SP_EOD_P. The active side may then hand over the transmission token by issuing the service primitive X_FX4_SP_GIVE_TOKEN_Q.

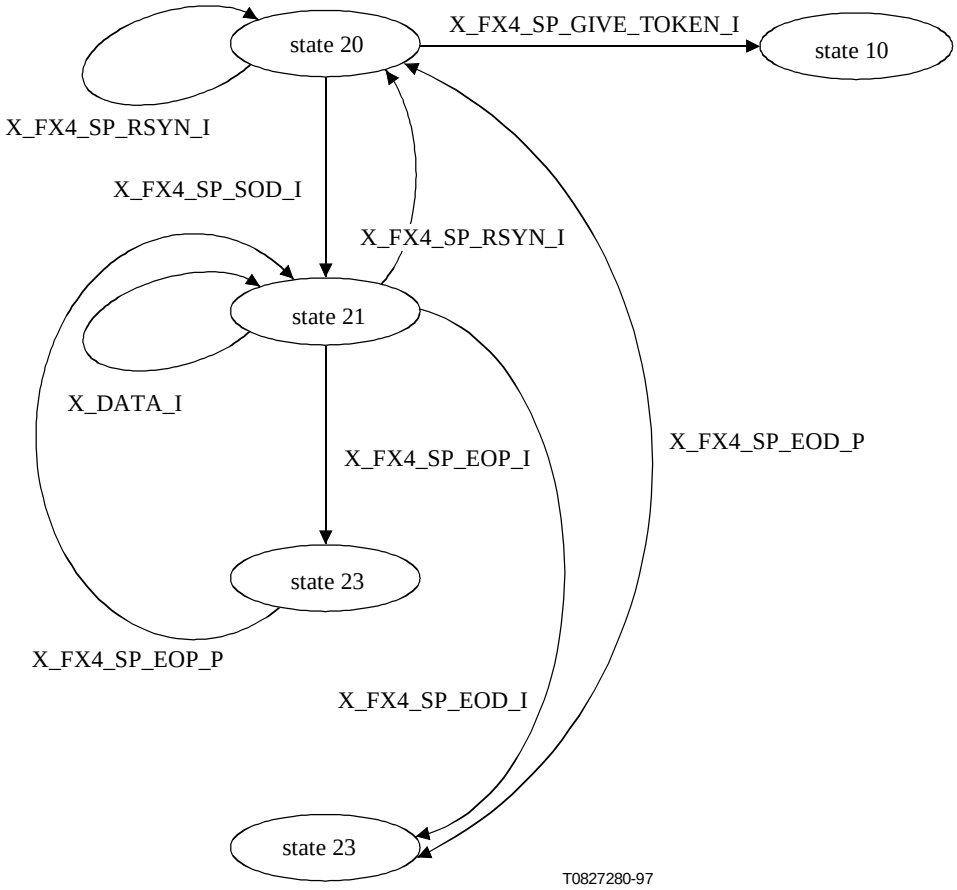


Figure I.3-3/T.180 – Data transfer states in passive mode

I.3.7.4 Connection Release service

I.3.7.4.1 Service description

The orderly release service allows FAX4/BFT service users to release the connection in an orderly manner. The release may only be requested by the side who has initiated the connection and who

owns the Data Token. The orderly release can not be requested during the document transfer phase (between X_FX4_SP_SOD_Q and X_FX4_SP_EOD_Q).

The service elements and their corresponding XAPI functions needed for Connection Release are described in Table I.3-40.

Table I.3-40/T.180 – Service elements and their corresponding XAPI functions for Connection Release

Service element	XAPI function	Description
Release Request	x_relreq()	The Release Request is passed to the provider to request a normal FAX4/BFT connection release.
Release Indication	x_relind()	The Release Indication is generated by the provider to indicate the orderly release of a FAX4/BFT connection by the remote side.
Release Response	x_relrsp()	The Release Response is passed to the provider as reaction to a previously received release indication as positive response.
Release Confirmation	x_relconf()	The Release Confirmation is generated by the provider as positive confirmation of a normal FAX4/BFT release.
End Indication	x_rcvend()	The End Indication is generated by the provider to indicate that the service provider is ready to establish a new connection. In addition it contains some information about the released connection.

I.3.7.4.2 Service parameters

Table I.3-41 specifies the parameters of the Connection Release service.

Table I.3-41/T.180 – Parameters of the Connection Release service

Parameter	Connection Release service				
	Request	Indication	Response	Confirmation	End Indication
X_P_CONN_TIME					M
X_P_DISC_TIME					M
X_P_CHARGE					C
X_P_DISC_REASON					C

I.3.7.4.3 Service parameter descriptions

Tables I.3-42 to I.3-45 describe the parameters for the Connection Release service.

Table I.3-42/T.180

Parameter name	X_P_CONN_TIME
Type of value	unsigned long
Legal values	the time in seconds since 01.01.1970 00:00:00 GMT
Default value	none
Description	This parameter contains the time of physical connection.

Table I.3-43/T.180

Parameter name	X_P_DISC_TIME
Type of value	unsigned long
Legal values	the time in seconds since 01.01.1970 00:00:00 GMT
Default value	none
Description	This parameter contains the time of physical disconnection. If X_P_CONN_TIME and X_P_DISCON_TIME both set to zero, no physical connection could be established.

Table I.3-44/T.180

Parameter name	X_P_CHARGE
Type of value	unsigned long
Legal values	any number
Default value	zero
Description	This parameter contains, if available, the charging unit of the connection. It is only set if both network and network connection provide this facility.

Table I.3-45/T.180

Parameter name	X_P_DISC_REASON
Type of value	unsigned long
Legal values	The values are directly mapped from the underlying hardware. See the hardware documentation to get further information.
Default value	zero
Description	This parameter contains, if available, the disconnection reason.

I.3.7.5 Connection abort service

I.3.7.5.1 Service description

The abort service provides the means by which either FAX4/BFT service user or provider can instantaneously release the connection and have the other FAX4/BFT service user informed of the release. Use of this service will cause loss of undelivered data.

The service elements and their corresponding XAPI functions needed for abort of FAX4/BFT connection are described in Table I.3-46.

Table I.3-46/T.180 – Service elements and their corresponding XAPI functions for Connection Abort

Service element	XAPI function	Description
Abort Request	x_snddis()	The Abort Request is passed to the provider to request an abnormal FAX4/BFT connection release.
Abort Indication	x_rcvdis()	The Abort Indication is generated by the provider to indicate the abnormal release of a FAX4/BFT connection.
End Indication	x_rcvend()	The End Indication is generated by the provider to indicate that the service provider is ready to establish a new connection. In addition it contains some information about the released connection.

I.3.7.5.2 Service parameters

Table I.3-47 specifies the parameters needed for an abortive FAX4/BFT release.

Table I.3-47/T.180 – Parameters of the Connection Abort service

Parameter	Abort Service		
	Request	Indication	End Indication
X_FX4_P_REASON	M	M	
X_P_CONN_TIME			M
X_P_DISC_TIME			M
X_P_CHARGE			C
X_P_DISC_REASON			C

I.3.7.5.3 Service parameter descriptions

Tables I.3-48 to I.3-52 describe the parameters for the abort service.

Table I.3-48/T.180

Parameter name	X_FX4_P_REASON
Type of value	long
Legal values	Legal values for Request and Indication: X_FX4_PV_R_NO_REASON /* no error reason given */ X_FX4_PV_R_LOCERR /* local terminal error */ X_FX4_PV_R_PROCERR /* procedure error */ X_FX4_PV_R_NOMEM /* no memory available */ X_FX4_PV_R_TRANSERR /* transmission error */ X_FX4_PV_R_PNOERR /* wrong page number */ X_FX4_PV_R_FMTERR /* format error */ Legal values only for the Indication: X_FX4_PV_R_NO_CTX /* no more contexts of this type */ X_FX4_PV_R_LOCPROCERR /* local terminal and procedure error */ X_FX4_PV_R_SSNAC /* session can not be accepted */ X_FX4_PV_R_ERRMSG /* error message exists (text) */ X_FX4_PV_R_RJCAP /* capabilities not supported */ X_FX4_PV_R_SVOK /* return code positive */ X_FX4_PV_R_SVNAV /* service not available */
Default value	X_FX4_PV_R_NO_REASON
Description	This parameter indicates the abort reason.

Table I.3-49/T.180

Parameter name	X_P_CONN_TIME
Type of value	unsigned long
Legal values	the time in seconds since 01.01.1970 00:00:00 GMT
Default value	none
Description	This parameter contains the time of physical connection.

Table I.3-50/T.180

Parameter name	X_P_DISC_TIME
Type of value	unsigned long
Legal values	the time in seconds since 01.01.1970 00:00:00 GMT
Default value	none
Description	This parameter contains the time of physical disconnection. If X_P_CONN_TIME and X_P_DISCON_TIME both set to zero, no physical connection could be established.

Table I.3-51/T.180

Parameter name	X_P_CHARGE
Type of value	unsigned long
Legal values	any number
Default value	zero
Description	This parameter contains, if available, the charging unit of the connection. It is only set if both network and network connection provide this facility.

Table I.3-52/T.180

Parameter name	X_P_DISC_REASON
Type of value	unsigned long
Legal values	The values are directly mapped from the underlying hardware. See the hardware documentation to get further information.
Default value	zero
Description	This parameter contains, if available, the disconnection reason.

I.3.7.6 Usage of XAPI functions

This subclause provides some protocol-specific remarks to XAPI functions. The functions are mentioned in alphabetical order. If a function is not listed, there are no special remarks.

- x_conconf The user_data buffer in the *call_struct* is empty, as transfer of user data is not available in the connection establishment phase.
- x_conind The user_data buffer in the *conind_struct* is empty, as transfer of user data is not available in the connection establishment phase.
- x_conreq The user_data buffer in the *call_struct* is empty, as transfer of user data is not available in the connection establishment phase. Usage of the MORE flag is not supported.
- x_conrsp The user_data buffer in the *call_struct* is empty, as transfer of user data is not available in the connection establishment phase. Usage of the MORE flag is not supported.
- x_rcvdata Expedited data are not available.
- x_rcvdis The user_data buffer in the *discon_struct* is empty, as transfer of user data is not available in the disconnection.
- x_relconf The user_data buffer in the *release_struct* is empty, as transfer of user data is not available in the connection release phase.
- x_relind The user_data buffer in the *release_struct* is empty, as transfer of user data is not available in the connection release phase.
- x_relreq The user_data buffer in the *release_struct* is empty, as transfer of user data is not available in the connection release phase. Usage of the MORE flag is not supported.
- x_relrsp The user_data buffer in the *release_struct* is empty, as transfer of user data is not available in the connection release phase. Usage of the MORE flag is not supported.

- x_snddata This function can only be called, if FAX4/BFT service provider is in state 12 or 13. Expedited data is not available at the XAPI FAX4/BFT service access point. Usage of the MORE flag is not supported.
- x_snddis The user_data buffer in the *discon_struct* may not be used, as transfer of user data is not available in the disconnection. Usage of the MORE flag is not supported.
- x_sndsp Usage of the MORE flag is not supported.

I.3.7.7 Tables of error codes

The XAPI error-level error codes are defined in Annex B.

I.3.7.7.1 CC_BADVALUE

An invalid mandatory parameter is given: *diagnostic* contains the corresponding parameter identifier.

I.3.7.7.2 CC_MANDMISS

A mandatory parameter is missing: *diagnostic* contains the number of missing mandatory parameters.

I.3.7.7.3 CC_BADEVENT

An invalid event is specified: *diagnostic* contains the bad event identifier.

I.3.7.7.4 CC_SEQ

An incorrect sequence number is given: *diagnostic* contains the bad sequence number.

I.3.7.7.5 CC_SPNAME

An invalid service primitive name is given: *diagnostic* contains the bad service primitive name.

I.3.7.7.6 CC_ADDCOMP

An additional parameter (neither address parameter nor service primitive parameter) is incorrect: no *diagnostic* is given.

I.3.7.7.7 CC_BADLENGTH

An address or parameter buffer contains an illegal length value: *diagnostic* contains the length.

I.3.7.7.8 CC_UNEXPECT

If the cause code indicates an unexpected event, the value of *diagnostic* will contain the actual state identifier in which the unexpected event caused the error indication. Table I.3-53 contains the values defined for the *diagnostic* parameter.

Table I.3-53/T.180 – Data transfer states

Name	Description
X_FX4_STATE_10	A service is requested, which is not expected in this state. Only the following services are expected: • X_FX4_SP_GIVE_TOKEN_Q • X_FX4_SP_SOD_Q • x_relreq() • x_snddis()
X_FX4_STATE_11	A service is requested, which is not expected in this state. Only the following service is expected: • x_snddis()
X_FX4_STATE_12	A service is requested, which is not expected in this state. Only the following services are expected: • x_snddata() • x_snddis()
X_FX4_STATE_13	A service is requested, which is not expected in this state. Only the following services are expected: • x_snddata() • X_FX4_SP_EOP_Q • X_FX4_SP_EOD_Q • x_snddis()
X_FX4_STATE_14	A service is requested, which is not expected in this state. Only the following service is expected: • x_snddis()
X_FX4_STATE_15	A service is requested, which is not expected in this state. Only the following service is expected: • x_snddis()
X_FX4_STATE_20	A service is requested, which is not expected in this state. Only the following service is expected: • x_snddis()
X_FX4_STATE_21	A service is requested, which is not expected in this state. Only the following service is expected: • x_snddis()
X_FX4_STATE_22	A service is requested, which is not expected in this state. Only the following services are expected: • X_FX4_SP_EOP_P • x_snddis()
X_FX4_STATE_23	A service is requested, which is not expected in this state. Only the following services are expected: • X_FX4_SP_EOD_P • x_snddis()

I.3.7.7.9 CC_NOTSUPPORT

An event is given, which is not supported; the value of *diagnostic* contains the identifier of the unsupported event which has been submitted with the XAPI function call that caused the error indication.

I.3.7.7.10 CC_OTHER

If the cause code indicates the CC_OTHER error code, the value of *diagnostic* contains the identifier which caused the error indication.

I.3.7.8 Table of service primitives

Table I.3-54/T.180

Name	Description
X_FX4_SP_GIVE_TOKEN_Q	Give token request
X_FX4_SP_GIVE_TOKEN_I	Give token indication
X_FX4_SP_SOD_Q	Start of document request
X_FX4_SP_SOD_I	Start of document indication
X_FX4_SP_SOD_C	Start of document confirmation
X_FX4_SP_EOP_Q	End of page request
X_FX4_SP_EOP_I	End of page indication
X_FX4_SP_EOP_P	End of page response
X_FX4_SP_EOP_C	End of page confirmation
X_FX4_SP_EOD_Q	End of document request
X_FX4_SP_EOD_I	End of document indication
X_FX4_SP_EOD_P	End of document response
X_FX4_SP_EOD_C	End of document confirmation

I.3.7.9 Table of service primitive parameters

Table I.3-55/T.180

Name	Legal values
X_FX4_P_ACCEPT	PV_TRUE PV_FALSE
X_FX4_P_BFT_COMPR	char [X_FX4_C_MAX_NAME]
X_FX4_P_BFT_FNAME	char [X_FX4_C_MAX_NAME]
X_FX4_P_BFT_PATH	char [X_FX4_C_MAX_PATH]
X_FX4_P_BLOCK_NR	any number
X_FX4_P_DATE_TIME	char [X_FX4_C_MAX_DATE]
X_FX4_P_DOCREF	char [X_FX4_C_MAX_DOC_REF]
X_FX4_P_DOC_TYPE	X_FX4_PV_BFT X_FX4_PV_FAX
X_FX4_P_OWN_TERMID	char [X_FX4_C_MAX_TERMID]

Table I.3-55/T.180 *(concluded)*

Name	Legal values
X_FX4_P_FAR_TERMID	char [X_FX4_C_MAX_TERMID]
X_FX4_P_FAX_COMPRESSION	PV_TRUE PV_FALSE
X_FX4_P_FAX_DENSITY	X_FX4_PV_DENS_200. X_FX4_PV_DENS_240. X_FX4_PV_DENS_300. X_FX4_PV_DENS_400.
X_FX4_P_FAX_DIMENSION	X_FX4_PV_DIM_A4 X_FX4_PV_DIM_B4 X_FX4_PV_DIM_A3 X_FX4_PV_DIM_AM_LETTER X_FX4_PV_DIM_AM_LEGAL X_FX4_PV_DIM_AM_LEDGER X_FX4_PV_DIM_JAP_LETTER X_FX4_PV_DIM_JAP_LEGAL
X_FX4_P_PAGE_NR	number greater than 0
X_FX4_P_PLEASE_TOKEN	PV_TRUE PV_FALSE
X_FX4_P_REASON	X_FX4_PV_R_NO_REASON X_FX4_PV_R_NO_CTX X_FX4_PV_R_LOCERR X_FX4_PV_R_PROCERR X_FX4_PV_R_LOCPROCERR X_FX4_PV_R_SSNAC X_FX4_PV_R_ERRMSG X_FX4_PV_R_NOMEM X_FX4_PV_R_TRANSERR X_FX4_PV_R_PNOERR X_FX4_PV_R_FMTERR X_FX4_PV_R_RJCAP X_FX4_PV_R_SVOK X_FX4_PV_R_SVNAV
X_FX4_P_REC_JEOP	PV_TRUE PV_FALSE
X_P_CONN_TIME	the time in seconds since 01.01.1970 00:00:00 GMT
X_P_DISC_TIME	the time in seconds since 01.01.1970 00:00:00 GMT
X_P_CHARGE	any number
X_P_DISC_REASON	The values are directly mapped from the underlying hardware. See the hardware documentation to get further information.

I.4 XAPI access to the service provider for ACSE and ROSE

This part of Appendix I describes an example of how the service provider can be implemented, if an application needs the access to the specified service.

I.4.1 Scope

The XAPI, which is an abbreviation for eXtensive Application Programming Interface, is an operating system and language-independent programming interface to general communication services. Detailed information about the XAPI is given in the main part of this Recommendation, which is very important for understanding this appendix.

Which services are made available via the XAPI depends on the installed service providers, and not on the XAPI which only provides the access mechanism.

This part describes the XAPI access to the ACSE/ROSE service provider.

In this provider, the connection establishment, release and abort service are performed via the ACSE accessible at XAPI, whereas in the connected state the ROSE service is available via XAPI.

Figure I.4-1 shows the structure of the protocol stack that is accessible via the XAPI.

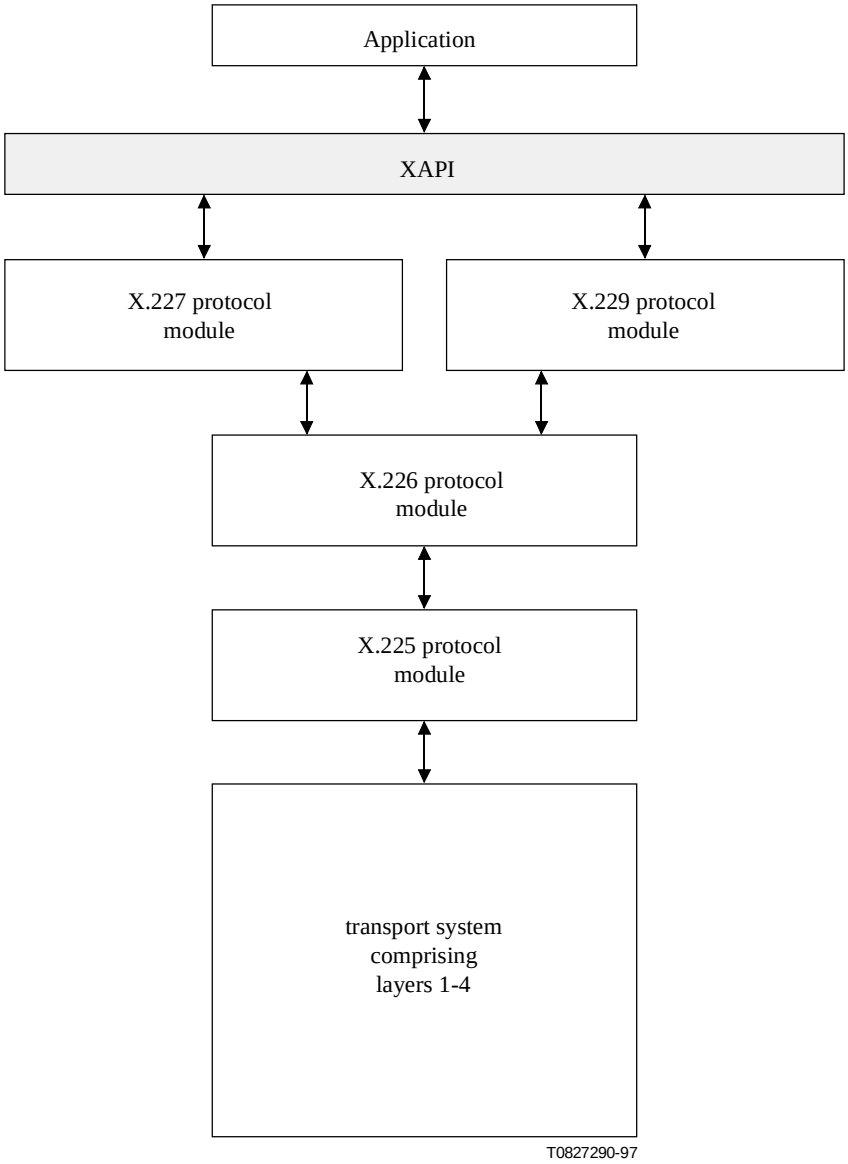


Figure I.4-1/T.180 – Structure of the ACSE/ROSE service provider

The XAPI user is able to select one transport system (comprising the layers 1 to 4) among the set of transport systems available in the XAPI communication platform to act as the underlying transport service provider.

The interface is specified following the pattern of the standardized ROSE service interface [X.219] and the standardized ACSE service interface [X.217].

I.4.2 References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; all users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published.

[I.430] ITU-T Recommendation I.430 (1995), *Basic user-network interface – Layer 1 specification.*

[I.431] ITU-T Recommendation I.431 (1993), *Primary rate user-network interface – Layer 1 specification.*

[T.90] CCITT Recommendation T.90 (1992), *Characteristics and protocols for terminals for telematic services in ISDN.*

[X.75] ITU-T Recommendation X.75 (1996), *Packet-switched signalling system between public networks providing data transmission services.*

[X.200] ITU-T Recommendation X.200 | ISO/IEC 7498-1:1994, *Information technology – Open Systems Interconnection – Basic reference model: The basic model.*

[X.208] CCITT Recommendation X.208 (1988), *Specification of Abstract Syntax Notation One (ASN 1).*

[X.214] ITU-T Recommendation X.214 (1995) | ISO/IEC 8072:1996, *Information technology – Open Systems Interconnection – Transport service definition.*

[X.217] ITU-T Recommendation X.217 (1995) | ISO/IEC 8649:1996, *Information technology – Open Systems Interconnection – Service definition for the association control service element.*

[X.219] CCITT Recommendation X.219 (1988) | ISO/IEC 9072-1:1989, *Remote operations: Model, notation and service definition.*

[X.224] ITU-T Recommendation X.224 (1995) | ISO/IEC 8073:1997, *Information technology – Open Systems Interconnection – Protocol for providing the connection-mode transport service.*

[X.225] ITU-T Recommendation X.225 (1995) | ISO/IEC 8327-1:1996, *Information technology – Open Systems Interconnection – Connection-oriented session protocol: Protocol specification.*

[X.226) ITU-T Recommendation X.226 (1994) | ISO/IEC 8823-1:1994, *Information technology – Open Systems Interconnection – Connection-oriented presentation protocol: Protocol specification.*

[X.227] ITU-T Recommendation X.227 (1995) | ISO/IEC 8650-1:1996, *Information technology – Open Systems Interconnection – Connection-oriented protocol for the association control service element: Protocol specification.*

[X.229] CCITT Recommendation X.229 (1988) | ISO/IEC 9072-2:1989, *Remote operations: Protocol specification.*

[ISO 8208] ISO/IEC 8208:1995, *Information technology – Data communiations – X.25 Packet Layer Protocol for Data Terminal Equipment.*

I.4.3 Definitions

I.4.4 Abbreviations

This part uses the following abbreviations:

ACS abbreviation string specifying service specific definitions of the ACSE provider (e.g. parameter names, service primitive names)

ACSE Association Control Service Element

APDU Application Protocol Data Unit

ASE Application Service Element

DSS 1 Digital Subscriber Signalling System No. 1

HDLC High-Level Data-Link Control

ISDN Integrated Services Digital Network

ISO International Organization for Standardization

NSAP Network Service Access Point

OSI Open Systems Interconnection

PSAP Presentation Service Access Point

PSEL Presentation Selector

ROS abbreviation string specifying service specific definitions of the ROSE provider (e.g. parameter names, service primitive names)

ROSE Remote Operations Service Element

RTSE Reliable Transfer Service Element

SSAP ACSE/ROSE Service Access Point

SSEL Session Service Selector

TSAP Transport Service Access Point

TSEL Transport Service Selector

XAPI eXtensive Application Programming Interface

I.4.5 Conventions

Each service is described via three kinds of tables.

In the first kind of table the service and its service elements are described. It contains a row for each service element of the service (where service element means either request, indication, response, or confirmation) with its corresponding XAPI function and a short description.

In the second kind of table the use of the service primitive parameters within the service elements are described. It contains a row for each service primitive parameter of the service. The service primitive parameter name is stored within the first column, the following columns contain the use of the parameter in the service elements:

Blank The service parameter is absent.

C Presence of the service parameter is conditional. Firstly, there may be a condition in the service provider to provide a parameter in an indication or confirmation; secondly, there may be interdependencies between parameters of the same or the preceding service primitive.

M Presence of the service parameter is mandatory.

U Presence of the service parameter is a user option. If the user does not specify a value for such a parameter and there is no default value for that parameter, nothing is passed to the service provider. If the user does not specify a value for such a parameter and there is a default value, the default value is passed to the service provider.

(=) The value of the service parameter is identical to the value of the corresponding service parameter in the preceding service element. In the special case of a parameter, whose presence in the preceding service is a user option, for which a default value is defined, and the parameter was not specified in the preceding service element, the symbol (=) indicates that the parameter value is identical to the default value.

The third kind of table is used to describe the service primitive parameter containing the type of value and its scope, possible default values and a detailed description. For each service primitive parameter, one description table is given.

A **naming convention** is used throughout the XAPI in order to distinguish between the common functions, identifiers and values, and service specific identifiers and values. All service-specific settings of the XAPI ACSE/ROSE service access are defined within the present part of this appendix and start with **X_ROS_**, **x_ros_**, **X_ACS**, or **x_acs**.

Extended naming conventions:

– SP_ Service Primitive;

– P_ Parameter;

– PV_ Parameter Value;

– O_ Option;

– OV_ Option Value.

I.4.6 Introduction to the ACSE/ROSE provider access

The ACSE/ROSE service is provided by the combination of a transport system and the Session, Presentation, ACSE and ROSE protocol module, selected by the application. The ROSE protocol module is implemented according to the requirements of [X.229] and the service which is provided follows the pattern of [X.219]. The ACSE protocol module is implemented according to the requirements of [X.227] and the service which is provided follows the pattern of [X.217].

It provides basic facilities for the control of an application-association between two application-entities which communicate by means of a presentation-connection, e.g.

- establish an association with another ACSE user;
- normal release of an association with another ACSE user; and
- abnormal release of an association with another ACSE user.

It provides the means of interactive applications between cooperating ACSE/ROSE users. An application can:

- invoke operations by one ACSE/ROSE user and perform operations by another ACSE/ROSE user;

- support synchronous and asynchronous operations;
- group operations into a set of linked operations;
- perform the control of the application association by ACSE.

The referenced Remote Operation notation (RO-notation) allows an Application Service Element (ASE) to be defined for the support of a particular application or group of applications. This ASE consists of a BIND operation (establishes an association), one or more operations to implement an interactive protocol, and an UNBIND operation (releases the association). The BIND and UNBIND operations are implemented using the services provided by ACSE while the other operations are implemented using the ROSE services. Therefore, the service provider gives access to the services of the ROSE plus those of the ACSE service.

I.4.6.1 Restrictions to Recommendations X.227 and X.229

The ACSE service supports two modes of operation, i.e. the normal mode and the X.410-1984 mode. At the XAPI, only the normal mode is accessible, which allows the ACSE user to take full advantage of the functionality provided by both ACSE and the presentation service.

Functional units are used for the purpose of identification of Presentation service user requirements and are negotiated during connection establishment.

Two categories of functional units are supported by the service provider:

a) Session functional units, comprising:
   - Kernel;
   - Duplex;
   - Half-duplex;
   - Minor synchronize;
   - Exception;
   - Capability data;
   - Activity management.

b) Presentation functional units, comprising:
   - Kernel.

The Presentation Context Definition List is also negotiated during connection establishment. To each abstract syntax a transfer syntax is assigned, which is used for encoding and decoding the user data during the lifetime of the connection. The possibility to negotiate a default context, as it is provided by the ACSE Recommendation, is not supposed. Therefore, the negotiation of the Presentation Context Definition List is mandatory during connection establishment. The negotiation is described in the next subclause.

I.4.6.2 Service-specific features

Due to the goal of a modular and object-oriented design of the protocol modules, special attention to the encoding/decoding mechanism of the user data is required.

The idea is, that each protocol itself encodes/decodes its PDUs to make the Presentation layer independent of the layers above with their abstract syntax and transfer syntaxes. As a consequence (and extending the standardized service interface), the application does not only provide the context identifier and abstract syntax for each presentation context (i.e. each protocol module above presentation) but also the (list of) transfer syntaxes supported by each protocol module.

This has the following consequences for the ACSE-user:

– *Negotiation of the presentation context definition list*

During connection establishment, a presentation context definition list is negotiated by the ACSE-user communication partners. It consists of a list of triples, one for each abstract syntax, which is proposed by the association initiator and accepted (or not accepted) by the association responder. Each triple consists of an odd integer identifying the triple, the abstract syntax and the (list of) supported transfer syntaxes. The odd integer identifying the triple is called presentation context identifier. After negotiation, only the presentation context identifier is used to identify an abstract syntax and the corresponding transfer syntax.

NOTE – It must be ensured that the transfer syntaxes proposed in the presentation context definition list match the supported transfer syntaxes of the protocol modules of the service provider.

– *Encoding and decoding of the user data*

ACSE user data are treated as transparent data by the ACSE protocol module. If there is any special format to be used for the ACSE user data, it is up to the application to generate and interpret that format. The application is required to encode/decode the ACSE user data passed/received to/from XAPI in the *user_data* parameter of *call_struct*, *release_struct* or *discon_struct*. The data in the user data must be encoded as a data stream comprising a list of one or more non-concatenated ASN.1 EXTERNAL types. The following rules apply to the components of the EXTERNAL type:

- If presentation context negotiation has been completed, the presentation context identifier also identifies the encoding rules (transfer syntax) for the data value and the "direct-reference OBJECT IDENTIFIER" shall not be included;
- If presentation context negotiation is not complete, an object identifier value is also needed which identifies the encoding rules (transfer syntax) used for the encoding.

ROSE defines five different Operation Classes which classify operations according to two possible operation modes (synchronous and asynchronous). The Association Class defines which ROSE user is allowed to invoke operations. The Operation Class and Association Class have to be agreed between ROSE users. It is not part of the ACSE/ROSE provider to negotiate these features. OSI Applications making use of ROSE (e.g. DTAM, DFR, MHS) define which Operation Class and Association Class are allowed within a specific application.

Due to the fact that there is no negotiation of Operation and Association Class within the ACSE/ROSE provider, the parameters Invoke-Id (to identify an operation and to correlate the request of an invocation with its replies) and Linked-Id (in the case of a child operation to identify the parent operation) can not be examined by ACSE/ROSE. In particular, it is not controlled if an invocation exists for an incoming reply, i.e. result, error or rejection, or if the specified parent operation exists for the invocation of a child operation.

ACSE/ROSE does not define a distinct abstract syntax for the encoding of its PDUs. Instead, it provides a set of abstract syntax definitions that are used by the application making use of ACSE/ROSE. Therefore, the service user must inform the service provider which abstract syntaxes are used within the ROSE PDUs. To support this feature, each ROSE service primitive must be provided with a service primitive parameter (X_ROS_P_CTXT_ID) that indicates the abstract syntax of the application.

BIND and UNBIND encoding

At the XAPI, the ACSE/ROSE BIND/UNBIND operations are implemented using the primitives of the ACSE services A_ASSOCIATE and A_RELEASE.

Application data encoding

The *argument*, *result*, or *parameter* components of ROSE APDUs are treated as transparent data by the ROSE protocol module. It is under the responsibility of the XAPI user to encode and decode the application data.

I.4.7 Description of the access to the ACSE/ROSE provider

I.4.7.1 Service initialization

I.4.7.1.1 Creation of a service access point with x_open()

A communication endpoint accessing the ACSE and ROSE protocol modules is created when calling the *x_open()* function with an appropriate service provider identification string. A communication endpoint accessing the ROSE protocol module is always created together. The available identifiers depend on the actual system configuration. In the standard configuration **"X_ACSE_ROSE_ISDN"** identifies the ISO ACSE/ROSE service provider with access to the ROSE and ACSE services. In this case the application system comprises the ISO Session, Presentation, ACSE and ROSE protocol modules; the transport system comprises ISO Transport Class 0, ISO 8208, HDLC LAP B and I.430/I.431.

I.4.7.1.2 Activation of a service access point with x_bind()

x_bind() is to be called to activate the service endpoint. The function has the following tasks:

- if the service provider, which has been selected with *x_open()*, does not comprise a transport system, link a transport system below the available protocol modules;
- bind an address to the service endpoint.

In the standard configuration, if **"X_ACSE_ROSE_ISDN"** was selected in the *x_open()* function, no transport system has to be specified as argument of the *x_bind()* function as the service provider's protocol stack is already complete.

I.4.7.1.3 Protocol addresses

The address to be used to identify the peer entity is the tuple of

(PSEL, SSEL, TSEL, NSAP Address),

where the selectors are optional depending on the peer's requirements.

The XAPI service provider itself does not support selectors as address information on the local side; the own address consists of the NSAP address only.

I.4.7.1.3.1 The application's own address

The own address (only NSAP is supported as explained above) may be specified in the *own_address* buffer of the *bind_struct* passed as argument to the *x_bind()* function. For a passive application, it is returned in the *called_addr* buffer of the *x_conind()* function.

For a passive application it is not supported to specify the own responding NSAP address in the *address* buffer of the *call_struct* in the *x_conrsp()* function, as this value is not transferred by the network.

Note that specification of the application's own protocol address is completely optional. If no address information is specified, the own address is derived from system configuration information and the value actually bound is returned as output parameter of the *x_bind()* function.

The own address consists of the NSAP address only. The NSAP address has to comprise the local ISDN outband address, i.e. the address information used in the D-channel. The own ISDN inband address and subaddress parameters as well as the protocol selectors are not supported. If specified, they will be ignored.

Table I.4-1 shows the address component which has to be specified in the *x_bind()* call.

Table I.4-1/T.180 – Address component specified in the *x_bind()* call

ISDN network	Address component	Value
ISDN/DSS 1	A_OUTBAND_ADR	A decimal digit, which is locally mapped on the Multiple Subscriber Number (MSN)

I.4.7.1.3.2 The address of the communication partner

On the sending side, the address of the communication partner has to be specified in the *address* buffer of the *call_struct* passed as argument to the *x_conreq()* function. On the receiving side, the address of the communication partner is returned in the *calling_addr* buffer of the *x_conind()* function.

The address of the communication partner comprises at least the peer's ISDN outband address. The peer's ISDN inband address and subaddress are meaningless, whereas usage of protocol selectors is according to the peer's requirements.

Table I.4-2 shows the address components to be used in the *address* buffer specifying the called Presentation address in an *x_conreq()* call:

Table I.4-2/T.180 – Address components specifying the called NSAP address in an *x_conreq()* call

ISDN network	Address component	Value
ISDN/DSS 1	A_OUTBAND_ADR	optional the country code, optional the area code and the multiple subscriber number (MSN)
ISDN/DSS 1	A_T_SELECTOR	called Transport selector (optional)
ISDN/DSS 1	A_S_SELECTOR	called Session selector (optional)
ISDN/DSS 1	A_P_SELECTOR	called Presentation selector (optional)

I.4.7.1.4 Configuration of the service provider

The protocol modules of the service provider behave according to configured protocol options. Protocol options are used to control the general behaviour of a protocol module (they must not be confounded with service primitive parameters). The pre-configured values of the protocol options are sufficient for the majority of communication relations.

Currently, there are no options for the protocol modules of the Presentation service provider, which could be set by the XAPI function *x_optmgmt()*.

I.4.7.2 Connection Establishment service

I.4.7.2.1 Service description

During the connection establishment phase, two users of the same service establish a connection between them. The XAPI user must already have prepared a service endpoint before the connection establishment phase can start.

The service elements and their corresponding XAPI functions needed for ACSE Connection Establishment are described in the Table I.4-3.

Table I.4-3/T.180 – Service elements and their corresponding XAPI functions for Connection Establishment

Service element	XAPI function	Description
Connect Request	x_conreq()	The Connect Request is passed to the provider to request the establishment of a connection.
Connect Indication	x_conind()	The Connect Indication is generated by the provider to indicate the request from a remote terminal to establish a connection.
Connect Response	x_conrsp()	The Connect Response is passed to the provider as reaction to a previously received Connect Indication as positive or negative response.
Connect Confirmation	x_conconf()	The Connect Confirmation is generated by the provider of a local establishment.

I.4.7.2.2 Service parameters

Table I.4-4 specifies the parameters of the ACSE Connection Establishment service.

NOTE – Some parameters belong together forming one data structure, so that they have to occur in a special order.

The first is the context definition list, which is formed by a list of triples comprising the presentation context id, the abstract syntax and a (list of) supported transfer syntaxes. Each triple must occur always in the same order within the parameter buffer. The order is X_ACS_P_CTXT_ID, X_ACS_P_AS and (possibly several times) X_ACS_P_TS.

The second is the context definition result list, which is also formed by a list of triples comprising the presentation context identifier, the result, i.e. acceptance or rejection of the proposed presentation context and, if the result is acceptance, the transfer syntax, which should be used. There are as many triples as there are in the presentation context definition list.

The value of diagnosis parameter (X_ACS_P_DIAG) depends on the value of the result source parameter (X_ACS_P_RES_SRC). Thus, it is important that the result source preceeds the diagnosis.

Table I.4-4/T.180 – Parameters of the Connection Establishment service

Parameter	Connect Service			
	Request	Indication	Response	Confirmation
X_ACS_P_MAX_SUDATA		M		C
X_ACS_P_APP_CTXT	M	M (=)	M	M (=)
X_ACS_P_CTXT_ID	M	M (=)		
X_ACS_P_AS	M	M (=)		
X_ACS_P_TS	U	C (=)		
X_ACS_P_CTXT_RES		M	M	M (=)
X_ACS_P_TS_RES		C	C	C (=)
X_ACS_P_RES			M	M (=)
X_ACS_P_RES_SRC				M
X_ACS_P_DIAG			U	C (=)
X_ACS_P_SUR	M	M (=)	M	M (=)
X_ACS_P_CAG_AEQ	U	C (=)		
X_ACS_P_CAG_AEID	U	C (=)		
X_ACS_P_CAG_APT	U	C (=)		
X_ACS_P_CAG_APID	U	C (=)		
X_ACS_P_CAD_AEQ	U	C (=)		
X_ACS_P_CAD_AEID	U	C (=)		
X_ACS_P_CAD_APT	U	C (=)		
X_ACS_P_CAD_APID	U	C (=)		
X_ACS_P_RES_AEQ			U	C (=)
X_ACS_P_RES_AEID			U	C (=)
X_ACS_P_RES_APT			U	C (=)
X_ACS_P_RES_APID			U	C (=)
X_ACS_P_INIT_SYN_POINT	C	C (=)	C	C (=)
X_ACS_P_TOKEN	C	C (=)	C	C (=)
X_ACS_P_S_CASUR	U	C (=)		
X_ACS_P_S_CDSUR			U	C (=)
X_ACS_P_S_CR	U	C (=)	U	C (=)
X_ACS_P_S_ARI	U	C (=)	U	C (=)
X_ACS_P_PROV_REJ			U	

I.4.7.2.3 Service parameter description

Tables I.4-5 to I.4-34 list the parameters for the Connection Establishment service.

Table I.4-5/T.180

Parameter name	X_ACS_P_MAX_SUDATA
Type of value	unsigned long
Legal values	X_ACS_PV_RESTRICTED, i.e. session user data are restricted X_ACS_PV_UNRESTRICTED, i.e. session user data are not restricted
Default value	none
Description	A restriction of the session service user data length causes restrictions for the ACSE user data. The total length of an ASN.1 coded presentation PDU embedding an ASN.1 coded ACSE PDU is restricted to 65 539 bytes.

Table I.4-6/T.180

Parameter name	X_ACS_P_APP_CTXT
Type of value	long[]
Legal values	Values must be given as a sequence of long values which form an object identifier identifying the application context name.
Default value	none
Description	The parameter identifies the application context. In the Response or Confirmation service either the same or a different application context is returned.

Table I.4-7/T.180

Parameter name	X_ACS_P_CTXT_ID
Type of value	long
Legal values	Odd integer values which must be different for each presentation context definition.
Default value	none
Description	The parameter indicates the presentation context identification. The parameter has to be immediately followed by the parameter X_ACS_P_AS and (several) X_ACS_P_TS at the XAPI interface. X_ACS_P_CTXT_ID, X_ACS_P_AS and (several) X_ACS_P_TS are components of the presentation context definition. There may be a list of those triples which constitute the presentation context definition list.

Table I.4-8/T.180

Parameter name	X_ACS_P_AS
Type of value	long[]
Legal values	Values must be given as a sequence of long values which form an object identifier identifying an abstract syntax.
Default value	none
Description	The parameter indicates the abstract syntax name. The parameter must follow the parameter X_ACS_P_CTXT_ID and must be followed by (several) X_ACS_P_TS at the XAPI interface. X_ACS_P_CTXT_ID, X_ACS_P_AS and (several) X_ACS_P_TS are components of the presentation context definition. There may be a list of those triples which constitute the presentation context definition list.

Table I.4-9/T.180

Parameter name	X_ACS_P_TS
Type of value	long[]
Legal values	Values must be given as a sequence of long values which form an object identifier identifying an transfer syntax.
Default value	{2, 1, 1} the object identifier of transfer syntax X.209
Description	The parameter indicates the transfer syntax name. The parameter must follow the parameter X_ACS_P_CTXT_ID and X_ACS_P_AS at the XAPI interface. X_ACS_P_CTXT_ID, X_ACS_P_AS and (several) X_ACS_P_TS are components of the presentation context definition. There may be a list of those triples which constitute the presentation context definition list.

Table I.4-10/T.180

Parameter name	X_ACS_P_CTXT_RES
Type of value	long
Legal values	The value indicates the acceptance or rejection of the presentation context definition. The following legal values are defined: X_ACS_PV_CTXT_ACC Acceptance; X_ACS_PV_CTXT_REJ_U User rejection; X_ACS_PV_CTXT_REJ_P Provider rejection; X_ACS_PV_CTXT_UNDEF Result is not specified yet.
Default value	none
Description	The parameter indicates the acceptance or rejection of each presentation context definition list parameter. The parameter must be followed by X_RTS_P_TS _RES at the XAPI interface. X_ACS_P_CTXT_RES and X_ACS_P_TS_RES are components of the presentation context definition result. There may be a list of those tuples which constitute the presentation context definition result list.

Table I.4-11/T.180

Parameter name	X_ACS_P_TS_RES
Type of value	long[]
Legal values	Values must be given as a sequence of long values which form an object identifier identifying a transfer syntax.
Default value	{2, 1, 1} the object identifier of transfer syntax X.209
Description	The parameter indicates the transfer syntax name, which has been chosen out of the list of proposed transfer syntaxes within the context definition list and which has to be used. The parameter must follow the parameter X_ACS_P_CTXT_RES at the XAPI interface only if the result is acceptance. X_ACS_P_CTXT_RES and X_ACS_P_TS_RES are components of the presentation context definition result. There may be a list of those tuples which constitute the presentation context definition result list.

Table I.4-12/T.180

Parameter name	X_ACS_P_RES	
Type of value	unsigned long	
Legal values	X_ACS_PV_ACCEPT	Association accepted;
	X_ACS_PV_REJ_PERM	Permanent rejection of the association;
	X_ACS_PV_REJ_TRANS	Transient rejection of the association.
Default value	none	
Description	The parameter indicates the result of the connection establishment procedure.	

Table I.4-13/T.180

Parameter name	X_ACS_P_RES_SRC	
Type of value	unsigned long	
Legal values	X_ACS_PV_ACSE_SU	ACSE service user;
	X_ACS_PV_ACSE_SP	ACSE service provider;
	X_ACS_PV_PRES_SP	Presentation service provider.
Default value	none	
Description	The parameter indicates the resulting source of the X_ACS_P_RES parameter. If the value of the X_ACS_P_RES parameter is X_ACS_PV_ACCEPT, the value of this parameter is X_ACS_PV_ACSE_SU.	

Table I.4-14/T.180

Parameter name	X_ACS_P_DIAG	
Type of value	unsigned long	
Legal values	X_ACS_PV_UNUSED	UNUSED
	X_ACS_PV_NO_ACSE	No common ACSE version;
	X_ACS_PV_NO_REASON	No reason given;
	X_ACS_PV_APP_CTXT_NAME	Application context name not supported;
	X_ACS_PV_CAG_AP_TITLE	Calling AP title not recognized;
	X_ACS_PV_CAG_AE_QUAL	Calling AE qualifier not recognized;
	X_ACS_PV_CAG_AP_ID	Calling AP invocation identifier not recognized;
	X_ACS_PV_CAG_AE_ID	Calling AE invocation identifier not recognized;
	X_ACS_PV_CAD_AP_TITLE	Called AP title not recognized;
	X_ACS_PV_CAD_AE_QUAL	Called AE qualifier not recognized;
	X_ACS_PV_CAD_AP_ID	Called AP invocation identifier not recognized;
	X_ACS_PV_CAD_AE_ID	Called AE invocation identifier not recognized.
Default value	none	
Description	The parameter provides diagnostic information about the result of the connection services. The parameter is only used if the X_ACS_P_RES parameter has the value X_ACS_PV_REJ_PERM or X_ACS_PV_REJ_TRANS. If the X_ACS_P_RES_SRC parameter has the value X_ACS_PV_ACSE_SP, this parameter can take the values X_ACS_PV_UNUSED and X_ACS_PV_NO_ACSE. If the X_ACS_P_RES_SRC parameter has the value X_ACS_PV_ACSE_SU, this parameter can take all values except X_ACS_PV_NO_ACSE.	

Table I.4-15/T.180

Parameter name	X_ACS_P_SUR
Type of value	unsigned long
Legal values	X_ACS_PV_SUR_HALFDUPLEX X_ACS_PV_SUR_DUPLEX X_ACS_PV_SUR_MINSYNC X_ACS_PV_SUR_EXCEPT X_ACS_PV_SUR_CDATA X_ACS_PV_SUR_ACTMGMT
Default value	none
Description	The parameter indicates the session user requirements = functional units. The values form a bitmask and must be given as one or more items in the integer format OR'ed together (e.g. X_ACS_PV_SUR_HALFDUPLEX | X_ACS_PV_SUR_ACTMGMT). Not all combination of values are allowed (see [X.217]).

Table I.4-16/T.180

Parameter name	X_ACS_P_CAG_AEQ
Type of value	char[]
Legal values	buffer with user encoded AE-qualifier
Default value	none
Description	The parameter indicates the calling AE-qualifier. It must be encoded as ANY single ASN.1 type.

Table I.4-17/T.180

Parameter name	X_ACS_P_CAG_AEID
Type of value	long
Legal values	any integer value
Default value	none
Description	This parameter indicates the calling AE invocation identifier.

Table I.4-18/T.180

Parameter name	X_ACS_P_CAG_APT
Type of value	char[]
Legal values	buffer with user encoded AP title
Default value	none
Description	The parameter identifies the calling AP-title. It must be encoded as ANY single ASN.1 type.

Table I.4-19/T.180

Parameter name	X_ACS_P_CAG_APID
Type of value	long
Legal values	any integer value
Default value	none
Description	This parameter indicates the calling AP invocation identifier.

Table I.4-20/T.180

Parameter name	X_ACS_P_CAD_AEQ
Type of value	char[]
Legal values	buffer with user encoded AE-qualifier
Default value	none
Description	The parameter indicates the called AE-qualifier. It must be encoded as ANY single ASN.1 type.

Table I.4-21/T.180

Parameter name	X_ACS_P_CAD_AEID
Type of value	long
Legal values	any integer value
Default value	none
Description	This parameter indicates the called AE invocation identifier.

Table I.4-22/T.180

Parameter name	X_ACS_P_CAD_APT
Type of value	char[]
Legal values	buffer with user encoded AP title
Default value	none
Description	The parameter identifies the called AP-title. It must be encoded as ANY single ASN.1 type.

Table I.4-23/T.180

Parameter name	X_ACS_P_CAD_APID
Type of value	long
Legal values	any integer value
Default value	none
Description	This parameter indicates the called AP invocation identifier.

Table I.4-24/T.180

Parameter name	X_ACS_P_RES_AEQ
Type of value	char[]
Legal values	buffer with user encoded AE-qualifier
Default value	none
Description	The parameter indicates the responding AE-qualifier. It must be encoded as ANY single ASN.1 type.

Table I.4-25/T.180

Parameter name	X_ACS_P_RES_AEID
Type of value	long
Legal values	any integer value
Default value	none
Description	This parameter indicates the responding AE invocation identifier.

Table I.4-26/T.180

Parameter name	X_ACS_P_RES_APT
Type of value	char[]
Legal values	buffer with user encoded AP title
Default value	none
Description	The parameter identifies the responding AP-title. It must be encoded as ANY single ASN.1 type.

Table I.4-27/T.180

Parameter name	X_ACS_P_RES_APID
Type of value	long
Legal values	any integer value
Default value	none
Description	This parameter indicates the responding AP invocation identifier.

Table I.4-28/T.180

Parameter name	X_ACS_P_INIT_SYN_POINT
Type of value	unsigned long
Legal values	integer value in the range 0 to 999999
Default value	none
Description	The parameter identifies the initial synchronization point.

Table I.4-29/T.180

Parameter name	X_ACS_P_TOKEN	
Type of value	unsigned long	
Legal values	X_ACS_PV_DATA_TOK_REQ	Data token assigned to requestor side;
	X_ACS_PV_DATA_TOK_ACP	Data token assigned to acceptor side;
	X_ACS_PV_DATA_TOK_ACP_C	Data token assigned to acceptor choice;
	X_ACS_PV_MIN_TOK_REQ	Synchronize-minor token assigned to requestor side;
	X_ACS_PV_MIN_TOK_ACP	Synchronize-minor token assigned to acceptor side;
	X_ACS_PV_MIN_TOK_ACP_C	Synchronize-minor token assigned to acceptor choice;
	X_ACS_PV_MAJ_TOK_REQ	Major/activity token assigned to requestor side;
	X_ACS_PV_MAJ_TOK_ACP	Major/activity token assigned to acceptor side;
	X_ACS_PV_MAJ_TOK_ACP_C	Major/activity token assigned to acceptor choice.
Default value	none	
Description	This parameter indicates a list of the initial sides to which the available tokens are assigned. The value of the parameter is dependent on X_ACS_P_SUR. Values of the parameter must be given as one or more items OR'ed together (e.g. X_ACS_PV_DATA_TOK_REQ | X_ACS_PV_MAJ_TOK_REQ). Not all combination of values are allowed.	

Table I.4-30/T.180

Parameter name	X_ACS_P_S_CASUR
Type of value	char[]
Legal values	any sequence of characters, not '\0' terminated. The value is within the range of 0 (zero) up to X_C_MAX_REF octets of characters.
Default value	none
Description	The parameter indicates the calling session service user reference, which contains the identification of the calling application. It is part of the session connection id, which is used to obviously identify the association.

Table I.4-31/T.180

Parameter name	X_ACS_P_S_CDSUR
Type of value	char[]
Legal values	any sequence of characters, not '\0' terminated. The value is within the range of 0 (zero) up to X_C_MAX_REF octets of characters.
Default value	none
Description	The parameter indicates the called session service user reference, which contains the identification of the called application. It is part of the session connection id, which is used to obviously identify the association.

Table I.4-32/T.180

Parameter name	X_ACS_P_S_CR
Type of value	char[]
Legal values	any sequence of characters, not '\0' terminated. The value is within the range of 0 (zero) up to X_C_MAX_REF octets of characters.
Default value	none
Description	The parameter indicates the common reference, which contains the date and time reference information showing the year, month, day, hour and minute. This time represents the local time at the calling terminal and is intended to represent the time of call origination. It is part of the session connection id, which is used to obviously identify the association.

Table I.4-33/T.180

Parameter name	X_ACS_P_S_ARI
Type of value	char[]
Legal values	any sequence of characters, not '\0' terminated. The value is within the range of 0 (zero) up to X_C_MAX_ARI octets of characters.
Default value	none
Description	The parameter indicates the additional information, which contains a document reference number. It is part of the session connection id, which is used to obviously identify the association.

Table I.4-34/T.180

Parameter name	X_ACS_P_PROV_REJ
Type of value	unsigned long
Legal values	PV_TRUE PV_FALSE
Default value	PV_FALSE
Description	The parameter has to be set to PV_TRUE when detecting a syntactical error in the encoded user data or a semantic error in the proposed context definition list. In this case the Presentation will perform a Presentation provider reject.

I.4.7.3 Services in the connected state

The ROSE protocol module provides several services in the connected state which are accessible via the XAPI:

- the RO-INVOKE service (request of a remote operation to be performed);
- the RO-RESULT service (return of the positive reply of a successfully performed operation);
- the RO-ERROR service (return of the negative reply of a unsuccessfully performed operation);
- the RO-REJECT-U service (rejection of the request or reply); and
- the RO-REJECT-P service (information about problem).

While the service endpoint used to access the provider is in state X_CONNECTED the corresponding service primitives can be passed to the provider or retrieved from the provider with calls of *x_sndsp() or x_rcvsp(),* respectively.

The ROSE services specify operations which exceed the transfer services of other protocol modules (e.g. pure transfer services in the RTSE, Session or Transport service access points). Transfer services in XAPI sense enable XAPI users to send and receive either normal or expedited data without additional parameters. ROSE operations transfer additional information beside the application data. Due to this reason the functions *x_sndsp()* and *x_rcvsp()* are used to send and receive, respectively, ROSE operations.

I.4.7.3.1 RO-INVOKE service

I.4.7.3.1.1 Service description

The RO-INVOKE service is used to request the start of a remote operation.

The service elements and their corresponding XAPI functions needed for RO-INVOKE are described in Table I.4-35.

Table I.4-35/T.180 – Service elements and their corresponding XAPI functions for the invoke service

Service Element	Service element identifier	XAPI function	Description
RO-INVOKE Request	X_ROS_SP_ROIV_Q	x_sndsp()	The RO-INVOKE Request is passed to the provider to send a request to start a remote operation.
RO-INVOKE Indication	X_ROS_SP_ROIV_I	x_rcvsp()	The RO-INVOKE Indication is passed to the provider to receive a request for the start of a remote operation.

I.4.7.3.1.2 Service parameters

Table I.4-36 specifies the parameters of the ROSE Invoke service.

Table I.4-36/T.180 – Parameters of the Invoke service

Parameter	RO-INVOKE service	
	Request	Indication
X_ROS_P_CTXT_ID	M	M
X_ROS_P_INV_ID	M	M
X_ROS_P_LINK_ID	U	C (=)
X_ROS_P_VAL_INT	C (Note 1)	C (=) (Note 1)
X_ROS_P_VAL_ID	C (Note 1)	C (=) (Note 1)
NOTE 1 – Either of the parameters X_ROS_P_VAL_INT or X_ROS_P_VAL_ID is present. NOTE 2 – The operation argument is contained in the *data* buffer of *sp_struct* of the *x_sndsp()* rsp. *x_rcvsp()* function. NOTE 3 – The parameter X_ROS_P_CTXT_ID is the presentation context identification which was negotiated in the ACSE connection establishment for the application.		

I.4.7.3.1.3 Service parameter description

Tables I.4-37 to I.4-41 specify the parameters for the ROSE Invoke services.

Table I.4-37/T.180

Parameter name	X_ROS_P_CTXT_ID
Type of value	long
Legal values	odd integer value
Default value	none
Description	The parameter indicates the presentation context identification of the application data of the RO-INVOKE service.

Table I.4-38/T.180

Parameter name	X_ROS_P_INV_ID
Type of value	long
Legal values	any integer value
Default value	none
Description	The parameter identifies the request of a RO-INVOKE service and is used to correlate this request with the corresponding replies or the invocation of linked operations.

Table I.4-39/T.180

Parameter name	X_ROS_P_LINK_ID
Type of value	long
Legal values	any integer value
Default value	none
Description	The parameter identifies a child-operation and the parameter identifies the invocation of the linked parent-operation. The value is that of the X_ROS_P_INV_ID parameter of the RO-INVOKE indication primitive of the parent-operation.

Table I.4-40/T.180

Parameter name	X_ROS_P_VAL_INT
Type of value	long
Legal values	any integer value
Default value	none
Description	The parameter indicates the identifier of the operation to be invoked. As the identifier of the operation may either be an integer value or an object identifier, either this parameter (X_ROS_P_VAL_INT) or the parameter (X_ROS_P_VAL_ID) may be present.

Table I.4-41/T.180

Parameter name	X_ROS_P_VAL_ID
Type of value	long[]
Legal values	Values must be given as a sequence of values in the decimal integer format that are separated by blanks and enclosed in braces (NumberForm of ASN.1 Object Identifier).
Default value	none
Description	The parameter indicates the identifier of the operation to be invoked. As the identifier of the operation may either be an integer value or an object identifier, either this parameter (X_ROS_P_VAL_ID) or the parameter (X_ROS_P_VAL_INT) may be present.

I.4.7.3.2 RO-RESULT service

I.4.7.3.2.1 Service description

The RO-RESULT service is used to reply to a previous RO-INVOKE service in the case of a successfully performed operation.

The service elements and their corresponding XAPI functions needed for RO-RESULT are described in Table I.4-42.

Table I.4-42/T.180 – Service elements and their corresponding XAPI functions for the Result service

Service element	Service element identifier	XAPI function	Description
RO-RESULT Request	X_ROS_SP_RORE_Q	x_sndsp()	The RO-RESULT Request is passed to the provider to send a reply to a previous RO_INVOKE service in the case of a successfully performed operation.
RO-RESULT Indication	X_ROS_SP_RORE_I	x_rcvsp()	The RO-RESULT Indication is generated by the provider to indicate the reply of a successfully performed operation.

I.4.7.3.2.2 Service parameters

Table I.4-43 specifies the parameters of the RO-RESULT service.

Table I.4-43/T.180 – Parameters of the Result service

Parameter	RO-Result service	
	Request	Indication
X_ROS_P_CTXT_ID	M	M
X_ROS_P_INV_ID	M	M (=)
X_ROS_P_VAL_INT	U	C (=)
X_ROS_P_VAL_ID	U	C (=)
NOTE 1 – The operation result is contained in the *data* buffer of *sp_struct* of the *x_sndsp()* rsp. *x_rcvsp()* function. NOTE 2 – The parameter X_ROS_P_CTXT_ID is the presentation context identification which was negotiated in the ACSE connection establishment for the application.		

I.4.7.3.2.3 Service parameter description

Tables I.4-44 to I.4-47 list the parameters for the RO-RESULT services.

Table I.4-44/T.180

Parameter name	X_ROS_P_CTXT_ID
Type of value	long
Legal values	odd integer value
Default value	none
Description	The parameter indicates the presentation context identification of the application data of the RO-RESULT service.

Table I.4-45/T.180

Parameter name	X_ROS_P_INV_ID
Type of value	long
Legal values	any integer value
Default value	none
Description	The parameter identifies the corresponding RO-INVOKE service. The value is that of the corresponding RO-INVOKE indication service (X_ROS_SP_ROIV_Q).

Table I.4-46/T.180

Parameter name	X_ROS_P_VAL_INT
Type of value	long
Legal values	any integer value
Default value	none
Description	The parameter indicates the identifier of an invoked and successfully performed operation. The value is that of the corresponding RO-INVOKE indication service (X_ROS_SP_ROIV_I) and is only present if application data are available on this service. As the identifier of the operation may either be an integer value or an object identifier, either this parameter (X_ROS_P_VAL_INT) or the parameter (X_ROS_P_VAL_ID) may be present.

Table I.4-47/T.180

Parameter name	X_ROS_P_VAL_ID
Type of value	long[]
Legal values	Values must be given as a sequence of values in the decimal integer format that are separated by blanks and enclosed in braces (NumberForm of ASN.1 Object Identifier).
Default value	none
Description	The parameter indicates the identifier of an invoked and successfully performed operation. The value is that of the corresponding RO-INVOKE indication service (X_ROS_SP_ROIV_I) and is only present if application data are available on this service. As the identifier of the operation may either be an integer value or an object identifier, either this parameter (X_ROS_P_VAL_ID) or the parameter (X_ROS_P_VAL_INT) may be present.

I.4.7.3.3 RO-ERROR service

I.4.7.3.3.1 Service description

The RO-ERROR service is used to reply to a previous RO-INVOKE service in the case of an unsuccessfully performed operation.

The service elements and their corresponding XAPI functions needed for RO-ERROR are described in Table I.4-48.

Table I.4-48/T.180 – Service elements and their corresponding XAPI functions for the Error service

Service element	Service element identifier	XAPI function	Description
RO-ERROR Request	X_ROS_SP_ROER_Q	x_sndsp()	The RO-ERROR Request is passed to the provider to send a reply to a previous RO-INVOKE service in the case of an unsuccessfully performed operation.
RO-ERROR Indication	X_ROS_SP_ROER_I	x_rcvsp()	The RO-RESULT Indication is generated by the provider to indicate the reply of an unsuccessfully performed operation.

I.4.7.3.3.2 Service parameters

Table I.4-49 specifies the parameters of the RO-ERROR service.

Table I.4-49/T.180 – Parameters of the Error service

Parameter	RO-ERROR service	
	Request	**Indication**
X_ROS_P_CTXT_ID	M	M
X_ROS_P_INV_ID	M	M (=)
X_ROS_P_VAL_INT	C (Note 1)	C (=) (Note 1)
X_ROS_P_VAL_ID	C (Note 1)	C(=) (Note 1)
NOTE 1 – Either of the parameters X_ROS_P_VAL_INT or X_ROS_P_VAL_ID is present. NOTE 2 – The operation parameter is contained in the *data* buffer of *sp_struct* of the *x_sndsp()* rsp. *x_rcvsp()* function. NOTE 3 – The parameter X_ROS_P_CTXT_ID is the presentation context identification which was negotiated in the ACSE connection establishment for the application.		

I.4.7.3.3.3 Service parameter description

Tables I.4-50 to I.4-53 list the parameters for the RO-ERROR services.

Table I.4-50/T.180

Parameter name	X_ROS_P_CTXT_ID
Type of value	long
Legal values	odd integer value
Default value	none
Description	The parameter indicates the presentation context identification of the application data of the RO-ERROR service.

Table I.4-51/T.180

Parameter name	X_ROS_P_INV_ID
Type of value	long
Legal values	any integer value
Default value	none
Description	The parameter identifies the corresponding RO-INVOKE service. The value is that of the corresponding RO-INVOKE indication service (X_ROS_SP_ROIV_Q).

Table I.4-52/T.180

Parameter name	X_ROS_P_VAL_INT
Type of value	long
Legal values	any integer value
Default value	none
Description	The parameter identifies the error that occurred during the execution of the operation. As the identifier of the error may either be an integer value or an object identifier, either this parameter (X_ROS_P_VAL_INT) or the parameter X_ROS_P_VAL_ID may be present.

Table I.4-53/T.180

Parameter name	X_ROS_P_VAL_ID
Type of value	long[]
Legal values	Values must be given as a sequence of values in the decimal integer format that are separated by blanks and enclosed in braces (NumberForm of ASN.1 Object Identifier).
Default value	none
Description	The parameter identifies the error that occurred during the execution of the operation. As the identifier of the error may either be an integer value or an object identifier, either this parameter (X_ROS_P_VAL_INT) or the parameter X_ROS_P_VAL_ID may be present.

I.4.7.3.4 RO-REJECT-U service

I.4.7.3.4.1 Service description

The RO-REJECT-U service is used by a ROSE user to reject a request (RO-INVOKE service) or a reply (RO-RESULT, RO-ERROR services) of the other ROSE user if it has detected a problem.

The service elements and their corresponding XAPI functions needed for RO-REJECT-U are described in Table I.4-54.

Table I.4-54/T.180 – Service elements and their corresponding XAPI functions for the Reject-U service

Service element	Service element identifier	XAPI function	Description
RO-REJECT-U Request	X_ROS_SP_REJU_Q	x_sndsp()	The RO-REJECT-U Request is passed to the provider to send a rejection to a previous RO-INVOKE, RO-RESULT or RO-ERROR service if a problem has been detected by the ROSE user.
RO-REJECT-U Indication	X_ROS_SP_REJU_I	x_rcvsp()	The RO-REJECT-U Indication is generated by the provider to indicate a rejection of a previous RO-INVOKE, RO-RESULT or RO-ERROR service.

I.4.7.3.4.2 Service parameters

Table I.4-55 specifies the parameters of the RO-REJECT-U service.

Table I.4-55/T.180 – Parameters of the Checkpoint service

Parameter	RO-REJECT-U service	
	Request	Indication
X_ROS_P_INV_ID	U	C (=)
X_ROS_P_CTXT_ID	M	M
X_ROS_P_REJ_REAS	M	M (=)
NOTE – The parameter X_ROS_P_CTXT_ID is the presentation context identification which was negotiated in the ACSE connection establishment for the application.		

I.4.7.3.4.3 Service parameter description

Tables I.4-56 to I.4-58 list the parameters for the RO-REJECT-U service.

Table I.4-56/T.180

Parameter name	X_ROS_P_INV_ID
Type of value	long
Legal values	any integer value
Default value	none
Description	The parameter identifies the corresponding RO-INVOKE service. The value is that of the corresponding RO-INVOKE indication service (X_ROS_SP_ROIV_Q).

Table I.4-57/T.180

Parameter name	X_ROS_P_CTXT_ID
Type of value	long
Legal values	odd integer value
Default value	none
Description	The parameter indicates the presentation context identification of the data of the RO-REJECT-U service.

Table I.4-58/T.180

Parameter name	X_ROS_P_REJ_REAS	
Type of value	unsigned long	
Legal values	X_ROS_PV_DUP_INV	Duplication-invocation;
	X_ROS_PV_UNREC_OP	Unrecognized operation;
	X_ROS_PV_MIST_ARG	Mistyped argument;
	X_ROS_PV_RES_LIM	Resource limitation;
	X_ROS_PV_INIT_REL	Initiator-releasing;
	X_ROS_PV_UNREC_LINKID	Unrecognized linked identifier;
	X_ROS_PV_UNEXP_LINKRES	Unexpected link-response;
	X_ROS_PV_UNEXP_CHDOP	Unexpected child operation;
	X_ROS_PV_RE_UNREC_INV	Unrecognized invocation (reject of RO_RESULT);
	X_ROS_PV_UNEXP_RERES	Unexpected result response;
	X_ROS_PV_MIST_RES	Mistyped result;
	X_ROS_PV_ER_UNREC_INV	Unrecognized invocation (reject of RO-ERROR);
	X_ROS_PV_UNEXP_ERRES	Unexpected error response;
	X_ROS_PV_UNREC_ER	Unrecognized error;
	X_ROS_PV_UNEXP_ER	Unexpected error;
	X_ROS_PV_MIST_PAR	Mistyped parameter.
Default value	none	
Description	The parameter specifies the reason for rejection.	

I.4.7.3.5 RO-REJECT-P service

I.4.7.3.5.1 Service description

The RO-REJECT-P service is used to advise a ROSE user of a problem detected by a ROSE provider.

The service elements and their corresponding XAPI functions needed for RO-REJECT-P are described in Table I.4-59.

Table I.4-59/T.180 – Service elements and their corresponding XAPI functions for the Reject-P service

Service element	Service element identifier	XAPI function	Description
RO-REJECT-P Indication	X_ROS_SP_ROREJP_I	x_rcvsp()	The RO-REJECT-P Indication is generated by the provider to indicate a provider rejection to a previous RO-INVOKE, RO-RESULT or RO-ERROR service if a problem has been detected by the ROSE provider.

I.4.7.3.5.2 Service parameters

Table I.4-60 specifies the parameters of the RO-REJECT-P service.

Table I.4-60/T.180 – Parameters of the Reject-P service

Parameter	RO-REJECT-P Service
	Indication
X_ROS_P_INV_ID	C
X_ROS_P_PREJ_REAS	M
NOTE – Due to the fact that the Linked-Id and Invoke-Id are not examined by the ROSE provider and the application data of the ROSE APDUs are already encoded by the XAPI user, the ROSE provider will accept all ROSE request service primitives from the XAPI user. Therefore, it is not necessary to support the Returned-parameters parameter of the RO-REJECT-P service at the XAPI.	

I.4.7.3.5.3 Service parameter description

Tables I.4-61 to I.4-62 list the parameters for the RO-REJECT-P service.

Table I.4-61/T.180

Parameter name	X_ROS_P_INV_ID
Type of value	long
Legal values	any integer value
Default value	none
Description	The parameter identifies the corresponding invocation. The value is that of the corresponding RO-INVOKE, RO-RESULT, RO-ERROR or RO-REJECT-U service.

Table I.4-62/T.180

Parameter name	X_ROS_P_PREJ_REAS
Type of value	unsigned long
Legal values	X_ROS_PV_UNREC_APDU unrecognized APDU X_ROS_PV_MIST_APDU mistyped APDU X_ROS_PV_BAD_STRU badly structured APDU
Default value	none
Description	The parameter specifies the reason for rejection.

I.4.7.3.6 State transition tables/diagrams

To handle the ROSE services while the service endpoint is in state X_CONNECTED, the ROSE service provider distinguishes no additional states.

I.4.7.4 Connection Release service

I.4.7.4.1 Service description

The orderly release service allows either ACSE service user to release the ACSE connection in an orderly manner. This is done cooperatively between the two ACSE service users without the loss of data after all in-transit data have been delivered and accepted by both ACSE service users.

The service elements and their corresponding XAPI functions needed for ACSE Connection Release service are described in Table I.4-63.

Table I.4-63/T.180 – Service elements and their corresponding XAPI functions for the Connection Release service

Service element	XAPI function	Description
Release Request	x_relreq()	The Release Request is passed to the provider to request a normal ACSE connection release.
Release Indication	x_relind()	The Release Indication is generated by the provider to indicate the orderly release of a ACSE connection by the remote side.
Release Response	x_relrsp()	The Release Response is passed to the provider as reaction to a previously received Release Indication as positive response.
Release Confirmation	x_relconf()	The Release Confirmation is generated by the provider as positive confirmation of a normal ACSE release.
End Indication	x_rcvend()	The End Indication is generated by the service provider to indicate that the service provider is ready to establish a new connection. In addition, it contains some information about the released connection.

I.4.7.4.2 Service parameters

Table I.4-64 specifies the parameters of the ACSE Connection Release service.

Table I.4-64/T.180 – Parameters of the Connection Release service

Parameter	Release service				
	Request	**Indication**	**Response**	**Confirmation**	**End Indication**
X_ACS_P_REASON	U	C (=)	U	C (=)	
X_P_CONN_TIME					M
X_P_DISC_TIME					M
X_P_CHARGE					C
X_P_DISC_REASON					C

I.4.7.4.3 Service parameter description

Tables I.4-65 to I.4-69 define the parameter for the Connection Release service.

Table I.4-65/T.180

Parameter name	X_ACS_P_REASON
Type of value	unsigned long
Legal values	When used on the Request or Response primitive: X_ACS_PV_NORMAL Normal; X_ACS_PV_USDEF User defined. When used on the Request primitive: X_ACS_PV_URGENT Urgent. When used on the reponse primitive: X_ACS_PV_NOT_FIN Not finished.
Default value	none
Description	The parameter indicates the general urgency of the Release Request and in the Release Reponse information about the Release Request.

Table I.4-66/T.180

Parameter name	X_P_CONN_TIME
Type of value	unsigned long
Legal values	the time in seconds since 01.01.1970 00:00:00 GMT
Default value	none
Description	This parameter contains the time of physical connection.

Table I.4-67/T.180

Parameter name	X_P_DISC_TIME
Type of value	unsigned long
Legal values	the time in seconds since 01.01.1970 00:00:00 GMT
Default value	none
Description	This parameter contains the time of physical disconnection. If X_P_CONN_TIME and X_P_DISCON_TIME were both set to zero, no physical connection could be established.

Table I.4-68/T.180

Parameter name	X_P_CHARGE
Type of value	unsigned long
Legal values	any number
Default value	zero
Description	This parameter contains, if available, the charging unit of the connection. It is only set if both network and network connection provide this facility.

Table I.4-69/T.180

Parameter name	X_P_DISC_REASON
Type of value	unsigned long
Legal values	The values are directly mapped from the underlying hardware. See the hardware documentation to get further information.
Default value	zero
Description	This parameter contains, if available, the disconnection reason.

I.4.7.5 Connection Abort service

I.4.7.5.1 Service description

The abort service provides the means by which either ACSE service user or ACSE itself can instantaneously release the ACSE connection and have the other ACSE service user informed of the release. Use of this service will cause loss of undelivered data.

The *origination* parameter of the *x_snddis()* and *x_rcvdis()* function must be set to ABORT in this case.

The service elements and their corresponding XAPI functions needed for user abort of ACSE connection are described in the Table I.4-70.

Table I.4-70/T.180 – Service elements and their corresponding XAPI functions for the Connection Abort service

Service element	XAPI function	Description
Abort Request	x_snddis()	The Abort Request is passed to the provider to request an abnormal ACSE connection release.
Abort Indication	x_rcvdis()	The Abort Indication is generated by the provider to indicate the abnormal release of an ACSE connection due to ACSE user.
End Indication	x_rcvend()	The End Indication is generated by the service provider to indicate that the service provider is ready to establish a new connection. In addition, it contains some information about the released connection.

I.4.7.5.2 Service parameters

Table I.4-71 specifies the ACSE service elements and their parameters needed for abortive ACSE release.

NOTE – If the Abort service primitive contains user data, additional information is necessary to qualify the user data. When relying on the XAPI ACSE/ROSE service provider, the coding/decoding of ACSE user data has to be done by the ACSE service user.

If the abort service primitive contains user data and the ACSE Connect Request service primitive contained a context definition list, a so-called "presentation context identifier list" (see also 6.4.2/[X.226]) has to be specified in the abort service primitive. It consists of a list of pairs X_ACS_P_CTXT_ID (representing a presentation context id) and X_ACS_P_TS (representing the name of the transfer syntax used for the coding of the specified presentation context).

The list has to contain one pair for each presentation context used in the user data parameter. The second parameter of such a pair is optional; if omitted, the default value representing X.209 is used. If the transfer syntax parameter is available, it has to follow immediately the corresponding context id parameter. Furthermore, the elements contained in the list have to follow immediately one another.

Table I.4-71/T.180 – Parameters of the Connection Abort service

Parameter	Abort service		
	Request	Indication	End Indication
X_ACS_P_AB_SRC		C	
X_ACS_P_CTXT_ID	C	C (=)	
X_ACS_P_TS	C	C (=)	
X_P_CONN_TIME			M
X_P_DISC_TIME			M
X_P_CHARGE			C
X_P_DISC_REASON			C

I.4.7.5.3 Service parameter description

Tables I.4-72 to I.4-78 list the parameters for the user abortive ACSE abort service.

Table I.4-72/T.180

Parameter name	X_ACS_P_AB_SRC
Type of value	unsigned long
Legal values	X_ACS_PV_SU ACSE service user X_ACS_PV_SP ACSE service provider
Default value	X_ACS_PV_SU
Description	The parameter indicates the initiating source of the abort.

Table I.4-73/T.180

Parameter name	X_ACS_P_CTXT_ID
Type of value	unsigned long
Legal values	identifier of presentation contexts contained in the user data
Default value	none
Description	This parameter identifies one of the presentation contexts used in the user data parameter of the service primitive. For each used presentation context, a value has to be provided.

Table I.4-74/T.180

Parameter name	X_ACS_P_TS
Type of value	long[]
Legal values	Values must be given as a sequence of long values which form an object identifier identifying a transfer syntax.
Default value	{2, 1, 1} the object identifier of transfer syntax X.209
Description	The parameter indicates the transfer syntax name belonging to the previously specified presentation context identifier. If available, it has to follow immediately the parameter X_ACS_P_CTXT_ID.

Table I.4-75/T.180

Parameter name	X_P_CONN_TIME
Type of value	unsigned long
Legal values	the time in seconds since 01.01.1970 00:00:00 GMT
Default value	none
Description	This parameter contains the time of physical connection.

Table I.4-76/T.180

Parameter name	X_P_DISC_TIME
Type of value	unsigned long
Legal values	the time in seconds since 01.01.1970 00:00:00 GMT
Default value	None
Description	this parameter contains the time of physical disconnection. If X_P_CONN_TIME and X_P_DISCON_TIME both set to zero, no physical connection could be established.

Table I.4-77/T.180

Parameter name	X_P_CHARGE
Type of value	unsigned long
Legal values	any number
Default value	zero
Description	This parameter contains, if available, the charging unit of the connection. It is only set if both network and network connection provide this facility.

Table I.4-78/T.180

Parameter name	X_P_DISC_REASON
Type of value	unsigned long
Legal values	The values are directly mapped from the underlying hardware. See the hardware documentation to get further information.
Default value	zero
Description	This parameter contains, if available, the disconnection reason.

I.4.7.6 Connection Provider Abort service

I.4.7.6.1 Service description

The provider abort service provides the means by which the ACSE protocol modules may indicate the release of the connection for reason below to the ACSE service provider.

The *origination* parameter of the *x_snddis()* and *x_rcvdis()* function must be set to PROVIDER_ABORT in this case.

The provider abort service elements and their corresponding XAPI functions are described in Table I.4-79.

Table I.4-79/T.180 – Service elements and their corresponding XAPI functions for the Connection Provider Abort service

Service element	XAPI function	Description
Provider Abort Indication	x_rcvdis()	The Provider Abort Indication service is generated by the provider to indicate the abnormal release of an ACSE connection by the ACSE provider.
End Indication	x_rcvend()	The End Indication is generated by the service provider to indicate that the service provider is ready to establish a new connection. In addition, it contains some information about the released connection.

I.4.7.6.2 Service parameters

Table I.4-80 specifies the parameters needed for provider abortive ACSE release.

Table I.4-80/T.180 – Parameters of the Connection Provider Abort service

Parameter	Provider Abort service	
	Indication	End Indication
X_ACS_P_PRO_REAS	C	
X_P_CONN_TIME		M
X_P_DISC_TIME		M
X_P_CHARGE		C
X_P_DISC_REASON		C

I.4.7.6.3 Service parameter description

Tables I.4-81 to I.4-85 list the parameters for the provider abortive ACSE release service.

Table I.4-81/T.180

Parameter name	X_ACS_P_PRO_REAS	
Type of value	unsigned long	
Legal values	X_ACS_PV_NOT_SPEC	Reason not specified;
	X_ACS_PV_UNREC_PPDU	Unrecognized PPDU;
	X_ACS_PV_UNEXP_PPDU	Unexpected PPDU;
	X_ACS_PV_UNEXP_SSP	Unexpected session service primitive;
	X_ACS_PV_UNREC_PAR	Unrecognized PPDU parameter;
	X_ACS_PV_UNEXP_PAR	Unexpected PPDU parameter;
	X_ACS_PV_INV_PARVAL	Invalid PPDU parameter value;
	X_ACS_PV_TRANS_DIS	Transport connection aborted;
	X_ACS_PV_SESS_PROTERR	Session protocol error;
	X_ACS_PV_SESS_PICS	Session unable to support requested feature.
Default value	X_ACS_PV_NOT_SPEC	
Description	The parameter indicates the reason for the termination of the association.	

Table I.4-82/T.180

Parameter name	X_P_CONN_TIME
Type of value	unsigned long
Legal values	the time in seconds since 01.01.1970 00:00:00 GMT
Default value	none
Description	This parameter contains the time of physical connection.

Table I.4-83/T.180

Parameter name	X_P_DISC_TIME
Type of value	unsigned long
Legal values	the time in seconds since 01.01.1970 00:00:00 GMT
Default value	none
Description	This parameter contains the time of physical disconnection. If X_P_CONN_TIME and X_P_DISCON_TIME both set to zero, no physical connection could be established.

Table I.4-84/T.180

Parameter name	X_P_CHARGE
Type of value	unsigned long
Legal values	any number
Default value	zero
Description	This parameter contains, if available, the charging unit of the connection. It is only set if both network and network connection provide this facility.

Table I.4-85/T.180

Parameter name	X_P_DISC_REASON
Type of value	unsigned long
Legal values	The values are directly mapped from the underlying hardware. See the hardware documentation to get further information.
Default value	zero
Description	This parameter contains, if available, the disconnection reason.

I.4.7.7 Usage of XAPI functions

This subclause provides some protocol-specific remarks to the use of the XAPI functions. The functions are mentioned in alphabetical order.

x_sndsp() This XAPI function contains in the *data* buffer of *sp_struct* the encoded *argument*, *result* and *parameter* components of the ROSE APDUs in encoded form.

x_rcvsp() This XAPI function contains in the *data* buffer of *sp_struct* the *argument*, *result* and *parameter* components of the ROSE APDUs in encoded format.

I.4.7.8 Tables of error codes

The XAPI error-level error codes are defined in Annex B.

I.4.7.8.1 CC_BADVALUE

If the cause code indicates a parameter error with a bad value, the value of *diagnostic* will contain the erroneous parameter identifier which has been submitted with the XAPI function call that caused the error indication or one of the identifiers X_ACS_CTX_DEF_LIST, X_ACS_CTX_DEF_RES_LIST, X_ACS_CTX_ID_LIST, if the error occurred within a context definition list, a context definition result list or a context identifier list, respectively.

I.4.7.8.2 CC_MANDMISS

If the cause code indicates that a mandatory parameter is missing, the value of *diagnostic* will contain the missing parameter identifier that caused the error indication.

I.4.7.8.3 CC_BADEVENT

If the cause code indicates a bad event, the value of *diagnostic* will contain the bad event identifier which has been submitted with the XAPI function call that caused the error indication.

I.4.7.8.4 CC_UNEXPECT

If the cause code indicates a unexpected event, the value of *diagnostic* will contain the actual state identifier in which the unexpected event caused the error indication.

I.4.7.8.5 CC_NOTSUPPORT

If the cause code indicates an unsupported event, the value of *diagnostic* will contain the identifier of the unsupported event which has been submitted with the XAPI function call that caused the error indication.

I.4.7.8.6 CC_OTHER

If the cause code indicates the CC_OTHER error code, the value of *diagnostic* will contain the identifier which caused the error indication.

I.5 XAPI access to a Service Provider for Audio and Video (AV) Control

This part of Appendix I describes an example of how the service provider can be implemented if an application needs access to the specified service.

I.5.1 Scope

The XAPI, which is an abbreviation for eXtensive Application Programming Interface, is an operating system and language-independent programming interface to general communication services. Detailed information about the XAPI is given in the main part of this Recommendation, which is very important for understanding this appendix.

Which services are made available via the XAPI depends on the installed service providers and not on the XAPI, which only provides the access mechanism.

The AV Codec can be seen as the "heart" of an audiovisual terminal. The service provider enables the XAPI user to:

- establish a connection to a remote communication partner;
- release or disconnect a previously established connection;
- control the communication to the remote partner; and
- control the behaviour of the local AV Codec.

The processing of audio and video signals is solely the task of the AV Codec. The AV Codec Controller only controls this processing. The communication path for the terminal-to-network control signals is provided in the D-channel while the AV communication with the remote entity, the path for the terminal-to-terminal control signals is provided as a single $B/H_0/H_{11}/H_{12}$-channel or multiple B/H_0-channels.

This part does not refer to a concrete AV Codec, but describes the access to an ideal service provider according to Recommendation H.320 and related Recommendations. A concrete service provider, which is built on a certain AV Codec hardware, need not implement exactly this ideal service provider. Thus, it is necessary to have some release notes for each concrete service provider that explain the differences to the ideal service provider.

Each concrete service provider has to implement all services described throughout this part and their standardized parameters. Note that the set of allowed values for a standardized service parameter may be restricted by a concrete service provider. It should not be necessary to extend the value range of a standardized parameter because it is defined as the maximum possible according to the Recommendations. The non-standardized service parameters described here provide a guideline for the implementation of useful non-standard features in a concrete service provider. They need not be implemented at all and even additional non-standard parameters, not mentioned here, may be implemented. But in order to provide portability of XAPI applications, it is a good idea to follow these guidelines as far as possible in the design of a concrete service provider. (For the same reason XAPI application programmers should not persist in the presence of a certain non-standard parameter.)

I.5.2 References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; all users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published.

[H.221] ITU-T Recommendation H.221 (1997), *Frame structure for a 64 to 1920 kbit/s channel in audiovisual teleservices*.

[H.230] ITU-T Recommendation H.230 (1997), *Frame-synchronous control and indication signals for audiovisual systems*.

[H.231] ITU-T Recommendation H.231 (1997), *Multipoint control units for audiovisual systems using digital channels up to 1920 Kbit/s*.

[H.242] ITU-T Recommendation H.242 (1993), *System for establishing communication between audiovisual terminals using digital channels up to 2 Mbit/s*.

[H.243] ITU-T Recommendation H.243 (1997), *Procedures for establishing communication between three or more audiovisual terminals using digital channels up to 1920 Kbit/s*.

[H.261] ITU-T Recommendation H.261 (1993), *Video codec for audiovisual services at* $p \times 64$ *kbit/s*.

[H.320] TU-T Recommendation H.320 (1997), *Narrow-band visual telephone systems and terminal equipment*.

[H.331] ITU-T Recommendation H.331 (1993), *Broadcasting type audiovisual multipoint systems and terminal equipment*.

[I.430] ITU-T Recommendation I.430 (1995), *Basic user network interface – Layer 1 specification*.

[I.431] ITU-T Recommendation I.431 (1993), *Primary rate user network interface – Layer 1 specification*.

I.5.3 Definitions

I.5.4 Abbreviations

This part uses the following abbreviations:

AC Additional Channel

AV Audio and Video

BAS Bit-rate Allocation Signal

C&I Control and Indication (H.320)

DSS 1 Digital Subscriber Signalling System No. 1

ECS Encryption Control Signal

FAS Frame Alignment Signal

H_0 Channel consisting of six 64 kbit/s time-slots (384 kbit/s)

H_{11} Channel consisting of twenty-four 64 kbit/s time-slots (1536 kbit/s)

H_{12} Channel consisting of thirty 64 kbit/s time-slots (1920 kbit/s)

HSD High-Speed Data

IC Initial Channel

ISDN Integrated Services Digital Network

ISO International Organization for Standardization

LSD Low-Speed Data

MBE Multiple Byte Extension

MCU Multipoint Control Unit

MLP Multi-Layer Protocol

NSAP Network Service Access Point

OSI Open Systems Interconnection

PHC is used as service identification part in the names defined by the Access to an AV Codec Controller Service. The letters PHC are short for "**P**hysical layer, access to an **H**-**C**odec controller".

SBE Single Byte Extension

SC Service Channel

TS Time Slot

XAPI eXtensive Application Programming Interface

I.5.5 Conventions

Each service is described via three kinds of tables.

In the first kind of table the service and its service elements are described. It contains a row for each service element of the service (where service element means either request, indication, response, or confirmation) with its corresponding XAPI function and a short description.

In the second kind of table the use of the service primitive parameters within the service elements are described. It contains a row for each service primitive parameter of the service. The service primitive parameter name is stored within the first column; the succeeding columns contain the use of the parameter in the service elements:

Blank The service parameter is absent.

C Presence of the service parameter is conditional. Firstly, there may be a condition in the service provider to provide a parameter in an indication or confirmation, secondly, there may be interdependencies between parameters of the same or the preceding service primitive.

M Presence of the service parameter is mandatory.

U Presence of the service parameter is a user option. If the user does not specify a value for such a parameter and there is no default value for that parameter, nothing is passed to the service provider. If the user does not specify a value for such a parameter and there is a default value, the default value is passed to the service provider.

(=) The value of the service parameter is identical to the value of the corresponding service parameter in the preceding service element. In the special case of a parameter, whose presence in the preceding service is a user option, for which a default value is defined, and the parameter was not specified in the preceding service element, the symbol (=) indicates that the parameter value is identical to the default value.

The third kind of table is used to describe the service primitive parameter containing the type of value and its scope, possible default values and a detailed description. For each service primitive parameter, one description table is given.

A **naming convention** is used throughout the XAPI in order to distinguish between the common functions, identifiers and values, and service-specific identifiers and values. All service-specific settings of the XAPI AV Codec service access are defined within the present part of this appendix and start with **X_PHC_** or **x_phc_**.

Extended naming conventions:

– SP_ Service Primitive;

– P_ Parameter;

– PV_ Parameter Value;

– O_ Option;

– OV_ Option Value.

I.5.6 Introduction to the video codec service provider access

Because of the technical progress in video processing and in telecommunication, it is possible to transfer AV data in a high quality in narrow-band networks. Recommendation H.320 and its related Recommendations define inband and outband signalling procedures as well as codings for audio and video data for visual telephone systems. Recommendation H.221 defines the frame structure for 64 to 1920 kbit/s channel in audiovisual services, and Recommendation H.242 defines a system for establishing communication between audiovisual terminals using digital channels up to 2 Mbit/s.

An important requirement of the H.320 series is the compatibility of videophones to ISDN – and analog telephones. All videophones start the connection establishment phase using one $B/H_0/H_{11}/H_{12}$-channel, audio mode G.711 (3.1 kHz, 64 kbit/s). After the connection is established, both terminals exchange their capabilities and change to the best common mode of operation. This initialization procedure is defined in Recommendation H.242. In order to increase bandwidth, additional B/H_0-channels may be established by the calling terminal. A larger bandwidth allows the improvement of the quality of speech as well as the best quality for the motion image sequences.

Table I.5-1 shows important video communication modes in ISDN according to Recommendations H.221 and H.242:

Table I.5-1/T.180 – Audio/Video communication modes in ISDN

Number of B-channels	Video	Audio
Use of one B-channel	46.4 kbit/s	16 kbit/s: 3.1 kHz
Use of two B-channels	68.8 kbit/s	56 kbit/s: 3.1 kHz or 7 kHz
	92.8 kbit/s	32 kbit/s: 7 kHz
	108.8 kbit/s	16 kbit/s: 3.1 kHz

NOTE – The use of two B-channels improves the quality of videophone communications. The video bandwidth of 108.8 kbit/s provides the best quality of picture in ISDN if two B-channels are used.

The XAPI Audio/Video Codec Control Service, as the name already suggests, provides the means to *control* a service provider performing audio/video communication conforming to the requirements of H.320 and its related Recommendations H.221, H.230, H.231, H.242 and H.243.

Service primitives for connection establishment and release as well as for switching the mode of communication (e.g. change bandwidth for the audio data, change number of used lines) or the local receive capabilities are provided.

The audio/video data themselves are actually not passed between XAPI user and service provider via XAPI function calls. These isochronous data streams have to be transferred without any delays between the system components. Commonly available hardware in personal computers does not allow to carry these data fast enough via the bus, therefore they are transferred by directly interconnecting audio/video I/O equipment, AV Codec and network interface.

According to Recommendation H.320 and related Recommendations, the subscribers are also able to initiate a transfer of asynchronous data (e.g. transmission of still images, transmission of group 3 or group 4 facsimile documents) besides audio/video communication. Transfer of asynchronous data is not covered by the restrictions explained above. Therefore, a data transfer service is provided to exchange asynchronous data between the XAPI user and the service provider. The available video bandwidth is decreased by the data transfer rate.

I.5.7 Description of the access to the Video Codec Service Provider

I.5.7.1 Service initialization

I.5.7.1.1 Creation of a AV Codec Service Access Point with x_open()

A communication endpoint accessing the AV Codec service provider is created when calling the *x_open()* function with an appropriate service provider identification string. The available identifiers depend on the actual system configuration. In the standard configuration, "**X_PHC_ISDN**" identifies the AV Codec control service provider with ISDN as underlying network.

I.5.7.1.2 Activation of a AV Codec Service Access Point with *x_bind()*

x_bind() is to be called to activate the AV Codec service endpoint. The function has the task to bind the application's own address to the service endpoint.

I.5.7.1.3 Addresses

The network address (NSAP) is used to identify the called or calling entity.

I.5.7.1.3.1 Specification of the application's own address

The own address may be specified in the *own_address* buffer of the *bind_struct* passed as argument to the *x_bind()* function. For a passive application it is returned in the *called_addr* buffer of the *x_conind()* function.

For a passive application it is not supported to specify the (own) responding NSAP address in the *address* buffer of the *call_struct* in the *x_conrsp()* call, because this value is not transferred by the network.

Note that specification of the application's own protocol address in the *x_bind()* call is optional. If no address information is specified, it is taken from the XAPI configuration and the actually bound address is returned as output parameter of the *x_bind()* function. It is strongly recommended to trust in the XAPI configuration and not to specify the own address in the *x_bind()* call.

The own address consists of the NSAP address only. The NSAP address has to comprise the local ISDN outband address, i.e. the address information used in the D-channel. The own ISDN inband address and subaddress parameters as well as the protocol selectors are meaningless in the Audio/Video Codec Control service. If specified, they will be ignored.

Table I.5-2 shows the address component which has to be specified in the *x_bind()* call:

Table I.5-2/T.180 – Address component specified in the *x_bind()* call

ISDN network	Address component	Value
ISDN/DSS 1	A_OUTBAND_ADR	A decimal digit, which is locally mapped to one of the available MSNs (Multiple Subscriber Numbers)

Binding of multiple addresses

In an audiovisual communication according to Recommendations H.221/H.241, multiple channels (up to six B-channels) may be used for one connection. Therefore, multiple addresses have to be bound to a service endpoint if this feature is to be used. The binding of multiple addresses can be achieved by specifying as many A_OUTBAND_ADR address components in the *own_address* buffer of the *x_bind ()* call, as addresses have to be bound. The number of A_OUTBAND_ADR components is limited to six for the AV Codec Control service provider.

A passive application has to specify one address for each B-channel to be established. If two B-channels shall be established on the same address, this address has to be mentioned twice in the *own_address* buffer. Or, the other way around, if an address is specified only once, then only one (ISDN) connection will be accepted for that address.

I.5.7.1.3.2 The address of the communication partner

On the active side, the address of the communication partner has to be specified in the *address* buffer of the *call_struct* passed as argument to the *x_conreq()* function. On the passive side, the address of the communication partner is returned in the *calling_addr* buffer of the *x_conind()* function.

The address of the communication partner comprises at least the peer's ISDN outband address. The peer's ISDN inband address and subaddress as well as the protocol selectors are meaningless for the Audio/Video Codec Control service.

Note that only one called outband address may be specified in the *x_conreq()*. It is used to establish the initial channel of the AV connection. Additional channels may be added later with the elements of the Mode Switch service. Therefore, only one calling address is returned in the connect indication on the passive side.

Table I.5-3 shows the address component to be used in the *address* buffer specifying the called NSAP address in a *x_conreq*() call :

Table I.5-3/T.180 – Address component specifying the called NSAP address in a *x_conreq()* call

ISDN network	Address component	Value
ISDN/DSS 1	A_OUTBAND_ADR	Optional the country code, optional the area code and the Multiple Subscriber Number (MSN)

I.5.7.1.4 Configuration of the service provider

Protocol options are used to control the general behavior of the service provider. There is a default value defined for each option. These preconfigured values are sufficient for the majority of communication relations. The Option Management service can be used to set protocol options and to retrieve the current or default values of protocol options.

I.5.7.2 Connection Establishment service

I.5.7.2.1 Service description

During the connection establishment phase, two users of the same service establish a connection between them. The XAPI user must already have prepared a service endpoint before the connection establishment phase can start.

For active establishment of an AV connection, the XAPI function *x_conreq()* is used. The user can optionally specify local capabilities as service parameters. The local capabilities define the receiver capabilities of the own AV terminal. The service provider generates a positive or negative connect confirmation primitive as a response to the connect request. The positive confirmation signals that an ISDN connection, the Initial Channel (IC), has been established successfully and that both terminals have started to exchange their capabilities on the IC. Later, when the mode initialization procedure is completed on the IC, Additional Channels (ACs) may be added to the connection to increase the total bandwidth. The establishment of additional channels is done with elements of the Mode Switch service. Note that only the AV terminal that initiated the establishment of the IC [the terminal which called *x_conreq()*], is allowed to establish additional channels.

At the passive side, the service provider generates a connect indication to indicate a pending call of a remote AV terminal. The XAPI user can accept or reject the call by calling the XAPI function *x_conrsp()*. If the connection is accepted, the capability exchange procedure and the mode initialization procedure according to Recommendation H.242 starts. In the *x_conrsp()* function call, the local receive capabilities may optionally be specified as service parameters. The default capabilities are used, if none are specified.

For both the active and the passive sides, the **X_PHC_SP_INIT_COMPL** service primitive indicates the successful completion of the mode initialization procedure. Both terminals have exchanged their receive capabilities and switched their receiver/transmitter to a common mode of communication. If the initialization procedure fails, the AV connection is cleared and the service provider generates a disconnect indication. The XAPI user then has to call *x_rcvdis()* as consuming function.

The elements of the Connection Establishment service and the corresponding XAPI functions are defined in Table I.5-4.

Table I.5-4/T.180 – Service elements and their corresponding XAPI functions for Connection Establishment

Service element	XAPI function	Description
Connect Request	x_conreq()	The Connect Request is passed to the provider to request the establishment of an AV connection. The own local capabilities to be used in the initial capability exchange may be specified as service parameters. If no capabilities are specified, the default values are used.
Connect Indication	x_conind()	The Connect Indication is generated by the provider to indicate a pending call from a remote terminal. The physical connection is not yet established. A positive connect response will accept the call and start the mode initialization procedure according to Recommendation H.242. The AV connection will be fully operational after successful completion of the initialization procedure. This is indicated with an X_PHC_SP_INIT_COMPL primitive. A negative connect response will reject the call.

Table I.5-4/T.180 – Service elements and their corresponding XAPI functions for Connection Establishment *(concluded)*

Service element	XAPI function	Description
Connect Response	x_conrsp()	The Connect Response is passed to the provider as reaction to a previously received connect indication. A positive connect response will accept the call and start the mode initialization procedure according to Recommendation H.242. A negative connect response will reject the call. If the call is accepted, the own local capabilities to be used in the initial capability exchange may be specified as service parameters. If no capabilities are specified, the default values are used.
Connect Confirmation	x_conconf()	The Connect Confirmation is generated by the provider as positive or negative response to a previous connection establishment request. A positive confirmation indicates that the remote terminal accepted the call and the mode initialization procedure is just going on. When it is finished successfully, the AV connection is fully operational. During mode initialization the audio path of the connection may be already available.

The element of the mode initialization procedure and the corresponding XAPI function are defined in Table I.5-5.

Table I.5-5/T.180 – Service elements and their corresponding XAPI function for mode initialization

Service element	XAPI function	Service element identifier	Description
Initialization Complete	x_rcvsp()	X_PHC_SP_INIT_COMPL	The X_PHC_SP_INIT_COMPL primitive is generated by the service provider to indicate that the mode initialization procedure according to Recommendation H.242 has been finished successfully. The local and remote capabilities and the selected receive and transmit communication mode are presented to the XAPI user as service parameters. Both terminals are operating in the indicated mode. The XAPI user should store the remote capabilities and observe them as limits to avoid failures in later mode switch requests.

I.5.7.2.2 Service parameters

The following tables specify the parameters of the Connection Establishment service. There are two groups of parameters: the capability parameters and the communication mode parameters.

The table below shows the local and remote receiving capability parameters. The capabilities are important for the selection of the communication mode used between the two AV terminals. Each side has to select a transmitting mode which is compatible with the receive capabilities of the other terminal. The selected transmitting mode is indicated to the partner with BAS commands and the own transmitter is switched to the selected mode. The other terminal has to switch its receiver according to the mode indicated by the received BAS commands. The local capability parameters share the prefix X_PHC_P_**L**CAP_ in their names, and the remote capability parameters share the prefix X_PHC_P_**R**CAP_ .

The specification of local capabilities is optional. If a capability parameter is not specified in a service element, the default value will be used by the service provider. The default value itself is defined by the value of a protocol option. For each capability parameter there is a corresponding protocol option which defines the default value for this parameter. For the protocol option, a constant default value is defined in the XAPI configuration.

The local receive capabilities are transmitted to the remote terminal during connection establishment. If the value X_PHC_PV_CAPNON is specified for a capability parameter (either explicit or implicit as default), this capability is not transmitted to the remote terminal.

The receive capabilities of the remote terminal will be available after successful completion of the H.242 mode initialization procedure. They are presented to the service user as parameters of the init-complete service element (X_PHC_SP_INIT_COMPL). For capabilities which were not transmitted by the remote terminal, the value X_PHC_PV_CAPNON is returned.

Table I.5-6 specifies the capability parameters of the Connection Establishment service.

Table I.5-6/T.180 – Capability Parameters of the Connection Establishment service

Parameter	Connect service				
	Request	**Indication**	**Response**	**Confirmation**	**Init Complete**
X_PHC_P_LCAP_AUDIO	U		U		M
X_PHC_P_LCAP_VIDEO	U		U		M
X_PHC_P_LCAP_DATA	U		U		M
X_PHC_P_LCAP_HDATA	U		U		M
X_PHC_P_LCAP_TFRATE	U		U		M
X_PHC_P_LCAP_TFLINES	U		U		M
X_PHC_P_LCAP_MISC	U		U		M
X_PHC_P_LCAP_DATAPPL	U		U		M
X_PHC_P_LCAP_NONSTD	U		U		C
X_PHC_P_FALLBACK	U				
X_PHC_P_RCAP_AUDIO					M
X_PHC_P_RCAP_VIDEO					M
X_PHC_P_RCAP_DATA					M
X_PHC_P_RCAP_HDATA					M
X_PHC_P_RCAP_TFRATE					M
X_PHC_P_RCAP_TFLINES					M
X_PHC_P_RCAP_MISC					M
X_PHC_P_RCAP_DATAPPL					M
X_PHC_P_RCAP_NONSTD					C

Table I.5-7 specifies the communication mode parameters. They are always returned by the init-complete primitive and indicate the receive and transmit mode that is currently used in the audiovisual communication on the actually established connection. The receive mode parameters share the prefix X_PHC_P_**R**MOD_ in their names and the transmit mode parameters share the prefix X_PHC_P_**T**MOD_.

Table I.5-7/T.180 – Communication mode parameters of the Connection Establishment service

Parameter	Connect service				
	Request	**Indication**	**Response**	**Confirmation**	**Init-complete**
X_PHC_P_RMOD_AUDIO					M
X_PHC_P_RMOD_VIDEO					M
X_PHC_P_RMOD_DATA					M
X_PHC_P_RMOD_HDATA					M
X_PHC_P_RMOD_TFRATE					M
X_PHC_P_RMOD_SYNC					M
X_PHC_P_RMOD_DATAPPL					M
X_PHC_P_RMOD_MISC					M
X_PHC_P_TMOD_AUDIO					M
X_PHC_P_TMOD_VIDEO					M
X_PHC_P_TMOD_DATA					M
X_PHC_P_TMOD_HDATA					M
X_PHC_P_TMOD_TFRATE					M
X_PHC_P_TMOD_SYNC					M
X_PHC_P_TMOD_DATAPPL					M
X_PHC_P_TMOD_MISC					M
X_PHC_P_MODE_TFLINES					M

I.5.7.2.3 Service parameter descriptions

Tables I.5-8 to I.5-26 define the capability parameters for the elements of the Connection Establishment service. The definitions of the communication mode parameters can be found below in I.5.7.3.2, "Mode Switch service".

I.5.7.2.3.1 Local capability parameters

Table I.5-8/T.180

Parameter name	X_PHC_P_LCAP_AUDIO
Type of value	long
Legal values	X_PHC_PV_CAPNON X_PHC_PV_AUDIO_711_A X_PHC_PV_AUDIO_711_U X_PHC_PV_AUDIO_722_64 X_PHC_PV_AUDIO_722_48 X_PHC_PV_AUDIO_16K X_PHC_PV_AUDIO_ISO
Default value	The default value is defined by the value of the X_PHC_O_LCAP_AUDIO option.
Description	The parameter specifies the audio reception capabilities of the local receiver. The parameter values correspond to the audio capabilities defined in the BAS table of Recommendation H.221: X_PHC_PV_CAPNON — No audio capability defined; X_PHC_PV_AUDIO_711_A — Audio decoding G.711, A-law; X_PHC_PV_AUDIO_711_U — Audio decoding G.711, μ-law; X_PHC_PV_AUDIO_722_64 — Audio decoding G.722 (mode 1) and G.711; X_PHC_PV_AUDIO_722_48 — Audio decoding G.722 (modes 1, 2, 3) and G.711; X_PHC_PV_AUDIO_16K — Audio decoding G.728 and G.711; X_PHC_PV_AUDIO_ISO — Audio decoding ISO standard at all rates up to 384 kbit/s; It is possible to select more than one of the audio capabilities listed above. In this case the parameter values have to be combined using a bit-wise OR. Each parameter value sets one bit position of the X_PHC_P_LCAP_AUDIO parameter.

Table I.5-9/T.180

Parameter name	X_PHC_P_LCAP_VIDEO
Type of value	long
Legal values	X_PHC_PV_CAPNON X_PHC_PV_VIDEO_QCIF X_PHC_PV_VIDEO_CIF X_PHC_PV_VIDEO_PINV1 X_PHC_PV_VIDEO_PINV2 X_PHC_PV_VIDEO_PINV3 X_PHC_PV_VIDEO_PINV4 X_PHC_PV_VIDEO_IMP X_PHC_PV_VIDEO_ISO X_PHC_PV_VIDEO_AVISO
Default value	The default value is defined by the value of the X_PHC_O_LCAP_VIDEO option.
Description	The parameter specifies the video reception capabilities of the local receiver. The parameter values correspond to the video capabilities defined in the BAS table of Recommendation H.221: X_PHC_PV_CAPNON — No video capability defined; X_PHC_PV_VIDEO_QCIF — Video decoding of QCIF only; X_PHC_PV_VIDEO_CIF — Video decoding of CIF and QCIF; X_PHC_PV_VIDEO_PINV1 — Minimum picture interval of 1/29.97; X_PHC_PV_VIDEO_PINV2 — Minimum picture interval of 2/29.97; X_PHC_PV_VIDEO_PINV3 — Minimum picture interval of 3/29.97; X_PHC_PV_VIDEO_PINV4 — Minimum picture interval of 4/29.97; X_PHC_PV_VIDEO_IMP — Improved video algorithm (for future use only!); X_PHC_PV_VIDEO_ISO — Video decoding to ISO standard. X_PHC_PV_VIDEO_AVISO — Video decoding composite audio/video signal to ISO standard. It is possible to select more than one of the video capabilities listed above. In this case the parameter values have to be combined using a bit-wise OR. Each parameter value sets one bit position of the X_PHC_P_LCAP_VIDEO parameter.

Table I.5-10/T.180

Parameter name	X_PHC_P_LCAP_DATA
Type of value	long
Legal values	X_PHC_PV_CAPNON X_PHC_PV_DATA_VAR X_PHC_PV_DATA_300 X_PHC_PV_DATA_1200 X_PHC_PV_DATA_4800 X_PHC_PV_DATA_6400 X_PHC_PV_DATA_8000 X_PHC_PV_DATA_9600 X_PHC_PV_DATA_14400 X_PHC_PV_DATA_16000 X_PHC_PV_DATA_24000 X_PHC_PV_DATA_32000 X_PHC_PV_DATA_40000 X_PHC_PV_DATA_48000 X_PHC_PV_DATA_56000 X_PHC_PV_DATA_62400 X_PHC_PV_DATA_64000 X_PHC_PV_DATA_MLP4000 X_PHC_PV_DATA_MLP6400 X_PHC_PV_DATA_VARMLP
Default value	The default value is defined by the value of the X_PHC_O_LCAP_DATA option.
Description	The parameter specifies the Low-Speed Data (LSD) reception capabilities of the local receiver. The parameter values correspond to the LSD capabilities defined in the BAS table of Recommendation H.221: X_PHC_PV_CAPNON No LSD capability defined; X_PHC_PV_DATA_VAR Can accept variable rate of LSD; X_PHC_PV_DATA_300 Can accept LSD at 300 bit/s; X_PHC_PV_DATA_1200 Can accept LSD at 1200 bit/s; X_PHC_PV_DATA_4800 Can accept LSD at 4800 bit/s; X_PHC_PV_DATA_6400 Can accept LSD at 6400 bit/s; X_PHC_PV_DATA_8000 Can accept LSD at 8000 bit/s; X_PHC_PV_DATA_9600 Can accept LSD at 9600 bit/s; X_PHC_PV_DATA_14400 Can accept LSD at 14 400 bit/s; X_PHC_PV_DATA_16000 Can accept LSD at 16 000 bit/s; X_PHC_PV_DATA_24000 Can accept LSD at 24 000 bit/s; X_PHC_PV_DATA_32000 Can accept LSD at 32 000 bit/s; X_PHC_PV_DATA_40000 Can accept LSD at 40 000 bit/s; X_PHC_PV_DATA_48000 Can accept LSD at 48 000 bit/s; X_PHC_PV_DATA_56000 Can accept LSD at 56 000 bit/s; X_PHC_PV_DATA_62400 Can accept LSD at 62 400 bit/s; X_PHC_PV_DATA_64000 Can accept LSD at 64 000 bit/s; X_PHC_PV_DATA_MLP4000 Can accept MLP at 4 kbit/s in the SC; X_PHC_PV_DATA_MLP6400 Can accept MLP at 6.4 kbit/s in the SC; X_PHC_PV_DATA_VARMLP Can accept MLP at up to 64 kbit/s in the IC. It is possible to select more then one of the low-speed data capabilities listed above. In this case the parameter values have to be combined using a bit-wise OR. Each parameter value sets one bit position of the X_PHC_P_LCAP_DATA parameter.

Table I.5-11/T.180

Parameter name	X_PHC_P_LCAP_HDATA
Type of value	long
Legal values	X_PHC_PV_CAPNON X_PHC_PV_HDATA_64 X_PHC_PV_HDATA_128 X_PHC_PV_HDATA_192 X_PHC_PV_HDATA_256 X_PHC_PV_HDATA_320 X_PHC_PV_HDATA_384 X_PHC_PV_HDATA_512 X_PHC_PV_HDATA_768 X_PHC_PV_HDATA_1152 X_PHC_PV_HDATA_1536 X_PHC_PV_HDATA_VAR X_PHC_PV_HDATA_MLP62 X_PHC_PV_HDATA_MLP64 X_PHC_PV_HDATA_MLP128 X_PHC_PV_HDATA_MLP192 X_PHC_PV_HDATA_MLP256 X_PHC_PV_HDATA_MLP320 X_PHC_PV_HDATA_MLP384 X_PHC_PV_HDATA_VARMLP
Default value	The default value is defined by the value of the X_PHC_O_LCAP_HDATA option.
Description	The parameter specifies the high-speed data reception capabilities of the local receiver. The parameter values correspond to the HSD capabilities defined in the BAS table of Recommendation H.221: X_PHC_PV_CAPNON No HSD capability defined; X_PHC_PV_HDATA_64 Can accept HSD at 64 kbit/s; X_PHC_PV_HDATA_128 Can accept HSD at 128 kbit/s; X_PHC_PV_HDATA_192 Can accept HSD at 192 kbit/s; X_PHC_PV_HDATA_256 Can accept HSD at 256 kbit/s; X_PHC_PV_HDATA_320 Can accept HSD at 320 kbit/s; X_PHC_PV_HDATA_384 Can accept HSD at 384 kbit/s; X_PHC_PV_HDATA_512 Can accept HSD at 512 kbit/s (for future use only); X_PHC_PV_HDATA_768 Can accept HSD at 768 kbit/s (for future use only); X_PHC_PV_HDATA_1152 Can accept HSD at 1152 kbit/s (for future use only); X_PHC_PV_HDATA_1536 Can accept HSD at 1536 kbit/s (for future use only); X_PHC_PV_HDATA_VAR Can accept variable rate of HSD; X_PHC_PV_HDATA_MLP62 Can accept MLP at 62.4 kbit/s; X_PHC_PV_HDATA_MLP64 Can accept MLP at 64 kbit/s; X_PHC_PV_HDATA_MLP128 Can accept MLP at 128 kbit/s; X_PHC_PV_HDATA_MLP192 Can accept MLP at 192 kbit/s; X_PHC_PV_HDATA_MLP256 Can accept MLP at 256 kbit/s; X_PHC_PV_HDATA_MLP320 Can accept MLP at 320 kbit/s; X_PHC_PV_HDATA_MLP384 Can accept MLP at 384 kbit/s; X_PHC_PV_HDATA_VARMLP Can accept MLP at variable rate. It is possible to select more then one of the high-speed data capabilities listed above. In this case the parameter values have to be combined using a bit-wise OR. Each parameter value sets one bit position of the X_PHC_P_LCAP_HDATA parameter.

Table I.5-12/T.180

Parameter name	X_PHC_P_LCAP_TFRATE
Type of value	long
Legal values	X_PHC_PV_CAPNON X_PHC_PV_TFRATE_128 X_PHC_PV_TFRATE_192 X_PHC_PV_TFRATE_256 X_PHC_PV_TFRATE_512 X_PHC_PV_TFRATE_768 X_PHC_PV_TFRATE_1152 X_PHC_PV_TFRATE_1472
Default value	The default value is defined by the value of the X_PHC_O_LCAP_TFRATE option.
Description	The parameter specifies the transfer rate reception capabilities of the local receiver. The parameter values correspond to the transfer rate capabilities defined in the BAS table of Recommendation H.221: X_PHC_PV_CAPNON No transfer rate capability defined; X_PHC_PV_TFRATE_128 Capable of accepting transfer rate 128 kbit/s; X_PHC_PV_TFRATE_192 Capable of accepting transfer rate 192 kbit/s; X_PHC_PV_TFRATE_256 Capable of accepting transfer rate 256 kbit/s; X_PHC_PV_TFRATE_512 Capable of accepting transfer rate 512 kbit/s; X_PHC_PV_TFRATE_768 Capable of accepting transfer rate 768 kbit/s; X_PHC_PV_TFRATE_1152 Capable of accepting transfer rate 1152 kbit/s; X_PHC_PV_TFRATE_1472 Capable of accepting transfer rate 1472 kbit/s. It is possible to select more then one of the transfer rate capabilities listed above. In this case the parameter values have to be combined using a bit-wise OR. Each parameter value sets one bit position of the X_PHC_P_LCAP_TFRATE parameter.

Table I.5-13/T.180

Parameter name	X_PHC_P_LCAP_TFLINES
Type of value	long
Legal values	X_PHC_PV_TFLINES_1B X_PHC_PV_TFLINES_2B X_PHC_PV_TFLINES_3B X_PHC_PV_TFLINES_4B X_PHC_PV_TFLINES_5B X_PHC_PV_TFLINES_6B X_PHC_PV_TFLINES_1H X_PHC_PV_TFLINES_2H X_PHC_PV_TFLINES_3H X_PHC_PV_TFLINES_4H X_PHC_PV_TFLINES_5H X_PHC_PV_TFLINES_H11 X_PHC_PV_TFLINES_H12
Default value	The default value is defined by the value of the X_PHC_O_LCAP_TFLINES option.
Description	The parameter specifies the transfer line reception capabilities of the local receiver. The parameter values correspond to the transfer line capabilities defined in the BAS table of Recommendation H.221: X_PHC_PV_TFLINES_1B Can handle AV data on one 64 kbit/s channel only; X_PHC_PV_TFLINES_2B Can handle AV data on one or two 64 kbit/s channels; X_PHC_PV_TFLINES_3B Can handle AV data on one to three 64 kbit/s channels; X_PHC_PV_TFLINES_4B Can handle AV data on one to four 64 kbit/s channels; X_PHC_PV_TFLINES_5B Can handle AV data on one to five 64 kbit/s channels; X_PHC_PV_TFLINES_6B Can handle AV data on one to six 64 kbit/s channels; X_PHC_PV_TFLINES_1H Can handle AV data on one 384 kbit/s channel; X_PHC_PV_TFLINES_2H Can handle AV data on one or two 384 kbit/s channels; X_PHC_PV_TFLINES_3H Can handle AV data on one to three 384 kbit/s channels; X_PHC_PV_TFLINES_4H Can handle AV data on one to four 384 kbit/s channels; X_PHC_PV_TFLINES_5H Can handle AV data on one to five 384 kbit/s channels; X_PHC_PV_TFLINES_H11 Can handle AV data on a 1536 kbit/s channel; X_PHC_PV_TFLINES_H12 Can handle AV data on a 1920 kbit/s channel. The selected parameter value defines the maximum number of transfer lines which can be handled by the local receiver. According to Recommendation H.242 the transfer lines capability has an explicit meaning (the ability to handle as many channels as indicated) but also an implicit meaning. The calling terminal implicitly signals its intention to establish the indicated number of channels and the called terminal implicitly signals its intention to accept the indicated number of connections.

Table I.5-14/T.180

Parameter name	X_PHC_P_LCAP_MISC
Type of value	long
Legal values	X_PHC_PV_CAPNON X_PHC_PV_MISC_ENCR X_PHC_PV_MISC_ESC X_PHC_PV_MISC_MBE X_PHC_PV_MISC_RESTRICT X_PHC_PV_MISC_6BHCOMP
Default value	The default value is defined by the value of the X_PHC_O_LCAP_MISC option.
Description	The parameter specifies the miscellaneous reception capabilities of the local receiver. The parameter values correspond to the capabilities defined in the BAS table of Recommendation H.221: X_PHC_PV_CAPNON — No capability of this type defined; X_PHC_PV_MISC_ENCR — Capable of handling the encryption on the ECS channel; X_PHC_PV_MISC_ESC — Capable of accepting non-zero call/family ESC codes; X_PHC_PV_MISC_MBE — Capable of handling multiple-byte extension messages; X_PHC_PV_MISC_RESTRICT — Restricted mode: can work only at $p \times 56$ kbit/s; X_PHC_PV_MISC_6BHCOMP — Capable of providing compatibility between 6B and H_0 terminals. It is possible to select more than one of the miscellaneous capabilities listed above. In this case the parameter values have to be combined using a bit-wise OR. Each parameter value sets one bit position of the X_PHC_P_LCAP_MISC parameter.

Table I.5-15/T.180

Parameter name	X_PHC_P_LCAP_DATAPPL
Type of value	long
Legal values	X_PHC_PV_CAPNON X_PHC_PV_APPL_STILLPIC_LSD X_PHC_PV_APPL_STILLPIC_HSD X_PHC_PV_APPL_STILLPIC_SPATIAL X_PHC_PV_APPL_STILLPIC_PROG X_PHC_PV_APPL_STILLPIC_ARITH X_PHC_PV_APPL_STILLIMAGE X_PHC_PV_APPL_CURSORDATA X_PHC_PV_APPL_FAX3 X_PHC_PV_APPL_FAX4 X_PHC_PV_APPL_V120_LSD X_PHC_PV_APPL_V120_HSD

Table I.5-15/T.180 *(concluded)*

Default value	The default value is defined by the value of the X_PHC_O_LCAP_DATAPPL option.
Description	The parameter specifies the reception application capabilities within LSD/HSD channels of the local receiver. The parameter values correspond to the capabilities defined in the BAS table of Recommendation H.221: X_PHC_PV_CAPNON — No capability of this type defined; X_PHC_PV_APPL_STILLPIC_LSD — Capable of the ISO still picture baseline mode on LSD; X_PHC_PV_APPL_STILLPIC_HSD — Capable of the ISO still picture baseline mode on HSD; X_PHC_PV_APPL_STILLPIC_SPATIAL — Capable of the ISO still picture and spatial modes; X_PHC_PV_APPL_STILLPIC_PROG — Capable of the ISO still picture and progressive modes; X_PHC_PV_APPL_STILLPIC_ARITH — Capable of the ISO still picture and arithmetic modes; X_PHC_PV_APPL_STILLIMAGE — Capable of still image encoded according to Recommendation H.261; X_PHC_PV_APPL_CURSORDATA — Can handle graphic cursor data; X_PHC_PV_APPL_FAX3 — Capable receiving Fax group 3; X_PHC_PV_APPL_FAX4 — Capable receiving Fax group 4; X_PHC_PV_APPL_V120_LSD — Capable of V.120 terminal adaptation within LSD; X_PHC_PV_APPL_V120_HSD — Capable of V.120 terminal adaptation within HSD. It is possible to select more than one of the application capabilities listed above. In this case the parameter values have to be combined using a bit-wise OR. Each parameter value sets one bit position of the X_PHC_P_LCAP_DATAPPL parameter.

Table I.5-16/T.180

Parameter name	X_PHC_P_LCAP_NONSTD
Type of value	unsigned char [ ], 5 ... 255 bytes or no value at all to revoke a previously specified non-standard capability
Legal values	a sequence of bytes not longer than X_PHC_C_MAX_NONSTD
Default value	none
Description	The parameter specifies the non-standard reception capabilities of the local receiver. The maximum length for a non-CCITT capabilities message is 255 bytes. The minimum length is 5 bytes. [byte 0] Country code according to Recommendation T.35; [byte 1] Country code; [byte 2, 3] Manufacturer code (e.g. company 4711); [byte 4-n] e.g. terminal type identity. The first two bytes define the country code and the next two bytes specify the terminal manufacturer code. The first byte of the country code is set according to Recommendation T.35; the second byte and bytes 3 to 4 of the manufacturer code are assigned nationally. Note that due to the effective bandwidth of the BAS (0.4 kbit/s or 50 bytes per second), the transmission time for a capability exchange including non-standard capabilities of maximum length may be more than 5 seconds.

Table I.5-17/T.180

Parameter name	X_PHC_P_FALLBACK
Type of value	unsigned long
Legal values	PV_TRUE PV_FALSE
Default value	PV_TRUE
Description	The parameter specifies the fallback feature of the local terminal. The X_PHC_P_FALL_BACK parameter is a user optional service parameter of the Connect Request service element. It enables/disables the automatic retrial if the initial connection establishment with service "video phone" failed. In the retrial the service "phone" is used what usually allows to establish a connection to a **non** video phone terminal, e.g. an ordinary ISDN phone. If the retrial is successful, the audio mode G.711 unframed is indicated in the init-complete. If the retrial is not successful, the connect request fails and a negative connect confirmation is passed to the user. If X_PHC_P_FALL_BACK is set to PV_FALSE the automatic "phone" retrial is omitted and a connect request will fail if the "video phone" call is not accepted.

I.5.7.2.3.2 Remote capability parameters

Table I.5-18/T.180

Parameter name	X_PHC_P_RCAP_AUDIO
Type of value	long
Legal values	X_PHC_PV_CAPNON X_PHC_PV_AUDIO_711_A X_PHC_PV_AUDIO_711_U X_PHC_PV_AUDIO_722_64 X_PHC_PV_AUDIO_722_48 X_PHC_PV_AUDIO_16K X_PHC_PV_AUDIO_ISO
Default value	none
Description	The parameter specifies the audio reception capabilities of the remote receiver. The parameter values are described above; see parameter X_PHC_P_LCAP_AUDIO. The value X_PHC_PV_CAPNON indicates that no value is set by the remote terminal.

Table I.5-19/T.180

Parameter name	X_PHC_P_RCAP_VIDEO
Type of value	long
Legal values	X_PHC_PV_CAPNON X_PHC_PV_VIDEO_QCIF X_PHC_PV_VIDEO_CIF X_PHC_PV_VIDEO_PINV1 X_PHC_PV_VIDEO_PINV2 X_PHC_PV_VIDEO_PINV3 X_PHC_PV_VIDEO_PINV4 X_PHC_PV_VIDEO_IMP X_PHC_PV_VIDEO_ISO X_PHC_PV_VIDEO_AVISO
Default value	none
Description	The parameter specifies the video reception capabilities of the remote receiver. The parameter values are described above; see parameter X_PHC_P_LCAP_VIDEO. The value X_PHC_PV_CAPNON indicates that no value is set by the remote terminal.

Table I.5-20/T.180

Parameter name	X_PHC_P_RCAP_DATA
Type of value	long
Legal values	X_PHC_PV_CAPNON X_PHC_PV_DATA_VAR X_PHC_PV_DATA_300 X_PHC_PV_DATA_1200 X_PHC_PV_DATA_4800 X_PHC_PV_DATA_6400 X_PHC_PV_DATA_8000 X_PHC_PV_DATA_9600 X_PHC_PV_DATA_14400 X_PHC_PV_DATA_16000 X_PHC_PV_DATA_24000 X_PHC_PV_DATA_32000 X_PHC_PV_DATA_40000 X_PHC_PV_DATA_48000 X_PHC_PV_DATA_56000 X_PHC_PV_DATA_62400 X_PHC_PV_DATA_64000 X_PHC_PV_DATA_MLP4000 X_PHC_PV_DATA_MLP6400 X_PHC_PV_DATA_VARMLP
Default value	none
Description	The parameter specifies the low-speed data reception capabilities of the remote receiver. The parameter values are described above; see parameter X_PHC_P_LCAP_DATA. The value X_PHC_PV_CAPNON indicates that no value is set by the remote terminal.

Table I.5-21/T.180

Parameter name	X_PHC_P_RCAP_HDATA
Type of value	long
Legal values	X_PHC_PV_CAPNON X_PHC_PV_HDATA_64 X_PHC_PV_HDATA_128 X_PHC_PV_HDATA_192 X_PHC_PV_HDATA_256 X_PHC_PV_HDATA_320 X_PHC_PV_HDATA_384 X_PHC_PV_HDATA_512 X_PHC_PV_HDATA_768 X_PHC_PV_HDATA_1152 X_PHC_PV_HDATA_1536 X_PHC_PV_HDATA_VAR X_PHC_PV_HDATA_MLP62 X_PHC_PV_HDATA_MLP64 X_PHC_PV_HDATA_MLP128 X_PHC_PV_HDATA_MLP192 X_PHC_PV_HDATA_MLP256 X_PHC_PV_HDATA_MLP320 X_PHC_PV_HDATA_MLP384 X_PHC_PV_HDATA_VARMLP
Default value	none
Description	The parameter specifies the high-speed data reception capabilities of the remote receiver. The parameter values are described above; see parameter X_PHC_P_LCAP_HDATA. The value X_PHC_PV_CAPNON indicates that no value is set by the remote terminal.

Table I.5-22/T.180

Parameter name	X_PHC_P_RCAP_TFRATE
Type of value	long
Legal values	X_PHC_PV_CAPNON X_PHC_PV_TFRATE_128 X_PHC_PV_TFRATE_192 X_PHC_PV_TFRATE_256 X_PHC_PV_TFRATE_512 X_PHC_PV_TFRATE_768 X_PHC_PV_TFRATE_1152 X_PHC_PV_TFRATE_1472
Default value	none
Description	The parameter specifies the transfer rate reception capabilities of the remote receiver. The parameter values are described above; see parameter X_PHC_P_LCAP_TFRATE. The value X_PHC_PV_CAPNON indicates that no value is set by the remote terminal.

Table I.5-23/T.180

Parameter name	X_PHC_P_RCAP_TFLINES
Type of value	long
Legal values	X_PHC_PV_TFLINES_1B X_PHC_PV_TFLINES_2B X_PHC_PV_TFLINES_3B X_PHC_PV_TFLINES_4B X_PHC_PV_TFLINES_5B X_PHC_PV_TFLINES_6B X_PHC_PV_TFLINES_1H X_PHC_PV_TFLINES_2H X_PHC_PV_TFLINES_3H X_PHC_PV_TFLINES_4H X_PHC_PV_TFLINES_5H X_PHC_PV_TFLINES_H11 X_PHC_PV_TFLINES_H12
Default value	none
Description	The parameter specifies the transfer line reception capabilities of the remote receiver. The parameter values are described above; see parameter X_PHC_P_LCAP_TFLINES.

Table I.5-24/T.180

Parameter name	X_PHC_P_RCAP_MISC
Type of value	long
Legal values	X_PHC_PV_CAPNON X_PHC_PV_MISC_ENCR X_PHC_PV_MISC_ESC X_PHC_PV_MISC_MBE X_PHC_PV_MISC_RESTRICT X_PHC_PV_MISC_6BHCOMP
Default value	none
Description	The parameter specifies the miscellaneous reception capabilities of the remote receiver. The parameter values are described above; see parameter X_PHC_P_LCAP_MISC. The value X_PHC_PV_CAPNON indicates that no value is set by the remote terminal.

Table I.5-25/T.180

Parameter name	X_PHC_P_RCAP_DATAPPL
Type of value	long
Legal values	X_PHC_PV_CAPNON X_PHC_PV_APPL_STILLPIC_LSD X_PHC_PV_APPL_STILLPIC_HSD X_PHC_PV_APPL_STILLPIC_SPATIAL X_PHC_PV_APPL_STILLPIC_PROG X_PHC_PV_APPL_STILLPIC_ARITH X_PHC_PV_APPL_STILLIMAGE X_PHC_PV_APPL_CURSORDATA X_PHC_PV_APPL_FAX3 X_PHC_PV_APPL_FAX4 X_PHC_PV_APPL_V120_LSD X_PHC_PV_APPL_V120_HSD
Default value	none
Description	The parameter specifies the reception application capabilities within LSD/HSD channels of the remote receiver. The parameter values are described above; see parameter X_PHC_P_LCAP_DATAPPL. The value X_PHC_PV_CAPNON indicates that no value is set by the remote terminal.

Table I.5-26/T.180

Parameter name	X_PHC_P_RCAP_NONSTD
Type of value	unsigned char [ ], 5 ... 255 bytes
Legal values	a sequence of bytes not longer than X_PHC_C_MAX_NONSTD
Default value	none
Description	The parameter specifies the non-standard reception capabilities of the remote receiver. The parameter values are described above; see parameter X_PHC_P_LCAP_NONSTD.

I.5.7.3 Services in the connected state

The connected state of an audiovisual communication is entered with the reception of the init-complete (X_PHC_SP_INIT_COMPL) service primitive. There are four services available that can be used in the connected state to control the communication:

- the Capability Exchange service to initiate a capability exchange with the remote terminal, e.g. to signal a different own capability set;
- the Mode Switch service to change the currently used communication mode, e.g. to open or close a low speed data channel;
- the Data Transfer service to send and receive data on a previously opened data channel;
- the Control and Indication (C&I) service to send BAS commands to the remote terminal or receive BAS commands sent by the remote terminal.

The capability exchange, the mode switch, and the C&I services all use the BAS for transmission. Therefore, requests from these services have to be scheduled so that the BAS is used by only one service at a time. The (effective) transfer rate available in the BAS is 0.4 kbit/s, which is equal to 50 bytes per second; thus the time needed for transmission of a command must not be neglected. The

longest possible non-standard command, for example, comprises 256 bytes, and the transmission in the BAS would take a little bit more than five seconds. During this time the BAS is occupied and cannot be used to transmit another request.

To avoid that the BAS is overrun, a flow control mechanism is used. After each request concerning the BAS, the application has to wait for the corresponding confirmation before the next BAS request is passed to the service provider. This does not only apply within a service but also between the three services using the BAS. If, for example, the application just requested a capability exchange, no mode switch request and no transmit C&I request may be passed to the service provider until the capability exchange confirmation is received. For each request concerning the BAS, the service provider checks whether there is an outstanding confirmation and will reject the request and generate an error indication with cause CC_UNEXPECT if there is one.

I.5.7.3.1 Capability Exchange service

The Capability Exchange service is used to change one or more receive capabilities of the local terminal dynamically while the AV communication is active. The service elements capability exchange request and capability exchange confirmation are provided for this. They are defined as service primitives that are exchanged with the service provider through the XAPI function *x_sndsp()* and *x_rcvsp()*.

I.5.7.3.1.1 Service description

The **capability exchange request** primitive initiates a capability exchange sequence according to Recommendation H.242. It forces framing in both directions of transmission and the exchange of terminal capability codes with the peer entity. Either terminal may initiate the sequence, and there is no problem caused by both doing so simultaneously or nearly simultaneously. The terminal X which initiates the capability exchange first reinstates framing with the procedure defined in Recommendation H.242 and then transmits its current capability set. When the other terminal Y detects the incoming capability set, it begins transmission of its own set of capability codes. Note that the receiving terminal Y need not change its capability set in response to the new capability set of terminal Y.

The successful completion of the capability exchange sequence is indicated to the initiator with an **capability exchange confirmation** service primitive. It conveys two capability sets as output service parameters: the own capabilities (that have been included in the request) and the capabilities of the remote terminal. Both capability sets are complete, i.e. all parameters are present with one exception: the non-standard capability parameter is not present if no non-standard capabilities are specified locally or received from the remote terminal. Note that a capability parameter will take the value X_PHC_PV_CAPNON if no such capability is present. The application should store the remote capabilities and observe them as limits to avoid failures in later mode switch requests.

The capability exchange request requires the BAS for transmission and thus has to be coordinated with the other BAS requests (see I.5.7.3 above). No capability exchange request may be submitted while a capability exchange-, an add line-, a mode switch-, or a transfer C&I confirmation is outstanding.

The capability exchange sequence is supervised by a timer (value about 10s). If the timer expires without multiframe alignment, an error indication is generated with cause code CC_OTHER and diagnostic **X_PHC_DC_UNSYNC**. If multiframe alignment could be achieved, but, even with retrials, no complete capability set of the remote terminal was received within one timer period, an error indication with cause CC_OTHER and diagnostic **X_PHC_DC_NOCAPSET** is generated.

The specification of the own, local capabilities parameters in the capability exchange request service primitive is optional. For each parameter not specified in the capability exchange request service primitive, the protocol machine adds the currently used value to the transmitted capability set. If the XAPI user wants to revoke a capability from the currently used set, it depends on the capability parameter which value has to be specified in the capability exchange request:

- For a parameter that represents a group of capabilities and whose actual value is formed as the bit-wise OR combination of several defined values, X_PHC_PV_CAPNON has to be specified to revoke all capabilities of this group. To revoke only some of the present capabilities, the new value is formed by switching off the bits corresponding to the capabilities to be revoked.

  Example: Assume that the variable *curr_data_cap* holds the current value of low-speed data capabilities and that this value is equal to X_PHC_PV_DATA_1200 | X_PHC_PV_DATA_9600 | X_PHC_PV_DATA_14400.

  To revoke the capability to receive LSD at 1200 bit/s, the value *curr_data_cap* & ~X_PHC_PV_DATA_1200 has to be specified for parameter X_PHC_P_LCAP_DATA in a capability exchange request.

- For a parameter that takes exactly one value (and not the bit-wise OR combination of several defined values), the new value has to be specified to revoke or decrease capabilities.

  Example: To decrease the number of transfer lines supported from at most six B-channels to at most two B-channels, the value of the X_PHC_P_LCAP_TFLINES has to be changed from X_PHC_PV_TFLINES_6B to X_PHC_PV_TFLINES_2B in a capability exchange.

- To revoke non-standard capabilities, the parameter X_PHC_LCAP_NONSTD has to be specified in a capability exchange request without a value.

Hint: There is a way to find out the currently active capability sets: a capability exchange request without any parameter will leave all local capabilities untouched and initiate a capability exchange with the currently used set. The capability exchange confirmation will yield the complete own and remote capability set.

The service elements and their corresponding XAPI functions needed for Capability Exchange are described in Table I.5-27.

Table I.5-27/T.180 – Service elements and their corresponding XAPI functions for Capability Exchange

Service element	XAPI function	Service element identifier	Description
Capability Exchange Request	x_sndsp()	X_PHC_SP_CAP_EXCHANGE_Q	The Capability Exchange Request is passed to the provider to initiate a capability exchange sequence according to Recommendation H.242. The service element may be used by the active and the passive communication partner. The identifier of the corresponding service primitive is X_PHC_SP_CAP_EXCHANGE_Q.

Table I.5-27/T.180 – Service elements and their corresponding XAPI functions for Capability Exchange *(concluded)*

Service element	XAPI function	Service element identifier	Description
Capability Exchange Confirmation	x_rcvsp()	X_PHC_SP_CAP_EXCHANGE_C	The Capability Exchange Confirmation is passed to the initiator of the capability exchange request as acknowledgement of the command. The possibly changed local and remote capabilities are indicated to the user as service parameters. The identifier of the corresponding service primitive is X_PHC_SP_CAP_EXCHANGE_C.

I.5.7.3.1.2 Service parameters

Table I.5-28 specifies the parameters of the Capability Exchange service.

Table I.5-28/T.180 – Parameters of the Capability Exchange service

Parameter	Capability Exchange service	
	Request	Confirmation
X_PHC_P_LCAP_AUDIO	U	M
X_PHC_P_LCAP_VIDEO	U	M
X_PHC_P_LCAP_DATA	U	M
X_PHC_P_LCAP_HDATA	U	M
X_PHC_P_LCAP_TFRATE	U	M
X_PHC_P_LCAP_TFLINES	U	M
X_PHC_P_LCAP_MISC	U	M
X_PHC_P_LCAP_DATAPPL	U	M
X_PHC_P_LCAP_NONSTD	U	C
X_PHC_P_RCAP_AUDIO		M
X_PHC_P_RCAP_VIDEO		M
X_PHC_P_RCAP_DATA		M
X_PHC_P_RCAP_HDATA		M
X_PHC_P_RCAP_TFRATE		M
X_PHC_P_RCAP_TFLINES		M
X_PHC_P_RCAP_MISC		M
X_PHC_P_RCAP_DATAPPL		M
X_PHC_P_RCAP_NONSTD		C

The definitions of the capability parameters can be found above in I.1.7.2, "Connection Establishment service".

I.5.7.3.2 Mode Switch service

The Mode Switch service enables the application program to dynamically switch the mode of an active AV communication or increase/decrease the number of connected channels in a multichannel connection. The service can be used at any time during a communication, after the mode initialization procedure has been completed, i.e. after an X_PHC_SP_INIT_COMPL has been received. The mode switching sequence as described in Recommendation H.242 is performed to change the used communication mode. Furthermore, the application is notified about mode switches which are initiated by the remote terminal.

I.5.7.3.2.1 Service description

There are service elements to increase or decrease the number of channels in a multichannel connection and there are service elements that initiate or indicate a change of the communication mode on the currently established channels.

In an AV communication the two transmission directions are, in principle, independent. There may be symmetrical modes where the mode in receive direction is the same as in transmit direction and asymmetrical ones where the two modes differ. For a conversational application, a videophone for example, a symmetrical mode would be best, whereas to operate a remote supervision camera would require an asymmetrical communication mode.

Thus, the communication mode as indicated to the application program distinguishes receive and transmission modes. For convenience, the global communication mode is split into several categories, e.g. video mode, audio mode, and data mode. For each category there is one service parameter giving the current receive mode and one for the transmit mode. The only exception is the number of established transfer lines (B-channels) because each channel provides a full-duplex connection that can be used in both directions simultaneously. There is only one parameter giving the number of currently connected lines.

After successful completion of the mode initialization procedure on the Initial Channel (IC), Additional Channels (ACs) may be added to the AV connection with an **add line request** to increase the total bandwidth. Only the active AV terminal that initiated the establishment of the IC [the terminal which called *x_conreq()*], is allowed to establish additional channels. Up to five ACs may be added simultaneously with one add line request. The number of channels to be established and the addresses to be used are specified as service parameters. The **add line confirmation,** which is generated as acknowledgment to an add line request, indicates the number of successfully established new channels. A value of 0 indicates that no new channels could be established. A mode switch indication (see below) will follow when the new channels are synchronized to the initial channel and included in the AV communication. The parameter X_PHC_P_MODE_TFLINES then indicates the now available bandwidth (number of channels). As default the new channels are used to increase the video bandwidth. No add line request may be submitted while an add line confirmation or any other confirmation from a BAS request (capability exchange-, mode switch-, or transfer C&I confirmation) is outstanding. Altogether, inclusive of the initial channel, no more channels will be established than those indicated by the value of the X_PHC_P_LCAP_TFLINES parameter. An add line request with this maximum number of channels already reached will be answered immediately with an add line confirmation of 0 channels.

The add line request may be used to establish additional B-channels in a multi-B-channel connection and to set up additional H_0-channels in a multi-H_0-channel connection. Only channels of the same type may be combined in a multichannel connection.

When establishing new channels, the service provider uses the addresses (phone numbers) in the same sequence as they are specified in the add line request. Each number is tried only once. The process stops when either the requested or maximum number of channels is reached or the specified

numbers are exhausted. Then the add line confirmation is generated which indicates the number of successful established channels and the corresponding addresses. Unused addresses are ignored. In the case of basic rate interfaces (S_0) one has to specify a number twice to set up two B-channels on the same interface. If an additional channel shall be established at the same interface as the initial channel, the number used in the *x_conreq()* has to be specified in the add line request too.

On the passive side, the service provider automatically accepts incoming calls for additional channels, as long as the maximum number of channels defined in X_PHC_P_LCAP_TFLINES is not yet reached. When the new channel is successfully synchronized with the initial channel, the passive application is informed about the new number of channels with a mode switch indication. As on the active side, the parameter X_PHC_P_MODE_TFLINES indicates the now available number of channels. Note that only the terminal that initiated establishment of the initial channel is allowed to establish additional channels. The service provider will generate an error indication with cause code CC_UNEXPECT if an add line request is submitted on the passive side of an AV connection.

With a **hang-up line request**, previously established additional channels may be disconnected again. A parameter specifies the number of channels to be closed. The hang-up line request implies a mode switch request that decreases the used transfer rate so that the specified number of channels becomes idle. These idle channels are then disconnected. Note that only additional channels may be closed with a hang-up line request; the initial channel has to be disconnected with *x_relreq()* or *x_snddis()*. As response to a hang-up line request, the application gets a mode switch indication in which the value of X_PHC_P_MODE_TFLINES is decremented by the number of closed channels. The hang-up line request may be used by either the active terminal or the passive terminal. If used on the passive side, it is advisable to decrease the value of X_PHC_P_LCAP_TFLINES accordingly in a following capability exchange, to prevent the active terminal from re-establishing the closed additional channels.

The **mode switch request** can be used by either terminal at any time after successful mode initialization (reception of X_PHC_SP_INIT_COMPL) to dynamically change the mode in an active AV communication. The mode switching sequence according to Recommendation H.242 is followed. If a mode switch is requested while the Codec is receiving or transmitting unframed, the appropriate frame reinstatement procedure is executed automatically prior to the mode switch to enforce a framed mode. The service provider generates a **mode switch confirmation** as response to a previous mode switch request. The new communication mode is indicated by the service parameters. The mode switch request requires the BAS for transmission and thus has to be coordinated with the other BAS requests (see I.5.7.3 above). No mode switch request may be submitted while a mode switch-, an add line-, a capability exchange-, or a transfer C&I confirmation is outstanding.

A mode switch request will fail:

- if one of the specified parameters has an invalid value. The service provider will generate an error indication with cause CC_BADVALUE in this case;
- if an inconsistent combination of audio, video, and data is specified, the service provider will generate an error indication with cause CC_OTHER and diagnostic **X_PHC_DC_BADCOMBI** in this case;
- if the requested mode is in conflict with the known receive and decode capabilities of the remote terminal. The service provider will generate an error indication with cause CC_OTHER and diagnostic **X_PHC_DC_CAPCONFLICT** in this case.

If a mode switch request fails, the current communication mode is retained.

The **mode switch indication** is generated by the service provider to inform the application about a mode switch initiated by the remote partner or an automatic mode switch triggered by a local condition. Usually no action is required on this indication.

Some notes to communication mode switching:

NOTE 1 – An AV terminal always has to switch its receiver/decoder according to the BAS commands received from the other end terminal, which itself has to transmit data as specified by the previously sent BAS commands. This, of course, applies for both directions. If the own transmitting mode is changed, the remote terminal has to change its receive mode accordingly. Conversely, if the remote terminal initiates a mode switch, the service provider automatically switches the local receiver to the new mode and indicates the mode switch to the application. No action is required on this indication.

NOTE 2 – The own transmitting mode may be changed only according to the known receive and decode capabilities of the remote terminal. The service provider checks whether the new mode specified in a mode switch request is within the known capabilities of the remote terminal. The current mode will not be changed, if the specified new mode is in conflict with the remote capabilities for one category, even if the values specified in other categories are supported by the remote terminal. In this case, the mode switch confirmation generated as response to the mode switch request, will indicate an unchanged communication mode.

NOTE 3 – The service provider will automatically initiate a mode switch if the remote terminal changes its capabilities so that the current mode is no longer receivable/decodable. The new transmission mode is selected by the service provider in accordance with the new capability set of the remote terminal. The new mode is indicated to the application. No action is required on this indication.

NOTE 4 – It is not possible to request a change of the own receive/decoding mode because it always has to correspond to the transmission/encoding mode of the remote terminal. (See also Note 1.) However, there is an indirect way to control the own receiver: A new, different capability set is indicated to the remote terminal with a capability exchange request. If the current reception mode is then no longer allowed with the new capability set, the remote terminal has to switch to another transmission mode and the own receiver has to follow this. But this only works if the own capability set is decreased. If the capability set is increased, i.e. previously unsupported capabilities are signalled, the remote terminal is not enforced to take advantage of the new capabilities and change its transmission mode. The currently used transmission mode may be kept as long as it corresponds to the receiving capabilities of the other side.

NOTE 5 – If the video bandwidth would become zero as the result of a mode switch request, the service provider automatically sends a video command "Freeze picture Request" (VCF) to the remote terminal before the mode switch is executed. The VCF will freeze the picture at the remote terminal until a picture release signal is sent or a time-out period of at least six seconds has expired. The VCF is not automatically repeated by the service provider. If the application wishes to continue the freezing of the picture, it should send VCF repeatedly with an appropriate period.

The service elements and their corresponding XAPI functions needed for Mode Switch are described in Table I.5-29.

Table I.5-29/T.180 – Service elements and their corresponding XAPI functions for Mode Switch

Service element	XAPI function	Service element identifier	Description
Add Line Request	x_sndsp()	X_PHC_SP_ADD_LINE_Q	The Add Line Request primitive is passed to the provider to establish additional channels for a multi channel connection and include the new channels in the AV communication. The number of ACs to be established and the addresses to be used are specified as parameters. X_PHC_SP_ADD_LINE_Q is the identifier of the service primitive.
Add Line Confirmation	x_rcvsp()	X_PHC_SP_ADD_LINE_C	The Add Line Confirmation is passed to the application as a response to a previous add line request. The number of established new channels and the corresponding phone numbers are indicated to the application as output service parameters. X_PHC_SP_ADD_LINE_C is the identifier of the service primitive.
Hang-up Line Request	x_sndsp()	X_PHC_SP_HUP_LINE_Q	The Hang-up Line Request may be submitted to close some or all of the additional channels. The initial channel is not affected. X_PHC_SP_HUP_LINE_Q is the identifier of the service primitive.
Mode Switch Request	x_sndsp()	X_PHC_SP_MODE_SWITCH_Q	The Mode Switch Request primitive is passed to the provider to initiate a change of the current transmission mode. The new mode is defined by the service parameters. Only categories to be changed must be specified. All others keep their values. X_PHC_SP_MODE_SWITCH_Q is the identifier of the service primitive.

Table I.5-29/T.180 – Service elements and their corresponding XAPI functions for Mode Switch *(concluded)*

Service element	XAPI function	Service element identifier	Description
Mode Switch Confirmation	x_rcvsp()	X_PHC_SP_MODE_SWITCH_C	The Mode Switch Confirmation is passed to the application as a response to a previous mode switch request. The service parameters indicate the current receive and transmit mode. If mode switching succeeded, the transmit mode is equal to the one specified in the request. X_PHC_SP_MODE_SWITCH_C is the identifier of the service primitive.
Mode Switch Indication	x_rcvsp()	X_PHC_SP_MODE_SWITCH_I	The Mode Switch Indication is passed to the application when the communication mode was changed by the provider without request from the application. See Notes above for the situations where such automatically mode changes are necessary. The service parameters indicate the new receive and transmit mode. X_PHC_SP_MODE_SWITCH_I is the identifier of the service primitive.

NOTE 6 – The current communication mode is indicated to the application with:

1) the X_PHC_SP_INIT_COMPL primitive;

2) the X_PHC_SP_MODE_SWITCH_C; and

3) the X_PHC_SP_MODE_SWITCH_I.

The application may save this mode information internally, if it is required to know the current mode at any time. If the application hasn't done so, or the saved mode information has unfortunately been lost, there is a simple way to retrieve the current communication mode from the service provider: a mode switch request with no parameters will leave the current communication mode unchanged and will be answered immediately by the provider with a mode switch confirmation that indicates the current mode within the service parameters.

I.5.7.3.2.2 Service parameters

Table I.5-30 specifies the parameters of the Mode Switch service.

Table I.5-30/T.180 – Parameters of the Mode Switch service

Parameter	Mode Switch service element		
	Mode Switch Request	**Mode Switch Indication**	**Mode Switch Confirmation**
X_PHC_P_RMOD_AUDIO		M	M
X_PHC_P_RMOD_VIDEO		M	M
X_PHC_P_RMOD_DATA		M	M
X_PHC_P_RMOD_HDATA		M	M
X_PHC_P_RMOD_TFRATE		M	M
X_PHC_P_RMOD_SYNC		M	M
X_PHC_P_RMOD_DATAPPL		M	M
X_PHC_P_RMOD_MISC		M	M
X_PHC_P_TMOD_AUDIO	U	M	M
X_PHC_P_TMOD_VIDEO	U	M	M
X_PHC_P_TMOD_DATA	U	M	M
X_PHC_P_TMOD_HDATA	U	M	M
X_PHC_P_TMOD_TFRATE	U	M	M
X_PHC_P_TMOD_SYNC		M	M
X_PHC_P_TMOD_DATAPPL	U	M	M
X_PHC_P_TMOD_MISC	U	M	M
X_PHC_P_MODE_TFLINES	U	M	M
X_PHC_P_NUM_LINES	M	M	M
A_OUTBAND_ADR	M	C	

I.5.7.3.2.3 Receive mode parameters

Tables I.5-31 to I.5-49 describe the parameters for the Mode Switch service.

Table I.5-31/T.180

Parameter name	X_PHC_P_RMOD_AUDIO	
Type of value	long	
Legal values	X_PHC_PV_MDAU_OFF_U X_PHC_PV_MDAU_OFF_F X_PHC_PV_MDAU_G711A_U X_PHC_PV_MDAU_G711A_F X_PHC_PV_MDAU_G711U_U X_PHC_PV_MDAU_G711U_F X_PHC_PV_MDAU_G722_M1 X_PHC_PV_MDAU_G722_M2 X_PHC_PV_MDAU_G722_M3 X_PHC_PV_MDAU_ISO_64 X_PHC_PV_MDAU_ISO_128 X_PHC_PV_MDAU_ISO_192 X_PHC_PV_MDAU_ISO_256 X_PHC_PV_MDAU_ISO_384 X_PHC_PV_MDAU_G728	
Default value	none	
Description	This parameter is returned by the service provider in mode switch indications and confirmations. It indicates the current audio receive mode.	
	X_PHC_PV_MDAU_OFF_U	No audio signal and no framing in the I-channel;
	X_PHC_PV_MDAU_OFF_F	No audio signal, FAS and BAS in use;
	X_PHC_PV_MDAU_G711A_U	G.711 audio at 64 kbit/s, A-law, no framing;
	X_PHC_PV_MDAU_G711A_F and	G.711 audio at 56 kbit/s, A-law, FAS BAS in use;
	X_PHC_PV_MDAU_G711U_U	G.711 audio at 64 kbit/s, μ-law, no framing;
	X_PHC_PV_MDAU_G711U_F	G.711 audio at 56 kbit/s, μ-law, FAS and BAS in use;
	X_PHC_PV_MDAU_G722_M1	G.722 7 kHz audio at 64 kbit/s, no framing (mode 1);
	X_PHC_PV_MDAU_G722_M2	G.722 7 kHz audio at 56 kbit/s, no framing (mode 2);
	X_PHC_PV_MDAU_G722_M3	G.722 7 kHz audio at 48 kbit/s, no framing (mode 3);
	X_PHC_PV_MDAU_ISO_64	ISO standard audio at 64 kbit/s in time-slot 2 (TS2) of an H_0 or greater channel;
	X_PHC_PV_MDAU_ISO_128	ISO standard audio at 128 kbit/s in TS2 and TS3 of an H_0 or greater channel;
	X_PHC_PV_MDAU_ISO_192	ISO standard audio at 192 kbit/s in TS2 to TS4 of an H_0 or greater channel;
	X_PHC_PV_MDAU_ISO_256	ISO standard audio at 256 kbit/s in TS2 to TS5 of an H_0 or greater channel;
	X_PHC_PV_MDAU_ISO_384	ISO standard audio at 384 kbit/s in TS2 to TS7 of a channel greater than H_0;
	X_PHC_PV_MDAU_G728	G.728 audio at 16 kbit/s.

Table I.5-32/T.180

Parameter name	X_PHC_P_RMOD_VIDEO
Type of value	long
Legal values	X_PHC_PV_MDVI_OFF X_PHC_PV_MDVI_H261 X_PHC_PV_MDVI_ISO X_PHC_PV_MDVI_AVISO X_PHC_PV_MDVI_PINV1 X_PHC_PV_MDVI_PINV2 X_PHC_PV_MDVI_PINV3 X_PHC_PV_MDVI_PINV4
Default value	none
Description	This parameter is returned by the service provider in mode switch indications and confirmations. It indicates the current video receive mode. A legal value of the video mode parameter is either X_PHC_PV_MDVI_OFF or a combination of one video coding value, one video picture format value, and one video frame rate value. The bit-wise OR operator is used to combine the three values. X_PHC_PV_MDVI_OFF — Video switched off; **Video coding** X_PHC_PV_MDVI_H261 — Video is on, coding is according to Recommendation H.261. Video data occupies all capacity not otherwise allocated; X_PHC_PV_MDVI_ISO — Video is on, coding is according to ISO standard. Video data occupies all capacity not otherwise allocated; X_PHC_PV_MDVI_AVISO — ISO standard composite audio/video data occupy all capacity not otherwise allocated. **Video frame rate** X_PHC_PV_MDVI_PINV1 — Picture Interval is 1/29.75 s; X_PHC_PV_MDVI_PINV2 — Picture Interval is 2/29.75 s; X_PHC_PV_MDVI_PINV3 — Picture Interval is 3/29.75 s; X_PHC_PV_MDVI_PINV4 — Picture Interval is 4/29.75 s.

Table I.5-33/T.180

Parameter name	X_PHC_P_RMOD_DATA
Type of value	long
Legal values	X_PHC_PV_MDAT_LSDOFF X_PHC_PV_MDAT_LSD300 X_PHC_PV_MDAT_LSD1200 X_PHC_PV_MDAT_LSD4800 X_PHC_PV_MDAT_LSD6400 X_PHC_PV_MDAT_LSD8000 X_PHC_PV_MDAT_LSD9600 X_PHC_PV_MDAT_LSD14400 X_PHC_PV_MDAT_LSD16000 X_PHC_PV_MDAT_LSD24000 X_PHC_PV_MDAT_LSD32000 X_PHC_PV_MDAT_LSD40000 X_PHC_PV_MDAT_LSD48000 X_PHC_PV_MDAT_LSD56000 X_PHC_PV_MDAT_LSD62400 X_PHC_PV_MDAT_LSD64000 X_PHC_PV_MDAT_LSDVAR X_PHC_PV_MDAT_MLPOFF X_PHC_PV_MDAT_MLP4000 X_PHC_PV_MDAT_MLP6400 X_PHC_PV_MDAT_MLPVAR
Default value	none
Description	This parameter is returned by the service provider in mode switch indications and confirmations. It indicates the current LSD/MLP receive bit rate. For each rate the occupied bits of the I-channel are mentioned. LSD and MLP may be switched on simultaneously, but the used bits must not overlap. A legal value of the LSD/MLP mode parameter is the bit-wise OR combination of one LSD value and one MLP value. The combination of variable LSD and variable MLP is not allowed. LSD and MLP both switched off is indicated by the value (X_PHC_PV_MDAT_LSDOFF | X_PHC_PV_MDAT_MLPOFF). X_PHC_PV_MDAT_LSDOFF LSD switched off; X_PHC_PV_MDAT_LSD300 LSD on at fixed rate of 300 bit/s in SC, octets 38 to 40; X_PHC_PV_MDAT_LSD1200 LSD on at fixed rate of 1200 bit/s in SC, octets 29 to 40; X_PHC_PV_MDAT_LSD4800 LSD on at fixed rate of 4800 bit/s in SC, octets 33 to 80; X_PHC_PV_MDAT_LSD6400 LSD on at fixed rate of 6400 bit/s in SC, octets 17 to 80; X_PHC_PV_MDAT_LSD8000 LSD on at fixed rate of 8000 bit/s in bit 7; X_PHC_PV_MDAT_LSD9600 LSD on at fixed rate of 9600 bit/s in bit 7 and octets 25 to 40 of SC;

Table I.5-33/T.180 *(concluded)*

	X_PHC_PV_MDAT_LSD14400	LSD on at fixed rate of 1200 bit/s in bit 7 and octets 25 to 40 of SC;
	X_PHC_PV_MDAT_LSD16000	LSD on at fixed rate of 16 000 bit/s in bits 6 and 7;
	X_PHC_PV_MDAT_LSD24000	LSD on at fixed rate of 24 000 bit/s in bits 5 to 7;
	X_PHC_PV_MDAT_LSD32000	LSD on at fixed rate of 32 000 bit/s in bits 4 to 7;
	X_PHC_PV_MDAT_LSD40000	LSD on at fixed rate of 40 000 bit/s in bits 3 to 7;
	X_PHC_PV_MDAT_LSD48000	LSD on at fixed rate of 48 000 bit/s in bits 2 to 7;
	X_PHC_PV_MDAT_LSD56000	LSD on at fixed rate of 56 000 bit/s in bits 1 to 7;
	X_PHC_PV_MDAT_LSD62400	LSD on at fixed rate of 62 400 bit/s in bits 1 to 7 and octets 17 to 80 of SC;
	X_PHC_PV_MDAT_LSD64000	LSD on at fixed rate of 64 000 bit/s in bits 1 to 8, no framing;
	X_PHC_PV_MDAT_LSDVAR	LSD on at variable rate occupying all I-channel capacity not allocated for audio, FAS, BAS, ECS, and fixed-rate MLP;
	X_PHC_PV_MDAT_MLPOFF	MLP switched off;
	X_PHC_PV_MDAT_MLP4000	MLP on at fixed rate of 4000 bit/s in SC, octets 41 to 80;
	X_PHC_PV_MDAT_MLP6400	MLP on at fixed rate of 6400 bit/s in SC, octets 17 to 80;
	X_PHC_PV_MDAT_MLPVAR	MLP on at variable rate occupying all I-channel capacity not allocated for audio, FAS, BAS, ECS, and fixed rate LSD.

Table I.5-34/T.180

Parameter name	X_PHC_P_RMOD_HDATA
Type of value	long
Legal values	X_PHC_PV_MDHD_HSDOFF X_PHC_PV_MDHD_HSD64 X_PHC_PV_MDHD_HSD128 X_PHC_PV_MDHD_HSD192 X_PHC_PV_MDHD_HSD256 X_PHC_PV_MDHD_HSD320 X_PHC_PV_MDHD_HSD384 X_PHC_PV_MDHD_HMLPOFF X_PHC_PV_MDHD_HMLP62 X_PHC_PV_MDHD_HMLP128 X_PHC_PV_MDHD_HMLP192 X_PHC_PV_MDHD_HMLP256 X_PHC_PV_MDHD_HMLP320 X_PHC_PV_MDHD_HMLP384
Default value	none
Description	This parameter is returned by the service provider in mode switch indications and confirmations. It indicates the current HSD/H-MLP receive mode. For each mode the occupied time-slots of an H_0 or greater channel are mentioned. HSD and H-MLP may be switched on simultaneously, but the used time-slots must not overlap. A legal value of the HSD/H-MLP mode parameter is the bit-wise OR combination of one HSD value and one H-MLP value. HSD and H-MLP both switched off is indicated by the value (X_PHC_PV_MDHD_HSDOFF | X_PHC_PV_MDHD_HMLPOFF). NOTE – HSD/H-MLP commands are defined in Annex A/H.221. X_PHC_PV_MDHD_HSDOFF HSD switched off; X_PHC_PV_MDHD_HSD64 HSD on in highest numbered channel/time-slot; FAS and BAS are removed in the case of multiple B-channels; X_PHC_PV_MDHD_HSD128 HSD on in 2 highest-numbered time-slots of an H_0 or greater channel; X_PHC_PV_MDHD_HSD192 HSD on in 3 highest-numbered time-slots of an H_0 or greater channel; X_PHC_PV_MDHD_HSD256 HSD on in 4 highest-numbered time-slots of an H_0 or greater channel; X_PHC_PV_MDHD_HSD320 HSD on in 5 highest-numbered time-slots of an H_0 or greater channel; X_PHC_PV_MDHD_HSD384 HSD on in highest numbered H_0 channel, or 6 highest numbered time-slots of a greater channel;

Table I.5-34/T.180 *(concluded)*

Description	X_PHC_PV_MDHD_HMLPOFF	H-MLP switched off;
	X_PHC_PV_MDHD_HMLP62	H-MLP on at 62.4 kbit/s in the second B-channel, occupying all bits except FAS and BAS positions;
	X_PHC_PV_MDHD_HMLP64	H-MLP on at 64 kbit/s in time-slot 2 of an H_0 or greater channel;
	X_PHC_PV_MDHD_HMLP128	H-MLP on at 128 kbit/s in time-slots 2 and 3 of an H_0 or greater channel;
	X_PHC_PV_MDHD_HMLP192	H-MLP on at 192 kbit/s in time-slots 2 to 4 of an H_0 or greater channel;
	X_PHC_PV_MDHD_HMLP256	H-MLP on at 256 kbit/s in time-slots 2 to 5 of an H_0 or greater channel;
	X_PHC_PV_MDHD_HMLP320	H-MLP on at 320 kbit/s in time-slots 2 to 6 of an H_0 or greater channel;
	X_PHC_PV_MDHD_HMLP384	H-MLP on at 384 kbit/s in time-slots 2 to 7 of a channel greater than H_0.

Table I.5-35/T.180

Parameter name	X_PHC_P_RMOD_DATAPPL	
Type of value	long	
Legal values	X_PHC_PV_MDAP_LSDISOSP X_PHC_PV_MDAP_HSDISOSP X_PHC_PV_MDAP_LSDCURS X_PHC_PV_MDAP_LSDFAX X_PHC_PV_MDAP_HSDFAX X_PHC_PV_MDAP_LSDV120 X_PHC_PV_MDAP_HSDV120	
Default value	none	
Description	This parameter is returned by the service provider in mode switch indications and confirmations. It indicates to which application the data belong that are currently received as LSD rsp. HSD. A legal value of the data application parameter is either a single LSD value, a single HSD value, or the bit-wise OR combination of one LSD value and one HSD value. (Note that LSD and HSD may be activated simultaneously.)	
	X_PHC_PV_MDAP_LSDISOSP	ISO still picture switched on in LSD;
	X_PHC_PV_MDAP_HSDISOSP	ISO still picture switched on in HSD;
	X_PHC_PV_MDAP_LSDCURS	Graphics cursor data switched on in LSD;
	X_PHC_PV_MDAP_LSDFAX	FAX switched on in LSD;
	X_PHC_PV_MDAP_HSDFAX	FAX switched on in HSD;
	X_PHC_PV_MDAP_LSDV120	V.120 terminal adaptation switched on in LSD;
	X_PHC_PV_MDAP_HSDV120	V.120 terminal adaptation switched on in HSD.

Table I.5-36/T.180

Parameter name	X_PHC_P_RMOD_TFRATE
Type of value	long
Legal values	X_PHC_PV_MDTR_1X64 X_PHC_PV_MDTR_2X64 X_PHC_PV_MDTR_3X64 X_PHC_PV_MDTR_4X64 X_PHC_PV_MDTR_5X64 X_PHC_PV_MDTR_6X64 X_PHC_PV_MDTR_1X384 X_PHC_PV_MDTR_2X384 X_PHC_PV_MDTR_3X384 X_PHC_PV_MDTR_4X384 X_PHC_PV_MDTR_5X384 X_PHC_PV_MDTR_128 X_PHC_PV_MDTR_192 X_PHC_PV_MDTR_256 X_PHC_PV_MDTR_512 X_PHC_PV_MDTR_768 X_PHC_PV_MDTR_1152 X_PHC_PV_MDTR_1472 X_PHC_PV_MDTR_1536 X_PHC_PV_MDTR_1920
Default value	none
Description	This parameter is returned by the service provider in mode switch indications and confirmations. It indicates the current receive transfer rate. If the transfer rate is less than the connected capacity, the information occupies the lowest-numbered channel(s)/time-slot(s). X_PHC_PV_MDTR_1X64 Signal occupies one 64 kbit/s channel; X_PHC_PV_MDTR_2X64 Signal occupies two 64 kbit/s channels with FAS/BAS in each; X_PHC_PV_MDTR_3X64 Signal occupies three 64 kbit/s channels with FAS/BAS in each; X_PHC_PV_MDTR_4X64 Signal occupies four 64 kbit/s channels with FAS/BAS in each; X_PHC_PV_MDTR_5X64 Signal occupies five 64 kbit/s channels with FAS/BAS in each; X_PHC_PV_MDTR_6X64 Signal occupies six 64 kbit/s channels with FAS/BAS in each; X_PHC_PV_MDTR_1X384 Signal occupies one 384 kbit/s channels with FAS and BAS in the first 64 kbit/s time-slot. The effective channel may be the whole of an H_0 channel or the lowest-numbered time-slots of an H_{11} or an H_{12} channel; X_PHC_PV_MDTR_2X384 Signal occupies two 384 kbit/s channels with FAS/BAS in each; X_PHC_PV_MDTR_3X384 Signal occupies three 384 kbit/s channels with FAS/BAS in each; X_PHC_PV_MDTR_4X384 Signal occupies four 384 kbit/s channels with FAS/BAS in each; X_PHC_PV_MDTR_5X384 Signal occupies five 384 kbit/s channels with FAS/BAS in each;

Table I.5-36/T.180 *(concluded)*

	X_PHC_PV_MDTR_128	Signal occupies 128 kbit/s, with FAS and BAS in the first 64 kbit/s time-slot (TS1). The effective channel occupies the lowest-numbered time-slots of a channel with corresponding or higher capacity;
	X_PHC_PV_MDTR_192	Signal occupies 192 kbit/s, with FAS and BAS in TS1;
	X_PHC_PV_MDTR_256	Signal occupies 256 kbit/s, with FAS and BAS in TS1;
	X_PHC_PV_MDTR_512	Signal occupies 512 kbit/s, with FAS and BAS in TS1;
	X_PHC_PV_MDTR_768	Signal occupies 768 kbit/s, with FAS and BAS in TS1;
	X_PHC_PV_MDTR_1152	Signal occupies 1152 kbit/s, with FAS and BAS in TS1;
	X_PHC_PV_MDTR_1472	Signal occupies 1472 kbit/s, with FAS and BAS in TS1;
	X_PHC_PV_MDTR_1536	Signal occupies 1536 kbit/s, with FAS and BAS in TS1;
	X_PHC_PV_MDTR_1920	Signal occupies 1920 kbit/s, with FAS and BAS in TS1.

Table I.5-37/T.180

Parameter name	X_PHC_P_RMOD_SYNC	
Type of value	long	
Legal values	X_PHC_PV_MDSY_NONE X_PHC_PV_MDSY_ICHAN X_PHC_PV_MDSY_ACHAN	
Default value	none	
Description	This parameter is returned by the service provider in mode switch indications and confirmations. It indicates the current synchronization status in receive direction.	
	X_PHC_PV_MDSY_NONE	All incoming signals are received unframed;
	X_PHC_PV_MDSY_ICHAN	FAS and BAS are received in the I-channel;
	X_PHC_PV_MDSY_ACHAN	FAS and BAS are received in all additional channels and they are synchronized with the I-channel, i.e. multiframe alignment is validated.

Table I.5-38/T.180

Parameter name	X_PHC_P_RMOD_MISC
Type of value	long
Legal values	X_PHC_PV_MDMS_ECSACT X_PHC_PV_MDMS_6BH0COMP X_PHC_PV_MDMS_RESTRICT X_PHC_PV_MDMS_DERESTRICT
Default value	none
Description	This parameter is returned by the service provider in mode switch indications and confirmations. It indicates miscellaneous attributes currently active in receive direction. The attributes are independent and represented as one bit within the parameter value. If the bit is set to 1, the attribute is active; otherwise, it is inactive. The bit-wise AND operator can be used to check a single attribute from the parameter value. X_PHC_PV_MDMS_ECSACT — ECS channel active; X_PHC_PV_MDMS_6BH0COMP — 6B-H_0 compatibility mode is active. (For H_0 terminals only); X_PHC_PV_MDMS_RESTRICT — Communicating over a restricted network (Bit 7 of the I-channel is treated as SC and bit 8 is discarded in every other channel); X_PHC_PV_MDMS_DERESTRICT — Revert to unrestricted network operation (Bit 8 of the I-channel is treated as SC).

I.5.7.3.2.4 Transmit mode parameters

Table I.5-39/T.180

Parameter name	X_PHC_P_TMOD_AUDIO
Type of value	long
Legal values	X_PHC_PV_MDAU_OFF_U X_PHC_PV_MDAU_OFF_F X_PHC_PV_MDAU_G711A_U X_PHC_PV_MDAU_G711A_F X_PHC_PV_MDAU_G711U_U X_PHC_PV_MDAU_G711U_F X_PHC_PV_MDAU_G722_M1 X_PHC_PV_MDAU_G722_M2 X_PHC_PV_MDAU_G722_M3 X_PHC_PV_MDAU_ISO_64 X_PHC_PV_MDAU_ISO_128 X_PHC_PV_MDAU_ISO_192 X_PHC_PV_MDAU_ISO_256 X_PHC_PV_MDAU_ISO_384 X_PHC_PV_MDAU_G728

Table I.5-39/T.180 *(concluded)*

Default value	none
Description	This parameter may be specified in a mode switch request to select the audio mode to be used in transmit direction. The parameter is returned by the service provider in mode switch indications and confirmations. There it indicates the currently active audio mode in transmit direction. See the corresponding receive mode parameter X_PHC_P_RMOD_AUDIO for a description of the defined values.

Table I.5-40/T.180

Parameter name	X_PHC_P_TMOD_VIDEO
Type of value	long
Legal values	X_PHC_PV_MDVI_OFF X_PHC_PV_MDVI_H261 X_PHC_PV_MDVI_ISO X_PHC_PV_MDVI_AVISO X_PHC_PV_MDVI_PINV1 X_PHC_PV_MDVI_PINV2 X_PHC_PV_MDVI_PINV3 X_PHC_PV_MDVI_PINV4
Default value	none
Description	This parameter may be specified in a mode switch request to select the video mode to be used in transmit direction. The parameter is returned by the service provider in mode switch indications and confirmations. There it indicates the currently active video mode in transmit direction. See the corresponding receive mode parameter X_PHC_P_RMOD_VIDEO for a description of the defined values. A legal value of the video mode parameter is either X_PHC_PV_MDVI_OFF or a combination of one video coding value, one video picture format value, and one video frame rate value. The bit-wise OR operator is used to combine the three values.

Table I.5-41/T.180

Parameter name	X_PHC_P_TMOD_DATA
Type of value	long
Legal values	X_PHC_PV_MDAT_LSDOFF X_PHC_PV_MDAT_LSD300 X_PHC_PV_MDAT_LSD1200 X_PHC_PV_MDAT_LSD4800 X_PHC_PV_MDAT_LSD6400 X_PHC_PV_MDAT_LSD8000 X_PHC_PV_MDAT_LSD9600 X_PHC_PV_MDAT_LSD14400 X_PHC_PV_MDAT_LSD16000 X_PHC_PV_MDAT_LSD24000 X_PHC_PV_MDAT_LSD32000 X_PHC_PV_MDAT_LSD40000 X_PHC_PV_MDAT_LSD48000 X_PHC_PV_MDAT_LSD56000 X_PHC_PV_MDAT_LSD62400 X_PHC_PV_MDAT_LSD64000 X_PHC_PV_MDAT_LSDVAR X_PHC_PV_MDAT_MLPOFF X_PHC_PV_MDAT_MLP4000 X_PHC_PV_MDAT_MLP6400 X_PHC_PV_MDAT_MLPVAR
Default value	none
Description	This parameter may be specified in a mode switch request to switch on/off LSD or MLP in transmit direction at a certain bit rate. LSD and MLP may be switched on simultaneously, but the occupied bits of the I-channel must not overlap. The parameter is returned by the service provider in mode switch indications and confirmations. There it indicates the currently enabled LSD/MLP bit rate in transmit direction. See the corresponding receive mode parameter X_PHC_P_RMOD_DATA for a description of the defined values. A legal value of the LSD/MLP mode parameter is the bit-wise OR combination of one LSD value and one MLP value. The combination of variable LSD and variable MLP is not allowed.

Table I.5-42/T.180

Parameter name	X_PHC_P_TMOD_HDATA
Type of value	long
Legal values	X_PHC_PV_MDHD_HSDOFF X_PHC_PV_MDHD_HSD64 X_PHC_PV_MDHD_HSD128 X_PHC_PV_MDHD_HSD192 X_PHC_PV_MDHD_HSD256 X_PHC_PV_MDHD_HSD320 X_PHC_PV_MDHD_HSD384 X_PHC_PV_MDHD_HMLPOFF X_PHC_PV_MDHD_HMLP62 X_PHC_PV_MDHD_HMLP128 X_PHC_PV_MDHD_HMLP192 X_PHC_PV_MDHD_HMLP256 X_PHC_PV_MDHD_HMLP320 X_PHC_PV_MDHD_HMLP384

Table I.5-42/T.180 *(concluded)*

Default value	none
Description	This parameter may be specified in a mode switch request to switch on/off HSD or H-MLP in transmit direction at a certain bit rate. HSD and H-MLP may be switched on simultaneously, but the occupied time-slots of an H_0 or greater channel must not overlap. The parameter is returned by the service provider in mode switch indications and confirmations. There it indicates the currently enabled HSD/H-MLP bit rate in transmit direction. See the corresponding receive mode parameter X_PHC_P_RMOD_HDATA for a description of the defined values. A legal value of the HSD/H-MLP mode parameter is the bit-wise OR combination of one HSD value and one H-MLP value.

Table I.5-43/T.180

Parameter name	X_PHC_P_TMOD_DATAPPL
Type of value	long
Legal values	X_PHC_PV_MDAP_LSDISOSP X_PHC_PV_MDAP_HSDISOSP X_PHC_PV_MDAP_LSDCURS X_PHC_PV_MDAP_LSDFAX X_PHC_PV_MDAP_HSDFAX X_PHC_PV_MDAP_LSDV120 X_PHC_PV_MDAP_HSDV120
Default value	none
Description	This parameter may be specified in a mode switch request to define the application that transmits data as LSD or HSD. The information is passed to the remote terminal as a BAS command. A legal value of the data application parameter is either a single LSD value, a single HSD value, or the bit-wise OR combination of one LSD value and one HSD value. (Note that LSD and HSD may be activated simultaneously.) The parameter is returned by the service provider in mode switch indications and confirmations. It indicates which application currently uses LSD or HSD for transmission. See the corresponding receive mode parameter X_PHC_P_RMOD_DATAPPL for a description of the defined values.

Table I.5-44/T.180

Parameter name	X_PHC_P_TMOD_TFRATE
Type of value	long
Legal values	X_PHC_PV_MDTR_1X64 X_PHC_PV_MDTR_2X64 X_PHC_PV_MDTR_3X64 X_PHC_PV_MDTR_4X64 X_PHC_PV_MDTR_5X64 X_PHC_PV_MDTR_6X64 X_PHC_PV_MDTR_1X384 X_PHC_PV_MDTR_2X384 X_PHC_PV_MDTR_3X384 X_PHC_PV_MDTR_4X384 X_PHC_PV_MDTR_5X384 X_PHC_PV_MDTR_128 X_PHC_PV_MDTR_192 X_PHC_PV_MDTR_256 X_PHC_PV_MDTR_512 X_PHC_PV_MDTR_768 X_PHC_PV_MDTR_1152 X_PHC_PV_MDTR_1472 X_PHC_PV_MDTR_1536 X_PHC_PV_MDTR_1920
Default value	none
Description	This parameter may be specified in a mode switch request to select the transfer rate in transmit direction. The specified value must not exceed the available connected capacity. If it is less than the connected capacity, the information occupies the lowest-numbered channel(s)/time-slot(s). Note the difference between e.g. X_PHC_PV_MDTR_2X64 and X_PHC_PV_MDTR_128. The first value is applicable for a connected capacity of two or more B-channels only, whereas the second can only be used if the connected capacity is a H_0 or greater channel. The parameter is returned by the service provider in mode switch indications and confirmations. There it indicates the currently active transfer rate in transmit direction. See the corresponding receive mode parameter X_PHC_P_RMOD_TFRATE for a description of the defined values.

Table I.5-45/T.180

Parameter name	X_PHC_P_TMOD_SYNC
Type of value	long
Legal values	X_PHC_PV_MDSY_UNFRAM X_PHC_PV_MDSY_IC_FRMD X_PHC_PV_MDSY_AC_FRMD
Default value	none
Description	This parameter is returned by the service provider in mode switch indications and confirmations. It indicates the currently synchronization state in transmit direction. X_PHC_PV_MDSY_UNFRAM — Transmission is unframed in all channels. X_PHC_PV_MDSY_IC_FRMD — FAS and BAS are transmitted in the I-channel and the remote terminal sends A = 0; X_PHC_PV_MDSY_AC_FRMD — FAS and BAS are transmitted in all additional channels and the remote terminal sends A = 0 in each, i.e. multiframe alignment is validated.

Table I.5-46/T.180

Parameter name	X_PHC_P_TMOD_MISC
Type of value	long
Legal values	X_PHC_PV_MDMS_ECSACT X_PHC_PV_MDMS_6BH0COMP X_PHC_PV_MDMS_RESTRICT X_PHC_PV_MDMS_DERESTRICT
Default value	none
Description	This parameter may be specified in a mode switch request to define miscellaneous attributes that shall be active in transmit direction. The attributes are independent and represented as one bit within the parameter value. If the bit is set to 1, the attribute is active; otherwise, it is not active. The bit-wise AND operator can be used to select a single attribute from the parameter value. The parameter is returned by the service provider in mode switch indications and confirmations. It indicates which miscellaneous attributes are currently active for transmission. See the corresponding receive mode parameter X_PHC_P_RMOD_MISC for a description of the defined values.

I.5.7.3.2.5 Transfer line parameters

Table I.5-47/T.180

Parameter name	X_PHC_P_MODE_TFLINES
Type of value	long
Legal values	X_PHC_PV_MDTL_NONE X_PHC_PV_MDTL_1XB X_PHC_PV_MDTL_2XB X_PHC_PV_MDTL_3XB X_PHC_PV_MDTL_4XB X_PHC_PV_MDTL_5XB X_PHC_PV_MDTL_6XB X_PHC_PV_MDTL_1XH0 X_PHC_PV_MDTL_2XH0 X_PHC_PV_MDTL_3XH0 X_PHC_PV_MDTL_4XH0 X_PHC_PV_MDTL_5XH0 X_PHC_PV_MDTL_H11 X_PHC_PV_MDTL_H12
Default value	none
Description	This parameter indicates the number of connected lines. Note that there is no difference between receive and transmit directions because each line can be used in both directions. X_PHC_PV_MDTL_NONE No connection X_PHC_PV_MDTL_1XB 1 B-channel connected; total capacity 64 kbit/s; X_PHC_PV_MDTL_2XB 2 B-channels connected; total capacity 128 kbit/s; X_PHC_PV_MDTL_3XB 3 B-channels connected; total capacity 192 kbit/s; X_PHC_PV_MDTL_4XB 4 B-channels connected; total capacity 256 kbit/s; X_PHC_PV_MDTL_5XB 5 B-channels connected; total capacity 320 kbit/s; X_PHC_PV_MDTL_6XB 6 B-channels connected; total capacity 384 kbit/s; X_PHC_PV_MDTL_1XH0 1 H_0-channel connected; total capacity 384 kbit/s; X_PHC_PV_MDTL_2XH0 2 H_0-channels connected; total capacity 768 kbit/s; X_PHC_PV_MDTL_3XH0 3 H_0-channels connected; total capacity 1152 kbit/s; X_PHC_PV_MDTL_4XH0 4 H_0-channels connected; total capacity 1536 kbit/s; X_PHC_PV_MDTL_5XH0 5 H_0-channels connected; total capacity 1920 kbit/s; X_PHC_PV_MDTL_H11 1 H_{11}-channel connected; total capacity 1536 kbit/s; X_PHC_PV_MDTL_H12 1 H_{12}-channel connected; total capacity 1920 kbit/s.

Table I.5-48/T.180

Parameter name	X_PHC_P_NUM_LINES
Type of value	long
Legal values	1 ... 5
Default value	none
Description	This parameter is mandatory for add line requests and hang-up line requests. In case of add line it specifies the number of additional channels to established and in case of hang-up line the number of additional channels to be closed. The parameter is returned by an add line confirmation. There it indicates the number of successfully established new channels. Including the initial channel, at most the number of lines indicated by the value of the X_PHC_P_LCAP_TFLINES parameter will be established. The value of X_PHC_P_NUM_LINES will be decremented internally if the requested value plus the number of already established channels would exceed this maximum value.

Table I.5-49/T.180

Parameter name	X_A_OUTBAND_ADR
Type of value	unsigned char []
Legal values	See the corresponding clause in the main part of this Recommendation.
Default value	none
Description	This parameter is mandatory for add line requests. It specifies one outband address (phone number) to be used for establishment of an additional channel. The parameter may be specified more than once in the parameter buffer, at most five times. Note that the number of channels to be established is defined by the X_PHC_P_NUM_LINES parameter and not by the number of addresses. Additional addresses are ignored. The parameter is returned in an add line confirmation if an additional channel was successfully established. The value is one of the addresses specified in the request. It indicates the number used to dial up the channel. The parameter will appear in the parameter buffer as many times as new additional channels were established.

I.5.7.3.3 Data Transfer service

Beside the AV communication, additional channels for asynchronous data transfer may be activated with the Mode Switch service. The data transfer service enables the application program to access a previously activated data channel. The audio and video channels cannot be accessed with the Data Transfer service. Other services have to be used to control AV communication.

NOTE 1 – According to Recommendation H.242 and T.120, MLP and H-MLP are combined to a single MLP subchannel, if both are activated. The bit rate of the combined channel is the sum of the MLP and H-MLP bit rates.

NOTE 2 – Activated data channels occupy a part of the total bandwidth and thus decrease the available bandwidth for video data. Audio bandwidth is not influenced by the activation of data channels.

NOTE 3 – LSD and HSD channels can be used in one direction only. To get a full-duplex connection for asynchronous data transfer, both terminals have to activate an LSD or HSD channel. The bit rates need not necessarily be the same in both directions.

I.5.7.3.3.1 Service description

The Data Transfer service comprises two elements. The XAPI function *x_snddata()* is used to transmit data on a previously established data channel, and the XAPI function *x_rcvdata()* is used to receive data from a data channel.

If a data channel has been opened for receiving, the service provider will store data at the service endpoint until it is received by the application with an *x_rcvdata()* call. If the application does not call *x_rcvdata()* for a longer time, the service provider may run out of storage if no flow control mechanism is used that would stop the sender before this happens. In case of data overflow, some or all of the stored data is released without notification. To avoid loss of data, an application should call *x_rcvdata()* as soon as data is available at the service endpoint.

The service elements and their corresponding XAPI functions needed for Data Transfer are described in Table I.5-50.

Table I.5-50/T.180 – Service elements and their corresponding XAPI functions for Data Transfer

Service element	XAPI function	Description
Data Request	x_snddata()	The Data Request is passed to the provider to transmit asynchronous data. – not applicable for AV Data: The service parameter X_PHC_P_DATA_TYPE is used to specify the type of data passed to the provider.
Data Indication	x_rcvdata()	The Data Indication is generated by the provider to indicate received data. – not applicable for AV Data: The service parameter X_PHC_P_DATA_TYPE specifies the type of data received by the provider.

I.5.7.3.3.2 Service parameter

Table I.5-51 specifies the parameter of the Data Transfer service.

Table I.5-51/T.180 – Parameter of the Data Transfer service

Parameter	Data Transfer service	
	Request	Indication
X_PHC_P_DATA_TYPE	M	M

I.5.7.3.3.3 Service parameter description

Table I.5-52 describes the parameter for the Data Transfer service.

Table I.5-52/T.180

Parameter name	X_PHC_P_DATA_TYPE
Type of value	long
Legal values	X_PHC_PV_LSD_DATA X_PHC_PV_HSD_DATA X_PHC_PV_MLP_DATA
Default value	none
Description	This parameter is used to specify the type of data sent or received by the service provider. The following values for the service parameter X_PHC_P_DATA_TYPE are defined: X_PHC_PV_LSD_DATA low-speed data; X_PHC_PV_HSD_DATA high-speed data; X_PHC_PV_MLP_DATA multilayer protocol.

I.5.7.3.4 Control and Indication service

The Control and Indication (C&I) service enables the application to transmit commands and indications to the remote communication partner and to receive commands and indications sent by the remote partner. The service can be used at any time during a communication after the mode initialization procedure has been completed, i.e. after an X_PHC_SP_INIT_COMPL has been received.

The C&I service provides the basic commands and indications defined in Recommendation H.230. They are classified into four groups:

1) C&I related to audio;
2) C&I related to video;
3) C&I for maintenance purposes; and
4) C&I related to simple multipoint conferences not using MLP.

Beside this, the C&I service provides some means to transmit and receive non-standard commands according to Recommendation H.221.

Recommendations H.242 and H.243 define some procedures to be followed in establishment and during an audiovisual communication. Recommendation H.242 deals with the communication of two AV terminals, while Recommendation H.243 concerns conference calls of three or more AV terminals and the communication between the MCU (Multipoint Control Unit) and an AV terminal. In order to provide interoperability with other AV terminals and communication equipment, the application should obey the rules given in these Recommendations.

I.5.7.3.4.1 Service description

The **transmit C&I request** can be used by either terminal at any time after successful mode initialization (reception of X_PHC_SP_INIT_COMPL) to transmit a single command or a sequence of commands within the BAS to the remote terminal. The BAS is accessible only if operating in a framed mode. No automatic frame reinstatement is provided. A transmit C&I request will fail if the I-channel is not synchronized at the time the transmission is requested. An error indication primitive with cause code CC_OTHER and diagnostic **X_PHC_DC_UNSYNC** is generated by the service

provider and an ERROR event occurs at the service endpoint. The application has to call *x_rcverror()* to consume the event and retrieve the error indication. To avoid this failure, the application should check the current synchronization state of the I-channel before a C&I transmission is requested. If transmission is currently unframed, the application has to switch to a framed mode prior to the C&I transmit request. (The parameter X_PHC_P_TMOD_SYNC of mode switch indications and confirmations shows the current synchronization state in transmission direction.)

The service provider generates a **transmit C&I confirmation** as response to a previous transmit C&I request. This confirmation has only local meaning and is used for flow control purposes. It indicates that transmission of the command or command sequence on the BAS completed and the BAS is now free again for other requests. The confirmation does not imply that the remote terminal understood or accepted the command. Note that due to the effective bandwidth of the BAS (0.4 kbit/s or 50 bytes per second), the transmission time for a longer command sequence will soon count in seconds, which is a really long time in modern computing.

The transmit C&I request requires the BAS for transmission and thus has to be coordinated with the other BAS requests (see I.5.7.3 above). No transmit C&I request may be submitted while a transmit C&I-, a mode switch-, an add line-, or a capability exchange confirmation is outstanding.

A **transmit C&I indication** is generated by the service provider to indicate received commands or indications to the application. It depends on the received command whether an action is required by the application or whether the service provider or Codec already took the appropriate action and the command is indicated for information only.

The service elements and their corresponding XAPI functions needed for Control and Indication are described in Table I.5-53.

Table I.5-53/T.180 – Service elements and their corresponding XAPI functions for Control and Indication

Service element	XAPI function	Service element identifier	Description
Transmit C&I Request	x_sndsp()	X_PHC_SP_TRA_CAI_Q	The Transmit C&I Request primitive is passed to the provider to send one or more commands or indications to the remote terminal. The commands and indications to be transmitted are specified as service parameters. X_PHC_SP_TRA_CAI_Q is the identifier of the service primitive.
Transmit C&I Confirmation	x_rcvsp()	X_PHC_SP_TRA_CAI_C	The Transmit C&I Confirmation primitive is generated by the service provider as acknowledgment of a previous transmit C&I request. It indicates that transmission completed. There are no service parameters defined. X_PHC_SP_TRA_CAI_C is the identifier of the service primitive.
Received C&I Indication	x_rcvsp()	X_PHC_SP_REC_CAI_I	The Received C&I Indication primitive is passed to the application to indicate the reception of a command or indication from the communication partner. The service parameters specify the received command or indication. X_PHC_SP_REC_CAI_I is the identifier of the service primitive.

I.5.7.3.4.2 Service parameters

Table I.5-54 specifies the parameters of the Control and Indication service. Each parameter corresponds to one command or indication. In a transmit C&I request, multiple parameters (commands) may be specified. The commands are transmitted one after another in the sequence of the parameters in the parameter buffer.

An X_PHC_SP_REC_CAI_I primitive always returns only one received command or indication to the application.

NOTE 1 – There are C&I parameters that have no value. These "no-value-parameters" represent simple commands like, Video Command "Freeze Picture". If the parameter is specified in a transmit C&I request, the corresponding command is sent to the remote partner. And, vice versa, if the parameter is returned by a received C&I indication, the corresponding command or indication has been received from the remote terminal (or MCU).

NOTE 2 – The service provider may act in one of the following ways upon the reception of a C&I:

1) The requested (appropriate) action is taken and no indication is passed to the application. Some commands that are handled by the Codec internally are treated in this way. A received video command "Freeze Picture Request", for example, is handled internally and the user will note the result: a frozen picture on the screen. It is not necessary to inform the application program.

2) The requested (appropriate) action is taken and an X_PHC_SP_REC_CAI_I is passed to the application program for information. Some commands that are handled by the Codec internally and/or require immediate reaction are treated in this way. A received loopback command "Video Loop Request", for example, is handled internally but maybe it would a good idea to inform the user that video loopback is active and thus the own picture is not transmitted to the remote terminal any more.

3) No action is taken and the received command or indication is passed on to the application program with an X_PHC_SP_REC_CAI_I. It depends on the C&I whether the application has to act on this indication or not. A received "Multipoint Indication Visualization" (MIV), for example, is simply passed on by the service provider. Such a MIV is transmitted by an MCU to indicate to a terminal that its video signal is being seen by the other terminals. It is up to the application to use this information and, e.g. put on a little red light to inform the person in front of the camera.

Table I.5-54/T.180 – Parameters of the Control and Indication service

Parameter	Control and Indication service	
	Request	Indication
X_PHC_P_H230_AIM	U	C
X_PHC_P_H230_AIA	U	C
X_PHC_P_H230_VCF	U	
X_PHC_P_H230_VCU	U	
X_PHC_P_H230_VIS	U	C
X_PHC_P_H230_VIA	U	C
X_PHC_P_H230_VIA2	U	C
X_PHC_P_H230_VIA3	U	C
X_PHC_P_H230_VIR	U	C
X_PHC_P_H230_LCA	U	C
X_PHC_P_H230_LCV	U	C
X_PHC_P_H230_LCD	U	C
X_PHC_P_H230_LCO	U	C
X_PHC_P_H230_MCV	U	
X_PHC_P_H230_MIV		C
X_PHC_P_H230_MCC		C
X_PHC_P_H230_CN_MCC		C

Table I.5-54/T.180 – Parameters of the Control and Indication service *(concluded)*

Parameter	Control and Indication service	
	Request	**Indication**
X_PHC_P_H230_MCS		C
X_PHC_P_H230_MCN		C
X_PHC_P_H230_MIZ		C
X_PHC_P_H230_CN_MIZ		C
X_PHC_P_H230_MIS		C
X_PHC_P_H230_CN_MIS		C
X_PHC_P_CMD_NONSTD	U	C

Tables I.5-55 to I.5-78 describe the parameters for the Control and Indication Service.

I.5.7.3.4.3 C&I service parameters related to audio

Table I.5-55/T.180

Parameter name	X_PHC_P_H230_AIM
Type of value	no value
Legal values	not applicable
Default value	not applicable
Description	This parameter is returned in a receive C&I indication if an Audio Indicate "Muted" (**AIM**) symbol has been received. The symbol is used to indicate that the content of the audio channel does not represent a normal audio signal. The audio encoder may be without audio input or an electronically-generated tone may have been substituted. The service provider does not take any action upon receipt of AIM. The parameter may be specified in a transmit C&I request to disconnect the audio source (microphone) from the input of the audio encoder and send an AIM to the remote terminal.

Table I.5-56/T.180

Parameter name	X_PHC_P_H230_AIA
Type of value	no value
Legal values	not applicable
Default value	not applicable
Description	This parameter is returned in a receive C&I indication if an Audio Indicate "Active" (**AIA**) symbol has been received. It is the complementary to AIM (see above). The service provider does not take any action upon receipt of AIA. The parameter may be specified in a transmit C&I request to reconnect the audio source (microphone) to the input of the audio encoder and send an AIA to the remote terminal.

I.5.7.3.4.4 C&I service parameters related to video

Table I.5-57/T.180

Parameter name	X_PHC_P_H230_VCF
Type of value	no value
Legal values	not applicable
Default value	not applicable
Description	This parameter may be specified in a transmit C&I request to send a Video Command "Freeze Picture Request" (**VCF**) symbol to the remote terminal. See Recommendation H230 for the meaning of VCF. The service provider does not indicate a received VCF to the application.

Table I.5-58/T.180

Parameter name	X_PHC_P_H230_VCU
Type of value	no value
Legal values	not applicable
Default value	not applicable
Description	This parameter may be specified in a transmit C&I request to send a Video Command "Fast Update Request" (**VCU**) symbol to the remote terminal. See Recommendation H230 for the meaning of VCU. The service provider does not indicate a received VCU to the application.

Table I.5-59/T.180

Parameter name	X_PHC_P_H230_VIS
Type of value	no value
Legal values	not applicable
Default value	not applicable
Description	This parameter is returned in a receive C&I indication if a Video Indicate "Suppressed" (**VIS**) symbol has been received. It is used to indicate that the content of the video does not represent a normal camera image. The video encoder may be without video input or an electronically-generated pattern may have been substituted. The service provider does not take any action upon receipt of VIS. The parameter may be specified in a transmit C&I request to disconnect the current video source from the input of the video encoder and send a VIS to the remote terminal.

Table I.5-60/T.180

Parameter name	X_PHC_P_H230_VIA
Type of value	no value
Legal values	not applicable
Default value	not applicable
Description	This parameter is returned in a receive C&I indication if a Video Indicate "Active" (**VIA**) symbol has been received. It is the complementary to VIS (see above). If there are more video sources to be distinguished, VIA indicates that "video No. 1" is now active. The service provider does not take any action upon receipt of VIA. The parameter may be specified in a transmit C&I request to connect video source No. 1 (usually the camera) to the input of the video encoder and send a VIA to the remote terminal.

Table I.5-61/T.180

Parameter name	X_PHC_P_H230_VIA2
Type of value	no value
Legal values	not applicable
Default value	not applicable
Description	This parameter is returned in a receive C&I indication if a Video Indicate "Source 2 Active" (**VIA2**) symbol has been received. It is equivalent to VIA, but designating "video No. 2" the as source. The service provider does not take any action upon receipt of VIA2. The parameter may be specified in a transmit C&I request to connect video source No. 2 to the input of the video encoder and send a VIA2 to the remote terminal.

Table I.5-62/T.180

Parameter name	X_PHC_P_H230_VIA3
Type of value	no value
Legal values	not applicable
Default value	not applicable
Description	This parameter is returned in a receive C&I indication if a Video Indicate "Source 3 Active" (**VIA3**) symbol has been received. It is equivalent to VIA, but designating "video No. 3" as the source. The service provider does not take any action upon receipt of VIA3. The parameter may be specified in a transmit C&I request to connect video source No. 3 to the input of the video encoder and send a VIA3 to the remote terminal.

Table I.5-63/T.180

Parameter name	X_PHC_P_H230_VIR
Type of value	no value
Legal values	not applicable
Default value	not applicable
Description	This parameter is returned in a receive C&I indication if a Video Indicate "Ready to Activate" (**VIR**) symbol has been received. The service provider does not take any action upon receipt of VIR. The parameter may be specified in a transmit C&I request to send a VIR to the remote terminal. The symbol is transmitted by a terminal whose user has decided not to send video unless she/he will also receive video from the other end.

I.5.7.3.4.5 C&I service parameters for maintenance purposes

Table I.5-64/T.180

Parameter name	X_PHC_P_H230_LCA
Type of value	no value
Legal values	not applicable
Default value	not applicable
Description	This parameter is returned in a receive C&I indication if a Loopback Command "Audio Loop Request" (**LCA**) symbol has been received. The service provider (Codec) acts on this command and connects the output of the audio decoder to the input of the audio encoder. The parameter may be specified in a transmit C&I request to send an LCA to the remote terminal.

Table I.5-65/T.180

Parameter name	X_PHC_P_H230_LCV
Type of value	no value
Legal values	not applicable
Default value	not applicable
Description	This parameter is returned in a receive C&I indication if a Loopback Command "Video Loop Request" (**LCV**) symbol has been received. The service provider (Codec) acts on this command and connects the output of the video decoder to the input of the video encoder. The parameter may be specified in a transmit C&I request to send an LCV to the remote terminal.

Table I.5-66/T.180

Parameter name	X_PHC_P_H230_LCD
Type of value	no value
Legal values	not applicable
Default value	not applicable
Description	This parameter is returned in a receive C&I indication if a Loopback Command "Digital Loop Request" (**LCD**) symbol has been received. The service provider (Codec) acts on this command and disconnects the output of the multiplexer from the outgoing path, replacing it with the input to the demultiplexer. In the case of multiple B or H_0 connections, loopback is activated in each connection. The parameter may be specified in a transmit C&I request to send an LCD to the remote terminal.

Table I.5-67/T.180

Parameter name	X_PHC_P_H230_LCO
Type of value	no value
Legal values	not applicable
Default value	not applicable
Description	This parameter is returned in a receive C&I indication if a Loopback Command "Loops Off" (**LCO**) symbol has been received. The service provider (Codec) acts on this command and disconnects all currently active loops and restores data, audio, and video signal paths to their normal condition. The parameter may be specified in a transmit C&I request to send an LCO to the remote terminal.

I.5.7.3.4.6 C&I service parameters related to multipoint conferences

Table I.5-68/T.180

Parameter name	X_PHC_P_H230_MCV
Type of value	no value
Legal values	not applicable
Default value	not applicable
Description	The parameter may be specified in a transmit C&I request to send a Multipoint Command "Visualization Forcing" (**MCV**). The symbol is transmitted by a terminal to force an associated MCU to broadcast the own video signal. (Used, e.g. to transmit the picture of the chairman or another VIP.)

Table I.5-69/T.180

Parameter name	X_PHC_P_H230_MIV
Type of value	no value
Legal values	not applicable
Default value	not applicable
Description	This parameter is returned in a receive C&I indication if a Multipoint Indication "Visualization" (**MIV**) symbol has been received. The symbol is transmitted by an MCU to indicate to a terminal that its video signal is being seen by the other terminals. Also known as "On-Air" indication. The service provider does not take any action upon receipt of MIV.

Table I.5-70/T.180

Parameter name	X_PHC_P_H230_MCC
Type of value	no value
Legal values	not applicable
Default value	not applicable
Description	This parameter is returned in a receive C&I indication if a Multipoint Command "Conference" (**MCC**) symbol has been received. The symbol is transmitted by an MCU to a terminal to force a symmetrical communication mode. The terminal receiving MCC must make its outgoing transfer rate equal to its incoming transfer rate and its outgoing audio rate equal to its incoming audio rate. If necessary, the service provider switches the used communication mode as required and informs the application with a mode switch indication. For further study: How to handle mode switch requests from the application while MCC is active? a) Ignore them, b) let them fail, c) handle as usual, d) restrict them to mode categories other then transfer and audio rate?

Table I.5-71/T.180

Parameter name	X_PHC_P_H230_CN_MCC
Type of value	no value
Legal values	not applicable
Default value	not applicable
Description	This parameter is returned in a receive C&I indication if a Multipoint Command "Cancel MCC" has been received. The symbol is transmitted by an MCU to cancel the effect of a previously sent MCC. The restrictions to mode switch requests which became active with the MCC are now lifted.

Table I.5-72/T.180

Parameter name	X_PHC_P_H230_MCS
Type of value	no value
Legal values	not applicable
Default value	not applicable
Description	This parameter is returned in a receive C&I indication if a Multipoint Command "Symmetrical Data Transmission" (**MCS**) symbol has been received. The symbol is transmitted by an MCU when setting up data broadcasting. On receipt, a terminal must prepare itself for data reception and insure, by mode change if necessary, that its outgoing data channel occupies the same capacity as its incoming data channel. A terminal in receipt of MCS cannot initiate data broadcasting. If necessary, the service provider switches the used communication mode as required and informs the application with a mode switch indication.

Table I.5-73/T.180

Parameter name	X_PHC_P_H230_MCN
Type of value	no value
Legal values	not applicable
Default value	not applicable
Description	This parameter is returned in a receive C&I indication if a Multipoint Command "Negating MCS" (**MCN**) symbol has been received. The symbol is transmitted by an MCU at the completion of data broadcasting. On receipt, a terminal close any outgoing data channel which it has opened as result of a previous reception of MCS. Following the end of data reception and the receipt of MCN, a terminal is permitted to initiate data broadcasting. If necessary, the service provider uses the outgoing data channels as required and informs the application with a mode switch indication.

Table I.5-74/T.180

Parameter name	X_PHC_P_H230_MIZ
Type of value	no value
Legal values	not applicable
Default value	not applicable
Description	This parameter is returned in a receive C&I indication if a Multipoint Indication "Zero-communication" (**MIZ**) symbol has been received. The symbol is transmitted by an MCU to a terminal for information, with the meaning that no other terminals are yet connected to the MCU. The service provider does not take any action upon receipt of MIZ.

Table I.5-75/T.180

Parameter name	X_PHC_P_H230_CN_MIZ
Type of value	no value
Legal values	not applicable
Default value	not applicable
Description	This parameter is returned in a receive C&I indication if a Multipoint Indication "Cancel MIZ" has been received. The symbol is transmitted by an MCU to revoke a previous MIZ. The service provider does not take any action upon receipt of Cancel MIZ.

Table I.5-76/T.180

Parameter name	X_PHC_P_H230_MIS
Type of value	no value
Legal values	not applicable
Default value	not applicable
Description	This parameter is returned in a receive C&I indication if a Multipoint Indication "Secondary Status" (**MIS**) symbol has been received. The symbol is transmitted by an MCU to a terminal for information, with the meaning that since other terminals of higher capability are participating in the conference-call, this terminal will not necessarily receive all the signals that are sent to those other terminals (see Recommendation H.243). The service provider does not take any action upon receipt of MIS.

Table I.5-77/T.180

Parameter name	X_PHC_P_H230_CN_MIS
Type of value	no value
Legal values	not applicable
Default value	not applicable
Description	This parameter is returned in a receive C&I indication if a Multipoint Indication "Cancel MIS" has been received. The symbol is transmitted by an MCU to revoke a previous MIS. The service provider does not take any action upon receipt of Cancel MIS.

I.5.7.3.4.7 C&I service parameters for non-standard commands

Table I.5-78/T.180

Parameter name	X_PHC_P_CMD_NONSTD
Type of value	unsigned char [ ], 5 ... 255 bytes
Legal values	Any sequence of bytes. The minimum length is 5, the maximum length is **X_PHC_C_MAX_NONSTD** (255). The first 4 bytes have a special meaning, see description below.
Default value	none
Description	This parameter may be specified in a transmit C&I request to send a non-CCITT command message to the remote terminal. The first two bytes of the value specify a country code, and the next two bytes specify a terminal manufacturer code. The meaning of all following bytes is defined by the manufacturer. [byte 0] Country code according to Recommendation T.35 [byte 1] Country code [bytes 2,3] Manufacturer Code (e.g. Company 4711) [bytes 5-n] Manufacturer specific The second byte of the country code and the manufacturer code are assigned nationally. This parameter is returned in a receive C&I indication if a non-CCITT command message has been received. The first four bytes have the same meaning as defined above. The service provider takes no action upon receipt of a non-CCITT command message and does not check country and/or manufacturer code. All bytes are passed on as received. The *len* field specifies the length of the message. Note that due to the effective bandwidth of the BAS (0.4 kbit/s rsp. 50 bytes per second), the transmission time for a non-standard command of maximum length will be more than 5 seconds.

I.5.7.4 Connection Release service

I.5.7.4.1 Service description

The Connection Release service enables the XAPI user to release the AV connection in an orderly manner. It may be used by either terminal after successful mode initialization (reception of service primitive X_PHC_SP_INIT_COMPL). The "mode 0 forcing" procedure is initiated and all B-/H_0-channels are released in a controlled manner as defined in Recommendation H.242. No orderly release indication (to be consumed with a *x_relind()* calls) is defined. If an orderly connection release is initiated by the remote terminal, the local terminal will get a mode switch indication (service primitive X_PHC_MODE_SWITCH_I) as a result of the mode 0 forcing. After this, the disconnection of the initial channel will be indicated to the application with a disconnect indication (see below).

The elements of the Connection Release service and the corresponding XAPI functions are described in Table I.5-79.

Table I.5-79/T.180 – Service elements and their corresponding XAPI functions for Connection Release

Service element	XAPI Function	Description
Release Request	x_relreq()	The Release Request is passed to the provider to request an orderly AV connection release.
Release Confirmation	x_relconf()	The Release Confirmation is generated by the provider as positive confirmation of an orderly AV Connection release.
End Indication	x_rcvend()	The End Indication is generated by the provider to indicate that the service provider is ready to establish a new connection. In addition it contains some information about the released connection.

I.5.7.4.2 Service parameters

Table I.5-80 specifies the parameters of the Connection Release service:

Table I.5-80/T.180

Parameter	Connection Release service		
	Request	Confirmation	End Indication
X_P_CONN_TIME			M
X_P_DISC_TIME			M
X_P_CHARGE			C
X_P_DISC_REASON			C

I.5.7.4.3 Service parameter descriptions

Tables I.5-81 to I.5-84 describe the parameters for the Connection Release service.

Table I.5-81/T.180

Parameter name	X_P_CONN_TIME
Type of value	unsigned long
Legal values	the time in seconds since 01.01.1970 00:00:00 GMT
Default value	none
Description	This parameter contains the time of physical connection of the I-channel.

Table I.5-82/T.180

Parameter name	X_P_DISC_TIME
Type of value	unsigned long
Legal values	the time in seconds since 01.01.1970 00:00:00 GMT
Default value	none
Description	This parameter contains the time of physical disconnection of the I-channel. If X_P_CONN_TIME and X_P_DISCON_TIME both set to zero, no physical connection could be established.

Table I.5-83/T.180

Parameter name	X_P_CHARGE
Type of value	unsigned long
Legal values	any number
Default value	zero
Description	This parameter contains the number of charging units for the connection (including all transfer lines used). It will be present only if both network and network connection provide this facility.

Table I.5-84/T.180

Parameter name	X_P_DISC_REASON
Type of value	unsigned long
Legal values	The values are directly mapped from the underlying hardware. See the hardware documentation to get further information.
Default value	zero
Description	This parameter contains, if available, the reason for disconnecting the I-channel.

I.5.7.5 Connection Abort service

I.5.7.5.1 Service description

The Connection Abort service allows either user of the AV Codec control service to disconnect all established channels at once. The mode 0 forcing procedure of Recommendation H.242 is not followed. Use of this service will cause loss of undelivered data of the data transfer service (LSD/HSD).

The service elements and their corresponding XAPI functions needed for disconnection of an AV Codec connection are described in Table I.5-85.

Table I.5-85/T.180 – Service elements and their corresponding XAPI functions for Connection Abort

Service element	XAPI function	Description
Disconnect Request	x_snddis()	The Disconnect Request is passed to the provider to request a disconnection.
Disconnect Indication	x_rcvdis()	The Disconnect Indication is generated by the provider to indicate the release of all channels of an AV connection including the initial channel. The disconnection may be initiated by the local service provider or the remote peer entity.
End Indication	x_rcvend()	The End Indication is generated by the provider to indicate that the endpoint is ready again to establish a new connection. In addition it contains some information about the aborted connection.

I.5.7.5.2 Service parameters

Table I.5-86 specifies the parameters of the Connection Abort service:

Table I.5-86/T.180

Parameter	Abort service		
	Request	Indication	End Indication
X_P_CONN_TIME			M
X_P_DISC_TIME			M
X_P_CHARGE			
X_P_DISC_REASON			

I.5.7.5.3 Service parameter descriptions

Tables I.5-87 to I.5-90 describe the parameters for the Connection Abort service.

Table I.5-87/T.180

Parameter name	X_P_CONN_TIME
Type of value	unsigned long
Legal values	the time in seconds since 01.01.1970 00:00:00 GMT
Default value	none
Description	This parameter contains the time of physical connection of the I-channel.

Table I.5-88/T.180

Parameter name	X_P_DISC_TIME
Type of value	unsigned long
Legal values	the time in seconds since 01.01.1970 00:00:00 GMT
Default value	none
Description	This parameter contains the time of physical disconnection of the I-channel. If X_P_CONN_TIME and X_P_DISCON_TIME both set to zero, no physical connection could be established.

Table I.5-89/T.180

Parameter name	X_P_CHARGE
Type of value	unsigned long
Legal values	any number
Default value	zero
Description	This parameter contains the number of charging units for the connection (including all transfer lines used). It will be present only if both network and network connection provide this facility.

Table I.5-90/T.180

Parameter name	X_P_DISC_REASON
Type of value	unsigned long
Legal values	The values are directly mapped from the underlying hardware. See the hardware documentation to get further information.
Default value	zero
Description	This parameter contains, if available, the reason for disconnecting the I-channel.

I.5.7.6 Local Control service

The Local Control service is based on the XAPI function *x_sndsp()* and enables the user of the AV Codec control service to trigger local actions. The service can be used by either partner at any time after successful mode initialization. One of the defined actions is, for example, to send a picture release signal within the H.261 data to cancel a previously submitted VCF (video command freeze picture request).

The elements of the Local Control service and their parameters are listed in Tables I.5-91 and I.5-92.

Table I.5-91/T.180 – Service element and their corresponding XAPI function for Local Control

Service element	XAPI function	Service element identifier	Description
Local Control Request	x_sndsp()	X_PHC_SP_LCTRL_Q	The Local Control Request primitive is submitted to the provider to initiate the local action specified as service parameter. Multiple actions with one request are not supported.

Table I.5-92/T.180 – Parameters of the Local Control

Parameter	Local Control service
	Request
X_PHC_P_H261CMD_SNDINTRA	U
X_PHC_P_H261CMD_PICTREL	U

I.5.7.6.1 Service parameter descriptions

Tables I.5-93 and I.5-94 describe the parameters for the Local Control service.

Table I.5-93/T.180

Parameter name	X_PHC_P_H261CMD_SNDINTRA
Type of value	no value
Legal values	not applicable
Default value	not applicable
Description	This parameter may be specified in a local control request to initiate the sending of one intra coded frame. This action is sometimes useful if an external video switch is used to change the source of video input signals. After switching the video source, an enforced intra frame will help the remote decoder to get back on the right way. The general H.261 encoding mode as specified by the X_PHC_O_H261_ENCODE option is not influenced by an enforced intra frame.

Table I.5-94/T.180

Parameter name	X_PHC_P_H261CMD_PICTREL
Type of value	no value
Legal values	not applicable
Default value	not applicable
Description	This parameter may be specified in a local control request to send a picture release signal within H.261 data to the remote terminal. On reception of picture release, the remote terminal will cancel a previously received VCF (video command freeze picture request).

I.5.7.7 Option management

The **management of protocol options** is based on the XAPI function *x_optmgmt()*. It enables the XAPI user to set the value of a protocol option, to check a value without changing the current value, and to retrieve the current or default value. See the main part of this Recommendation for a detailed description of option management in general and the manual page of the *x_optmgmt()* function. The option management may be used in any state of the service endpoint. Protocol options have only local meaning. If a connection is established on an endpoint, changes in protocol options are not indicated to the remote partner. But the data transmission over this connection may be influenced by the current setting of some protocol options. For each protocol option, there is a default value defined in the XAPI configuration. This default is sufficient for the majority of applications and usually need not be changed.

The AV Codec control service supports five groups of protocol options:

- the default capability options: For each local capability parameter there is a corresponding protocol option which defines the default value for this parameter. If the parameter is not specified in a request, the value of the corresponding protocol option is used instead;
- standard H.261 options which control the video encoding process;
- non-standard H.261 options;
- non-standard audio options;
- non-standard maintenance options.

The standard options have to be implemented by each concrete service provider while the non-standard options need not be present in all implementations. Sometimes it depends on the Codec hardware if an option can be implemented or not. All options have permanent scope, i.e. they are in effect for the whole lifetime of the service endpoint. The default option values are usually configured to the maximum values supported by the underlying hardware.

I.5.7.7.1 Protocol option descriptions

Tables I.5-95 to I.5-118 describe the protocol options.

I.5.7.7.1.1 Default capability options

Table I.5-95/T.180

Option name	X_PHC_O_LCAP_AUDIO
Type of value	long
Legal values	X_PHC_PV_CAPNON X_PHC_PV_AUDIO_711_A X_PHC_PV_AUDIO_711_U X_PHC_PV_AUDIO_722_64 X_PHC_PV_AUDIO_722_48 X_PHC_PV_AUDIO_16K X_PHC_PV_AUDIO_ISO
Default value	The configured default value may be retrieved with the *x_optmgmt()* function.
Description	The option defines the default value of parameter X_PHC_P_LCAP_AUDIO, which specifies the local audio reception capabilities in the connection establishment and capability exchange service. The option may take the same values as defined for the parameter. See above for a description of these values.

Table I.5-96/T.180

Option name	X_PHC_O_LCAP_VIDEO
Type of value	long
Legal values	X_PHC_PV_CAPNON X_PHC_PV_VIDEO_QCIF X_PHC_PV_VIDEO_CIF X_PHC_PV_VIDEO_PINV1 X_PHC_PV_VIDEO_PINV2 X_PHC_PV_VIDEO_PINV3 X_PHC_PV_VIDEO_PINV4 X_PHC_PV_VIDEO_IMP X_PHC_PV_VIDEO_ISO X_PHC_PV_VIDEO_AVISO
Default value	The configured default value may be retrieved with the *x_optmgmt()* function.
Description	The option defines the default value of parameter X_PHC_P_LCAP_VIDEO, which specifies the local video reception capabilities in the connection establishment and capability exchange service. The option may take the same values as defined for the parameter. See above for a description of these values.

Table I.5-97/T.180

Option name	X_PHC_O_LCAP_DATA
Type of value	long
Legal values	X_PHC_PV_CAPNON X_PHC_PV_DATA_VAR X_PHC_PV_DATA_300 X_PHC_PV_DATA_1200 X_PHC_PV_DATA_4800 X_PHC_PV_DATA_6400 X_PHC_PV_DATA_8000 X_PHC_PV_DATA_9600 X_PHC_PV_DATA_14400 X_PHC_PV_DATA_16000 X_PHC_PV_DATA_24000 X_PHC_PV_DATA_32000 X_PHC_PV_DATA_40000 X_PHC_PV_DATA_48000 X_PHC_PV_DATA_56000 X_PHC_PV_DATA_62400 X_PHC_PV_DATA_64000 X_PHC_PV_DATA_MLP4000 X_PHC_PV_DATA_MLP6400 X_PHC_PV_DATA_VARMLP
Default value	The configured default value may be retrieved with the *x_optmgmt()* function.
Description	The option defines the default value of parameter X_PHC_P_LCAP_DATA, which specifies the local LSD and MLP reception capabilities in the connection establishment and capability exchange service. The option may take the same values as defined for the parameter. See above for a description of these values.

Table I.5-98/T.180

Option name	X_PHC_O_LCAP_HDATA
Type of value	long
Legal values	X_PHC_PV_CAPNON X_PHC_PV_HDATA_64 X_PHC_PV_HDATA_128 X_PHC_PV_HDATA_192 X_PHC_PV_HDATA_256 X_PHC_PV_HDATA_320 X_PHC_PV_HDATA_384 X_PHC_PV_HDATA_512 X_PHC_PV_HDATA_768 X_PHC_PV_HDATA_1152 X_PHC_PV_HDATA_1536 X_PHC_PV_HDATA_VAR X_PHC_PV_HDATA_MLP62 X_PHC_PV_HDATA_MLP64 X_PHC_PV_HDATA_MLP128 X_PHC_PV_HDATA_MLP192 X_PHC_PV_HDATA_MLP256 X_PHC_PV_HDATA_MLP320 X_PHC_PV_HDATA_MLP384 X_PHC_PV_HDATA_VARMLP
Default value	The configured default value may be retrieved with the *x_optmgmt()* function.
Description	The option defines the default value of parameter X_PHC_P_LCAP_HDATA, which specifies the local HSD and HMLP reception capabilities in the connection establishment and capability exchange service. The option may take the same values as defined for the parameter. See above for a description of these values.

Table I.5-99/T.180

Option name	X_PHC_O_LCAP_TFRATE
Type of value	long
Legal values	X_PHC_PV_CAPNON X_PHC_PV_TFRATE_128 X_PHC_PV_TFRATE_192 X_PHC_PV_TFRATE_256 X_PHC_PV_TFRATE_512 X_PHC_PV_TFRATE_768 X_PHC_PV_TFRATE_1152 X_PHC_PV_TFRATE_1472
Default value	The configured default value may be retrieved with the *x_optmgmt()* function.
Description	The option defines the default value of parameter X_PHC_P_LCAP_TFRATE, which specifies the local transfer rate capabilities in the connection establishment and capability exchange service. The option may take the same values as defined for the parameter. See above for a description of these values.

Table I.5-100/T.180

Option name	X_PHC_O_LCAP_TFLINES
Type of value	long
Legal values	X_PHC_PV_TFLINES_1B X_PHC_PV_TFLINES_2B X_PHC_PV_TFLINES_3B X_PHC_PV_TFLINES_4B X_PHC_PV_TFLINES_5B X_PHC_PV_TFLINES_6B X_PHC_PV_TFLINES_1H X_PHC_PV_TFLINES_2H X_PHC_PV_TFLINES_3H X_PHC_PV_TFLINES_4H X_PHC_PV_TFLINES_5H X_PHC_PV_TFLINES_H11 X_PHC_PV_TFLINES_H12
Default value	The configured default value may be retrieved with the *x_optmgmt()* function.
Description	The option defines the default value of parameter X_PHC_P_LCAP_TFLINES, which specifies the local transfer lines capabilities in the connection establishment and capability exchange service. The option may take the same values as defined for the parameter. See above for a description of these values.

Table I.5-101/T.180

Option name	X_PHC_O_LCAP_MISC
Type of value	long
Legal values	X_PHC_PV_CAPNON X_PHC_PV_MISC_ENCR X_PHC_PV_MISC_ESC X_PHC_PV_MISC_MBE X_PHC_PV_MISC_RESTRICT X_PHC_PV_MISC_6BHCOMP
Default value	The configured default value may be retrieved with the *x_optmgmt()* function.
Description	The option defines the default value of parameter X_PHC_P_LCAP_MISC, which specifies some local miscellaneous capabilities in the connection establishment and capability exchange service. The option may take the same values as defined for the parameter. See above for a description of these values.

Table I.5-102/T.180

Option name	X_PHC_O_LCAP_DATAPPL
Type of value	long
Legal values	X_PHC_PV_CAPNON X_PHC_PV_APPL_STILLPIC_LSD X_PHC_PV_APPL_STILLPIC_HSD X_PHC_PV_APPL_STILLPIC_SPATIAL X_PHC_PV_APPL_STILLPIC_PROG X_PHC_PV_APPL_STILLPIC_ARITH X_PHC_PV_APPL_STILLIMAGE X_PHC_PV_APPL_CURSORDATA X_PHC_PV_APPL_FAX3 X_PHC_PV_APPL_FAX4 X_PHC_PV_APPL_V120_LSD X_PHC_PV_APPL_V120_HSD
Default value	The configured default value may be retrieved with the *x_optmgmt()* function.
Description	The option defines the default value of parameter X_PHC_P_LCAP_DATAPPL, which specifies the local data application capabilities in the connection establishment and capability exchange service. The option may take the same values as defined for the parameter. See above for a description of these values.

I.5.7.7.1.2 Standard H.261 options

Table I.5-103/T.180

Option name	X_PHC_O_H261_ENCODE
Type of value	long
Legal values	X_PHC_OV_H261_SIMPLE X_PHC_OV_H261_CIF X_PHC_OV_H261_INTER X_PHC_OV_H261_MOTNEST X_PHC_OV_H261_LOOPFILTER
Default value	The configured default value may be retrieved with the *x_optmgmt()* function.
Description	This option controls the H.261 video encoding. A value of X_PHC_OV_H261_SIMPLE specifies the simplest possible encoding, intra frames only in QCIF format. The other defined values each control one additional encoding feature that may be enabled by setting this value. To enable more than one feature, the corresponding values have to be combined with the bit-wise OR operator. X_PHC_OV_H261_CIF Use CIF instead of QCIF; X_PHC_OV_H261_INTER Use intra and inter frames; X_PHC_OV_H261_MOTNEST Use intra and inter frames and motion estimation; X_PHC_OV_H261_LOOPFILTER Switch the loop filter on. The values X_PHC_OV_H261_INTER and X_PHC_OV_H261_MOTNEST must not be specified together. The option is not an absolute requirement. The specified value may be diminished by the service provider if the requested feature is not supported by the Codec hardware.

Table I.5-104/T.180

Option name	X_PHC_O_H261_CODERPINV
Type of value	long
Legal values	1 ... 100
Default value	The configured default value may be retrieved with the *x_optmgmt()* function.
Description	This option controls the minimum picture interval used in H.261 video encoding. The interval is not specified as absolute value but relative to the maximum speed supported by the Codec hardware. The option value specifies the percentage of the maximum speed to be used in encoding. A value of 50, for example, tells the Codec to work as fast as possible, and a value of 100 means "full speed ahead". Note that a decreased picture interval usually results in improved picture quality because more data can be transmitted for one picture.

Table I.5-105/T.180

Option name	X_PHC_O_H261_AUDIODELAY
Type of value	long
Legal values	0 ... 1000
Default value	0
Description	This option controls the delay of received audio signals. It compensates the signal processing time of the video decoder to maintain lip synchronization. The option value specifies the audio delay in milliseconds.

I.5.7.7.1.3 Non-standard H.261 options

Table I.5-106/T.180

Option name	X_PHC_O_H261NS_SOURCE
Type of value	long
Legal values	1 ... 16
Default value	The configured default value may be retrieved with the *x_optmgmt()* function. Usually it is set to 1.
Description	This option selects the source for input video signals. The option value specifies the source number. It depends on the hardware which number corresponds to which connector. If the requested source is not available or not ready, the *x_optmgmt()* call fails with NOTSUPPORT, and the current setting is retained.

Table I.5-107/T.180

Option name	X_PHC_O_H261NS_DCODFILTER
Type of value	unsigned char [] with at least one element
Legal values	The legal values depend on the concrete service provider. See the release notes for further information. There is one option value that has to be implemented by all concrete service providers supporting this option: an array with only one element that is set to 0. This value specifies that no filter is enabled.
Default value	0, no filter enabled
Description	This option selects and enables one of the present video decoding filters. It depends on the concrete service provider what kind of filters are present and with which numbers they are denoted. The first byte of the option value specifies the filter to be used. A value of 0 disables all filters. The additional elements of the option value (if there are any at all) are parameters which control the operation of the filter. The number of filter parameters and their meaning depend on the filter selected by the first element of the option value. This option can be used, for example, to enable a filter that smoothes the edges in the decoded video picture.

Table I.5-108/T.180

Option name	X_PHC_O_H261NS_INTERPAR
Type of value	unsigned char [4]
Legal values	The legal values depend on the concrete service provider. See the release notes for further information.
Default value	The configured default value may be retrieved with the *x_optmgmt()* function.
Description	This option defines four parameters that control the coding of inter frames in Recommendation H.261. byte [0] upper boundary; byte [1] lower boundary; byte [2] fixed value; byte [3] switch on/off adaptive quantization; a value of 1 means on, 0 means off.

Table I.5-109/T.180

Option name	X_PHC_O_H261NS_MOTNPAR
Type of value	unsigned char [] with at least one element
Legal values	The legal values depend on the concrete service provider. See the release notes for further information.
Default value	The configured default value may be retrieved with the *x_optmgmt()* function.
Description	This option selects one of the present algorithms for motion estimation in H.261 video encoding. It depends on the concrete service provider which algorithms are present and with which numbers they are denoted. The first byte of the option value selects the algorithm by number. The additional elements of the option value (if there are any at all) are parameters used within the algorithm. The number of parameters and their meaning depend on the selected algorithm. Note that the use of motion estimation is enabled rsp. disabled with the standard option H.261 encoding (X_PHC_O_H261_ENCODE).

Table I.5-110/T.180

Option name	X_PHC_O_H261NS_LOOPPAR
Type of value	unsigned char [] with at least one element
Legal values	The legal values depend on the concrete service provider. See the release notes for further information.
Default value	The configured default value may be retrieved with the *x_optmgmt()* function.
Description	This option selects one of the present loop filters for use in H.261 video encoding. It depends on the concrete service provider which filters are present and with which numbers they are denoted. The first byte of the option value selects the filter by number. The additional elements of the option value (if there are any at all) are parameters that control the filter. The number of parameters and their meaning depend on the selected filter. Note that the use of motion estimation is enabled rsp. disabled with the standard option H.261 encoding (X_PHC_O_H261_ENCODE).

Table I.5-111/T.180

Option name	X_PHC_O_H261NS_PREFILTER
Type of value	unsigned char [] with at least one element
Legal values	The legal values depend on the concrete service provider. See the release notes for further information. There is one option value that has to be implemented by all concrete service providers supporting this option: an array with only one element that is set to 0. This value specifies that no filter is enabled.
Default value	0, no filter enabled
Description	This option selects and enables one of the present video pre-filters. It depends on the concrete service provider what kind of filters are present and with which numbers they are denoted. The first byte of the option value specifies the filter to be used. A value of 0 disables all filters. The additional elements of the option value (if there are any at all) are parameters which control the operation of the filter. The number of filter parameters and their meaning depend on the filter selected by the first element of the option value. This option can be used, for example, to enable a filter that smoothes the edges in the decoded video picture.

Table I.5-112/T.180

Option name	X_PHC_O_H261NS_POSTFILTER
Type of value	unsigned char [] with at least one element
Legal values	The legal values depend on the concrete service provider. See the release notes for further information. There is one option value that has to be implemented by all concrete service providers supporting this option: an array with only one element that is set to 0. This value specifies that no filter is enabled.
Default value	0, no filter enabled
Description	This option selects and enables one of the present video post filters. It depends on the concrete service provider what kind of filters are present and with which numbers they are denoted. The first byte of the option value specifies the filter to be used. A value of 0 disables all filters. The additional elements of the option value (if there are any at all) are parameters which control the operation of the filter. The number of filter parameters and their meaning depend on the filter selected by the first element of the option value. This option can be used, for example, to enable a filter that smoothes the edges in the decoded video picture.

Table I.5-113/T.180

Option name	X_PHC_O_H261NS_OUTPUT
Type of value	long
Legal values	X_PHC_OV_VFRMT_PAL_N X_PHC_OV_VFRMT_PAL_M X_PHC_OV_VFRMT_PAL_BG X_PHC_OV_VFRMT_PAL_443 X_PHC_OV_VFRMT_SECAM X_PHC_OV_VFRMT_NTSC_M X_PHC_OV_VFRMT_NTSC_443 X_PHC_OV_VFRMT_BW
Default value	The configured default value may be retrieved with the *x_optmgmt()* function.
Description	This option controls the video output format. Only one output format may be selected at one time, i.e. the defined values must not be combined. X_PHC_OV_VFRMT_PAL_N PAL with 50 Hz; X_PHC_OV_VFRMT_PAL_M PAL with 60 Hz; X_PHC_OV_VFRMT_PAL_443 PAL with 60 Hz; X_PHC_OV_VFRMT_SECAM SECAM with 50 Hz; X_PHC_OV_VFRMT_NTSC_M NTSC with 60 Hz; X_PHC_OV_VFRMT_NTSC_443 NTSC with 60 Hz; X_PHC_OV_VFRMT_BW monochrome black and white. The option is an absolute requirement. The *x_optmgmt()* call will fail if the requested video output format is not supported by the Codec hardware.

Table I.5-114/T.180

Option name	X_PHC_O_H261NS_INPUT
Type of value	long
Legal values	X_PHC_OV_VFRMT_PAL_N X_PHC_OV_VFRMT_PAL_M X_PHC_OV_VFRMT_PAL_BG X_PHC_OV_VFRMT_PAL_443 X_PHC_OV_VFRMT_SECAM X_PHC_OV_VFRMT_NTSC_M X_PHC_OV_VFRMT_NTSC_443 X_PHC_OV_VFRMT_BW
Default value	The configured default value may be retrieved with the *x_optmgmt()* function.
Description	This option controls the video input format. Only one input format may be selected at one time, i.e. the defined values must not be combined. The option may take the same values as the output format option. The selected format applies to all video signal sources, if there is more than one available. The option is an absolute requirement. The *x_optmgmt()* call will fail if the requested video input format is not supported by the Codec hardware.

I.5.7.7.1.4 Non-standard audio options

Table I.5-115/T.180

Option name	X_PHC_O_NSAUD_SOURCE
Type of value	long
Legal values	1 ... 16
Default value	The configured default value may be retrieved with the *x_optmgmt()* function. Usually it is set to 1.
Description	This option selects the source for input audio signals. The option value specifies the source number. It depends on the hardware which number corresponds to which connector rsp. signal level. If the requested source is not available or not ready, the *x_optmgmt()* call fails with NOTSUPPORT and the current setting is retained.

Table I.5-116/T.180

Option name	X_PHC_O_NSAUD_SPEAKER
Type of value	long
Legal values	0, 1
Default value	1
Description	This option controls the local speaker. A value of 1 switches the speaker on, and a value of 0 switches it off. The setting of the audio volume is not influenced by this option.

Table I.5-117/T.180

Option name	X_PHC_O_NSAUD_VOLUME
Type of value	long
Legal values	any value in the range of 0 ... 127 with 127 being loudest
Default value	The configured default value may be retrieved with the *x_optmgmt()* function. Usually it is set to 63.
Description	This option controls the audio output volume.

I.5.7.7.1.5 Non-standard maintenance options

Table I.5-118/T.180

Option name	X_PHC_O_NSMNT_LOCAL_LOOP
Type of value	long
Legal values	X_PHC_OV_LLP_OFF X_PHC_OV_LLP_AUDIO X_PHC_OV_LLP_VIDEO X_PHC_OV_LLP_DIGITAL
Default value	X_PHC_OV_LLP_OFF, all local loops deactivated
Description	This option controls a local loop back of output signals to the input. Corresponding to the remote loops there are three local loops defined: audio, video and digital. For each loop there is one value defined that has to be specified as option value to activate the loop. To activate more than one loop simultaneously, the option has to be set to the bit-wise OR combination of the corresponding values. The option value X_PHC_OV_LLP_OFF deactivates all local loops X_PHC_OV_LLP_OFF — All local loops deactivated; X_PHC_OV_LLP_AUDIO — Local loop back of audio output signals to audio input; X_PHC_OV_LLP_VIDEO — Local loop back of video output signals to video input; X_PHC_OV_LLP_DIGITAL — Local loop back of digital output to digital input.

I.5.7.8 Usage of XAPI functions

This subclause provides some protocol-specific remarks on the use of the XAPI functions. The functions are mentioned in alphabetical order.

x_conconf The user_data buffer in the *call_struct* is empty, as transfer of user data is not supported in the connection establishment phase.

x_conind The user_data buffer in the *conind_struct* is empty, as transfer of user data is not supported in the connection establishment phase.

x_conreq The user_data buffer in the *call_struct* has to be empty, as transfer of user data is not supported in the connection establishment phase. Usage of the MORE flag is not supported.

x_conrsp The user_data buffer in the *call_struct* has to be empty, as transfer of user data is not supported in the connection establishment phase. Usage of the MORE flag is not supported.

x_bind The address buffer *own_address.buf* of the *bind_struct* may be set to the NULL pointer. In this case the endpoint is bound to a preconfigured address. The service endpoint may be bound to multiple addresses. To achieve this, the address parameter A_OUTBAND_ADR may be repeated in the address buffer.

x_open To create an endpoint accessing the AV Codec control service provider for ISDN, the service provider name "X_PHC_ISDN" must be used.

x_relconf The user_data buffer in the *release_struct* is empty, as transfer of user data is not supported in the connection release phase.

x_relind The AV Codec control service provider does not support passive orderly release. The XAPI event RELIND will never occur when accessing this provider. Thus the *x_relind()* consuming function need not be called by the application.

x_relreq The user_data buffer in the *release_struct* has to be empty, as transfer of user data is not supported in the connection release phase. Usage of the MORE flag is not supported.

x_relrsp The AV Codec control service provider does not support passive orderly release. No release indication is defined to which the application could respond. Thus the application need not call the XAPI function *x_relresp()*.

x_rcvdata Expedited data is not supported. This service is not applicable for AV data.

x_rcvdis The user_data buffer in the *discon_struct* is empty, as transfer of user data is not supported by the connection abort service.

x_snddata Expedited data is not supported. This service is not applicable for AV data.

x_snddis The user_data buffer in the *discon_struct* has to be empty, as transfer of user data is not supported by the connection abort service. Usage of the MORE flag is not supported.

I.5.7.9 Tables of error codes

The XAPI error-level error codes are defined in Annex B.

I.5.7.9.1 CC_BADVALUE

The cause code CC_BADVALUE indicates that a parameter with a bad value was specified. The value of *diagnostic* then indicates the erroneous parameter identifier which has been submitted with the XAPI function call that caused the error indication.

I.5.7.9.2 CC_MANDMISS

The cause code CC_MANDMISS indicates that a mandatory parameter is missing. The value of *diagnostic* then indicates the missing parameter identifier that caused the error indication.

I.5.7.9.3 CC_BADEVENT

The cause code CC_BADEVENT indicates that an unknown event occurred. The value of *diagnostic* then indicates the bad event identifier which has been submitted with the XAPI function call that caused the error indication.

I.5.7.9.4 CC_UNEXPECT

The cause code CC_UNEXPECT indicates that an event occurred which is unexpected in the current state of communication. The value of *diagnostic* then indicates the actual state identifier in which the unexpected event caused the error indication.

I.5.7.9.5 CC_NOTSUPPORT

The cause code CC_NOTSUPPORT indicates that an unsupported event occurred. The value of *diagnostic* then indicates the identifier of the unsupported event which has been submitted with the XAPI function call that caused the error indication.

I.5.7.9.6 CC_OTHER

The cause code CC_OTHER is used if none of the cause codes mentioned above applies. The following diagnostic codes are defined in the AV Codec control service:

X_PHC_DC_UNSYNC	A transmit C&I request was submitted while the initial channel is not synchronized.
X_PHC_DC_NOCAPSET	No complete capability set of the remote terminal was received in a capability exchange.
X_PHC_DC_BADCOMBI	An inconsistent combination of audio, video, and data is specified in a mode switch request.
X_PHC_DC_CAPCONFLICT	The new mode specified in a mode switch request is in conflict with the known receive and decode capabilities of the remote terminal.

I.6 XAPI access to the service provider for the T.120 conference control

This part of Appendix I describes an example of how the service provider can be implemented, if an application needs access to the specified service.

I.6.1 Scope

The XAPI, which is an abbreviation for eXtensive Application Programming Interface, is an operating system and language-independent programming interface to general communication services. Detailed information about the XAPI is given in the main part of this Recommendation, which is very important for understanding this appendix.

Which services are made available via the XAPI depends on the installed service providers and not on the XAPI which only provides the access mechanism.

This part describes the XAPI access to the T.120 conference control. The Generic Conference Control (GCC) protocol is specified in Recommendation T.124.

Figure I.6-1 shows the structure of the protocol stack that is accessible via the XAPI when selecting the conference control provider.

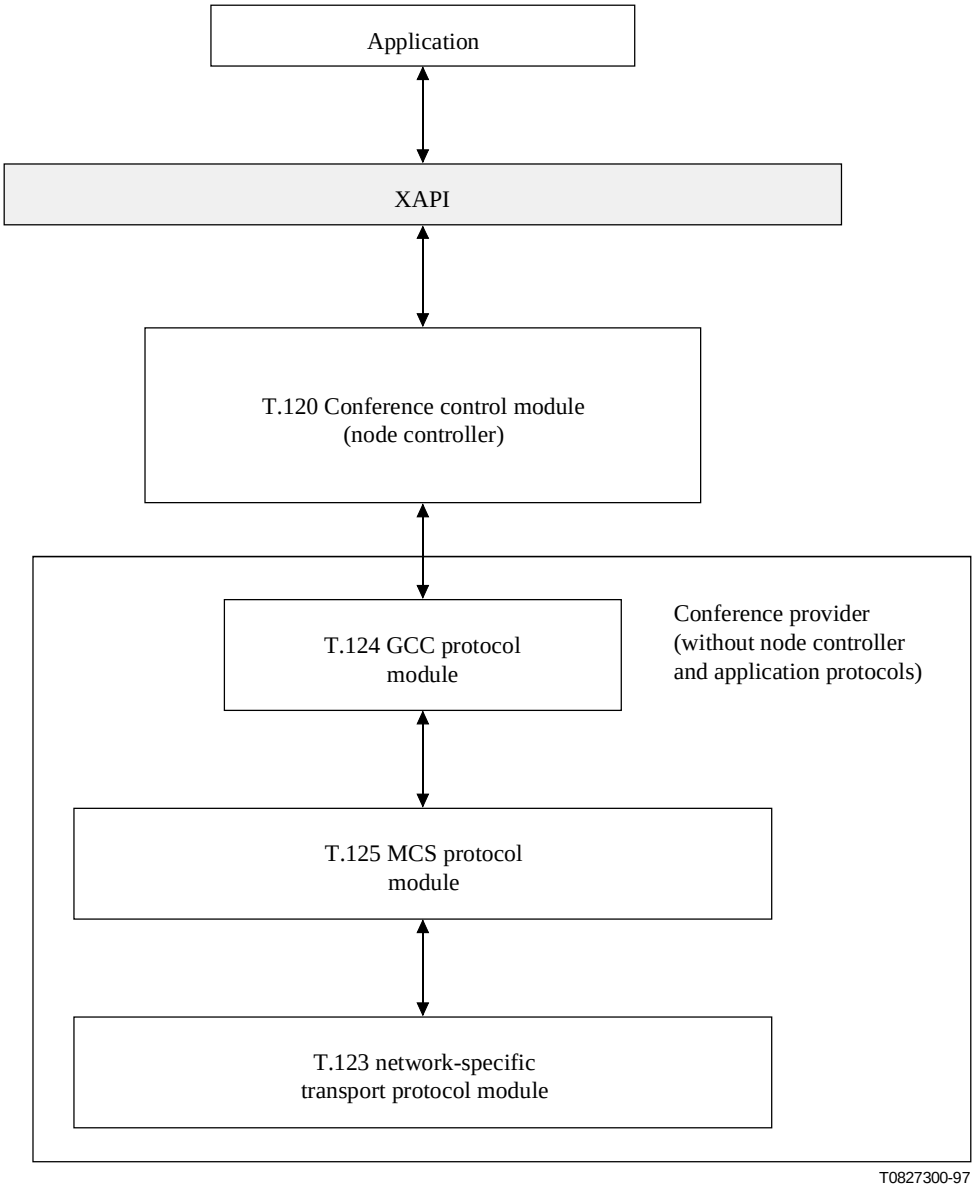


Figure I.6-1/T.180 – Structure of the T.120 conference control provider

The XAPI user is able to select one transport system (comprising the layers 1 to 4) among the set of transport systems available in the XAPI communication platform to act as the underlying transport service provider.

The conference control provider consists of the T.120 conference control module (node controller) and the T.124 GCC protocol module, the T.125 MCS protocol module, and the T.123 Network-specific-transport protocol stack modules.

The conference provider comprises the T.124 GCC protocol module, the T.125 MCS protocol module, the T.123 Network-specific transport protocol stack modules, the node controller module and the application protocol modules (e.g. the T.127 MBFT protocol module).

When a user (at the terminal) is connected to a conference, all modules of the conference provider are initialized and activated. First the node controller module, the T.124 GCC protocol module, the T.125 MCS protocol module, and the T.123 Network-specific transport protocol stack modules will be initialized and activated. Afterwards, applications may be connected to the conference (see I.7, "XAPI access to the service provider for T.127 MBFT").

The reader should be familiar with the T.120-series of Recommendations (see References).

I.6.2 References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; all users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published.

[T.120] ITU-T Recommendation T.120 (1996), *Data protocols for multimedia conferencing*.

[T.122] ITU-T Recommendation T.122 (1993), *Multipoint communication service – Service definition*.

[T.123] ITU-T Recommendation T.123 (1996), *Network specific data protocol stacks for multimedia conferencing*.

[T.124] ITU-T Recommendation T.124 (1998), *Generic Conference Control*.

[T.125] ITU-T Recommendation T.125 (1998), *Multipoint communication service protocol specification.*

I.6.3 Definitions

I.6.4 Abbreviations

This part uses the following abbreviations:

DSS 1 Digital Subscriber Signalling System No.1

GCC Generic Conference Control

HDLC High-Level Data-Link Control

ISDN Integrated Services Digital Network

MCS Multipoint Communication Service

MCU Multipoint Control Unit

MBFT Multipoint Binary File Transfer

PDU Protocol Data Unit

SP Service Primitive

XAPI eXtensive Application Programming Interface

I.6.5 Conventions

Each service is described via three kinds of tables.

In the first kind of table the service and its service elements are described. It contains a row for each service element of the service (where service element means either request, indication, response, or confirmation) with its corresponding XAPI function and a short description.

In the second kind of table the use of the service primitive parameters within the service elements are described. It contains a row for each service primitive parameter of the service. The service primitive parameter name is stored within the first column; the succeeding columns contain the use of the parameter in the service elements:

Blank The service parameter is absent.

C Presence of the service parameter is conditional. Firstly, there may be a condition in the service provider to provide a parameter in an indication or confirmation; secondly, there may be interdependencies between parameters of the same or the preceding service primitive.

M Presence of the service parameter is mandatory.

U Presence of the service parameter is a user option. If the user does not specify a value for such a parameter and there is no default value for that parameter, nothing is passed to the service provider. If the user does not specify a value for such a parameter and there is a default value, the default value is passed to the service provider.

(=) The value of the service parameter is identical to the value of the corresponding service parameter in the preceding service element. In the special case of a parameter, whose presence in the preceding service is a user option, for which a default value is defined, and the parameter was not specified in the preceding service element, the symbol (=) indicates that the parameter value is identical to the default value.

The third kind of table is used to describe the service primitive parameter containing the type of value and its scope, possible default values and a detailed description. For each service primitive parameter one description table is given.

A **naming convention** is used throughout the XAPI in order to distinguish between the common functions, identifiers and values, and service-specific identifiers and values. All service-specific settings of the node controller service access are defined within the present part of this appendix and start with **X_CON_** or **x_con_**.

Extended naming conventions:

– SP_ Service Primitive;

– P_ Parameter;

– PV_ Parameter Value;

– O_ Option;

– OV_ Option Value.

I.6.6 Introduction to the conference control provider access

The conference control provider enables a user to participate in a conference.

The service is provided by a combination of a transport system according to Recommendation T.123, the MCS protocol module according to Recommendations T.125 and T.122, the GCC protocol module according to Recommendation T.124, and functions of the node controller.

The T.120 conference control provider offers the XAPI user the following groups of services as defined in the T.124 GCC service:

- conference establishment and termination;
- conference roster;
- application roster;
- conference conductorship;
- miscellaneous functions.

All mandatory and some conditional services of the T.124 GCC service are supported. The following services of the above specified groups are supported:

- *Conference establishment and termination* – conference creation, conference query, joining a conference, invitation to a conference (passive only), leaving a conference, indication of ejection of a user, indication of termination of a conference;

- *Conference roster* – indication of the actual conference roster, announce of the presence in a conference (this service is not visible to the XAPI user);
- *Application roster* – application roster report indication;
- *Conference conductorship* – indication of conductor assign and release, information about the conference conductor, if any;
- *Miscellaneous functions*.

Recommendation T.120 supports the following configurations:

- direct connection between two terminals without MCU

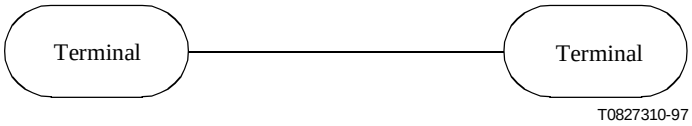


Figure I.6-2/T.180 – Example configuration: Direct connection between two terminals without MCU

- connection between terminals and multiport terminals without MCUs

Figure I.6-3/T.180 – Example configuration: Connection between terminals and multiport terminals without MCUs

- hierarchical configuration of MCUs and terminals

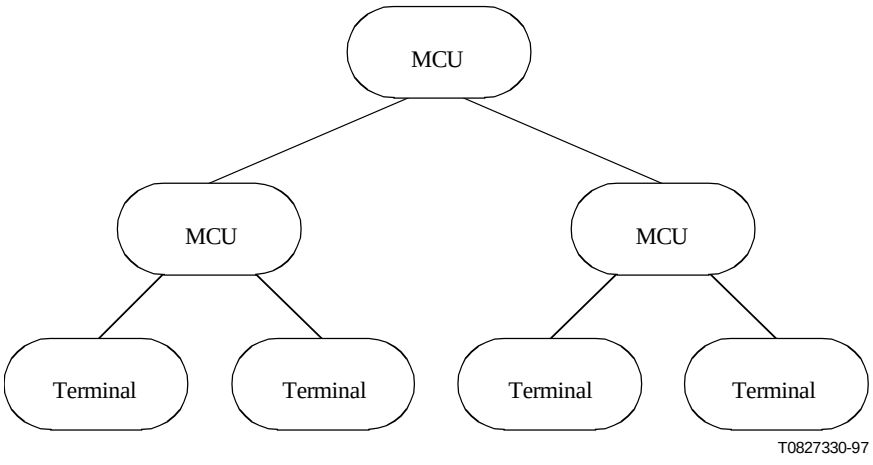


Figure I.6-4/T.180 – Example configuration: Hierarchical configuration of MCUs and terminals

Restrictions, which are made for implementation:

Only the hierarchical configuration of MCUs and terminals is supported. Each terminal is directly connected to a MCU (see Figure I.6-4).

The XAPI interface is taken into consideration only in a terminal. Therefore, only GCC services which correspond to a terminal are allowed.

Optional and most conditional services are not supported:

- *Conference establishment and termination* – active invitation to conference, lock or unlock the conference, information about change between locked or unlocked, active termination of a conference, active ejection of a user, active conference transfer;
- *Conference roster* – information about the actual conference roster;
- *Application roster* – active or passive invocation of an application;
- *Conference conductorship* – active conductor assign, active conductor release, conductor please, conductor give, conductor permission ask and grant;
- *Miscellaneous functions* – conference time remaining, conference time inquire, conference extend, conference assistance, conference text message.

I.6.7 Description of the access to the conference control provider

I.6.7.1 Service initialization

I.6.7.1.1 Creation of a conference control service access point with x_open()

A communication endpoint accessing conference control service provider is created when calling the *x_open()* function with an appropriate service provider identification string. The available identifiers depend on the actual system configuration. In the standard configuration, "**X_T.120_CONF_ISDN**" identifies the conference control provider with the node controller, T.124 GCC and T.122/T.125 MCS as underlying functionality as well as a transport system based on the ISDN network. On the protocol levels two to four, there are implementations of HDLC LAP B, ISO 8208 and T.70.

I.6.7.1.2 Selection of the underlying transport system with x_bind()

x_bind() is to be called to activate the conference control service endpoint. The function has the following tasks:

- If the service provider, which has been selected with *x_open()*, does not comprise a transport system, link a transport system below the available protocol modules;
- Bind an address to the service endpoint.

If the service provider X_T.120_CONF_ISDN has been selected in the *x_open()* function, no transport system has to be specified in the *x_bind()* function.

I.6.7.1.3 Protocol addresses

The protocol address to be used for the conference control service is the NSAP address. Selectors are meaningless for the conference control service.

I.6.7.1.3.1 The application's own address

The own address may be specified in the *own_address* buffer of the *bind_struct* passed as argument to the *x_bind()* function. It is returned in the *called_addr* buffer of the *x_conind()* function.

For a passive application it is not supported to specify the own responding NSAP address in the *address* buffer of the *call_struct* in the *x_conrsp()* function, as this value is not transferred by the network.

Note that specification of the application's own protocol address is completely optional. If no address information is specified, the own address is derived from system configuration information and the bound value is returned as output parameter of the *x_bind()* function.

The own address consists of the NSAP address only. The NSAP address has to comprise the local ISDN outband address, i.e. the address information used in the D-channel. The own ISDN inband address and subaddress parameters as well as the protocol selectors are meaningless for the conference control service. If specified, they will be ignored.

Table I.6-1 shows the address component which has to be specified in the *x_bind()* call:

Table I.6-1/T.180 – Address component specified in the *x_bind()* call

ISDN network	Address component	Value
ISDN/DSS 1	A_OUTBAND_ADR	A decimal digit, which is locally mapped on the Multiple Subscriber Number (MSN)

I.6.7.1.3.2 The address of the communication partner

On the sending side, the address of the communication partner has to be specified in the *address* buffer of the *call_struct* passed as argument to the *x_conreq()* function. On the receiving side, the address of the communication partner is returned in the *calling_addr* buffer of the *x_conind()* function.

The address of the communication partner comprises at least the peer's ISDN outband address. The peer's ISDN inband address and subaddress as well as the protocol selectors are meaningless for the conference control service.

Table I.6-2 shows the address component to be used in the *address* buffer specifying the called NSAP address in a *x_conreq()* call:

Table I.6-2/T.180 – Address component specifying the called NSAP address in a *x_conreq()* call

ISDN network	Address component	Value
ISDN/DSS 1	A_OUTBAND_ADR	Optional the country code, optional the area code and the Multiple Subscriber Number (MSN)

I.6.7.1.4 Configuration of the service provider

Protocol options are used to control the general behavior of the service provider. There is a default value defined for each option. These preconfigured values are sufficient for the majority of communication relations.

XAPI enables applications to express their wishes about the settings of protocol options by using the XAPI function *x_optmgmt()*.

Tables I.6-3 and I.6-4 define the protocol options.

Table I.6-3/T.180

Option name	X_CON_O_AROSTER
Type of value	unsigned long
Legal values	PV_TRUE PV_FALSE
Default value	PV_TRUE
Attributes	
Description	Receiving of application roster indications.

Table I.6-4/T.180

Option name	X_CON_O_CROSTER
Type of value	unsigned long
Legal values	PV_TRUE PV_FALSE
Default value	PV_TRUE
Attributes	
Description	Receiving of Conference Roster Indications.

I.6.7.2 Connection Establishment service

I.6.7.2.1 Service description

The Connection Establishment service enables the XAPI user to become a participant of a conference. There are two possibilities to become a participant:

- The XAPI user itself will actively become a participant of a conference. In this case the x_conreq and x_conconf connection establishment service elements are used.
- The XAPI user is invited to a conference. In this case the x_conind and x_conrsp connection establishment service elements are used.

When the terminal is a participant of the conference, all supported applications (e.g. T.127 MBFT) can be started.

The elements of the Connection Establishment service and the corresponding XAPI functions for the active participation are defined in Table I.6-5.

Table I.6-5/T.180 – Service elements and their corresponding XAPI functions for Connection Establishment for active participation

Service element	XAPI function	Description
Connect Request	x_conreq()	The Connect Request is passed to the provider to request the establishment of a connection to the Top provider of the conference. The own local capabilities to be used may be specified as service parameters. If no capabilities are specified, the default values are used.
Connect Confirmation	x_conconf()	The Connect Confirmation is a positive or negative response to a previous connection establishment request. A positive confirmation indicates that the terminal is a participant of the conference.

The elements of the Connection Establishment service and the corresponding XAPI functions for invitation to participate in a conference are defined in Table I.6-6.

Table I.6-6/T.180 – Service elements and their corresponding XAPI functions for Connection Establishment for invitation to participate

Service element	XAPI function	Description
Connect Indication	x_conind()	The Connect Indication is passed from the provider to the user to request the establishment of a connection to the conference. The own local capabilities to be used may be specified as service parameters. If no capabilities are specified, the default values are used.
Connect Response	x_conrsp()	The Connect Response is a positive or negative response to a previous connection establishment indication. A positive response indicates that the terminal will be a participant of the conference.

I.6.7.2.2 Service parameters for active participation

Table I.6-7 specifies the parameters of the Connection Establishment service for active participation in a conference.

The specification of provider capabilities is optional. If a capability parameter is not specified in a service element, the default value will be used by the service provider. The default value itself is defined by the value of a protocol option. For each capability parameter there is a corresponding protocol option which defines the default value for this parameter. For the protocol option, a constant default value is defined in the XAPI configuration.

Table I.6-7/T.180 – Parameters of the active participation service

Parameter	Connect service	
	Request	Confirmation
X_CON_P_CR_JOIN	M	
X_CON_P_CONF_NAME	M	M (=)
X_CON_P_CNAME_M_CD	C	
X_CON_P_CNAME_M_CG	U	
X_CON_P_CONV_PASS	U	
X_CON_P_PASS	C	C
X_CON_P_PASS_CLEAR		C
X_CON_P_LOCKED	C	C
X_CON_P_LISTED	C	C
X_CON_P_CONDUCT	C	C
X_CON_P_TERM_MOD	C	C
X_CON_P_COND_PRIV	C	C
X_CON_P_COND_M_PRIV	C	C
X_CON_P_NCOND_M_PRIV	C	C
X_CON_P_CONF_DESCR	C	C
X_CON_P_CALLER_ID	U	
X_CON_P_LOC_NETADDR	U	
X_CON_P_NODE_NAME	U	
X_CON_P_PART_NAME	U	
X_CON_P_SITE_INFO	U	
X_CON_P_NODE_ID		C
X_CON_P_DATA_PRI	U	C
X_CON_P_RESULT		M

I.6.7.2.3 Service parameter description for active participation

Tables I.6-8 to I.6-30 define the parameters for the elements x_conreq and x_conconf of the Connection Establishment service.

Table I.6-8/T.180

Parameter name	X_CON_P_CR_JOIN
Type of value	unsigned long
Legal values	X_CON_PV_CREATE X_CON_PV_JOIN
Default value	X_CON_PV_CREATE
Description	The parameter indicates if a new conference shall be created or if the user wishes to join an existing conference.

Table I.6-9/T.180

Parameter name	X_CON_P_CONF_NAME

Type of value	struct X_CON_P_CONF_NAME {
	string<256> numeric_string;
	string<256> text_string;
	};

Legal values	a numerical string along with an optional text string, from zero to 255 characters each
Default value	none
Description	Name by which the conference is identified

Table I.6-10/T.180

Parameter name	X_CON_P_CNAME_M_CD
Type of value	string<255>
Legal values	a numerical string up to 255 digits in length
Default value	none
Description	Name Modifier by which the conference is identified in the called node if a conference with identical conference name already exists. This parameter is only meaningful if joining a conference (X_CON_P_CR_JOIN = X_CON_PV_JOIN).

Table I.6-11/T.180

Parameter name	X_CON_P_CNAME_M_CG
Type of value	string<255>
Legal values	a numerical string up to 255 digits in length
Default value	none
Description	Name Modifier by which the conference is identified in the calling node if a conference with identical conference name already exists

Table I.6-12/T.180

Parameter name	X_CON_P_CONV_PASS

Type of value	struct X_CON_P_CONV_PASS {
	string<256> numeric_string;
	string<256> text_string;
	};

Legal values	a numerical string along with an optional text string, from zero to 255 characters each
Default value	None
Description	Password used by the convener to identify itself. It can be used in later operations, when he rejoins the conference.

Table I.6-13/T.180

Parameter name	X_CON_P_PASS
Type of value	struct X_CON_P_PASS { string<256> numeric_string; string<256> text_string; };string
Legal values	a numerical string along with an optional text string, from zero to 255 characters each
Default value	none
Description	Password for a password-protected conference. If the XAPI user creates the conference (X_CON_P_CR_JOIN = X_CON_PV_CREATE), the conference may be protected with a password in the x_conreq function. If the XAPI user joins the conference (X_CON_P_CR_JOIN = X_CON_PV_JOIN), the x_conreq function must contain this parameter for a password-protected conference. If the password is not o.k., in this case the x_conconf function may contain more information about the password.

Table I.6-14/T.180

Parameter name	X_CON_P_PASS_CLEAR
Type of value	unsigned long
Legal values	PV_TRUE PV_FALSE
Default value	none
Description	It indicates whether it is a password-protected conference. This parameter is used only in the x_conconf function, when it is a response to a x_conreq with the parameter (X_CON_P_CR_JOIN = X_CON_PV_JOIN).

Table I.6-15/T.180

Parameter name	X_CON_P_LOCKED
Type of value	unsigned long
Legal values	PV_TRUE PV_FALSE
Default value	PV_FALSE
Description	Setting this parameter indicates that the conference is locked. Nobody can join the conference by itself. New conference participants can only be invited. In the x_conreq function the value PV_TRUE is allowed only in the case when creating a conference (i.e. X_CON_P_CR_JOIN = X_CON_PV_CREATE).

Table I.6-16/T.180

Parameter name	X_CON_P_LISTED
Type of value	unsigned long
Legal values	PV_TRUE PV_FALSE
Default value	PV_TRUE
Description	This parameter indicates that this conference may be listed when using the conference information service. This parameter must be used in the x_conreq function when a conference is created (X_CON_P_CR_JOIN = X_CON_PV_CREATE) but not when joining a conference (X_CON_P_CR_JOIN = X_CON_PV_JOIN). This parameter is used in the x_conconf function when a conference is joined (X_CON_P_CR_JOIN = X_CON_PV_JOIN) but not when creating a conference (X_CON_P_CR_JOIN = X_CON_PV_CREATE).

Table I.6-17/T.180

Parameter name	X_CON_P_CONDUCT
Type of value	PV_TRUE PV_FALSE
Legal values	PV_FALSE
Default value	none
Description	This parameter indicates that the conference may be placed in the conducted mode. This parameter must be used in the x_conreq function when a conference is created (X_CON_P_CR_JOIN = X_CON_PV_CREATE) but not when joining a conference (X_CON_P_CR_JOIN = X_CON_PV_JOIN). This parameter is used in the x_conconf function when a conference is joined (X_CON_P_CR_JOIN = X_CON_PV_JOIN) but not when creating a conference (X_CON_P_CR_JOIN = X_CON_PV_CREATE).

Table I.6-18/T.180

Parameter name	X_CON_P_TERM_MOD
Type of value	unsigned long
Legal values	X_CON_PV_AUTOMATIC X_CON_PV_MANUAL
Default value	X_CON_PV_AUTOMATIC
Description	This parameter indicates whether the conference shall remain in existence until explicitly terminated (X_CON_PV_MANUAL) or if the conference will terminate in the case no nodes are joined to it (X_CON_PV_AUTOMATIC). This parameter must be used in the x_conreq function when a conference is created (X_CON_P_CR_JOIN = X_CON_PV_CREATE) but not when joining a conference (X_CON_P_CR_JOIN = X_CON_PV_JOIN). This parameter is used in the x_conconf function when a conference is joined (X_CON_P_CR_JOIN = X_CON_PV_JOIN) but not when creating a conference (X_CON_P_CR_JOIN = X_CON_PV_CREATE).

Table I.6-19/T.180

Parameter name	X_CON_P_COND_PRIV
Type of value	sequence<unsigned long, 6>
Legal values	PV_TRUE / PV_FALSE Conference termination; PV_TRUE / PV_FALSE Eject a user; PV_TRUE / PV_FALSE Add a user; PV_TRUE / PV_FALSE Lock a conference; PV_TRUE / PV_FALSE Unlock a conference; PV_TRUE / PV_FALSE Conference transfer.
Default Value	PV_TRUE, PV_TRUE, PV_TRUE, PV_TRUE, PV_TRUE, PV_TRUE
Description	This parameter indicates which functions the convener is designating as allowable to be used by the conference conductor, if any. The values are specified in the same order as listed above. This parameter may be used in the x_conreq function when a conference is created (X_CON_P_CR_JOIN = X_CON_PV_CREATE) but not when joining a conference (X_CON_P_CR_JOIN = X_CON_PV_JOIN). This parameter may be in the x_conconf function when a conference is joined (X_CON_P_CR_JOIN = X_CON_PV_JOIN) but not when creating a conference (X_CON_P_CR_JOIN = X_CON_PV_CREATE).

Table I.6-20/T.180

Parameter name	X_CON_P_COND_M_PRIV
Type of value	sequence<unsigned long, 6>
Legal values	PV_TRUE / PV_FALSE Conference termination; PV_TRUE / PV_FALSE Eject a user; PV_TRUE / PV_FALSE Add a user; PV_TRUE / PV_FALSE Lock a conference; PV_TRUE / PV_FALSE Unlock a conference; PV_TRUE / PV_FALSE Conference transfer.
Default value	PV_TRUE, PV_TRUE, PV_TRUE, PV_TRUE, PV_TRUE, PV_TRUE
Description	This parameter indicates which functions the convener is designating as allowable to be used by any node in a conducted-mode conference. The values are specified in the same order as listed above. This parameter may be used in the x_conreq function when a conference is created (X_CON_P_CR_JOIN = X_CON_PV_CREATE) but not when joining a conference (X_CON_P_CR_JOIN = X_CON_PV_JOIN). This parameter may be in the x_conconf function when a conference is joined (X_CON_P_CR_JOIN = X_CON_PV_JOIN) but not when creating a conference (X_CON_P_CR_JOIN = X_CON_PV_CREATE).

Table I.6-21/T.180

Parameter name	X_CON_P_NCOND_M_PRIV
Type of value	sequence <unsigned long, 6>
Legal values	PV_TRUE / PV_FALSE Conference termination; PV_TRUE / PV_FALSE Eject a user; PV_TRUE / PV_FALSE Add a user; PV_TRUE / PV_FALSE Lock a conference; PV_TRUE / PV_FALSE Unlock a conference; PV_TRUE / PV_FALSE Conference transfer.
Default value	PV_TRUE, PV_TRUE, PV_TRUE, PV_TRUE, PV_TRUE, PV_TRUE
Description	This parameter indicates which functions the convener is designating as allowable to be used by any node in a non-conducted-mode conference. The values are specified in the same order as listed above. This parameter may be used in the x_conreq function when a conference is created (X_CON_P_CR_JOIN = X_CON_PV_CREATE) but not when joining a conference (X_CON_P_CR_JOIN = X_CON_PV_JOIN). This parameter may be in the x_conconf function when a conference is joined (X_CON_P_CR_JOIN = X_CON_PV_JOIN) but not when creating a conference (X_CON_P_CR_JOIN = X_CON_PV_CREATE).

Table I.6-22/T.180

Parameter name	X_CON_P_CONF_DESCR
Type of value	string<256>
Legal values	a text string, from zero to 255 characters
Default value	none
Description	This parameter may be used to describe the conference. This parameter may be used in the x_conreq function when a conference is created (X_CON_P_CR_JOIN = X_CON_PV_CREATE) but not when joining a conference (X_CON_P_CR_JOIN = X_CON_PV_JOIN). This parameter may be in the x_conconf function when a conference is joined (X_CON_P_CR_JOIN = X_CON_PV_JOIN) but not when creating a conference (X_CON_P_CR_JOIN = X_CON_PV_CREATE).

Table I.6-23/T.180

Parameter name	X_CON_P_CALLER_ID
Type of value	string<256>
Legal values	a text string, from zero to 255 characters
Default value	none
Description	This parameter may be used to describe the calling node.

Table I.6-24/T.180

Parameter name	X_CON_P_LOC_NETADDR
Type of value	unsigned long
Legal values	PV_TRUE PV_FALSE
Default value	PV_FALSE
Description	This parameter is used to indicate whether the local provider shall send the own Network Address parameter or not. This parameter is included in the conference description list.

Table I.6-25/T.180

Parameter name	X_CON_P_NODE_NAME
Type of value	string<255>
Legal values	a string up to 255 characters
Default value	none
Description	This parameter is used to indicate the name of a node. This parameter is included in the conference description list.

Table I.6-26/T.180

Parameter name	X_CON_P_PART_NAME
Type of value	sequence<string<255> >
Legal values	a list of strings each up to 255 characters
Default value	none
Description	This parameter is used to indicate the names of the participants.

Table I.6-27/T.180

Parameter name	X_CON_P_SITE_INFO
Type of value	string<255>
Legal values	a string up to 255 characters
Default value	none
Description	This parameter is used to indicate information about the node. This parameter is included in the conference description list.

Table I.6-28/T.180

Parameter name	X_CON_P_NODE_ID
Type of value	unsigned long
Legal values	an integer between 1001 and 65535
Default value	none
Description	This parameter is used to indicate the own node identification from the local provider to the user.

Table I.6-29/T.180

Parameter name	X_CON_P_DATA_PRI
Type of value	unsigned long
Legal values	an integer between 1 and 4
Default value	1
Description	This parameter is used to indicate the number of data transfer priorities.

Table I.6-30/T.180

Parameter name	X_CON_P_RESULT	
Type of value	unsigned long	
Legal values	X_CON_PV_ACCEPT	Accepted;
	X_CON_PV_USER_REJECTED	User rejected;
	X_CON_PV_RESOURCE_NOT_AVAIL	Resources not available;
	X_CON_PV_SYMMETRY_BREAK	Symmetry breaking;
	X_CON_PV_ILOCKED_NOT_SUPP	Locked conference not supported;
	X_CON_PV_NAME/MOD_EXIST	Conference name and modifier already exist;
	X_CON_PV_DOM_PAR_UNACC	Domain parameters unacceptable;
	X_CON_PV_DOM_NOT_HIERARCH	Domain not hierarchical;
	X_CON_PV_LOWER_LAY_DIS	Lower layer initiated disconnect;
	X_CON_PV_UNSPECIFIED	Unspecified failure;
	X_CON_PV_INV_CONF	Invalid conference;
	X_CON_PV_INV_PASSW	Invalid password;
	X_CON_PV_CHALLENGE_RSP_REQ	Challenge response required;
	X_CON_PV_INV_CHALLENGE_RSP	Invalid challenge response;
	X_CON_PV_INV_CONVENER_PASSW	Invalid convener password.
Default value	X_CON_PV_ACCEPT	
Description	This parameter indicates the success or failure of the connection establishment request.	

I.6.7.2.4 Service parameters for invitation to participate in a conference

Table I.6-31/T.180 – Parameters for invitation to participate service

Parameter	**Connect service**	
	Indication	**Response**
X_CON_P_CONF_NAME	M	M (=)
X_CON_P_CNAME_MOD		C
X_CON_P_CALLER_ID	U	
X_CON_P_PASS_CLEAR	M	
X_CON_P_LOCKED	M	
X_CON_P_LISTED	M	

Table I.6-31/T.180 – Parameters for invitation to participate service *(concluded)*

Parameter	Connect service	
	Indication	**Response**
X_CON_P_CONDUCT	M	
X_CON_P_TERM_MOD	M	
X_CON_P_COND_PRIV	C	
X_CON_P_COND_M_PRIV	C	
X_CON_P_NCOND_M_PRIV	C	
X_CON_P_CONF_DESCR	C	
X_CON_P_LOC_NETADDR		U
X_CON_P_DATA_PRI	C	C
X_CON_P_RESULT		M

I.6.7.2.5 Service parameter description for invitation to participate in a conference

Tables I.6-32 to I.6-46 define the parameters for the elements x_conind and x_conrsp of the Connection Establishment service.

Table I.6-32/T.180

Parameter name	X_CON_P_CONF_NAME

Type of value	struct X_CON_P_CONF_NAME {
	string<256> numeric_string;
	string<256> text_string;
	};

Legal values	a numerical string along with an optional text string, from zero to 255 characters each
Default value	none
Description	Name by which the conference is identified.

Table I.6-33/T.180

Parameter name	X_CON_P_CNAME_MOD
Type of value	string<255>
Legal values	a numerical string up to 255 digits in length
Default value	none
Description	Name Modifier by which the conference is identified in the called node if a conference with identical conference name already exists.

Table I.6-34/T.180

Parameter name	X_CON_P_CALLER_ID
Type of value	string<256>
Legal values	This parameter is a text string, from zero to 255 characters.
Default value	none
Description	This parameter may be used to describe the calling node.

Table I.6-35/T.180

Parameter name	X_CON_P_PASS_CLEAR
Type of value	unsigned long
Legal values	PV_TRUE PV_FALSE
Default value	none
Description	It indicates whether it is a password-protected conference.

Table I.6-36/T.180

Parameter name	X_CON_P_LOCKED
Type of value	unsigned long
Legal values	PV_TRUE PV_FALSE
Default value	PV_FALSE
Description	Setting this parameter indicates that the conference is locked. Nobody can join the conference itself. A new conference participant can only be invited.

Table I.6-37/T.180

Parameter name	X_CON_P_LISTED
Type of value	unsigned long
Legal values	PV_TRUE PV_FALSE
Default value	PV_TRUE
Description	This parameter indicates that this conference may be listed when using the conference information service.

Table I.6-38/T.180

Parameter name	X_CON_P_CONDUCT
Type of value	PV_TRUE PV_FALSE
Legal values	PV_FALSE
Default value	none
Description	This parameter indicates that the conference may be placed in the conducted mode.

Table I.6-39/T.180

Parameter name	X_CON_P_TERM_MOD
Type of value	unsigned long
Legal values	X_CON_PV_AUTOMATIC X_CON_PV_MANUAL
Default value	X_CON_PV_AUTOMATIC
Description	This parameter indicates whether the conference shall remain in existence until explicitly terminated (X_CON_PV_MANUAL) or if the conference will terminate when there are no nodes joined to it (X_CON_PV_AUTOMATIC).

Table I.6-40/T.180

Parameter name	X_CON_P_COND_PRIV
Type of value	sequence <unsigned long>
Legal values	PV_TRUE / PV_FALSE Conference termination; PV_TRUE / PV_FALSE Eject a user; PV_TRUE / PV_FALSE Add a user; PV_TRUE / PV_FALSE Lock a conference; PV_TRUE / PV_FALSE Unlock a conference; PV_TRUE / PV_FALSE Conference transfer.
Default value	PV_TRUE , PV_TRUE , PV_TRUE , PV_TRUE , PV_TRUE , PV_TRUE
Description	This parameter indicates which functions the convener has designated as allowable to be used by the conference conductor, if any. The values are specified in the same order as listed above.

Table I.6-41/T.180

Parameter name	X_CON_P_COND_M_PRIV
Type of value	sequence<unsigned long, 6>
Legal values	PV_TRUE / PV_FALSE Conference termination; PV_TRUE / PV_FALSE Eject a user; PV_TRUE / PV_FALSE Add a user; PV_TRUE / PV_FALSE Lock a conference; PV_TRUE / PV_FALSE Unlock a conference; PV_TRUE / PV_FALSE Conference transfer.
Default value	PV_TRUE, PV_TRUE, PV_TRUE, PV_TRUE, PV_TRUE, PV_TRUE
Description	This parameter indicates which functions the convener has designated as allowable to be used by any node in a conducted-mode conference. The values are specified in the same order as listed above.

Table I.6-42/T.180

Parameter name	X_CON_P_NCOND_M_PRIV
Type of value	sequence <unsigned long, 6>
Legal values	PV_TRUE / PV_FALSE Conference termination; PV_TRUE / PV_FALSE Eject a user; PV_TRUE / PV_FALSE Add a user; PV_TRUE / PV_FALSE Lock a conference; PV_TRUE / PV_FALSE Unlock a conference; PV_TRUE / PV_FALSE Conference transfer.
Default value	PV_TRUE, PV_TRUE, PV_TRUE, PV_TRUE, PV_TRUE, PV_TRUE
Description	This parameter indicates which functions the convener has designated as allowable to be used by any node in a non-conducted-mode conference. The values are specified in the same order as listed above.

Table I.6-43/T.180

Parameter name	X_CON_P_CONF_DESCR
Type of value	string<256>
Legal values	This parameter is a text string, from zero to 255 characters.
Default value	none
Description	This parameter may be used to describe the conference.

Table I.6-44/T.180

Parameter name	X_CON_P_LOC_NETADDR
Type of value	unsigned long
Legal values	PV_TRUE PV_FALSE
Default value	PV_FALSE
Description	This parameter is used to indicate whether the local provider shall send the own Network Address parameter or not. This parameter is included in the conference description list.

Table I.6-45/T.180

Parameter name	X_CON_P_DATA_PRI
Type of value	unsigned long
Legal values	an integer between 1 and 4
Default value	1
Description	This parameter is used to indicate the number of data transfer priorities.

Table I.6-46/T.180

Parameter name	X_CON_P_RESULT	
Type of value	unsigned long	
Legal values	X_CON_PV_ACCEPT	Accepted;
	X_CON_PV_USER_REJECT	User rejected;
	X_CON_PV_UNSPECIFIED	Unspecified failure.
Default value	X_CON_PV_ACCEPT	
Description	This parameter indicates the success or failure of the connection establishment request.	

In the non-connected state there is also the Query Information service available.

I.6.7.3 Services in the connected state

There are no services available except all information services.

I.6.7.4 Disconnect service

I.6.7.4.1 Service description

The Disconnect service enables the XAPI user to leave a conference. There are two possibilities to leave the conference:

- The XAPI user itself will actively leave a conference. In this case the x_snddis XAPI function is used.
- The XAPI user is disconnected from a conference. In this case the x_rcvdis XAPI function is used.

The service elements of the Disconnect service and the corresponding XAPI functions are defined in Table I.6-47.

Table I.6-47/T.180 – Service elements and their corresponding XAPI functions for the Disconnect service

Service element	XAPI function	Description
Disconnect Request	x_snddis()	The Disconnect Request is passed to the provider to disconnect itself from the conference.
Disconnect Indication	x_rcvdis()	The Disconnect Indication is passed to the user to indicate that the node is no longer a participant of the conference.
End Indication	x_rcvend()	The End Indication is generated by the provider to indicate that the service provider is ready to establish a new connection.

I.6.7.4.2 Service parameters

Table I.6-48 specifies the parameters of the Disconnect service.

Table I.6-48/T.180 – Parameters of the Disconnect service

Parameter	Disconnect service	
	Request	Indication
X_CON_P_TERM_NODE		C
X_CON_P_REASON		C

I.6.7.4.3 Service parameter description

Tables I.6-49 and I.6-50 define the parameters for the functions x_snddis and x_rcvdis of the Disconnect service.

Table I.6-49/T.180

Parameter name	X_CON_P_TERM_NODE
Type of value	unsigned long
Legal values	an integer between 1001 and 65535
Default value	none
Description	This parameter indicates the node ID of that node which was requesting the termination.

Table I.6-50/T.180

Parameter name	X_CON_P_REASON	
Type of value	unsigned long	
Legal values	X_CON_PV_USER_INI	User initiated;
	X_CON_PV_EJECTED_NODE	Ejected node;
	X_CON_PV_UNKNOWN	Unknown;
	X_CON_PV_NORM_TERM	Requested normal termination;
	X_CON_PV_TIMED_TERM	Requested timed conference termination;
	X_CON_PV_NO_PART	No more participants in automatically terminating conference;
	X_CON_PV_ERROR	Error termination;
	X_CON_PV_HNODE_DIS	Higher node disconnected;
	X_CON_PV_HNODE_EJECT	Higher node ejected.
Default value	none	
Description	This parameter indicates the reason for the disconnection.	

In the non-connected state there is also the Query Information service available.

I.6.7.5 Conference Information service

I.6.7.5.1 Conference Query service

This service may be used to determine what conferences are currently in existence at a particular MCU. This service is available to the user in every state of the XAPI.

I.6.7.5.1.1 Service description

The service elements and their corresponding XAPI functions needed for the Conference Query service are described in Table I.6-51.

Table I.6-51/T.180 – Service elements and their corresponding XAPI functions for the Query service

Service element	Service element identifier	XAPI function	Description
Query Request	X_CON_QUERY_Q	x_sndinfo()	The Information Request is passed to the provider to request the information about actual listed conferences in the requested MCU.
Query Confirmation	X_CON_QUERY_C	x_rcvinfo()	The Information Confirmation is a positive or negative response to a previous information request.

I.6.7.5.1.2 Service parameters

Table I.6-52 specifies the parameters of the Query service.

Table I.6-52/T.180 – Parameters of the Query service

Parameter	Conference Query service	
	Request	Confirmation
X_CON_P_DESCR_LIST		C
X_CON_P_USER_DATA	O	O
X_CON_P_RESULT		M

I.6.7.5.1.3 Service parameter description

Tables I.6-53 to I.6-55 specify the parameters of the Query service.

Table I.6-53/T.180

Parameter name	X_CON_P_DESCR_LIST		

Type of value	typedef struct Description {
	string Conf_Name;
	string Name_Modifier;
	string Conf_Descript;
	unsigned long Lock_Unlock;
	unsigned long Passw_in_Clear;
	string Net_Addr;
	};
	sequence<Description>;		

Legal values	Conf_Name:	A text string	Conference name;
	Name_Modifier:	A text string	Conference name modifier;
	Conf_Descript:	A text string	Conference description;
	Lock_Unlock:	PV_TRUE,PV_FALSE	Conference locked or unlocked;
	Passw_in_Clea	PV_TRUE,PVFALSE	Password in the clear required;
	Net_Addr:	A text string	Network address.
Default value	none		
Description	This parameter contains the conference description list with the specified parameters for each listed conference in the requested node.		

Table I.6-54/T.180

Parameter name	X_CON_P_USER_DATA
Type of value	string<256>
Legal values	This parameter is a text string, from zero to 255 characters.
Default value	none
Description	This parameter may be used for optional user data.

Table I.6-55/T.180

Parameter name	X_CON_P_RESULT	
Type of value	unsigned long	
Legal values	X_CON_PV_ACCEPT	Accepted;
	X_CON_PV_USER_REJECTED	User rejected;
	X_CON_PV_DOM_PAR_UNACC	Domain parameters unacceptable;
	X_CON_PV_DOM_NOT_HIERARCH	Domain not hierarchical;
	X_CON_PV_LOWER_LAY_DIS	Lower layer initiated disconnect;
	X_CON_PV_UNSPECIFIED	Unspecified failure.
Default value	X_CON_PV_ACCEPT	
Description	This parameter indicates the success or failure of the Conference Query service.	

I.6.7.5.2 Conference Conductor Inquire service

This service may be issued to find out whether the conference is conducted or not. If so, further information is given:

– which node is the conductor;

– if the requesting node has been granted conducted-mode permission.

This service is available to the user only in the connected state of the XAPI.

I.6.7.5.2.1 Service description

The service elements and their corresponding XAPI functions needed for the Conductor Inquire service are described Table I.6-56.

Table I.6-56/T.180 – Service elements and their corresponding XAPI functions for the Conductor Inquire service

Service element	Service element identifier	XAPI function	Description
Conductor Inquire Request	X_COND_INQUIRE_Q	x_sndinfo()	The Information Request is passed to the provider to request the information about the conductor.
Conductor Inquire Confirmation	X_COND_INQUIRE_C	x_rcvinfo()	The Information Confirmation is a positive or negative response to a previous information request.

I.6.7.5.2.2 Service parameters

Table I.6-57 specifies the parameters of the conference Conductor Inquire service.

Table I.6-57/T.180 – Parameter of the Conductor Inquire service

Parameter	Conference Conductor Inquire service	
	Request	Confirmation
X_CON_P_CONDUCTED		M
X_CON_P_COND_NODE		C
X_CON_P_PERMISSION		C

I.6.7.5.2.3 Service parameter description

Tables I.6-58 to I.6-60 specify the parameters of the Conductor Inquire service.

Table I.6-58/T.180

Parameter name	X_CON_P_CONDUCTED
Type of value	unsigned long
Legal values	PV_TRUE PV_FALSE
Default value	PV_FALSE
Description	This parameter indicates whether the conference is currently in conducted or non-conducted mode.

Table I.6-59/T.180

Parameter name	X_CON_P_COND_NODE
Type of value	unsigned long
Legal values	an integer between 1001 and 65 535
Default value	none
Description	This parameter indicates the node ID of the current conference conductor. It is only available if the conference is in the conducted mode.

Table I.6-60/T.180

Parameter name	X_CON_P_PERMISSION
Type of value	unsigned long
Legal values	PV_TRUE PV_FALSE
Default value	none
Description	This parameter indicates if the conference is in the conducted mode, whether or not the local node has been granted conducted mode permission.

I.6.7.5.3 Conference Roster Report service

This service informs the user, in the case when the conference roster has changed for any reason. This service is available to the user only in the connected state of the XAPI.

I.6.7.5.3.1 Service description

The service element and their corresponding XAPI function needed for the Conference Roster Report service are described in Table I.6-61.

Table I.6-61/T.180 – Service element and their corresponding XAPI function for the Conference Roster Report service

Service element	Service element identifier	XAPI function	Description
Conference Roster Indication	X_CON_C_REP_I	x_rcvinfo()	The service informs the user in the case when the Conference Roster changes.

I.6.7.5.3.2 Service parameter

Table I.6-62 specifies the parameter of the Conference Roster Report service.

Table I.6-62/T.180 – Parameter of the Conference Roster Report service

Parameter	Conference Roster Service
	Indication
X_CON_P_CROSTER	M

I.6.7.5.3.3 Service parameter description

Table I.6-63 specifies the parameter of the Conference Roster Report service.

Table I.6-63/T.180

Parameter name	X_CON_P_CROSTER

Type of value	typedef struct Conference {
	unsigned long NodeID;
	unsigned long Node_Type;
	string Node_Name;
	sequence<string<255> > Part_List;
	string Site_Info;
	string Net_Addr;
	string User_Data;
	};
	sequence<Conference>;

Legal values	NodeID: A number in the range 1 001 to 65 535; Node_Type: X_CON_PV_TERMINAL, X_CON_PV_MULTIPORT_TERMINAL, or X_CON_PV_MCU; Node_Name: A text string; Part_List: A list of text strings Participants' names; Site_Info: A text string Additional information about the node; Net_Addr: A text string Network Address; User_Data: A text string Additional user data.
Default value	none
Description	This parameter contains a description of each node joined to the conference. The parameters Node Name, Participants Names, Site Information, Network Address, and User Data in the List of Conference Nodes are conditional.

I.6.7.5.4 Application Roster Report service

This service informs the user in the case when a application roster has changed. This service is available to the user only in the connected state of the XAPI.

I.6.7.5.4.1 Service description

The service element and their corresponding XAPI function needed for the Application Roster Report service are described in Table I.6-64.

Table I.6-64/T.180 – Service element and their corresponding XAPI function for the Application Roster Report service

Service Element	Service element identifier	XAPI function	Description
Application Roster Indication	X_CON_A_REP_I	x_rcvinfo()	The Service informs the user in the case when the Application Roster changes. The Application Roster Indication is generated by the provider to support a session member with session-specific information.

I.6.7.5.4.2 Service parameter

Table I.6-65 specifies the parameters of the Application Roster Report service.

Table I.6-65/T.180 – Parameter of the Application Roster Report service

Parameter	Application Roster Service
	Indication
X_CON_P_AROSTER	M

I.6.7.5.4.3 Service parameter description

Table I.6-66 specifies the parameter of the Application Roster Report service.

Table I.6-66/T.180

Parameter name	X_CON_P_AROSTER

Type of value	typedef struct ApplicationRecord {
	unsigned long NodeID;
	unsigned long EntityID;
	unsigned long Active;
	unsigned long Cond_Cap;
	string UserApplicationID;
	sequence<string> NCollapsCapsList;
	};
	typedef struct ApplicationCapability {
	unsigned long CapabilityID;
	unsigned long CapabilityValue;
	};
	typedef struct Session{
	string ApplicationProtocolKey;
	unsigned long SessionID;
	sequence<ApplicationRecord> ApplicationRecordList;
	sequence<ApplicationCapability> ApplicationCapabilityList;
	};
	sequence<Session>;

Table I.6-66/T.180 *(concluded)*

Legal values	NodeID	A number in the range 1 001 to 65 535	
	EntityID	A 16-bit numeric identifier	
	Active	PV_TRUE / PV_FALSE	Ready to receive data;
	Cond_Cap	PV_TRUE / PV_FALSE	Conducting capable;
	UserApplicationID	A text string	Identifies the conference control user application;
	NCollapsCapsList	A list of text strings	Non-collapsing capabilities;
	CapabilityID	A number	Identifies an application capability;
	CapabilityValue	A number	Dependent from the capability class;
	ApplicationProtocolKey	A text string	Identifies the GCC protocol;
	SessionID	A number in the range 1 to 65 535.	
Default value	none		
Description	The Application Roster includes a list of roster entries for all Protocol Session.		

I.6.7.5.5 Conductor Report service

I.6.7.5.5.1 Service description

The service element and their corresponding XAPI function needed for the Conductor Report service are described in Table I.6-67.

Table I.6-67/T.180 – Service element and their corresponding XAPI function for the Conductor Report service

Service element	Service element identifier	XAPI function	Description
Conductor Report Indication	X_CON_COND_INFO	x_rcvinfo()	The service informs the user in the case when information is received about changes in the conductorship.

I.6.7.5.5.2 Service parameters

Table I.6-68 specifies the parameters of the Conductor Report service.

Table I.6-68/T.180 – Parameters of the Conductor Report service

Parameter	Conductor Service
	Indication
X_CON_P_COND_MODE	M
X_CON_P_REQ_NODE	C

I.6.7.5.5.3 Service parameter description

Tables I.6-69 and I.6-70 specify the parameters of the Conductor Report service.

Table I.6-69/T.180

Parameter name	X_CON_P_COND_MODE
Type of value	unsigned long
Legal values	PV_TRUE PV_FALSE
Default value	PV_FALSE
Description	This parameter indicates that the conference is now in the conducted mode (PV_TRUE) or in the non-conducted mode (PV_FALSE).

Table I.6-70/T.180

Parameter name	X_CON_P_REQ_NODE
Type of value	unsigned long
Legal values	an integer between 1001 and 65 535
Default value	none
Description	This parameter is used to indicate the node identification of the conductor. This parameter is only meaningful when the conference is in the conducted mode (X_CON_P_COND_MODE = PV_TRUE).

I.6.7.6 Tables of error codes

The XAPI error-level error codes are defined in Annex B.

I.6.7.6.1 CC_BADVALUE

If the cause code indicates a parameter error with a bad value, the value of *diagnostic* will contain the erroneous parameter identifier which has been submitted with the XAPI function call that caused the error indication or one of the identifiers X_ACS_CTX_DEF_LIST, X_ACS_CTX_DEF_RES_LIST, X_ACS_CTX_ID_LIST, if the error occurred within a context definition list, context definition result list or context identifier list, respectively.

I.6.7.6.2 CC_MANDMISS

If the cause code indicates a mandatory parameter is missing, the value of *diagnostic* will contain the missing parameter identifier that caused the error indication.

I.6.7.6.3 CC_BADEVENT

If the cause code indicates a bad event, the value of *diagnostic* will contain the bad event identifier which has been submitted with the XAPI function call that caused the error indication.

I.6.7.6.4 CC_UNEXPECT

If the cause code indicates a unexpected event, the value of *diagnostic* will contain the actual state identifier in which the unexpected event caused the error indication.

I.6.7.6.5 CC_NOTSUPPORT

If the cause code indicates an unsupported event, the value of *diagnostic* will contain the identifier of the unsupported event which has been submitted with the XAPI function call that caused the error indication.

I.6.7.6.6 CC_OTHER

If the cause code indicates the CC_OTHER error code, the value of *diagnostic* will contain the identifier which caused the error indication.

I.7 XAPI access to the service provider for T.127 MBFT

This part of Appendix I describes an example how the service provider can be implemented, if an application needs access to the specified service.

I.7.1 Scope

The XAPI, which is an abbreviation for eXtensive Application Programming Interface, is an operating system and language-independent programming interface to general communication services. Detailed information about the XAPI is given in the main part of this Recommendation, which is very important for understanding this appendix.

Which services are made available via the XAPI depends on the installed service providers and not on the XAPI, which only provides the access mechanism.

This part describes the XAPI access to the Multipoint Binary File Transfer (MBFT) Protocol. The MBFT protocol is specified in Recommendation T.127.

Figure I.7-1 shows the structure of the protocol stack that is accessible via the XAPI when selecting the MBFT service provider.

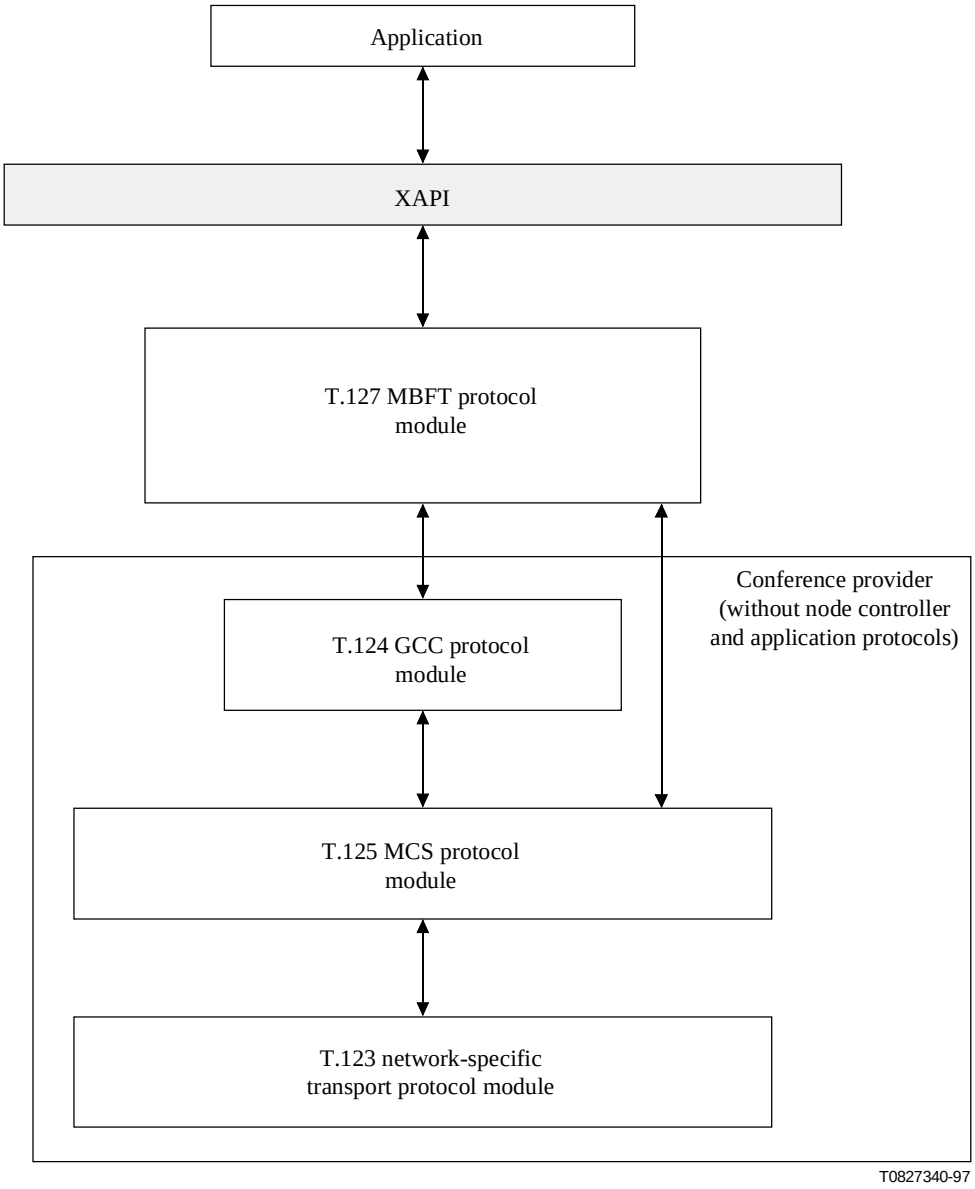


Figure I.7-1/T.180 – Structure of the T.127 MBFT service provider

The MBFT service provider consists of the T.127 MBFT protocol module, the T.124 GCC protocol module, the T.125 MCS protocol module, and the T.123 Network-specific transport protocol stack modules.

The conference provider comprises the T.124 GCC protocol module, the T.125 MCS protocol module, the T.123 Network-specific transport protocol stack modules, the node controller module, and the application protocol modules (e.g. the T.127 MBFT protocol module).

All modules of the conference provider – with the exception of the application protocol modules – are initialized and activated, when the user (at the terminal) is connected to a conference.

Afterwards, applications may be connected to the conference. The location of an application protocol module is shown in Figure I.7-1: each application protocol uses the protocol stack which was initialized and activated in the phase, when a terminal was connected to the conference (see I.6, XAPI access to the T.120 conference control provider).

The reader should be familiar with the T.120 suite of standards (see References).

I.7.2 References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; all users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published.

[T.120] ITU-T Recommendation T.120 (1996), *Data protocols for multimedia conferencing.*

[T.121] ITU-T Recommendation T.121 (1996), *Generic application template.*

[T.122] ITU-T Recommendation T.122 (1998), *Multipoint communication service – Service definition.*

[T.123] ITU-T Recommendation T.123 (1996), *Network specific data protocol stacks for multimedia conferencing.*

[T.124] ITU-T Recommendation T.124 (998), *Generic Conference Control.*

[T.125] ITU-T Recommendation T.125 (1994), *Multipoint communication service protocol specification.*

[T.127] ITU-T Recommendation T.127 (1995), *Multipoint binary file transfer protocol.*

I.7.3 Definitions

This part defines the following terms:

I.7.3.1 application: Synonymously used for User Application.

I.7.3.2 application protocol entity: The instantiation of an Application Protocol in a terminal. Application Protocol Entities are employed by User Applications, but are not themselves User Applications.

I.7.3.3 application protocol module: Synonymously used for Application Protocol Entity.

I.7.3.4 application record: A set of information for a specific Application Protocol Entity at a specific node.

I.7.3.5 session: A set of peer Application Protocol Entities.

A group of file transfer applications communicating with each other are said to be participating in the same file transfer session.

I.7.3.6 MBFT specific transaction: A sequence of one or more actions (i.e. sending and receiving MBFT PDUs, MCS PDUs, and GCC PDUs respectively) which are needed to distribute a file.

I.7.3.7 user application: An entity which makes use of one or more Application Protocol Entities. User Applications have access to an Application Protocol Entity via the XAPI.

I.7.4 Abbreviations

This part uses the following abbreviations:

APE Application Protocol Entity

BFT Binary File Transfer

GCC Generic Conference Control

MCS Multipoint Communication Service

MBFT Multipoint Binary File Transfer

PDU Protocol Data Unit

SP Service Primitive

XAPI eXtensive Application Programming Interface

I.7.5 Conventions

Each service is described via three kinds of tables. The kind of description is derived from the kind of service and protocol description within OSI Standards.

In the first kind of table the service and its service elements are described. It contains a row for each service element of the service (where service element means either request, indication, response, or confirmation) with its corresponding XAPI function and a short description.

In the second kind of table the use of the service primitive parameters within the service elements are described. It contains a row for each service primitive parameter of the service. The service primitive parameter name is stored within the first column; the succeeding columns contain the use of the parameter in the service elements:

Blank The service parameter is absent.

C Presence of the service parameter is conditional. Firstly, there may be a condition in the service provider to provide a parameter in an indication or confirmation; secondly, there may be interdependencies between parameters of the same or the preceding service primitive.

M Presence of the service parameter is mandatory.

U Presence of the service parameter is a user option. If the user does not specify a value for such a parameter and there is no default value for that parameter, nothing is passed to the service provider. If the user does not specify a value for such a parameter and there is a default value, the default value is passed to the service provider.

(=) The value of the service parameter is identical to the value of the corresponding service parameter in the preceding service element. In the special case of a parameter, whose presence in the preceding service is a user option, for which a default value is defined, and the parameter was not specified in the preceding service element, the symbol (=) indicates that the parameter value is identical to the default value.

The third kind of table is used to describe the service primitive parameter containing the type of value and its scope, possible default values and a detailed description. For each service primitive parameter, one description table is given.

A **naming convention** is used throughout the XAPI in order to distinguish between the common functions, identifiers and values, and service-specific identifiers and values. All service-specific settings of the T.127 MBFT service provider access are defined within the present part of this appendix and start with **X_MBF_** or **x_mbf_**.

Extended naming conventions:

– SP_ Service Primitive;

– P_ Parameter;

– PV_ Parameter Value;

– O_ Option;

– OV_ Option Value.

I.7.6 Introduction to the MBFT service provider access

The T.127 MBFT Protocol [T.127] supports the interchange of binary files within an interactive conferencing or group working environment where the [T.120] series of Recommendations is in use. It provides mechanisms for:

- simultaneous distribution of multiple files;
- broadcasting of files to all participants within a conference;
- selective distribution of files to a subset of participants;
- conductor control of file distribution;
- retrieval of files from remote sites;
- partial retransmission of files following an interruption;
- remote directory access.

The following services of the above specified groups are supported:

- simultaneous distribution of multiple files;
- broadcasting of files to all participants within a conference;
- selective distribution of files to a subset of participants;
- conductor control of file distribution.

A file transfer user application initiates a file transfer session via its MBFT protocol entity, specifying the application capabilities and session mode. Once a session has been established, all MBFT specific transactions are performed by the MBFT protocol entity on behalf of the user application.

I.7.7 Description of the access to the MBFT service provider

I.7.7.1 Service initialization

I.7.7.1.1 Creation of a T.127 MBFT service access point with x_open()

A communication endpoint accessing the T.127 MBFT service provider is created when calling the *x_open()* function with an appropriate service provider identification string. The available identifiers depend on the actual system configuration. In the standard configuration, "**X_T.127_MBFT**" identifies the T.127 MBFT protocol module.

I.7.7.1.2 Activation of a T.127 MBFT service access point with *x_bind()*

x_bind() is to be called to activate the T.127 MBFT service endpoint. The function has the task to bind the application's own address to the service endpoint.

In the standard configuration, if "X_T.127_MBFT" was selected in the *x_open()* function, a suitable conference system, for example "**X_T.120_CONF_ISDN**" has to be specified as argument of the *x_bind()* function in order to complete the service provider's protocol stack.

Figure I.7-1 illustrates the protocol modules contained in X_T.120_CONF_PRO.

I.7.7.1.3 Addresses

Prior to connecting an application to a conference, the user has to become a participant within a conference. The address information, which is needed to identify a node within a conference is handled in this early phase (i.e. in the phase of creating or joining a conference). Therefore, binding the application's own address to the service endpoint will rely on this information.

I.7.7.1.3.1 Specification of the application's own address

No specific address is in use when activating the service endpoint.

No other parameters are needed.

It is strongly recommended to trust in the XAPI configuration and not to specify the own address or other parameters in the *x_bind()* call.

I.7.7.1.3.2 The address of the communication partner

This parameter has been provided by the conference connection establishment phase [T.124].

I.7.7.1.4 Configuration of the service provider

Protocol options are used to control the general behaviour of the service provider. There is a default value defined for each option. These preconfigured values are sufficient for the majority of communication relations.

XAPI enables applications to express their wishes about the settings of protocol options by using the XAPI function *x_optmgmt()*.

Tables I.7-1 to I.7-8 define the protocol options.

Table I.7-1/T.180

Option name	X_MBF_O_M_TYPE
Type of value	unsigned long
Legal values	X_MBF_OV_RCV_ONLY File Receive Only MBFT Protocol module X_MBF_OV_SND_ONLY File Send Only MBFT Protocol module X_MBF_OV_SND_RCV File Receive and Send MBFT Protocol module
Default value	none
Attributes	
Description	Indicates the type of the T.127 MBFT protocol module.

Table I.7-2/T.180

Option name	X_MBF_O_MAX_FILE_SIZE
Type of value	unsigned long
Legal values	X_MBF_OV_FS_UNLIM Unlimited; X_MBF_OV_FS_MAX_VAL Maximum file data payload in octets.
Default value	X_MBF_OV_FS_UNLIM
Description	Indicates the maximum file size. Each T.127 MBFT protocol module must specify the maximum file data payload in octets that it is capable of receiving.

Table I.7-3/T.180

Option name	X_MBF_O_MAX_DAT_PAYL
Type of value	unsigned long
Legal values	X_MBF_OV_DP_DEF Default value (8192 octets); X_MBF_OV_DP_MAX_VAL Maximum number (less or equal 65 536 octets).
Default value	X_MBF_OV_DP_DEF
Description	Indicates the maximum data payload. This is the maximum number of octets allowed in the data field of T.127 MBFT start and data PDUs.

Table I.7-4/T.180

Option name	X_MBF_O_COMPR
Type of value	unsigned long
Legal values	PV_TRUE PV_FALSE
Default value	PV_FALSE
Attributes	
Description	Compresses files before being transferred. When actually transferring files, it is possible to select or deselect this option.

Table I.7-5/T.180

Option name	X_MBF_O_NB_OF_CWOR
Type of value	unsigned long
Legal values	X_MBF_OV_CW_DEF Default value (512 codewords); X_MBF_OV_CW_NUM Total number of codewords (less or equal 65 536).
Default value	X_MBF_OV_CW_DEF
Description	Specifies the total number of codewords to be used by the V.42 *bis* compression algorithm. This is an upper bound on V.42 *bis* parameter P1.

Table I.7-6/T.180

Option name	X_MBF_O_MAX_STR_LNGH
Type of value	unsigned long
Legal values	X_MBF_OV_STR_DEF Default value (6); X_MBF_OV_STR_VAL Maximum string length (less or equal 250).
Default value	X_MBF_OV_STR_DEF
Description	Specifies the maximum string length input to the V.42 *bis* encoder. This is an upper bound on V.42 *bis* parameter P2.

Table I.7-7/T.180

Option name	X_MBF_O_AROSTER
Type of value	unsigned long
Legal values	X_MBF_OV_AR_NO Application Roster not desired; X_MBF_OV_AR_OWN Receive only Application Roster Entries of the session(s) in which the user application is enrolled; X_MBF_OV_AR_ALL Receive all Application Roster Entries of all MBFT sessions.
Default value	X_MBF_OV_AR_ALL
Attributes	
Description	Regulates the contents of an Application Roster Indication.

Table I.7-8/T.180

Option name	X_MBF_O_CROSTER
Type of value	unsigned long
Legal values	PV_TRUE PV_FALSE
Default value	PV_TRUE
Attributes	
Description	Receiving of Conference Roster Indications.

I.7.7.2 Connection Establishment service

I.7.7.2.1 Service description

Prior to connecting a file transfer application to a conference, the user has to become a conference participant.

All application-related activities within a conference (e.g. transferring or receiving files) are organized as sessions.

During connection establishment phase, a file transfer user application establishes a connection (i.e. creating a new session or joining an already existing session) to other file transfer user applications. The connection is identified by the conference name and the session name respectively. The file transfer user application must already have prepared a service endpoint before the connection establishment phase can start.

Only active establishment of a connection is provided (i.e. the XAPI function *x_conreq()* applies). The service provider generates a positive or negative connect confirmation primitive as response to the connect request. The positive confirmation signals that the file transfer user application is now participating in the requested session.

The elements of the Connection Establishment service and the corresponding XAPI functions are defined in Table I.7-9.

Table I.7-9/T.180 – Service elements and their corresponding XAPI functions for Connection Establishment

Service element	XAPI function	Description
Connect Request	x_conreq()	The Connect Request is passed to the provider to request the establishment of an connection for file transfer purposes.
Connect Confirmation	x_conconf()	The Connect Confirmation is generated by the provider as positive or negative response to a previous connection establishment request. A positive confirmation indicates that the provider accepted the call.

I.7.7.2.2 Service parameters

Table I.7-10 specifies the parameters of the Connection Establishment service.

There are two groups of parameters: the parameters which specify general aspects of a MBFT session and the MBFT service provider capability parameters.

The parameters specifying general aspects of a MBFT session are:

– the user application ID;

– the conference name;

– the session name;

– whether or not the provider should support the conductor role;

– the type of activity: transmit files only, receive files only, transmit and receive files; files which are neither transmitted nor received (application inactive);

– the actual list of participants in the session.

The specification of provider capabilities is optional. If a capability parameter is not specified in a service element, the default value will be used by the service provider. The default value itself is defined by the value of a protocol option. For each capability parameter there is a corresponding protocol option which defines the default value for this parameter. For the protocol option, a constant default value is defined in the XAPI configuration.

Table I.7-10/T.180 – Parameters of the Connection Establishment service

Parameter	Connect Service	
	Request	Confirmation
X_MBF_P_CNAME	M	M(=)
X_MBF_P_CNAME_M	C	C(=)
X_MBF_P_SESS_TYPE	M	M
X_MBF_P_SESS_ID	U	M
X_MBF_P_LIST_OF_MEMB	U	
X_MBF_P_USER_APP_ID	U	
X_MBF_P_COND_OP_CAP	U	
X_MBF_P_M_MODE	U	M
X_MBF_P_MAX_FILE_SIZE	U	C

Table I.7-10/T.180 – Parameters of the Connection Establishment service *(concluded)*

Parameter	Connect Service	
	Request	**Confirmation**
X_MBF_P_MAX_DAT_PAYL	U	C
X_MBF_P_COMPR	U	C
X_MBF_P_NB_OF_CWOR	U	C
X_MBF_P_MAX_STR_LNGH	U	C
X_MBF_P_NON_STD_CAP	C	C
X_MBF_P_NODE_ID		C
X_MBF_P_ENTITY_ID		C
X_MBF_P_RESULT		M

I.7.7.2.3 Service parameter descriptions

Tables I.7-11 to I.7-27 describe the parameters for the Connection Establishment service.

Table I.7-11/T.180

Parameter name	X_MBF_P_CNAME

Type of value	struct X_MBF_P_CNAME {
	string<256> numeric_string;
	string<256> text_string;
	};

Legal values	a numerical string along with an optional text string, from zero to 255 characters each
Default value	none
Description	Name by which the conference is identified.

Table I.7-12/T.180

Parameter name	X_MBF_P_CNAME_M
Type of value	string<255>
Legal values	a numerical string up to 255 digits in length
Default value	none
Description	Name Modifier by which the conference is identified if a conference with identical conference name already exists.

Table I.7-13/T.180

Parameter name	X_MBF_P_SESS_TYPE
Type of value	unsigned long
Legal values	X_MBF_PV_SESS_REG Registration session; X_MBF_PV_SESS_STD Standard Base session; X_MBF_PV_SESS_NSTD Non-Standard base session; X_MBF_PV_SESS_PUB Public session; X_MBF_PV_SESS_PRI Private session.
Default value	X_MBF_PV_SESS_STD
Description	Parameter by which the session type is identified.

Table I.7-14/T.180

Parameter name	X_MBF_P_SESS_ID
Type of value	unsigned long
Legal values	a number in the range 1 to 65 535
Default value	none
Description	Session Identifier.

Table I.7-15/T.180

Parameter name	X_MBF_P_LIST_OF_MEMB
Type of value	sequence<unsigned long>
Legal values	a list of numbers in the range 1001 to 65 535
Default value	none
Description	This parameter is used to invite participants (specified by their NodeIDs) to a private session.

Table I.7-16/T.180

Parameter name	X_MBF_P_USER_APP_ID
Type of value	string
Legal values	a text string
Default value	none
Description	User Application Identifier.

Table I.7-17/T.180

Parameter name	X_MBF_P_COND_OP_CAP
Type of value	unsigned long
Legal values	PV_TRUE PV_FALSE
Default value	PV_FALSE
Description	The Conducting Operation Capable parameter indicates whether the MBFT Protocol module is capable of operating as a session conductor. Each MBFT Session has its own conductor, namely the MBFT Protocol module at the conducting node participating in that Session. If there is more than one MBFT Protocol module at the conducting node in a given MBFT Session, then it is a local matter for GCC at the conducting node to decide which MBFT Protocol module adopts the role of session conductor.

Table I.7-18/T.180

Parameter name	X_MBF_P_M_MODE
Type of value	unsigned long
Legal values	X_MBF_PV_MM_REC Receive files only; X_MBF_PV_MM_SND Transmit files only; X_MBF_PV_MM_SND_REC Transmit and receive files; X_MBF_PV_MM_N_SND_REC Neither transmit nor receive files.
Default value	none
Description	Indicates the T.127 MBFT protocol module mode. It is dependent from the protocol option X_MBF_O_M_TYPE.

Table I.7-19/T.180

Parameter name	X_MBF_P_MAX_FILE_SIZE
Type of value	unsigned long
Legal values	X_MBF_PV_FS_UNLIM Unlimited; X_MBF_PV_FS_MAX_VAL Maximum file data payload in octets.
Default value	X_MBF_PV_FS_UNLIM
Description	Indicates the maximum file size. Each T.127 MBFT protocol module must specify the maximum file data payload in octets that it is capable of receiving.

Table I.7-20/T.180

Parameter name	X_MBF_P_MAX_DAT_PAYL
Type of value	unsigned long
Legal values	X_MBF_PV_DP_DEF Default value (8192 octets); X_MBF_PV_DP_MAX_VAL Maximum number (less or equal 65 536 octets).
Default value	X_MBF_PV_DP_DEF
Description	Indicates the maximum data payload. This is the maximum number of octets allowed in the data field of T.127 MBFT start and data PDUs.

Table I.7-21/T.180

Parameter name	X_MBF_P_COMPR
Type of value	unsigned long
Legal values	PV_TRUE PV_FALSE
Default value	PV_FALSE
Description	This parameter is used to negotiate use of V.42*bis* compression for file data. T.127 MBFT protocol modules should assert this capability if they are able to receive V.42*bis* compressed data.

Table I.7-22/T.180

Parameter name	X_MBF_P_NB_OF_CWOR
Type of value	unsigned long
Legal values	X_MBF_PV_CW_DEF Default value (512 codewords); X_MBF_PV_CW_NUM Total number of codewords (less or equal 65 536).
Default value	X_MBF_PV_CW_DEF
Description	Specifies the total number of codewords to be used by the V.42 *bis* compression algorithm. This is an upper bound on V.42 *bis* parameter P1.

Table I.7-23/T.180

Parameter name	X_MBF_P_MAX_STR_LNGH
Type of value	unsigned long
Legal values	X_MBF_PV_STR_DEF Default value (6); X_MBF_PV_STR_VAL Maximum string length (less or equal 250).
Default value	X_MBF_PV_STR_DEF
Description	Specifies the maximum string length input to the V.42 *bis* encoder. This is an upper bound on V.42 *bis* parameter P2.

Table I.7-24/T.180

Parameter name	X_MBF_P_NON_STD_CAP
Type of value	unsigned long
Legal values	X_MBF_PV_NSC_UNSP Unspecified.
Default value	X_MBF_PV_NSC_UNSP
Description	This parameter is used to negotiate non-standard functions, including non-standard compression techniques.

Table I.7-25/T.180

Parameter name	X_MBF_P_NODE_ID
Type of value	unsigned long
Legal values	a number in the range 1001 to 65 535
Default value	none
Description	Indicates the own Node ID.

Table I.7-26/T.180

Parameter name	X_MBF_P_ENTITY_ID
Type of value	unsigned long
Legal values	a 16-bit numeric identifier
Default value	none
Description	Indicates the Entity ID of the corresponding MBFT Protocol module. The combination of the Entity ID and the Node ID uniquely identifies the MBFT Protocol module in a conference.

Table I.7-27/T.180

Parameter name	X_MBF_P_RESULT	
Type of value	unsigned long	
Legal values	X_MBF_PV_R_ACCEPT	Connection accepted;
	X_MBF_PV_R_INV_CONF	Rejection; invalid conference;
	X_MBF_PV_R_T_CONG	Rejection; temporary congestion;
	X_MBF_PV_R_TM_RES	Rejection; too many resources;
	X_MBF_PV_R_TM_USER	Rejection; too many users;
	X_MBF_PV_R_NS_SESS_TYPE	Rejection; session type not supported;
	X_MBF_PV_R_WRONG_SESS_TYPE	Rejection; wrong session type;
	X_MBF_PV_R_NO_REASON	Rejection, reason not specified.
Default value	X_MBF_PV_R_ACCEPT	
Description	Indicates the success or failure of the connection establishment request.	

I.7.7.3 Services in the connected state

The purpose of the services provided in the connected state is to distribute files among the session members. Additional services comprise getting information about the session (e.g. getting the actual list of the session members) and changing (some of) the session parameters negotiated in the connection establishment phase.

I.7.7.3.1 Services which apply to file distribution

Once the session has been established, all MBFT specific transactions are performed by the T.127 MBFT protocol module on behalf of the user application.

I.7.7.3.1.1 File Offer service

I.7.7.3.1.1.1 Service description

The File Offer service is used to offer a file to all intended recipients. These recipients are either all participants of the conference or a subset of them. Establishing a private sub-session within the existing session or using an already existing sub-session is part of the File Offer service and may be used to restrict the number of recipients.

The service elements and their corresponding XAPI functions needed are described in Table I.7-28.

Table I.7-28/T.180 – Service elements and their corresponding XAPI functions for File Offer

Service element	Service element identifier	XAPI function	Description
File Offer Request	X_MBF_SP_FO_Q	x_sndsp()	The File Offer Request is passed to the provider to send a request to all intended recipients.
File Offer Indication	X_MBF_SP_FO_I	x_rcvsp()	The File Offer Indication is generated by the provider to indicate the offering of a file.
File Offer Response	X_MBF_SP_FO_P	x_sndsp()	The File Offer Response is passed to the provider to indicate acceptance or rejection.
File Offer Confirmation	X_MBF_SP_FO_C	x_rcvsp()	The File Offer Confirmation is generated by the provider as acknowledgement of the previous request.

Two modes of offering a file are supported by the File Offer service. The recipients may either accept (reject) the offer or they are not asked to acknowledge the offer. In the latter case, only the request and the indication primitives apply.

I.7.7.3.1.1.2 Service parameters

Table I.7-29 specifies the parameters of the File Offer service.

Table I.7-29/T.180 – Parameters of the File Offer service

Parameter	File Offer service			
	Request	Indication	Response	Confirmation
X_MBF_P_NODE_ID		M		
X_MBF_P_ENTITY_ID		M		
X_MBF_P_FHEADER_T434	M	M (=)		
X_MBF_P_SELECTIVE	M			
X_MBF_P_FTS_MEMB	C			
X_MBF_P_FILE_REF	M	M (=)	C (=)	C (=)
X_MBF_P_COMPR	U	C (=)		

Table I.7-29/T.180 – Parameters of the File Offer service *(concluded)*

Parameter	File Offer service			
	Request	**Indication**	**Response**	**Confirmation**
X_MBF_P_NB_OF_CWOR	U	C (=)		
X_MBF_P_MAX_STR_LNGH	U	C (=)		
X_MBF_P_ACK_Q	M	M (=)		
X_MBF_P_REJ_FTS_MEMB				C
X_MBF_P_ACK_P			C	
X_MBF_P_FO_REASON			C	

I.7.7.3.1.1.3 Service parameter descriptions

Tables I.7-30 to I.7-39 list the parameters of the File Offer service. The parameters X_MBF_P_COMPR, X_MBF_P_NB_OF_CWOR, and X_MBF_P_MAX_STR_LNGH are described in the connection establishment phase. Therefore, their description is omitted in this subclause.

Table I.7-30/T.180

Parameter name	X_MBF_P_NODE_ID
Type of value	unsigned long
Legal values	a number in the range 1001 to 65 535
Default value	none
Description	Indicates the Node ID of the file offering user application.

Table I.7-31/T.180

Parameter name	X_MBF_P_ENTITY_ID
Type of value	unsigned long
Legal values	a 16-bit numeric identifier
Default value	none
Description	Indicates the MBFT Protocol Entity ID of the file offering user application. The combination of the Entity ID and the Node ID uniquely identifies the MBFT Protocol Entity in a conference and so the corresponding MBFT user application.

Table I.7-32/T.180

Parameter name	X_MBF_P_FHEADER_T434
Type of value	struct
Legal values	a list of optional BFT file attributes defined in Recommendation T.434
Default value	T.434 protocol version number
Description	Uses the T.434 file header structure and should include sufficient information to allow intending recipients to determine whether the file is required.

Table I.7-33/T.180

Parameter name	X_MBF_P_SELECTIVE
Type of value	unsigned long
Legal values	PV_TRUE Distribution to a subgroup; PV_FALSE Broadcasting.
Default value	PV_FALSE
Description	Indicates the distribution of the file to all or to a subgroup of file transfer session members.

Table I.7-34/T.180

Parameter name	X_MBF_P_FTS_MEMB
Type of value	sequence<sequence<unsigned long, 2> >
Legal values	a list of tuples, each consisting of two numbers: NodeID and EntityID
Default value	none
Description	Contains a list of user applications for establishing a private sub-session.

Table I.7-35/T.180

Parameter name	X_MBF_P_FILE_REF
Type of value	string
Legal values	a numerical number
Default value	none
Description	Indicates the file reference number (i.e. dataflow identification). This number should start by 1 and should be incremented for every file transfer which is transmitted in the same connection. Leading zeros will be ignored.

Table I.7-36/T.180

Parameter name	X_MBF_P_ACK_Q
Type of value	unsigned long
Legal values	PV_TRUE PV_FALSE
Default value	PV_FALSE
Description	Indicates that receiving user applications must signal whether or not they wish to accept the file.

Table I.7-37/T.180

Parameter name	X_MBF_P_REJ_FTS_MEMB
Type of value	sequence<sequence<unsigned long, 3> >
Legal values	A list of tripels, each consisting of three numbers: NodeID, EntityID, and Reason for Rejection.
Default value	none
Description	Indicates the members which are unable or do not wish to receive the offered file. The possible reasons for rejection are: X_MBF_PV_FOR_NORESP No response; X_MBF_PV_FOR_UNSPEC Unspecified; X_MBF_PV_FOR_FEXIST File exists; X_MBF_PV_FOR_FNOTREQUIRE File not required; X_MBF_PV_FOR_INSUFFRES Insufficient resources; X_MBF_PV_FOR_UNSUPPCOMPR Compression unsupported; algorithm identified in File Offer not supported.

Table I.7-38/T.180

Parameter name	X_MBF_P_ACK_P
Type of value	unsigned long
Legal values	PV_TRUE PV_FALSE
Default value	PV_FALSE
Description	Indicates the intention to receive the offered file.

Table I.7-39/T.180

Parameter name	X_MBF_P_FO_REASON	
Type of value	unsigned long	
Legal values	X_MBF_PV_FOR_UNSPEC X_MBF_PV_FOR_FEXIST X_MBF_PV_FOR_FNOTREQUIRE X_MBF_PV_FOR_INSUFFRES X_MBF_PV_FOR_UNSUPPCOMPR	Unspecified; File exists; File not required; Insufficient resources; Compression unsupported; algorithm identified in File Offer not supported.
Default value	none	
Description	Informs the transmitter why the intended recipient is unable or does not wish to receive the file offered.	

I.7.7.3.1.2 Start of File service

I.7.7.3.1.2.1 Service description

The Start of File service indicates the start of the document to the receiver. It also indicates the start of the first data part.

The service elements and their corresponding XAPI functions needed are described in Table I.7-40.

Table I.7-40/T.180 – Service elements and their corresponding XAPI functions for Start of File

Service element	Service element identifier	XAPI function	Description
Start of File Request	X_MBF_SP_SOF_Q	x_sndsp()	The Start of File Request is passed to the provider to send a request to all intended recipients.
Start of File Indication	X_MBF_SP_SOF_I	x_rcvsp()	The Start of File Indication is generated by the provider to indicate the start of a file.

I.7.7.3.1.2.2 Service parameters

Table I.7-41 specifies the parameters of the Start of File service.

Table I.7-41/T.180 – Parameters of the Start of File service

Parameter	Start of File service	
	Request	Indication
X_MBF_P_NODE_ID		M
X_MBF_P_ENTITY_ID		M
X_MBF_P_FHEADER_T434	M	M (=)
X_MBF_P_FILE_REF	M (=)	M (=)
X_MBF_P_EOF	M	M (=)

Table I.7-41/T.180 – Parameters of the Start of File service *(concluded)*

Parameter	Start of File service	
	Request	**Indication**
X_MBF_P_CRC	M	M (=)
X_MBF_P_COMPR	U	C (=)
X_MBF_P_NB_OF_CWOR	U	C (=)
X_MBF_P_MAX_STR_LNGH	U	C (=)
X_MBF_P_DATA_OFFSET	M	M (=)

I.7.7.3.1.2.3 Service parameter descriptions

Tables I.7-42 to I.7-48 list the parameters of the Start of File service. The parameters X_MBF_P_COMPR, X_MBF_P_NB_OF_CWOR, and X_MBF_P_MAX_STR_LNGH are described in the connection establishment phase. Therefore, their description is omitted in this subclause.

Table I.7-42/T.180

Parameter name	X_MBF_P_NODE_ID
Type of value	unsigned long
Legal values	a number in the range 1001 to 65 535
Default value	none
Description	Indicates the Node ID of the file offering user application.

Table I.7-43/T.180

Parameter name	X_MBF_P_ENTITY_ID
Type of value	unsigned long
Legal values	a 16-bit numeric identifier
Default value	none
Description	Indicates the MBFT Protocol Entity ID of the file offering user application. The combination of the Entity ID and the Node ID uniquely identifies the MBFT Protocol module in a conference.

Table I.7-44/T.180

Parameter name	X_MBF_P_FHEADER_T434
Type of value	struct
Legal values	a list of optional BFT file attributes defined in Recommendation T.434
Default value	T.434 protocol version number
Description	Uses the T.434 file header structure and should include all fields defined in the source file.

Table I.7-45/T.180

Parameter name	X_MBF_P_FILE_REF
Type of value	string
Legal values	a numerical number
Default value	none
Description	Indicates the file reference number which was given in the File Offer Request.

Table I.7-46/T.180

Parameter name	X_MBF_P_EOF
Type of value	unsigned long
Legal values	PV_TRUE PV_FALSE
Default value	PV_FALSE
Description	Indicates the end of the file.

Table I.7-47/T.180

Parameter name	X_MBF_P_CRC
Type of value	unsigned long
Legal values	X_MBF_PV_CRC_NO No CRC; X_MBF_PV_CRC_SINGLE Single CRC sent at end of File Transfer; X_MBF_PV_CRC_CUMU Cumulative CRC.
Default value	X_MBF_PV_CRC_NO
Description	Indicates the use of a cyclic redundancy check.

Table I.7-48/T.180

Parameter name	X_MBF_P_DATA_OFFSET
Type of value	unsigned long
Legal values	
Default value	none
Description	Specifies the starting offset in octets from the beginning of the file data (so zero denotes the origin).

I.7.7.3.1.3 Data Transfer service

I.7.7.3.1.3.1 Service Description

The Data Transfer service is used to transfer files between session members.

The service elements and their corresponding XAPI functions needed are described in Table I.7-49.

Table I.7-49/T.180 – Service elements and their corresponding XAPI functions for Data Transfer

Service element	XAPI function	Description
Data Transfer Request	x_snddata()	The Data Transfer Request is passed to the provider to transmit data.
Data Transfer Indication	x_rcvdata()	The Data Transfer Indication is generated by the provider to indicate the received data.

I.7.7.3.1.3.2 Service parameters

Table I.7-50 specifies the parameters of the Data Transfer service.

Table I.7-50/T.180 – Parameters of the Data Transfer service

Parameter	Data Transfer service	
	Request	Indication
X_MBF_P_FILE_REF	M (=)	M (=)
X_MBF_P_EOF	M	M (=)
X_MBF_P_ABORT	M	M (=)
X_MBF_P_CRC_CHECK	C	C

I.7.7.3.1.3.3 Service parameter descriptions

Tables I.7-51 to I.7-54 list the parameters of the Data Transfer service.

Table I.7-51/T.180

Parameter name	X_MBF_P_FILE_REF
Type of value	string
Legal values	a numerical number
Default value	none
Description	Indicates the file reference number which was given in the File Offer Request.

Table I.7-52/T.180

Parameter name	X_MBF_P_EOF
Type of value	unsigned long
Legal values	PV_TRUE PV_FALSE
Default value	PV_FALSE
Description	Indicates the end of the file.

Table I.7-53/T.180

Parameter name	X_MBF_P_ABORT
Type of value	unsigned long
Legal values	PV_TRUE PV_FALSE
Default value	PV_FALSE
Description	This parameter is set to TRUE if the file transfer is being aborted.

Table I.7-54/T.180

Parameter name	X_MBF_P_CRC_CHECK
Type of value	unsigned long
Legal values	a 32-bit number
Default value	none
Description	The cyclic redundancy check.

I.7.7.3.1.4 Please Privilege service

I.7.7.3.1.4.1 Service description

The Please Privilege service is used in the conducted mode by user applications at non-conducting nodes to request permission to act from the conductor. The following privileges are available:

– source file transfers;
– request file transfers;
– selective file transfers;
– source file transfers with higher priority;
– issue a file transfer abort;
– use non-standard extensions.

The service elements and their corresponding XAPI functions needed are described in Table I.7-55.

Table I.7-55/T.180 – Service elements and their corresponding XAPI functions for Please Privilege

Service element	Service element identifier	XAPI function	Description
Please Privilege Request	X_MBF_SP_PP_Q	x_sndsp()	The Please Privilege Request is passed to the provider to request privileges from the conductor.
Please Privilege Indication	X_MBF_SP_PP_I	x_rcvsp()	The Please Privilege Indication is generated by the provider to indicate a privilege request.

I.7.7.3.1.4.2 Service parameters

Table I.7-56 specifies the parameters of the Please Privilege service.

Table I.7-56/T.180 – Parameters of the Please Privilege service

Parameter	Please Privilege service	
	Request	Indication
X_MBF_P_PRI_FT	U	C (=)
X_MBF_P_PRI_FQ	U	C (=)
X_MBF_P_PRI_SEL	U	C (=)
X_MBF_P_PRI_ORITY	U	C (=)
X_MBF_P_PRI_ABORT	U	C (=)
X_MBF_P_PRI_NSTD	U	C (=)

I.7.7.3.1.4.3 Service parameter descriptions

Tables I.7-57 to I.7-62 list the parameters of the Please Privilege service.

Table I.7-57/T.180

Parameter name	X_MBF_P_PRI_FT
Type of value	unsigned long
Legal values	PV_TRUE PV_FALSE
Default value	PV_FALSE
Description	Indicates whether the requesting user application wants permission to source file transfers.

Table I.7-58/T.180

Parameter name	X_MBF_P_PRI_FQ
Type of value	unsigned long
Legal values	PV_TRUE PV_FALSE
Default value	PV_FALSE
Description	Indicates whether the requesting user application wants permission to request files or retransmissions.

Table I.7-59/T.180

Parameter name	X_MBF_P_PRI_SEL
Type of value	unsigned long
Legal values	PV_TRUE PV_FALSE
Default value	PV_FALSE
Description	Indicates whether the requesting user application wants permission to selective file transfer.

Table I.7-60/T.180

Parameter name	X_MBF_P_PRI_ORITY
Type of value	unsigned long
Legal values	PV_TRUE PV_FALSE
Default value	PV_FALSE
Description	Indicates whether the requesting user application wants permission to send with a higher priority.

Table I.7-61/T.180

Parameter name	X_MBF_P_PRI_ABORT
Type of value	unsigned long
Legal values	PV_TRUE PV_FALSE
Default value	PV_FALSE
Description	Indicates whether the requesting user application wants permission to issue a file transfer abort.

Table I.7-62/T.180

Parameter name	X_MBF_P_PRI_NSTD
Type of value	unsigned long
Legal values	PV_TRUE PV_FALSE
Default value	PV_FALSE
Description	Indicates whether the requesting user application wants permission to use any negotiated non-standard extensions to the MBFT protocol.

I.7.7.3.1.5 Assign Privilege service

I.7.7.3.1.5.1 Service description

The Assign Privilege service is used in the conducted mode by the conductor to grant or revoke privileges for user applications at non-conducting nodes. The following privileges are available:

– source file transfers;
– request file transfers;
– selective file transfers;
– source file transfers with higher priority;
– issue a file transfer abort;
– use non-standard extensions.

The service elements and their corresponding XAPI functions needed are described in Table I.7-63.

Table I.7-63/T.180 – Service elements and their corresponding XAPI functions for Assign Privilege

Service element	Service element identifier	XAPI function	Description
Assign Privilege Request	X_MBF_SP_AP_Q	x_sndsp()	The Assign Privilege Request is passed to the provider to assign privileges to user applications at non-conducting nodes.
Assign Privilege Indication	X_MBF_SP_AP_I	x_rcvsp()	The Assign Privilege Indication is generated by the provider to indicate assigned privileges from the conductor.

I.7.7.3.1.5.2 Service parameters

Table I.7-64 specifies the parameters of the Assign Privilege service.

Table I.7-64/T.180 – Parameter of the Assign Privilege service

Parameter	Assign Privilege service	
	Request	Indication
X_MBF_P_PRI_LST	M	M (=)

I.7.7.3.1.5.3 Service parameter descriptions

Table I.7-65 describes the parameter of the Assign Privilege service.

Table I.7-65/T.180

Parameter name	X_MBF_P_PRI_LST
Type of value	sequence<sequence<unsigned long, 8> >
Legal values	A list of 8-tuples, each consisting of eight numbers: NodeID, EntityID, and the six privileges parameters.
Default value	none
Description	Contains a list of one or more user applications and the privileges assigned to them: The File Transfer parameter indicates whether the user application has been granted permission to source file transfers. The File Request parameter indicates whether the user application has been granted permission to request files or retransmissions. The Selective File Transfer parameter indicates whether the user application has been granted permission to selective file transfer. The Priority parameter indicates whether the user application has been granted permission to send with a higher priority. The Abort parameter indicates whether the user application has been granted permission to issue a file transfer abort. The Non-Standard parameter indicates whether the user application has been granted permission to issue any negotiated non-standard extensions to the MBFT protocol.

I.7.7.3.2 Information services

The services described in this subclause allow session members to get information about the current session. The member may either request information or he will get information from the provider in cases where specific parameters characterizing the current session have changed their values.

I.7.7.3.2.1 Application Roster service

I.7.7.3.2.1.1 Service description

The Application Roster service is used to support members with session-specific information.

The service elements and their corresponding XAPI functions needed are described in Table I.7-66.

Table I.7-66/T.180 – Service element and their corresponding XAPI function for Application Roster

Service element	Service element identifier	XAPI function	Description
Application Roster Indication	X_MBF_SP_AR_I	x_rcvinfo()	The Application Roster Indication is generated by the provider to support a session member with session-specific information.

I.7.7.3.2.1.2 Service parameters

Table I.7-67 specifies the parameters of the Application Roster service.

Table I.7-67/T.180 – Parameter of the Application Roster service

Parameter	Application Roster service
	Indication
X_MBF_P_AROSTER	M

I.7.7.3.2.1.3 Service parameter descriptions

Table I.7-68/T.180 defines the parameter for the Application Roster service.

Table I.7-68/T.180

Parameter name	X_MBF_P_AROSTER

Type of value	typedef struct ApplicationRecord {
	unsigned long NodeID;
	unsigned long EntityID;
	unsigned long Active;
	unsigned long Cond_Cap;
	string UserApplicationID;
	sequence<string> NCollapsCapsList;
	};
	typedef struct ApplicationCapability {
	unsigned long CapabilityID;
	unsigned long CapabilityValue;
	};
	typedef struct Session{
	string ApplicationProtocolKey;
	unsigned long SessionID;
	sequence<ApplicationRecord> ApplicationRecordList;
	sequence<ApplicationCapability> ApplicationCapabilityList;
	};
	sequence<Session>;

Legal values	NodeID: A number in the range 1001 to 65 535; EntityID: A 16-bit numeric identifier; Active: PV_TRUE/PV_FALSE Ready to receive data; Cond_Cap: PV_TRUE/PV_FALSE Conducting capable; UserApplicationID: A text string Identifies the MBFT user application; NCollapsCapsList: A list of text strings Non-Collapsing capabilities; CapabilityID: A number Identifies an Application Capability; CapabilityValue: A number Dependent from the Capability Class; ApplicationProtocolKey: A text string Identifies the MBFT Protocol; SessionID: A number in the range 1 to 65 535.
Default value	none
Description	The Application Roster includes a list of roster entries for a specific or each MBFT Protocol Session.

I.7.7.3.2.2 Conference Roster service

I.7.7.3.2.2.1 Service description

The service elements and their corresponding XAPI functions needed are described in Table I.7-69.

Table I.7-69/T.180 – Service element and their corresponding XAPI function for Conference Roster

Service element	Service element identifier	XAPI function	Description
Conference Roster Indication	X_MBF_SP_CR_I	x_rcvinfo()	The Service indicates whenever the Conference Roster changes.

I.7.7.3.2.2.2 Service parameters

Table I.7-70 specifies the parameter of the Conference Roster service.

Table I.7-70/T.180 – Parameter of the Conference Roster service

Parameter	Conference Roster service
	Indication
X_MBF_P_CROSTER	M

I.7.7.3.2.2.3 Service parameter descriptions

Table I.7-71 defines the parameter for the Conference Roster service.

Table I.7-71/T.180

Parameter name	X_MBF_P_CROSTER

Type of value	typedef struct Conference {
	unsigned long NodeID;
	unsigned long Node_Type;
	string Node_Name;
	sequence<string<255> > Part_List;
	string Site_Info
	string Net_Addr;
	string User_Data;
	};
	sequence<Conference>;

Table I.7-71/T.180 *(concluded)*

Legal values	NodeID:	A number in the range 1001 to 65 535;	
	Node_Type:	X_CON_PV_TERMINAL, X_CON_PV_MULTIPORT_TERMINAL, or X_CON_PV_MCU;	
	Node_Name:	A text string;	
	Part_List:	A list of text strings	Participants' Names;
	Site_Info	A text string	Additional Information about the node;
	Net_Addr:	A text string	Network address;
	User_Data:	A text string	Additional user data.
Default value	none		
Description	This parameter contains a description of each node joined to the conference. The parameters Node Name, Participants Names, Site Information, Network Address, and User Data in the List of Conference Nodes are conditional.		

I.7.7.3.2.3 Conference Conductor Inquire service

This service may be issued to find out whether the conference is conducted or not, and if so, which node is the conductor, and if the requesting node has been granted conducted-mode permission.

I.7.7.3.2.3.1 Service description

The service elements and their corresponding XAPI functions needed are described in Table I.7-72.

Table I.7-72/T.180 – Service elements and their corresponding XAPI functions for Conference Conductor Inquire

Service element	Service element identifier	XAPI function	Description
Conference Conductor Inquire Request	X_MBF_SP_CCI_Q	x_sndinfo()	The Conference Conductor Inquire Request is passed to the provider to request information about the conductor.
Conference Conductor Inquire Confirmation	X_MBF_SP_CCI_C	x_rcvinfo()	The Conference Conductor Inquire Confirmation is a positive or negative response to a previous information request.

I.7.7.3.2.3.2 Service parameters

Table I.7-73 specifies the parameters of the Conference Conductor Inquire service.

Table I.7-73/T.180 – Parameters of the Conference Conductor Inquire service

Parameter	Conference Conductor Inquire service	
	Request	Confirmation
X_MBF_P_CONDUCTED		M
X_MBF_P_COND_NODE		C
X_MBF_P_PERMISSION		C

I.7.7.3.2.3.3 Service parameter descriptions

Tables I.7-74 to I.7-76 define the parameters for the Conference Conductor Inquire service.

Table I.7-74/T.180

Parameter name	X_MBF_P_CONDUCTED
Type of value	unsigned long
Legal values	PV_TRUE PV_FALSE
Default value	none
Description	Indicates the conducted mode.

Table I.7-75/T.180

Parameter name	X_MBF_P_COND_NODE
Type of value	unsigned long
Legal values	a number between 1001 and 65 535
Default value	none
Description	Indicates the node ID of the current conference conductor. It is only available if the conference is in the conducted mode.

Table I.7-76/T.180

Parameter name	X_MBF_P_PERMISSION
Type of value	unsigned long
Legal values	PV_TRUE PV_FALSE
Default value	none
Description	Indicates whether or not the local node has been granted conducted mode permission. It is only available if the conference is in the conducted mode.

I.7.7.3.2.4 Conductor Report service

I.7.7.3.2.4.1 Service description

The service element and their corresponding XAPI function needed is described in Table I.7-77.

Table I.7-77/T.180 – Service elements and their corresponding XAPI functions for Conductor Report

Service element	Service element identifier	XAPI function	Description
Conductor Report Indication	X_MBF_CONDR_I	x_rcvinfo()	The Conductor Report Indication informs about changes in the conductorship.

I.7.7.3.2.4.2 Service parameters

Table I.7-78 specifies the parameters of the Conductor Report service.

Table I.7-78/T.180 – Parameters of the Conductor Report service

Parameter	Conductor Report Service
	Indication
X_MBF_P_COND_MODE	M
X_MBF_P_REQ_NODE	C

I.7.7.3.2.4.3 Service parameter descriptions

Tables I.7-79 and I.7-80 define the parameters for the Conductor Report service.

Table I.7-79/T.180

Parameter name	X_CON_P_COND_MODE
Type of value	unsigned long
Legal values	PV_TRUE PV_FALSE
Default value	PV_FALSE
Description	This parameter indicates that the conference is now in the conducted mode (PV_TRUE) or in the non-conducted mode (PV_FALSE).

Table I.7-80/T.180

Parameter name	X_CON_P_REQ_NODE
Type of value	unsigned long
Legal values	a number between 1001 and 65 535
Default value	none
Description	Indicates the conductor node identification. This parameter is only meaningful when the conference is now in the conducted mode (X_MBF_P_COND_MODE = PV_TRUE).

I.7.7.3.3 Services which change session parameters

A T.127 MBFT protocol module may change its capabilities at any time by re-enrolling. To do this, the user application issues a service request with the revised parameter values. Any changes of

parameter values will not affect transactions already in progress. Changes will take effect when the next transaction is initiated.

I.7.7.3.3.1 Change Resource service

I.7.7.3.3.1.1 Service description

The Change Resource service is used to request the provider to change the values of specific session-related parameters.

The service elements and their corresponding XAPI functions needed are described in Table I.7-81.

Table I.7-81/T.180 – Service elements and their corresponding XAPI functions for Change Resource

Service element	Service element identifier	XAPI function	Description
Change Resource Request	X_MBF_SP_CHR_Q	x_sndsp()	The Change Resource Request is passed to the provider to request the change of values of session-related parameters.
Change Resource Confirmation	X_MBF_SP_CHR_C	x_rcvsp()	The Change Resource Confirmation is generated by the provider as acknowledgement of the previous request.

I.7.7.3.3.1.2 Service parameters

Table I.7-82 specifies the parameters of the Change Resource service.

Table I.7-82/T.180 – Parameters of the Change Resource service

Parameter	Change Resource service	
	Request	Confirmation
X_MBF_P_COND_OP_CAP	U	
X_MBF_P_M_MODE	U	M
X_MBF_P_LIST_OF_MEMB	U	
X_MBF_P_EXPEL_MEMB	U	
X_MBF_P_MAX_FILE_SIZE	U	C
X_MBF_P_MAX_DAT_PAYL	U	C
X_MBF_P_COMPR	U	C
X_MBF_P_NB_OF_CWOR	U	C
X_MBF_P_MAX_STR_LNGH	U	C
X_MBF_P_NON_STD_CAP	C	C
X_MBF_P_CHR_RESULT		M

I.7.7.3.3.1.3 Service parameter descriptions

Tables I.7-83 to I.7-93 list the parameters of the Change Resource service.

Table I.7-83/T.180

Parameter name	X_MBF_P_COND_OP_CAP
Type of value	unsigned long
Legal values	PV_TRUE PV_FALSE
Default value	PV_FALSE
Description	The Conducting Operation Capable parameter indicates whether the MBFT Protocol module is capable of operating as a session conductor. Each MBFT Session has its own conductor, namely the MBFT Protocol module at the conducting node participating in that Session. If there is more than one MBFT Protocol module at the conducting node in a given MBFT Session, then it is a local matter for GCC at the conducting node to decide which MBFT Protocol module adopts the role of session conductor.

Table I.7-84/T.180

Parameter name	X_MBF_P_M_MODE	
Type of value	unsigned long	
Legal values	X_MBF_PV_MM_REC	Receive files only;
	X_MBF_PV_MM_SND	Transmit files only;
	X_MBF_PV_MM_SND_REC	Transmit and receive files;
	X_MBF_PV_MM_N_SND_REC	Neither transmit nor receive files.
Default value	none	
Description	Indicates the T.127 MBFT protocol module mode. It is dependent from the protocol option X_MBF_O_M_TYPE.	

Table I.7-85/T.180

Parameter name	X_MBF_P_LIST_OF_MEMB
Type of value	sequence<unsigned long>
Legal values	A list of numbers in the range 1001 to 65 535
Default value	none
Description	This parameter is used to invite participants (specified by their NodeIDs) to a private session.

Table I.7-86/T.180

Parameter name	X_MBF_P_EXPEL_MEMB
Type of value	sequence<unsigned long>
Legal values	A list of numbers in the range 1001 to 65 535
Default value	none
Description	This parameter is used to expel members (specified by their NodeIDs) from a private session.

Table I.7-87/T.180

Parameter name	X_MBF_P_MAX_FILE_SIZE	
Type of value	unsigned long	
Legal values	X_MBF_PV_FS_UNLIM	Unlimited;
	X_MBF_PV_FS_MAX_VAL	Maximum file data payload in octets.
Default value	X_MBF_PV_FS_UNLIM	
Description	Indicates the maximum file size. Each T.127 MBFT protocol module must specify the maximum file data payload in octets that it is capable of receiving.	

Table I.7-88/T.180

Parameter name	X_MBF_P_MAX_DAT_PAYL	
Type of value	unsigned long	
Legal values	X_MBF_PV_DP_DEF	Default value (8192 octets)
	X_MBF_PV_DP_MAX_VAL	Maximum number (less or equal 65 536 octets)
Default value	X_MBF_PV_DP_DEF	
Description	Indicates the maximum data payload. This is the maximum number of octets allowed in the data field of T.127 MBFT start and data PDUs.	

Table I.7-89/T.180

Parameter name	X_MBF_P_COMPR
Type of value	unsigned long
Legal values	PV_TRUE PV_FALSE
Default value	PV_FALSE
Description	This parameter is used to negotiate use of V.42 *bis* compression for file data. T.127 MBFT protocol modules should assert this capability if they are able to receive V.42 *bis* compressed data.

Table I.7-90/T.180

Parameter name	X_MBF_P_NB_OF_CWOR
Type of value	unsigned long
Legal values	X_MBF_PV_CW_DEF Default value (512 codewords) X_MBF_PV_CW_NUM Total number of codewords (less or equal 65 536).
Default value	X_MBF_PV_CW_DEF
Description	Specifies the total number of codewords to be used by the V.42 *bis* compression algorithm. This is an upper bound on V.42 *bis* parameter P1.

Table I.7-91/T.180

Parameter name	X_MBF_P_MAX_STR_LNGH
Type of value	unsigned long
Legal values	X_MBF_PV_STR_DEF Default value (6) X_MBF_PV_STR_VAL Maximum string length (less or equal 250)
Default value	X_MBF_PV_STR_DEF
Description	Specifies the maximum string length input to the V.42 *bis* encoder. This is an upper bound on V.42 *bis* parameter P2.

Table I.7-92/T.180

Parameter name	X_MBF_P_NON_STD_CAP
Type of value	unsigned long
Legal values	X_MBF_PV_NSC_UNSP Unspecified
Default value	X_MBF_PV_NSC_UNSP
Description	This parameter is used to negotiate non-standard functions, including non-standard compression techniques.

Table I.7-93/T.180

Parameter name	X_MBF_P_CHR_RESULT
Type of value	unsigned long
Legal values	X_MBF_PV_CHR_SUCC Success; X_MBF_PV_R_T_CON Rejection; temporary congestion; X_MBF_PV_R_DOM_DIS Rejection; conference domain disconnected; X_MBF_PV_R_TM_RES Rejection; too many resources; X_MBF_PV_R_TM_USER Rejection; too many users; X_MBF_PV_R_NO_REASON Rejection, reason not specified.
Default value	X_MBF_PV_CHR_SUCC
Description	Indicates the success or failure of the Change Resource Request.

I.7.7.4 Disconnect Service

I.7.7.4.1 Service description

The Disconnect service is used to disconnect a member from the session. The Disconnect may be performed:

- by any session member;
- by the session provider.

The Disconnect service does not guarantee delivery of information once the release phase is entered.

The service elements and their corresponding XAPI functions needed for disconnection are described in Table I.7-94.

Table I.7-94/T.180 – Service elements and their corresponding XAPI functions for Disconnect

Service element	XAPI function	Description
Disconnect Request	x_snddis()	The Disconnect Request is passed to the provider to request a session disconnection.
Disconnect Indication	x_rcvdis()	The Disconnect Indication is generated by the provider to indicate the session disconnection initiated by the session provider.
End Indication	x_rcvend()	The End Indication is generated by the provider to indicate that the service provider is ready to establish a new connection.

I.7.7.4.2 Service Parameters

There are no MBFT protocol-specific parameters of the Disconnect service.

I.7.7.5 Tables of error codes

The XAPI error-level error codes are defined in Annex B.

I.7.7.5.1 CC_BADVALUE

If the cause code indicates a parameter error with a bad value, the value of *diagnostic* will contain the erroneous parameter identifier which has been submitted with the XAPI function call that caused the error indication.

I.7.7.5.2 CC_MANDMISS

If the cause code indicates a mandatory parameter is missing, the value of *diagnostic* will contain the missing parameter identifier that caused the error indication.

I.7.7.5.3 CC_BADEVENT

If the cause code indicates a bad event, the value of *diagnostic* will contain the bad event identifier which has been submitted with the XAPI function call that caused the error indication.

I.7.7.5.4 CC_UNEXPECT

If the cause code indicates a unexpected event, the value of *diagnostic* will contain the actual state identifier in which the unexpected event caused the error indication.

I.7.7.5.5 CC_NOTSUPPORT

If the cause code indicates an unsupported event, the value of *diagnostic* will contain the identifier of the unsupported event which has been submitted with the XAPI function call that caused the error indication.

I.7.7.5.6 CC_OTHER

If the cause code indicates the CC_OTHER error code, the value of *diagnostic* will contain the identifier which caused the error indication.

APPENDIX II

Tutorial: XAPI and selected providers

This appendix describes the overall structure of selected providers and gives some information with respect to the XAPI. Subclauses I.4 (ACSE/ROSE), I.6 (T.120 Conference Control), and I.7 (MBFT) supply the reader with the full information needed to access these providers via the XAPI.

II.1 XAPI and the ACSE/ROSE provider

In this subclause the overall structure of the OSI Association Control Service Element (ACSE) and the OSI Remote Operation Service Element (ROSE) is presented. These services describe the communication between two OSI application entities, namely an invoking (or client) application entity, a performing (or server) application entity and their usage of remote operation services.

The application entities will first establish an application association and then, in the connected state, will use remote operations for communication purposes.

ACSE provides the elements that are used to manage the establishment and release of an application association. The association control service is provided by the service primitives:

- A-ASSOCIATE request/indication/response/confirmation;
- A-RELEASE request/indication/response/confirmation;
- A-ABORT request/indication;
- A-P-ABORT indication.

Recommendation X.227 specifies the protocol for the ACSE. The protocol data units are listed below:

- A-ASSOCIATE-REQUEST application-protocol-data-unit (AARQ);
- A-ASSOCIATE-RESPONSE application-protocol-data-unit (AARE);
- A-RELEASE-REQUEST application-protocol-data-unit (RLRQ);
- A-RELEASE-RESPONSE application-protocol-data-unit (RLRE);
- A-ABORT application-protocol-data-unit (ABRT).

ROSE provides the capability to request that a process be executed on the same or on another system. It passes a description of the operation and associated parameters from the client to the server that may provide a result on the outcome. The ROSE services are provided in conjunction with the ACSE.

The remote operation service is provided by the service primitives:

- RO-INVOKE request/indication;
- RO-RESULT request/indication;
- RO-ERROR request/indication;
- RO-REJECT-U request/indication;
- RO-REJECT-P indication.

Recommendation X.229 specifies the protocol and procedures for the ROSE. The protocol data units are listed below:

- RO-INVOKE application-protocol-data-unit (ROIV);
- RO-RESULT application-protocol-data-unit (RORS);
- RO-ERROR application-protocol-data-unit (ROER);
- RO-REJECT application-protocol-data-unit (RORJ).

Figure II.1 shows the structure of the protocol stack that is accessible via the XAPI when selecting an provider comprising ACSE and ROSE. The ROSE service is provided by the combination of a Transport system, the Session, the Presentation, and the ROSE protocol module, selected by the application.

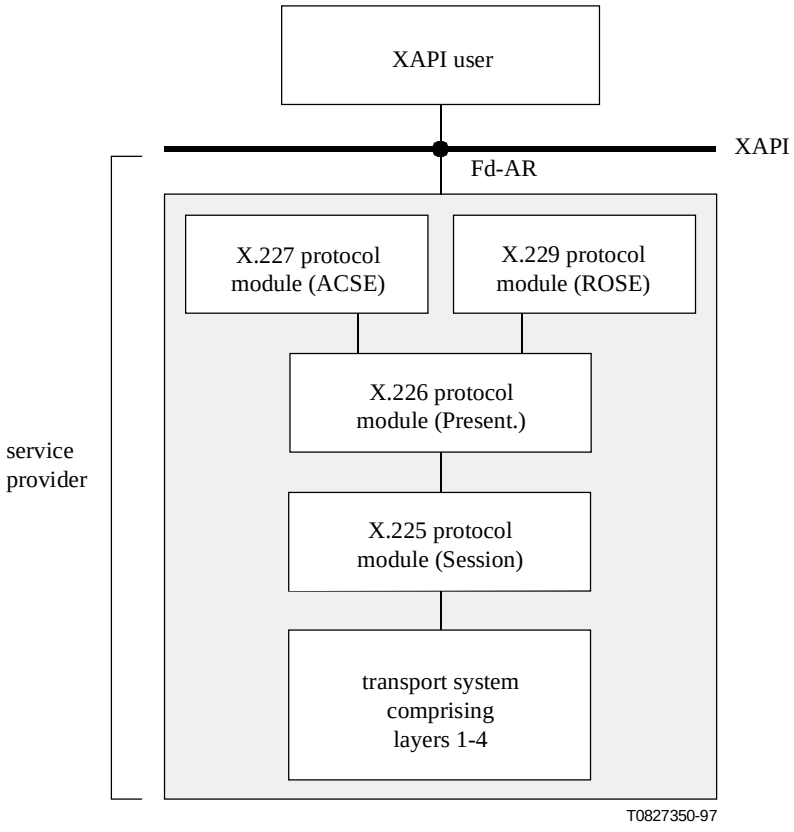


Figure II.1/T.180 – Structure of the ACSE/ROSE provider

Figure II.2 shows the client/server model. Both systems are structured as shown in Figure II.1. In addition, two ROSE protocol-data-units are depicted which may serve as examples for the communication between the two application entities.

An application can:

- invoke operations by one ROSE user and perform operations by an other ROSE user;
- perform the control of the application association by ACSE.

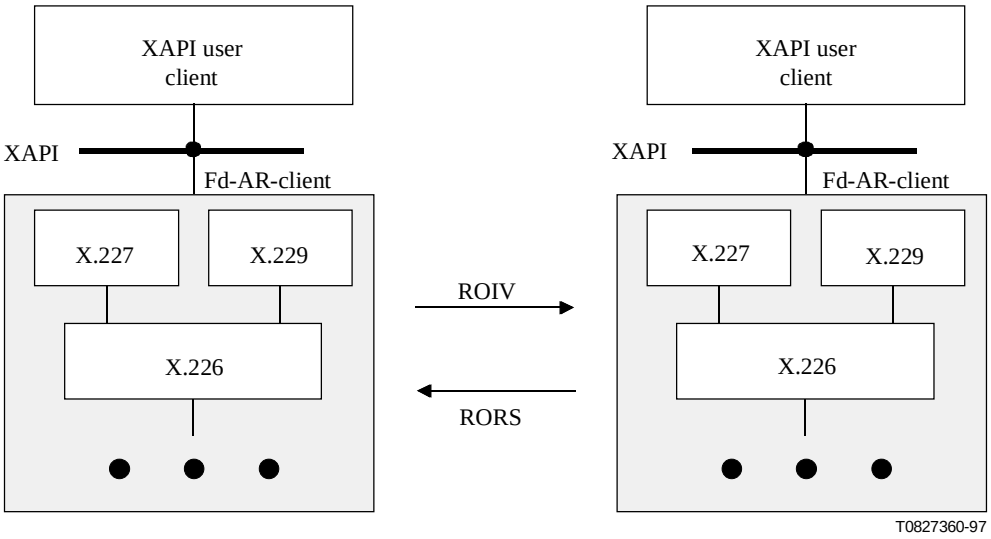


Figure II.2/T.180 – Client/server model based on XAPI ACSE/ROSE provider

Figure I.3 shows an example sequence of XAPI function calls. The client system initiates an association, connecting the client to the server. Afterwards, and in the connected state, the client requests a remote service operation which is performed by the server. The positive reply is returned to the client.

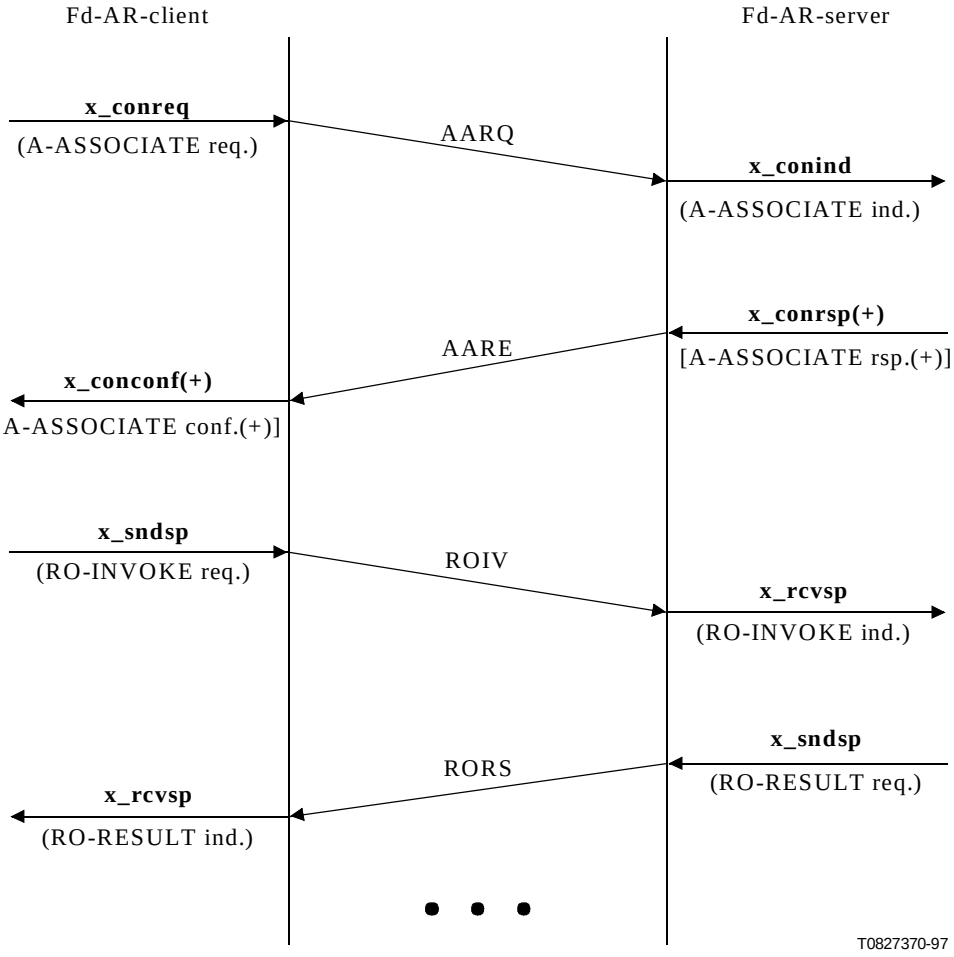


Figure II.3/T.180 – An example sequence

A service endpoint (e.g. Fd-AR-client) accessing the ACSE/ROSE provider is created when using the x_open function and is activated when using the x_bind function:

x_open

– Name of the service provider ("X_ACSE_ROSE_ISDN");
– Execution mode;
– Information (e.g. characteristics of the local service provider);
– xerror (error codes or the parameter *Fd-AR-client*).

x_bind

– Fd-AR-client;
– Name of the transport system (not needed in the standard configuration);
– Requested own protocol address;
– Bound own protocol address;
– Information;
– xerror (error codes).

In the standard configuration, if "X_ACSE_ROSE_ISDN" was selected in the x_open function, no transport system has to be specified as argument of the x_bind function as the service provider's protocol stack is already complete.

Connection establishment is performed by ACSE. The address to be used to identify the peer entity (called address) is the tuple of (Presentation-selector, Session-selector, Transport-selector, NSAP Address), where the selectors are optional depending on the peer's requirements.

x_conreq

– Fd-AR-client,
– Called address (see above), user data, protocol parameters (see below),
– xerror.

The protocol parameters are listed below (list not complete). For brevity, only those parameters are listed which apply for the Request primitive and which are mandatory for this primitive.

– X_ACS_P_APP_CTXT: This parameter identifies the application context. In the Response or Confirmation primitive either the same or a different application context is returned.
– X_ACS_P_CTXT_ID: This parameter identifies the presentation context identification.
– X_ACS_P_AS: This parameter indicates the abstract syntax name.
– X_ACS_P_SUR: This parameter indicates the session user requirements.

ROSE defines five different Operation Classes which classify operations according to two possible operation modes (synchronous and asynchronous). The Association Class defines which ROSE user is allowed to invoke operations. The Operation Class and Association Class have to be agreed between ROSE users. It is not part of the ROSE provider to negotiate these features. OSI Applications making use of ROSE (e.g. DTAM, MHS) define which Operation Class and Association Class are allowed within a specific application.

Due to the fact that there is no negotiation of Operation and Association Classes within the ROSE provider, the parameters Invoke-Id (to identify an operation and to correlate the request of an invocation with its replies) and Linked-Id (in the case of a child operation to identify the parent operation) can not be examined by ROSE. Especially it is not controlled, if an invocation exists to an incoming reply, i.e. result, error or rejection or if the specified parent operation exists to the invocation of an child operation.

ROSE does not define a distinct abstract syntax for the encoding of its PDUs. Instead, it provides a set of abstract syntax definitions that are used by the application making use of ROSE. Therefore, the service user must inform the service provider which abstract syntax are used within the ROSE PDUs. To support this feature, each ROSE service primitive must be provided with a service primitive parameter (X_ROS_P_CTXT_ID) that indicates the abstract syntax of the application.

While the service endpoint used to access the provider is in the connected state, the corresponding service primitives can be passed to the provider or retrieved from the provider using the XAPI function calls x_sndsp and x_rcvsp.

x_sndsp/x-rcvsp

– Fd-AR-client;
– Level (identifies the protocol module which shall get the service primitive);
– Spname (e.g. RO-INVOKE);
– Sp (parameters of the service primitive (see below) and user data);
– xerror.

The RO-INVOKE primitive is used to request the start of a remote operation. The parameters of this primitive are listed below.

– X_ROS_P_CTXT_ID: This parameter indicates the presentation context identification of the application data of the RO-INVOKE primitive. This parameter is the presentation context identification which was negotiated in the ACSE establishment for the application.

– X_ROS_P_INV_ID: This parameter identifies the RO-INVOKE primitive and is used to correlate this primitive with the corresponding replies.

– X_ROS_P_LINK_ID: This parameter identifies a child-operation and identifies the invocation of the linked parent-operation. The value is that of the X_ROS_P_INV_ID parameter of the RO-INVOKE indication primitive of the parent-operation.

– X_ROS_P_VAL_INT: This parameter indicates the identifier of the operation to be invoked. As the identifier of the operation may either be an integer value or an object identifier, either this parameter (legal values are any integer values) or the parameter X_ROS_P_VAL_ID may be present.

– X_ROS_P_VAL_ID: This parameter indicates the identifier of the operation to be invoked. As the identifier of the operation may either be an integer value or an object identifier, either this parameter or the parameter X_ROS_P_VAL_INT may be present.

II.2 XAPI and the specific T.120 conference provider

II.2.1 The T.120 system model

The T.120 protocol provides a means of telecommunicating all forms of data/telematic media between two or more multimedia terminals and of managing such communication. They provide a multipoint data communication service that has a particular application in multimedia conferencing.

The T.120 protocol can handle one or more simultaneous "conferences"; any terminal may participate in more than one of these if authorized to do so; the convener of any one conference may control the participation in that conference and the flow of information in that conference.

Audio/Video applications and/or data applications such as File Transfer or Still Image may be used in a conference.

Figure II.4 shows the structure of a T.120-series-based Conference platform. Furthermore, the location of the XAPI is depicted.

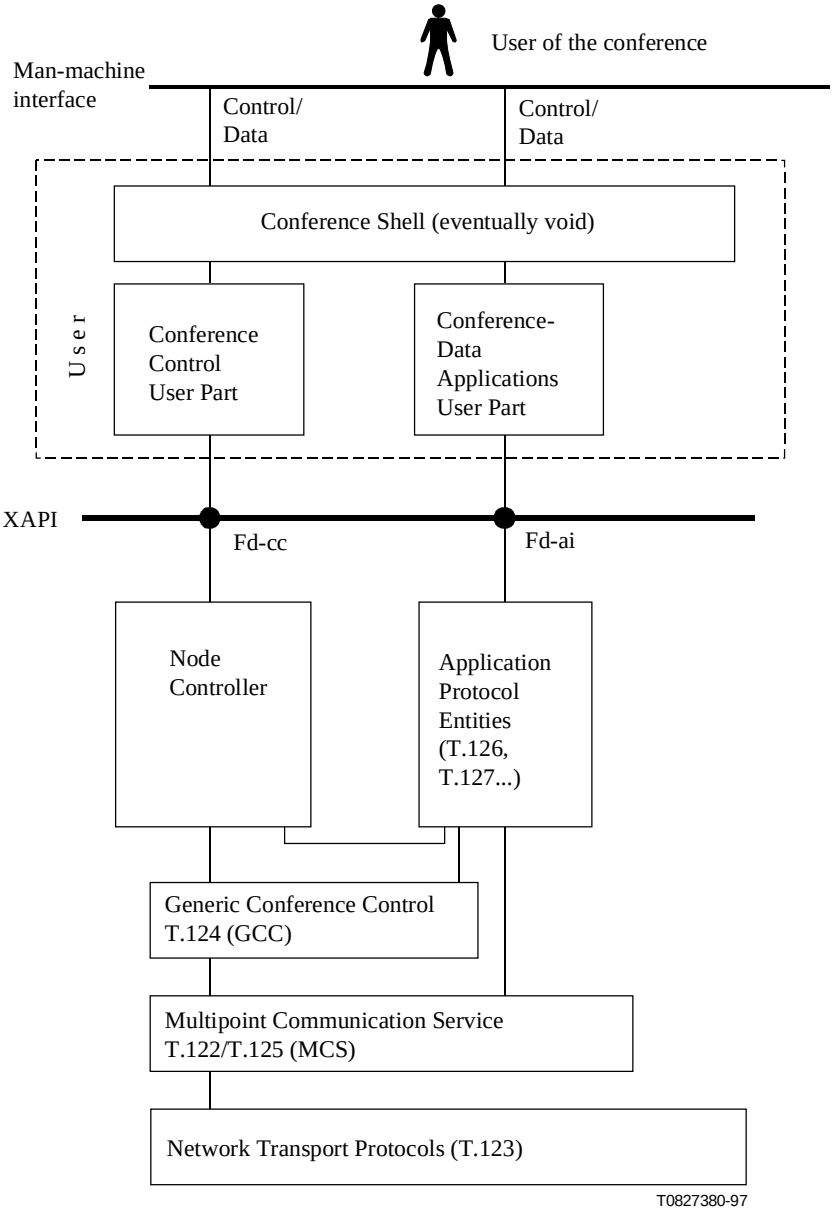


Figure II.4/T.180 – Structure of a T.120-series conference terminal

The XAPI conference user (see Figure 12) comprises the user parts of:

– the Conference Control communicating with the provider (i.e. the Node Controller) via the service endpoint Fd-cc,

– zero, one or more (standard or non-standard) Data Applications each communicating with the provider via a separate service endpoint Fd-a1, ..., -ak.

A Conference Shell functionality making the access of a human user to conference services easier may or may not be part of the XAPI user.

The XAPI conference provider comprises the:

– Node Controller;

– (standard or non-standard) Data Application Protocol Entities;

– the remaining provider functions as shown in Figure II.4 (GCC, MCS, ...).

The term "Node Controller" is used to describe the element that provides the T.120 management function or role at a terminal or Multipoint Control Unit (MCU).

The T.124 Generic Conference Control (GCC) provides a set of services for setting up and managing the multipoint conference.

The T.122/T.125 Multipoint Communication Service (MCS) provides a general multipoint connection-oriented data service. It collects point-to-point Transport connections and combines them to form a Multipoint Domain. Within that Domain a large number of logical channels are provided that can provide one-to-one, one-to-many and many-to-one data delivery.

II.2.2 T.120 MBFT conferencing

Recommendation T.127 defines the Multipoint Binary File Transfer (MBFT) Protocol. This protocol supports the interchange of binary files within an interactive conferencing or group working environment where the T.120-series of Recommendations is in use.

A basic file transfer application conforming to this Recommendation may simply offer the ability to broadcast one file at a time to all applications which support the MBFT protocol. Optional advanced features defined in T.127 include:

– broadcast of multiple files simultaneously;

– "private" distribution of files to a selected subset of the conference;

– conductor control of file distribution.

Prior to activating a file transfer application, the user has to create or to join a conference (the node that creates a conference is called the convener). In general, the XAPI conference user at a terminal has access to conference management functions such as:

c1 information services to determine what conferences are currently in existence;

c2 creation of a conference;

c3 joining an existing conference;

c4 leaving a conference;

c5 adding a node to an existing conference;

c6 terminating the entire conference;

c7 forcing a particular node to be disconnected from a conference;

c8 requesting that an Application Protocol Entity be invoked at a specified set of nodes,

... ...

via the service endpoint Fd-cc (see Figure 12).

Functions such as disconnecting a node from a conference are only defined for the convener or some other specific nodes.

Figure II.5 shows an example configuration. User A (at terminal A) creates a call-through conference, i.e. A calls into MCU K and adds the users B (at terminal B) and C (at terminal C), which are called by that MCU.

A Multipoint Control Unit (MCU) is a special network element that serves to connect terminals and other MCU's in a multipoint fashion.

NOTE – The example given in Figure II.5 represents the same conference configuration as that given in clause 7 (e.g. Figure 7).

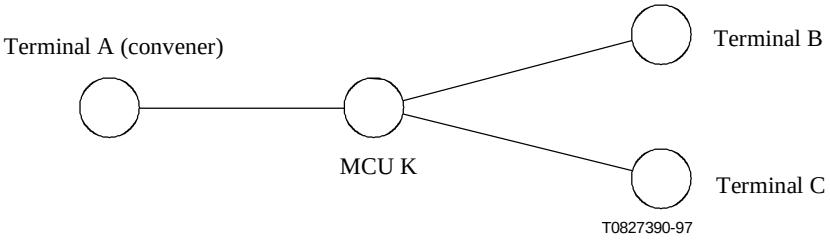


Figure II.5/T.180 – An example conference configuration

Figure II.6 shows an example sequence at terminal A sketching the creation of the conference specified above in terms of XAPI function calls.

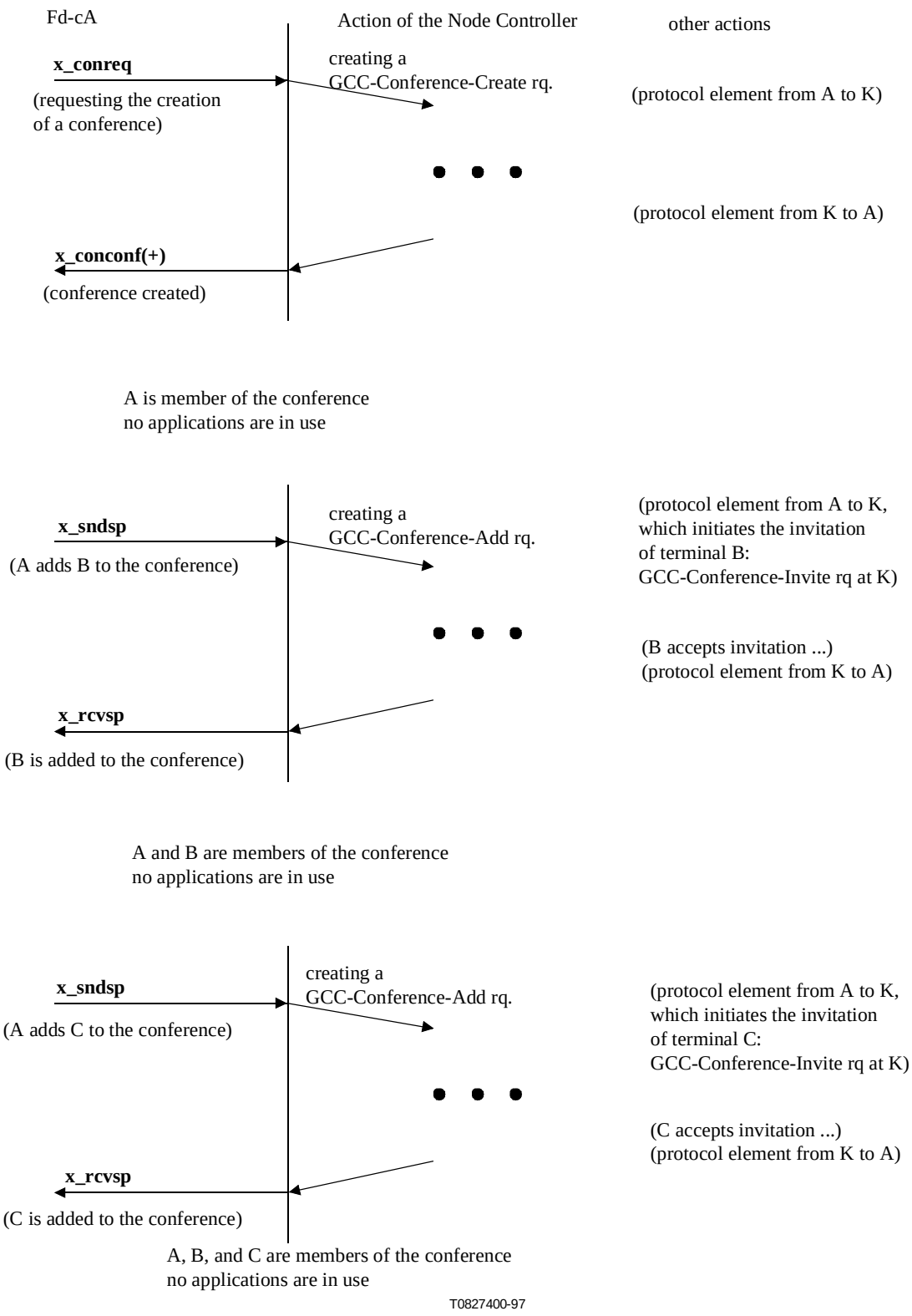


Figure II.6/T.180 – Creation of a conference example sequence

The service endpoint Fd-cA accessing the conference control provider (Node Controller of terminal A) is created when using the x_open function and is activated when using the x_bind function:

x_open

– Name of the service provider ("X_T.120_CONF_ISDN", ...);

– Execution mode;
– Information (e.g. characteristics of the local service provider);
– xerror (error codes or the parameter "Fd-cA").

x_bind

– Fd-cA;
– Requested own protocol address;
– Bound own protocol address;
– Information;
– xerror.

Communication establishment (creating a new conference at MCU K to which terminal A is automatically joined) is supervised by the Node Controller. User A initiates the function:

x_conreq

– Fd-cA;
– Called address, user data, protocol parameters (see below);
– xerror.

The protocol parameters are listed below (list not complete):

– Conference Name: Name by which the conference to be created is identified;
– Conference Locked: Setting this flag immediately locks a conference;
– Conference Listed: The TRUE setting of this flag indicates that this conference may be listed;
– Conference Conductible: The TRUE setting of this flag indicates that this conference may be placed in conducted mode;
– Termination Method: This parameter indicates whether the conference shall remain in existence until explicitly terminated by the Convener or convener-designated node.

Having successfully performed all actions shown in Figure II.6, the XAPI users A (convener), B and C are members of the conference. No applications are in use at this point in time. At that time, the control connection is defined by the tuple (Fd-cA, Fd-cB, Fd-cC, K). It consists in the GCC Broadcast Channel, the Convenor Channel, and – for each node – the Node ID channel (see Annex A/Recommendation T.120).

In the remainder of this subclause, MBFT-related structures and actions are described.

The conference user at a terminal has access to MBFT management and information transfer functions such as:

f1 connecting the MBFT application to the conference;

f2 offer a file;

f3 accept a file,

... ...

via the service endpoint Fd-file-transfer.

The purpose of the example conference is to "broadcast" files from one participant to others. Therefore, a file transfer application is used which is supported by the MBFT T.127 protocol.

Two types of channels are used within T.127: control channels and data channels. Control channels are used for managing all aspects of the file transfer (offering files, requesting files), whereas data channels are used exclusively for the transfer of file data. Only one file can be transmitted on each

data channel at a time, but additional data channels can be used to allow distribution of multiple files simultaneously.

Figure II.7 shows the terminals, the MCU, and the MBFT channel structure for the example configuration. MBFT-CHANNEL-0 is the name of the control channel and MBFT-CHANNEL-1 is the name of a broadcast data channel on which files are distributed. Fd-fA, Fd-fB, and Fd-fC are the respective service endpoints.

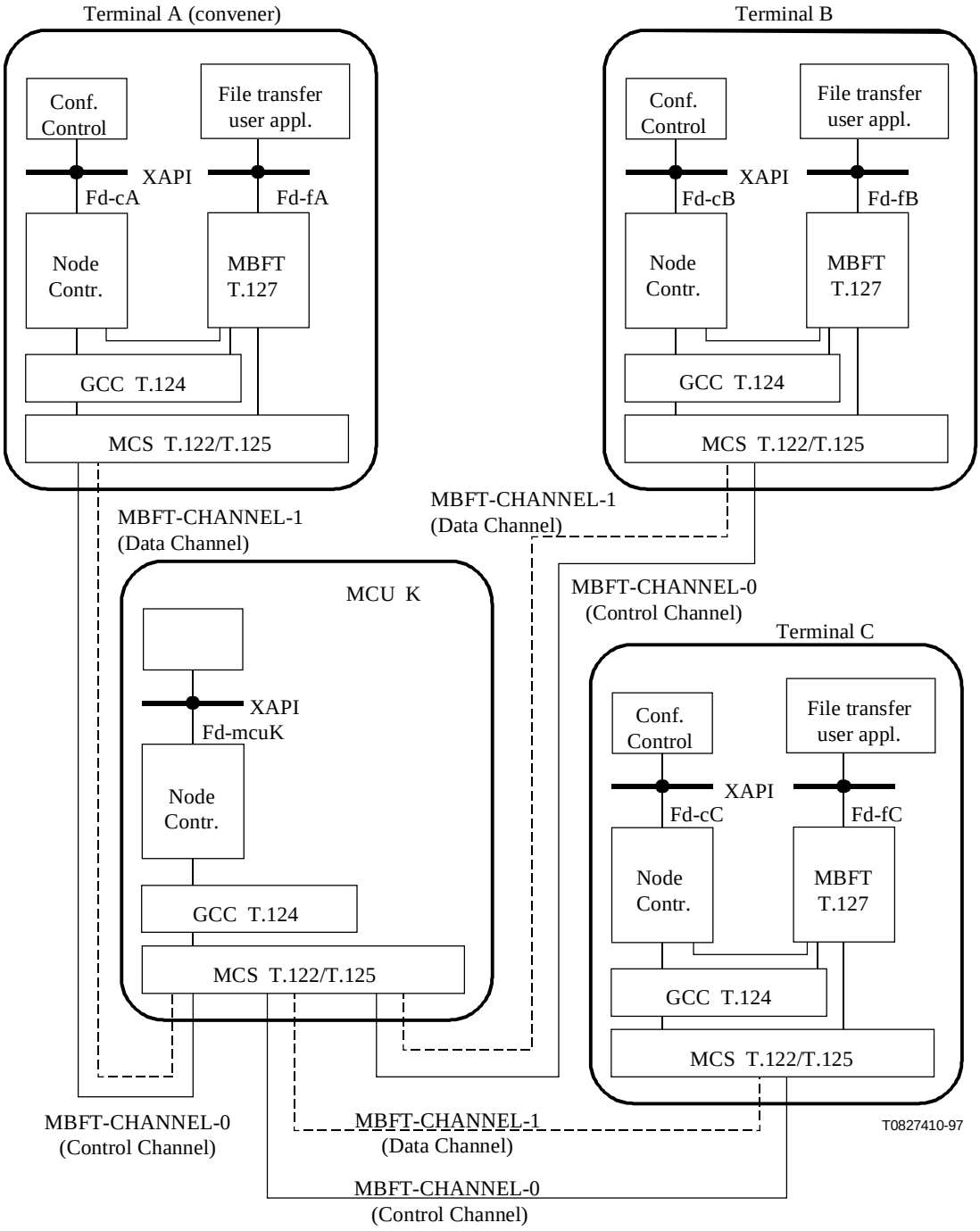


Figure II.7/T.180 – The MBFT channel structure (example)

Other channels, such as the GCC Broadcast Channel, or the Convener Channel are not shown in Figure II.7.

NOTE – In general, the functionality of a MCU will differ from that of a terminal. Nevertheless, using a XAPI for MCUs as shown in Figure II.7 may contribute to a more sophisticated architecture of such a node and may ease software development.

A MBFT session is characterized by:

– a single control channel (see Figure II.7);

– a single broadcast data channel (see Figure II.7);

– zero or more acknowledged data channels (not used for the example);

– zero or more private sub-sessions (not used for the example);

– a session ID.

A MBFT session may now be initiated locally by the user application or remotely through use of the GCC-Application-Invoke mechanism.

Figure II.8 shows user A connecting the file transfer application to the conference and receiving a file.

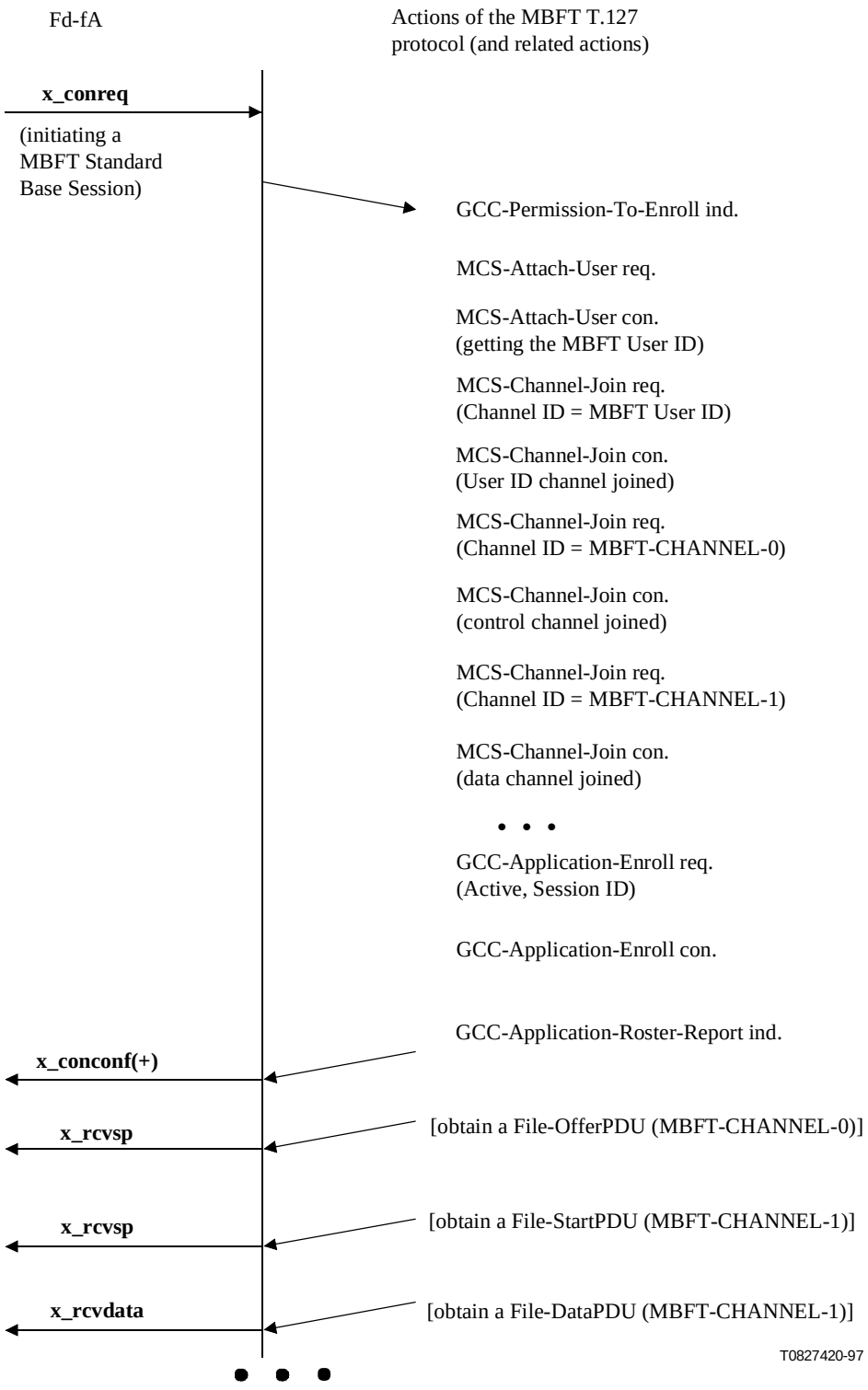


Figure II.8/T.180 – User A: Connecting the file transfer application to the conference and receiving a file

After having obtained a MBFT User ID, the T.127 protocol shall join control and data channels by issuing two MCS-Channel-Join requests, specifying MBFT-CHANNEL-0 and MBFT-CHANNEL-1 as respective channels to join. Once positive confirmation of joining these channels has been received ("channel joined" in Figure II.8), the application shall enrol active by issuing a GCC-Application-Enrol request from the T.127 protocol to the GCC provider.

Afterwards, files may be exchanged. Figure II.8 shows the receipt of a file.

A service endpoint (e.g. Fd-fA) accessing the MBFT T.127 protocol is created when using the x_open function and is activated when using the x_bind function:

x_open

– Name of the service provider ("X_T.127_MBFT", ...);
– Execution mode;
– Information (e.g. characteristics of the local service provider);
– xerror (error codes or the parameter "Fd-fA").

x_bind

– Fd-fA;
– xerror.

During the connection establishment phase, an MBFT user establishes a connection (i.e. creating a new session or joining an already existing session) to other MBFT users. The connection is identified by the conference name and the session identifier respectively.

x_conreq

– Fd-fA;
– Conference name, session type, session identifier, other parameters;
– xerror.

Some protocol parameters are listed below:

– Conference Name: Name by which the conference is identified;
– Session type: Parameter by means of which the session type is identified;
– Session Identifier: Number by which the session is identified;
– Other parameters: e.g. indicating the maximum file size, negotiating use of V.42 *bis* compression for file data, ...

NOTE – In the example above the application connection is defined by (Fd-fA, Fd-fB, Fd-fC). It consists in MBFT-CHANNEL-0, MBFT-CHANNEL-1, and – for each MBFT application protocol entity – the MBFT User ID Channel.

ITU-T RECOMMENDATIONS SERIES

Series A	Organization of the work of the ITU-T
Series B	Means of expression: definitions, symbols, classification
Series C	General telecommunication statistics
Series D	General tariff principles
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media, digital systems and networks
Series H	Audiovisual and multimedia systems
Series I	Integrated services digital network
Series J	Transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Construction, installation and protection of cables and other elements of outside plant
Series M	TMN and network maintenance: international transmission systems, telephone circuits, telegraphy, facsimile and leased circuits
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Telephone transmission quality, telephone installations, local line networks
Series Q	Switching and signalling
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	**Terminals for telematic services**
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	Data networks and open system communications
Series Y	Global information infrastructure
Series Z	Programming languages