EUROPEAN TELECOMMUNICATION STANDARD

ETS 300 835

September 1998

Source: MTA

Reference: DE/MTA-001043

ICS: 33.020

Key words: API, FT, ISDN, PCI, terminal

Multimedia Terminals and Applications (MTA); Programmable Communication Interface (PCI) for file transfer over Integrated Services Digital Network (ISDN)

ETSI

European Telecommunications Standards Institute

ETSI Secretariat

Postal address: F-06921 Sophia Antipolis CEDEX - FRANCE
Office address: 650 Route des Lucioles - Sophia Antipolis - Valbonne - FRANCE
Internet: secretariat@etsi.fr - http://www.etsi.fr - http://www.etsi.org

Tel.: +33 4 92 94 42 00 - Fax: +33 4 93 65 47 16

Contents

Foreword

This European Telecommunication Standard (ETS) has been produced by the Multimedia Terminals and Applications (MTA) Project of the European Telecommunications Standards Institute (ETSI).

Transposition dates	
Date of adoption of this ETS:	18 September 1998
Date of latest announcement of this ETS (doa):	31 December 1998
Date of latest publication of new National Standard or endorsement of this ETS (dop/e):	30 June 1999
Date of withdrawal of any conflicting National Standard (dow):	30 June 1999

Blank page

1 Scope

This European Telecommunication Standard (ETS) specifies the Programmable Communication Interface (PCI) for interactive services, i.e. for file transfer services over Integrated Services Digital Network (ISDN). In particular it defines the access to and administration of the file transfer services:

- File Transfer, Access and Management (FTAM) simple file transfer as defined in ETS 300 388 [5];
- Eurofile transfer as defined in ETS 300 383 [4].

The simple file transfer teleservice over ISDN was specified in ETS 300 388 [5] in conformance with the international standard FTAM (ISO/IEC 8571, parts 1 to 5 [9]) and the ISO/IEC profile specification AFT11 (see ISO/IEC 10607-3 [10]). The FTAM simple file transfer over ISDN is a subset of FTAM (File Transfer, Access and Management) and is called SimpleFTAM in the present document.

Eurofile is an ISDN teleservice specified in ETS 300 383 [4], in which end-to-end compatibility between terminals is guaranteed and which supports file exchanges between different types of equipment. This teleservice is also called Easyfile.

This ETS completes the work related to the need for a standardized simple file transfer protocol over ISDN by fulfilling the part 3 of the overall project. The outcome of part 1 of this overall project was ETR 074 [8], the outcome of part 2 was ETS 300 383 [4] and ETS 300 388 [5]. This ETS is therefore based upon the documents ETR 074 [8], ETS 300 383 [4] and ETS 300 388 [5].

Additionally the PCI of this ETS was specified according to the philosophy, the general principles and rules of the APPLI/COM interface. APPLI/COM is a PCI which was designed to unify the access and the administration of facsimile group 3, facsimile group 4, teletex, telex, e-mail and file transfer in the master-slave relationship. This ETS establishes therefore an extension of ETS 300 243-1 [3], the APPLI/COM specification, for interactive services, i.e. for file transfer services, defining:

- the services offered at the PCI;
- the information exchanged at the PCI;
- the method how the information are exchanged at the PCI.

The exchange method of this ETS shall be a general mechanism applicable to different types of interactive services. Therefore this ETS shall be used as a framework for future recommendations which addresses other file transfer protocols or interactive services and other networks than ISDN.

2 Normative references

This ETS incorporates by dated and undated reference, provisions from other publications. These normative references are cited at the appropriate places in the text and the publications are listed hereafter. For dated references, subsequent amendments to or revisions of any of these publications apply to this ETS only when incorporated in it by amendment or revision. For undated references the latest edition of the publication referred to applies.

[1] ETS 300 075: "Terminal Equipment (TE); Processable data, File transfer".

[2] ETS 300 079 (1991): "Integrated Services Digital Network (ISDN); Syntax-based videotex End-to-end protocols, circuit mode DTE-DTE".

[3] ETS 300 243-1 (1995): "Terminal Equipment (TE); Programmable Communication Interface (PCI) APPLI/COM for facsimile group 3, facsimile group 4, teletex and telex services; Part 1: CCITT Recommendation T.611 (1992) [modified]".

[4] ETS 300 383 (1995): "Integrated Services Digital Network (ISDN); File transfer over the ISDN EUROFILE transfer profile".

[5] ETS 300 388 (1995): "Integrated Services Digital Network (ISDN); File Transfer Access Management (FTAM) over ISDN based on simple file transfer profile".

[6] ETS 300 409 (1995): "Integrated Services Digital Network (ISDN); Eurofile transfer teleservice; Service description".

[7] ETS 300 410 (1995): "Integrated Service Digital Network (ISDN): File Transfer & Access Management (FTAM) teleservice; Service description".

[8] ETR 074 (1993): "Terminal Equipment (TE); File transfer over the Integrated Services Digital Network (ISDN)".

[9] ISO/IEC 8571 (1988): "Information processing systems - Open Systems Interconnection - File Transfer, Access and Management:
Part 1: General introduction;
Part 2: Virtual Filestore Definition;
Part 3: File Service Definition;
Part 4: File Protocol Specification;
Part 5: Protocol Implementation Conformance Statement Proforma (1990)".

[10] ISO/IEC 10607-3 (1995): "Information technology - International Standardized Profiles AFTnn - File Transfer, Access and Management; Part 3: AFT11 - Simple File Transfer Service (unstructured)".

[11] ITU-T Recommendation Q.931: "Digital Subscriber Signalling System No 1 (DSS 1); ISDN user-network interface layer 3 specification for basic call control".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of this ETS, the following definitions apply:

m: Any feature denoted by "m" (mandatory) shall be supported by all implementations claiming conformance to this ETS.

o: Any feature denoted by "o" (optional) is left to the implementations claiming conformance to this ETS whether this feature is implemented or not.

c: Any feature denoted by "c" (conditional) shall be supported under the conditions specified in this ETS by all implementations claiming conformance to this ETS.

x: Any feature denoted by "x" (excluded) shall not be implemented by all implementations claiming conformance to this ETS.

i: Any feature denoted by "i" (outside the scope) is outside the scope of this ETS.

-: Any feature denoted by "-" (not applicable) is not applicable for the context of this ETS.

3.2 Abbreviations

For the purposes of this ETS, the following abbreviations apply:

ASE	Application Service Element
CA	Communication Application
DLL	Dynamic Link Library
FADU	File Access Data Unit
FTAM	File Transfer, Access and Management
FTP	File Transfer Protocol
FU	Functional Unit
ICE	Interface Configuration Environment
IDU	Interface Data Unit
ISDN	Integrated Service Digital Network
LA	Local Application
LAN	Local Area Network

MSN	Multiple Subscriber Number
NBS	National Bureau of Standards
NSAP	Network Service Access Point
PCI	Programmable Communication Interface
RIC	Redirection of Incoming Calls
UA	User Agent
VFS	Virtual File Store (FTAM)

4 Overview

The interface specified in this ETS is designed for interactive communications services, i.e. for the file transfer services SimpleFTAM and Easyfile.

The general model, the general principles and aspects which have guided the specification of the Programmable Communication Interface (PCI) for file transfer over ISDN are described in clause 5.

Clause 6 contains a description of the flow of information at the interface specified in this ETS.

The functions which shall be used to exchange information at the interface, the parameters which shall be used independently from the file transfer service at the interface and principles which shall be used to handle and manage the exchange of information at the interface are specified in clause 7.

The service primitives used to exchange information at the interface are defined in clause 8.

The meaning, the structure and the data types of the Interface Data Units are defined in clause 9. This clause contains:

- the specification of general, service independent parameters of each Interface Data Unit (IDU);

- the SimpleFTAM service definition comprising the Interface Data Units (IDUs) and their parameters as applicable for SimpleFTAM;

- the Easyfile service definition comprising the Interface Data Units (IDUs) and their parameters as applicable for Easyfile.

The interface of this ETS was specified independently from programming languages, hardware platforms and operating systems. Implementors who wants to develop a product according to this ETS will face parts which are product specific and which depend on the real computing environment. The decisions how to realize such parts are left to the product developers. To guarantee that the components of a communications system are capable of communicating with each other, those product specific features and services shall be provided by the manufacturer of a product as described in clause 10.

The binary encoding scheme of the IDUs are specified in clause 11 using the programming language C as a formal description language.

Clause 12 contains the most important platform dependencies to assist implementors to develop their product according to this ETS.

Annex A contains the description of error codes.

Two informative annexes are provided in this ETS:

- Annex B: File Transfer Protocol (FTP) service definition. The file transfer service FTP is a widely spread and accepted standard. Annex A shows how to use the FTP service at the PCI of this ETS.

- Annex C: Examples of information exchange. This annex illustrates the exchange of information at the PCI using concrete examples.

5 General principles

The general model, the general principles and aspects which have guided the specification of the PCI of this ETS are defined in the following subclauses.

5.1 General model

The interface specified in this ETS was designed to be independent from:

- operating systems;
- hardware platforms;
- programming languages.

The interface of this ETS is a Programmable Communication Interface (PCI) in an open communications system. Such a communications system comprises a set of components which are located either on the same physical medium (one computer) or on distributed physical media (more than one computer connected via e.g. an LAN). The basic components of a communications system addressed in this ETS are:

- Communication Application (CA): the CA is the component of a communications system which provides communications services;

- Local Application (LA): the LA is the component of a communications system which uses the communications services offered by the CA.

Different products as a realization of the CA may offer their services in many different ways. This PCI was specified to abstract from such individual properties, to unify the access and performance of the services offered by the CA, and to make LA and CA independent from each other.

The following figure illustrates in a simplified way the main components of a communications system addressed in this ETS.

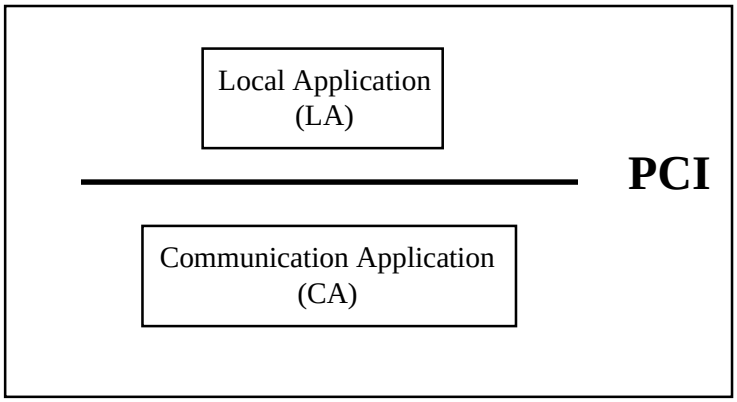


Figure 1: Simplified general model

The relationship between LA and CA as presented in figure 1 is called one-to-one relationship (1:1 relationship). This simple relationship between LA and CA does not cover the need of communications systems as they are commonly used and widely spread today. Different LAs may want to use the services of a CA at the same time. Furthermore, an LA may want to access the services of different CAs over the same PCI simultaneously.

Therefore, the LA-CA relationships which shall be supported at the PCI of this ETS are:

- one-to-one relationship (1:1 relationship): one LA uses the offered services of one CA exclusively;

- many-to-one relationship (m:1 relationship): more than one LA use the offered services of one CA at the same time;

- one-to-many relationship (1:n relationship): one LA uses the services of different CAs at the same time;

- many-to-many relationship (m:n relationship): more than one LA use the services of different CAs at the same time.

The many-to-many relationship is the superset of the possible LA-CA relationships in a communications system which shall be supported by implementations claiming to be conformant to this ETS. The following picture shows an example of the many-to-many relationship.

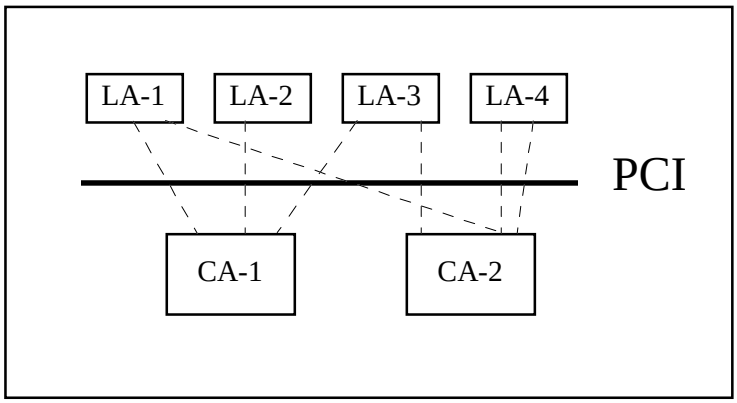


Figure 2: m:n relationships

The communications system in this example consists of four LAs (LA-1, LA-2, LA-3, LA-4) and two CAs (CA-1, CA-2) communicating with each other through the same PCI.

LA-1 and LA-3 communicate with CA-1 and CA-2. That means that both LAs have established one local communications channel with CA-1 and one local communications channel with CA-2 to access the services offered by CA-1 and CA-2. LA-1 and LA-3 shall be able to handle these local communication channels and the flow of information exchanged. LA-2 has established one communications channel with CA-1, whereas LA-4 has established two communications channels with CA-2 to access the services offered by CA-2.

CA-1 shall be able to handle the local communications channels with LA-1, LA-2 and LA-3 and to offer its services to the three LAs through the PCI at the same time. In addition, CA-2 shall be able to handle the local communications channels with LA-1, LA-3 and LA-4, and to offer its services to the three LAs through the PCI at the same time.

A communications system designed according to this ETS provides benefits for both LAs and CAs:

- interoperability:
  components which are developed by different manufacturers and which are conformant with this ETS work well together if they are used in the same communications system;

- compatibility:
  components which offer the same service(s) and which are developed in conformance with this ETS are compatible in their behaviour; they can be exchanged with each other without affecting the other components of the system;

- extensibility:
  a system built according to this ETS is easily extendible:

  - new CA(s) may be incorporated in an existing communications system which offer the same or different communications service(s) and/or access to the same or different communications network(s) through the PCI;

  - an existing system may be extended by new applications (i.e. new LAs) using the service(s) offered at the PCI;

- ease of use and understandability:
  from the viewpoint of an LA, the PCI of this ETS simplifies the access to the communications service(s) and network(s); this means that no exhaustive knowledge of the corresponding communications protocols is necessary.

5.2 Field of application

The PCI of this ETS is an application layer oriented interface, designed for interactive communications services.

The interactive services addressed in this ETS are the file transfer services SimpleFTAM and Easyfile. Therefore, a CA which claims to be conformant with this ETS shall provide:

- access to at least one of the file transfer services specified in this ETS;
- access to physical ISDN;
- the many-to-many relationship between the LAs and CAs.

Other interactive services (e.g. voice communications services) may be specified following the philosophy, the general model, the general rules and concepts of this ETS. This means in particular that all those clauses (i.e. clause 9 and clause 11) which deal with file transfer services exclusively shall be adapted or extended with the requirements of the communications protocols of such interactive services.

5.3 Exchange of information

The information exchanged at the PCI is structured information composed of a set of parameters. These parameters are either service independent or service dependent.

Service independent parameters are used locally at the PCI to manage the flow of information between LA(s) and CA(s). They have no influence on the services requested at the PCI and the communication with a remote system. These parameters shall be used for all services offered at the PCI. They are called general PCI parameters. The general PCI parameters are specified in clause 7 of this ETS.

Service dependent parameters are those parameters which carry service specific information. The collection of those parameters necessary to request one service at the PCI are called Interface Data Units (IDU). The parameters of the IDUs are divided into:

- general IDU parameters are used to handle and control the IDUs exchanged between LA(s) and CA(s) at the PCI; consequently these parameters shall be used for each service requested with an IDU; the general IDU parameters are specified in subclause 9.1;

- service specific IDU parameters are used to interchange data necessary to perform the requested interactive communications service; consequently these parameters are dependent from the services which are supported by a CA.

The general structure of the information exchanged at the PCI of this ETS are presented in the figure 3.

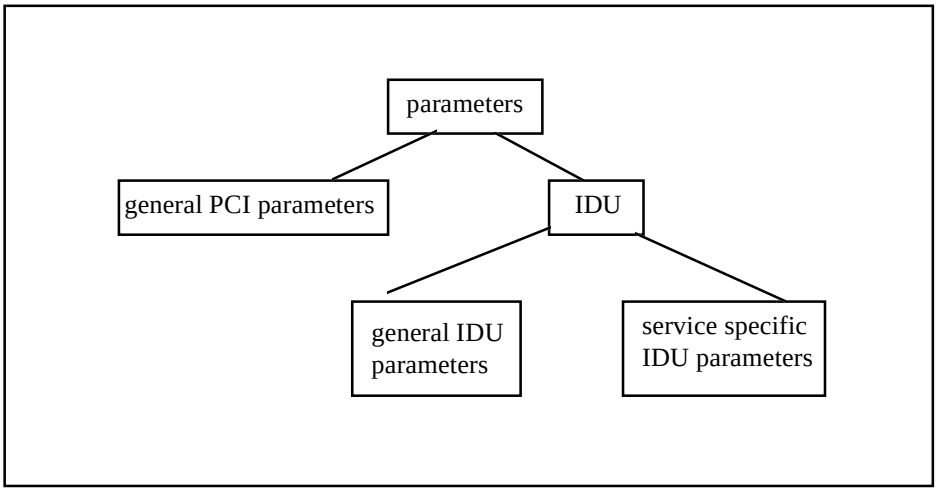


Figure 3: General structure of the exchanged information

The service specific IDU parameters for SimpleFTAM are called SimpleFTAM specific IDU parameters. They shall be used as specified in subclause 9.2. The service specific IDU parameters for Easyfile are called Easyfile specific IDU parameters and shall be used as specified in subclause 9.2.

5.4 PCI services

The IDUs which are specified in this ETS shall be used to convey information necessary to access and to perform file transfer services.

The network connection to a remote system shall be established before any file transfer service is requested at the PCI. After successful establishment of this network connection the file transfer connection shall be established. File transfer operations, such as "send a file" and "receive a file", shall be performed only if also the file transfer connection has been successfully established.

Additionally to these services, local PCI services are specified in this ETS. The local PCI services shall be used to access and perform operations associated with the local system, such as "list local filestore information" which does not require any network connection or file transfer connection.

Each service offered at this PCI belongs to one of the following groups:

- network connection services: these services shall be used to establish and release a network connection, i.e. the ISDN connection to a remote system; network connection services are independent from the file transfer protocol(s) supported by a CA;

- file transfer connection services: these services shall be used to establish, release and abort a file transfer connection on a previous by established network connection; the file transfer connection services shall be requested either by an LA or by a remote system; the file transfer services are related to the file transfer protocol(s) supported by a CA;

- file transfer services: these services shall be used to perform file operations on the files of a filestore on a previous by established file transfer connection; the file transfer services shall be requested either by an LA to perform file operations on the filestore of a remote system, or by a remote system to perform file operations on the local filestore; the file transfer services are related to the file transfer protocol(s) supported by a CA;

- local PCI services: these services shall be used to get information about the local system; the local operations shall be requested by an LA and shall be processed by a CA; they shall not have any influence on the communication with a remote system; consequently, the local PCI services are independent from the file transfer protocol(s) supported by a CA.

5.5 Service profiles

The features and capabilities of the file transfer services specified in this ETS are classified to be mandatory, optional or conditional. To allow the implementation of CAs with different level of complexity and to improve interoperability between LAs and CAs from different manufacturers the concept of "profiles" is introduced.

A profile defines a functional subset of the services offered at the PCI. Two profiles are defined in this ETS:

- a basic profile;
- an enhanced profile.

The enhanced profile is a superset of the basic profile. Therefore, the support of the basic profile is mandatory for all CAs which claim to be conformant to this ETS.

The SimpleFTAM profiles are specified in subclause 9.2.7, the Easyfile profiles are specified in subclause 9.3.7 of this ETS.

6 Functional behaviour

6.1 PCI mechanism

The PCI provides, to transfer data, an exchange mechanism between LAs and CAs which consist of a set of functions. This mechanism allows the exchange of data between an LA and a CA.

Specific functions are defined to allow the support of interactive exchange, giving the possibility to a CA to wake up an LA and to pass data related to that event to the LA:

- the monitoring mechanism allowing a CA to wake up an LA on specific monitoring events;
- the alarm mechanism allowing a CA to wake up an LA on specific alarm events.

These functions allow, if supported by the CA, an LA to ask for notification when an event occurs.

Other functions are provided to assist the routeing of incoming calls:

- an LA selection mechanism giving a way to select LA on incoming calls;
- a redirection mechanism to dispatch an incoming call from one LA to another LA.

6.1.1 Monitoring

An optional monitoring mechanism provided to the LA a feedback on the communication. This allow an LA to get information on the network connection, protocol status or on the progression of file transfer operations.

The monitoring events are:

- network status information which provide to the LA a feedback on network related events (e.g. call in progress, charging information);

- file transfer status information which provides to the LA a feedback on the status of the protocol machine (e.g. establishing protocol, sending a file, receiving a file);

- file transfer information which provide to the LA a feedback on the progression of file transfer operations (e.g. time, file size, size transferred, etc.).

Clause 7 related to the exchange method give a full description of these events.

6.1.2 Alarm handling

A mandatory alarm mechanism allow a CA to wake up an LA on specific alarm events. This mechanism provide to the LA the indications on the file transfer events or on local PCI events.

These alarm events are:

- file transfer confirmation and indications events;
- events relative to the local CA and to the exchange mechanism;
- system alarms relative to the local operating system.

Clause 7 related to the exchange method give a full description of these events.

6.1.3 LA selection mechanism

A mandatory mechanism is defined to provide for the CA a way to select an LA to which an incoming call shall be passed. This mechanism give also priorities to the LA on incoming calls.

To use the service an LA shall announce to the CA for registration as follow:

- <No incoming calls>, the LA will not get any incoming calls;

- <Admin>, the LA will get all incoming calls including specific;

- <Specific name>, the LA will get only those incoming calls which are addressed to that LA with the <Specific name>, To convey this information a user to user identification parameter (uuid) is defined. The mapping of this parameter to lower layers selection mechanisms (i.e. Network Service Access Point (NSAP), etc.) shall be provided by the CA;

- <Default>, the LA will be wake for all incoming calls except specific or <Admin> (if they are any); if there is no <Admin> or specific LA logged, the LA get all incoming calls.

The CA shall use the appropriate priorities for each registered LA. The CA select the LA for which the incoming call will be passed using the following priority scheme: the higher priority is dedicated to LA registered as <Admin>, the next priority is given to LA registered on <Specific name>, the lowest priority is to LA registered as <Default>. If no LA match these priorities, the incoming call is rejected.

6.1.4 Redirection of incoming calls

On systems where the CA may be used by several LAs simultaneously, it is necessary to assist the routeing of incoming calls to the intended recipient. This process is called the Redirection of Incoming Calls (RIC) and is optional in this ETS.

When a CA receives an incoming call from the network, it shall be assigned to a single recipient LA.

If the CA supports the RIC then the recipient LA may dispatch a received incoming call to the appropriate LA by means of an administration request to the CA. The CA assigned the network connection to the LA to which the incoming call is dispatched. The incoming call is received by the selected LA.

As this CA feature is optional, the support of RIC shall be declared in the CA capability description.

6.2 Description of PCI identifier

To cope with the various interchanges of information through the interface, it is required to identify unambiguously the communicating entities and the communication events. The following identifiers shall be supported:

- identification of the LA-CA local communication channel (connection_id);
- identification of the CA that accepts the local communication channel (ca_id);
- identification of a successful established network connection (netcon_id);
- identification of the service requested with the IDU (service_id);
- identification of a successful file transfer connection (ft_con_id).

Following subclauses detail these identifiers.

6.2.1 Connection identifier

In order to identify an LA-CA communications path, the connection_id is defined. The connection_id shall be computed by the CA on invocation of the IA_login request (defined in clause 7). The LA shall use this identifier throughout the connection with the CA until the LA logs out.

6.2.2 CA identifier

The ca_id identifies a CA. This identifier allows the LA to have simultaneous interactions with multiple CAs. Therefore the ca_id is a parameter used in each exchange mechanism function.

6.2.3 Network connection identifier

The netcon_id identifies an established network connection between an LA and a remote communication system. This identifier allows the LA to have simultaneous interactions with multiple network connections. Therefore the netcon_id is a parameter to be used in each file transfer operation. This identifier shall be handled by the CA upon the confirmation of a network connection. Meanwhile, the LA shall use a temporary value in order to be able to release a network connection prior to the confirmation of the network connection.

6.2.4 File transfer service identifier

The service_id identifies a file transfer operation requested with an IDU.

6.2.5 File transfer connection identifier

The ft_con_id identifies an established file transfer connection. The ft_con_id is a parameter to be used in each file transfer operation. This identifier shall be set up, in this context by the CA upon the confirmation of a file transfer connection. Meanwhile, the LA shall use a temporary value in order to be able to release a file transfer connection prior to the confirmation of the file transfer connection.

6.3 Sequences using the PCI

6.3.1 Login to a CA

Selection of a CA is performed by specifying a parameter which gives precision to the CA on the related functions used by the LA.

The CA controls the access of an LA by checking the name of the LA and the authentication given by the LA. However, the extension to which control of the access rights is performed is up to the CA implementation.

The CA shall check the parameters and if they match, it then generates a connection_id the LA shall use subsequently in all other function calls.

6.3.2 Sending and receiving IDUs

Firstly, an LA needs to login (function IA_Login) to the requested CA. No IDU exchange can take place before the login procedure is completed.

When an LA wishes to send an IDU to a CA, it shall carry out the following steps:

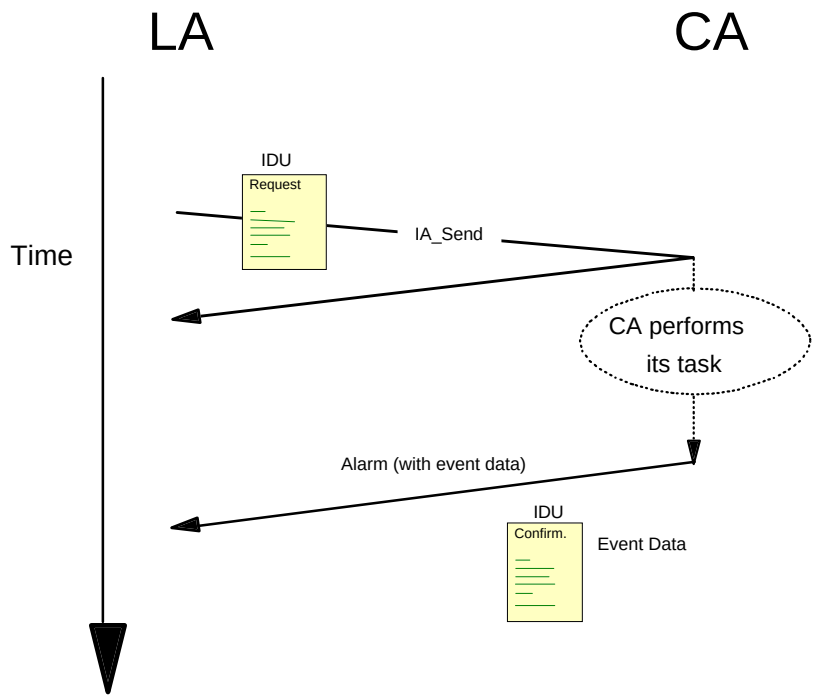
1) Build the IDU (by any appropriate means).

2) Invoke the IA_Send. As a result the IDU is conveyed from the LA to the CA.

3) When an LA no longer needs to maintain the dialogue with a CA, it shall log out (function IA_Logout) from this CA. The CA willthus know the LA-CA communications path is broken, and will not use the alarm mechanism any longer.

The confirmation IDU will be received through the alarm from the CA. The LA shall handle the buffer (event associated IDU) returned with this alarm.

Figure 4 illustrates the behaviour described above.



NOTE: It is assumed that the LA has already logged into the CA.

Figure 4: Sample sequence of the exchange mechanism to send IDU

When an LA wishes to receive IDUs, it shall carry out the following steps:

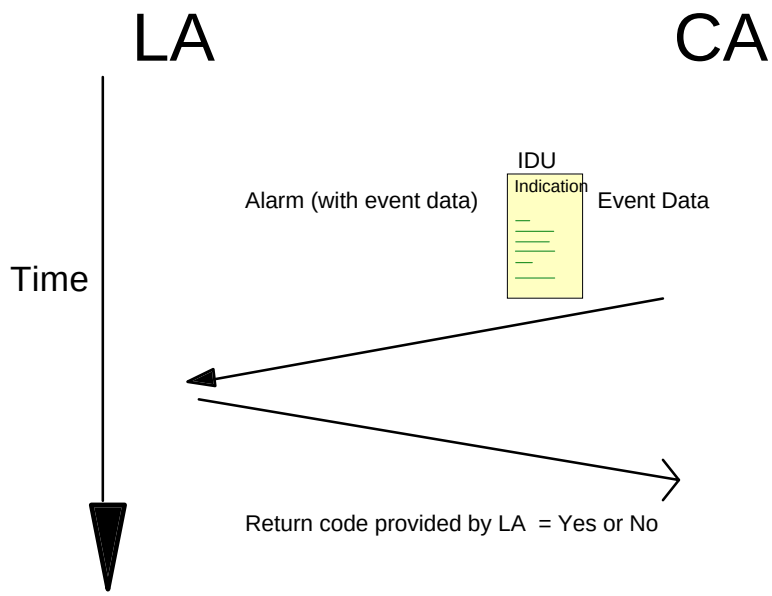
1) Invoke the IA_Login function specifying the events type needed.

2) To announce incoming events and exceptional or erroneous situations, the LA will be wake up through the alarm function from the CA, the LA shall handle the buffer (event associated data) returned with the alarm.

3) If a response is needed, the LA shall pass an appropriate return code of the alarm function and may also provide a response IDU. This return code allows the LA to accept, reject or to provide a response IDU to the incoming IDU.

When an LA no longer needs to maintain dialogue with a CA, it shall log out (function IA_Logout) from this CA. The CA will thus know the LA-CA communication path is broken and will not use the alarm mechanism any longer.

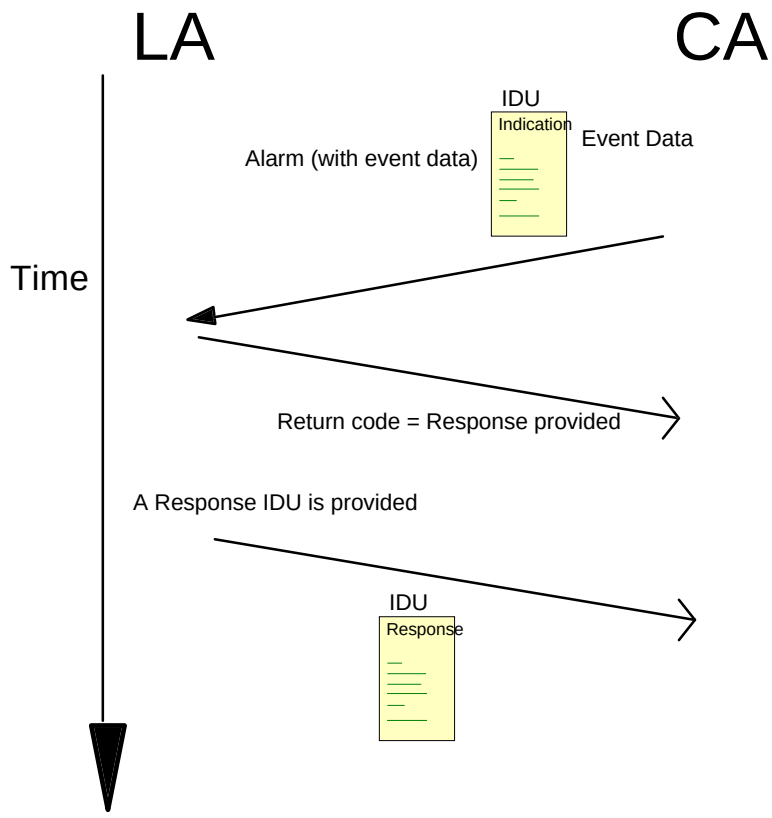
Figure 5 illustrates the exchange when no response IDU is provided by the LA.



NOTE: It is assumed that the LA has already logged into the CA.

Figure 5: Sample sequence of the exchange mechanism to receive IDU

Figure 6 illustrates the exchange when a response IDU is provided by the LA.

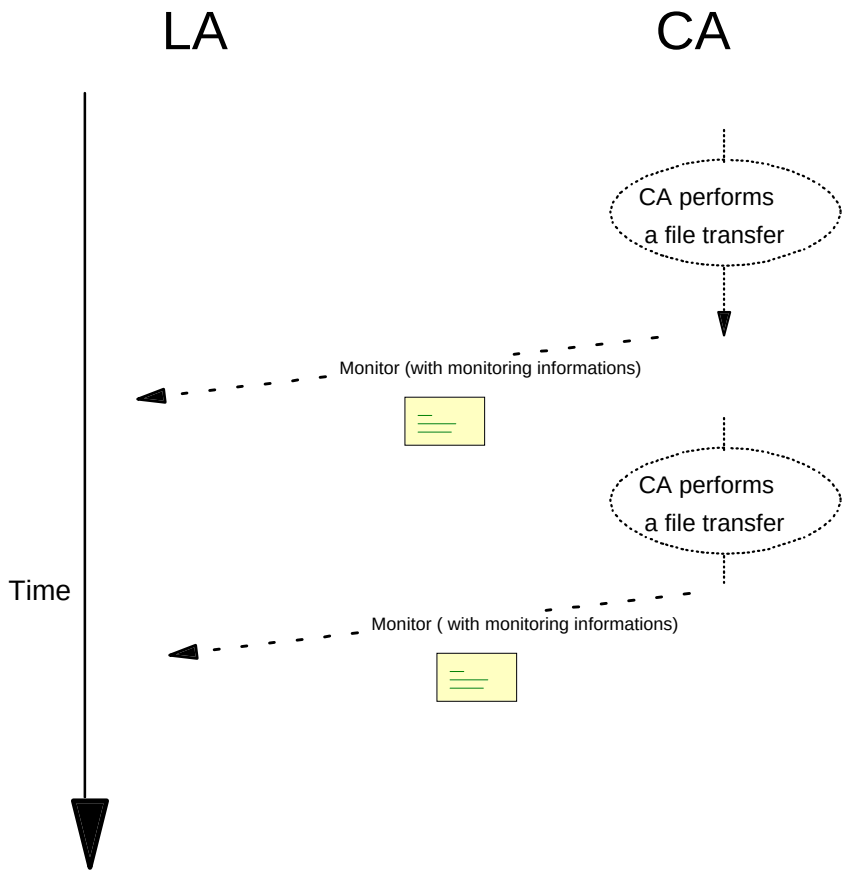


NOTE: It is assumed that the LA has already logged into the CA.

Figure 6: Sample sequence of the exchange mechanism to receive IDU and send a response IDU

6.3.3 Monitoring

When an LA wishes to monitor a communication, it shall declare a function address in the IA_login which will be used by the CA pass monitoring information to the CA.



NOTE: It is assumed that the LA has already logged into the CA with a monitoring function address.

Figure 7: Sample sequence of the exchange mechanism to monitor a file transfer progression

6.3.4 Receiving a file

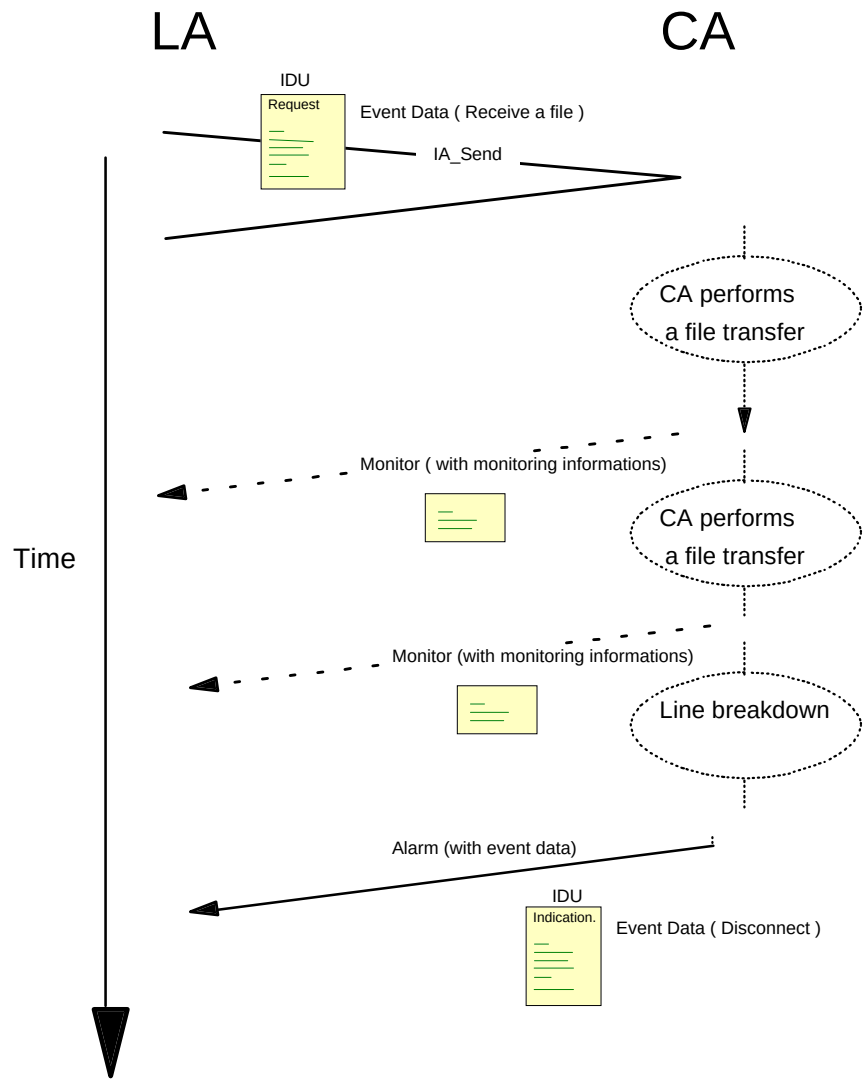
When an LA wishes to receive a file from the remote filesystem, it shall carry out the following steps assuming that the LA is already logged on to the CA and has established a file transfer connection:

1) Build the IDU for receiving a file (by any appropriate means).

2) Invoke the IA_Send. As a result the IDU shall bwe conveyed from the LA to the CA.

3) If the LA is logged on with a monitoring function address, the LA will be woken up through the monitoring function and the LA will follow the file transfer progression with the monitoring events.

4) If the file transfer is ended normally or through a line breakdown, the LA shall be woken up through an alarm function from the CA; the LA shall handle the buffer returned with the alarm.



NOTE: It is assumed that the LA has already logged into the CA with a monitoring function address.

Figure 8: Sample sequence of the exchange mechanism to receive a file

6.3.5 Logout

The logout process allows the decoupling of the LA and the CA: it shall be invoked by the LA when the LA wishes to release the communication channel with the CA. As a result of this logout process, the connection_id shall be discarded by the CA and becomes invalid. No further communication or information exchange on behalf of that connection_id shall be possible for the LA.

6.4 Transitions at the PCI

Table 1: General states of the PCI

State name	Means
"Idle"	No LA-CA local communication channel is used.
"Logged"	The LA is logged to the CA. A connection_id is assigned.
"Network connected"	The network connection has been established with a remote entity. A netcon_id is assigned.
"File Transfer connected"	A file transfer connection has been established with a remote entity. A ft_con_id is assigned.

Transitions from one state to another are governed by actions internal to the CA, coming from the communications network, or issued by the LA.

The following describes the state transitions in relation to the transmission and reception events occurring at the CA.

6.4.1 Idle state

From the "Idle" state an LA shall log on to the CA to establish an LA-CA local communication channel to exchange data. No other operations shall be allowed.

6.4.2 Logged state

After successful establishment of the local communication channel the state changes from "Idle" to "Logged". From the "Logged" state an LA may establish a network connection or ask for PCI local services. No file transfer operations shall be allowed at this state.

6.4.3 Network connected state

After successful establishment of the network connection the state changes from "Logged" to "Network connected". From the "Network connected" state an LA may establish a file transfer connection or ask for PCI local services or release the network connection. No file transfer operations shall be allowed at this state.

6.4.4 File Transfer connected state

After successful establishment of the file transfer connection the state changes from "Network connected" to "File transfer connected". From the "File transfer connected" state an LA may use the file operations services or ask for PCI local services or release the file transfer connection.

6.5 Error Handling

Errors present at the PCI are classified through different types:

- PCI errors;
- protocol errors;
- network errors.

6.5.1 PCI errors

PCI errors cover local errors which shall lead to a rejection of an IDU or to a system crash. The following errors are covered:

- IDU syntactical errors;
- function error codes;
- system errors;
- manufacturer errors.

All syntactical errors (i.e. unrecognized syntax elements, missing syntax elements classified as mandatory, conflicting syntax elements, multiple occurrences of a single syntax element, parameters out of range within an IDU) shall lead to the rejection of the IDU.

Function errors (i.e. incorrect references, etc.), system errors (i.e. CA unavailable, CA busy, etc.) and manufacturer error shall lead to the rejection of the exchange method function and of the IDU.

The PCI errors shall be conveyed through the status parameter of the functions (IA_Login, IA_Logout, IA_Send) or through the alarm mechanism. These errors are described in annex A.

6.5.2 Protocol error codes

Protocol errors are referenced directly with the corresponding cause of the file transfer protocol. They shall be conveyed within appropriate parameters of the IDUs. These errors are described in clause 9.

6.5.3 Network error codes

Network errors correspond to the equivalent of the network protocol. They shall be conveyed within appropriate parameters of the IDUs. These errors are described in clause 9.

7 Exchange mechanism

This clause describes how IDUs (Interface Data Units) are transferred between LAs and CAs.

To transfer the IDUs this ETS defines an abstract exchange mechanism between LAs and CAs which consists of a set of functions.

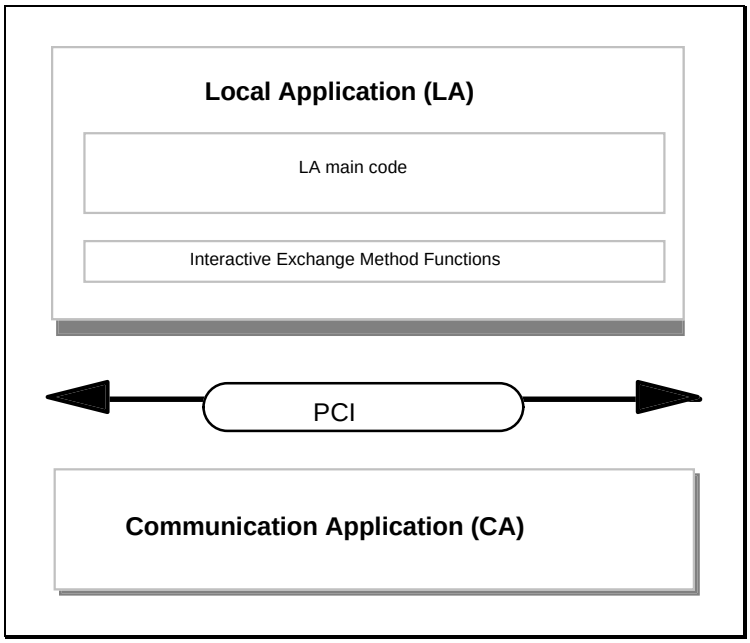


Figure 9: Usage of the interactive exchange mechanism functions

7.1 Overview of interactive exchange mechanism functions

The functions used are depicted in table 2:

Table 2: Summary of exchange mechanism functions

Function	Type	Purpose
IA_Login	m	LA opens a local communication channel between LA and CA.
IA_Logout	m	LA closes the local communication channel opened with IA_Login ().
IA_Send	m	LA send data (IDU) to a CA.
IA_Alarm	m	CA wake up the LA and pass data (IDUs) to the LA.
IA_Monitor	o	CA wake up the LA and pass monitoring data to the LA.

7.2 General PCI parameters

Table 3

Parameter	Comment
login_name	Name of the LA which wants to establish a local communication channel to the CA.
password	Authentication of the LA which wants to establish a local communication channel.
selector	A user-provided string specifying the requested service, i.e. the requested file transfer service.
ca_id	Identifier of the CA that accepts the local communication channel.
connection_id	Identifier of the LA-CA communication channel. Returned by the CA if the IA_Login function succeeds. Otherwise (e.g. identification failed) the CA sets the value to zero and a appropriate return code is given by the status parameter (see below).
status	Return code; a value of zero indicates successful operation.
mode	Type of data channel required.
uuid	User to user identification to identify an application. This enables the CA to select an LA which will get the incoming call (see 6.1.3 LA selection mechanism).
monitor_handler	Entry point of the CA-LA data channel used to monitor the interactive data transfer. This address shall be used by the CA for the IA_Monitor function (see 6.1 PCI mechanism).
monitor_data	Points to a buffer containing the data to be monitored.
monitor_length	Length of monitor data.
alarm_handler	Entry point of the LA alarm handler. This entry point shall be called by the CA when one of the registered events occurs. This parameter as well as the alarm handler itself is platform-dependent and LA dependent. This address shall be used by the CA for the IA_Alarm function (see subclause 6.1).
alarm_type	Identifies the interactive function to which alarm is relevant (system error or file transfer data).
alarm_data	Points to a buffer containing data associated with the alarm.
alarm_length	Length of alarm data.
IDU	The actual data to be transmitted to the CA.

7.2.1 Alarm support

The CA shall support the alarm mechanism. This mechanism allows a CA to wake up an LA on specific alarm events. The return code of the call-back routine may be used as a response for the CA.

7.2.2 Monitor support

If the CA supports the optional monitor feature this function allows a CA to wake up an LA on specific monitoring events.

7.3 Interactive exchange mechanism functions

Since the implementation of the functions is platform dependent and programming language dependent, the functions are described in a generic way in the following subclauses.

7.3.1 Function IA_Login

The function IA_Login shall be supported by the CA. It shall be invoked by the LA before any LA-CA interchange of IDUs.

7.3.1.1 Purpose

The function IA_Login returns to the LA a **connection_id** that shall be used throughout the LA-CA interaction until the LA logs out.

Selection of a CA is performed by specifying a string containing keywords (e.g. "Easyfile", "SimpleFTAM") separated by spaces, in the **selector** parameter. This parameter gives precision to the CA on the related functions used by the LA.

The **mode** parameter is used to specify the format of data carried out by the data channel. For file transfer, the mode parameter has a mandatory value set to "File Transfer". This parameter may be extendable to other future services.

The **uuid** parameter is used to specify the application identity. This parameter shall be used by the incoming call selection mechanism to select the LA. This allow the LA to select the incoming calls it want to react on. This parameter shall take one of the following values: no incoming calls, only specific incoming calls, admin, default (see clause 6).

The IA_Login function allow the CA to control the access of an LA. This can be achieved by checking the **login_name** and the **password** given by the LA. However, the extent to which control of the access rights is performed is up to the CA implementation.

The parameter **alarm_handler** function address is used by the LA to announce to the connected CA the IA_Alarm function the LA supports.

The CA shall record the location of the **alarm_handler** function assigned by the LA. The CA shall be able to handle as many **alarm_handler** function instances as there are different logged-in LAs.

The **monitor_handler** function address allows the LA to register to the connected CA the IA_Monitor function the LA supports.

7.3.1.2 Behaviour

The CA shall check the parameters of the IA_Login call. If they are valid, it shall then generate the **connection_id** the LA is to use subsequently in function calls. In addition, if the **selector** parameter or the **mode** parameter specifies criteria, the function shall return the **ca_id** of a CA that match the criteria. If the **connection_id** returned is set to zero, it means the CA failed to connect with the LA.

7.3.1.3 Parameters

Table 4: Parameters of the IA_Login function

Parameter	Comment	Type	Direction
login_name	Name of the LA which wants to establish a local communication channel to the CA.	m	Input
password	Authentication of the LA which wants to establish a local communication channel.	o	Input
selector	A user-provided string specifying the requested service selection.	m	Input
ca_id	Identifier of the CA that accepts the local communication channel.	m/c	Output
connection_id	Identifier of the LA-CA data channel. Returned by the CA if the IA_Login function succeeds. Otherwise (e.g. identification failed) the CA sets the value to zero and a appropriate return code is given by the status parameter (see below).	m/c	Output
status	Return code; a value of zero indicates successful operation.	m	Output
mode	Type of data channel required.	m	Input
uuid	User to user identification to identify an application.	c	Input
monitor_handler	Information addressing the entry point of the CA-LA data channel used to monitor the interactive data transfers.	o	Input
alarm_handler	Information addressing the entry point of the LA alarm handler. This entry point is called by the CA when an error or a file transfer operation occurs. This parameter as well as the alarm handler itself is platform-dependent and LA dependent.	m	Input

7.3.2 Function IA_Logout

The function IA_Logout shall be supported by the CA. It shall be invoked by the LA on completion of any LA-CA interchange of IDUs.

7.3.2.1 Purpose

The function IA_Logout returns to the LA a **status** that states whether the LA-CA interaction has been terminated.

7.3.2.2 Behaviour

Before completing the LA-CA dialogue, the CA may (but is not obliged to) process all pending IDUs that were issued by that LA.

7.3.2.3 Parameters

Table 5: Parameters of the IA_Logout function

Parameter	Comment	Type	Direction
connection_id	Identifier (handle) of the LA-CA data channel returned by the IA_Login function.	m	Input
ca_id	Identifier of the CA that accepts the local communication channel.	m	Input
status	Return code; a value of zero indicates successful operation.	m	Output

7.3.3 Function IA_Send

The function IA_Send shall be supported by the CA. The function IA_Send is a function allowing the LA to send data (IDU) to the CA. The LA provides the CA with a buffer containing the IDUs.

7.3.3.1 Purpose

The function IA_Send allows access to the file transfer services and the local PCI services passing IDU from the LA to the CA.

The **status** parameter specifies the error code generated by the CA if the function failed. If the function succeeded, then the error code is set to zero.

The **IDU** parameter is a structure address provided by the LA. The actual content, the IDU, depends on the type of operation to be carried out.

7.3.3.2 Behaviour

The CA shall return from this function after reading and examining the IDUr. When the function has been completed the IDU becomes unavailable to the CA.

7.3.3.3 Parameters

Table 6: Parameters of the IA_Send function

Parameter	Comment	Type	Direction
connection_id	Identifier (handle) of the LA-CA data channel returned by the IA_Login function.	m	Input
ca_id	Identifier of the CA.	m	Input
status	Return code; a value of zero indicates successful operation.	m	Output
IDU	The actual data to be transmitted to the CA, i.e. the appropriate IDU.	c	Input

7.3.4 Function IA_Monitor

The function IA_Monitor shall be supported by the CA. The function IA_Monitor is a function allowing the CA to send monitoring data to the LA. The CA shall provide the LA with a buffer containing this information.

7.3.4.1 Purpose

This function shall be used to convey data to the LA which shall be used as monitoring information by the LA.

This function informs the LA of the number of bytes transmitted, the progression of the file transfer, the foreseen file transfer duration, the progression of the network connection, etc.

The **function** parameter is set to the interactive information to which IA_Monitor is relevant (e.g. Network status, File transfer status, File transfer info).

The **session_Id** parameter is used to identify the network connection between an LA and a remote system relevant for the monitoring information. The parameter take the value of the netcon-id (see clause 9).

The **monitor_data** parameter is a structure address provided by the CA. The **monitor_data** describes the information needed to provide to the LA feedback on the communication session. This parameter shall contain for file transfer one of the following data structures:

- network_status;
- ft_status;
- ft_infos.

The **status** parameter specifies the error code generated by the LA if the function failed. If the function succeeded, then the error code is set to zero.

7.3.4.2 Behaviour

The IA_Monitor shall be implemented by the LA. The CA shall call the function when it needs to pass monitoring information to the LA. It is up to the CA to decide when information will be passed to the LA.

The LA shall return immediately after reading the buffer. When the function has been completed the buffer becomes unavailable to the LA.

7.3.4.3 Parameters

Table 7: Parameters of the IA_Monitor function

Parameter	Comment	Type	Direction
ca_id	Identifier of the CA that accepts the local communication channel.	m	Output
connection_id	Identifier of the LA-CA data channel. Returned by the CA if the IA_Login function succeeds. Otherwise (e.g. identification failed) the CA sets the value to zero and a appropriate return code is given by the status parameter (see below).	m	Output
function	Identifies the monitor function to which IA_Monitor is relevant.	m	Output
netcon_id	Identifies the network connection between the LA and a remote system to which IA_Monitor is relevant.	m	Output
monitor_data	Points to a buffer containing the data to be monitored (network status, file transfer status, file transfer information).	m	Output
status	Return code; a value of zero indicates successful LA operation.	m	Input

The **network_status** provide to the LA a feedback on network related events (Call in progress, etc.). The fields provided shall be:

- Network information parameter for the current network information available;

- Network status parameter for the current status of the connection (Established, Call in progress, Idle, Ringing).

Table 8: Parameters of the network status

Parameter	Comment
network_information	Current network information string provided by the CA.
network_status	Identifies the status of the connection.

The **ft_status** provides to the LA feedback on the status of the protocol machine (establishing protocol, master role, sending file, receiving file, deleting file, renaming file, etc.). The fields provided shall be:

- Protocol information for the current protocol information provided by the CA;
- Current operation field for the current file transfer operation;
- Current parameter field associated with the Currentoperation;
- Elapsed time field for the elapsed time of connection.

Table 9: Parameters of the ft_status

Parameter	Comment
protocol_information	Current protocol information string. This parameter provide additional information to the LA.
currentoperation	Current file transfer operation. This parameter is identical to the service_id (see subclause 6.2).
currentparameter	Current parameter string associated with the current operation. This parameter is provided by the CA and gives precision on the current operation (e.g. filename associated with a sending operation).
elapsedtime	Elapsed time for the network connection.

The **ft_infos** provides to the LA a feedback on the progression of file transfer (time, file size, size transferred,...) either in receiving a file or sending a file.

The possible parameters presented to the LA are as shown in table 10:

Table 10: Parameters of the ft_info

Parameter	Comment
bytespersecond	Approximate rate of transfer.
currentindex	Index of the current file (1 to n).
currentsize	Total bytes in current file to transfer.
currentpercentdone	Percent of the transfer completed.
currenttransferred	Number of bytes transferred of the current file.
numfiles	Number of files to transferred.
totalpercentdone	Percentage of the entire transfer completed.
totalsize	Total bytes in all files to transfer.
transferseconds	Elapsed time of the file transfer.
sourcefilename	Current source filename.
destfilename	Current destination filename.
filedescription	Name describing the group of transferred files.
directionoftransfer	Indication of receive or send operation.

7.3.5 Function IA_Alarm

The function IA_Alarm shall be supported by the CA. The function IA_Alarm is a function allowing the CA to invoke the LA and to send data to the LA. The CA shall provide the LA with a buffer containing the information.

7.3.5.1 Purpose

The IA_Alarm function defines a mechanism that allows a CA to alert the LA if an event occurs with a possible interaction from the LA.

The **alarm_type** received by the LA shall be defined as below:

- ERROR_ALARM for an error event relative to the local CA and to the exchange mechanism or the local operating system; or

- FT_DATA for an event relative to file transfer data.

The **alarm_data** is a structure to pass the data associated with the file transfer operation. This structure is used to convey either IDUs for indication and confirmation, or error events.

The related error event data refers to CA error, corrupted IDUs, missing mandatory parameters, authorization failure, system error, product specific error.

The **status** parameter specifies the return code generated by the LA. The parameter shall take one of the following values: *yes*, *no* or *response provided*.

7.3.5.2 Behaviour

The IA_Alarm shall be implemented by the LA. The CA shall call that function when it needs to trigger an alarm. The events the CA may trigger are specified by the LA by means of the alarm_type.

The LA shall return immediately after reading the buffer. When the function has been completed, the buffer shall become unavailable to the LA.

7.3.5.3 Parameters

Table 11: IDU parameters passed to the IA_Alarm

Parameter	Comment	Type	Direction
ca_id	Identifier of the CA that accepts the local communication channel.	m	Output
connection_id	Identifier of the LA-CA data channel. Returned by the CA if the IA_Login function succeeds. Otherwise (e.g. identification failed) the CA sets the value to zero and a appropriate return code is given by the **status** parameter (see below).	m	Output
alarm_type	Identifies the type of event (file transfer data or error).	m	Output
alarm_data	Points to the data associated with the alarm.	c	Output
alarm_length	Length of alarm data.	c	Output
status	Return code of the alarm handler; the possible values are yes, no or response provided.	m	Input

Table 12: Parameters of the alarm_data for error alarm events

Parameter	Comment
error_type	Type of the error event.
error_description	String or structure describing the error (error specific).This field is CA dependent.

Table 13: Parameters of the alarm_data for file transfer events

Parameter	Comment
IDU	The actual data to be transmitted from the CA to the LA.

8 Interface Data Units (IDUs)

The information exchanged between LA and CA to perform a service offered at the PCI is composed by general PCI parameters and Interface Data Units (IDUs) as defined in clause 5. The general PCI parameters and IDUs shall be conveyed at the PCI with the functions as defined in clause 7.

To distinguish between IDUs passed from a CA to an LA with the function IA_Alarm, and IDUs passed from an LA to a CA with the function IA_Send, four types of IDUs are defined in this ETS:

- request IDUs:
  with an request IDU (also called Req in the following), an LA asks the CA to perform one of the services offered at the PCI;

- confirmation IDUs:
  with an confirmation IDU (also called Conf in the following), a CA passes the result and additional data (if there is any) of a previous Req to an LA;

- indication IDUs:
  with an indication IDU (also called Ind in the following), a CA passes incoming data from a remote system to an LA;

- response IDUs:
  with a response IDU (also called Resp in the following), an LA passes the result and additional data (if there are any) related to a previous Ind to a CA.

Figure 10 illustrates the general use of request IDUs, confirmation IDUs, indication IDUs, and response IDUs exchanged at the PCI.

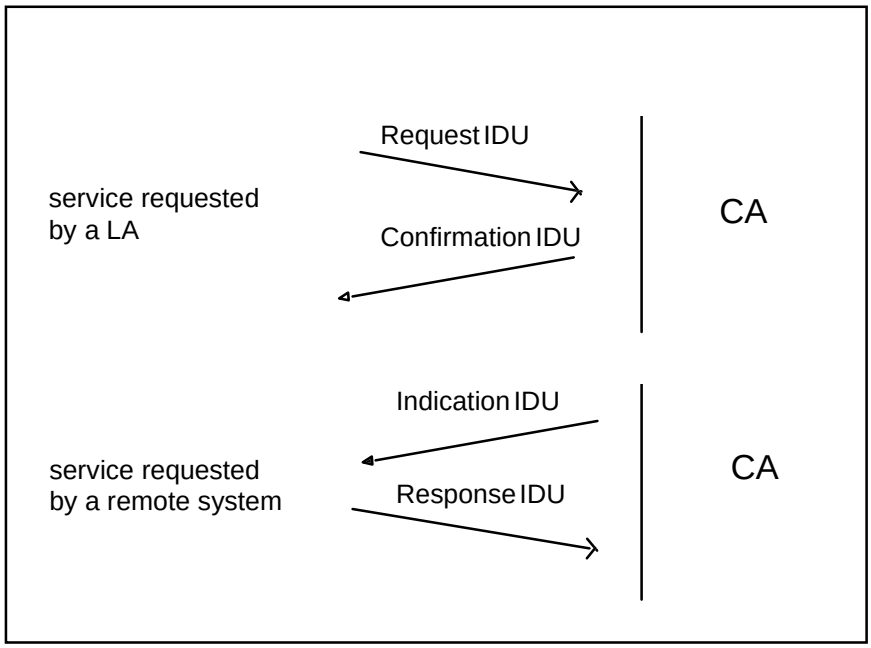


Figure 10: General sequence of IDUs exchanged at the PCI

The sequence of IDUs (Req, Conf, Ind, Resp) necessary to perform one of the services offered at the PCI depends on the service itself.

A confirmed service like the establishment of a network connection to a remote system requires all four IDU types (Req, Conf, Ind, Resp). With an appropriate request IDU, an LA requests the establishment of a network connection to a specific remote system. The LA will receive the result of the request with the corresponding confirmation IDU. If an incoming call from a remote system arrives, the CA passes the appropriate indication IDU to the LA. The LA then has to decide whether the requested network connection shall be accepted or refused by passing the corresponding response IDU to the CA.

A confirmed service like the abrupt termination of a file transfer connection requires only three IDU types (Req, Conf, Ind). With an appropriate request IDU, an LA requests a CA to abort a successfully established file transfer connection to a remote system. The CA shall inform the LA that the abrupt network connection termination procedure is processed with the correspondent confirmation IDU. If a remote system has aborted a file transfer connection, the CA passes the appropriate indication IDU to the LA. No response IDU is necessary: the file transfer connection is already terminated.

A confirmed service like list local filestore information requires only two IDU types (Req, Conf). With an appropriate request IDU, an LA requests a CA to rename a file. The LA will receive the result of the request with the corresponding confirmation IDU.

The behaviour of an LA and the behaviour of a CA related to every single service offered at the PCI are defined in clause 6.

The structure of an IDU is independent from its type (Req, Conf, Ind, Resp) as described in clause 5. The parameters of each IDU and their meaning are specified in clause 9 of this ETS. The data types of the IDU parameters are specified in clause 11 of this ETS.

The services offered at the PCI are grouped into:

- network connection services;
- file transfer connection services;
- file transfer services;
- local PCI services.

The IDUs which are offered at the PCI to access and perform a single service are defined in the following subclauses.

8.1 Network connection services

The network connection services shall be used to establish and release a physical network connection between the local system and a remote system. The physical network addressed in this ETS is ISDN. The following network connection services shall be offered at the PCI:

- *establishment of a network connection;*
- *release of a network connection;*
- *send transparent data.*

The services *establishment of a network connection* and *release of a network connection* are applicable for both file transfer protocols, SimpleFTAM and Easyfile. The service *send transparent data* is applicable applied only for Easyfile.

The use of the network connection services is conditional. If an LA or a remote user wants to perform file transfer operations the network connection needs to be established before any file operation can be processed. If an LA wants to perform only local PCI services, the establishment of a network connection is not necessary.

If there is no longer the need to perform file transfer operations, the established network connection should be terminated to release resources which are bound with the network connection (e.g. ISDN resources, CA resources, memory resources).

8.1.1 Establishment of a network connection

The network connection between the local system and a remote system shall be established either by an LA or by a remote user. The network connection establishment service is applicable for SimpleFTAM and for Easyfile.

The IDUs offered at the PCI for the network connection establishment procedure are:

- ConnectRequest IDU (ConReq);
- ConnectConfirmation IDU (ConConf);
- ConnectIndication IDU (ConInd);
- ConnectResponse IDU (ConResp).

The use of these IDUs is conditional and depends on the actual situation at the PCI (see clause 6).

8.1.2 Release a network connection

The network connection between the local system and a remote system shall be released if no further communication is necessary, i.e. if no further file transfer operations are to be performed. The network connection release service is applicable for SimpleFTAM and for Easyfile.

The IDUs offered at the PCI for the network connection release procedure are:

- DisconnectReq IDU (DiscReq);
- DisconnectConf IDU (DiscConf);
- DisconnectInd IDU (DiscInd).

The use of these IDUs is conditional and depends on the actual situation at the PCI (see clause 6).

8.1.3 Send transparent data

After successful establishment of the network connection, data may be exchanged between the local system and the remote system. This data shall be exchanged before the file transfer connection is established. The *send transparent data* service is applicable only for Easyfile.

The IDUs offered at the PCI for the send transparent data procedure are:

- SendDataRequest IDU (SendDataReq);
- SendDataConfirmation IDU (SendDataConf);
- SendDataIndication IDU (SendDataInd).

The use of these IDUs is conditional and depends on the actual situation at the PCI (see clause 6).

8.2 File transfer connection services

After successful establishment of the network connection, the file transfer connection services shall be used to establish, release and abort a file transfer connection.

The following file transfer services are offered at the PCI:

- *establishment of a file transfer connection;*
- *termination of a file transfer connection;*
- *establishment of a file transfer association;*
- *termination of a file transfer association;*
- *establishment of a file access;*
- *release of a file access;*
- *abort of a file transfer connection.*

Two different procedures to establish and terminate a file transfer connection may be offered at the PCI:

1) The one step procedure:
   The one step procedure is performed by using the service *establishment of a file transfer connection* to establish a file transfer connection and by using the service *termination of a file transfer connection* to terminate the established file transfer connection;

2) The two step procedure:
   the two step procedure is performed by using the services *establishment of a file transfer association* and *establishment of a file access* to establish a file transfer connection. To terminate a file transfer connection established with the two step procedure, the services *release of a file access* and *termination of a file transfer association* shall be used.

The two different procedures shall be used exclusively. That means that an LA shall perform either the one step procedure or the two step procedure to establish and terminate a file transfer connection.

For Easyfile, either procedure may be used whereas for SimpleFTAM, only the one step procedure may be used.

The use of the file transfer connection services is conditional. If an LA or a remote user wants to perform file transfer operations, the file transfer connection shall be established before any file operation can be processed. If an LA wants to perform only local PCI services, the establishment of a file transfer connection is not necessary.

If there is no longer the need to perform file transfer operations, the established file transfer connection should be terminated to release resources which are bound with the file transfer connection (e.g. CA resources, memory resources).

8.2.1 One step procedure

The one step establishment and termination procedure for a file transfer connection shall be performed by using the services:

- *establishment of a file transfer connection;*
- *termination of a file transfer connection.*

8.2.1.1 Establishment of a file transfer connection

If an LA or a remote user wants to perform file operations a file transfer connection shall be established between them. Therefore, no file operation on a filestore (e.g. send a file, receive a file, rename a file) can be performed before the file transfer connection has been established successfully. The file transfer connection establishment service is applicable to SimpleFTAM and for Easyfile.

The IDUs offered at the PCI for the file transfer connection establishment procedure are:

- FileTransferEstablishmentRequest IDU (EstabReq);
- FileTransferEstablishmentConfirmation IDU (EstabConf);
- FileTransferEstablishmentIndication IDU (EstabInd);
- FileTransferEstablishmentResponse IDU (EstabResp).

The use of these IDUs is conditional and depends on the actual situation at the PCI (see clause 6).

8.2.1.2 Termination of a file transfer connection

If no further file operation is to be performed, the file transfer connection should be released. The file transfer connection termination service is applicable to both file transfer protocols, SimpleFTAM and Easyfile.

The IDUs offered at the PCI for the file transfer termination procedure are:

- ReleaseRequest IDU (RelReq);
- ReleaseConfirmation IDU(RelConf);
- ReleaseIndication IDU (RelInd);
- ReleaseResponse IDU (RelResp).

The use of these IDUs is conditional and depends on the actual situation at the PCI (see clause 6).

8.2.2 Two step procedure

The two step establishment and termination procedure for a file transfer connection shall be performed by using the services:

- *establishment of a file transfer association;*
- *termination of a file transfer association;*
- *establishment of a file access;*
- *release of a file access.*

8.2.2.1 Establishment of a file transfer association

After successful establishment of the network connection, an LA or a remote user may perform file operations. No file operation on a filestore (e.g. send a file, receive a file, rename a file) may be performed before the file transfer connection has been established successfully. This means for the two step procedure that the service file transfer association establishment shall be performed after the network connection and prior to the establishment of the file access. The file transfer association establishment service is not applicable for SimpleFTAM.

The IDUs offered at the PCI for the file transfer association establishment procedure are:

- AssociationRequest IDU (AssocReq);
- AssociationConfirmation IDU (AssocConf);
- AssociationIndication IDU (AssocInd);
- AssociationResponse IDU (AssocResp).

The use of these IDUs is conditional and depends on the actual situation at the PCI (see clause 6).

8.2.2.2 Establishment of a file access

After the file transfer association has been established successfully (see subclause 8.2.2.1), the file access shall be established with the file access establishment service before any file operation (e.g. send a file, receive a file, rename a file) are attempted. The file access establishment service is not applicable for SimpleFTAM.

The IDUs offered at the PCI for the file access establishment procedure are:

- FileAccessEstablishmentRequest IDU (AccessReq);
- FileAccessEstablishmentConfirmation IDU (AccessConf);
- FileAccessEstablishmentIndication IDU (AccessInd);
- FileAccessEstablishmentResponse IDU (AccessResp).

The use of these IDUs is conditional and depends on the actual situation at the PCI (see clause 6).

8.2.2.3 Release of a file access

If no further file operation are to be processed and the two step procedure was used to establish a file transfer connection, the file access shall be released before the established file transfer association can be terminated. To perform this, the file access release service shall be used. The file access release service is not applicable for SimpleFTAM.

The IDUs offered at the PCI for the file access release procedure are:

- FileAccessReleaseRequest IDU (EndAccessReq);
- FileAccessReleaseConfirmation IDU (EndAccessConf);
- FileAccessReleaseIndication IDU (EndAccessInd);
- FileAccessReleaseResponse IDU (EndAccessResp).

The use of these IDUs is conditional and depends on the actual situation at the PCI (see clause 6).

8.2.2.4 Termination of a file transfer association

After successful release of the file access, the file transfer association shall be terminated with the file transfer termination service. The file transfer termination service is not applicable for SimpleFTAM.

The IDUs offered at the PCI for the termination of the association procedure are:

- TerminateRequest IDU (TerminateReq);
- TerminateConfirmation IDU (TerminateConf);
- TerminateIndication IDU (TerminateInd);
- TerminateResponse IDU (TerminateResp).

The use of these IDUs is conditional and depends on the actual situation at the PCI (see clause 6).

8.2.3 Abort of a file transfer connection

In exceptional situations, the file transfer connection may be released with the abrupt termination procedure. No further file operation will be possible. The file transfer connection abort service is applicable to both file transfer protocols, SimpleFTAM and for Easyfile.

The IDUs offered at the PCI for the abrupt file transfer termination procedure are:

- AbortRequest IDU (AbortReq);
- AbortConfirmation IDU (AbortConf);
- AbortIndication IDU (AbortInd).

The use of these IDUs is conditional and depends on the actual situation at the PCI (see clause 6).

8.3 File transfer services

After successful establishment of the file transfer connection and in the case of Easyfile the successful establishment of a file access, the file transfer services shall be used to perform file operations on that file transfer connection.

The file transfer services are dependent of the file transfer protocol which is to be used to perform the file operations. The following file transfer services are offered at the PCI:

- *send file;*
- *send message;*
- *receive file;*
- *delete file;*
- *rename file;*
- *cancel a file operation;*
- *end of file operation;*
- *read file attributes;*
- *change file attributes;*
- *list remote filestore information;*
- *navigation on the remote filestore.*

The file transfer services:

- *send file;*
- *receive file;*
- *delete file;*
- *rename file;*
- *cancel a file operation;*
- *end of file operation;*
- *list remote filestore information;*
- *navigation on the remote filestore;*

are applicable to for SimpleFTAM and Easyfile.

The file transfer services, *read file attributes* and *change file attributes* are applicable only for SimpleFTAM, whereas the file transfer service *send message* shall be applied only for Easyfile.

8.3.1 Send file

The file transfer service *send file* shall be used to transmit a file to a filestore on the established file transfer connection. This service is applicable to SimpleFTAM and for Easyfile.

The IDUs offered at the PCI for the *send file* procedure are:

- SendRequest IDU (SendReq);
- SendConfirmation IDU (SendConf);
- SendIndication IDU (SendInd);
- SendResponse IDU (SendResp).

The use of these IDUs is conditional and depends on the actual situation at the PCI (see clause 6).

8.3.2 Send message

The file transfer service *send message* shall be used to transmit a message on the established file transfer connection. This service shall be applied only for Easyfile.

The IDUs offered at the PCI for the *send message* procedure are:

- SendMessageRequest IDU (SendMsgReq);
- SendMessageConfirmation IDU (SendMsgConf);
- SendMessageIndication IDU (SendMsgInd).

The use of these IDUs is conditional and depends on the actual situation at the PCI (see clause 6).

8.3.3 Receive file

The file transfer service *receive file* shall be used to get a file from a remote filestore on the established file transfer connection. This service is applicable to SimpleFTAM and for Easyfile.

The IDUs offered at the PCI for the receive file procedure are:

- ReceiveRequest IDU (ReceiveReq);
- ReceiveConfirmation IDU (ReceiveConf);
- ReceiveIndication IDU (ReceiveInd);
- ReceiveResponse IDU (ReceiveResp).

The use of these IDUs is conditional and depends on the actual situation at the PCI (see clause 6).

8.3.4 Delete file

The file transfer service *delete file* shall be used to delete a file of a remote filestore. This service is applicable to SimpleFTAM and for Easyfile.

The IDUs offered at the PCI for the delete file procedure are:

- DeleteRequest IDU (DelReq);
- DeleteConfirmation IDU (DelConf);
- DeleteIndication IDU (DelInd);
- DeleteResponse IDU (DelResp).

The use of these IDUs is conditional and depends on the actual situation at the PCI (see clause 6).

8.3.5 Rename file

The file transfer service *rename file* shall be used to rename a file of a remote filestore. This service is applicable to SimpleFTAM and for Easyfile.

The IDUs offered at the PCI for the rename file procedure are:

- RenameRequest IDU (RenameReq);
- RenameConfirmation IDU (RenameConf);
- RenameIndication IDU (RenameInd);
- RenameResponse IDU (RenameResp).

The use of these IDUs is conditional and depends on the actual situation at the PCI (see clause 6).

8.3.6 Cancel a file operation

The file transfer service *cancel a file operation* shall be used to cancel a previous requested file operation. This service is applicable to SimpleFTAM and for Easyfile.

The IDUs offered at the PCI for the cancel a file operation procedure are:

- CancelRequest IDU (CancelReq);
- CancelConfirmation IDU (CancelConf);
- CancelIndication IDU (CancelInd).

The use of these IDUs is conditional and depends on the actual situation at the PCI (see clause 6).

8.3.7 End of a file operation

If a file operation was requested by a remote user, the LA which is in charge of that transfer connection shall know at what time the requested operation is performed. Therefore, the end of a file operation service shall be used to inform the responsible LA that the requested operation has been processed. This service is applicable to SimpleFTAM and for Easyfile.

The IDUs offered at the PCI for the end of a file operation procedure are:

- EndOfFileOperationIndication IDU (EndOperationInd).

The use of these IDUs is conditional and depends on the actual situation at the PCI (see clause 6).

8.3.8 Read file attributes

The file transfer service *read file attributes* shall be used to retrieve the attributes of file on a remote filestore. This service shall be applied only for SimpleFTAM.

The IDUs offered at the PCI for the read file attributes procedure are:

- ReadAttributeRequest IDU (ReadAttribReq);
- ReadAttributeConfirmation IDU (ReadAttribConf);
- ReadAttributeIndication IDU (ReadAttribInd);
- ReadAttributeResponse IDU (ReadAttribResp).

The use of these IDUs is conditional and depends on the actual situation at the PCI (see clause 6).

8.3.9 Change file attributes

The file transfer service *change file attributes* shall be used to change the attributes of file on a remote filestore. This service shall be applied only for SimpleFTAM.

The IDUs offered at the PCI for the change file attributes procedure are:

- ChangeAttributeRequest IDU (ChangeAttribReq);
- ChangeAttributeConfirmation IDU (ChangeAttribConf);
- ChangeAttributeIndication IDU (ChangeAttribConf);
- ChangeAttributeResponse IDU (ChangeAttribResp).

The use of these IDUs is conditional and depends on the actual situation at the PCI (see clause 6).

8.3.10 List remote filestore information

The file transfer service *list remote filestore information* shall be used to get information about a remote filestore, e.g. name of the files of a directory on the remote filestore. This service is applicable to SimpleFTAM and for Easyfile.

The IDUs offered at the PCI for the list remote filestore information procedure are:

- ListRemoteFilestoreInfoRequest IDU (ListReq);
- ListRemoteFilestoreInfoConfirmation IDU (ListConf);
- ListRemoteFilestoreInfoIndication IDU (ListInd);
- ListRemoteFilestoreInfoResponse IDU (ListResp).

The use of these IDUs is conditional and depends on the actual situation at the PCI (see clause 6).

8.3.11 Navigation on the remote filestore

The file transfer service *navigation on the remote filestore* shall be used to navigate on the remote filestore, i.e. change the current directory. This service is applicable to SimpleFTAM and for Easyfile.

The IDUs offered at the PCI for the navigation on the remote filestore procedure are:

- NavigateRequest IDU (NavigateReq);
- NavigateConfirmation IDU (NavigateConf);
- NavigateIndication IDU (NavigateInd);
- NavigateResponse IDU (NavigateResp).

The use of these IDUs is conditional and depends on the actual situation at the PCI (see clause 6).

8.4 Local PCI services

Beside the network connection services, the file transfer connection services, and the file transfer services as defined in the previous subclauses, local services are also specified in this ETS.

These PCI services shall be used to get information about the local system. Local PCI operations shall be requested by an LA and shall be processed by a CA after successful establishment of a local communication channel between LA and CA. They shall not have any influence on the communication with a remote system. Consequently the local PCI services are independent from the file transfer protocol supported by a CA. The local PCI services are applicable to SimpleFTAM and for Easyfile.

Local PCI services are CA dependent and have great impact on the implementation of a CA. Therefore only a minimum set of those services is specified. The local PCI services offered at the PCI are:

- *redirection of incoming calls;*
- *list connected LAs;*
- *list local filestore information;*
- *read the CA capabilities;*
- *read CA configuration;*
- *read phonebook entry;*
- *read logbook;*
- *read access control list entry.*

8.4.1 Redirection of incoming calls

An LA may request the CA to redirect an incoming call to another LA. Incoming calls can be redirected only to an LA which have established a local communications channel to the CA. The local PCI service *redirection of incoming calls* is applicable to SimpleFTAM and for Easyfile if the corresponding CA supports this optional service.

The IDUs offered at the PCI for the list local filestore information procedure are:

- RedirectionRequest IDU (RedirectReq);
- RedirectionConfirmation IDU (RedirectConf).

The use of these IDUs is optional (see clause 6).

8.4.2 List connected LAs

The LA may request the CA to provide information about the LAs which are actually connected to the CA. The local PCI service *list connected LAs* is applicable to SimpleFTAM and for Easyfile if the correspondent CA supports this optional service.

The IDUs offered at the PCI for the list local filestore information procedure are:

- ListConnectedLARequest IDU (ListLAReq);
- ListConnectedLAConfirmation IDU (ListLAConf).

The use of these IDUs is optional (see clause 6).

8.4.3 List local filestore information

The LA may request the CA to provide information about the local filestore, e.g. name of the files of a directory on the local filestore. The local PCI service *list local filestore information* is applicable to SimpleFTAM and for Easyfile if the correspondent CA supports this optional service.

The IDUs offered at the PCI for the list local filestore information procedure are:

- ListLocalFilestoreInfoRequest IDU (LocListReq);
- ListLocalFilestoreInfoConfirmation IDU (LocListConf).

The use of these IDUs is optional (see clause 6).

9.4.4 Read CA capabilities

This service enables an LA to get information about the product specific features and services of the CA (e.g. supported file transfer protocol, structure of an entry in the phonebook) with which it has established a local communications channel. The product specific capabilities of the CA which shall be provided by the CA manufacturer (see clause 10) shall be conveyed to the LA with this service. The local PCI service *read CA capabilities* is applicable to SimpleFTAM and for Easyfile if the correspondent CA supports this optional service.

The IDUs offered at the PCI for the read CA capabilities procedure are:

- ReadCACapabilityRequest IDU (CapabilityReq);
- ReadCACapabilityConfirmation IDU (CapabilityConf).

The use of these IDUs is optional (see clause 6).

8.4.5 Read CA configuration

The LA may request information about the configuration of a CA, e.g. configuration of each layer of the file transfer protocol stack. The local PCI service *read CA configuration* is applicable to SimpleFTAM and for Easyfile if the correspondent CA supports this optional service.

The IDUs offered at the PCI for the read CA configuration procedure are:

- ReadConfigurationRequest IDU (ConfigReq);
- ReadConfigurationConfirmation IDU (ConfigConf).

The use of these IDUs is optional (see clause 6).

8.4.6 Read phonebook entry

The LA may request information about a communication partner, e.g. the network address. The local PCI service *read phonebook entry* is applicable to SimpleFTAM and for Easyfile if the correspondent CA supports this optional service.

The IDUs offered at the PCI for the read phonebook entry procedure are:

- ReadPhonebookEntryRequest IDU (PhonebookReq);
- ReadPhonebookEntryConfirmation IDU (PhonebookConf).

The use of these IDUs is optional (see clause 6).

8.4.7 Read logbook

The LA may request information of the logbook maintained by the CA, e.g. result of a send file request. The local PCI service *read logbook* is applicable to SimpleFTAM and for Easyfile if the correspondent CA supports this optional service.

The IDUs offered at the PCI for the read logbook procedure are:

- ReadLogbookRequest IDU (LogbookReq);
- ReadLogbookConfirmation IDU (LogbookConf).

The use of these IDUs is optional (see clause 6).

8.4.8 Read access control information

The LA may request the access control information of the local filestore. The local PCI service *read access control information* is applicable to SimpleFTAM and for Easyfile if the correspondent CA supports this optional service.

The IDUs offered at the PCI for the read access control information procedure are:

- ReadAccessControlRequest IDU (AccessControlReq);
- ReadAccessControlConfirmation IDU (AccessControlConf).

The use of these IDUs is optional (see clause 6).

9 Service definition

The information necessary to perform one of the services offered at the PCI is composed by general PCI parameters (see clause 7) and IDUs as defined in clause 8.

The structure of an IDU is independent from its type (Req, Conf, Ind, Resp). The parameters of each IDU belong to one of the following categories:

- general IDU parameters:
  a CA claiming conformance to this ETS shall support the general IDU parameters; the general IDU parameters are specified in subclause 9.1;

- SimpleFTAM specific IDU parameters:
  a CA claiming conformance to the SimpleFTAM services as specified in this ETS shall support the SimpleFTAM specific IDU parameters; the SimpleFTAM specific IDU parameters are specified in subclause 9.2;

- Easyfile specific IDU parameters: a CA claiming conformance to the Easyfile services specified in this ETS shall support the Easyfile specific IDU parameters; the Easyfile specific IDU parameters are specified in subclause 9.3.

9.1 General IDU parameters

The general IDU parameters which are specified in this ETS shall be used to:

- distinguish the different services requested with the IDUs;
- distinguish the different types of IDUs, the PCI service primitives, related to one service;
- to handle and control the flow of IDUs exchanged between LA and CA at the PCI.

Table 14 summarizes the general IDU parameters and their meaning.

Table 14

Parameter	Abbreviation	Purpose
file transfer protocol specific IDU parameters	ft_parameters	placeholder for the set of those IDU parameters which are file transfer protocol specific, i.e. which are dependent on the file transfer protocols *SimpleFTAM* and *Easyfile*. these file transfer protocol specific IDU parameters are specified in subclauses 9.2.2 and 9.3.2.
IDU reference	IDU_ref	unique identification of the IDUs exchanged at the PCI; this sequence number is maintained by the LA for all request IDUs; for all indication IDUs this sequence number is maintained by the CA; is used also to align a confirmation IDU from the CA to a previous request IDU from the LA and to align a response IDU from the LA to a previous indication IDU from the CA.
network connection identifier	netcon_id	unique identification of a successfully established network connection between an LA and a remote system or a remote user; this parameter is handled by the CA; there is only one exception of this rule: if an LA wants to establish a network connection, it shall assign a temporary value to this parameter to be able to disconnect the requested network connection before it is established successfully; the temporarily assigned value may be overwritten by the CA after successful establishment of the network connection.
result	result	result of the requested service; possible values: success (0), transient-error (1), permanent-error (2).
service identifier	service_id	unique identification of the service requested with the IDU (e.g. establishment of the network connection, send file, rename local file).
service primitive identifier	sp_id	unique identification of the PCI service primitive of the requested service; possible values: request (1), indication (2), response (3), confirmation (4).

Table 15 shows the use of the general IDU parameters related to the PCI service primitives.

Table 15

Parameter	Request	Indication	Response	Confirmation
ft_parameters	o	o	o	o
IDU_ref	m	m	m	m
netcon_id	m	m	m	m
result	-	-	m	m
service_id	m	m	m	m
sp_id	m	m	m	m

9.2 SimpleFTAM service definition

This subclause specifies the parameters of an IDU which shall be used for the file transfer service and protocol SimpleFTAM.

SimpleFTAM is specified in accordance with:

- ETS 300 410 [7];
- ETS 300 388 [5];
- ISO/IEC 8571, parts 1 to 5 [9];
- ISO/IEC 8571, part 3 [9];
- ISO/IEC 8571, part 3, Amendment 1 [9].

ISO/IEC 8571 [9] defines a general file model and specifies a set of functions for the access of such a virtual filestore using a defined protocol, the FTAM file transfer protocol and the FTAM file service. It provides sufficient facilities to support file transfer, file access and file management. SimpleFTAM as specified in this ETS is a subset of FTAM. The concepts and mechanisms, data structures and data types defined in ISO/IEC 8571 [9] have been applied for SimpleFTAM. SimpleFTAM is therefore conformant to ISO/IEC 8571 [9].

Major impacts on the SimpleFTAM service definition derive from ETS 300 388 [5] were specific requirements necessary for the ISDN terminal-to-terminal environment are defined. These requirements are additional constraints to the ISO/IEC profile specification AFT11, see ISO/IEC 10607-3 [10]. They are marked in the following subclauses.

ETS 300 388 [5] distinguishes between implementations claiming basic conformance and implementations claiming full conformance to the FTAM simple file transfer teleservice. To cover this differences two SimpleFTAM profiles are specified in subclause 9.2.7.

In subclause 9.1 the parameter "ft-parameters" was defined as a placeholder. This placeholder is to be replaced by the SimpleFTAM specific IDU parameters as specified for each service in the following subclauses.

9.2.1 SimpleFTAM IDUs

In clause 8 all IDUs were defined which may be used at the PCI. For SimpleFTAM, only the IDUs shown in table 16 shall be applied.

Table 16: Simple FTAM IDUs

Network connection services	IDUs
establishment of a network connection	ConReq, ConConf, ConInd, ConResp
release of a network connection	DiscReq, DiscConf, DiscInd
File transfer connection services	IDUs
establishment of a file transfer connection	EstabReq, EstabConf, EstabInd, EstabResp
termination of a file transfer connection	RelReq, RelConf, RelInd, RelResp
abort of a file transfer connection	AbortReq, AbortConf, AbortInd
File transfer services	IDUs
send file	SendReq, SendConf, SendInd, SendResp
receive file	ReceiveReq, ReceiveConf, ReceiveInd, ReceiveResp
delete file	DelReq, DelConf, DelInd, DelResp
rename file	RenameReq, RenameConf, RenameInd, RenameResp
cancel a file operation	CancelReq, CancelConf, CancelInd
end of file operation	EndOperationInd
read file attributes	ReadAttribReq, ReadAttribConf, ReadAttribInd, ReadAttribResp
change file attributes	ChangeAttribReq, ChangeAttribConf, ChangeAttribInd, ChangeAttribResp
list remote filestore information	ListReq, ListConf, ListInd, ListResp
navigation on the remote filestore	NavigateReq, NavigateConf, NavigateInd, NavigateResp
Local PCI services	IDUs
redirection of incoming calls	ListLAReq, ListLAConf
list connected LAs	ListLAReq, ListLAConf
list local filestore information	LocListReq, LocListConf
read CA capabilities	CapabilityReq, CapabilityConf
read CA configuration	ConfigReq, ConfigConf
read phonebook entry	PhonebookReq, PhonebookConf
read logbook	LogbookReq, LogbookConf
read access control information	AccessControlReq, AccessControlConf

9.2.2 SimpleFTAM specific IDU parameters

The SimpleFTAM specific IDU parameters which shall be used at the PCI are a subset of the FTAM parameters as specified in ISO/IEC 8571-3 [9]. Additionally parameters which are related to the network ISDN and parameters which are introduced for the specific needs of the PCI as specified in this ETS are described in the following.

Address information related parameters

Address information related parameters shall be used to convey information associated with the calling and called entity. At the PCI the following address information related parameters shown in table 17 are used:

Table 17: Address information related parameters

Address related parameter	Abbreviation	Purpose
initiator identity	initiator_id	identifies the user which initiates the file transfer connection.
called application title	called_application_ title	identifies the title of the entity called by the initiator of the file transfer connection; for SimpleFTAM form 2 shall be used: AP_Title: Object Identifier; AE_Qualifier: Integer.
called presentation address	called_presentation_ addr	identifies the Presentation Service Access Point (PSAP) of the entity called by the initiator of the file transfer connection.
called address	called_addr	network address, i.e. ISDN address, called by the initiator of the network connection.
called subaddress	called_subaddr	network subaddress, i.e. ISDN network subaddress (MSN), called by the initiator of the network connection.
calling address	calling_addr	network address, i.e. ISDN address, of the initiator of a network connection.
calling subaddress	calling_subaddr	network subaddress, i.e. ISDN network subaddress (MSN), of the initiator of a network connection.
called name	called_name	unique identification (alias name) of the called communication partner; this parameter shall be used only if the phonebook feature is supported by a CA.
responding address	responding_addr	network address, i.e. ISDN address, of the responding entity.
responding subaddress	responding_subaddr	network subaddress, i.e. ISDN network subaddress (MSN).

Network related parameters

Network related parameters shall be used to convey information associated with the physical network, i.e. with ISDN. At the PCI the network related parameters shown in table 18 are used.

Table 18: Network related parameters

Network related parameter	Abbreviation	Purpose
network charging	net_charge	Charging information.
network reason	net_reason	reason for rejection of a network connection service on the D-channel (cause according to ITU-T RecommendationQ.931 [11]).
network diagnostics	net_diag	additional information to the network reason. This parameter may provide additional cause and diagnostic for the release of the communication on the B-channel.
network parameter	net_param	this parameter is a placeholder for additional network parameters which are supported by a CA but not specified in this ETS; shall be specified by the CA in the ICE.
user to user identification	uuid	see subclause 7.2.

Passwords

Passwords shall be used to verify if the user requesting a specific service is authorized to perform that service. A password is either a GraphicString or an OCTET STRING. At the PCI the types of passwords shown in table 19 are used.

Table 19: Password parameters

Passwords	Abbreviation	Purpose
change attribute password	change_attribute_ password	password used to check the permission to change the attributes of a file of the remote virtual filestore.
create password	create_password	password used to check the permission to create a file on a filestore.
delete password	delete_password	password used to delete a file of the remote virtual filestore.
filestore password	filestore_password	password used to check the general permission to access a filestore.
list password	list_password	password is used to check if an LA is allowed to get the list of LAs which have been established a local communications channel to the CA.
read attribute password	read_attribute_password	password used to read the attributes of a file of the remote virtual filestore.
read password	read_password	password used to read the content of a file of the remote virtual filestore.
write password	write_password	password used to write data into a file of the remote virtual filestore: possible write access actions which can be performed on whole files and are supported at the PCI are: extend, insert, replace.

File access related parameters

File access related parameters shall be used to convey information associated with the access and the content of a file on a filestore. At the PCI the file access shown in table 20 related parameters are used:

Table 20: File access related parameters

File access related parameters	Abbreviation	Purpose
contents type	contents_type	specifies the content of a file using the document type; for SimpleFTAM the parameter content type shall be a string; the following values are supported at the PCI: FTAM-1, FTAM-2, FTAM-3, NBS-9 and unknown (NULL); the value unknown shall be used only when opening an already existing file; specific ETS 300 388 [5] requirements: FTAM-1: mandatory for basic conformance and full conformance; FTAM-2: optional for basic conformance and full conformance; FTAM-3: mandatory for basic conformance and full conformance; NBS-9: optional for basic conformance and mandatory for full conformance.
contents type list	contents_type_list	specifies in form of a list the possible content types which can be accessed during the file transfer connection; the parameter contents type list shall be a list of file contents which can be accessed during a file transfer connection (see contents type).
override	override	defines the create behaviour which shall be taken if a file already exists; this parameter shall have one of the following values: create-failure(0), select-old-file(1), delete-and-create-with-old-attributes(2), delete-and-create-with-new-attributes(3).
requested access	requested_access	indicates the operations to be performed on a file; for SimpleFTAM the parameter requested access shall have one of the following values: read(0), insert(1), replace(2), extend(3), read-attribute(5), change-attribute(6), delete-file(7).
reset	reset	shall be used to navigate on the filestore in conjunction with the parameter directory name (see below); it is a boolean parameter which shall not be used if the parameter directory name contains the complete path to the destination directory; if the parameter directory name contains the incomplete path to the destination directory the parameter reset has the following meaning: TRUE: reset the current directory to the directory specified at the creation of the FTAM regime and than append the value of the parameter directory name; FALSE: append the value of the parameter directory name to the current directory.
concurrency control	concurrency_control	identifies the access rights of users which wants to perform a file transfer operation at the same time; the parameter concurrency control is a list of access rights which shall be specified for the each file operation (read, insert, replace, extend, erase, read-attribute, change-attribute, delete-file) at the beginning of a regime; for each element of the list one of the following integer values shall be assigned: not-required(0), shared(1), exclusive(2), no-access(3)
directory name	dir_name	path to the directory of a local or remote filestore; pathnames shall be provided in the naming convention of the corresponding implementation; shall be specified by the CA manufacturer.

File attribute related parameters

File attribute parameters shall be used to convey information of the file attributes associated with a file on a filestore. At the PCI the file attribute parameters shown in table 21 are used:

Table 21: File attribute related parameters

File attribute related parameters	Abbreviation	Purpose
attribute groups	attribute_groups	shall be used to negotiate the set of file attribute groups to be accessible during the file transfer connection; the selection of the following attribute groups is possible: storage(0), security(1), private(2); default: if no value is specified, only the kernel attributes (filename-attribute, permitted-actions-attribute, contents-type-attribute) are accessible; specific ETS 300 388 [5] requirements: storage: shall be optional for basic conformance and mandatory for full conformance; security: shall be optional for the responder; should not be used with the Enhanced File Management functional unit; private: not applicable for SimpleFTAM.

attribute names	attribute_names	applicable for the attributes of those attribute groups negotiated for the file transfer connection; the attribute names specified with this parameter shall be those attributes of a file which values shall be read;
		attribute values may be requested for:
		kernel group attributes:
		read-filename(0), read-permitted-actions(1), read-contents-type(2);
		storage group attributes:
		read-storage-account(3), read-date-and-time-of-creation(4), read-date-and-time-of-last-modification(5), read-date-and-time-of-last-read-access(6), read-date-and-time-of-last-attribute-modification(7), read-identity-of-creator(8), read-identity-of-last-modifier(9), read-identity-of-last-reader(10), read-identity-of-last-attribute-modifier(11), read-file-availability(12), read-filesize(13), read-future-filesize(14);
		security group attributes:
		read-access-control(15), read-legal-qualifications(16);
		specific ETS 300 388 [5] requirements:
		private group attributes (read-private-use(17)) shall not be used for SimpleFTAM.

(continued)

Table 21 (continued): File attribute related parameters

File attribute related parameters	Abbreviation	Purpose
attribute values	attribute_values	applicable for the attributes of those attribute groups negotiated for the file transfer connection; this parameter provides the actual attribute values of a file; the following attribute values may be provided: kernel group attributes (filename-attribute, permitted-actions-attribute, contents-type-attribute), storage group attributes (account-attribute, date-and-time-of-creation-attribute, date-and-time-of-last-modification-attribute, date-and-time-of-last-read-access-attribute, date-and-time-of-last-attribute-modification-attribute, identity-of-creator-attribute, identity-of-last-modifier-attribute, identity-of-last-reader-attribute, identity-of-last-attribute-modifier-attribute, file-availability-attribute, filesize-attribute, future-filesize-attribute), security group attributes (access-control-attribute, legal-qualification-attribute); specific ETS 300 388 [5] requirements: private group attributes (private-use-attribute) shall not be used for SimpleFTAM.
change attributes	change_attributes	applicable for the attributes of those attribute groups negotiated for the file transfer connection; the attribute values specified with this parameter shall be the new attribute values of a file on the remote virtual filestore; change attribute values may be specified for: kernel group attributes (filename-attribute), storage group attributes (account-attribute, file-availability-attribute, future-filesize-attribute), security group attributes (access-control-attribute, legal-qualifications-attribute); specific ETS 300 388 [5] requirements: private group attributes (private-use-attribute) shall not be used for SimpleFTAM.
initial attributes	initial_attributes	applicable for the attributes of those attribute groups negotiated for the file transfer connection; the attribute values specified with this parameter shall be the attribute values of a newly created file on the remote virtual filestore; initial attribute values may be specified for: kernel group attributes (filename-attribute, permitted-actions-attribute, contents-type-attribute), storage group attributes (account-attribute, file-availability-attribute, filesize-attribute), security group attributes (access-control-attribute, legal-qualification-attribute); specific ETS 300 388 [5] requirements: private group attributes (private-use-attribute) shall not be used for SimpleFTAM.
access control information	access_control_info	used to retrieve the actual values of the attribute access-control of a file of the local virtual filestore; this list consists of the list of operations (read, insert, replace, extend, read-attribute, change-attribute, delete-file) allowed for that file, the concurrency control for each operation, the password for each operation, the initiator identity and the application entity title.

(continued)

Table 21 (concluded): File attribute related parameters

File attribute related parameters	Abbreviation	Purpose
file list	file_list	used to retrieve the list of files in a directory of the local and remote virtual filestore; this parameter shall be used in conjunction with the parameter attribute names (see above); if the parameter attribute names is empty only the filenames shall be retrieved; otherwise the values of the specified attribute names shall be retrieved for each file of the directory; the parameter file list is therefore a list of the parameter attribute values.
source effect	src_effect	used to identify which action shall be performed after transmission of a local file: delete the file after successful transmission on the local filestore (move) or store the file on the local filestore (copy); possible values: copy-file(0), move-file(1).
source filename	src_filename	name of a source file of a filestore; filenames shall be provided in the naming convention of the corresponding implementation; shall be specified by the CA manufacturer.
target filename	tgt_filename	used to identify the name of a file on a remote filestore in conjunction with the parameter source filename (see above); if the target filename is to be different to the source filename, this parameter shall be used; otherwise this parameter shall not be used; filenames shall be provided in the naming convention of the corresponding implementation; shall be specified by the CA manufacturer.
filename	filename	used to retrieve information from a CA in the named file; filenames shall be provided in the naming convention of the corresponding implementation; shall be specified by the CA manufacturer.

Protocol related parameters

Protocol related parameters shall be used to negotiate facilities for file transfer connection. At the PCI the protocol related parameters shown in table 22 are used:

Table 22: Protocol related parameters

Protocol related parameters	Abbreviation	Purpose
service class	service_class	negotiates the service class of the file transfer connection; for SimpleFTAM this parameter shall have one of the following values: management-class(1), transfer-class(2), transfer-and-management-class(3); specific ETS 300 388 [5] requirements: transfer-class: mandatory for basic conformance and for full conformance transfer-and-management-class: optional for basic conformance and mandatory for full conformance
functional units	FU	negotiates the set of functional unit capabilities for the file transfer connection; in correspondence to the parameter service class this parameter of type BIT STRING may have the following values: read(2), write(3), file-access(4), limited-file-management(5), enhanced-file-management(6), grouping(7), fadu-locking(8), recovery(9), restart-data-transfer(10); specific ETS 300 388 [5] requirements: read: mandatory for basic conformance and for full conformance; write: mandatory for basic conformance and for full conformance; file-access: not applicable limited-file-management: mandatory for basic conformance and for full conformance; enhanced-file-management: optional for basic conformance and mandatory for full conformance grouping: mandatory for basic conformance and for full conformance; fadu-locking: not applicable; recovery: optional for basic conformance and mandatory for full conformance; restart-data-transfer: optional for basic conformance and for full conformance.
quality of service	QoS	determines the quality of service which shall be negotiated for the file transfer connection; this parameter shall have one of the following values: no-recovery(0), class-1-recovery(1), class-2-recovery(2), class-3-recovery(3).

Cost related parameters

Several regimes are specified in ISO/IEC 8571 [9]. The costs which occur during a regime shall be charged to a specific account. At the PCI the cost related parameters shown in table 23 are used:

Table 23: Cost related parameters

Cost related parameters	Abbreviation	Purpose
account	account	identifies the account which shall be in charge of the costs occurred during a regime; the account shall be determined at the beginning of a regime.
charging	charging	carries information about the costs occurred during a regime; the parameter charging shall be used only if the parameter account was used at the beginning of a regime; the value of this parameter is a list of triples; elements in one triple: resource-identifier (GraphicString), charging-unit (GraphicString), charging-value (INTEGER).
charging units	charging_unit	number of units accumulated on the network connection.

Result parameters

Result parameters shall be used to convey detailed information on the result of a requested file transfer operation. At the PCI the result parameters shown in table 24 are used:

Table 24: Result parameters

Result parameters	Abbreviation	Purpose
diagnostics	diagnostics	the parameter diagnostics is a list of diagnostic information; each element in the list consists of: diagnostic-type (INTEGER), error-identifier (INTEGER), error-observer (INTEGER), error-source (INTEGER), suggested-delay (INTEGER), further-details (GraphicString).

Specific PCI parameters

Specific PCI parameters are introduced for specific PCI matters. At the PCI the specific PCI parameters shown in table 25 are used:

Table 25: Specific PCI parameters

Specific PCI parameters	Abbreviation	Purpose
file transfer connection identifier	ft_con_id	used to identify uniquely the file transfer connection requested by an LA at the PCI.
network connection type	netcon_type	type of the network connection which shall be used; for the time being there is only one valid network selectable at the PCI: ISDN.
list of LA login names	login_name_list	list login names of those LAs which are locally connected to the CA.
redirect IDU	redirect-IDU	the IDU, carrying the incoming data from a remote system, which shall be redirected to another actually logged in LA.
new LA login name	new_LA	login name of the LA to which the incoming data shall be redirected.
directory name	dir_name	path to the directory on the local or remote filestore.
CA identification	ca_id	unique identification of a CA (see also subclause 7.2).
CA configuration	CA_config	list of configuration information of a SimpleFTAM CA; the parameter CA_config is a collection of the facilities offered by a CA; shall be specified by the CA manufacturer.
CA capabilities	CA_capabilities	information about the capabilities of a SimpleFTAM CA as declared in the ICE; content, structure and data format of the parameter CA_descriptor is product dependent; shall be specified by the CA manufacturer.
logbook information	logbook_info	part of the logbook which is maintained by a CA; content, structure and data format of the parameter logbook_info is product dependent; shall be specified by the CA manufacturer.
selection criteria	selection_criteria	unique identification to select a part of the logbook which shall be retrieved.
phonebook entry	phonebook_entry	collection of information (e.g. network address) of the different LAs and remote systems or remote communication partners; content, structure and data format of the parameter phonebook_entry is product dependent; shall be specified by the CA manufacturer.
search key	search_key	identifier to search a specific entry in the phonebook which shall be retrieved.
sort key	sort_key	identifier to sort the list of files of a filestore.

9.2.3 Network connection services

The network connection services offered at the PCI are:

- establishment of a network connection;
- release of a network connection.

9.2.3.1 Establishment of a network connection

The network connection shall be established before a file transfer connection shall be established.

Table 26

Parameter	Purpose
calling_addr	network address, i.e. ISDN address, of the initiator of a network connection.
calling_subaddr	network subaddress, i.e. ISDN network subaddress (MSN).
called_addr	network address, i.e. ISDN address, called by the initiator of the network connection.
called_subaddr	network subaddress, i.e. ISDN network subaddress (MSN).
called_name	unique identification (alias name) of the called communication partner; this parameter shall be used only if the phonebook feature is supported by a CA.
responding_addr	network address, i.e. ISDN address, called by the initiator of the responding entity.
responding_subaddr	network subaddress, i.e. ISDN network subaddress (MSN).
netcon_type	type of the network connection which shall be used; for the time being the only valid network selectable at the PCI is ISDN.
net_reason	reason for rejection of a network connection service.
net_diag	additional information to the network reason.
net_param	this parameter is a placeholder for additional network parameters which are supported by a CA but not specified in this ETS; shall be specified by the CA manufacturer.
uuid	unique identification of a user.

Table 27

Parameter	Request	Indication	Response	Confirmation
calling_addr	c	c	-	-
calling_subaddr	o	o	-	-
called_addr	m	o	-	-
called_subaddr	o	o	-	-
called_name	c	-	-	-
responding_addr	-	-	c	c
responding_subaddr	-	-	o	o
netcon_type	o	o	-	-
net_reason	-	-	-	o
net_diag	-	-	-	o
net_param	o	o	-	-
uuid	o	o	-	-

9.2.3.2 Release of a network connection

The network connection should be released if there is no longer the need to perform file transfer connection services or file transfer services.

Table 28

Parameter	Purpose
net_charge	network charging information.
net_reason	reason for rejection of a network connection service.
net_diag	additional information to the network reason.

Table 29

Parameter	Request	Indication	Confirmation
net_charge	-	o	o
net_reason	-	o	o
net_diag	-	o	o

9.2.4 File transfer connection services

9.2.4.1 Establishment of a file transfer connection

The establishment of the file transfer connection is mapped to the confirmed FTAM service F-Initialize.

The following parameters of the F-Initialize service primitives are not exchanged at the PCI:

- calling application title (related to the application over ACSE and therefore equal for each LA);
- calling presentation address (FTAM service provider related and therefore CA specific);
- responding application title (necessary for recovery; recovery is not visible at the PCI);
- responding presentation address (necessary for recovery; recovery is not visible at the PCI);
- application context name (the support of {ISO/IEC 8571 [9] application-context(1) iso-ftam (1)} is required (see AFT11); if no value is provided by the user (LA) that Object Identifier {1 0 8571 1 1} shall be used);
- shared ASE information (used in conjunction with the CCR and actually deleted in ISO/IEC 8571 [9]; no use of this parameter known);
- communication QoS (as noted for ACSE and Presentation; should be part of the configuration of the FTAM service provider (CA) if necessary);
- implementation information (used in the F-Initialize PDUs but not in the related F-Initialize service primitives);
- state result (state of the FTAM protocol machine after execution of the function; information which is not necessary for the PCI user).

NOTE: If the FTAM service provider supports the phonebook feature the following address information should be provided for each remote communication partner and each LA:

- application title;
- presentation address.

Table 30

Parameter	Purpose
ft_con_id	unique identification of the file transfer connection.
called_application_title	is used to identify the title of the entity called by the initiator of the file transfer connection; for SimpleFTAM form 2 shall be used; if the phonebook feature is supported by a CA, the parameter called_name should be used.
called_presentation_addr	is used to identify the Presentation Service Access Point (PSAP) of the entity called by the initiator of the file transfer connection; if the phonebook feature is supported by a CA, the parameter called_name should be used.
called_name	unique identification of the called communication partner (alias name); this parameter shall be used only if the phonebook feature is supported by a CA.
service_class	negotiates the service class (management-class, transfer-class or transfer-and-management-class) of the file transfer connection.
FU	negotiates the set of functional unit (kernel, read, write, limited-file-management, enhanced-file-management, grouping, recovery, restart-data-transfer) for the file transfer connection.
attribute_groups	used to negotiate the set of optional file attribute groups which shall be available on the file transfer connection.
QoS	determines the quality of service which shall be negotiated for the file transfer connection.
contents_type_list	list of document types (see contents type).
initiator_id	identifies the entity which initiates the file transfer connection.
account	identifies the account which shall be in charge of the costs occurred during a regime.
filestore_password	password used to access the virtual filestore.
diagnostics	used to convey detailed information on the failure of a requested file transfer operation; the parameter diagnostics is a list of diagnostic information.

Table 31

Parameter	Request	Indication	Response	Confirmation
ft_con_id	m	m	m	m
called_application_title	c	c	-	-
called_presentation_address	c	c	-	-
called_name	c	c	-	-
service_class (see note 1)	m	m	m	m
FU (see note 2)	m	m	m	m
attribute_groups	m	m	m	m
QoS	m	m	m	m
contents_type_list	m	m	m	m
initiator_id	o	o	-	-
account	o	o	-	-
filestore_password (see note 3)	o	o	-	-
diagnostics	-	-	-	o

NOTE 1: The support of the service class "Transport and Management Class" is mandatory for full conformance (see ETS 300 388 [5]; in AFT11 it is outside the scope, in ISO/IEC 8571 [9] it is optional).

NOTE 2: The support of the FTAM FU "Limited File Management" is mandatory for the service class "Transfer Class" (see ETS 300 388 [4]; in ATF11 and ISO/IEC 8571 [9] it is optional).
For full conformance the FTAM FU "Recovery" is mandatory (see ETS 300 388 [5]; in ATF11 and ISO/IEC 8571 [9] it is optional).
The FTAM FU "Enhanced File Management" is mandatory for full conformance (see ETS 300 388 [5]; in ATF11 and ISO/IEC 8571 [9] it is optional).

NOTE 3: The parameter "filestore password" shall be mandatory for initiators and optional for responders (see ETS 300 388 [5]).

9.2.4.2 Termination of a file transfer connection

The termination of the file transfer connection is mapped to the confirmed FTAM service F-Terminate.

The following parameters of the F-Terminate service primitives are not exchanged at the PCI:

- shared ASE information (used in conjunction with the CCR and actually deleted in ISO/IEC 8571 [9]; no use of this parameter known).

Table 32

Parameter	Purpose
ft_con_id	unique identification of the file transfer connection.
charging	carries information about the costs occurred during a regime.

Table 33

Parameter	Request	Indication	Response	Confirmation
ft_con_id	m	m	m	m
charging (see note)	-	o	-	o

NOTE: Only used if the account parameter was provided by the initiator of the file transfer connection.

9.2.4.3 Abort of a file transfer connection

The abrupt termination of the file transfer connection is mapped to the not confirmed FTAM services F-U-Abort and F-P-Abort.

Table 34

Parameter	Purpose
ft_con_id	unique identification of the file transfer connection.
diagnostics	used to convey detailed information on the failure of a requested file transfer operation; the parameter diagnostics is a list of diagnostic information.

Table 35

Parameter	Request	Indication	Confirmation
ft_con_id	m	m	m
diagnostic	-	-	o

9.2.5 File transfer services

9.2.5.1 Send file

FTAM offers different possibilities to send a file to the remote filestore:

- send a "new" file;
- override or extend the content of an existing file.

Therefore sending a file to the remote Virtual File Store (VFS) is mapped to the confirmed FTAM services F-Select, F-Create, F-Open, F-Close, F-Deselect and the not confirmed service F-Write.

The following parameters of the FTAM service primitives are not exchanged at the PCI:

- shared ASE information (used in conjunction with the CCR and actually deleted in ISO/IEC 8571 [9]; no use of this parameter known);

- enable File Access Data Unit (FADU) locking (used for opening a file; mandatory parameter which is set to false);

- FADU lock (outside the scope of Simple FTAM);

- state result (state of the FTAM protocol machine after execution of the function; not necessary at the PCI);

- depending on the content of the file to be send (contents type) the parameters requested access, processing mode, FADU operation and FADU identity have the following values:

Table 36

Content of the file: FTAM-1 and FTAM-3 Access context: UA			
requested access	**processing mode**	**FADU operation**	**FADU identity**
replace	replace	replace	first
extend	extend	extend	first
Content of the file: FTAM-2 Access context: FA			
requested access	**processing mode**	**FADU operation**	**FADU identity**
insert	insert	insert	end

Table 37

Parameter	Purpose
ft_con_id	unique identification of the file transfer connection.
src_filename	name of the file to be send.
src_effect	delete the file of the local filestore after successful transmission (move) or not (copy).
tgt_filename	name of the file on the remote filestore; default: if no filename is specified, the name of the remote file shall be the name of the source file (src_filename).
override	defines the action which shall be taken if a file already exists.
initial_attributes	applicable for the attributes of those attribute groups negotiated for the file transfer connection; the attribute values specified with this parameter shall be the attribute values of a newly created file on the remote virtual filestore.
create_password	password used to create a new file on the remote virtual filestore.
contents_type	specifies the document type (unknown, FTAM-1, FTAM-2, FTAM-3, NBS-9) of a file.
requested_access	indicates the operations (read, insert, replace, extend, read-attribute, change-attribute and/or delete-file) to be performed on a file.
write_password	password used to write data into file of the remote virtual filestore.
concurrency_control	identifies the access rights for each file operation (read, insert, replace, extend, erase, read-attribute, change-attribute, delete-file) at the beginning of a regime.
account	identifies the account which shall be in charge of the costs occurred during a regime.
charging	carries information about the costs occurred during a regime; the value of this parameter is a list of triples.
diagnostics	used to convey detailed information on the failure of a requested file transfer operation; the parameter diagnostics is a list of diagnostic information.

Table 38

Parameter	Request	Indication	Response	Confirmation
ft_con_id	m	m	m	m
src_filename	m	m	-	-
src_effect	m	-	-	-
tgt_filename	o	-	-	o
override	m	m	-	-
initial_attributes (see note 1)	o	o	o	o
create_password (see note 2)	o	o		
contents_type	m	m	o	o
requested_access	m	m	-	-
write_password	o	o	-	-
concurrency_control	o	o	-	-
account	o	o	-	-
charging	-	o	-	o
diagnostics	-	-	-	o

NOTE 1: The parameter "initial-attributes" shall be supported only if a new file shall be transmitted.

NOTE 2: The parameter "create-password" shall be supported for initiators and optional for responders (see ETS 300 388 [5]).

9.2.5.2 Receive file

Receiving a file from the remote VFS is mapped to the confirmed FTAM services F-Select, F-Open, F-Close, F-Deselect and the not confirmed service F-Read.

The following parameters of the FTAM service primitives are not exchanged at the PCI:

- shared ASE information (used in conjunction with the CCR and actually deleted in ISO/IEC 8571 [9]; no use of this parameter known);

- enable FADU locking (used for opening a file; mandatory parameter which is set to false);
- FADU lock (outside the scope of Simple FTAM);

- state result (state of the FTAM protocol machine after execution of the function; not necessary at the PCI);

- access context (depending of the contents type);

- depending on the content of the file to be received (contents type) the parameters requested access, processing mode and FADU identity have the following values:

Table 39

Content of the file: FTAM-1 and FTAM-3 Access context: UA		
requested access	**processing mode**	**FADU identity**
read	read	first
Content of the file: FTAM-2 Access context: UA or FA		
requested access	**processing mode**	**FADU identity**
read	read	begin

Table 40

Parameter	Purpose
ft_con_id	unique identification of the file transfer connection.
src_filename	name of the file which shall be received.
tgt_filename	name of the file in the local filestore; default: if no filename is specified, the name of the file is the name of the remote file (src_filename).
contents_type	specifies the document type (unknown, FTAM-1, FTAM-2, FTAM-3, NBS-9) of a file.
read_password	password used to read the content of a file of the remote virtual filestore.
concurrency_control	identifies the access rights for each file operation (read, insert, replace, extend, erase, read-attribute, change-attribute, delete-file) at the beginning of a regime.
account	identifies the account which shall be in charge of the costs occurred during a regime.
charging	carries information about the costs occurred during a regime; the value of this parameter is a list of triples.
diagnostics	used to convey detailed information on the failure of a requested file transfer operation; the parameter diagnostics is a list of diagnostic information.

Table 41

Parameter	Request	Indication	Response	Confirmation
ft_con_id	m	m	m	m
src_filename	m	m	-	-
tgt_filename	o	-	-	o
contents_type	m	m	m	m
read_password	o	o	-	-
concurrency_control	o	o	-	-
account	o	o	-	-
charging	-	o	-	o
diagnostics	-	-	-	o

9.2.5.3 Delete file

Deleting a file in the remote VFS is mapped to the confirmed FTAM services F-Select and F-Delete.

The following parameters of the FTAM service primitives are not exchanged at the PCI:

- shared ASE information (used in conjunction with the CCR and actually deleted in ISO/IEC 8571 [9]; no use of this parameter known);

- requested access (there exists only one access requested for deleting a file: delete; used for selection of a file);

- state result (state of the FTAM protocol machine after execution of the function; not necessary at the PCI).

Table 42

Parameter	Purpose
ft_con_id	unique identification of the file transfer connection.
src_filename	name of the file to be deleted.
delete_password	password used to delete a file of the remote virtual filestore.
read_password	password used to read the content of a file of the remote virtual filestore.
contents_type	specifies the document type (unknown, FTAM-1, FTAM-2, FTAM-3, NBS-9) of a file.
concurrency_control	identifies the access rights for each file operation (read, insert, replace, extend, erase, read-attribute, change-attribute, delete-file) at the beginning of a regime.
account	identifies the account which shall be in charge of the costs occurred during a regime.
charging	carries information about the costs occurred during a regime; the value of this parameter is a list of triples.
diagnostics	used to convey detailed information on the failure of a requested file transfer operation; the parameter diagnostics is a list of diagnostic information.

Table 43

Parameter	Request	Indication	Response	Confirmation
ft_con_id	m	m	m	m
src_filename	m	m	-	-
delete_password	o	o	-	-
read_password (see note)	o	o	-	-
contents_type	o	o	-	-
concurrency_control	o	o	-	-
account	o	o	-	-
charging	-	o	-	o
diagnostics	-	-	-	o
NOTE: If the negotiated service class is the transfer-class the delete operation can be used only with the bulk transfer (F-Open, F-Read or F-Write, F-Close). Therefore this parameter shall be used only in that case.				

9.2.5.4 Rename file

Renaming a file in the remote VFS is mapped to the confirmed FTAM services F-Select, F-Change-Attrib and F-Deselect.

The following parameters of the FTAM service primitives are not exchanged at the PCI:

- shared ASE information (used in conjunction with the CCR and actually deleted in ISO/IEC 8571 [9]; no use of this parameter known);

- requested access (there exists only one access requested for renaming a file: change-attribute; used for selection of a file);

- state result (state of the FTAM protocol machine after execution of the function; not necessary at the PCI).

Table 44

Parameter	Purpose
ft_con_id	unique identification of the file transfer connection.
src_filename	name of the file to be renamed.
tgt_filename	new name of the file.
change_attribute_password	password used to change the attributes of a file of the remote virtual filestore.
concurrency_control	identifies the access rights for each file operation (read, insert, replace, extend, erase, read-attribute, change-attribute, delete-file) at the beginning of a regime.
account	identifies the account which shall be in charge of the costs occurred during a regime.
charging	carries information about the costs occurred during a regime; the value of this parameter is a list of triples.
diagnostics	used to convey detailed information on the failure of a requested file transfer operation; the parameter diagnostics is a list of diagnostic information.

Table 45

Parameter	Request	Indication	Response	Confirmation
ft_con_id	m	m	m	m
src_filename	m	m	-	-
tgt_filename	m	m	-	-
change_attribute_password	o	o	-	-
concurrency_control	o	o	-	-
account	o	o	-	-
charging	-	o	-	o
diagnostics	-	-	-	o

9.2.5.5 Cancel file operation

The cancel file operation abandons a file operation in progress at the PCI. This operation may cause the lost of data and information. The file transfer connection remains established. The cancel file operation is mapped to the confirmed FTAM service F-Cancel.

The following parameters of the FTAM service primitives are not exchanged at the PCI:

- shared ASE information (used in conjunction with the CCR and actually deleted in ISO/IEC 8571 [9]; no use of this parameter known).

Table 46

Parameter	Purpose
ft_con_id	unique identification of the file transfer connection.
IDU_ref	unique identification of the operation (via the correspondent IDU) which shall be cancelled.
diagnostics	used to convey detailed information on the failure of a requested file transfer operation; the parameter diagnostics is a list of diagnostic information.

Table 47

Parameter	Request	Indication	Confirmation
ft_con_id	m	m	m
IDU_ref	m	m	m
diagnostics	-	-	o

9.2.5.6 End of file operation

The end of file operation is used to indicate that a requested file operation from the remote user has been processed.

Table 48

Parameter	Purpose
ft_con_id	unique identification of the file transfer connection.

Table 49

Parameter	Indication
ft_con_id	m

9.2.5.7 Read file attributes

Reading the attributes of a file in the remote VFS is mapped to the confirmed FTAM services F-Select, F-Read-Attrib and F-Deselect.

The following parameters of the FTAM service primitives are not exchanged at the PCI:

- shared ASE information (used in conjunction with the CCR and actually deleted in ISO/IEC 8571 [9]; no use of this parameter known);

- requested access (there exists only one access requested for reading file attributes: read-attribute; used for selection of a file);

- state result (state of the FTAM protocol machine after execution of the function; not necessary at the PCI).

Table 50

Parameter	Purpose
ft_con_id	unique identification of the file transfer connection.
src_filename	name of the file which file attribute(s) shall be retrieved.
read_attribute_password	password used to read the attributes of a file of the remote virtual filestore.
concurrency_control	identifies the access rights for each file operation (read, insert, replace, extend, erase, read-attribute, change-attribute, delete-file) at the beginning of a regime.
account	identifies the account which shall be in charge of the costs occurred during a regime.
attribute_names	applicable for the attributes of those attribute groups negotiated for the file transfer connection; the attribute names specified with this parameter shall be those attributes of a file which values shall be read.
attribute_values	applicable for the attributes of those attribute groups negotiated for the file transfer connection; with this parameter actual attribute values of a file shall be retrieved.
charging	carries information about the costs occurred during a regime; the value of this parameter is a list of triples.
diagnostics	used to convey detailed information on the failure of a requested file transfer operation; the parameter diagnostics is a list of diagnostic information.

Table 51

Parameter	Request	Indication	Response	Confirmation
ft_con_id	m	m	m	m
src_filename	m	m	-	-
read_attribute_password	o	o	-	-
concurrency_control	o	o	-	-
account	o	o	-	-
attribute_names (see note)	o	o	-	-
attribute_values	-	-	-	m
charging	-	o	-	o
diagnostics	-	-	-	o
NOTE: If no file attributes are specified, the values of all attributes are requested.				

9.2.5.8 Change file attributes

Changing the attributes of a file in the remote VFS is mapped to the confirmed FTAM services F-Select, F-Change-Attrib and F-Deselect.

The following parameters of the FTAM service primitives are not exchanged at the PCI:

- shared ASE information (used in conjunction with the CCR and actually deleted in ISO/IEC 8571 [9]; no use of this parameter known);

- requested access (there exists only one access requested for changing file attributes: change-attribute);

- state result (state of the FTAM protocol machine after execution of the function; not necessary at the PCI).

Table 52

Parameter	Purpose
ft_con_id	unique identification of the file transfer connection.
src_filename	name of the file which file attribute(s) shall be to be retrieved.
change_attributes	applicable for the attributes of those attribute groups negotiated for the file transfer connection; the attribute values specified with this parameter shall be the new attribute values of a file on the remote virtual filestore.
change_attribute_password	password used to change the attributes of a file of the remote virtual filestore.
concurrency_control	identifies the access rights for each file operation (read, insert, replace, extend, erase, read-attribute, change-attribute, delete-file) at the beginning of a regime.
account	identifies the account which shall be in charge of the costs occurred during a regime.
charging	carries information about the costs occurred during a regime; the value of this parameter is a list of triples.
diagnostics	used to convey detailed information on the failure of a requested file transfer operation; the parameter diagnostics is a list of diagnostic information.

Table 53

Parameter	Request	Indication	Response	Confirmation
ft_con_id	m	m	m	m
src_filename	m	m	-	-
change_attributes	m	m	-	-
change_attribute_password	o	o	-	-
concurrency_control	o	o	-	-
account	o	o	-	-
charging	-	o	-	o
diagnostics	-	-	-	o

9.2.5.9 List remote filestore information

The list remote filestore directory operation is used to get the content of a remote VFS directory. This operation uses the FTAM document type NBS-9 to retrieve directory information from the remote filestore. The operation may fail if the remote VFS does not support the present document type. The list remote filestore directory operation is an implicit mapping to the confirmed FTAM service F-Select, F-Open, F-Close, F-Deselect and the unconfirmed service F-Read.

The following parameters of the FTAM service primitives are not exchanged at the PCI:

- shared ASE information (used in conjunction with the CCR and actually deleted in ISO/IEC 8571 [9]; no use of this parameter known);
- requested access (there exists only one access requested for receive a file: read; used for selection of a file);
- processing mode (there exists only one processing mode requested for receive a file: read; used for opening a file);
- contents type (the document carrying the necessary directory information is of type NBS-9; no other value possible);
- access context (for NBS-9 files the access context is UA);
- enable FADU locking (only used for opening a file; mandatory parameter which is set to false);
- FADU identity (mandatory parameter; the value of this parameter is "first" for the access context UA);
- FADU lock (outside the scope of Simple FTAM);
- state result (state of the FTAM protocol machine after execution of the function; not necessary at the PCI).

Table 54

Parameter	Purpose
ft_con_id	unique identification of the file transfer connection.
sort_key	identifies the key according to which the requested list shall be sorted; possible values: date(0), name(1), size(2).
dir_name	path to the directory for which a file list shall be retrieved.
attribute_names	applicable for the attributes of those attribute groups negotiated for the file transfer connection; the attribute names specified with this parameter shall be those attributes of a file which values shall be read.
read_password	password used to read the content of a file of the remote virtual filestore.
concurrency_control	identifies the access rights for each file operation (read, insert, replace, extend, erase, read-attribute, change-attribute, delete-file) at the beginning of a regime.
account	identifies the account which shall be in charge of the costs occurred during a regime.
file_list	used to retrieve the list of files and their attributes in a directory (see parameter *dir_name*) of the remote virtual filestore; if this parameter is used the list of files and their attributes shall be conveyed in a buffer; shall not be used if the parameter *filename* is used.
filename	used to retrieve the list of files and their attributes in a directory (see parameter *dir_name*) of the remote virtual filestore; if this parameter is used the list of files and their attributes shall be conveyed in the named file; shall not be used if the parameter *file_list* is used.
charging	carries information about the costs occurred during a regime; the value of this parameter is a list of triples.
diagnostics	used to convey detailed information on the failure of a requested file transfer operation; the parameter diagnostics is a list of diagnostic information.

Table 55

Parameter	Request	Indication	Response	Confirmation
ft_con_id	m	m	m	m
sort_key	m	-	-	-
dir_name	m	m	-	-
attribute_names	o	o	-	-
read_password	o	o	-	-
concurrency_control	o	o	-	-
account	o	o	-	-
file_list	-	-	-	c
filename	-	-	-	c
charging	-	o	-	o
diagnostics	-	-	-	o

9.2.5.10 Navigation on the remote filestore

The operation navigation on the remote filestore is used to change the current directory of the remote filestore. This operation is a mapping to the confirmed FTAM service F-Change-Prefix.

All parameters of the FTAM service primitives are used.

Table 56

Parameter	Purpose
ft_con_id	unique identification of the file transfer connection.
dir_name	path to the target directory.
reset	shall be used to navigate on the filestore in conjunction with the parameter dir_name; meaning: TRUE: reset the current directory to the directory specified at the creation of the FTAM regime and than append the value of the parameter dir_name FALSE: append the value of the parameter dir_name to the current directory.
filestore_password	password used to access the remote virtual filestore.
diagnostics	used to convey detailed information on the failure of a requested file transfer operation; the parameter diagnostics is a list of diagnostic information.

Table 57

Parameter	Request	Indication	Response	Confirmation
ft_con_id	m	m	m	m
dir_name	m	m	-	-
reset	o	o	-	-
filestore_password	o	o	-	-
diagnostics	-	-	-	o

9.2.6 Local PCI services

9.2.6.1 Redirection of incoming calls

The redirection of incoming calls operation is used to redirect an incoming to an LA which has been established a local communications channel to the CA.

Table 58

Parameter	Purpose
new_LA	unique identification of the LA which shall take the incoming call.
redirect_IDU	the IDU (i.e. the incoming call from a remote system) which shall be redirected.

Table 59

Parameter	Request	Confirmation
new_LA	m	-
redirect_IDU	m	-

9.2.6.2 List connected LAs

The list connected LAs operation is used to get the LAs which are locally connected to the CA.

Table 60

Parameter	Purpose
list_password	password used to check if an LA is allowed to get the list of LAs which are locally connected to the CA.
login_name_list	list the login names of those LAs which are locally connected to the CA.

Table 61

Parameter	Request	Confirmation
list_password	o	-
login_name_list	-	m

9.2.6.3 List local filestore information

The list local filestore directory operation is used to get the content of a directory of the local filestore.

Table 62

Parameter	Purpose
dir_name	path to the local directory for which a file list shall be retrieved.
attribute_names	applicable for the attributes of those attribute groups negotiated for the file transfer connection; the attribute names specified with this parameter shall be those attributes of a file which values shall be read.
sort_key	identifies the key according to which the requested list shall be sorted; possible values: date(0), name(1), size(2)
file_list	used to retrieve the list of files and their attributes in a directory (see parameter *dir_name*) of the local filestore; if this parameter is used the list of files and their attributes shall be conveyed in a buffer; shall not be used if the parameter *filename* is used.
filename	used to retrieve the list of files and their attributes in a directory (see parameter *dir_name*) of the local filestore; if this parameter is used the list of files and their attributes shall be conveyed in the named file; shall not be used if the parameter *file_list* is used.

Table 63

Parameter	Request	Confirmation
dir_name	m	o
attribute_names	m	-
sort_key	m	-
file_list	-	c
filename	-	c

9.2.6.4 Read CA capability

The read CA capabilities operation is used to get the features and services that the CA offers to the LAs. These product specific capabilities shall be provided by the CA manufacturer (see also clause 10).

Table 64

Parameter	Purpose
CA_capabilities	the product specific CA capabilities.

Table 65

Parameter	Request	Confirmation
CA_capabilities	-	m

9.2.6.5 Read CA configuration

Table 66

Parameter	Purpose
CA_config	list of configuration information of a SimpleFTAM CA.

Table 67

Parameter	Request	Confirmation
CA_config	-	m

9.2.6.6 Read phonebook entry

The operation read phonebook entry is used to get the address information of a specific local or remote user.

Table 68

Parameter	Purpose
search_key	unique identification to search a specific entry in the phonebook which shall be retrieved.
phonebook_entry	requested entry of the phonebook.

Table 69

Parameter	Request	Confirmation
search_key	m	o
phonebook_entry	-	m

9.2.6.7 Read logbook

The operation read logbook is used to get logbook information maintained by the service provider (CA).

Table 70

Parameter	Purpose
selection_criteria	unique identification to select a part of the logbook which shall be retrieved (e.g. date, login_name, ca_id).
logbook_info	requested information of the logbook; if this parameter is used the logbook information shall be conveyed in a buffer; shall not be used if the parameter *filename* is used.
filename	used to retrieve the information of the logbook; if this parameter is used the logbook information shall be conveyed in the named file; shall not be used if the parameter *logbook_info* is used.

Table 71

Parameter	Request	Confirmation
selection_criteria	m	o
logbook_info	-	c
filename	-	c

9.2.6.8 Read access control information

The operation read access control information is used to get the access control information of a file in the local filestore.

Table 72

Parameter	Purpose
dir_name	path to the directory of the local filestore containing the file which access control list which shall be retrieved.
src_filename	name of the file which access control list shall be retrieved.
access_control_info	used to retrieve the actual values of the attribute access_control of a file of the local virtual filestore.

Table 73

Parameter	Request	Confirmation
dir_name	m	-
src_filename	m	-
access_control_info		m

9.2.7 SimpleFTAM profile specification

Two profiles are specified for SimpleFTAM:

- the Basic SimpleFTAM profile: a CA which claims to be basically conformant to SimpleFTAM shall support the Basic SimpleFTAM profile as specified in this subclause;

- the Enhanced SimpleFTAM profile: a CA which claims to be fully conformant to SimpleFTAM shall support the Enhanced SimpleFTAM profile as specified in this subclause.

A CA which claims conformance to this ETS shall support at least one of the SimpleFTAM profiles. The Enhanced SimpleFTAM profile is a superset of the Basic SimpleFTAM profile. This implies that the support of the Basic SimpleFTAM profile is mandatory for all CAs claiming to be conformant to this ETS.

The specification of the SimpleFTAM profiles defines:

- the IDUs which shall be supported;
- the parameters which shall be supported.

Table 74 contains the IDUs which shall be supported by a CA claiming to be conformant to the Basic SimpleFTAM profile or to the Enhanced SimpleFTAM profile:

Table 74

Network connection services	IDUs	Basic SimpleFTAM	Enhanced SimpleFTAM	Restrictions and conditions
establishment of the network connection	ConReq ConConf ConInd ConResp	m m m m	m m m m	
release of the network connection	DiscReq DiscConf DiscInd	m m m	m m m	
send transparent data	SendDataReq SendDataInd	- -	- -	not applicable for SimpleFTAM. not applicable for SimpleFTAM.
establishment of a file transfer connection	EstabReq EstabConf EstabInd EstabResp	m m m m	m m m m	one step procedure for the establishment of a file transfer connection.
termination of a file transfer connection	RelReq RelConf RelInd RelResp	m m m m	m m m m	one step procedure for the termination of a file transfer connection.
abort of a file transfer connection	AbortReq AbortConf AbortInd	m m m	m m m	
establishment of a file transfer association	AssocReq AssocConf AssocInd AssocResp	- - - -	- - - -	not applicable for SimpleFTAM. not applicable for SimpleFTAM. not applicable for SimpleFTAM. not applicable for SimpleFTAM.
establishment of a file transfer access	AccessReq AccessConf AccessInd AccessResp	- - - -	- - - -	not applicable for SimpleFTAM. not applicable for SimpleFTAM. not applicable for SimpleFTAM. not applicable for SimpleFTAM.
termination of a file transfer access	EndAccessReq EndAccessConf EndAccessInd EndAccessResp	- - - -	- - - -	not applicable for SimpleFTAM. not applicable for SimpleFTAM. not applicable for SimpleFTAM. not applicable for SimpleFTAM.
termination of a file transfer association	TerminateReq TerminateConf TerminateInd TerminateResp	- - - -	- - - -	not applicable for SimpleFTAM. not applicable for SimpleFTAM. not applicable for SimpleFTAM. not applicable for SimpleFTAM.

Table 74 (continued)

File transfer services	IDUs	Basic SimpleFTAM	Enhanced SimpleFTAM	Restrictions and conditions
send file	SendReq	m	m	
	SendConf	m	m	
	SendInd	m	m	
	SendResp	m	m	
send message	SendMsgReq	-	-	not applicable for SimpleFTAM.
	SendMsgConf	-	-	not applicable for SimpleFTAM.
	SendMsgInd	-	-	not applicable for SimpleFTAM.
receive file	ReceiveReq	m	m	
	ReceiveConf	m	m	
	ReceiveInd	m	m	
	ReceiveResp	m	m	
delete file	DelReq	o	m	
	DelConf	o	m	
	DelInd	o	m	
	DelResp	o	m	
rename file	RenameReq	o	m	
	RenameConf	o	m	
	RenameInd	o	m	
	RenameResp	o	m	
cancel a file operation	CancelReq	m	m	
	CancelConf	m	m	
	CancelInd	m	m	
end of file operation	EndOperationInd	m	m	
read file attributes	ReadAttribReq	o	m	
	ReadAttribConf	o	m	
	ReadAttribInd	o	m	
	ReadAttribResp	o	m	
change file attributes	ChangeAttribReq	o	m	
	ChangeAttribConf	o	m	
	ChangeAttribInd	o	m	
	ChangeAttribResp	o	m	

(continued)

Table 74 (concluded)

File transfer services	IDUs	Basic SimpleFTAM	Enhanced SimpleFTAM	Restrictions and conditions
list remote filestore information	ListReq ListConf ListInd ListResp	o o o o	m m m m	
navigation on the remote filestore	NavigateReq NavigateConf NavigateInd NavigateResp	o o o o	m m m m	
Local PCI services	**IDUs**	**Basic SimpleFTAM**	**Enhanced SimpleFTAM**	**Restrictions and conditions**
redirection of incoming calls	RedirectReq RedirectConf	o o	o o	
list connected LAs	ListLAReq ListLAConf	o o	o o	
list local filestore information	LocListReq LocListConf	o o	o o	
read CA capabilities	CapabilityReq CapabilityConf	o o	o o	
read CA configuration	ConfigReq ConfigConf	o o	o o	
read phonebook entry	PhonebookReq PhonebookConf	o o	o o	
read logbook	LogbookReq LogbookConf	o o	o o	
read access control information	AccessControlReq AccessControlConf	o o	o o	

Table 75 contains the parameters which shall be supported by a CA claiming to be conformant to the Basic SimpleFTAM profile:

Table 75

Parameter	Related IDUs	Basic SimpleFTAM	Enhanced SimpleFTAM	Comment
access_control_info	AccessControlConf	o	o	shall be supported only if the local PCI service *read access control information* is supported by the CA.
account	EstabReq, EstabInd SendReq, SendInd ReceiveReq, ReceiveInd DelReq, DelInd RenameReq, RenameInd ReadAttribReq, ReadAttribInd ChangeAttribReq, ChangeAttribInd ListReq, ListInd	m	m	
attribute_groups	EstabReq, EstabInd, EstabConf, EstabResp	m	m	storage group: optional for the Basic SimpleFTAM profile, mandatory for the Enhanced SimpleFTAM profile; security group: shall not be used in conjunction with the functional unit Enhanced Management; private group: shall not be used for SimpleFTAM.
attribute_names	ReadAttribReq, ReadAttribInd ListReq, ListInd LocListReq	o	m	private group attributes: shall not be used for SimpleFTAM.
attribute_values	ReadAttribConf	o	m	private group attributes: shall not be used for SimpleFTAM.
CA_capabilities	CapabilityConf	o	o	
CA_config	ConfigConf	o	o	shall be supported only if the local PCI service *read CA configuration* is supported by the CA.
called_addr	ConReq, ConInd	m	m	
called_application_title	EstabReq, EstabInd	m	m	form 2 shall be used: AP-Title: Object Identifier; AE-Qualifier: Integer.
called_name	ConReq, ConInd EstabReq, EstabInd	m	m	shall be used only if the phonebook feature is supported by a CA.
called_presentation_addr	EstabReq, EstabInd	m	m	
called_subaddr	ConReq, ConInd	o	m	
calling_addr	ConReq, ConInd	m	m	

(continued)

Table 75 (continued)

Parameter	Related IDUs	Basic SimpleFTAM	Enhanced SimpleFTAM	Comment
calling_subaddr	ConReq, ConInd	o	m	
change_attribute_password	RenameReq, RenameInd ChangeAttribReq, ChangeAttribInd	o	m	private group attributes: shall not be used for SimpleFTAM.
change_attributes	ChangeAttribReq, ChangeAttribInd	o	m	private group attributes: shall not be used for SimpleFTAM.
charging	RelConf, RelInd SendConf, SendInd ReceiveConf, ReceiveInd DelConf, DelInd RenameConf, RenameInd ReadAttribConf, ReadAttribInd ChangeAttribConf, ChangeAttribInd ListConf, ListInd	m	m	
concurrency_control	SendReq, SendInd ReceiveReq, ReceiveInd DelReq, DelInd RenameReq, RenameInd ReadAttribReq, ReadAttribInd ChangeAttribReq, ChangeAttribInd ListReq, ListInd	m	m	
contents_type	SendReq, SendConf, SendInd, SendResp ReceiveReq, ReceiveConf, ReceiveInd, ReceiveResp DelReq, DelInd	m	m	FTAM-1: mandatory for the Basic SimpleFTAM and Enhanced SimpleFTAM profile; FTAM-2: optional for the Basic SimpleFTAM and Enhanced SimpleFTAM profile; FTAM-3: mandatory for the Basic SimpleFTAM and Enhanced SimpleFTAM profile; NBS-9: optional for the Basic SimpleFTAM and mandatory for the Enhanced SimpleFTAM profile.
contents_type_list	EstabReq, EstabInd, EstabConf, EstabResp	m	m	
create_password	SendReq, SendInd	m	m	
delete_password	DelReq, DelInd	o	m	

(continued)

Table 75 (continued)

Parameter	Related IDUs	Basic SimpleFTAM	Enhanced SimpleFTAM	Comment
diagnostics	EstabConf AbortConf SendConf ReceiveConf DelConf RenameConf CancelConf ReadAttribConf ChangeAttribConf ListConf NavigateConf	m	m	
dir_name	ListReq, ListInd NavigateReq, NavigateInd AccessControlReq LocListReq	m	m	
file_list	ListConf LocListConf	o	m	
filename	ListConf LocListConf LogbookConf	o	m	
filestore_password	EstabReq, EstabInd NavigateReq, NavigateInd	m	m	mandatory for the initiator of a file transfer connection, optional for the responder of a file transfer connection.

(continued)

Table 75 (continued)

Parameter	Related IDUs	Basic SimpleFTAM	Enhanced SimpleFTAM	Comment
ft_con_id	EstabReq, EstabInd, EstabResp, EstabConf RelReq, RelInd, RelResp, RelConf AbortReq, AbortInd, AbortConf SendReq, SendInd, SendResp, SendConf ReceiveReq ReceiveInd, ReceiveResp, ReceiveConf DelReq, DelInd, DelResp, DelConf RenameReq, RenameInd, RenameResp, RenameConf CancelReq, CancelInd, CancelConf EndOperationInd ReadAttribReq, ReadAttribInd, ReadAttribResp, ReadAttribConf ChangeAttribReq, ChangeAttribInd, ChangeAttribResp, ChangeAttribConf ListReq, ListInd, ListResp, ListConf NavigateReq, NavigateInd, NavigateResp, NavigateConf	m	m	
FU	EstabReq, EstabInd, EstabConf, EstabResp	m	m	kernel: mandatory for both profiles; read: mandatory for both profiles; write: mandatory for both profiles; file access: not applicable for SimpleFTAM; limited file management: mandatory for both profiles; enhanced file management: optional for the Basic SimpleFTAM profile, mandatory for the Enhanced SimpleFTAM profile; grouping: mandatory for both profiles; FADU locking: not applicable for SimpleFTAM; recovery: optional for the Basic SimpleFTAM profile, mandatory for the Enhanced SimpleFTAM profile; restart: optional for both profiles.
IDU_ref	CancelReq, CancelInd, CancelConf	m	m	
initial_attributes	SendReq, SendConf, SendInd, SendResp	m	m	private group attributes: shall not be used for SimpleFTAM.

(continued)

Table 75 (continued)

Parameter	Related IDUs	Basic SimpleFTAM	Enhanced SimpleFTAM	Comment
initiator_id	EstabReq, EstabInd	m	m	
login_name_list	ListLAConf	o	o	
list_password	ListLAReq	o	o	
logbook_info	LogbookConf	o	o	
net_charge	DiscInd, DiscConf	m	m	
net_diag	ConConf DiscInd, DiscConf	m	m	
net_param	ConReq, ConInd DiscInd, DiscConf	m	m	
net_reason	ConConf DiscInd, DiscConf	m	m	
new_LA	RedirectReq	o	o	
override	SendReq, SendInd	m	m	
phonebook_entry	PhonebookConf	o	o	
QoS	EstabReq, EstabInd, EstabConf, EstabResp	m	m	
read_attribute_password	ReadAttribReq, ReadAttribInd	o	m	
read_password	ReceiveReq, ReceiveInd DelReq, DelInd ListReq, ListInd	m	m	
redirect_IDU	RedirectReq	o	o	
requested_access	SendReq, SendInd	m	m	shall have one of the following values: read(0), insert(1), replace(2), extend(3), read-attribute(5), change-attribute(6), delete-file(7).
reset	NavigateReq, NavigateInd	m	m	
responding_addr	ConResp, ConConf	m	m	
responding_subaddr	ConResp, ConConf	m	m	
search_key	PhonebookReq, PhonebookConf	o	o	
selection_criteria	LogbookReq, LogbookConf	o	o	
service_class	EstabReq, EstabInd, EstabConf, EstabResp	m	m	transfer-class: mandatory for the Basic SimpleFTAM and Enhanced SimpleFTAM profile; transfer-and-management-class: optional for the Basic SimpleFTAM profile and mandatory for the Enhanced SimpleFTAM profile.

(continued)

Table 75 (concluded)

Parameter	Related IDUs	Basic SimpleFTAM	Enhanced SimpleFTAM	Comment
sort_key	ListReq LocListReq	o	m	
src_effect	SendReq	m	m	
src_filename	SendReq, SendInd ReceiveReq, ReceiveInd DelReq, DelInd RenameReq, RenameInd ReadAttribReq, ReadAttribInd ChangeAttribReq, ChangeAttribInd AccessControlReq	m	m	
tgt_filename	SendReq, SendConf ReceiveReq, ReceiveConf RenameReq, RenameInd	m	m	
uuid	ConReq, ConInd	m	m	
write_password	SendReq, SendInd	m	m	

9.3 Easyfile Service definition

This clause specifies the IDUs specific to Easyfile and the parameters which are used only for Easyfile.

Easyfile is specified in accordance with:

- ETS 300 409 [6];
- ETS 300 383 [4];
- ETS 300 075 [1];
- ETS 300 079 [2].

In order to take in account various implementations, two profiles are specified for Easyfile:

- a basic profile providing a simple access to the ETS 300 383 [4] file transfer;
- an enhanced profile providing a more complete access and control of the protocol.

9.3.1 Easyfile IDUs

The Easyfile service make use of specific IDUs to have a complete access of the Easyfile protocol.

Those IDUs are:

- Transparent data: the transparent data service allow the LA to send data on an establish connection outside the file transfer connection.

- Establishment of a file transfer association: the establishment of a file transfer association service.

- Termination of a file transfer association: the release of a file transfer association service.

- Establishment of a file access: the establishment of a file access service;

- Release of a file transfer access: the release of a file access service.

To cover the Easyfile file transfer protocol the following IDUs are used:

Table 76

Service Type	Purpose
establishment of a network connection	establish the communication link with a remote entity.
release of a network connection	release the communication link with a remote entity.
transparent data	transparent flow of data outside the file transfer regime. This allow the use of ETS 300 079 [2] commands outside Easyfile.
establishment of a file transfer connection	establish the Easyfile transfer regime (association and access regime).
termination of a file transfer connection	release the Easyfile transfer regime.
establishment of a file transfer association	establish the Easyfile association regime.
termination of a file transfer association	release the Easyfile association regime.
establishment of a file transfer access	establish the Easyfile access regime.
termination of a file transfer access	release the Easyfile access regime.
abort of a file transfer connection	abort the Easyfile association regime.
send file	send a file to the remote entity.
send message	send a message to the remote entity.
receive file	receive a file from the remote entity.
delete file	delete a file on the remote filesystem.
rename file	rename a file on the remote filesystem.
cancel a file operation	cancel a send or receive file operation.
end of file operation	indication of the end of a file transfer operation for the slave.
list remote filestore information's	list the remote files or the remote filestores of the filesystem.
navigation on the remote filestore	select a remote filestore.

9.3.2 Easyfile specific IDU parameters

The Easyfile specific IDU parameters which shall be used at the PCI are a subset of the ETS 300 383 [4] parameters. Additionally parameters which are related to the network ISDN and parameters which are introduced for the specific needs of the PCI as specified in this ETS are described in the following.

Table 77 summarizes the Easyfile specific IDU parameters and their meaning.

Table 77

Address information related parameters		
Parameter	**Abbreviation**	**Purpose**
called address	called_addr	network address, i.e. ISDN address, called by the initiator of the network connection; if the phonebook facility is supported by the CA the called_address may be also a human readable name of the called entity.
called subaddress	called_subaddr	network subaddress, i.e. ISDN network subaddress (MSN), called by the initiator of the network connection.
calling address	calling_addr	network address, i.e. ISDN address, of the initiator of a network connection; if the phonebook facility is supported by the CA the calling_address may be also a human readable name of the initiator of a network connection.
calling subaddress	calling_subaddr	network subaddress, i.e. ISDN network subaddress (MSN), of the initiator of a network connection.
responding address	responding_addr	network address, i.e. ISDN address, of the responding entity; if the phonebook facility is supported by the CA the responding _addr may be also a human readable name of the responding entity.
responding subaddress	responding_subaddr	network subaddress, i.e. ISDN network subaddress (MSN).
called name	called_name	unique identification of the called communication partner (alias name); this parameter shall be used only if the phonebook feature is supported by a CA.
ISDN related parameters		
Parameter	**Abbreviation**	**Purpose**
network reason	net_reason	reason for rejection of a network connection service on the D-channel (see cause according to ITU-T RecommendationQ.931 [11]).
network diagnostic	net_diag	additional information to the network reason. This parameter may provide additional cause and diagnostic for the release of the communication on the B-channel.
network charging	net_charge	network charging information.
network parameters	net_param	this parameter is used by the CA to provide additional network parameters related to the remote communication partner.
user to user identification	uuid	User to user identification to identify an application. This allow the CA to have an LA selection mechanism for incoming calls based on this parameter.
Protocol related parameters		
Parameter	**Abbreviation**	**Purpose**
SBV user data	sbv_data	transparent data to be transmitted to the remote entity.
application name	application_name	name of the application (see ETS 300 075 [1]).
calling address	ft_calling_addr	address of the entity which originated the file transfer connection.
called address	ft_called_addr	address of the entity which has sent a response.
time-outs	timeouts	response time-out (see ETS 300 075 [1]).
identification key	identifier	name and password (see ETS 300 383 [4]).
caller identification	calling_identifier	name and password (see ETS 300 383 [4]).
called identification	called_identifier	name and password (see ETS 300 383 [4]).
request identification	ident_req	requirement for the identification of the remote entity.
user data associate	user_data	private data in T-Associate.
reason associate	reason	T-associate result parameter conform to ETS 300 075 [1].
user data release	user_data	user data for T-Release.
result release	result	to convey the T-Release result.
role	role	master or slave role.
functions	functions	functions supported.
recovery	recovery	ability to support the recovery mechanism.
user data access	user_data	user data for the T-Access conform to ETS 300 075 [1].
domain supported	domain_support	supported primitives for each domain (A, B, C) for the slave.

(continued)

Table 77 (continued)

Protocol related parameters		
Parameter	**Abbreviation**	**Purpose**
navigation	navigation	ability to support navigation.
compression list	compression_list	list of supported compression modes.
user data end access	user_data	parameter conform to ETS 300 075 [1].
result end access	result	to convey the result of T-End-Access.
file designation	designation	transfer name.
recovery use	recovery_use	indicate the use of the recovery.
user data transfer	user_data	parameter conform to ETS 300 075 [1].
domain selection	domain	group of files (A, B, C).
fcs32	fcs32	32 bits FCS.
file header	header	file header conform to ETS 300 075 [1].
filename	filename	filename associated with the transfer name.
destination name	destination_name	destination filename.
compression mode	compression_mode	compression mode used.
user field	user_field	comments related to the file.
local filename	local_filename	local filename of the transfer name to be received.
result transfer	result	response: indicate acceptance of the file transfer; confirmation: indicate end of the file transfer operation.
new name	new_name	new name.
reason cancel	reason	user reason of the abort.
list designation	designation	criteria to select a list.
list mode	list_mode	the type of list files provided by the remote system: - simple list of files; - extended list of files; - list of filestores; - sublist of filestores.
sort key	sort_key	sorting method required for the file list (date, name, size).
filename (Recipient)	local_filename	the local file recipient of the list of files.
filestore	filestore	remote filestore name.
result navigation	result	result of the operation.
user data message	message	message to be transmitted to the remote entity.
Management related parameters		
Parameter	**Abbreviation**	**Purpose**
logbook information (see note 3)	logbook_info	part of the logbook which is maintained by a CA.
list password	list_password	password used to check if an LA is allowed to get the list of LAs which are locally connected to the CA.
phonebook entry (see note 4)	phonebook_entry	collection of information (e.g. network address) of the different LAs and remote systems or remote communication partners.
search key	search_key	unique identification to search a specific entry in the phonebook or in the access control list which shall be retrieved.
selection criteria	selection_criteria	unique identification to select a part of the logbook which shall be retrieved.
CA configuration (see note 2)	CA_config	list of configuration information of a Easyfile CA.
CA capabilities (see note 1)	CA_capabilities	information about the capabilities of a Easyfile CA.
authorizations granted to the caller	authorizations	list of services granted to the caller. This list comprise the following services: load, save, list and optionally delete and rename.
working area	working_area	workspace comprising all the files accessible to the caller.
support of navigation service	navigation_support	indication if the service is offered to the caller.
navigation filestores links	navigation_links	if necessary, indication of the logical links, which may exist, between the navigation filestores and the mandatory working area.

(continued)

Table 77 (concluded)

PCI specific parameters		
Parameter	**Abbreviation**	**Purpose**
file transfer connection identifier	ft_con_id	unique identification of the file transfer connection.
network connection type	netcon_type	type of the network connection which shall be used; for the time being there is only one valid network selectable at the PCI (ISDN).
CA identification	ca_id	unique identification of a CA (see also subclause 7.2).
list of LA login names	login_name_list	list login names of those LAs which are locally connected to the CA.
redirect IDU	redirect_IDU	the IDU, carrying the incoming data from a remote system, which shall be redirected to another actually logged in LA.
new LA login name	new_LA	login name of the LA to which the incoming data shall be redirected.
source effect	src_effect	action which shall be performed after transmission of a local file: delete the file after successful transmission (move) or store the file on the local virtual filestore (copy).

NOTE 1: Content, structure and data format of the parameter CA_capab is product dependent. Shall be specified by the service provider (CA).

NOTE 2: The parameter CA_config is a collection of the facilities offered by a CA. Shall be specified by the service provider (CA).

NOTE 3: Content, structure and data format of the parameter logbook_info is product dependent. Shall be specified by the service provider (CA).

NOTE 4: Content, structure and data format of the parameter phonebook_entry is product dependent. Shall be specified by the service provider (CA).

9.3.3 Network connection services

9.3.3.1 Establishment of a network connection

The establishment of the network connection make use of the SBV_Establish of ETS 300 079 [2].

Easyfile may use the following specific parameters.

Table 78

Parameters	Purpose
calling_addr	network address, i.e. ISDN address, or a human readable name of the initiator of a network connection.
calling_subaddr	network subaddress, i.e. ISDN network subaddress (MSN).
called_addr	network address, i.e. ISDN address, or a human readable name called by the initiator of the network connection.
called_subaddr	network subaddress, i.e. ISDN network subaddress (MSN).
responding_addr	network address, i.e. ISDN address, or a human readable name called by the initiator of the responding entity.
responding_subaddr	network subaddress, i.e. ISDN network subaddress (MSN).
netcon_type	type of the network connection which shall be used; for the time being there is only one valid network selectable at the PCI (ISDN).
net_param	this parameter is used by the CA to provide additional network parameters related to the remote communication partner.
called_name	unique identification of the called communication partner (alias name); this parameter shall be used only if the phonebook feature is supported by a CA. Other PCI parameters have priority on the called_name associated parameters.
uuid	User to user identification to identify an application. This allow the CA to have an LA selection mechanism for incoming calls based on this parameter.

Table 79 shows the use of the parameters related to the service primitives:

Table 79

Parameter	Request	Indication	Response	Confirmation
calling_addr	o	o	-	-
calling_subaddr	o	o	-	-
called_addr	m	o	-	-
called_subaddr	o	o	-	-
responding_addr	-	-	o	o
responding_subaddr	-	-	o	o
netcon_type	m	m	-	-
net_param	o	o	-	-
called_name	o	-	-	-
uuid	o	o	-	-

9.3.3.2 Release of a network connection

The release of a network connection make use of the SBV_Release of ETS 300 079 [2].

Table 80

Parameter	Purpose
net_charge	network charging information.
net_reason	additional information from ISDN after progression of the requested network connection release.
net_diag	additional information from ISDN B channel after progression of the requested network connection release.

Table 81 shows the use of the parameters related to the service primitives:

Table 81

Parameter	Request	Indication	Confirmation
net_charge	-	o	o
net_reason	-	o	o
net_diag	-	o	o

9.3.3.3 Transparent data

This specific IDU is used to send data to the remote entity prior to the file transfer connection.

Table 82

Parameters	Purpose
sbv_data	data to be transmitted to the remote entity directly.

Table 83 shows the use of the parameters related to the service primitives:

Table 83

Parameter	Request	Indication
sbv_data	m	m

9.3.4 File transfer connection services

9.3.4.1 One step procedure

9.3.4.1.1 Establishment of a file transfer connection

This IDU is used to establish the Easyfile transfer regime with the remote entity and is mapped to the SBV_TPD_Begin, T-Associate and T-Access.

Table 84

Parameter	Purpose
ft_con_id	unique identification of the file transfer connection.
calling_identifier	name and password (see ETS 300 075 [1]).
called_identifier	name and password (see ETS 300 075 [1]).
user_data	private data of T-Associate.
functions	functions supported (send, receive, list, rename, delete, message).
recovery	ability to support the recovery mechanism.
navigation	ability to support navigation.
compression_list	list of supported compression modes (see capabilities list in ETS 300 383 [4]).
reason	result parameter conform to ETS 300 075 [1].

Table 85 shows the use of the parameters related to the service primitives:

Table 85

Parameter	Request	Indication	Response	Confirmation
ft_con_id	m	m	m	m
calling_identifier	o	o	-	-
called_identifier (see note)	-	-	c	c
user_data	o	o	o	-
reason	-	-	o	o
functions	o	o	o	o
recovery	o	o	o	o
navigation	o	o	o	o
compression_list	o	o	o	o
NOTE: called_identifier may be present if requested.				

9.3.4.1.2 Termination of a file transfer connection

This IDU is used to end the transfer regime and is mapped to the T-End-Access, T-Release and SBV_TPD_End.

Table 86

Parameters	Purpose
ft_con_id	unique identification of the file transfer connection.
user data	user data for T-End-Access.
result	to convey the result.

Table 87 shows the use of the parameters related to the service primitives:

Table 87

Parameter	Request	Indication	Response	Confirmation
ft_con_id	m	m	m	m
user_data	o	o	-	-
result	-	-	m	m

9.3.4.2 Two steps procedure

9.3.4.2.1 Establishment of a file transfer association

This IDU is used to establish the Easyfile association regime with the remote entity and is mapped to the T-Associate of ETS 300 383 [4] and SBV_TPD_Begin of ETS 300 079 [2].

This command help to identify the application towards the remote entity.

The following parameters are not exchanged at the PCI:

- service class (this parameter is used to indicate the service class in use and is set to symmetrical service for Easyfile);

- explicit confirmation (this parameter is set to indicate that confirmation is requested).

Table 88

Parameter	Purpose
ft_con_id	unique identification of the file transfer connection.
application_name	name of the application (see ETS 300 075 [1]), set to !K for Easyfile.
ft_calling_addr	address of the entity which originated the file transfer connection.
timeouts	response time-out in seconds (see ETS 300 075 [1]).
calling_identifier	name and password (see ETS 300 075 [1]).
called_identifier	name and password (see ETS 300 075 [1]).
ident_req	requirement for the identification of the remote entity.
user_data	private data.
ft_called_addr	address of the entity which has sent a response.
reason	result parameter conform to ETS 300 075 [1].

Table 89 shows the use of the parameters related to the service primitives:

Table 89

Parameter	Request	Indication	Response	Confirmation
ft_con_id	m	m	m	m
application_name (see note)	o	o	-	-
ft_calling_addr	o	o	-	-
timeouts	o	o	o	-
calling_identifier	o	o	-	-
called_identifier	-	-	c	c
ident_req	o	o	-	-
user_data	o	o	o	o
ft_called_addr	o	o	o	o
reason	-	-	o	o
NOTE: Other values than "!K" may be implement by the application in a fallback mode (see ETS 300 075 [1]).				

9.3.4.2.2 Establish file transfer access

This specific IDU is used to establish the file transfer Access regime with the remote entity and is mapped to the T-Access.

The parameters help to provide the selection of Easyfile services towards the remote.

The following parameters are not exchanged at the PCI:

- transfer unit size (this parameter is used to indicate the maximum size of the application data contained in a file transfer block);

- anticipation window (this parameter is used to indicate the maximum number of consecutive block received before issuing an answer);

- transfer mode (this parameter shall not be used for Easyfile).

Table 90

Parameters		Purpose
ft_con_id		unique identification of the file transfer connection.
role		master or slave role.
functions		received functions supported (send, receive, list, rename, delete, message).
recovery		ability to support the recovery mechanism.
user_data [1]		conform to ETS 300 075 [1].
	domain_support [2]	groups supported for each function (A, B, C).
	navigation [2]	ability to support navigation.
	compression_list [2]	list of supported compression modes (see capabilities list, in ETS 300 383 [4]).
reason		result parameter conform to ETS 300 075 [1].
NOTE 1: This parameter is exclusive with [2].		
NOTE 2: This parameter is exclusive with [1].		

Table 91 shows the use of the parameters related to the service primitives:

Table 91

Parameter		Request	Indication	Response	Confirmation
ft_con_id		m	m	m	m
role		o	o	-	-
functions		o	o	o	o
recovery		o	o	o	o
user_data		o	o	o	o
	domain_support	o	o	o	o
	navigation	o	o	o	o
	compression_list	o	o	o	o
reason		-	-	o	o

9.3.4.2.3 Release file transfer access

This specific IDU is used to end the access regime and is mapped to the T-End-Access.

Table 92

Parameters	Purpose
ft_con_id	unique identification of the file transfer connection.
user_data	parameter conform to ETS 300 075 [1].
result	to convey the result of T-End-Access.

Table 93 shows the use of the parameters related to the service primitives:

Table 93

Parameter	Request	Indication	Response	Confirmation
ft_con_id	m	m	m	m
user_data	o	o	o	o
result	-	-	m	m

9.3.4.2.4 Termination of a file transfer association

This IDU is used to end the association regime and is mapped to the T-Release and SBV_TPD_End.

Table 94

Parameters	Purpose
ft_con_id	unique identification of the file transfer connection.
user data	user data for T-Release.
result	to convey the T-Release result.

Table 95 shows the use of the parameters related to the service primitives:

Table 95

Parameter	Request	Indication	Response	Confirmation
ft_con_id	m	m	m	m
user_data	o	o	o	o
result	-	-	m	m

9.3.4.3 Abort a file transfer connection

This IDU is used to abort all the currently file transfer established regimes and is mapped to the T-U-Abort and T-P-Abort.

Table 96

Parameters	Purpose
ft_con_id	unique identification of the file transfer connection.
reason	to convey the reason of the abort.

Table 97 shows the use of the parameters related to the service primitives:

Table 97

Parameter	Request	Indication	Confirmation
ft_con_id	m	m	m
reason	m	m	-

9.3.5 File transfer services

9.3.5.1 Send file

This IDU is used to send a file to the remote entity and the parameters are mapped to the T-Save and to the file header structure.

The following parameters are not exchanged at the PCI:

- Recovery point (this parameter is used to indicate the recovery point).

Table 98

Parameters		Purpose
ft_con_id		unique identification of the file transfer connection.
designation		transfer name.
recovery_use		indicate the use of the recovery.
user_data [1]		parameter conform to ETS 300 075 [1].
	domain [2]	group of files (A, B, C) applicable for the command.
	fcs32 [2]	32 bits FCS.
header [3]		file header conform to ETS 300 075 [1].
	filename [4]	filename may be associated with the transfer name.
	destination_name [4]	destination indicating a drive, device and/or directory name.
	compression_mode [4]	compression mode used.
	user_field [4]	comments related to the file or to the application.
src_effect		delete the file of the local filestore after successful transmission(move) or not (copy).
result		response: indicate acceptance of the file transfer; confirmation: indicate end of the file transfer operation.
NOTE 1: This parameter is exclusive with [2]. NOTE 2: This parameter is exclusive with [1]. NOTE 3: This parameter is exclusive with [4]. NOTE 4: This parameter is exclusive with [3].		

Table 99 shows the use of the parameters related to the service primitives:

Table 99

Parameter		Request	Indication	Response	Confirmation
ft_con_id		m	m	m	m
designation		m	m	-	-
recovery_use		o	o	-	-
user_data		o	o	o	o
	domain	o	o	-	-
	fcs32	o	o	-	-
header		o	-	-	-
	filename	o	-	-	-
	destination_name	o	-	-	-
	compression_mode	o	-	-	-
	user_field	o	-	-	-
src_effect		o	-	-	-
result		-	-	m	m

9.3.5.2 Message

This IDU is used to send a message to the remote entity and mapped to T-typed-data.

This command is confirmed locally by the CA.

Table 100

Parameters	Purpose
ft_con_id	unique identification of the file transfer connection.
message	message to be transmitted to the remote entity.

Table 101 shows the use of the parameters related to the service primitives:

Table 101

Parameter	Request	Indication	Confirmation
ft_con_id	m	m	m
message	m	m	-

9.3.5.3 Receive file

This IDU is used to receive a file from the remote entity and the parameters are mapped to the T-Load and to the file header structure.

The following parameters are not exchanged at the PCI:

- Recovery point (this parameter is used to indicate the recovery point).

Table 102

Parameters		Purpose
ft_con_id		unique identification of the file transfer connection.
designation		transfer name to be received.
recovery_use		indicate the use of the recovery.
user_data [1]		parameter conform to ETS 300 075 [1].
	domain [2]	group of files (A, B, C) applicable for the command.
	fcs32 [2]	32 bits FCS may be used in conjunction with the recovery.
header [3]		file header conform to ETS 300 075 [1].
	filename [4]	filename associated with the transfer name.
	destination_name [4]	destination name.
	compression_mode [4]	compression mode used.
	user_field [4]	comments related to the file.
local_filename		local filename of the transfer name to be received.
result		response: indicate acceptance of the file transfer; confirmation: indicate end of the file transfer operation.
NOTE 1: This parameter is exclusive with [2]. NOTE 2: This parameter is exclusive with [1]. NOTE 3: This parameter is exclusive with [4]. NOTE 4: This parameter is exclusive with [3].		

Table 103 shows the use of the parameters related to the service primitives:

Table 103

Parameter		Request	Indication	Response	Confirmation
ft_con_id		m	m	m	m
designation		m	m	-	-
recovery_use		o	o	-	-
user_data		o	o	-	-
	domain	o	o	-	-
	fcs32	o	o	-	-
header		-	-	o	-
	filename	-	-	o	-
	destination_name	-	-	o	-
	compression_mode	-	-	o	-
	user_field	-	-	o	-
local_filename		o	-	o	-
result		-	-	m	m

9.3.5.4 Delete file

This IDU is used to delete a file from the remote entity and the parameters are mapped to the T-Delete.

Table 104

Parameters		Purpose
ft_con_id		unique identification of the file transfer connection.
designation		transfer name to be deleted.
user_data [1]		parameter conform to ETS 300 075 [1].
	domain [2]	group of files (A, B, C) applicable for the command.
result		result of the operation.
NOTE 1: This parameter is exclusive with [2].		
NOTE 2: This parameter is exclusive with [1].		

Table 105 shows the use of the parameters related to the service primitives:

Table 105

Parameter		Request	Indication	Response	Confirmation
ft_con_id		m	m	m	m
designation		m	m	-	-
user_data		o	o	-	-
	domain	o	o	-	-
result		-	-	m	m

9.3.5.5 Rename file

This IDU is used to rename a file from the remote entity and the parameters are mapped to the T-Rename.

Table 106

Parameters		Purpose
ft_con_id		unique identification of the file transfer connection.
designation		transfer name to be renamed.
new_name		new name.
user_data [1]		parameter conform to ETS 300 075 [1].
	domain [2]	group of files (A, B, C) applicable for the command.
result		result of the operation.
NOTE 1: This parameter is exclusive with [2].		
NOTE 2: This parameter is exclusive with [1].		

Table 107 shows the use of the parameters related to the service primitives:

Table 107

Parameter		Request	Indication	Response	Confirmation
ft_con_id		m	m	m	m
designation		m	m	-	-
new_name		m	m	-	-
user_data		o	o	-	-
	domain	o	o	-	-
result		-	-	m	m

9.3.5.6 Cancel file operation

This IDU is used to cancel a send or receive file operation and is mapped to T-Transfer_reject or T-P_Exception.

This command is confirmed locally by the CA.

Table 108

Parameters	Purpose
ft_con_id	unique identification of the file transfer connection.
reason	user reason of the abort.

Table 109 shows the use of the parameters related to the service primitives:

Table 109

Parameter	Request	Indication	Confirmation
ft_con_id	m	m	m
reason	m	m	-

9.3.5.7 End of file operation

This IDU is used to indicate to the slave the end of a file operation.

Table 110

Parameters	Purpose
ft_con_id	unique identification of the file transfer connection.
result	result of the file transfer operation.

Table 111 shows the use of the parameters related to the service primitives:

Table 111

Parameter	Indication
ft_con_id	m
result	m

9.3.5.8 List remote filestore information

This IDU is used to get a list of remote files or a list of remote filestore and is mapped to T-Directory or T-Load (see ETS 300 075 [1]).

Table 112

Parameters	Purpose
ft_con_id	unique identification of the file transfer connection.
sort_key	identifies the key according to which the requested list shall be sorted; possible values: date(0), name(1), size(2).
designation	criteria to select a list.
list_mode	the type of list files provided by the remote system: - simple list of files; - extended list of files (description files); - list of filestores; - sublist of filestores.
domain	domain to select the remote list (A, B, C).
local_filename	the local file recipient of the remote list of files.
result	response: indicate acceptance of the file transfer; confirmation: indicate end of the file transfer operation.

Table 113 shows the use of the parameters related to the service primitives:

Table 113

Parameter	Request	Indication	Response	Confirmation
ft_con_id	m	m	m	m
designation	m	m	-	-
list_mode	m	m	-	-
domain	m	m	-	-
sort_key	m	-	-	-
local_filename	m	-	-	-
result	-	-	m	m

9.3.5.9 Navigation on the remote filestore

This IDU is used to select a remote filestore and mapped to T-Save navigation command (see ETS 300 075 [1]).

Table 114

Parameters	Purpose
ft_con_id	unique identification of the file transfer connection.
filestore	remote filestore name.
result	result of the operation.

Table 115 shows the use of the parameters related to the service primitives:

Table 115

Parameter	Request	Indication	Response	Confirmation
ft_con_id	m	m	m	m
filestore	m	m	-	-
result	-	-	m	m

9.3.6 Local PCI services

9.3.6.1 Redirection of incoming calls

The redirection of incoming calls operation is used to redirect an incoming to an LA which has been established a local communications channel to the CA.

Table 116

Parameter	Purpose
new_LA	unique identification of the LA which shall take the incoming call.
redirect_IDU	the IDU (i.e. the incoming call data from a remote system) which shall be redirected.

Table 117 shows the use of the parameters related to the service primitives:

Table 117

Parameter	Request	Confirmation
new_LA	m	-
redirect_IDU	m	-

9.3.6.2 List connected LAs

The list connected LAs operation is used to get the LAs which are locally connected to the CA.

Table 118

Parameter	Purpose
list_password	password used to check if an LA is allowed to get the list of LAs which are locally connected to the CA.
login_name_list	list the login names of those LAs which are locally connected to the CA.

Table 119 shows the use of the parameters related to the service primitives:

Table 119

Parameter	Request	Confirmation
list_password	o	-
login_name_list	-	m

9.3.6.3 List local filestore information

The list local filestore directory operation is used to get the content of a directory of the local filestore.

Table 120

Parameters	Purpose
designation	criteria to select a list, conform to ETS 300 075 [1].
list_mode	the type of list files provided by the local system: - simple list of files; - extended list of files; - list of filestores; - sublist of filestores; - presentation files list.
sort_key	sorting method required for the file list (date, name, size).
domain	domain to select the list (A, B, C).
local_filename	the local file recipient of the local list of files.
result	indicate the result of the local operation.

Table 121 shows the use of the parameters related to the service primitives:

Table 121

Parameter	Request	Confirmation
designation	m	-
list_mode	m	-
sort_key	m	-
domain	m	-
local_filename	m	-
result	-	m

9.3.6.4 Read CA capabilities

The list CA capabilities operation is used to get the technical abilities a CA offers to the LAs.

Table 122

Parameter	Purpose
CA_capabilities	list of the requested capabilities.

Table 123 shows the use of the parameters related to the service primitives:

Table 123

Parameter	Request	Confirmation
CA_capabilities	-	m

9.3.6.5 Read CA configuration

Table 124

Parameter	Purpose
CA_config	list of configuration information of an Easyfile CA.

Table 125 shows the use of the parameters related to the service primitives:

Table 125

Parameter	Request	Confirmation
CA_config	-	m

9.3.6.6 Read phonebook entry

The operation read phonebook entry is used to get the address information of a specific local or remote user.

Table 126

Parameter	Purpose
search_key	unique identification to search a specific entry in the phonebook which shall be retrieved.
phonebook_entry	requested entry of the phonebook.
filestore	working area comprising all the files available for the exchange with the called party.

Table 127 shows the use of the parameters related to the service primitives:

Table 127

Parameter	Request	Confirmation
search_key	m	o
phonebook_entry	-	m
filestore	-	o

9.3.6.7 Read logbook

The operation read logbook is used to get logbook information maintained by the service provider (CA).

Table 128

Parameter	Purpose
selection_criteria	unique identification to select a part of the logbook which shall be retrieved (e.g. date, login_name, ca_id).
logbook_info	requested information of the logbook.
local_filename	local filename to store the requested information of the logbook.

Table 129 shows the use of the parameters related to the service primitives:

Table 129

Parameter	Request	Confirmation
selection_criteria	m	o
logbook_info	-	c
local_filename	o	c

9.3.6.8 Read access control information

The operation read access control information is used to get the access control information of a file in the local filestore.

Table 130

Parameter	Purpose
search_key	unique identification to search a specific entry in the access control list which shall be retrieved.
identifier	identifier which provide access points for the services associated with a received identifier.
authorizations	list of services granted to the caller. This list comprise the following services: load, save, list and optionally delete and rename.
working_area	workspace comprising all the files accessible to the caller.
navigation_support	indication if the service is offered to the caller.
navigation_links	if necessary, indication of the logical links, which may exist, between the navigation filestores and the mandatory working area.

Table 131 shows the use of the parameters related to the service primitives:

Table 131

Parameter	Request	Confirmation
search_key	m	o
identifier	-	m
authorizations	-	m
working_area	-	m
navigation_support	-	o
navigation_links	-	o

9.3.7 Easyfile profile specification

Two profiles are specified for Easyfile:

- the basic Easyfile profile: a CA which claims to be basically conformant to Easyfile shall support the Basic Easyfile profile as specified in this subclause;

- the enhanced Easyfile profile: a CA which claims to be fully conformant to Easyfile shall support the Enhanced Easyfile profile as specified in this subclause.

Basic Profile cover the mandatory communication services in ETS 300 383 [4] others are optional or not used (Transparent and Access/EndOfAccess). Basic Profile convey response IDU through the Alarm return Code (Accepted or Refused).Enhanced profile cover all the communications services.

Table 132 contains the IDUs which shall be supported by a CA claiming to be conformant to the Basic Easyfile profile or to the Enhanced Easyfile profile.

Table 132

Network connection services	**IDUs**	**Basic Profile**	**Enhanced Profile**
establishment of the network connection	ConReq ConConf ConInd ConResp	m m m m	m m m m
release of the network connection	DiscReq DiscConf DiscInd	m m m	m m m
Transparent data	SendDataReq SendDataInd	- -	m m
File transfer connection services	**IDUs**	**Basic Profile**	**Enhanced Profile**
establishment of a file transfer connection	EstabReq EstabConf EstabInd EstabResp	m m m m	m m m m
termination of a file transfer connection	RelReq RelConf RelInd RelResp	m m m m	m m m m
establishment of a file transfer association	AssocReq AssocConf AssocInd AssocResp	- - - -	m m m m
establishment of a file transfer access	AccessReq AccessConf AccessInd AccessResp	- - - -	m m m m
termination of a file transfer access	EndAccessReq EndAccessConf EndAccessInd EndAccessResp	- - - -	m m m m
termination of a file transfer association	TerminateReq TerminateConf TerminateInd TerminateResp	- - - -	m m m m
abort of a file transfer connection	AbortReq AbortConf AbortInd	m m m	m m m
File transfer services	**IDUs**	**Basic Profile**	**Enhanced Profile**
send file	SendReq SendConf SendInd SendResp	m m m m	m m m m
Message	SendMsgReq SendMsgConf SendMsgInd	o o o	m m m
receive file	ReceiveReq ReceiveConf ReceiveInd ReceiveResp	m m m m	m m m m

(continued)

Table 132 (concluded)

Network connection services	IDUs	Basic Profile	Enhanced Profile
delete file	DelReq DelConf DelInd DelResp	o o o o	m m m m
rename file	RenameReq RenameConf RenameInd RenameResp	o o o o	m m m m
Cancel file transfer	CancelReq CancelConf CancelInd	m m m	m m m
End of file operation	EndOperationInd	m	m
list file	ListReq ListConf ListInd ListResp	m m m m	m m m m
Navigation	NavigateReq NavigateConf NavigateInd NavigateResp	o o o o	m m m m

Table 133

Local PCI services	IDUs	Basic Profile	Enhanced Profile
redirection of incoming calls	RedirectReq RedirectConf	o o	o o
list connected LAs	ListLAReq ListLAConf	o o	o o
list local filestore information	LocListReq LocListConf	o o	o o
read CA capabilities	CapabilityReq CapabilityConf	o o	o o
read CA configuration	ConfigReq ConfigConf	o o	o o
read phonebook entry	PhonebookReq PhonebookConf	o o	o o
read logbook	LogbookReq LogbookConf	o o	o o
read access control information	AccessControlReq AccessControlConf	o o	o o

Table 134 contains the parameters which shall be supported by a CA claiming to be conformant.

Table 134

Parameters	Related IDUs	Comment
application_name	AssocReq, AssocInd	
authorizations	AccessControlConf	
CA_capabilities	CapabilityConf	
CA_config	ConfigConf	shall be supported only if the local PCI *read CA configuration* is supported by the CA.
called_addr	ConReq, ConInd	
called_name	ConReq	shall be used only if the phonebook feature is supported by a CA.
called_subaddr	ConReq, ConInd	
calling_addr	ConReq, ConInd	
calling_subaddr	ConReq, ConInd	
compression_list	EstabReq, EstabInd, EstabResp, EstabConf, AccessReq, AccessInd, AccessResp, AccessConf	
compression_mode	SendReq, SendInd, ReceiveConf	
designation	SendReq, SendInd, ReceiveReq ReceiveInd, DelReq, DelInd, RenameReq, RenameInd, ListReq, ListInd, LocListReq	
destination_name	SendReq, SendInd, ReceiveConf	
domain_support	AccessReq, AccessInd, AccessResp, AccessConf	
domain	SendReq, SendInd, ReceiveReq, ReceiveInd, DelReq, DelInd, RenameReq, RenameInd, ListReq, ListInd, LocListReq	
fcs32	SendReq, SendInd, ReceiveReq, ReceiveInd	
file_list	LocListConf	
filename	SendReq, SendInd, ReceiveConf	
filestore	NavigateReq, NavigateInd, PhoneBookConf	
ft_called_addr	AssocReq, AssocInd	
ft_calling_addr	AssocReq, AssocInd	

(continued)

Table 134 (continued)

Parameters	Related IDUs	Comment
ft_con_id	EstabReq, EstabInd, EstabResp, EstabConf, AssocReq, AssocInd, AssocResp, AssocConf, EndAccessReq, EndAccessInd, EndAccessConf, EndAccessResp RelReq, RelInd, RelResp, RelConf TerminateReq, TerminateInd, TerminateResp, TerminateConf AbortReq, AbortInd, AbortConf SendReq, SendInd, SendResp, SendConf ReceiveReq ReceiveInd, ReceiveResp, ReceiveConf DelReq, DelInd, DelResp, DelConf RenameReq, RenameInd, RenameResp, RenameConf CancelReq, CancelInd, ListReq, ListInd, ListResp, ListConf NavigateReq, NavigateInd, NavigateResp, NavigateConf EndOperationInd	
functions	EstabReq, EstabInd, EstabResp, EstabConf AccessReq, AccessInd, AccessResp, AccessConf	
header	SendReq, SendInd, ReceiveConf	
ident_req	AssocReq, AssocInd	
identifier	AccessControlConf	
calling_identifier	EstabReq, EstabInd, AssocReq, AssocInd	
called_identifier	EstabResp, EstabConf, AssocResp, AssocConf	
login_name_list	ListLAConf	
list_mode	ListReq, ListInd, LocListReq	
list_password	ListLAReq	
local_filename	ReceiveReq, ListReq, LocListReq, LogbookReq, LogbookConf	
logbook_info	LogbookConf	
message	SendMsgReq, SendMsgInd	
navigation	EstabReq, EstabInd, EstabResp, EstabConf AccessReq, AccessInd, AccessResp, AccessConf	
navigation_links	AccessControlConf	
navigation_support	AccessControlConf	
net_charge	DiscInd, DiscConf	
net_diag	DiscInd, DiscConf	
net_param	ConReq, ConInd	
net_reason	DiscInd, DiscConf	
netcon_type	ConReq, ConInd	
new_LA	RedirectReq	
new_name	RenameReq, RenameInd	
phonebook_entry	PhonebookConf	
reason	EstabResp, EstabConf, AssocResp, AssocConf AbortReq, AbortInd, CancelReq, CancelInd	

(continued)

Table 134 (concluded)

Parameters	Related IDUs	Comment
recovery	EstabReq, EstabInd, EstabResp, EstabConf AccessReq, AccessInd, AccessResp, AccessConf	
recovery_use	SendReq, SendInd, ReceiveReq, ReceiveInd	
redirect_IDU	RedirectReq	
responding_addr	ConResp, ConConf	
responding_subaddr	ConResp, ConConf	
result	EndAccessReq, EndAccessInd, EndAccessConf, EndAccessResp, TerminateResp, TerminateConf, RelResp, RelConf, SendResp, SendConf, ReceiveResp, ReceiveConf, DelResp, DelConf, RenameResp, RenameConf, EndOperationInd, ListResp, ListConf, NavigateResp, NavigateConf, LocListConf	
role	AccessReq, AccessInd, AccessResp, AccessConf	
sbv_data	SendDataReq, SendDataInd	
search_key	PhonebookReq, PhonebookConf, AccessControlReq, AccessControlConf	
selection_criteria	LogbookReq, LogbookConf	
sort_key	ListReq, LocListReq	
src_effect	SendReq	
timeouts	AssocReq, AssocInd, AssocResp	
user_data	EstabReq, EstabInd, EstabResp, AssocReq, AssocInd, AssocResp, AccessReq, AccessInd, AccessResp, AccessConf EndAccessReq, EndAccessInd, EndAccessConf, EndAccessResp, TerminateReq, TerminateInd, RelReq, RelInd, SendReq, SendInd, ReceiveReq, ReceiveInd, DelReq, DelInd, RenameReq, RenameInd	
user_field	SendReq, SendInd, ReceiveConf	
uuid	ConReq, ConInd	
working_area	AccessControlConf	

10 CA capability description

The interface of this ETS was specified independently from programming languages, hardware platforms and operating systems. Implementors who wish to develop a CA according to this ETS will face parts which are product specific and which depend on the real computing environment. Optional features, facilities and services are specified in this ETS, i.e. in the profile specification, which may be supported by a CA product which claims to be conformant to this ETS.

To guarantee that LA(s) and CA(s) are capable of communicating with each other and of interchanging information through the PCI, an LA must know how the product specific capabilities of a CA are realized. Therefore, the CA manufacturer shall provide the CA capability description which shall be accessible by all LAs within a given commuications system.

The CA capabilities which shall be specified by the CA manufacturer are:

- declaration of the supported interactive service(s): a CA may offer one or more than one interactive services, i. e. file transfer services; therefore, the CA manufacturer shall declare if his product supports SimpleFTAM or Easyfile or both services;

- declaration of the supported profile(s): a CA may support the basic profile, the enhanced profile or both profiles; therefore, the CA manufacturer shall declare for each supported file transfer service the supported profile(s); additionally all optional and conditional features (e.g. the local PCI services) of the respective profile which are supported by the CA shall be declared;

- declaration of the supported network(s): the ISDN shall be supported by a CA which claims to be conformant to this ETS; access to other networks may be specified in future extensions of this ETS;

- LA selection mechanism (see also clause 6): a CA shall be able to select the appropriate LA to which an incoming call shall be passed using the parameter uuid as specified in subclause 7.2; the parameter uuid shall be conveyed in a field of the lower layer protocol, e.g. NSAP, MSN, User Data; the CA manufacturer shall declare in which field the uuid parameter shall be conveyed and the data structure and format of this parameter;

- CA identification: the different CAs within a given system shall be identified uniquely and unambiguously using the parameter CA_ID; therefore, the CA_ID(s) (see subclause 7.2) shall be provided by the product manufacturer which enables an LA to select the CA appropriate for its requirements;

- CA incorporation and deletion: the communications systems addressed in this ETS may consist of multiple LAs and multiple CAs (see subclause 5.1); the manufacturer of a CA which claims to be conformant to this ETS shall specify how a new CA shall be incorporated into a given system and how a CA shall be deleted from a given system;

- the specification of a phonebook entry: if the optional phonebook feature is supported by a CA, the data structure and format of a phonebook entry shall be specified by the CA manufacturer;

- the specification of a logbook entry: if the optional logbook feature is supported by a CA, the data structure and format of a logbook entry shall be specified by the CA manufacturer;

- the specification of an access control information: if the optional access control information feature is supported by a CA, the data structure and format of this information shall be specified by the CA manufacturer;

- specification of the CA configuration: each layer of the file transfer service contains elements which have to be negotiated during the connection establishment phase; such protocol parameters and the range of valid values shall be specified by the CA manufacturer;

- specification of the virtual filestore: the directory structure of the local filestore and the data structure and format of the path to a directory or subdirectory shall be specified by the CA manufacturer;

- specification of filename: data structure and format of filenames of the local filestore shall be specified by the CA manufacturer;

- specification of the passwords: data structure and format of the supported passwords (e.g. the filestore password) shall be specified by the CA manufacturer.

If a CA supports the file transfer service SimpleFTAM the following capabilities shall be specified by the CA manufacturer:

- specification of the supported content types;
- specification of the suppored attribute groups;
- specification of the supported service classes;
- specification of the supported Functional Units (FUs).

The declaration of the product specific capabilities of a CA may be provided using:

- the ICE concept as specified in APPLI/COM, see ETS 300 243-1 [3];
- a context sensitive help system (e.g. in Windows environments);
- text coded files;
- the product documentation.

11 Binary encoding scheme

This clause specifies the binary encoding scheme for the IDUs using the C programming language as a generic description language.

The purpose of this description is to provide binary compatibility of IDUs between different vendors' applications using the binary encoding scheme.

The descriptions provided have to be adapted to the supporting platform to perform properly.

This generic description has to be adapted to the "real" environment of a running system (e.g. hardware platform, operating system and programming language used).

11.1 General types definition

Table 135

Generic C Data type	Comment
INT16	a signed integer value, coded in 16 bits.
UINT32	an unsigned integer value, coded in 32 bits.
CHAR	a character coded in 8 bits.

```
/***
*   Type definition for BOOLEAN
***/
typedef  enum  {
    false=0,
    true
    }   Boolean_type;

/***
*   Type definition for GraphicString
***/
typedef  CHAR*  GraphicString_type;

/***
*   Type definition for OCTETSTRING
***/
typedef  struct  {
    CHAR*   value;
    INT16   length;
    }   OctetString_type;
```

11.2 IDU encoding scheme

Each IDU consists of:

- general parameters which shall be common for the file transfer services specified in this ETS, i.e. SimpleFTAM and Easyfile; the collection of these parameters are called IDU_header in the following;

- file transfer parameters shall be dependent from the file transfer service; the collection of these parameters are called IDU_Ft in the following.

```
/***
*   <IDU> syntax element
***/
struct IDU {
struct IDU_header    header;    /* <IDU Header> syntax element, see below */
union IDU_Ft  filetransfer; /* <IDU Ft> syntax element see below */
};
/***
*   <IDU header> structure
***/
struct header_idu {     /* General parameters */
INT16     IDU_ref;  /* <IDUref> */
Netcon_type   netcon_id;/* <NetconID> */
Service_id_type  service_id;   /* <ServiceID> */
SP_id_typesp_id; /* <SPID> */
Result_type   result;    /* <Result> */
};
/***
*   <IDU Ft> structure
***/
union {       /* File transfer service specific parameters */
union sft_IDU sft_IDU;  /* <sftIDUf> */
union eft_IDU eft_IDU;  /* <eftIDU> */
};
/***
*   Possible values of Netcon_type
***/
typedef  enum  {
   ISDN=0
   }  Netcon_type;


/***
/*  Possible values of Service_id_type */
***/
#define    IA_FT_CONNECT        0x1001
#define    IA_FT_DISCONNECT     0x1002
#define    IA_FT_ESTABLISH      0x1003
#define    IA_FT_RELEASE        0x1004
#define    IA_FT_ABORT          0x1005
#define    IA_FT_SEND           0x1006
#define    IA_FT_RECEIVE        0x1007
#define    IA_FT_DELETE         0x1008
#define    IA_FT_RENAME         0x1009
#define    IA_FT_LIST           0x1010
#define    IA_FT_CANCEL         0x1011
#define    IA_FT_END            0x1012
#define    IA_FT_NAVIGATE       0x1013

/***
/*  SimpleFTAM specific values of Service_id_type */
***/
#define    IA_FT_READ_ATTRIB    0x1201
#define    IA_FT_CHANGE_ATTRIB  0x1202

/***
/*  Easyfile specific values of Service_id_type */
***/
```

```
#define   IA_FT_TRANSPARENT   0x1101
#define   IA_FT_ASSOCIATE     0x1102
#define   IA_FT_TERMINATE     0x1103
#define   IA_FT_ACCESS        0x1104
#define   IA_FT_ENDACCESS     0x1105
#define   IA_FT_MESSAGE       0x1106

/***
/*  Possible values of the local PCI services specific Service_id_type */
***/
#define   IA_LOC_REDIRECT     0x2001
#define   IA_LOC_LIST_LA      0x2002
#define   IA_LOC_LIST         0x2003
#define   IA_LOC_READ_CA      0x2004
#define   IA_LOC_READ_CONF    0x2005
#define   IA_LOC_READ_PHBOOK  0x2006
#define   IA_LOC_READ_LOG     0x2007
#define   IA_LOC_READ_ACCESS  0x2008

/***
    Possible values of SP_id_type */
***/
typedef  enum  {
    request=1,
    indication,
    response,
    confirmation} SP_id_type;

/***
*   Possible values of Result_type */
***/
typedef  enum  {
    success=0,
    transient-error,
    permanent-error} Result_type;
```

11.3 SimpleFTAM specific encoding scheme

11.3.1 General constants and type definitions

```
/***
*   Type definition for BITSTRING
***/
typedef  struct  {
    CHAR*   value;
    INT16   unused_bits;
    }   BitString_type;

/***
*   Type definition for GeneralizedTime
***/
typedef  CHAR*  GeneralizedTime_type;

/***
*   Type definition for OBJECT IDENTIFIER
***/
typedef  struct  objectidentifier_type  {
    INT16   value;
    struct objectidentifier_type    *next;
    }   ObjectIdentifier_type;

/***
*   Type definition for NULL
***/
typedef  Null_type;
```

11.3.2 SimpleFTAM IDUs

```
/***
*   SimpleFTAM IDUs
***/

union  sft_idu  {
    struct sft_connect; /* <sftconnect> */
    struct sft_disconnect;  /* <sftdisconnect> */
    struct sft_establish;   /* <sftestablish> */
    struct sft_release; /* <sftrelease> */
    struct sft_abort;   /* <sftabort> */
    struct sft_send;    /* <sftsend> */
    struct sft_receive; /* <sftreceive> */
    struct sft_delete;  /* <sftdelete> */
    struct sft_rename;  /* <sftrename> */
    struct sft_cancel;  /* <sftcancel> */
    struct sft_end; /* <sftend> */
    struct sft_read_attribute;  /* <sftreadattribute> */
    struct sft_change_attribute;    /* <sftchangeattribute> */
    struct sft_list_remote; /* <sftlistremote> */
    struct sft_navigation;  /* <sftnavigation> */
    struct sft_redirection; /* <sftredirection> */
    struct sft_list_la; /* <sftlistLA> */
    struct sft_list_local;  /* <sftlistlocal> */
    struct sft_read_capabilities;   /* <sftreadCAcapabilities> */
    struct sft_read_configuration;  /* <sftreadCAconfiguration> */
    struct sft_read_phonebook;  /* <sftreadphonebookentry> */
    struct sft_read_logbook;    /* <sftreadlogbook> */
    struct sft_read_access_control; /* <sftreadaccesscontrol> */
};

/***
*   Establishment of a network connection
***/
struct sft_connect {    /* <sftconnect> */
    OctetString_type  calling_addr; /* calling address (ISDN) */
    OctetString_type  calling_subaddr;  /* calling subaddress (ISDN) */
    OctetString_type  called_addr;  /* called address (ISDN) */
    OctetString_type  called_subaddr;   /* called subaddress (ISDN) */
    OctetString_type  called_name;  /* alias name */
    OctetString_type  responding_addr;  /* responding address (ISDN) */
    OctetString_type  responding_subaddr;   /* responding subaddress (ISDN) */
    Netcon_value  netcon_type;  /* network connection type (ISDN) */
    OctetString_type  net_reason;   /* network reason (ISDN) for rejection */
    OctetString_type  net_diag; /* network diagnostics (ISDN) */
    OctetString_type  net_param;    /* additional network parameters (ISDN) */
    OctetString_type  uuid; /* user to user identification */
};

/***
*   Release of a network connection
***/
struct sft_disconnect { /* <sftdisconnect> */
    OctetString_type  net_charge;   /* network charging information (ISDN) */
    OctetString_type  net_reason;   /* network reason (ISDN) for rejection */
    OctetString_type  net_diag; /* network diagnostics (ISDN) */
};

/***
*   Establishment of a file transfer connection
***/
struct sft_establish {  /* <sftestablish> */
    INT16  ft_con_id;   /* file transfer connection id */
    INT16  called_application_title;    /* called application title */
    ObjectIdentifier_type  called_application_title;
    GraphicString_type called_presentation_addr;    /* called presentation address */
    OctetString_type  called_name;  /* alias name */
    BitString_type service_class;   /* service class */
    BitString_type fu;  /* functional units */
    BitString_type attribute_groups;    /* attribute groups */
    Qos_value  qos; /* quality of service */
    struct Contents_Type_List  contents_type_list;  /* list of content types */
    GraphicString_type initiator_id;    /* initiator_identity */
    GraphicString_type account; /* account */
    union Password filestore_password;  /* remote_virtual_filestore_password */
    struct Diagnostic  diagnostic;  /* diagnostic information */
};

/***
*   Termination of a file transfer connection
***/
struct sft_release {    /* <sftrelease> */
    INT16  ft_con_id;   /* file transfer connection identifier */
```

```
    struct Charging  charging;  /* charging information */
};

/***
*   Abort of a file transfer connection
***/
struct sft_abort {  /* <sftabort> */
    INT16  ft_con_id;   /* file transfer connection identifier */
    struct Diagnostic  diagnostic;  /* diagnostic information */
};

/***
*   Send file
***/
struct sft_send {   /* <sftsend> */
    INT16 ft_con_id;    /* file transfer connection identifier */
    struct Filename  src_filename;  /* source filename */
    INT16 src_effect;   /* source effect */
    struct Filename  tgt_filename;  /* target filename */
    Override_value  override;   /* override */
    struct Initial_Attributes  initial_attributes;  /* initial file attributes */
    union Password  create_password;    /* create password *?
    GraphicString_type  contents_type;  /* content type */
    BitString_type  requested_access;   /* requested access */
    union Password  write_password; /* write password */
    struct Concurrency_Control  concurrency_control;    /* concurrency control */
    GraphicString_type  account;    /* accounting */
    struct Charging  charging;  /* charging information */
    struct Diagnostic  diagnostic;  /* diagnostic information */
};

/***
*   Receive file
***/
struct sft_receive {    /* <sftreceive> */
    INT16  ft_con_id;   /* file transfer connection identifier */
    struct Filename  src_filename;  /* source filename */
    struct Filename  tgt_filename;  /* target filename */
    GraphicString_type  contents_type;  /* content type */
    union Password  read_password;  /* read password */
    struct Concurrency_Control  concurrency_control;    /* concurrency control */
    GraphicString_type  account;    /* accounting */
    struct Charging  charging;  /* charging information */
    struct Diagnostic  diagnostic;  /* diagnostic information */
};

/***
*   Delete file
***/
struct sft_delete { /* <sftdelete> */
    INT16  ft_con_id;   /* file transfer connection identifier */
    struct Filename  src_filename;  /* source filename */
    union Password  delete_password;    /* delete password */
    union Password  read_password;  /* read password */
    GraphicString_type  contents_type;  /* content type */
    struct Concurrency_Control  concurrency_control;    /* concurrency control */
    GraphicString_type  account;    /* accounting */
    struct Charging  charging;  /* charging information */
    struct Diagnostic  diagnostic;  /* diagnostic information */
};

/***
*   Rename file
***/
struct sft_rename { /* <sftrename> */
    INT16  ft_con_id;   /* file transfer connection identifier */
    struct Filename  src_filename;  /* source filename */
    struct Filename  tgt_filename;  /* target filename */
    union Password  change_attribute_password;  /* change attribute password */
    struct Concurrency_Control  concurrency_control;    /* concurrency control */
    GraphicString_type  account;    /* accounting */
    struct Charging  charging;  /* charging information */
    struct Diagnostic  diagnostic;  /* diagnostic information */
};

/***
*   Cancel file operation
***/
struct sft_cancel { /* <sftcancel> */
    INT16  ft_con_id;   /* file transfer connection identifier */
    INT16 IDU_ref;  /* unique IDU identification */
    struct Diagnostic  diagnostic;  /* diagnostic information */
};

/***
```

```
*   End of file operation
***/
struct sft_end {    /* <sftend> */
    INT16  ft_con_id;   /* file transfer connection identifier */
};

/***
*   Read file attributes
***/
struct sft_read_attribute { /* <sftreadattribute> */
    INT16  ft_con_id;   /* file transfer connection identifier */
    struct Filename  src_filename;  /* source filename */
    union Password  read_attribute_password;    /* read attribute password */
    struct Concurrency_Control  concurrency_control;    /* concurrency control */
    GraphicString_type  account;    /* accounting */
    BitString_type attribute_names; /* attribute names */
    struct Attribute_Values  attribute_values;  /* attribute values */
    struct Charging  charging;  /* charging information */
    struct Diagnostic  diagnostic;  /* diagnostic information */
};

/***
*   Change file attributes
***/
struct sft_change_attribute {   /* <sftchangeattribute> */
    INT16  ft_con_id;   /* file transfer connection identifier */
    struct Filename  src_filename;  /* source filename */
    struct Change_Attributes  change_attributes;    /* change attribute */
    union Password  change_attribute_password;  /* change attribute password*/
    struct Concurrency_Control  concurrency_control;    /* concurrency control */
    GraphicString_type  account;    /* accounting */
    struct Charging  charging;  /* charging information */
    struct Diagnostic  diagnostic;  /* diagnostic information */
};

/***
*   List remote filestore information
***/
struct sft_list_remote {    /* <sftlistremote> */
    INT16  ft_con_id;   /* file transfer connection identifier */
    SortKey_value sort_key; /* sort key */
    struct Dir_name  dir_name;  /* directory name */
    BitString_type attribute_names; /* attribute names */
    union Password  read_password;  /* read password */
    struct Concurrency_Control  concurrency_control;    /* concurrency control */
    GraphicString_type  account;    /* accounting */
    struct File_List  file_list;    /* list of files */
    struct Filename  filename;  /* name of the file containing the list of files */
    struct Charging  charging;  /* charging information */
    struct Diagnostic  diagnostic;  /* diagnostic information */
};

/***
*   Navigation on the remote filestore
***/
struct sft_navigation { /* <sftnavigation> */
    INT16  ft_con_id;   /* file transfer connection identifier */
    struct Dir_name  dir_name;  /* directory name */
    Boolean_type reset; /* reset */
    union Password  filestore_password; /* filestore pasword */
    struct Diagnostic  diagnostic;  /* diagnostic information */
};

/***
*   Redirection of incoming calls
***/
struct sft_redirection {    /* <sftredirection> */
    GraphicString_type  new_LA; /* target LA login name */
    struct IDU  redirect_IDU;   /* redirect IDU */
    };
};
/***
*   List connected LAs
***/
struct sft_list_la {    /* <sftlistLA> */
    union Password  list_password;  /* list password */
    struct LA_login_name  login_name_list;  /* list of actually logged in LAs */
};

/***
```

```
*   List local filestore information
***/
struct sft_list_local { /* <sftlistlocal> */
    struct Dir_name  dir_name;  /* directory name */
    BitString_type  attribute_names;    /* attribute names */
    SortKey_value  sort_key;    /* sort key */
    struct File_List  file_list;    /* list of files */
    struct Filename  filename;  /* name of the file containing the list of files */
};

/***
*   Read CA capabilities
***/
struct sft_read_descriptor {    /* <sftreadCAcapability> */
    GraphicString_type  ca_capabilities;    /* CA capabilities */
};

/***
*   Read CA configuration
***/
struct sft_read_configuration { /* <sftreadCAconfiguration> */
    GraphicString_type  ca_conf;    /* CA configuration */
};

/***
*   Read phonebook entry
***/
struct sft_read_phonebook { /* <sftreadphonebookentry> */
    CHAR*  search_key;  /* search key */
    GraphicString_type  phonebook_entry;    /* phonebook entry */
};

/***
*   Read logbook
***/
struct sft_read_logbook {   /* <sftreadlogbook> */
    CHAR*  selection_criteria;  /* selection criteria */
    GraphicString_type  logbook_info;   /* logbook information */
    struct Filename  filename;  /* name of the file containing the logbook information */
};

/***
*   Read access control information
***/
struct sft_read_access_control {    /* <sftreadaccesscontrol> */
    struct Dir_name  dir_name;  /* directory name */
    struct Filename  src_filename;  /* source file */
    struct Access_Control_Info  access_control_info;    /* access control information */
};
```

11.3.2.1 SimpleFTAM IDU parameters

```
union Access_Control_Attribute {    /* CHOICE */
    Null_type   no_value_available;
    struct Access_Control_Element   actual_values;
};

union Access_Control_Change_Attribute { /* CHOICE */
    Null_type   no_value_available;
    struct Actual_values    actual_values;
};

struct Access_Control_Element { /* SET OF */
    BitString_type  access_request;
    struct Concurrency_Access   concurrency_access;
    union User_Identity identity;
    struct Access_Passwords passwords;
    struct Application_Entity_Title location;
    struct Access_Control_Element   *next;
};

struct Access_Control_Info {    /* SEQUENCE OF SEQUENCE */
    BitString_type  access_request;
    struct Concurrency_Access   concurrency_access;
    union User_Identity identity;
    struct Access_Passwords passwords;
    struct Application_Entity_Title location;
    struct Access_Control_Info  *next;
};
```

```
struct Access_Passwords {   /* SEQUENCE */
    union Password  read_password;
    union Password  insert_password;
    union Password  replace_password;
    union Password  extend_password;
    union Password  erase_password;
    union Password  read_attribute_password;
    union Password  change_attribute_password;
    union Password  delete_password;
};

union Account_Attribute {   /* CHOICE */
    Null_type   no_value_available;
    GraphicString_type  actual_values;
};

struct Actual_values {  /* SEQUENCE */
    struct Access_Control_Element   insert_values;
    struct Access_Control_Element   actual_values;
};

struct Application_Entity_Title {   /* ACSE_1_AE_Title_type */
    ObjectIdentifier_type   AP_title_form2;
    INT16   AE_qualifier_form2;
};

struct Attribute_Values {   /* SEQUENCE */
    struct Filename_Attribute   filename;
    BitString_type  permitted_actions;
    CHAR*   contents_type;
    union Account_Attribute storage_account;
    union Date_and_Time_Attribute   date_and_time_of_creation;
    union Date_and_Time_Attribute   date_and_time_of_last_modification;
    union Date_and_Time_Attribute   date_and_time_of_last_read_access;
    union Date_and_Time_Attribute   date_and_time_of_last_attribute_modification;
    union User_Identity_Attribute   identity_of_creator;
    union User_Identity_Attribute   identity_of_last_modifier;
    union User_Identity_Attribute   identity_of_last_reader;
    union User_Identity_Attribute   identity_of_last_attribute_modifier;
    union File_Availability_Attribute   file_availability;
    union Filesize_Attribute    filesize;
    union Filesize_Attribute    future_filesize;
    union Access_Control_Attribute  access_control;
    union Legal_Qualification_Attribute legal_qualification;
};

struct Change_Attributes {  /* SEQUENCE */
    struct Filename_Attribute   filename;
    union Account_Attribute storage_account;
    union File_Availability_Attribute   file_availability;
    union Filesize_Attribute    future_filesize;
    union Access_Control_Change_Attribute   access_control;
    union Legal_Qualification_Attribute legal_qualification;
};

struct Charging {   /* SEQUENCE OF SEQUENCE */
    GraphicString_type  resource_identifier;
    GraphicString_type  charging_unit;
    INT16   charging_value;
    struct Charging *next;
    };

struct Concurrency_Access { /* SEQUENCE */
    BitString_type  read;
    BitString_type  insert;
    BitString_type  replace;
    BitString_type  extend;
    BitString_type  erase;
    BitString_type  read_attribute;
    BitString_type  change_attribute;
    BitString_type  delete_file;
};

struct Concurrency_Control {    /* SEQUENCE */
    Lock_value  read;
    Lock_value  insert;
    Lock_value  replace;
    Lock_value  extend;
    Lock_value  erase;
    Lock_value  read_attribute;
    Lock_value  change_attribute;
    Lock_value  delete_files;
};
```

```
struct Constraint_Set { /* SEQUENCE */
    ObjectIdentifier_type   constraint_set_name;
    ObjectIdentifier_type   abstract_syntax_name;
};

struct Contents_Type_List { /* SEQUENCE OF GraphicString */
    GraphicString_type  value;
    struct Contents_Type_List   *next;
};

struct Diagnostic { /* SEQUENCE OF SEQUENCE */
    DiagnosticType_value    diagnostic_type;
    INT16   error_identifier;
    EntityReference_value   error_observer;
    EntityReference_value   error_source;
    INT16   suggested_delay;
    GraphicString_type  further_details;
    struct Diagnostic   *next;
};

struct Dir_name {   /* SEQUENCE OF GraphicString */
    GraphicString_type  value;
    struct Dir_name *next;
};

union Date_and_Time_Attribute { /* CHOICE */
    Null_type   no_value_available;
    GeneralizedTime_type    actual_values;
};

union File_Availability_Attribute { /* CHOICE */
    Null_type   no_value_available;
    F_A_A_ActualValues_value    actual_values;
};

struct File_List {  /* SEQUENCE OF SEQUENCE */
    struct Filename_Attribute   filename;
    BitString_type  permitted_actions;
    CHAR*   contents_type;
    union Account_Attribute storage_account;
    union Date_and_Time_Attribute   date_and_time_of_creation;
    union Date_and_Time_Attribute   date_and_time_of_last_modification;
    union Date_and_Time_Attribute   date_and_time_of_last_read_access;
    union Date_and_Time_Attribute   date_and_time_of_last_attribute_modification;
    union User_Identity_Attribute   identity_of_creator;
    union User_Identity_Attribute   identity_of_last_modifier;
    union User_Identity_Attribute   identity_of_last_reader;
    union User_Identity_Attribute   identity_of_last_attribute_modifier;
    union File_Availability_Attribute   file_availability;
    union Filesize_Attribute    filesize;
    union Filesize_Attribute    future_filesize;
    union Access_Control_Attribute  access_control;
    union Legal_Qualification_Attribute legal_qualification;
    struct File_List    *next;
};

struct Filename {   /* SEQUENCE OF GraphicString */
    GraphicString_type  value;
    struct Filename *next;
};

struct Filename_Attribute { /* SEQUENCE OF GraphicString */
    GraphicString_type  value;
    struct Filename *next;
};

union Filesize_Attribute {  /* CHOICE */
    Null_type   no_value_available;
    INT16   actual_values;
};

struct Initial_Attributes { /* SEQUENCE */
    struct Filename_Attribute   filename;
    BitString_type  permitted_actions;
    CHAR*   contents_type;
    union Account_Attribute storage_account;
    union File_Availability_Attribute   file_availability;
    union Filesize_Attribute    future_filesize;
    union Access_Control_Attribute  access_control;
    union Legal_Qualification_Attribute legal_qualification;
};
```

```
struct LA_login_name  { /* list of the login names of the actually connected LAs */
    GraphicString_type  value;
    struct LA_login_name    *next;
};

union Legal_Qualification_Attribute {   /* CHOICE */
    Null_type   no_value_available;
    GraphicString_type  actual_values;
};

union Password {    /* CHOICE */
    CHAR*   octet_value;
    GraphicString_type  graphicString_value;
};

union User_Identity {   /* CHOICE */
    Null_type   no_value_available;
    GraphicString_type  actual_values;
};

union User_Identity_Attribute { /* CHOICE */
    Null_type   no_value_available;
    union User_Identity actual_values;
};
```

11.3.2.1 SimpleFTAM IDU values

```
/***
*   Values used for Access Context
***/
typedef  enum  {
    hierarchical_all_data_units=0,
    hierarchical_no_data_units,
    flat_all_data_units,
    flat_one_level_data_units,
    flat_single_data_unit,
    unstructured_all_data_units,
    unstructerd_single_data_unit
    }   AccessContext_value;

/***
*   Values used for Action Result
***/
typedef  enum  {
    success=0,
    transient_error,
    permanent_error
    }   ActionResult_value;

/***
*   Values used for Diagnostic Type
***/
typedef  enum  {
    informative=0,
    transient,
    permanent
    }   DiagnosticType_value;

/***
*   Values used for Entity Reference
***/
typedef  enum  {
    no_categorization_possible=0,
    initiating_file_service_user,
    initiating_file_protocol_machine,
    service_supporting_the_file_protocol_machine,
    responding_file_protocol_machine,
    responding_file_service_user
    }   EntityReference_value;

/***
*   Values used for FADU Lock
***/
typedef  enum  {
    off=0,
    on
    }   FADULock_value;
```

```
/***
*   Values used for FADU Operation
***/
typedef  enum  {
    insert=0,
    replace,
    extend
    }   FADUOperation_value;

/***
*   Values used for File_Availability_Atribute  Actual_Values
***/
typedef  enum {
    immediate_availability=0,
    deferred_availability
    }   F_A_A_ActualValues_value;

/***
*   Values used for Lock
***/
typedef  enum {
    not_required=0,
    shared,
    exclusive,
    no_access
    }   Lock_value;

/***
*   Values used for Netcon
***/
typedef  enum  {
    ISDN=0
    }   Netcon_value;


/***
*   Values used for Override
***/
typedef  enum  {
    create_failure=0,
    select_old_file,
    delete_and_create_with_old_attribute,
    delete_and_create_with_new_attribute
    }   Override_value;

/***
*   Values used for Recovery Mode
***/
typedef  enum  {
    none=0,
    at_start_of_file,
    at_any_active_checkpoint
    }   RecoveryMode_value;

/***
*   Values used for Quality_of_service
***/
typedef  enum  {
    no_recovery=0,
    class_1_recovery,
    class_2_recovery,
    class_3_recovery
    }   Qos_value;

/***
*   Values used for Sort_key
***/
typedef  enum  {
    date=0,
    name,
    size
    }   SortKey_value;

/***
*   Values used for State Result
***/
typedef  enum  {
    success=0,
    failure
    }   StateResult_value;
```

```
/***
*   Constants used for Access Request
***/
#define  AR_read    0x00;   /* read (0) */
#define  AR_insert  0x80;   /* insert (1) */
#define  AR_replace 0x40;   /* replace (2) */
#define  AR_extend  0x20;   /* extend (3) */
#define  AR_erase   0x10;   /* erase (4) */
#define  AR_read_attribute  0x08;   /* read_attribute (5) */
#define  AR_change_attribute    0x04;   /* change_attribute (6) */
#define  AR_delete_file 0x02;   /* delete_file (7) */

/***
*   Constants used for Attribute Groups
***/
#define  AG_storage 0x00;   /* storage (0) */
#define  AG_security    0x80;   /* security (1) */
#define  AG_private 0x40;   /* private (2); shall not be used for
    SimpleFTAM*/

/***
*   Constants used for Attribute Names
***/
    /***    Kernel group    ***/
#define  AN_read_filename   0x000000;   /* read_filename(0) */
#define  AN_read_permitted_actions  0x800000;   /* read_permitted_actions(1) */
#define  AN_read_contents_type  0x400000;   /* read_contents_type(2) */
    /***    Storage group   ***/
#define  AN_read_storage_account    0x200000;   /* read_storage_account(3) */
#define  AN_read_d_t_creation   0x100000;   /* read_date_and_time_of_creation(4) */
#define  AN_read_d_t_last_mod   0x080000;   /* read_date_and_time_of_last_
    modification(5) */
#define  AN_read_d_t_last_read_acc  0x040000;   /* read_date_and_time_of_last_read_
    access(6) */
#define  AN_read_d_t_last_attr_mod  0x020000;   /* read_date_and_time_of_last_attribute_
    modification(7) */
#define  AN_read_id_creator 0x010000;   /* read_identity_of_creator (8) */
#define  AN_read_id_last_modifier   0x008000;   /* read_identity_of_last_modifier(9) */
#define  AN_read_id_last_reader 0x004000;   /* read_identity_of_last_reader(10) */
#define  AN_read_id_last_attr_mod   0x002000;   /* read_identity_of_last_attribute_
    modifier(11) */
#define  AN_read_file_avail 0x001000;   /* read_file_availability(12) */
#define  AN_read_filesize   0x000800;   /* read_filesize(13) */
#define  AN_read_future_filesize    0x000400;   /* read_future_filesize(14) */
    /***    Security group  ***/
#define  AN_read_access_control 0x000200;   /* read_access_control(15) */
#define  AN_read_legal_qualifications   0x000100;   /* read_legal_qualifications(16) */
    /***    Private group   shall not be used for SimpleFTAM***/
#define  AN_read_priv_use   0x000080;   /* read_private_use(17) */

/***
*   Constants used for Concurrency Access
***/
#define  CA_not_required    0x00;   /* not_required (0) */
#define  CA_shared  0x80;   /* shared (1) */
#define  CA_exclusive   0x40;   /* exclusive (2) */
#define  CA_no_access   0x20;   /* no_access (3) */

/***
*   Constants used for Functional Units
***/
#define  FU_kernel  0x0000; /* kernel ( ) */
#define  FU_read    0x4000; /* read (2) */
#define  FU_write   0x2000; /* write (3) */
#define  FU_lim_file_mgt    0x0800; /* limited_file_management (5) */
#define  FU_enh_file_mgt    0x0400; /* enhanced_file_management (6) */
#define  FU_grouping    0x0200; /* grouping (7) */
#define  FU_recovery    0x0080; /* recovery (9) */
#define  FU_restart_data_xfer   0x0040; /* restart_data_transfer (10) */

/***
*   Constants used for Permitted Actions
***/
#define  PA_read    0x0000; /* read (0) */
#define  PA_insert  0x8000; /* insert (1) */
#define  PA_replace 0x4000; /* replace (2) */
#define  PA_extend  0x2000; /* extend (3) */
#define  PA_erase   0x1000; /* erase (4) */
#define  PA_read_attribute  0x0800; /* read_attribute (5) */
#define  PA_change_attribute    0x0400; /* change_attribute (6) */
#define  PA_delete_file 0x0200; /* delete_file (7) */
#define  PA_traversal   0x0100; /* traversal (8) */
#define  PA_reverse_traversal   0x0010; /* reverse_traversal (9) */
#define  PA_random_order    0x0020; /* random_order (10) */
```

```
/***
*   Constants used for Processing Mode
***/
#define  PM_f_read  0x00;   /* f_read (0) */
#define  PM_f_insert    0x80;   /* f_kernel (1) */
#define  PM_f_replace   0x40;   /* f_replace (2) */
#define  PM_f_extend    0x20;   /* f_extend (3) */
#define  PM_f_erase 0x10;   /* f_erase (4) */

/***
*   Constants used for Service Class
***/
#define  SC_mgt_class   0x80;   /* management_class (1) */
#define  SC_xfer_class  0x40;   /* transfer_class (2) */
#define  SC_xfer_mgt_class  0x20;   /* transfer_and_management_class (3) */
```

11.4 Easyfile specific encoding scheme

```
/***
*   <Easyfile IDU> structure
***/
union {
struct eft_connect; /*<eftconnect>*/
struct eft_disconnect;  /*<eftdisconnect>*/
struct eft_transparent; /*<efttransparent>*/
struct eft_establish;   /*<eftestablish>*/
struct eft_release; /*<eftrelease>*/
struct eft_associate;   /*<eftassociate>*/
struct eft_terminate;   /*<eftterminate>*/
struct eft_access;  /*<eftaccess>*/
struct eft_endaccess;   /*<eftendaccess>*/
struct eft_abort;   /*<eftabort>*/
struct eft_send;    /*<eftsend>*/
struct eft_message; /*<eftmessage>*/
struct eft_receive; /*<eftreceive>*/
struct eft_delete;  /*<eftdelete>*/
struct eft_rename;  /*<eftrename>*/
struct eft_cancel;  /*<eftcancel>*/
struct eft_end; /*<eftend>*/
struct eft_list_remote; /*<eftlistremote>*/
struct eft_navigate;    /*<eftnavigate>*/
struct eft_redirection; /* <eftredirection> */
    struct eft_list_la; /* <eftlistLA> */
    struct eft_list_local;  /* <eftlistlocal> */
    struct eft_read_capabilities;   /* <eftreadCAcapabilities> */
    struct eft_read_configuration;  /* <eftreadCAconfiguration> */
    struct eft_read_phonebook;  /* <eftreadphonebookentry> */
    struct eft_read_logbook;    /* <eftreadlogbook> */
    struct eft_read_access_control; /* <eftreadaccesscontrol> */
}; eft_idu

/***
*   Easyfile: Service specific structure for <ConnectIDU>
***/

/***
*       eft_connect
***/

struct network_address {
    OctetString_type calling_addr;
    OctetString_type calling_subaddr;
    OctetString_type called_addr;
    OctetString_type called_subaddr;
    OctetString_type responding_addr;
    OctetString_type responding_sub_addr;
};


typedef struct {
    struct network_address  net_address;
    Netcon_Type         netcon_type;
    GraphicString_type      net_param;
    GraphicString_type      uuid;
    GraphicString_type      called_name;
}eft_connect;
```

```
/***
*   Easyfile: Service specific structure for <TransparentIDU>
***/


/***
*       eft_transparent
***/

typedef struct {
    OctetString_type    sbv_data;
}eft_transparent;

/***
*   Easyfile: Service specific structure for <DisconnectIDU>
***/


/***
*       eft_disconnect
***/

typedef struct {
    OctetString_Type        net_reason;
    OctetString_Type        net_diag;
    OctetString_Type        net_charge;
}eft_disconnect;



/***
*   Easyfile: Service specific structure for <EstablishIDU>
***/

/***
*       eft_establish
***/

typedef struct {
    INT16       ft_con_id;
    GraphicString_type      calling_identifier;
    GraphicString_type      called_identifier;
    struct functions_supported  functions;
    Boolean_type        recovery;
    Boolean_type        navigation;
    OctetString_type        compression_list;
    OctetString_type        user_data;
    OctetString_type        reason;
}eft_establish;

/***
*   Easyfile: Service specific structure for <ReleaseIDU>
***/

/***
*       eft_release
***/

typedef struct {
    INT16       ft_con_id;
    OctetString_type    user_data;
    OctetString_type    result;
}eft_release;

/***
*   Easyfile: Service specific structure for <AssociateIDU>
***/

/***
*       eft_associate
***/
typedef struct {
    INT16       ft_con_id;
    GraphicString_type      application_name;
    GraphicString_type      ft_calling_addr;
    INT16       timeouts;
    GraphicString_type      calling_identifier;
    GraphicString_type      called_identifier;
    Boolean_type        ident_req;
    OctetString_type        user_data;
    GraphicString_type      ft_called_addr;
    OctetString_type        reason;
}eft_associate;
```

```
/***
*   Easyfile: Service specific structure for <TerminateDU>
***/

/***
*       eft_terminate
***/

typedef struct {
    INT16   ft_con_id;
    OctetString_type    user_data;
    OctetString_type    result;
}eft_terminate;


/***
*   Easyfile: Service specific structure for <AccessIDU>
***/

/***
*       eft_access
***/

typedef struct {
    INT16       ft_con_id;
    Role_value      role;
    struct functions_supported  functions;
    Boolean_type        recovery;
    union {
        OctetString_type user_data;
        struct access_data{
            Domain_support_value    domain_support;
            Boolean_type        navigation;
            OctetString_type        compression_list;
        };
    };
}eft_access;

/***
*   Easyfile: Service specific structure for <EndaccessIDU>
***/

/***
*       eft_endaccess
***/

typedef struct {
    INT16   ft_con_id;
    OctetString_type    user_data;
    OctetString_type    result;
}eft_endaccess;

/***
*   Easyfile: Service specific structure for <AbortDU>
***/

/***
*       eft_abort
***/

typedef struct {
    INT16   ft_con_id;
    OctetString_type    reason;
}eft_abort;


/***
*   Easyfile: Service specific structure for <SendIDU>
***/

/***
*       eft_send
***/

typedef struct {
    INT16       ft_con_id;
    GraphicString_type      designation;
    Boolean_type        recovery_use;
    union {
        OctetString_type user_data;
        struct ft_data{
            Domain_value    domain;
            UINT32      fcs32;
        };
    };
```

```
    union {
        OctetString_type header;
        struct header_data{
            GraphicString_type  filename;
            GraphicString_type  destination_name;
            Compression_type    compression_mode;
            OctetString_type    user_field;
        };
    };
    Source_effect_value     src_effect;
    OctetString_type        result;
}eft_send;

/***
*   Easyfile: Service specific structure for <MessageIDU>
***/

/***
*       eft_message
***/

typedef struct {
    INT16   ft_con_id;
    OctetString_type    message;
}eft_message;

/***
*   Easyfile: Service specific structure for <ReceiveIDU>
***/

/***
*       eft_receive
***/

typedef struct {
    INT16       ft_con_id;
    GraphicString_type      designation;
    Boolean_type        recovery_use;
    union {
        OctetString_type user_data;
        struct ft_data{
            Domain_value    domain;
            UINT32      fcs32;
        };
    };
    union {
        OctetString_type header;
        struct header_data{
            GraphicString_type  filename;
            GraphicString_type  destination_name;
            Compression_type    compression_mode;
            OctetString_type    user_field;
        };
    };
    GraphicString_type      local_filename;
    OctetString_type        result;
}eft_receive;


/***
*   Easyfile: Service specific structure for <DeleteIDU>
***/


/***
*       eft_delete
***/

typedef struct {
    INT16       ft_con_id;
    GraphicString_type      designation;
    union {
        OctetString_type    user_data;
        Domain_value        domain;
    };
    OctetString_type        result;
}eft_delete;
```

```
/***
*   Easyfile: Service specific structure for <RenameIDU>
***/

/***
*       eft_rename
***/

typedef struct {
    INT16       ft_con_id;
    GraphicString_type      designation;
    GraphicString_type      new_name;
    union {
        OctetString_type    user_data;
        Domain_value        domain;
    };
    OctetString_type        result;
}eft_rename;

/***
*   Easyfile: Service specific structure for <CancelDU>
***/


/***
*       eft_cancel
***/

typedef struct {
    INT16   ft_con_id;
    OctetString_type    reason;
}eft_cancel;




/***
*   Easyfile: Service specific structure for <EndIDU>
***/

/***
*       eft_end
***/

typedef struct {
    INT16   ft_con_id;
    OctetString_type    result;
}eft_end;

/***
*   Easyfile: Service specific structure for <ListIDU>
***/

/***
*       eft_list_remote
***/

typedef struct {
    INT16       ft_con_id;
    GraphicString_type      designation;
    list_mode_type      list_mode;
    SortKey_value       sort_key;
    Domain_value        domain;
    GraphicString_type      local_filename;
    OctetString_type        result;
}eft_list_remote;



/***
*   Easyfile: Service specific structure for <NavigateIDU>
***/

/***
*       eft_navigate
***/

typedef struct {
    INT16       ft_con_id;
    GraphicString_type      filestore;
    OctetString_type        result;
}eft_navigate;
/***
*   Easyfile: Local PCI services specific structure for <RedirectionIDU>
***/
```

```
/***
*       eft_redirection
***/

typedef struct {
    INT16   new_LA;
    struct IDU  redirect_IDU;
}eft_redirection;

/***
*   Easyfile: Local PCI services specific structure for <ListLAIDU>
***/
/***
*       eft_list_la
***/

typedef struct {
union Password  list_password;      /* list password */
struct LA_login_name    login_name_list;    /* list of actually logged in LAs */
}eft_list_la;

/***
*   Easyfile: Local PCI services specific structure for <ListLocalDU>
***/
/***
*       eft_list_local
***/

typedef struct {
    GraphicString_type      designation;
    list_mode_type      list_mode;
    SortKey_value       sort_key;
    Domain_value        domain;
    GraphicString_type      local_filename;
    OctetString_type        result;
}eft_list_local;

/***
*   Easyfile: Local PCI services specific structure for <ReadCAIDU>
***/
/***
*       eft_read_capabilities
***/

typedef struct {
    GraphicString_type  CA_capabilities;    /* CA capabilities */
}eft_read_capabilities;
/***
*   Easyfile: Management and service specific structure for <ReadConfIDU>
***/
/***
*       eft_read_configuration
***/

typedef struct {
    GraphicString_type  CA_config;  /* CA configuration */
}eft_read_configuration;
/***
*   Easyfile: Management and service specific structure for <ReadPhonebookfIDU>
***/
/***
*       eft_read_phonebook
***/

typedef struct {
    GraphicString_type      search_key;     /*  */
    phonebook-entry-type    phonebook_entry;    /*  */
    GraphicString_type      filestore;
 }eft_read_phonebook;

/***
*   Easyfile: Management and service specific structure for <ReadLogfIDU>
***/
/***
*       eft_read_logbook
***/

typedef struct {
    GraphicString_type      selection_criteria;     /*  */
    logbook-info-type       logbook_info;       /*  */
    GraphicString_type      local_filename;
 }eft_read_logbook;
```

```
/***
*   Easyfile: Management and service specific structure for <ReadAccessIDU>
***/
/***
*       eft_read_access_control
***/

typedef struct {
    GraphicString_type      search_key;         /*  */
    GraphicString_type      identifier;         /*  */
    authorizations-list     authorizations;     /*  */
    GraphicString_type      working_area;       /*  */
    Boolean_type        navigation_support;
    GraphicString_type      navigation_links;
}eft_read_access_control;
```

11.4.1 EasyfileIDU parameters

```
struct LA_login_name  { /* list of the login names of the actually connected LAs */
    GraphicString_type  value;
    struct LA_login_name    *next;
};

struct functions_supported {    /* list of functions supported */
    Boolean_type    message;
    Boolean_type    send;
    Boolean_type    receive;
    Boolean_type    delete;
    Boolean_type    rename;
    Boolean_type    list;
};


union Password {    /* CHOICE */
    CHAR*   octet_value;
    GraphicString_type  graphicString_value;
};
```

11.4.2 EasyfileIDU values

```
/***
*   Values used for compression mode
***/
typedef enum { basic=4/0, v42b=4/1,gzip=4/2 , ....., applidef=4/15 } Compression_type;

/***
*   Values used for domain
***/
typedef  enum  {
    GroupA=0,
    GroupB,
    GroupC
    }   Domain_value;

/***
*   Values used for domain_support
***/
typedef  enum  {
    GroupA=0,
    GroupAB
    }   Domain_support_value;

/***
*   Values used for list_mode
***/
typedef  enum  {
    simple_list=0,
    extended_list,
    list_filestores,
    sublist_of_filestores,
    presentation_files
    }   ListMode_value;
```

```
/***
*   Values used for Netcon
***/
typedef  enum  {
    ISDN=0
    }   Netcon_value;


/***
*   Values used for Role
***/
typedef  enum  {
    slave=0,
    master
    }   Role_value;


/***
*   Values used for Sort_key
***/
typedef  enum  {
    date=0,
    name,
    size
    }   SortKey_value;

/***
*   Values used for Source effect
***/
typedef  enum  {
    copy_file=0,
    move_file
    }   Source_effect_value;


/***
*   Values used for State Result
***/
typedef  enum  {
    success=0,
    failure
    }   StateResult_value;
```

12 Platform dependencies

12.1 Implementation dependencies

This clause describes the implementation dependencies for the various platforms.

The only platform dependent issue of this recommendation is the implementation of the exchange method.

12.2 General types and definitions

```
/***
*
*   Some values are reserved for the selector parameter: "Easyfile", "SimpleFTAM".
*
***/
/***
*
*   Some values are reserved for the uuid parameter: "none", "admin","default".
*
***/

/***
*   Values used for alarm_type parameter
***/
typedef enum { Error=0, File_transfer_data=1 } Alarm_type;
/***
*   Values used for mode parameter
***/
typedef enum { File_transfer=0 } Mode_type;

/***
*   Values used for function parameter
***/
typedef enum { ft_status=0, net_status=1,ft_infos=2 } Monitor_function_type;

/***
*   Values used for network_status parameter
```

```
***/
typedef enum { idle=0, call_in_progress=1,ringing=2, established=3} network_status_type;
/***
*   Values used for the return code of the IA_Alarm
***/
typedef enum { yes=0, no=1,response_provided=2} status_type;

/***
*   Values used for the wayoftransfer parameter
***/
typedef enum { send=0, receive=1} Way_of_transfer_type;
```

12.3 Windows C function prototypes and definitions

To access the exchange method functions, the CA provider shall offer a Dynamic Link Library (DLL).

The Name of the DLL shall be stated at the appropriate place in the CA description. In order to access the DLL functions, the LA shall first issue a Windows "LoadLibrary" system function call, using the name of the chosen DLL as parameter.

12.3.1 IA_Login function

```
/***
*       IA_Login ()
***/

void FAR PASCAL IA_Login(
    LPSTR   login_name,     /* User name */
    LPSTR   password,       /* User's password */
    LPSTR   selector,       /* CA selector */
    int far *   ca_id,      /* CA Identifier */
    int far *   connection_id,      /* Connection ID */
    int far *   status,     /* 0 success, 1 fail */
    WORD    mode,       /* Mode */
    LPSTR   uuid,       /* UUID */
    FARPROC monitor_handler,    /* Monitor handler entry */
    FARPROC alarm_handler,      /* Alarm handler entry */
    int alarm_event,        /* Alarm event , combination of alarm type */
);
```

12.3.2 IA_Monitor function

```
/***
*       IA_Monitor ()
***/

int FAR PASCAL IA_Monitor (
    int     connection_id,      /* Connection ID */
    int     ca_id,      /* CA Identifier */
    WORD    function,       /* Function type: network status, file transfer
    status, file transfer informations*/
    void far *  monitor_data,       /* additional parameter buffer */
);

struct ft_status {
    WORD    protocol_information;
    WORD    currentoperation;
    WORD    currentparameter;
    DWORD   elapsedtime;
};

struct ft_infos {
    DWORD   bytespersecond;
    DWORD   currentindex;
    DWORD   currentsize;
    UINT    currentpercent;
    DWORD   currenttransferred;
    UINT    numfiles;
    UINT    totalpercent;
    DWORD   totalsize;
    DWORD   transferseconds;
    LPSTR   sourcefile;
    LPSTR   destfile;
    LPSTR   filedescription;
    BOOL    wayoftransfer;  /* 0 send, 1 receive */
};
```

```
struct network_status {
    WORD    network_status;
    LPSTR   network_information;
}

typedef struct {
    WORD    netcon_id;
    union {
        struct network_status       net;
        struct ft_status            ft;
        struct ft_infos         tf;
    }infos;
}IA_MONITOR_DATA;
```

12.3.3 IA_Alarm function

```
/***
*       IA_Alarm ()
***/

int FAR PASCAL IA_Alarm (
    int connection_id,      /* Connection ID */
    int ca_id,      /* CA Identifier */
    WORD    alarm_type,     /* Alarm type */
    LPSTR   alarm_data,     /* additional parameter buffer */
    WORD    alarm_length        /* length of parameter buffer */
);
typedef struct {
    WORD    error_type;
    LPSTR   error_description
}Error;


typedef union {
    struct IDU  IDU;
    struct Error    Error;
}IA_ALARM_DATA;
```

12.3.4 IA_Logout function

```
/***
*       IA_Logout ()
***/

void FAR PASCAL IA_Logout(
    int connection_id,      /* Connection ID */
    int ca_id,
    int far *   status      /* 0 success, 1 fail */
    );
```

12.3.5 IA_Send function

```
/***
*       IA_Send
***/


VOID    IA_Send {
    int     connection_id,
    int ca_id,
    int far *   status,     /* 0 success, 1 fail */
    struct  IDU IDU
};
```

12.4 UNIX

For further study.

12.5 OS/2

For further study.

12.6 MacOS

For further study.

Annex A (normative): Error codes

Table A.1 states the error codes which shall be used commonly by CA implementations conforming to this ETS.

This errors codes shall be conveyed with the status parameter of the exchange mechanism.

Table A.1: List of error codes

Error code	Meaning	Comment
0000	Success	No error - successfull operation.
0001-4999	*Private use*	Reserved for CA private use.
5000	CA Exchange Method Error	
5001	Unsupported service specified	
5002	Invalid "selector"	
5003	Invalid "mode"	
5004	Invalid "connection_id"	
5005	Invalid "ca_id"	
5006	Alarm handler not specified	
5011-5499	*Reserved*	Reserved *for further study.*
5500-5999	*Private use*	Reserved for CA private use.
6000	IDU Syntax Error	
6001	Unknown Function requested	
6002	IDU is empty	
6003	IDU Header is invalid	Invalid length or contents.
6010	Mandatory parameter is missing	
6011	Undefined parameter	
6012	Parameter length out of range	
6013	Parameter invalid	
6020	Invalid "IDU_ref"	
6021	Invalid "netcon_id"	
6022	Invalid "sp_id"	
6023	Invalid "service_id"	
6024-6499	*Reserved*	Reserved *for further study.*
6500-6999	*Private use*	Reserved for CA private use.
7000	Hardware/System related Error	
7001	CA not available	CA is no longer available.
7002	CA busy	CA is unable currently to process the request.
7003	Number of LAs exceeded	CA can support no more LAs.
7004-7499	*Reserved*	Reserved *for further study.*
7500-7999	*Private use*	Reserved for CA private use.
9000	Transmission Error	
9012-9499	*Reserved*	Reserved *for further study.*
9500-9999	*Private use*	Reserved for CA private use.
NOTE: Error codes are given in decimal.		

Annex B (informative): File Transfer Protocol (FTP) service definition

This annex specifies the IDUs and the parameters which shall be used to support FTP. A mapping of IDU and FTP commands is provided. FTP is specified with the RFC 959.

B.1 FTP IDUs

To cover the FTP file transfer protocol the following IDUs are used:

Table B.1

Service Type	FTP RFC 959 reference	Usual FTP commands	Purpose
establishment of a network connection		open	establish the TCP connection from the user to the server.
release of a network connection	quit, abort	close, bye, quit, disconnect	release the TCP connection with the server.
establishment of a file transfer connection.	user name and password	----	Telnet string identifying the user.
termination of a file transfer connection	reinitialize	----	close the account and all i/o with the FTP server.
abort of a file transfer connection	abort	----	
send file	store	put, mput, send	send a file to the remote entity.
receive file	retrieve	get, mget, recv	receive a file from the remote entity.
delete file	delete	delete, mdelete	delete a file on the remote filesystem.
rename file	rename from, rename to	rename	rename a file on the remote filesystem.
cancel a file operation		CTRL-C	cancel a send or receive file operation.
end of file operation			indication of the end of a file transfer operation for the slave.
list remote filestore information's	list	ls, mls, dir, mdir	list the remote files or the remote filestores of the filesystem.
navigation on the remote filestore	cdup	cd, lcd	select a remote filestore.

B.2 FTP specific IDU parameters

Tables B.2 to B.4 summarize the FTP specific IDU parameters and, their meaning.

Table B.2

Address information related parameters		
Parameter	**Abbreviation**	**Purpose**
host name	host_name	host name or IP address.

Table B.3

Protocol related parameters		
Parameter	**Abbreviation**	**Purpose**
filename	filename	transfer name.
destination name	destination	destination filename.
local filename	local_name	local filename of the transfer name to be received.

Table B.4

PCI specific parameters		
Parameter	**Abbreviation**	**Purpose**
file transfer connection identifier	ft_con_id	unique identification of the file transfer connection.
network connection type	netcon_type	type of the network connection which shall be used; for the time being there is only one valid network selectable at the PCI (ISDN).
CA identification	ca_id	unique identification of a CA (see also subclause 7.2).
new LA identifier	new_LA	unique identification of the LA which shall take the incoming call.
redirect IDU	redirect_IDU	the IDU, carrying the incoming call data from a remote system, which shall be redirected.
source effect	src_effect	action which shall be performed after transmission of a local file: delete the file after successful transmission (move) or store the file on the local virtual filestore (copy).

B.3 Network connection services

B.3.1 Establishment of a network connection

The establishment of the network connection makes use of the FTP open command.

FTP may use the following specific parameters.

Table B.5

Parameters	Purpose
host_name	host name or IP address.

Table B.6 shows the use of the parameters related to the service primitives:

Table B.6

Parameter	Request	Indication	Response	Confirmation
host_name	m	-	-	-

B.3.2 Release of a network connection

The release of a network connection make use of the FTP close command.

Table B.7

Parameter	Purpose
net_diag	additional information from ISDN B channel after progression of the requested network connection release.

Table B.8 shows the use of the parameters related to the service primitives:

Table B.8

Parameter	Request	Indication	Confirmation
net_diag	-	o	o

B.4 File transfer connection services

B.4.1 Establishment of a file transfer connection

This command help to identify the application towards the remote entity.

Table B.9

Parameter	Purpose
ft_con_id	unique identification of the file transfer connection.

Table B.10 shows the use of the parameters related to the service primitives:

Table B.10

Parameter	Request	Indication	Response	Confirmation
ft_con_id	m	m	m	m

B.4.2 Termination of a file transfer connection

This IDU is used to end the file transfer session.

Table B.11

Parameters	Purpose
ft_con_id	unique identification of the file transfer connection.

Table B.12 shows the use of the parameters related to the service primitives:

Table B.12

Parameter	Request	Indication	Response	Confirmation
ft_con_id	m	m	m	m

B.4.3 Abort a file transfer connection

This IDU is used to abort all the currently file transfer established regimes.

Table B.13

Parameters	Purpose
ft_con_id	unique identification of the file transfer connection.

Table B.14 shows the use of the parameters related to the service primitives:

Table B.14

Parameter	Request	Indication	Confirmation
ft_con_id	m	m	m

B.5 File transfer services

B.5.1 Send File

This IDU is used to send a file to the remote entity and is mapped to the FTP put command.

Table B.15

Parameters	Purpose
ft_con_id	unique identification of the file transfer connection.
filename	file to be transmitted.
result	response: indicate acceptance of the file transfer; confirmation: indicate end of the file transfer operation.

Table B.16 shows the use of the parameters related to the service primitives:

Table B.16

Parameter	Request	Indication	Response	Confirmation
ft_con_id	m	m	m	m
filename	m	m	-	-
result	-	-	m	m

B.5.2 Receive File

This IDU is used to receive a file from the remote entity and is mapped to the get FTP command.

Table B.17

Parameters	Purpose
ft_con_id	unique identification of the file transfer connection.
filename	file to be received.
local_filename	local filename of the transfer name to be received.
result	response: indicate acceptance of the file transfer; confirmation: indicate end of the file transfer operation.

Table B.18 shows the use of the parameters related to the service primitives:

Table B.18

Parameter	Request	Indication	Response	Confirmation
ft_con_id	m	m	m	m
designation	m	m	-	-
local_filename	o	-	-	-
result	-	-	m	m

B.5.3 Delete File

This IDU is used to delete a file from the remote entity and is mapped to the FTP delete command.

Table B.19

Parameters	Purpose
ft_con_id	unique identification of the file transfer connection.
filename	transfer name to be deleted.
result	result of the operation.

Table B.20 shows the use of the parameters related to the service primitives:

Table B.20

Parameter	Request	Indication	Response	Confirmation
ft_con_id	m	m	m	m
filename	m	m	-	-
result	-	-	m	m

B.5.4 Rename file

This IDU is used to rename a file from the remote entity and is mapped tpo the FTP rename command.

Table B.21

Parameters	Purpose
ft_con_id	unique identification of the file transfer connection.
filename	transfer name to be renamed.
new_name	new name.
result	result of the operation.

Table B.22 shows the use of the parameters related to the service primitives:

Table B.22

Parameter	Request	Indication	Response	Confirmation
ft_con_id	m	m	m	m
filename	m	m	-	-
new_name	m	m	-	-
result	-	-	m	m

B.5.5 Cancel File Operation

This IDU is used to cancel a send or receive file operation.

Table B.23

Parameters	Purpose
ft_con_id	unique identification of the file transfer connection.

Table B.24 shows the use of the parameters related to the service primitives:

Table B.24

Parameter	Request	Indication
ft_con_id	m	m

B.5.6 End of File Operation

This IDU is used to indicate to the slave the end of a file operation.

Table B.25

Parameters	Purpose
ft_con_id	unique identification of the file transfer connection.
result	result of the file transfer operation.

Table B.26 shows the use of the parameters related to the service primitives:

Table B.26

Parameter	Indication
ft_con_id	m
result	m

B.5.7 List remote filestore information

This IDU is used to get a list of remote files or remote filestores and is mapped to the FTP ls command.

Table B.27

Parameters	Purpose
ft_con_id	unique identification of the file transfer connection.
sort_key	identifies the key according to which the requested list shall be sorted; possible values: date(0), name(1), size(2).
designation	criteria to select a list.
local_filename	the local file recipient of the remote list of files.
result	response: indicate acceptance of the file transfer; confirmation: indicate end of the file transfer operation.

Table B.28 shows the use of the parameters related to the service primitives:

Table B.28

Parameter	Request	Indication	Response	Confirmation
ft_con_id	m	m	m	m
designation	m	m	-	-
sort_key	m	-	-	-
local_filename	m	-	-	-
result	-	-	m	m

B.5.8 Navigation on the remote filestore

This IDU is used to select a remote filestore and mapped to the FTP cd command.

Table B.29

Parameters	Purpose
ft_con_id	unique identification of the file transfer connection.
filestore	remote filestore name.
result	result of the operation.

Table B.30 shows the use of the parameters related to the service primitives:

Table B.30

Parameter	Request	Indication	Response	Confirmation
ft_con_id	m	m	m	m
filestore	m	m	-	-
result	-	-	m	m

Annex C (informative): Examples of information exchange

This informative annex gives an example of the use of the PCI.

An LA-running on a Windows based system - wants to send a document, say the document "c:\doc\document.doc" to an address via Easyfile services. What the LA shall do in sequence is:

- look for a CA which is capable of performing the Easyfile service by inspecting the manufacturer capability description and, if found, perform a IA_Login to that CA by a IA_Login function call (if not already done),

- build a ConReq IDU;

- transmit the IDU to the CA with the IA_Send function;

- get informed by the alarm function until the confirmation IDU becomes available;

- build a EstabReq IDU;

- transmit the IDU to the CA with the IA_Send function;

- get informed by the alarm function until the confirmation IDU becomes available;

- build a SendReq IDU;

- transmit the IDU to the CA with the IA_Send function;

- get informed by the alarm function until the confirmation IDU becomes available;

- perform other functions with the same CA or proceed to the release of the communication link and the logout.

Let us assume that the LA has already logged into the CA.

Table C.1: Establish the communication link request

Parameter	Value	Comments
ca_i	EasyfileCA	
connection_id	my_connection	
IDU_ref	g_0815	IDU reference, generated by LA.
netcon_id	temp_value	temporary value.
service_id	IA_FT_CONNECT	
sp_id	REQUEST	
called_address	1234567890	ISDN address of the server.
netcon_type	ISDN	type of network.
status		status of the operation (Response).

Once the IDU is prepared, the LA internally calls the exchange mechanism function IA_Send which hands over the IDU to the CA.

The LA wait to get informed by the alarm function until the confirmation IDU becomes available.

The alarm will look like:

Table C.2: Alarm function

Parameter	Value	Comments
CA-Id	EasyfileCA	
connection_id	my_connection	
alarm_type	ft_data	
alarm_data		IDU.
IDU_ref	g_0815	IDU reference, generated by LA.
netcon_id	netcon_value	
service_id	IA_FT_CONNECT	
sp_id	CONFIRMATION	
status	YES	LA response (yes, no, response).

The LA will answer positively and will establish the file transfer connection.

Table C.3: Establish the file transfer connection

Parameter	Value	Comments
CA-Id	EasyfileCA	
connection_id	my_connection	
IDU_ref	g_0816	IDU reference, generated by LA.
netcon_id	netcon_value	
service_id	IA_FT_ESTABLISH	
sp_id	REQUEST	
ft_con_id	file_transfer_reference	
status		status of the operation (Response).

The LA will wait to get informed by the alarm function until the confirmation IDU becomes available.

The alarm will look like:

Table C.4: Alarm function

Parameter	Value	Comments
CA-Id	EasyfileCA	
connection_id	my_connection	
alarm_type	ft_data	
alarm_data		IDU.
IDU_ref	g_0816	IDU reference, generated by LA.
netcon_id	netcon_value	
service_id	IA_FT_ESTABLISH	
sp_id	CONFIRMATION	
ft_con_id	file_transfer_reference	
status	YES	LA response (yes, no, response).

The LA will answer positively and will send the file.

Table C.5: Send the file

Parameter	Value	Comments
CA-Id	EasyfileCA	
connection_id	my_connection	
IDU_ref	g_0817	IDU reference, generated by LA.
netcon_id	netcon_value	
service_id	IA_FT_SEND	
sp_id	REQUEST	
ft_con_id	file_transfer_reference	
designation	document.doc	
filename	c:\doc\document.doc	
status		status of the operation (Response).

Once the file transfer is completed the CA will wake up the LA through the alarm function. In between, the LA may receive monitoring events to indicate the progression of the file transfer.

History

Document history			
October 1997	Public Enquiry	PE 9805:	1997-10-03 to 1998-01-30
June 1998	Vote	V 9836:	1998-06-29 to 1998-09-11
September 1998	First Edition		

ISBN 2-7437-2533-8
Dépôt légal : Septembre 1998