INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

T.611

(11/94)

TERMINALS FOR TELEMATIC SERVICES

PROGRAMMING COMMUNICATION INTERFACE (PCI) APPLI/COM FOR FACSIMILE GROUP 3, FACSIMILE GROUP 4, TELETEX, TELEX, E-MAIL AND FILE TRANSFER SERVICES

ITU-T Recommendation T.611

(Previously "CCITT Recommendation")

FOREWORD

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

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

The approval of Recommendations by the Members of the ITU-T is covered by the procedure laid down in WTSC Resolution No. 1 (Helsinki, March 1-12, 1993).

ITU-T Recommendation T.611 was revised by ITU-T Study Group 8 (1993-1996) and was approved under the WTSC Resolution No. 1 procedure on the 11th of November 1994.

____

NOTE

In this Recommendation, the expression “Administration” is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

CONTENTS

SUMMARY

This Recommendation as approved in 1992 defines a Programming Communication Interface called “APPLI/COM”, providing unified access to Telefax Group 3, Telefax Group 4, telex and teletex services.

Recommendation T.611 has been revised for clarity and to include E-mail services in general, Message Handling Systems (MHS), as described in the ITU-T X.400-series of Recommendations, in particular and File Transfer Services.

Special consideration has been taken to assure backwards compatibility with the 1992 version of this Recommendation.

Recommendation T.611

PROGRAMMING COMMUNICATION INTERFACE (PCI) APPLI/COM FOR FACSIMILE GROUP 3, FACSIMILE GROUP 4, TELETEX, TELEX, E-MAIL AND FILE TRANSFER SERVICES

(revised 1994)

PART I – GENERAL DESCRIPTION

1 Scope

This Recommendation defines the Programming Communication Interface (PCI) called "APPLI/COM".

The general concepts of PCIs are defined in the ITU-T Recommendation F.581 (1992).

This PCI provides unified access to different communications services, such as telefax group 3 or other telematic services.

This Recommendation currently specifies the access to the following telematic services:

– Telefax group 3;

– Telefax group 4;

– Teletex;

– Telex[1];

– E-Mail services;

– File transfer services.

The principle of this interface is to exchange messages between two entities (i.e. the Local Application and the Communication Application). This Recommendation describes the structure, contents of those messages and the way to exchange them. Two message encoding schemes are defined: text based and binary structured.

Implementing this PCI is not required to participate in a telematic service. However, use of this interface will provide binary compatibility between the above mentioned entities. As a result end-users will benefit from plug-in compatibility between products coming from different manufacturers.

This Recommendation provides a framework for future extensions that keep functionalities consistent and upward compatible.

To summarize, this Recommendation forms a high level API (Application Programming Interface) which shields all telecommunication peculiarities but gives powerful control and monitoring on the telecommunication activity to the application designers.

2 Definitions and References

2.1 Interface Definitions

The following definitions apply to the interface.

2.1.1 communication application (CA); A Communication Application (CA) is provided by hardware and/or software allowing to participate in standardized telecommunications services. The telecommunications services supported by this Recommendation are listed in clause 1.

[1] The access to the telex service does not include the dialogue facility.

2.1.2 local application (LA): A Local Application (LA) is an application able to generate files or documents and possibly able to manage communications dialogues. Such LAs can be word processors, spread sheets, graphic editors, file editors, etc. The LA shall generate messages conforming to this Recommendation.

2.1.3 task data description (TDD): A Task Data Description (TDD) describes the structure of the messages exchanged between a LA and a CA (not the way to effectively exchange them). A Request TDD describes an exchange originated by a LA towards a CA. A Response TDD describes an exchange originated by a CA towards a LA.

2.1.4 exchange method (EM): The Exchange Method (EM) describes how the TDDs are exchanged between a CA and a LA. This Recommendation defines a generic Exchange Method which shall be adapted to the target operating system by the means described in this Recommendation. Local area and wide area network environments are supported by the generic Exchange Method.

2.1.5 CA-descriptor: The CA-Descriptor is a collection of information concerning the use of a CA. The CA-Descriptor describes the facilities and features of a given CA so that any LA that uses that CA knows how to proceed. The CA-Descriptor is a component of the Interface Configuration Environment (ICE).

2.1.6 interface configuration environment (ICE): The Interface Configuration Environment (ICE) is composed of the Master ICE and related CA-Descriptors. The Master ICE lists all CAs that can be reached from within a given LA. The aim of the ICE is to assist the LA in the selection of an appropriate CA with respect to the LA requirements.

2.2 File Definitions

This Recommendation deals with various kind of files: those interchanged between a LA and a CA and those exchanged through a telecommunications network. The following definitions are used throughout the body of this Recommendation.

2.2.1 transfer files: As shown in Figure 1 the Transfer Files are those files exchanged between a LA and a CA. The format of these files (Transfer Format) is further defined in this Recommendation.

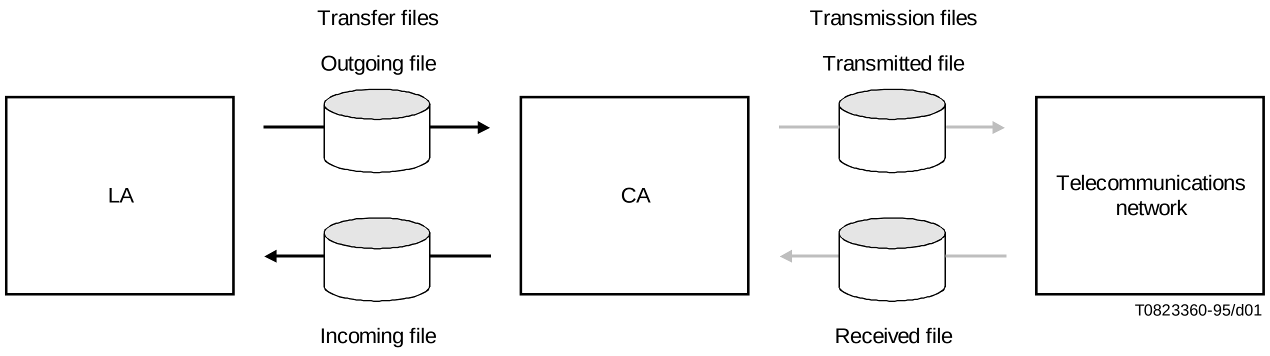


FIGURE 1/T.611

The distinct types of files defined by APPLI/COM

2.2.1.1 outgoing file: The Outgoing File is a file that the LA transmits to the CA in order to be transmitted by the CA through a telecommunications network. The format of the file is from one of the possible Transfer Formats.

2.2.1.2 incoming file: The Incoming File is a file transmitted to the LA by the CA. It generally corresponds to files received from the network. The format of the file is from one of the possible Transfer Formats.

2.2.1.3 transfer format: The Transfer Format defines the structure of the Transfer Files. Depending on the telecommunication services used, some Transfer Formats are more suitable than others. The possible Transfer Formats are defined in clause 8. A Transfer Format is identified by the Convert-ID. The identification and selection of a Transfer Format is further described in 5.4.5.

2.2.2 transmission files: The Transmission Files are files exchanged by a CA through the network. Figure **Error! Bookmark not defined.** depicts this. The format of these files, the Transmission Format, is defined intrinsically by the telecommunications service used.

2.2.2.1 transmitted file: The Transmitted File is a file sent by the CA across a telecommunications network in a format suitable for exchange by the protocol used in the telecommunications service.

2.2.2.2 received file: The Received File is a file built by the CA from information received through the telecommunications network in the format used for exchange by the protocol used in the telecommunications service.

2.2.2.3 transmission format: The Transmission Format defines the structure of the Transmission Files depending on the telecommunications service used. The Transmission Format is identified by the Type-ID (see 5.4.6).

2.3 References

This Recommendation refers to the following ITU-T Recommendations:

– Recommendation F.59, *General characteristics of the international telex service.*

– Recommendation F.60, *Operational provisions for the international telex service*.

– Recommendation F. 160, *General operational provisions for the international public facsimile services*.

– Recommendation F. 184, *Operational provisions for international facsimile service between subscriber stations with group 4 machines (telefax group 4)*.

– Recommendation F. 200, *Teletex Service.*

– Recommendation F. 581, *Service Recommendation for Programming Communication Interfaces*.

– Recommendation S.1, *International telegraph alphabet No. 2*.

– Recommendation T.4, *Standardization of group 3 facsimile apparatus for document transmission*.

– Recommendation T.6, *Facsimile coding schemes and coding control functions for group 4 facsimile apparatus*.

– Recommendation T.30, *Procedures for document facsimile transmission in the general switched telephone network*.

– Recommendation T.35, *Procedure for the allocation of ITU members codes*.

– Recommendation T.50, *International Reference Alphabet*.

– Recommendation T.51, *Latin Alphabet coded character sets for Telematic services*.

– Recommendation T.52, *Non-Latin coded character sets for Telematic services*.

– Recommendation T.61, *Character repertoire and coded character sets for the international Teletex service*.

NOTE – Recommendation T.61 has been superseeded by Recommendation T.50-Series since 1993.

– Recommendation T.62, *Control procedures for Teletex and group 4 facsimile services*.

– Recommendation T.434, *Specification of Binary File Transfer (BFT).*

– Recommendation T.500, *General overview of the T.500-Series of Recommendations*.

– Recommendation T.563, *Terminal characteristics for group 4 facsimile apparatus*.

– Recommendation T.571, *Terminal characteristics for the Telematic File Transfer within the facsimile group 4 and teletex apparatus*.

– Recommendation X.208, *Specification of Abstract Syntax Notation one (ASN.1)*.

– Recommendation X.209, *Specification of Basic Encoding Rules for Abstract Syntax Notation one (ASN.1).*

– Recommendation X.400, *Series of ITU-T Recommendations for Message Handling Systems*.

It also refers to the following International Standards:

– ISO/IEC 639.2, *Code for the presentation of code of languages*.

– SO/IEC 9735: 1990, *Electronic Data Interchange For Administration Commerce and Transport (EDIFACT) – Application level syntax rules.*

2.4 Abbreviations and Acronyms

API	Application Programming Interface
APPLI/COM	An Interface between Local Applications and Communication Applications
ASCII	American Standard Codes for Information Interchange
ASN.1	Abstract Syntax Notation one as defined per ITU-T Recommendations X.208 and X.209
BFT	Binary File Transfer as defined per ITU-T Recommendation T.434
BNF	Backus Naur Form
BTM	Basic Transparent mode as defined per ITU-T Recommendation T.434
CA	Communication Application
CIL	Call Identification Line
CR	Carriage return
CTL	Control Document; document type as defined per ITU-T Recommendation T.62
DCX	A multipage PCX file
DLL	Dynamic Link Library
DRF	Dispatch Received Files
DTM	Document Transparent mode as defined per ITU-T Recommendation T.434
E-Mail	General used term for Electronic Mail services
EBCDIC	Extended Binary Coded Data Interchange Code
EDI	Electronic Data Interchange as defined per ITU-T Recommendation T.571
EM	Exchange Method
EMail	Service-ID for E-Mail services

ESC	Escape
FC	Functional Class
FCA	Functional Class A
FCB	Functional Class B
FF	Form Feed
FT	Service-ID for File Transfer services
FX3	Service-ID for Telefax group 3 services
FX4	Service-ID for Telefax group 4 services
ICE	Interface Configuration Environment
IPM	Interpersonal Message – Message exchanged between users via UA (X.400 Terminology)
IRA	International Reference Alphabet (T.50)
IRV	International Reference Version (T.50)
ISDN	Integrated Services Digital Network
LA	Local Application
LAN	Local Area Network
LF	Line Feed
MD	Monitor Document; document type as defined per ITU-T Recommendation T.62
MH	Modified Hufman code; defined per ITU-T Recommendation T.4
MHS	Message Handling System as defined per ITU-T X.400 Series of Recommendations
MR	Modified Read code; defined per ITU-T Recommendation T.4
MS	Message Store (X.400 Terminology)
MTA	Message Transfer Agent (X.400 Terminology)
MTS	Message Transfer System (X.400 Terminology)
O/R-Name	Originator/Recipient name (X.400 Terminology)
OPD	Operator Document; document type as defined per ITU-T Recommendation T.62
P1	MTS Transfer Protocol; protocol defining the MHS envelope for MTA-MTA exchange as defined per ITU-T Recommendations X.411 (1988) and X.419 (1988, abstract syntax definition)
P2	Interpersonnal Messaging Format (IPM), defined per ITU-T Recommendation X.420 (1984, or as P22 1988)
P3	MTS Access Protocol, defined per ITU-T Recommendation X.419 (1988)
P7	MS Access Protocol, defined per ITU-T Recommendation X.413 (1988)
PCI	Programmable Communication Interface
PCX	A variety of raster information file used on personal computer software

P_{edi}	EDI user to user exchange format for X.400, defined per ITU-T Recommendations F.435 and X.435
PSTN	Public Switched Telephone Network
SP	Space
TDD	Task Data Description
TFT	Telematic File Transfer as defined per ITU-T Recommendation T.571
TIFF	Tagged Image File Format
TLX	Service-ID for Telex services (without dialogue facility)
TTX	Service-ID for Teletex services
TX	Service-ID for Telex via Teletex services
UA	User Agent (X.400 Terminology)
WAN	Wide Area Network

2.5 Operating Systems

The versions of the operating systems mentioned in this Recommendation are:

MacOS	All versions
MS-DOS	Version 3.1 or higher
UNIX	All versions
Windows	Version 3.0 or higher
OS/2	Version 1.0 or higher

Throughout this Recommendation the term DOS stands for MS-DOS compatible operating system versions.

2.6 Trade Marks

The following names are registered trade marks and belong to their respective holders:

MacOS	Apple Computer, Inc.
MS-DOS	Microsoft Corporation
UNIX	AT&T
TIFF	Aldus Corporation
Windows	Microsoft Corporation
OS/2	International Business Machines Corporation

3 Structure of this Recommendation

This Recommendation covers:

– Interface requirements;
– Definitions;
– General principles;
– Application features;
– Classes of implementation;
– Task Data Description (TDD);
– Exchange Method;
– Incoming and outgoing files;
– Communications control – CA-Record.

3.1 Extensions to this Recommendation

To accommodate the evolution of telecommunication technologies, this Recommendation has been structured so as to accept new telecommunication services, or new options of existing telecommunication services without destroying the overall outline.

In addition, this Recommendation provides extension mechanisms to enhance:

- the exchange method between LAs and CAs;
- the messages (TDDs);
- the transfer formats,

while preserving compatibility with the baseline Recommendation.

4 General Principles

The interface is guided by the following principles:

- Be independent of computer hardware (for instance, the interface can be supported by a communications hardware or software).
- Be independent of operating systems and programming languages (the Exchange Method is the only part that depends on operating systems and programming languages). This Recommendation provides directives to achieve compatibility on the MS-DOS, OS/2, Windows and Unix operating systems. Other operating systems will be considered on demand.
- The formal description of interface messages is based on a BNF-like[2)] description. Various coding schemes may be used to present the interface messages.
- Be task submission oriented.
- Take into account the demand for multiple applications running on the same server as well as LAN/WAN applications. The interface can be used when several local applications and/or several communication applications are involved.
- Be extensible and flexible.

4.1 Model

The LA-CA interaction uses the client-server model as shown in Figure 2 below.

In this context:

- the CA is considered as a server providing some telecommunications services to the LA;
- the LA is considered as the client of a CA using the telecommunications services provided by this CA.

Consequently:

- as a server, the CA must comply with one of the two Functional Classes defined in clause 10, and with the Exchange Methods defined in this Recommendation;
- as a client, no constraint applies on the LA, with respect to the interface.

As a result, the initiation of information exchange between LA and CA always belongs to the LA. The LA may choose not to be disturbed in its local work by possible events coming from the CA.

4.1.1 Role of the LA

Regarding the interface, the LA may be functionally separated into two parts:

- the software which generates the outgoing files and/or read the incoming files;
- the software which manages the communication.

[2)] BNF Backus Naur Form.

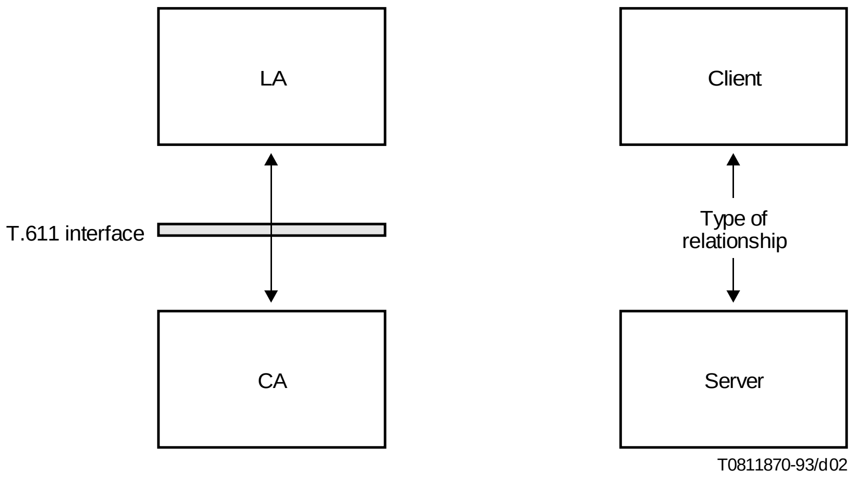


FIGURE 2/T.611

The relationship between LA and CA is modelled on the client-server model

The latter provides:

1) Man-machine dialogues or automatic processing for sending the outgoing files, handling the incoming files (display, print out, save), monitoring of CA activity, and requesting for particular system and/or service management action.

2) Conversion of documents from/to a Transfer Format suitable for the CA.

3) Access to optional CA features as stated in the CA-Descriptor of the ICE.

NOTE – The LA, as a client, is not obliged to use all the CA features.

4.1.2 Role of the CA

The CA, as a server, is in charge of:

– the management of communications;

– the conversion of file formats from the Transfer Format to the Transmission Format (and vice versa);

– the management of CA features (if any) described in the CA-Descriptor of the ICE.

4.2 Information Exchange

Information to be accessed through the interface comprises:

– Outgoing Files: files or documents handed by the LA to the CA in order to be transmitted by the CA on a network.

– Incoming Files: files or documents handed by the CA to the LA after their reception by the CA from the network.

– CA-Records: groups of information that record any transmission and reception events managed by the CA.

Information exchanged between CAs and LAs are carried out by Request TDDs and Response TDDs which fall in one of the groups shown in Table 1. The table lists all TDD groups defined in this Recommendation. It specifies in particular whether a Response TDD is generated by the CA for a given Request TDD. The TDDs themselves are described in detail in clause 6.

TABLE 1/T.611

TDD Groups

TDD Group	Response?	Comment
Send	Yes if request requires one	Asks the CA to send one or more files through the network, using one telecommunications service, to a set of recipients
Receive	Yes	Asks the CA to retrieve an incoming file resulting from a received file. The Response TDD is filled with the incoming file location and the transfer format
Trace	Yes	Asks the CA to perform some action on one or a set of CA-Record(s) belonging to a specific state. The action to be performed is described inside the Trace request
Submit	Yes	Asks the CA to perform an action like converting a file to a specific format, printing out a file according to telecommu-nications service specific rules
Extend	Yes	Provides the possibility for extensions to the Recommenda-tion. Can be implemented as formal changes to the Recom-mendation, as national or as private extensions

Proper operation of the interface requires the detailed description of a data definition and an interaction specification between the LA and the CA. Exchange Methods may coexist depending on the particular needs of different environments. Two methods of information exchange are defined (see clause 7):

– one is a file exchange method which may be easily implemented over a wide range of computers and operating systems but provides lower computer throughput;

– the other is a primitive exchange method using operating system dependent mechanisms to transport information. It provides higher throughput but reduced portability.

Therefore, the concept of Task Data Description (TDD) is introduced (see Figure 3). A TDD is an abstract data structure interchanged between LA and CA, describing a particular task that the CA must carry out or describing a CA response to one of these tasks.

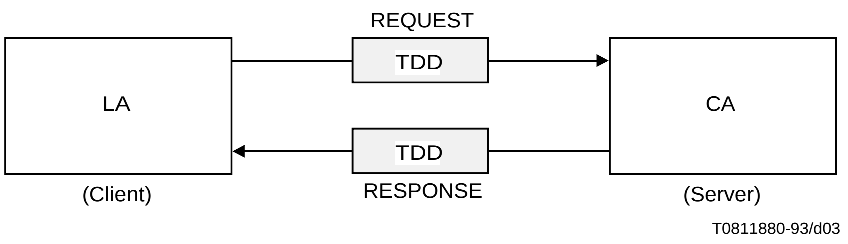


FIGURE 3/T.611
Task Data Description (TDDs)

The TDDs are independent of the information exchange method used, in order to:

– simplify the testing procedures;

– leave the choice of the TDD encoding method open.

A TDD carries information about the task the LA expects the CA to do, plus all the appropriate parameters. Since the communication of these tasks follows a "client-server" model, two kinds of TDDs are actually exchanged:

– a Request TDD generated by the LA toward the CA, describing the action to be accomplished;

– a Response TDD generated by the CA toward the LA, describing the response to a previous request.

By construction, a LA may send many Request TDDs without waiting for the corresponding Response TDDs. Some requests are not expecting responses at all.

A CA may manage Request TDDs in any order.

4.2.1 TDD Coding

As described in more detail in 6.1, the TDDs may be encoded with different schemes, which are either text based or binary structured.

4.2.2 Format of Exchanged Data

The format data are exchanged between LAs and CAs, namely the format of Outgoing and Incoming Files, shall conform to well defined coding and presentation rules. This rules are called Transfer Formats. The Transfer Formats are defined in detail in clause 8.

4.3 Interface Configuration Environment (ICE)

The Interface Configuration Environment (ICE) concept is used to identify the services offered by CAs within a computing environment and the characteristics of each individual CA. The ICE structure has two levels :

1) The Master-ICE is a universally accessible file which lists all available CAs. It states how to gain access to the detailed configuration information for each CA.

2) The CA-Descriptor is the source of the detailed configuration information about each CA. Access to the CA-Descriptor information may be in the form of a file or be generated via dynamic methods tailored to the computing environment such as executable files, Dynamic Link Libraries (DLLs) or other approaches.

Each CA has its own unique features. The CA-Descriptor provides access to a complete list of those features. The combination of the Global ICE and CA-Descriptors provides a standardized means for LAs to access original and/or extra features of CAs. Figure **Error! Bookmark not defined.** depicts the relation between the Master ICE and the CA-Descriptors.

The ICE represents a global source for all LAs conforming to this Recommendation. The Master ICE shall contain the following header information for each CA:

– the Services supported by the CA;

– the Exchange Method(s) supported by the CA;

– one or more Access methods for each CA-Descriptor.

The CA-Descriptor contains at the minimum the following set of interface related configuration information:

– the Exchange Method used to interchange TDDs between LAs and the CA;

– details of the Exchange Method (paths, buffers, entry points);

– TDD coding;

– functional Class;

– supported telecommunications services;

– CA facilities.

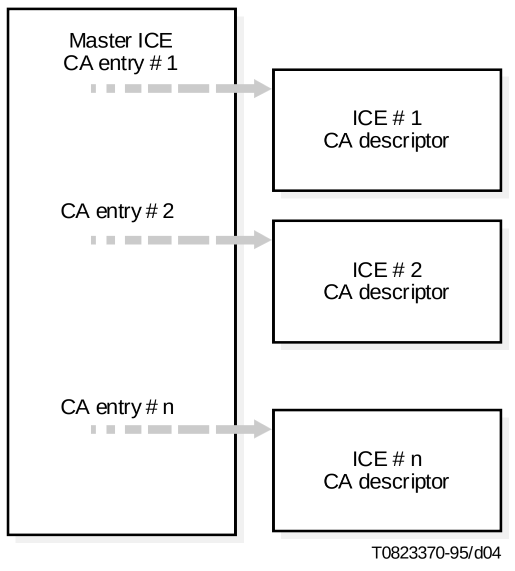


FIGURE 4/T.611

Relation between Master ICE and CA-Descriptor

Provision of the CA-Descriptor is mandatory for any CA. Any LA may rely on information included in the Master ICE or CA-Descriptor. LAs shall not modify information contained in the Master-ICE or a CA-Descriptor.

The Master ICE is a logical file. The syntax and format of the Master-ICE and the CA-Descriptor is described in clause 9. The location of the Master ICE on a given platform is described in Annex B.

4.4 Submission Principle

Since CAs and LAs may share differently the functions required to conform to a telecommunications service, some of these functions are possibly duplicated or missing. Examples are:

- checking the conformance of a Transfer Format to specific service requirements;
- printing a document in accordance to the telecommunications service through which it was (or will be) conveyed;
- converting Transmission Formats to or from Transfer Formats.

The submission principle allows LAs to take advantage of functions that the CA supports itself. This relieves the LA from supporting functions that are already handled by the CA. This ensures that all the required telecommunications service functions for any complying equipment are actually supported.

5 Functional Behaviour

5.1 Functional Classes and Service Profiles

Two Functional Classes (FCs) are defined. These classes are called Functional Class A (FCA) and Functional Class B (FCB). Functional Class B is a superset of Functional Class A. For more detailed information refer to clause 10.

The Functional Class of a CA shall be explicitly stated in the manufacturer documentation and in the appropriate field of the CA-Descriptor.

Furthermore, the interface provides access to additional CA features. These features are not essential with respect to the ITU-T Recommendations. Any of these additional features may be used by a LA independent from its Functional Class, provided that the CA supports them. The additional features supported by a given CA shall be explicitly listed in the ICE (see clause 9).

To further help reduce incompatibilities between different vendors' applications, this Recommendation supports the concept of Service Profiles. A Service Profile, defined for some services, groups a defined set of service features that must be supported by the implementations claiming compatibility to this Recommendation (refer to clause 10).

5.2 Error Handling

5.2.1 Simple Errors

Syntax elements out of context (e.g. specification of transmission speed for the telex service) shall be ignored by the CA. Execution shall continue.

5.2.2 Rejection

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 a TDD) may lead to the rejection of the TDD. Any invalid outgoing file format may also provoke the rejection of the corresponding TDD by the CA. The CA may also discard the outgoing file. Furthermore, the CA may reject the TDD if the companion files are not transferred to the CA within a certain delay.

5.2.3 Error Code Ranges

See Table 2.

TABLE 2/T.611

Assignment of error codes

Error code	Meaning
0000	Success (no error) shall be supported by all CAs
0001-4999	Reserved for CA private use
5000-5999	Miscellaneous errors not fitting in other categories
6000-6999	Syntax errors
7000-7999	Resource and input/output errors (operating system and hardware related errors)
8000-8999	Conversion or Transfer Format related errors
9000-9999	Service dependent errors (service signals and failures)

A complete list of the error codes assigned within this Recommendation is given in Annex C.

5.3 Multiple LAs and multiple CAs

Thanks to the interface, multiple LAs may be connected to one or several CAs simultaneously. In order to control access to multiple CAs, the ICE has been defined. On behalf of the ICE, the LA shall carry out a two-step interface set-up procedure during its initialization phase:

– first, the local application shall select an appropriate communications application by inspecting the Interface Configuration Environment (ICE);

– then, the local application shall "login" to the selected CA.

Once the local application has logged into the communications application, it is free to use whatever service the CA provides, until the local application logs out from the CA.

5.3.1 Step 1: Selection of a CA

The first step to be carried out by the local application during the interface set-up procedure is to gain access to the ICE, which provides a list of CAs accessible from within the system (see 4.3 and clause 9).

5.3.2 Step 2: Login of a LA to a CA

Once the local application has selected an appropriate CA from the ICE, it has to register with the selected CA. This is called the login process. The login process is performed using an unique login-name and returns a Connection-ID. This Connection-ID is computed by the CA.

To perform the login, the LA shall rely on the information found in the CA-Descriptor.

The logout process allows the decoupling of the LA and the CA: it is invoked by the LA when the LA wishes to be disconnected from the CA. As a result of this logout process, the Connection-ID shall be discarded by the CA and becomes invalid. No further access on behalf of that Connection-ID shall be possible for the LA.

The login and logout processes facilitate the implementation of security schemes which are especially important on multi-user systems. They also provide means to implement security mechanisms between the LA and the CA.

5.4 Identification Means

To cope with the various interchanges of information through the interface, it is required to identify unambiguously the communicating entities and the communication events. These identifiers provide the means to:

– distinguish the various telecommunications events (COM-ID);

– identify the LAs logged in a given CA (LA-ID);

– identify the requests generated by one LA toward a given CA (REQ-ID);

– define the Transfer Format to be used (Convert-ID);

– define the Transmission Format to be used (Type-ID).

Following subclauses detail these identifiers.

5.4.1 Identification of CA Communications (COM-ID)

As long as a CA can handle different requests from different LAs and/or from the network, it is necessary to give an identification to each of those events.

The COM-ID identifier is a unique identifier provided by the CA and assigned to each communications event that occurred in a CA. The COM-ID is a field of the CA-Record (see 5.6.2).

A new COM-ID is generated by the CA when a CA-Record is generated. This is the case when:

1) a LA issues a Send request to the CA;

2) a LA issues a Trace request, function Reschedule to the CA;

3) the CA processes a received file.

The COM-ID ensures that a particular LA can retrieve any scheduled transmission request, even if the LA-CA dialogue was terminated for any reason.

5.4.2 Identification of LAs within a CA (LA-ID)

Since this Recommendation allows multiple LAs or multiple instances of a LA using a given CA simultaneously, LAs have to be identified uniquely to a given CA.

For the purpose of distinguishing different LAs from one another, this Recommendation defines the LA-ID identifier as the unique identifier designating a particular LA instance communicating with a CA. The LA-ID is a field of the CA-Record (see 5.6.2).

The CA may refuse to process any Request TDDs that contain a LA-ID not known by that CA. That provides a means to control the accesses to a CA (see 5.6.1).

The LA-ID is statically assigned to the LA or to the LA instance. The rule by which the LA-ID is assigned is beyond the scope of this Recommendation.

5.4.3 Identification of LA requests (REQ-ID)

This Recommendation allows a LA to generate multiple requests to a CA. Since the interface allows Response TDDs to be delivered to the LA in an order different from the one the requests were given, it is necessary to identify the Request TDDs and the corresponding Response TDDs. This Recommendation thus defines the REQ-ID as the unique request reference assigned to each Request TDD and to its corresponding Response TDD.

The REQ-ID is computed by the LA, by any appropriate means that guarantees uniqueness of REQ-IDs for that LA. The algorithm to use for the REQ-ID computation is beyond the scope of this Recommendation. The REQ-ID is a field of the CA-Record (see 5.6.2).

Recovery procedures that could be derived from the use of the REQ-ID are beyond the scope of this Recommendation.

5.4.4 Reference to LA requests (REQ-REF)

The reference to a LA request (REQ-ID) is used in the Trace TDD for referencing previous Send and/or Receive TDDs.

5.4.5 Identification of Transfer Formats (Convert-ID)

The Transfer Format – the format used to transfer a document between LA and CA – is identified by the Convert-ID. For each service this Recommendation defines a mandatory set of different Convert-IDs, which shall be supported by the CA implementation. See also Table H.1 and related service specific sections in Part II of this Recommendation.

5.4.6 Identification of Transmission Formats (Type-ID)

The Transmission Format – the format used to transmit a document through the service – is identified by the Type-ID. Each service defines its set of Transmission Formats. See also related service specific sections in Part II of this Recommendation.

5.5 Dispatch Received Files Facility (DRF)

Some telecommunications services do not offer subaddressing facilities. On systems where the CA may be used by several LAs simultaneously, it is necessary to assist the routing of the incoming documents to the intended recipients. This process is called Dispatch Received Files (DRF).

When a CA receives a file from the network, it shall be assigned to a single recipient LA. Then, only the recipient LA may access the received file. Selection of the recipient LA is a CA private process which is beyond the scope of this Recommendation.

The CA-Record corresponding to the received file is then assigned the LA-ID of the recipient LA.

If the CA supports the Dispatch Received Files facility (DRF) then the recipient LA may dispatch received files to the appropriate LA by means of the Trace:DISPATCH request (see 6.6.3). The CA-Record is then assigned the LA-ID of the LA to which the received file is dispatched.

As this CA feature is optional, the support of DRF shall be declared in the CA-Descriptor (see clause 9).

If the CA does not support DRF, then the CA shall assign (by any appropriate means) the CA-Record to any LA using the LA-ID field. When subaddressing is provided by the telecommunications service, the CA shall assign the CA-Record to the intended recipient LA automatically.

The algorithms to use for dispatching the incoming calls to the appropriate LAs are beyond the scope of this Recommendation.

5.6 Communications Control – CA-Record

Since the exchange method between a LA and a CA is based on a client-server model, the LA always originates Request TDDs to a CA. Response TDDs from the CA are not spontaneous; the LA shall poll the CA in order to know whether any Response TDDs are available.

This Recommendation provides the means for a LA to trace any communications events occurring in a CA; there is no requirement for LAs to make use of those means.

A CA communications event gathers a collection of information, like date and time, originator, recipient(s), communications service, etc.

The CA-Record is the functional collection of information kept by a CA in order to process a Send request or an incoming call from the network. This information is kept into fields, each of which has a special purpose.

A CA-Record is generated by the CA when one of the following events occurs:

– The CA receives a Send request from a LA. In this case, if the Send request contains many recipients, the CA shall expand the list of recipients and generate the same number of CA-Records as the number of recipients in the list.

– The CA receives a Trace:RESCHEDULE request from a LA.

– The CA receives an incoming call from the network.

Optionally, a CA can generate a CA-Record in the “failed” state when the CA encounters error conditions which were not directly the consequence of a Send request or of an incoming call.

CA-Records are destroyed when the LA issues Trace:PURGE requests or by CA-specific means.

NOTE – A CA-Record can only be destroyed when it has reached a final state, i.e. the "retrieved", "sent" or "failed" state (see also Figures 5 and 6).

The CA-Record is an internal structure of the CA. The way to implement the CA-Record is beyond the scope of this Recommendation.

5.6.1 Access Control of CA-Record

All actions originated by LAs on CA-Records are performed by means of the Trace request. Depending on the configuration of the CA, a CA may elect to hide some information to a LA. For example, a CA may refuse the access to "delayed" CA-Records originated by another LA.

To help control accesses to a CA, this Recommendation provides a mechanism to identify LAs by means of the LA-ID. For example, all Trace requests originated by LAs incorporate the LA-ID.

Thanks to this mechanism, a particular CA may elect to restrict or extend the use of some control requests to only a particular set of LAs. For example, the use of the Trace:DISPATCH request may be offered to only one LA or to a set of different LAs, depending on the system configuration; the access to the Trace:PURGE request might also be reserved to a single LA for administrative reasons.

The CA manufacturer shall state in its documentation how access controls (if any) are handled and, if appropriate, how access controls may be customized to accommodate users' configurations. Customizing CAs shall be exercized by specific means that are beyond the scope of this Recommendation.

5.6.2 Fields of the CA-Record

The CA-Record contains a minimum list of fields which shall be supported by all CAs. CAs may support additional fields; those fields shall be declared in the ICE.

Table 3 shows the minimum list of CA-Record fields that any CA shall support in the shown order:

TABLE 3/T.611

Minimum list of CA-Record fields

Field name	Syntax	Purpose
COM-ID	COM-ID	Maintain the unique communica-tion identifier assigned to the CA-Records
LA-ID	La-id	Assigns the CA-Record to the LA which originated it or to which it is destined
REQ-ID	Req-id	Maintains the reference of the request
STATE	State	Indicates the current state of the CA-Record
DIRECTION	"Xmit"/"Receive"	Indicates whether the CA-Record was generated for a transmission or a reception
NOTE – Support of additional fields (e.g. charging info, time stamps, addresses) may be added by CA Manufacturers. See also 6.6.3.2.		

The CA-Record provides a LA the ability to control the CA it is logged in. A CA-Record is at any time in one of the six following states (see Table 4).

TABLE 4/T.611

States of CA-Record

State name	Concerns	Means
“delayed”	Transmission	The associated document(s) have not yet been processed by the CA. It waits until the CA moves it in the sending state
“sending”	Transmission	The associated document(s) have been processed by the CA for transmission; the CA has not yet finished processing it (because it is currently being sent or because the transmission had failed and the CA will carry out other attempts shortly)
“sent”	Transmission	The associated document(s) have successfully been transmitted to the recipient
“failed”	Transmission, reception	The associated document(s) failed to be transmitted, (or) errors occurred during reception (or) an internal error occurred in the CA
“reception”	Reception	The record describes a document that is being received by the CA at that time, or has been received but not retrieved by the recipient LA
“retrieved”	Reception	The received document has been retrieved by the recipient LA by means of the Receive request

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

LAs can read the state fields of a CA-Record by means of the Trace:COPY request. A particular LA may access only those records that have a matching LA-ID identifier value (unless the CA extends the access rights of some LAs). This ensures a particular LA may consult only its relevant CA-Records.

The only exception to this rule is when the CA does not support DRF (see 5.5), in which case any LA can access any CA-Record which is in the reception state.

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

5.6.3 Transmission Process – State Transitions

Transmissions are initiated by Send requests. When the CA receives a valid Send request from a logged-in LA, the CA builds as many CA-Records as the number of recipients contained in the Send request.

Transmissions can also be originated by rescheduling transmissions that failed previously, by means of the Trace:RESCHEDULE request. In some cases this can save the delay of the conversion of the documents to suitable Transmission Formats, because those documents were already converted in a previous Send request.

The CA shall assign a COM-ID to each new CA-Record. If a Send request specifies multiple recipients, the CA assigns a different COM-ID to each resulting CA-Record (one per recipient). The CA also sets the state field to the “delayed” state. The CA then copies all information from the Send request into the CA-Record.

The CA scans at its own pace the list of the CA-Records in the “delayed” state and processes one of them. The choice of the “delayed” CA-Record to process first is based on scheduled dates and times that were specified in the <SendTime> field of the Send request (see Table 13, subclause 6.2). The elected CA-Record is switched to the “sending” state.

The CA transmits all documents contained in the “sending” CA-Record, one by one. If transmission fails because of transmission errors, then the CA may keep the CA-Record in the “sending” state as long as it retries its transmission attempts.

If the transmission eventually fails, the CA-Record is switched to the “failed” state. If the transmission eventually succeeds, the CA-Record is switched to the “sent” state. Between two attempts on the same CA-Record, the CA may pick up another CA-Record in the “delayed” state and process it as described above. Consequently, more than one CA-Record may be in the “sending” state at a given moment.

Figure 5 illustrates these state transitions.

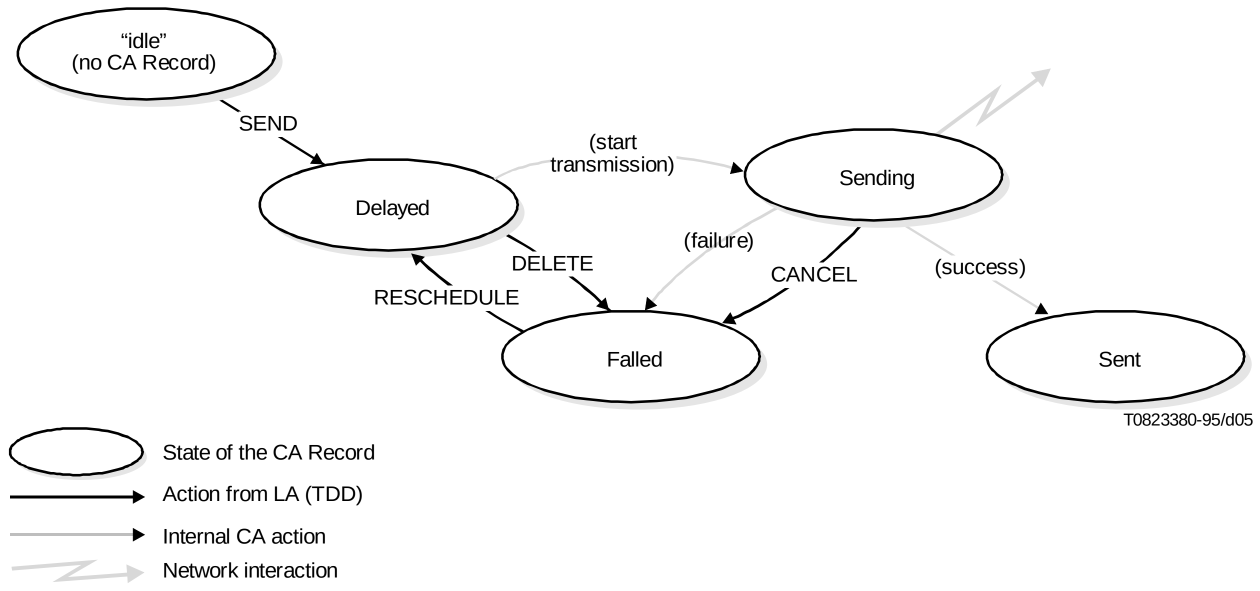


FIGURE 5/T.611

State transitions of the CA-Record (transmissions)

5.6.4 Reception process – State Transitions

When a CA has received an incoming communication from the network, it builds a CA-Record. This CA-Record is assigned a unique COM-ID identifier and set to the “reception” state. All information concerning the incoming call (like originator, date and time, etc.) is then copied to the CA-Record in the appropriate fields.

If failure has occured during reception, the CA-Record is switched to the “failed” state. Otherwise the CA-Record is switched to the “retrieved” state, if a LA has retrieved it by means of the Receive request.

The Trace:DISPATCH request has no effect on the state of the CA-Record, i.e. it remains in the “reception” state; the CA-Record is no longer visible to the “dispatching LA” and becomes visible to the recipient LA.

Figure 6 illustrates these state transitions.

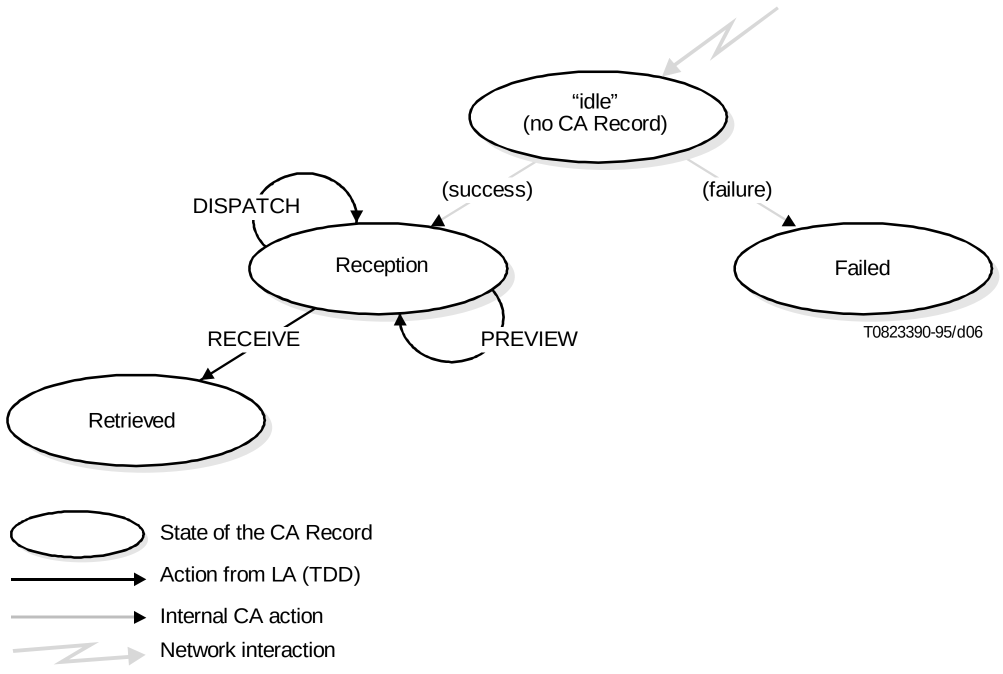


FIGURE 6/T.611

State transitions of the CA-Record (reception)

5.6.5 Actions – Notation Conventions

For the sake of legibility, the following conventions apply for the notation of the various actions involving CA-Records (see Table 5).

TABLE 5/T.611

Notation of actions involving the CA-Record

Notation	Means
Trace:DELETE rq	DELETE request of Trace TDD
Recipient LA	The intended recipient of the incoming call. The recipient can be determined either automatically (example: subaddressing mechanism) or by manual dispatch. Automatic processing is a CA private matter whereas the manual dispatching can be controlled through the interface (see 10.6.1)
Originating LA	The LA that originated the CA-Record (via a Send request)
Transmit	The CA attempts to carry out the transmit action on the CA-Record
NOTE – The CA may allow LAs of its choice to behave like a recipient or originating LA. This can be useful for many applications (example: Administration).	

5.6.6 Actions – Transmissions

This subclause describes all actions that impact the various states of the CA-Records that are involved in the transmissions.

5.6.6.1 Delayed State

A CA-Record is in the “delayed” state as long as it has not been processed by the CA (i.e. no attempt was made to transmit it) or as long as the originating LA did not delete it. The following actions may apply to a CA-Record in the “delayed” state (see Table 6).

TABLE 6/T.611

Transmission actions for the “delayed” state

Action	Originator	Purpose	Resulting State
Trace:DELETE rq	Originating LA	To delete a Send request. This action operates on only one CA-Record at a time	“failed”
Trace:COPY rq	Originating LA	To build a logical file containing a copy of the list of the “delayed” CA-Records. This action applies to all “delayed” CA-Records originating from a single LA	“delayed”
Transmit	CA	The CA decides to handle the CA-Record, because it has become due to process	“sending”

5.6.6.2 Sending State

A CA-Record is in the “sending” state when the CA attempts to transmit the information contained in it.

If the attempt fails, the CA-Record may remain in the “sending” state waiting for the next attempt; the CA shall delay the next attempt with respects to the “retry delay” unless the “maximum number of attempts” is reached, in which case the CA-Record is set to the “failed” status.

The following actions may apply to a CA-Record in the “sending” state (see Table 7).

TABLE 7/T.611

Transmission actions for the “sending” state

Action	Originator	Purpose	Resulting State
Trace:CANCEL rq	Originating LA	To cancel a Send request. This affects only one CA-Record at a time	“failed”
Trace:COPY rq	Originating LA	To build a logical file containing a copy of the list of the “sending” CA-Records	“sending”

5.6.6.3 Sent State

A CA-Record is in the “sent” state when the CA succeeded in transmitting the information contained in it.

The following actions may apply to a CA-Record in the “sent” state (see Table 8).

TABLE 8/T.611

Transmission actions for the "sent" state

Action	Originator	Purpose	Resulting state
Trace:PURGE rq	Originating LA	To remove all the CA-Records being in the "sent" state	Not applicable because removed (purged)
Trace:COPY rq	Originating LA	To build a logical file containing a copy of the list of the "sent" CA-Records	"sent"

5.6.6.4 Failed State

A CA-Record is in the "failed" state when the CA failed to transmit the information contained in it for any reason.

NOTE – The failed state also applies for the reception process. See below for relevant information.

The following actions may apply to a CA-Record in the "failed" state (see Table 9).

TABLE 9/T.611

Transmission actions for the "failed" state

Action	Originator	Purpose	Resulting State
Trace:PURGE rq	Originating LA	To remove CA-Records being in the "failed" state	Not applicable because removed (purged)
Trace:RESCHEDULE rq	Originating LA	To ask for reprocessing of a failed request. This can take advantage of previous document conversions	"delayed"
Trace:COPY rq	Originating LA	To build a logical file containing a copy of the list of the "failed" CA-Records	"failed"

5.6.7 Actions – Receptions

This subclause describes all actions that impact the states of the CA-Records that relate to reception.

5.6.7.1 Reception State

A CA-Record is in the "reception" state when the CA successfully received an incoming call from the network, and the CA-Record has not been dispatched already.

The following actions may apply to a CA-Record in the "reception" state (see Table 10).

TABLE 10/T.611

Reception actions for the "reception" state

Action	Originator	Purpose	Resulting State
RECEIVE rq	Recipient LA	To retrieve the document(s) relevant to the CA-Record (as well as related information in the Response-TDD)	"retrieved"
Trace:PREVIEW rq	any LA	To receive a copy of the document(s) relevant to the CA-Record. The CA-Record remains in the "reception" state	"reception"
Trace:DISPATCH rq	Recipient LA or Admin	To assign one or more LA-ID(s) to a received CA-Record (see also Note below)	"reception"
Trace:COPY rq	Recipient LA	To build a logical file containing a copy of the list of the CA-Records in "reception" state, thus not having been retrieved already. A recipient LA "sees" only its relevant CA-Records, i.e. those matching the LA-ID	"reception"
NOTE – The Management of the Admin (Administrator) is a private matter of the CA.			

5.6.7.2 Retrieved State

A CA-Record is in the "retrieved" state when the CA-Record has been retrieved by a LA with a matching LA-ID by means of the Receive request.

The following actions may apply to a CA-Record in the "retrieved" state (see Table 11).

TABLE 11/T.611

Reception actions for the "retrieved" state

Action	Originator	Purpose	Resulting State
Trace:PURGE rq	Recipient LA	To remove CA-Records which are in the "retrieved" state	Not applicable because removed (purged)
Trace:COPY rq	Recipient LA	To build a logical file containing a copy of the list of the CA-Records in "retrieved" state. A recipient LA "sees" only its relevant CA-Records, i.e. those matching the LA-ID	"retrieved"

5.6.7.3 Failed State

A CA-Record is in the "failed" state when the CA failed to receive an incoming call from the network for any reason.

NOTE – The failed state also applies for the transmission process. See 5.6.6.4 for relevant information.

When a CA fails to receive an incoming call, the CA switches the CA-Record from the "reception" state to the "failed" state.

When a CA-Record is in the “failed” state, no LA-ID is assigned to it. Depending on the configuration, the CA may (but is not obliged to) assign a LA-ID on those “failed” records by any appropriate means. The Trace:RESCHEDULE request does not apply on received “failed” CA-Records.

Table 12 details the actions that may apply on the CA-Records in the “failed” state.

TABLE 12/T.611

Reception actions for the “failed” state

Action	Originator	Purpose	Resulting State
Trace:PURGE rq	Recipient LA	To remove all the CA-Records being in the “failed” state	Not applicable because removed (purged)
Trace:COPY rq	Recipient LA	To build a logical file containing a copy of the list of the CA-Records in “failed” state. A recipient LA “sees” all CA-Records in this state	“failed”

6 Task Data Descriptions (TDDs)

The functionality offered by the CA through the interface is invoked by TDDs, which are passed as requests from the LA to the CA and – depending on the function invoked – passed vice versa as response.

This clause describes the semantic, functionality, syntax and encoding of the various Request TDDs and responses.

6.1 Generic TDD Presentation

The TDDs can be encoded with different schemes. Some of these coding schemes are based on text descriptions, i.e. ASCII or EBCDIC. Other coding schemes cannot be represented by text. Examples of such coding schemes include the binary encoding scheme, which is presented in C-language notation in Part III of this Recommendation.

The various TDDs are hereafter described generically with a BNF-style syntax. The details of this syntax are explained in A.1.

As an aid to understanding, together with the generic BNF-style syntax given below the text based encoding is used in the subclauses to follow. However, the binary encoding scheme (defined in Part III of this Recommendation) may alternatively be used for implementations.

```
<TDD>  :=                <TDD Header>
                         <Send> | <Receive> | <Trace> | <Submit> | <Extend>

<TDD Header>  :=         -- dependent of encoding used. For text based encoding see 6.4.3,
                         -- for binary encoding scheme see appropriate clause in Part III
                         -- of this Recommendation

<Send>  :=               <SendTDD> | <SendAckTDD>

<Receive>  :=            <ReceiveTDD>

<Trace>  :=              <DeleteTDD> | <CopyTDD> | <CancelTDD> | <PurgeTDD> |
                         <RescheduleTDD> | <DispatchTDD> | <PreviewTDD>
```

```
<Submit>  :=                     <PrintTDD> | <ConvertTDD> | <CheckTDD>


<Extend>  :=                     <ExtendTDD> | <NationalTDD> | <PrivateTDD>

<SendTDD>  :=                    <SendFunction> <LaId> <ReqId> <Service>
                                 [<SendTime>] [<Comment>] [<LastTime>] [<UserKey>]
                                 <ServiceDependentKeywordsSend>

<ServiceDependentKeywordsSend>  :=
                                 --  defined in appropriate clause of Part II for each service

<SendAckTDD>  :=                 <SendAckFunction> <LaId> <ReqId> <Service> <Error>
                                 <Status> [<ComId>] [<SendTime>] [<Comment>] [<LastTime>]
                                 [<UserKey>] [<Minor>] [<Warning>]
                                 <ServiceDependentKeywordsSendack>

<ServiceDependentKeywordsSendack>  :=
                                 --  defined in appropriate clause of Part II for each service

<ReceiveTDD>  :=                 <ReceiveFunction> <LaId> <ReqId> <Error> <Status> [<ComId>]
                                 [<ReceiveTime>] [<Service>] [<Delete>] [<Minor>] [<Warning>]
                                 <ServiceDependentKeywordsReceive>

<ServiceDependentKeywordsReceive>  :=
                                 --  defined in appropriate clause of Part II for each service

<DeleteTDD>  :=                  <DeleteFunction> <LaId> <ReqId> <Error> (<ComId> | <ReqRef>)
                                 [<Minor>] [<Warning>]

<CopyTDD>  :=                    <CopyFunction> <LaId> <ReqId> <Error> <State> <Target> <Layout>
                                 (<ComId> | <ReqRef>) [<Minor>] [<Warning>] [<Previewed>]
                                 [<Dispatched>] [<Direction>] [<Time_Range>] [<Service>]
                                 <ServiceDependentKeywordsCopy>

<ServiceDependentKeywordsCopy>  :=
                                 --  defined in appropriate clause of Part II for each service

<CancelTDD>  :=                  <CancelFunction> <LaId> <ReqId> <Error> (<ComId> | <ReqRef>)
                                 [<Minor>] [<Warning>]

<PurgeTDD>  :=                   <PurgeFunction> <LaId> <ReqId> <Error> <State>
                                 (<ComId> | <ReqRef>) [<Minor>] [<Warning>]

<RescheduleTDD>  :=              <RescheduleFunction> <LaId> <ReqId> <Error> (<ComId> | <ReqRef>)
                                 [<Address>] [<SendTime>] [<LastTime>] [<Minor>] [<Warning>]

<DispatchTDD>  :=                <DispatchFunction> <LaId> <NewLa> {<NewLa>} <ReqId> <Error>
                                 <ComId> [<Minor>] [<Warning>]

<PreviewTDD>  :=                 <PreviewFunction> <LaId> <ReqId> <Target> <Convert> <Error>
                                 <ComId> [<Minor>] [<Warning>]

<PrintTDD>  :=                   <PrintFunction> <LaId> <ReqId> <Error> <FileName> <InFormat>
                                 [<Printer>] [<Minor>] [<Warning>]

<ConvertTDD>  :=                 <ConvertFunction> <LaId> <ReqId> <Error> <FileName> <Target>
                                 <InFormat> <OutFormat> [<Minor>] [<Warning>]

<CheckTDD>  :=                   <CheckFunction> <LaId> <ReqId> <Error> <FileName> <Check>
                                 [<Minor>] [<Warning>]
```

```
<ExtendTDD>  :=                <ExtendFunction> <SubFunction> <LaId> <ReqId> <Error> [<Minor>]
                                [Warning] [<ExtendSubFunctionKeywords>]

<ExtendSubFunctionKeywords>  :=
                                --  defined in appropriate clause of Part II for each service

<NationalTDD>  :=              <NationalFunction> <SubFunction> <LaId> <ReqId> <Error> [<Minor>]
                                [Warning] [<NationalSubFunctionKeywords>]

<NationalSubFunctionKeywords>  :=
                                --  for further study

<PrivateTDD>  :=               <PrivateFunction> <SubFunction> <LaId> <ReqId> <Error> [<Minor>]
                                [Warning] [<PrivateSubFunctionKeywords>]

<PrivateSubFunctionKeywords>  :=
                                --  for further study
```

6.2 Description of TDD Elements

In general, there are two types of syntax elements within a TDD:

1) those which parameters are only dependent on the functionality of the TDD;

2) those which parameters are dependent on the service the TDD is applied to.

In case the TDD is dealing with sending or receiving files or documents, the parameters of some syntax elements may also be specific to

– the addressee (e.g. the attribute primary or copy recipient for E-Mail services);

– the document itself (e.g. Transfer and Transmission Format).

In Table 13 the *non* service dependent syntax elements of the TDDs are explained. The service, adressee or document specific syntax elements of the TDDs are described in the appropriate clauses of Part II of this Recommendation.

6.3 Code-ID

The very first octet of every TDD, independent of its encoding, indicates the coding scheme. This octet is called the Code-ID. Table 14 lists the possible Code-IDs assigned by this Recommendation.

6.4 Text Based Encoding

This subclause refers to encodings with the Code-ID set to "A", "B", "I" or "E".

For description of the text based encoding, the BNF-style syntax explained in A.1 is used in the following subclauses.

6.4.1 Syntax and Formatting

The syntax elements listed in 6.1 are encoded using keyword/parameter pairs. The mapping of the keyword/parameter pairs to the syntax elements is shown in Table 15 below. Since some syntax elements and the corresponding functionality depends on the underlying telecommunications service, only those beeing invariant to the telecommunications service are presented. The syntax elements depending on a telecommunications service are described in the appropriate clause of Part II of this Recommendation.

TABLE 13/T.611

Summary of non service dependent TDD syntax elements

Syntax Element	Purpose
<CancelFunction>	Identifies the function requested by the TDD
<Check>	Gives the format a given File shall be checked against. Used only in the <CheckTDD>
<CheckFunction>	Identifies the function requested by the TDD
<ComId>	ComId stands for COMmunication-ID. See 5.4.1
<Comment>	Used to attach a free comment to a document, which shall be transmitted. This comment then is stored in the CA-Record, rather than transmitted with the document through the telecommunications service
<ConvertFunction>	Identifies the function requested by the TDD
<CopyFunction>	Identifies the function requested by the TDD
<Delete>	Tells the CA whether a document received shall be deleted from the internal storage of the CA after successful retrieval carried out by the calling LA
<DeleteFunction>	Identifies the function requested by the TDD
<Direction>	Specifies whether the CA-Record belongs to an incoming or an outgoing document
<Dispatched>	Specifies if the <DispatchTDD> has been applied to the specific CA-Record
<DispatchFunction>	Identifies the function requested by the TDD
<Error>	This syntax element returns the error code generated by the CA in the Response-TDD. It shall be checked by the LA for successful operation
<ExtendFunction>	Identifies the function requested by the TDD
<ExtendSubFunction-Keywords>	See appropriate clause of Part II of this Recommendation
<FileName>	Contains the path to the file the document is stored in
<InFormat>	Specifies the input format of a document. Only used in the <Convert-TDD>
<LaId>	LaId stands for LA-ID. See 5.4.2
<LastTime>	Gives the latest time a CA shall try to send the document specified in the related TDD
<Layout>	Defines the layout of the target file of the <CopyTDD>
<Minor>	This syntax element returns an additional error code in the Response-TDD
<NationalFunction>	Identifies the function requested by the TDD
<NationalSubFunction-Keywords>	For further study
<NewLa>	Contains the new LA-ID for use with the <DispatchTDD>
<OutFormat>	Specifies the output format of a document. Only used in the <Convert-TDD>
<Previewed>	Specifies if the <PreviewTDD> has been applied to the specific CA-Record

TABLE 13/T.611 (*end*)

Summary of non service dependent TDD syntax elements

Syntax Element	Purpose
<Printer>	ID of selected printer for use with the <PrintTDD>. Depends on the supporting operating system. The CA manufacturer shall state in his documentation how to address the printers
<PrintFunction>	Identifies the function requested by the TDD
<PrivateFunction>	Identifies the function requested by the TDD
<PrivateSubFunction-Keywords>	For further study
<PurgeFunction>	Identifies the function requested by the TDD
<ReceiveFunction>	Identifies the function requested by the TDD
<ReqId>	ReqId stands for REQ-ID. See 5.4.3
<ReceiveTime>	Gives the time the CA received the document specified in the related TDD
<ReqRef>	ReqId stands for REQ-REF. See 5.4.4
<RescheduleFunction>	Identifies the function requested by the TDD
<SendackFunction>	Identifies the function requested by the TDD
<SendFunction>	Identifies the function requested by the TDD
<SendTime>	Gives the time the CA shall send the document specified in the related TDD
<Service>	Specifies the telecommunications service used. See also Part II of this Recommendation
<ServiceDependent-KeywordsReceive>	See appropriate clause of Part II of this Recommendation
<ServiceDependent-KeywordsSend>	See appropriate clause of Part II of this Recommendation
<ServiceDependent-KeywordsSendack>	See appropriate clause of Part II of this Recommendation
<State>	Specifies the state of a CA-Record. See also Table 4 in 5.6.2
<Status>	Returns the status of a transmission or reception in the Response-TDD
<SubFunction>	Identifies the function requested by the TDD
<Target>	Used in the <ConvertTDD> and in the <CopyTDD> to specify the path to the output file generated by the CA
<TimeRange>	Specifies a time range for selecting specific CA-Records
<UserKey>	Contains a key attached to the Request-TDD of the <SendTDD> and the <SendAckTDD>
<Warning>	This syntax element returns an additional warning code in the Response-TDD

TABLE 14/T.611

TDD Encoding – Code-IDs

Code-ID		TDD Representation
Value	Text Presentation	
41_{hex}	"A"	Readable text, organized as lines of keyword/parameter pairs. The character set utilized shall correspond to the APPLI/COM Extended ASCII character set (see Table 50)
42_{hex}	"B"	Readable text, organized as lines of keyword/parameter pairs. The character set used shall correspond to a national variant of the character set as defined per ITU-T Recommendation T.50
43_{hex}	"C"	Binary encoding scheme (represented using C-language notation in Part III of this Recommendation)
$C5_{hex}$ a)	"E"	Readable text, organized as lines of keyword/parameter pairs. The character set utilized shall correspond to an EBCDIC coded character set
49_{hex}	"I"	Readable text, organized as lines of keyword/parameter pairs. The character set utilized shall correspond to the APPLI/COM Standard ASCII character set (see Table 53)
50_{hex}	"P"	The TDD presentation and syntax shall be regarded as being defined by private rules
Other values		All other values not listed above are reserved for future standardization

a) If this Code-ID has to appear in the ICE, its binary value shall be made in accordance to the code presentation chosen for the ICE itself, i.e. if the Code-ID of the ICE Header is "I" (APPLI/COM Standard ASCII) then the binary value of the "E" shall be coded as 45_{hex}.

TABLE 15/T.611

Mapping of keyword/parameter pairs for Text Based Encoding

(A ↵ denotes the new line format effector)

Syntax Element	Keyword/Parameter Pair
<CancelFunction>	"FUNCTION" ":" "Cancel"↵
<Check>	"CHECK" ":" <Convert-id-parameter> ↵
<CheckFunction>	"FUNCTION" ":" "Check"↵
<ComId>	"COMID" ":" <Com-id-parameter> ↵
<Comment>	"COMMENT" ":" <Comment-parameter> ↵
<ConvertFunction>	"FUNCTION" ":" "Convert"↵
<CopyFunction>	"FUNCTION" ":" "Copy"↵
<Delete>	"DELETE" ":" <Boolean-parameter> ↵
<DeleteFunction>	"FUNCTION" ":" "Delete"↵
<Direction>	"DIRECTION" ":" <Direction-parameter> ↵
<Dispatched>	"DISPATCHED" ":" <Boolean-parameter> ↵
<DispatchFunction>	"FUNCTION" ":" "Dispatch"↵
<Error>	"ERROR" ":" <Error-parameter> ↵

TABLE 15/T.611 (*end*)

Mapping of keyword/parameter pairs for Text Based Encoding

(A ⏎ denotes the new line format effector)

Syntax Element	Keyword/Parameter Pair
<ExtendFunction>	"FUNCTION" ":" "Extend"⏎
<FileName>	"FILENAME" ":" <Path-parameter> ⏎
<InFormat>	"INFORMAT" ":" <Convert-id-parameter> ⏎
<LaId>	"LA-ID" ":" <La-id-parameter> ⏎
<LastTime>	"LASTTIME" ":" <Date-time-parameter> ⏎
<Layout>	"LAYOUT" ":" <Layout-id-parameter> ⏎
<Minor>	"MINOR" ":" <Error-parameter> ⏎
<NationalFunction>	"FUNCTION" ":" "National"⏎
<NewLa>	"NEWLA" ":" <La-id-parameter> ⏎
<OutFormat>	"OUTFORMAT" ":" <Convert-id-parameter> ⏎
<Previewed>	"PREVIEWED" ":" <Boolean-parameter> ⏎
<Printer>	"PRINTER" ":" <Printer-id-parameter> ⏎
<PrintFunction>	"FUNCTION" ":" "Print"⏎
<PrivateFunction>	"FUNCTION" ":" "Private"⏎
<PurgeFunction>	"FUNCTION" ":" "Purge"⏎
<ReceiveFunction>	"FUNCTION" ":" "Receive"⏎
<ReceiveTime>	"RCVTIME" ":" <Date-time-parameter> ⏎
<ReqId>	"REQ-ID" ":" <Req-id-parameter> ⏎
<ReqRef>	"REQREF" ":" <Req-id-parameter> ⏎
<RescheduleFunction>	"FUNCTION" ":" "Reschedule"⏎
<SendAckFunction>	"FUNCTION" ":" "SendAck"⏎
<SendFunction>	"FUNCTION" ":" "Send"⏎
<SendTime>	"SENDTIME" ":" <Send-time-parameter> ⏎
<Service>	"SERVICE" ":" <Service-parameter> ⏎
<State>	"STATE" ":" <State-parameter> ⏎
<Status>	"STATUS" ":" <Status-parameter> ⏎
<SubFunction>	"SUBFUNC" ":" <Subfunc-parameter> ⏎
<Target>	"TARGET" ":" <Path-parameter> ⏎
<TimeRange>	"RANGE" ":" <Date-time-parameter> "," <Date-time-parameter> ⏎
<UserKey>	"USERKEY" ":" <Userkey-parameter> ⏎
<Warning>	"WARNING" ":" <Error-parameter> ⏎

To improve the legibility of the TDD, all keyword/parameter pairs that appear shall be formatted by the use of format effectors. Following rules apply:

– the syntax elements shall be arranged as lines of keyword/parameter pairs;

– the FUNCTION keyword must be always the first keyword presented; if a SUBFUNC keyword exists for the TDD it must be second; the order of all other keywords is undefined;

- the line shall be terminated by a new line format effector (see Table **Error! Bookmark not defined.**);
- the parameter shall be separated from the keyword by a colon (":" character);
- each line may contain a comment. The comment shall be introduced by a ";" character and extents to the end of the line. The comment is not interpreted by the CA. If a statement has to contain a ";" character, it shall be escaped with a backslash ("\" character);
- a line may contain white space format effectors (see Table **Error! Bookmark not defined.**);
- the keywords are not case sensitive (upper case and lower case are treated equally);
- the parameter presentation is not case sensitive as well, except in those cases where it is required by the service or the underlying operating system.

A Request TDD contains input parameters. The Response TDD may contain output parameters. The CA shall respect following building rules for the Response TDD[3]:

- all input parameters of the Request TDD shall be returned in the Response TDD;
- at minimum all output parameters declared basic shall be put into the Response TDD by the CA;
- the order of the parameters contained in the Request TDD may be changed by the CA, except that the FUNCTION keyword shall be the first keyword presented, followed by the SUBFUNC keyword if applicable;
- the case of the parameters contained in the Request TDD may be changed by the CA;
- comments contained in the Request TDD may be stripped.

TABLE 16/T.611

Format effectors for the Text Based Encoding

Format effector	Code-ID dependent coding			
	"A"	"B"	"E"	"I"
New line	0D0A$_{hex}$ or 0A$_{hex}$	0D0A$_{hex}$ or 0A$_{hex}$	For further study	0D0A$_{hex}$ or 0A$_{hex}$
White space	20$_{hex}$ or 09$_{hex}$	20$_{hex}$ or 09$_{hex}$	For further study	20$_{hex}$ or 09$_{hex}$

Subclause 6.6 tells in detail which TDD parameters are used as input and which are used as output parameters.

6.4.2 Mapping of Keywords

6.4.3 Encoding of the TDD Header

In every TDD (and within the ICE) the <TDD Header> is always the first element specified. The <TDD Header> is structured as shown below (for explanation of the BNF-based grammar see A.1):

<TDD Header> :=	<Code-ID> <Identification> <Version> <Standard> <Reserved>
<Code-ID> :=	"A" | "B" | "E" | "I" -- *for presentation of the Code-ID see also Table* ***Error! Bookmark not defined.***

[3] These rules have changed considerably compared to the 1992 version of this Reccommendation. The rules defined there are far more strict.

<Identification> :=	"*APPLI/COM*"
<Version> :=	"1994" -- *<Version> denotes year of approval of this Recommendation*
<Standard> :=	"*ITU-T*"
<Reserved> :=	STRING (SIZE(0..16)) -- *<Reserved> is reserved for further extensions of this* -- *Recommendation*

NOTE – The <TDD Header> syntax element contains the Code-ID as first element.

6.4.4 Encoding of Non Service Dependent Parameters

This subclause defines the encoding of the non service dependent parameters addressed in Table **Error! Bookmark not defined.**. Service-dependent parameters are mentioned in 6.4.5. They are fully specified in the relevant subclauses of Part II of this Recommendation.

6.4.4.1 Boolean-parameter

The Boolean-parameter is encoded as a STRING taking the presentations "YES" or "NO".

General Syntax:

<Boolean-parameter> :=	"YES" | "NO"

6.4.4.2 Com-id-parameter

The Com-id-parameter is encoded as a STRING containing the Communication-ID (COM-ID) computed by the CA. See also 5.4.1.

General Syntax:

<Com-id-parameter> :=	STRING

6.4.4.3 Comment-parameter

The Comment-parameter is encoded as a STRING containing the Comment the LA specifies for the specific communication event.

General Syntax:

<Comment-parameter> :=	STRING

6.4.4.4 Date-time-parameter

The Date-time-parameter is encoded as a STRING containing the date and time following the conventions "YY-MM-DD-HH:MM" (year-month-day-hour:minutes).

General Syntax:

<Date-time-parameter> :=	<year> "-" <month> "-" <day> "-" <hours> ":" <minutes>
<year> :=	<digit> <digit>
<month> :=	"0" | "1" <digit>
<day> :=	"0" | ... | "3" <digit>
<hours> :=	"0" | ... | "2" <digit>
<minutes> :=	"0" | ... | "5" <digit>
<digit> :=	"0" | ... | "9"

6.4.4.5 Direction-parameter

The Direction-parameter is encoded as a STRING which can take the presentations "XMIT" or "RECEIVE".

General Syntax:

<Direction-parameter> :=	"XMIT" | "RECEIVE"

6.4.4.6 Error-parameter

The Error-parameter is encoded as a structured STRING with the following layout:

"nnnn"/"Text..."

"nnnn" stands for a 4-digit, right-justified error number, filled to the left with "0"s. "0000" = success (no error). The value of the error number itself depends on the communications software used. Error codes are counted decimal. For the assignment of the error-code ranges refer to Table **Error! Bookmark not defined.** of 5.2.3.

"Text..." stands for a plain text with a length of up to 79 characters that describes the error. If the parameter field is not long enough to accept plain text, the text is abbreviated as necessary. The plain text for error number "0000" is "Success".

The error numbers which may be assigned are detailed in 5.2.3 and Annex C respectively.

General Syntax:

<Error-parameter> :=	<digit> <digit> <digit> <digit> / STRING(SIZE(0..79))
<digit> :=	"0" | ... | "9"

6.4.4.7 La-id-parameter

The La-id-parameter contains the reference of a LA-ID. It is presented as a STRING. The purpose of the parameter is to identify the "owning" LA of a request.

General Syntax:

<La-id-parameter> :=	STRING

6.4.4.8 Layout-id-parameter

The Layout-id-parameter is encoded as NUMERIC-STRING, which can take the values "0", "1" or "2".

General Syntax:

<Layout-id-parameter> :=	"0" | "1" | "2"

6.4.4.9 Path-parameter

The Path-parameter is encoded as PATH.

General Syntax:

<Path-parameter> :=	STRING -- *limited to the platform-specific file naming conventions*

6.4.4.10 Printer-id-parameter

The Printer-id-parameter contains the ID of a selected printer, encoded as a STRING. The contents of the STRING depends on the supporting operating system. The CA manufacturer shall state in his documentation how to address printers.

General Syntax:

<Printer-id-parameter> := STRING

6.4.4.11 Req-id-parameter

The Req-id-parameter contains the REQ-ID (see 5.4.3). The parameter value is represented as a STRING. The purpose of the parameter is to identify the relation of a response to a previous request. So the REQ-ID shall be unique within a LA. It is the responsibility of the LA to ensure the REQ-ID is unique.

General Syntax:

<Req-id-parameter> := STRING

6.4.4.12 Send-time-parameter

The Send-time-parameter is encoded as a STRING containing the date and time following the conventions "YY-MM-DD-HH:MM" (year-month-day-hour:minutes) or a specific indication for immediate or urgent treatment.

General Syntax:

<Send-time-parameter> := "IMMEDIATE" | "URGENT" | <Date-time-parameter>
-- *The <Date-time-parameter> is described in 6.4.4.4*

6.4.4.13 State-parameter

The State-parameter specifies the state of the CA-Record. It is encoded as STRING and may take one of the values shown below:

"delayed"	concerns transmission: the record has not yet been processed by the CA.
"sending"	concerns transmission: the record is being processed by the CA for transmission.
"sent"	concerns transmission: the record has been successfully transmitted to the recipient.
"failed"	concerns transmission and reception: the record failed to be fully transmitted or errors occurred during reception or an internal error inside of the CA did occur.
"reception"	concerns reception: the record has been received but not yet retrieved.
"retrieved"	concerns reception: the record has already been retrieved.

General Syntax:

<State-parameter> := "delayed" | "sending" | "sent" | "failed" | "reception" | "retrieved"

6.4.4.14 Status-parameter

The Status-parameter reflects the status of a send or receive event. It may show the following STRING values:

"+"	positive;
"+–"	partially negative;
"–"	negative;
"?"	unknown.

General Syntax:

<Status-parameter> := "+" | "+–" | "–" | "?"

6.4.4.15 Subfunction-parameter

The Subfunction-parameter is encoded as a STRING which specifies the Subfunction to be invoked.

General Syntax:

```
<Subfunction-parameter>  :=        STRING
```

6.4.4.16 Userkey-parameter

The Userkey-parameter is encoded as a STRING containing a User-key the LA specifies for the specific communication event.

General Syntax:

```
<Userkey-parameter>  :=            STRING
```

6.4.5 Encoding of Service Specific Parameters

This subclause describes some of the service specific parameters used in a similar fashion for various services. However, a full description of the parameters mentioned is given in the appropriate clauses of Part II of this Recommendation.

6.4.5.1 Service-id-parameter

This parameter occurs in all send and receive TDDs. It specifies the service to be used. It is encoded as STRING and may take one of the following values:

"FX3"	for facsimile group 3 services
"FX4"	for facsimile group 4 services
"TTX"	for the teletex service
"TX"	for the telex via teletex facilities service (without dialogue ability)
"TLX"	for the telex service
"EMAIL"	for e-mail services
"FT"	for file transfer services

General Syntax:

```
<Service-id-parameter>  :=         "FX3" | "FX4" | "TTX" | "TX" | "TLX" | "EMAIL" | "FT"
```

6.4.5.2 File-of-addrspec and Address-parameter

These parameters appear in all send and receive TDDs and is used in a similar fashion throughout all services.

The File-of-addrspec parameter is encoded as PATH which points to a file containing Addrspec-parameters, which contain the Address-parameter.

The Address-parameter is encoded as STRING. For some telecommunications services this string may be further restricted.

General Syntax:

```
<File-of-addrspec>  :=             PATH
                                   --  Path to a file containing a list of <Addrspec-parameter>s

<Addrspec-parameter>  :=           <Address-parameter> <Other-parameters>
                                   --  The <Addrspec-parameter> itself is fully detailed for each service
                                   --  in the appropriate clause of Part II of this Recommendation.

<Address-parameter>  :=            --  fully detailed in the appropriate clauses of Part II of this
                                   --  Recommendation.
```

6.4.5.3 Convert-id-parameter

This parameter appears in various TDDs. Some of them, i.e. the send and receive TDDs are service dependent.

In all cases the Convert-id-parameter is encoded as a STRING and contains the Convert-ID, which specifies the transfer format. See also 5.4.5, clause 8 and Table H.**Error! Bookmark not defined.** as well as the appropriate clauses of Part II of this Recommendation for more information about the Convert-ID.

General Syntax:

```
<Convert-id-parameter>  :=          STRING
```

6.4.5.4 Type-id-parameter

This parameter appears in all send and receive TDDs and is used in a similar fashion throughout all services.

The Type-id-parameter is encoded as a STRING and contains the Type-ID, which specifies the Transmission Format. See also 5.4.6, and Table H.**Error! Bookmark not defined.** as well as the appropriate clauses of Part II of this Recommendation for more information about the Type-ID.

General Syntax:

```
<Type-id-parameter>  :=             STRING
```

6.4.5.5 File-of-filespec and File-parameter

This parameter appears in all send and receive TDDs and is used in a similar fashion throughout all services. The File-of-filespec parameter is encoded as a PATH which points to a file containing Filespec-parameters, which contain the File-parameter.

The File-parameter itself is also encoded as PATH, which points to the file transferred.

General Syntax:

```
<File-of-filespec>  :=              PATH
                                    --  Path to a file containing a list of <Filespec-parameter>s

<Filespec-parameter>  :=            <File-parameter> <Other-parameters>
                                    --  The <Filespec-parameter> itself is fully detailed for each service
                                    --  in the appropriate clause of Part II of this Recommendation.

<File-parameter>  :=                PATH
                                    --  Path to the file transferred.
```

6.5 Handling of Documents

6.5.1 Definitions used

Some telecommunications services offer the possibility to transport a single file or a collection of files within a single communication event. The following definitions are useful:

6.5.1.1 document: A document is regarded to be a piece of information belonging to a common context which shall be (or has been) transmitted within a session through an underlying telecommunications or communications service. A document may consist of several files.

6.5.1.2 session: A session is an association between two end systems, which allows the end systems to interchange data without loss or visible interruption.

6.5.1.3 file: A file is a compilation of data as it is maintained by the underlying operating system.

6.5.2 Sending documents

Following functionality for sending documents is supported:

- send a document, consisting of one file, to one or many adressees;
- send a document, consisting of many files, to one or many adressees.

In order to successfully exchange a document via a telecommunications service, at least three items need to be specified:

- the path to the file (or the paths to the files) making up the document;
- the Transfer Format (Convert-ID), in which the document is exchanged between LA and CA;
- the Transmission Format (Type-ID), in which the document is exchanged via the network.

If the documents consists of one file, the path to the file may be given in the <Document> syntax element, the Transmission Format in the <Type> syntax element and the Transfer Format in the <Convert> syntax element. Since these syntax elements are service dependent in their values, they are described for each service in Part II of this Recommendation.

If the document consists of more files, the three components shall be gathered conforming to a service specific syntax. Since this syntax is service specific, it is also described for each service in Part II of this Recommendation. According to this syntax, the gathered components shall be put into an indirection file by the LA. The path to this indirection file then shall be specified in the appropriate syntax element of the TDD.

NOTE – The path to an indirection file is always specified with a leading "@" character.

6.5.3 Receiving documents

Following functionality for receiving documents is supported:

- receive a document, consisting of one file;
- receive a document, consisting of many files.

If a received document consists of many files, the CA builds an indirection file as described in 6.5.2 and prepends the path of the file with a "@" character.

If the received document consists of one file, the path (optionally) given by the LA shall be used to store the document itself. The Transmission Format then will be returned in the <Type>, the Transfer Format in the <Convert> syntax element.

6.5.4 Format of the indirection file

The indirection file contains lines of service specific syntax elements, one line for each file. The code presentation shall conform to the Code-ID used for TDDs. For a generic description of the syntax elements contained in the indirection file refer to 6.4.5.5.

NOTE – The indirection file shall be processed in order of appearence.

6.6 TDD Functionality

This subclause describes the functionality which is triggered by the Request TDDs. It gives information how the syntax elements shall be used in the request and in the response. Table **Error! Bookmark not defined.** explains the columns of the TDD tables used in the following subclauses.

TABLE 17/T.611

Explanation of the column headlines for the TDD tables

Column	Content
Syntax Element	Lists the syntax elements used for the TDD
C (Class)	Stands for the class of the keyword. "B" stands for basic, which means that the keyword shall be supported by all CAs, "+" means the keyword is supported by the CA if and only if the keyword is declared in the ICE
T (Type)	Specifies whether the keyword is mandatory ("m") or optional ("o"). When the keyword is optional, then the "Default" cell specifies a default value for the parameter
I/O (Input/Output)	The Input/Output column lists the requirements concerning the parameter following the keyword in the request and the response When the column states "I", it means the parameter contains an "input" value, i.e. the CA shall interpret the value as the one requested by the LA. The CA shall not modify the value and return it in the response When the column states "I/O", it means the parameter contains an "input" value (like when "I" is specified) and an "output" value is expected (i.e. set by the CA in the Response TDD) When the column states "O", it means the parameter is an "output" value (i.e. set by the CA in the Response TDD) and no value shall be set in the request (i.e. no parameter value shall be specified in the request). The response shall contain the value used for that parameter by the CA
Keyword	Gives the name of the keyword as it reads in the TDD when it is encoded as readable text. Case is NOT significant, e.g. "Function" and "funcTion" shall be interpreted equally. See also 6.4.1
Parameter	Lists the possible values for the parameter when it is encoded as readable text. The mapping of the parameter types is described in 6.4.4
Default	Lists the default values for the parameter (when applicable, i.e. when the "Type" cell reads "O")
Comment	Gives short explanation for the use of the keywords and parameters

The general rules to be used are:

– A given TDD (request and response) belongs to a Functional Class. As a consequence, the related keywords do not depend on the Functional Class.

– Keywords are divided into two categories: basic and additive ("+"). Additive keywords can be used by LAs only if they are declared in the ICE (CA-Descriptor). Basic keywords are supported by all CAs.

– The use or not of a keyword has the same signification for both basic and additive keywords.

– If a keyword is absent in a TDD, then the default value of the parameter applies. A default value is always an input parameter, i.e. the "Input/Output" classification is always "I".

– When a parameter is classified as "I" (input), then the CA shall not modify the value in the Response TDD.

Table **Error! Bookmark not defined.** explains how the "Type" and "Input/Output" classifications are used:

TABLE 18/T.611

Usage of "Type" and "Input/Output" classifications

Type	Input/Output	The Keyword
"m"	"I"	Shall be specified in the request; a parameter value shall also be specified. In the response, the keyword is also specified, with its parameter value unchanged
	"O"	Shall be specified in the request with an empty parameter value. In the response, the keyword is also present, with a significant parameter value
	"I/O"	Shall be specified in the request; the parameter value shall also be specified. In the response, the keyword is also specified, with a possibly different parameter value. The parameter value in the response is significant
"o"	"I"	In the request, the keyword may be present or not; if absent, the default value applies. The response may include the keyword only if it was specified in the request; in this case, the parameter value is unchanged
	"O"	May be specified in the request (without any parameter value) if it is desired to obtain a response parameter value for the keyword; in this case, a parameter value must be specified in the response. Otherwise no response parameter value shall be returned. No default applies to the keyword
	"I/O"	May be specified in the request with a parameter value. If not specified in the request, the default parameter value applies. In the response, the keyword is present only if it was present in the Request; in this case, the response parameter value is significant. If the request does not specify the keyword, then the response may not specify it either

6.6.1 Send

The <SendTDD> and the <SendAckTDD> are used to send one or more documents to one or more recipients. The <SendAckTDD> response acts as an acknowledgement of a previous request. For the <SendTDD> request no response is generated by the CA.

The <SendTDD> may be used to send documents to any number of recipients. <SendAckTDD> allows for only one recipient per request. <SendAckTDD> is designed for those LAs that implement the Functional Class A only; they therefore need a completion report from the CA. <SendTDD> is designed for LAs that implement the Functional Class B; they therefore have the means to trace TDD requests via the Trace function. However, both type of LAs may use <SendTDD> and <SendAckTDD>.

If a LA wants to send documents to a list of recipients, it shall use the <SendTDD> along with the <AddrList> syntax element. The transmission status (<Status> syntax element) is not available through this function, since no Response-TDD is generated. However, the status of the transmission can be obtained using the Trace:COPY functionality as described in 6.6.3.

NOTE – The Trace functionality is only available through CAs conforming to Functional Class B (FCB).

Two tables are provided: one for the <SendTDD>, the other for the <SendAckTDD> situation.

The <SendAckTDD> response is always understood as a completion status, i.e. if the request succeeds, the response shall be generated at the completion of the <SendAckTDD> request processing. So, whenever the Response-TDD is available, the LA can rely definitely on contents of the status parameter showing the result of the transmission. The status parameter will indicate one of the following 4 statuses:

1) document successfully and completely transmitted;

2) transmission failed, all attempts to transmit are given up by CA;

3) transmission partially failed, parts are already transmitted, but no complete transmission could be achieved;

4) status unknown.

The <SendTDD> and the <SendAckTDD> are belonging to Functional Classes A and B.

If multiple recipients are specified in a <SendAckTDD> request, then the <Status> and <Error> may not be specified (see Tables 19 and 20).

TABLE 19/T.611

Syntax elements of the <SendAckTDD>

Syntax Element	C	T	I/O	Text Based Encoding			Comment
				Keyword	Parameter	Default	
<SendAckFunction>	B	m	I	FUNCTION	"SendAck"	–	CA shall generate a response
<LaId>	B	m	I	LA-ID	<La-id-parameter>	–	
<ReqID>	B	m	I	REQ-ID	<Req-id-parameter>	–	
<Service>	B	m	I	SERVICE	<Service-id-parameter>	–	
<Error>	B	m	O	ERROR	<Error-parameter>	–	The error returned by the CA
<Status>	B	m	O	STATUS	<Status-parameter>	–	Returns status from the CA
<ComId>	B	o	O	COMID	<Com-id-parameter>	–	Identification of the communication (computed by the CA)
<SendTime>	B	o	I	SENDTIME	<Send-time-parameter>	"IMMEDIATE"	CA shall actually process the request at the time specified
<Comment>	+	o	I	COMMENT	<Comment-parameter>	–	
<LastTime>	+	o	I	LASTTIME	<Date-time-parameter>	CA dependent	Latest time for processing the request expressed as an absolute time
<Minor>	+	o	O	MINOR	<Error-parameter>	–	
<UserKey>	+	o	I	USERKEY	<Userkey-parameter>	–	Request and response shall contain the same parameter value. The CA shall not interpret this parameter in any manner
<Warning>	+	o	O	WARNING	<Error-parameter>	–	

TABLE 20/T.611

Syntax elements of the <SendTDD>

Syntax Element	C	T	I/O	Text Based Encoding			Comment
				Keyword	Parameter	Default	
<SendFunction>	B	m	I	FUNCTION	"Send"	–	CA shall NOT generate a response
<LaId>	B	m	I	LA-ID	<La-id-parameter>	–	
<ReqId>	B	m	I	REQ-ID	<Req-id-parameter>	–	
<Service>	B	m	I	SERVICE	<Service-id-parameter>	–	
<SendTime>	B	o	I	SENDTIME	<Send-time-parameter>	"IMMEDIATE"	CA shall actually process the request at the time specified
<Comment>	+	o	I	COMMENT	STRING	–	
<LastTime>	+	o	I	LASTTIME	<Date-Time-parameter>	CA-dependent	Latest time for processing the request expressed as an absolute time
<UserKey>	+	o	I	USERKEY	STRING	–	The CA shall not interpret this parameter in any manner

6.6.2 Receive

The <ReceiveTDD> requests for retrieval of an incoming document already received by the CA. The <ReceiveTDD> belongs to Functional Classes A and B (see Table 21).

The request specifies the telecommunications service, the storage area for the incoming file and the desired Transfer Format. For some services, the recipient's sub-address can also be used as a selector (see appropriate clause of Part II of this Recommendation).

In order to retrieve (receive) a document, the LA may use two different approaches:

– either the LA retrieves what comes next without any pre-selections, in this case the LA must be able to properly handle the received document independent of the document Transfer Format;

– or the LA first inspects a copy of a list of received documents by using the Trace <CopyTDD> request. With that list, the LA may obtain the COM-ID of the document it wishes to retrieve. This is possible only if the CA supports Functional Class B and the LA supports the <CopyTDD> function.

The <Convert> syntax element is used to store the actual Transfer Format of the document inside the Response TDD. The <Status> and <Error> syntax elements are necessary to convey the status and error code of the receive event. If the LA knows the COM-ID of the document it wishes to retrieve, it may also specify the <ComId> syntax element in the request in order to extract that document.

The *additive* <Delete> syntax element controls document deletion within the CA. For normal operation, <Delete> shall be true and the CA shall remove the document from its storage after document transfer. Thus, a document can only be received once, forcing the CA-Record into the "retrieved" state, as described in 5.6.7.1.

If <Delete> is set to false, the document shall be kept by the CA and the CA-Record shall remain in the reception state. As a consequence, on the next attempt the same document might be retrieved again.

NOTE – The CA implementation may decide to perform the deletion only virtually and may keep the "deleted" documents internally, e.g. to make them available for other LAs or to keep them for archiving purposes.

TABLE 21/T.611

Syntax elements of the <ReceiveTDD>

Syntax Element	C	T	I/O	Text Based Encoding			Comment
				Keyword	Parameter	Default	
<ReceiveFunction>	B	m	I	FUNCTION	"Receive"	–	
<LaId>	B	m	I	LA-ID	<La-id-parameter>	–	
<ReqId>	B	m	I	REQ-ID	<Req-id-parameter>	–	
<Error>	B	m	O	ERROR	<Error-parameter>	–	The error returned by the CA
<Status>	B	m	O	STATUS	<Status-parameter>	–	Returns status from the CA. When no document is available, the Response TDD shall specify the value "–"
<ComId>	B	o	I/O	COMID	<Com-id-parameter>	–	Identification of the communication (computed by the CA)
<Service>	B	o	I/O	SERVICE	<Service-id-parameter>	–	If used in the Request, used as a selection criterion
<Delete>	+	o	I	DELETE	<Boolean-parameter>	"Yes"	If true, CA shall delete retrieved document from its internal buffer
<Minor>	+	o	O	MINOR	<Error-parameter>	–	
<ReceiveTime>	+	o	O	RCVTIME	<Date-time-parameter>	–	Time of the document reception by the CA
<Warning>	+	o	O	WARNING	<Error-parameter>	–	

6.6.3 Trace

The TDDs belonging to the Trace group are used to manage the CA-Records. The Trace functionality is specific to Functional Class B.

The purpose of the various TDDs is to control the operation of a CA. Following TDDs are defined:

- <PurgeTDD> to PURGE any CA-Records in "failed", "retrieved" and "sent" states; this feature is useful to clear the CA-Records which have reached an inactive state;
- <CopyTDD> to COPY any CA-Records in any state into a file; this feature is useful to build what is commonly known as "journals" or "logs";
- <CancelTDD> to CANCEL any CA-Record in the "sending" state; this feature allows the LA to interrupt and terminate the transmission that the CA is performing[4)];
- <DeleteTDD> to DELETE any CA-Record in the "delayed" state in order to cancel it;
- <RescheduleTDD> to RESCHEDULE any CA-Record in the "failed" state to give the CA a chance to re-transmit it; this feature facilitates the management of transmissions that failed because the recipient was busy or the request was badly formed, for example;
- <DispatchTDD> to DISPATCH any CA-Record in the "reception" state to assign it to the actual recipient LA;
- <PreviewTDD> to PREVIEW a document associated with a CA-Record in the "reception" state.

It should be noted that all requests are performed on the behalf of a given LA (through the LA-ID). This limits the scope of the functions above to those CA-Records assigned to that LA-ID (except for the obvious situations where the LA-ID is not specified).

In order to delete a previous request being in the "send" state, the <DeleteTDD> shall be used.

If known, the LA may specify the <ComId> syntax element instead of the <ReqRef> syntax element.

6.6.3.1 Trace:DELETE

See Table 22.

6.6.3.2 Trace:COPY

In order to get a copy of CA-Records which refers to a specific state, the <CopyTDD> shall be used.

The <State> syntax element specifies the state of the CA-Records intended to be copied. The <Target> syntax element specifies the path to the file into which the CA will generate its output. <Error> is used to store the error code of the operation in the Response-TDD. The copied list is always in the format implied by the Code-ID of the requesting TDD. The layout of the list is implied by the RECORDS entry of the ICE. The order of the fields in the target list shall be the same as the order in which the corresponding keywords are declared by the RECORD entry.

If known, the LA may specify the <ComId> syntax element instead of the <ReqRef> syntax element.

The optional <Layout> syntax element can be used to specify the layout of the resulting copy. The values this parameter may take are explained in Table **Error! Bookmark not defined.**, examples are given in Figures **Error! Bookmark not defined.** and **Error! Bookmark not defined.**. The character representation of the copy is implied by the <TDD Header> of the corresponding <CopyTDD> (see Table 24).

[4)] It is not guaranteed that the cancel operation succeeds because of the nature of the interface.

TABLE 22/T.611

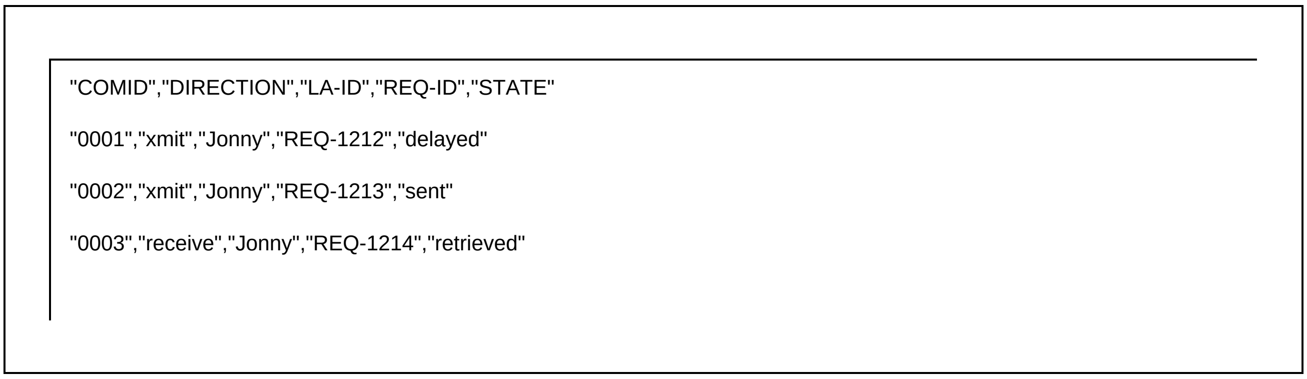
Syntax elements of the <DeleteTDD>

Syntax Element	C	T	I/O	Text Based Encoding			Comment
				Keyword	Parameter	Default	
<DeleteFunction>	B	m	I	FUNCTION	"Delete"	–	Deletes CA-Records in the "delayed" state
<LaId>	B	m	I	LA-ID	<La-id-parameter>	–	
<ReqId>	B	m	I	REQ-ID	<Req-id-parameter>	–	
<Error>	B	m	O	ERROR	<Error-parameter>	–	The error returned by the CA
<Minor>	+	o	O	MINOR	<Error-parameter>	–	
<Warning>	+	o	O	WARNING	<Error-parameter>	–	
Use of <ComId> syntax element							
<ComId>	B	m	I	COMID	<Com-id-parameter>	–	Identification of the communication computed by the CA
Use of <ReqRef> syntax element							
<ReqRef>	B	m	I	REQREF	<Req-id-parameter>	–	Reference to a previous REQ-ID

TABLE 23/T.611

Impact of the <Layout> syntax element

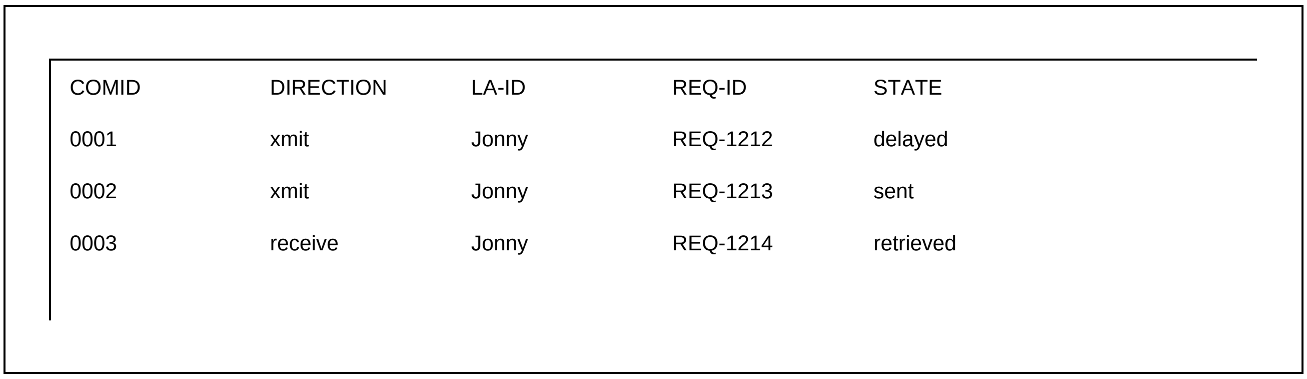
Layout-id	Explanation
0	Default value. The amount, layout and order of the fields presented in the resulting copy target are defined within the ICE by the RECORD entry. The RECORD entry gives the length of the fields and implies their order. The fields of a CA-Record are presented in one row, padded with SPACE to the length specified in the RECORD entry. A headline is not provided
1	The fields are presented row by row as comma separated values (csv), included in apostrophes. The first field row contains the keywords the field column refers to. See also Figure 7
2	The fields are presented row by row, separated by tabulator characters. The first field row contains the keywords the field column refers to. See also Figure 8



```
"COMID","DIRECTION","LA-ID","REQ-ID","STATE"

"0001","xmit","Jonny","REQ-1212","delayed"

"0002","xmit","Jonny","REQ-1213","sent"

"0003","receive","Jonny","REQ-1214","retrieved"
```

FIGURE 7/T.611

Example of comma separated value layout (Layout-id = 1)



```
COMID       DIRECTION   LA-ID       REQ-ID      STATE

0001        xmit        Jonny       REQ-1212    delayed

0002        xmit        Jonny       REQ-1213    sent

0003        receive     Jonny       REQ-1214    retrieved
```

FIGURE 8/T.611

Example of tabulator separated value layout (Layout-id = 2)

TABLE 24/T.611

Syntax elements of the <CopyTDD>

Syntax Element	C	T	I/O	Text Based Encoding			Comment
				Keyword	Parameter	Default	
<CopyFunction>	B	m	I	FUNCTION	"Copy"	–	Copy all the CA-Records of a specific state into the file pointed to by the TARGET keyword
<LaId>	B	m	I	LA-ID	<La-id-parameter>	–	
<ReqId>	B	m	I	REQ-ID	<Req-id-parameter>	–	
<State>	B	m	I	STATE	<State-parameter>	All states	State of the CA-Record
<Target>	B	m	I	TARGET	<Path-parameter>	–	Destination file name
<Error>	B	m	O	ERROR	<Error-parameter>	–	The error returned by the CA
<Layout>	B	o	I	LAYOUT	<Layout-id-parameter>	"0"	Specifies layout of resulting copy
<Direction>	B	o	I	DIRECTION	<Direction-parameter>	Both directions	
<Dispatched>	B	o	I	DISPATCHED	<Boolean-parameter>	All CA-Records	
<Previewed>	B	o	I	PREVIEWED	<Boolean-parameter>	All CA-Records	
<Service>	B	o	I	SERVICE	<Service-id-parameter>	All services	
<TimeRange>	B	o	I	RANGE	<Date-time-parameters>	All CA-Records	
<Minor>	+	o	O	MINOR	<Error-parameter>	–	
<Warning>	+	o	O	WARNING	<Error-parameter>	–	
Use of <ComId> syntax element							
<ComId>	B	m	I	COMID	<Com-id-parameter>	–	Identification of the communication computed by the CA
Use of <ReqRef> syntax element							
<ReqRef>	B	m	I	REQREF	<Req-id-parameter>	–	Reference to a previous REQ-ID

6.6.3.3 Trace:CANCEL

To cancel a previous "send" request, the LA shall use the <CancelTDD> (see Table 25).

If it is known, the LA may also specify the <ComId> syntax element instead of the <ReqRef> syntax element.

NOTE – Cancelation of a Send Request may not succeed due to the nature of the interface.

TABLE 25/T.611

Syntax elements of the <CancelTDD>

Syntax Element	C	T	I/O	Text Based Encoding			Comment
				Keyword	Parameter	Default	
<CancelFunction>	B	m	I	FUNCTION	"Cancel"	–	Cancel a CA-Record generated by a previous request
<LaId>	B	m	I	LA-ID	<La-id-parameter>	–	
<ReqId>	B	m	I	REQ-ID	<Req-id-parameter>	–	
<Error>	B	m	O	ERROR	<Error-parameter>	–	The error returned by the CA
<Minor>	+	o	O	MINOR	<Error-parameter>	–	
<Warning>	+	o	O	WARNING	<Error-parameter>	–	
Use of <ComId> syntax element							
<ComId>	B	m	I	COMID	<Com-id-parameter>	–	Identification of the communication computed by the CA
Use of <ReqRef> syntax element							
<ReqRef>	B	m	I	REQREF	<Req-id-parameter>	–	Reference to a previous REQ-ID

6.6.3.4 Trace:PURGE

In order to purge CA-Records from the CA, the <PurgeTDD> shall be used (see Table 26).

The <State> syntax element specifies the state of the CA-Records to be purged (see 5.6). The <Error> syntax element stores the error code of the operation in the Response-TDD.

If known, the LA may also specify the <ComId> syntax element instead of the <ReqRef> syntax element.

TABLE 26/T.611

Syntax elements of the <PurgeTDD>

Syntax Element	C	T	I/O	Text Based Encoding			Comment
				Keyword	Parameter	Default	
<PurgeFunction>	B	m	I	FUNCTION	"Purge"	–	Purge a CA-Record generated by a previous request
<LaId>	B	m	I	LA-ID	<La-id-parameter>	–	
<ReqId>	B	m	I	REQ-ID	<Req-id-parameter>	–	
<State>	B	m	I	STATE	<State-parameter>	All states[a)]	State of the CA-Record
<Error>	B	m	O	ERROR	<Error-parameter>	–	The error returned by the CA
<Minor>	+	o	O	MINOR	<Error-parameter>	–	
<Warning>	+	o	O	WARNING	<Error-parameter>	–	
Use of <ComId> syntax element							
<ComId>	B	m	I	COMID	<Com-id-parameter>	–	Identification of the communication computed by the CA
Use of <ReqRef> syntax element							
<ReqRef>	B	m	I	REQREF	<Req-id-parameter>	–	Reference to a previous REQ-ID

a) "All states" in this case stands for the states "sent", "failed" and "retrieved". Application of the "purge" functionality on other states shall be ignored by the CA.

6.6.3.5 Trace:RESCHEDULE

To reschedule a failed send Request, the <RescheduleTDD> shall be used. If known, the LA may also specify the <ComId> syntax element instead of the <ReqRef> syntax element (see Table 27).

TABLE 27/T.611

Syntax elements of the <RescheduleTDD>

Syntax Element	C	T	I/O	Text Based Encoding			Comment
				Keyword	Parameter	Default	
<Reschedule-Function>	B	m	I	FUNCTION	"Reschedule"	–	Reschedule a CA-Record generated by a previous send request
<LaId>	B	m	I	LA-ID	<La-id-parameter>	–	
<ReqId>	B	m	I	REQ-ID	<Req-id-parameter>	–	
<Error>	B	m	O	ERROR	<Error-parameter>	–	The error returned by the CA
<Address>	B	o	I	ADDRESS	<Address-parameter>	–	Used for the "reschedule" function only; specifies the new recipient's address
<SendTime>	B	o	I	SENDTIME	<Send-time-parameter>	"IMMEDIATE"	Process the request at the time specified
<LastTime>	+	o	I	LASTTIME	<Date-time-parameter>	CA dependent	Limit time for processing the request
<Minor>	+	o	O	MINOR	<Error-parameter>	–	
<Warning>	+	o	O	WARNING	<Error-parameter>	–	
Use of <ComId> syntax element							
<ComId>	B	m	I	COMID	<Com-id-parameter>	–	Identification of the communication computed by the CA
Use of <ReqRef> syntax element							
<ReqRef>	B	m	I	REQREF	<Req-id-parameter>	–	Reference to a previous REQ-ID

6.6.3.6 Trace:DISPATCH

To dispatch a received file to a LA, the <DispatchTDD> shall be used (see Table 28).

The <NewLa> syntax element, which may be specified repeatedly, contains the LA-ID of the new LA. On successfull return the new LA owns the received document and may retrieve it.

NOTE – The use of the dispatch function may be restricted.

TABLE 28/T.611

Syntax elements of the <DispatchTDD>

Syntax Element	C	T	I/O	Text Based Encoding			Comment
				Keyword	Parameter	Default	
<DispatchFunction>	B	m	I	FUNCTION	"Dispatch"	–	Dispatches documents associated to the CA-Record designated by the COM-ID
<LaId>	B	m	I	LA-ID	<La-id-parameter>	–	
<ReqId>	B	m	I	REQ-ID	<Req-id-parameter>	–	
<NewLa>+	B	m	I	NEWLA	<La-id-parameter>	–	Specifies the name of the new "owner" of the document. Can be repeatedly specified
<Error>	B	m	O	ERROR	<Error-parameter>	–	The error returned by the CA
<ComId>	B	m	I	COMID	<Com-id-parameter>	–	Identification of the communi-cation computed by the CA
<Minor>	+	o	O	MINOR	<Error-parameter>	–	
<Warning>	+	o	O	WARNING	<Error-parameter>	–	

6.6.3.7 Trace:PREVIEW

The <PreviewTDD> allows an administrator of the CA to retrieve and dispatch the received documents, while still keeping them available (for further dispatching). The PREVIEW function acts like the Receive function except that the CA-Record stays in the reception state instead of changing to the received state (see Table 29).

The CA-Record will also have to record whether the received document was viewed by the administrator, and whether the document was already dispatched. This allows a Trace:COPY function to search for the received documents that were not already viewed or dispatched to their final recipients.

The following principles also apply:

When the CA generates a CA-Record (called "primary" CA-Record for the purpose of the explanation) after receiving an incoming document, the primary CA-Record is assigned to the "Administrator" user. The primary CA-Record flags "viewed" and "dispatched" are set to "no".

To dispatch the incoming document, the Administrator "previews" the document thanks to the Trace:PREVIEW function. The "viewed" primary CA-Record flag is then set to "yes". Then the Administrator dispatches the document to one or many recipients. The primary CA-Record flag "dispatched" is then set to "yes".

A new copy of the primary CA-Record is generated internally for each recipient by the CA and has the following attributes:

- State = Reception
- LA-ID = Login name of the recipient to which it was dispatched
- Viewed = "no"
- Dispatched = "no"

This CA-Record is called "secondary" CA-Record for the sake of clarity. Each secondary CA-Record is now owned by the recipient to which it was dispatched. The recipient can retrieve it with the usual Trace:COPY function, and receive it with the Receive function. If the system permits, the user might also dispatch or preview the received document thanks to the same function calls.

This mechanisms allows the administrator of the CA to list all the recipients of a given document, watch who has retrieved them, redispatch a document on demand, etc.

TABLE 29/T.611

Syntax elements of the <PreviewTDD>

Syntax Element	C	T	I/O	Text Based Encoding			Comment
				Keyword	Parameter	Default	
<PreviewFunction>	B	m	I	FUNCTION	"Preview"	–	Retrieves documents associated to the CA-Record designated by the COM-ID
<LaId>	B	m	I	LA-ID	<La-id-parameter>	–	
<ReqId>	B	m	I	REQ-ID	<Req-id-parameter>	–	
<ComId>	B	m	I	COMID	<Com-id-parameter>	–	Identification of the communication computed by the CA
<Convert>	B	m	O	CONVERT	<Convert-id-parameter>	–	Transfer format of the target file
<Target>	B	m	I	TARGET	<Path-parameter>	–	Target file name
<Error>	B	m	O	ERROR	<Error-parameter>	–	The error returned by the CA
<Minor>	+	o	O	MINOR	<Error-parameter>	–	
<Warning>	+	o	O	WARNING	<Error-parameter>	–	

6.6.4 Submit

The submit functionality is designed to enable the CA to perform various utility functions.

The purpose of the submit functionality is to ask the CA to perform additional functions that it may implement, like file format conversions, or printing incoming documents. These tasks are normally not accomplished by the CA but some CA manufacturers may wish to support them. An example of a situation where the "printing" feature may be useful could be the situation of a "CA server" on a LAN. The submit functionality supports the following TDDs:

– <PrintTDD> to PRINT a document, given its path and format;

– <ConvertTDD> to CONVERT a document, given its path, input format, output format and output filename;

– <CheckTDD> to CHECK the Transfer Format of a document, given its path and the Transfer Format against which it should be checked.

6.6.4.1 Submit: PRINT

See Table 30.

TABLE 30/T.611

Syntax elements of the <PrintTDD>

Syntax Element	C	T	I/O	Text Based Encoding			Comment
				Keyword	Parameter	Default	
<PrintFunction>	B	m	I	FUNCTION	"Print"	–	Submit the printing task to the CA
<LaId>	B	m	I	LA-ID	<La-id-parameter>	–	
<ReqId>	B	m	I	REQ-ID	<Req-id-parameter>	–	
<FileName>	B	m	I	FILENAME	<Path-parameter>	–	Original file
<InFormat>	B	m	I	INFORMAT	<Convert-id-parameter>	–	Original format
<Error>	B	m	O	ERROR	<Error-parameter>	–	The error returned by the CA
<Printer>	+	o	I	PRINTER	<Printer-id-parameter>	"STD"	"STD" stands for the standard printer configured inside of the CA. Further possible values for the values of the Printer-id are declared on a per CA basis in the "PRINT" component of the ICE (see 9.5)
<Minor>	+	o	O	MINOR	<Error-parameter>	–	
<Warning>	+	o	O	WARNING	<Error-parameter>	–	

6.6.4.2 Submit: CONVERT

See Table 31.

TABLE 31/T.611

Syntax elements of the <ConvertTDD>

Syntax Element	C	T	I/O	Text Based Encoding			Comment
				Keyword	Parameter	Default	
<ConvertFunction>	B	m	I	FUNCTION	"Convert"	–	Submit the converting task to the CA
<LaId>	B	m	I	LA-ID	<La-id-parameter>	–	
<ReqId>	B	m	I	REQ-ID	<Req-id-parameter>	–	
<FileName>	B	m	I	FILENAME	<Path-parameter>	–	Original file
<Target>	B	m	I	TARGET	<Path-parameter>	–	Target file
<InFormat>	B	m	I	INFORMAT	<Convert-id-parameter>	–	Original format
<OutFormat>	B	m	I	OUTFORMAT	<Convert-id-parameter>	–	Target format
<Error>	B	m	O	ERROR	<Error-parameter>	–	The error returned by the CA
<Minor>	+	o	O	MINOR	<Error-parameter>	–	
<Warning>	+	o	O	WARNING	<Error-parameter>	–	

6.6.4.3 Submit: CHECK

See Table 32.

TABLE 32/T.611

Syntax elements of the <CheckTDD>

Syntax Element	C	T	I/O	Text Based Encoding			Comment
				Keyword	Parameter	Default	
<DeleteFunction>	B	m	I	FUNCTION	"Check"	–	Submit the checking task to the CA
<LaId>	B	m	I	LA-ID	<La-id-parameter>	–	
<ReqId>	B	m	I	REQ-ID	<Req-id-parameter>	–	

TABLE 32/T.611 (*end*)

Syntax elements of the <CheckTDD>

Syntax Element	C	T	I/O	Text Based Encoding			Comment
				Keyword	Parameter	Default	
<FileName>	B	m	I	FILENAME	<Path-parameter>	–	Original file
<Check>	B	m	I	CHECK	<Convert-id-parameter>	–	Format to be checked against
<Error>	B	m	O	ERROR	<Error-parameter>	–	The error returned by the CA
<Minor>	+	o	O	MINOR	<Error-parameter>	–	
<Warning>	+	o	O	WARNING	<Error-parameter>	–	

6.6.5 Extend

The extend functionality is divided into of three different TDD types. These are:

- <ExtendTDD> to EXTEND the functionality on a general basis;
- <NationalTDD> to extend the functionality on a NATIONAL basis;
- <PrivateTDD> to extend the functionality on a PRIVATE basis.

6.6.5.1 Extend: EXTEND

The <ExtendTDD> is designed to provide extended features that are not vital for the functioning of the interface, but use of which becomes widespread or necessary. The <ExtendTDD> is an additional facility (see Table 33).

TABLE 33/T.611

Minimum syntax elements of the <ExtendTDD>

Syntax Element	C	T	I/O	Text Based Encoding			Comment
				Keyword	Parameter	Default	
<ExtendFunction>	B	m	I	FUNCTION	"Extend"	–	
<SubFunction>	B	m	I	SUBFUNC	<Subfunction -parameter>	–	For assignement of values see appropriate clause of Part II of this Recommendation
<LaId>	B	m	I	LA-ID	<La-id-parameter>	–	
<ReqId>	B	m	I	REQ-ID	<Req-id-parameter>	–	
<Error>	B	m	O	ERROR	<Error-parameter>	–	The error returned by the CA

6.6.5.2 Extend: NATIONAL

The <NationalTDD> is designed to provide national features that are specific to each country. The <NationalTDD> is an additional facility (see Table 34).

TABLE 34/T.611

Minimum syntax elements of the <NationalTDD>

Syntax Element	C	T	I/O	Text Based Encoding			Comment
				Keyword	Parameter	Default	
<NationalFunction>	B	m	I	FUNCTION	"National"	–	
<SubFunction>	B	m	I	SUBFUNC	<Subfunction -parameter>	–	Parameter values are for further study
<LaId>	B	m	I	LA-ID	<La-id-parameter>	–	
<ReqId>	B	m	I	REQ-ID	<Req-id-parameter>	–	
<Error>	B	m	O	ERROR	<Error-parameter>	–	The error returned by the CA

6.6.5.3 Extend: PRIVATE

The <PrivateTDD> is designed to provide private features that are specific to each manufacturer. The <PrivateTDD> is an additional facility (see Table 35).

TABLE 35/T.611

Minimum syntax elements of the <PrivateTDD>

Syntax Element	C	T	I/O	Text Based Encoding			Comment
				Keyword	Parameter	Default	
<PrivateFunction>	B	m	I	FUNCTION	"Private"	–	
<SubFunction>	B	m	I	SUBFUNC	<Subfunction -parameter>	–	Parameter values are for further study
<LaId>	B	m	I	LA-ID	<La-id-parameter>	–	
<ReqId>	B	m	I	REQ-ID	<Req-id-parameter>	–	
<Error>	B	m	O	ERROR	<Error-parameter>	–	The error returned by the CA

7 Exchange Method

This clause describes how TDDs and related data are transferred between LAs and CAs.

To transfer the TDDs and associated data, this Recommendation defines an abstract Exchange Method between LAs and CAs. This Exchange Method can be realized by different means, which are more or less platform dependent.

To aid LA developers in writing portable code, the platform dependencies have been encapsulated on a functional level which consists of the Basic Exchange Method Functions (BEM Functions).

Figure 9 depicts this situation.

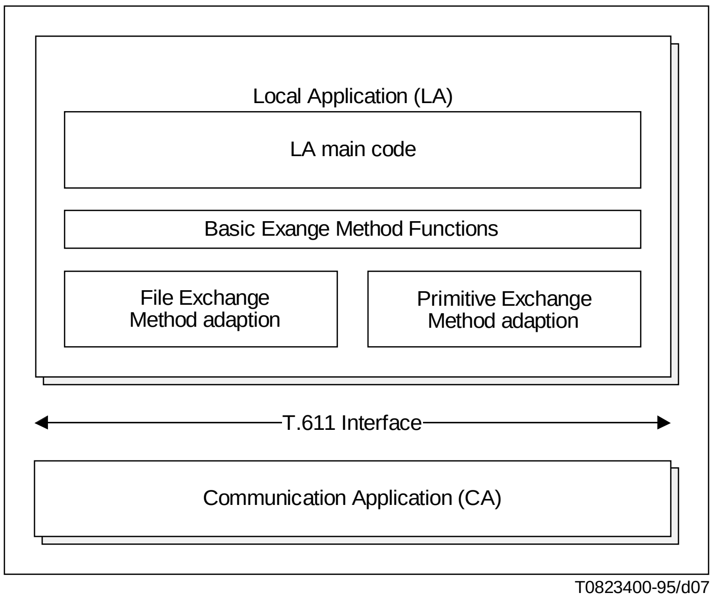

Local Application (LA)
LA main code
Basic Exange Method Functions
File Exchange Method adaption
Primitive Exchange Method adaption
T.611 Interface
Communication Application (CA)
T0823400-95/d07

FIGURE 9/T.611

Usage of the Basic Exchange Method Functions

The BEM Functions present a functional interface, which decouples the code of the LA from the real exchange methods to be used.

The real exchange methods defined for the various platforms are either of type File Exchange Method or Primitive Exchange Method. This two types of exchange method are fully detailed in 7.3.

NOTE – Since it is only required that the binary exchange of TDDs and associated data conforms to this Recommendation, the use of the BEM Functions within the code of the LA is optional. The only requirement the LA shall meet is the conformation to the real exchange method defined for each platform. Thus a LA implementor may decide to interface directly to one of the real exchange methods defined.

7.1 Overview of Basic Exchange Method Functions

NOTE – The Basic Exchange Method Functions described in this subclause superseds the method documented in the 1992 version of this Recommendation. For details of the differences refer to Annex F where the previous version of the Basic Exchange Method Functions has been included for convenience.

Different implementations of the Basic Exchange Method are possible. All have in common the Basic Exchange Method Functions shown in Table 36[5].

[5] In order to differentiate the set of BEM Functions defined in this Recommendation from the set defined in the 1992 version of this Recommendation, the names of the BEM Functions are starting with an uppercase letter "E".

TABLE 36/T.611

Summary of Exchange Method Functions

BEM Function	Purpose
ELogin	LA opens a communication channel between LA and CA
ELogout	LA closes the communication channel opened with ELogin ( )
EPutTDD	LA hands over a TDD to a CA
EPutData	LA hands over data referring to a specific TDD
EPollTDD	LA asks CA whether a Response TDD is available
EGetTDD	LA gets a Response TDD from the CA
EGetData	LA gets data referring to a specific TDD
ESetAlarm	LA sets an alarm inside of the CA. CA then will wake up LA on specific events
EAbortData	LA aborts transfer of data
NOTE – The Basic Exchange Method has been designed to work with many configurations, such as LA and CA being on the same equipment or CA being a communication server on a LAN. The only assumption is that the ICE shall be accessible from each LA.	

In general, the Basic Exchange Method Functions rely on a login procedure (ELogin) which returns a Connection-ID. This Connection-ID is then used in all subsequent invocations of the Basic Exchange Method functions. The login procedure is comparable to opening an interaction channel between a LA and a CA.

7.1.1 Sequence of Basic Exchange Method Functions

Firstly, a LA needs to login (function ELogin) to the requested CA. No TDD exchange can take place before the login procedure is completed. By nature, the login procedure is a synchronous mechanism, i.e. the login requires a response before another action can take place.

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

– build the Request TDD (by any appropriate means);

– invoke the EPutTDD function;

– invoke the EPutData function, if associated data have to be transferred.

When a LA needs to be informed of possible events bound for it, the LA shall:

– invoke the EPollTDD function.

When a LA needs to retrieve a Response TDD (the EPollTDD function above indicated available Response TDDs), the LA shall:

– invoke the EGetTDD function;

– invoke the EGetData function, if associated data have to be retrieved.

When a LA receives an alarm from the CA – applicable only if the CA supports alarms as stated in the ICE (see clause 9) and if the LA implements the alarm invocation function –, it shall:

- invoke the EPollTDD function;
- invoke the EGetTDD function;
- invoke the EGetData function, if associated data have to be retrieved.

When a LA no longer needs to dialogue with a CA, it shall logout (function ELogout) from this CA. The CA thus knows the LA-CA communications path is broken and thus will not use the alarm mechanism any longer.

The EPutTDD, EPollTDD, EGetTDD, ESetAlarm, ECallBackRoutine, ELogin and ELogout functions are synchronous, i.e. the LA can proceed only when those functions have returned. The functions and their calling and return parameters are shown in 7.2.

Figure 10 illustrates the behaviour described above.

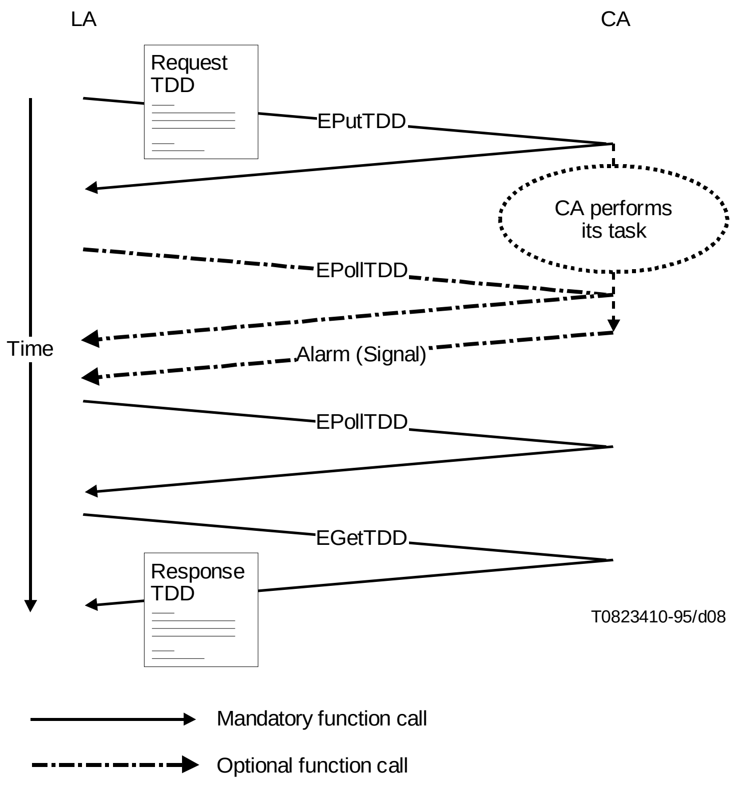


FIGURE 10/T.611

Sample sequence of Basic Exchange Method Functions

It is assumed that the LA has already logged into the CA. Data transfer functions are not shown.

7.1.2 Alarm Support

If the CA implements the optional alarm feature, it means that the CA implements the ESetAlarm function. This function allows a CA to wake up a given LA on specific alarm events.

Support of the alarm feature shall be stated in the ICE (see clause 9).

7.1.3 Connection-ID

In order to identify a LA-CA communications path, the Connection-ID is defined. The Connection-ID is computed by the CA on invocation of the Login request. The LA shall use this identifier throughout the interchange with the same CA until the LA logs out.

NOTE – The Connection-ID is different from the COM-ID; the COM-ID identifies communication events occurring in a CA.

7.1.4 CA-ID

The CA-ID identifies a CA. This identifier allows a LA to have simultaneous interactions with multiple LAs. Therefore the CA-ID is a parameter used in each EM function call.

7.2 Basic Exchange Method Functions

The functions described below replace or supplement the functions of the basic exchange mechanism of the 1992 version of this Recommendation (see also Annex F). For differentiation purposes, the names of the functions defined within this Recommendation start with the uppercase letter "E".

All functions perform synchroneously, i.e the caller cannot continue execution until the called function returns.

Since the implementation of the BEM Functions is platform dependent, they are described in a generic way in the subclauses to follow. The implementation of the functions for the various platforms is described in Part IV of this Recommendation.

NOTE – The order the parameters are appearing in the following description of the BEM Functions is important for compatibility.

The data types used for the generic description of the BEM Functions are defined in Table 37. They are mapped to real data types in the appropriate clauses of Part IV of this Recommendation.

The Direction classification "Input" stands for "parameter shall be presented on function call", "Output" stands for "parameter valid on return of function".

TABLE 37/T.611

Data types used for description of BEM Functions

Data type	Explanation
String	Stands for a string of characters
Integer	Stands for a number in the minimum range of $+2^{15}$ ... $(2^{15} - 1)$
Boolean	Stands for a variable taking the value true or false
Memory address	Stands for an address inside of LA’s memory

7.2.1 Function ELogin

The function ELogin shall be supported by the CA. It shall be invoked by the LA before any LA-CA interchange of Request TDDs and responses.

7.2.1.1 Purpose

The function ELogin returns to the LA a Connection-ID that will be used all along the LA-CA interaction until the LA logs out.

Furthermore the ELogin function allows to select a CA on user-provided criteria and returns a CA-ID, which shall be used in subsequent BEM function calls to address the selected CA.

Selection of a CA is performed by specifying a string containing keywords like "FAX", "ECM" or "EMAIL", separated by spaces, in the Selector parameter. Since it can be performed by external processes, the selection mechanism is out of the scope of this Recommendation.

If the Selector parameter is specified, then the CA-ID parameter shall not be specified; when the function call returns, either the CA-ID parameter contains a valid CA identifier that can be used in further BEM function calls, or the CA-ID parameter is empty if no CA matched the selection criteria. If the Selector parameter is not specified in the function call, then no selection is to be performed. In this case the CA-ID parameter shall be specified.

The ELogin function is the place where a CA may control access of a LA to it. 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.

7.2.1.2 Behaviour

The CA checks the parameters of the ELogin call. If they match, it then generates a Connection-ID the LA shall use subsequently in other BEM Function calls. In addition, if the Selector parameter specifies criteria, the function returns 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.

7.2.1.3 Parameters

See Table 38.

TABLE 38/T.611

Parameters of the ELogin function

Parameter	Data type	Comment	Direction
Login-name	String	Name of the LA user connecting to the CA (differs from the LA-ID)	Input
Password	String	A password string	Input
Selector	String	A user-provided string specifying connection criteria	Input
CA-ID	Integer	Identifier of the CA that accepts the connection	Output
Connection-ID	Integer	Identifier (handle) of the LA-CA data channel. Returned by the CA if the ELogin function succeeds. Otherwise (e.g. identification failed) the CA sets the value to zero and an appropriate return code is given by the Status parameter (see below)	Output
Status	Integer	Return code; a value of zero indicates successful operation	Output

7.2.2 Function EPutTDD

The function EPutTDD shall be supported by the CA. It may be invoked by a LA. The EPutTDD also supports the identification of the data files to be transferred to the CA.

Such data files are typically address-list files or documents or both.

7.2.2.1 Purpose

The EPutTDD function conveys a Request TDD to the CA and allows the LA to specify which files need to be transferred to the CA. The actual transfer of data files to the CA is performed with the EPutData function.

7.2.2.2 Behaviour

The CA copies the Request TDD carried by the EPutTDD function into its internal structures. The result is reported immediately to the requesting LA. If the LA requests a Data-ID by specifying a Data-ID address, CA returns the identifier of a data group as Data-ID.

If this identifier is NULL, then no data files (or buffers) are to be transmitted to the CA. If the Data-ID is not NULL, then the LA performs the subsequent data file transfers with EPutData function. The transfers shall occur immediately after the call to EPutTDD function.

NOTE – The Data-ID returned may be valid for a limited time only.

7.2.2.3 Parameters

See Table 39.

TABLE 39/T.611

Parameters of the EPutTDD function

Parameter	Data type	Comment	Direction
Connection-ID	Integer	Identifier (handle) of the LA-CA data channel returned by the ELogin function	Input
CA-ID	Integer	Identifier of the CA as returned by the ELogin function	Input
TDD-location	Memory address	Memory location of the TDD to be passed to the CA. After return from the function the LA's TDD may be deleted or used for other purposes	Input
TDD-size	Integer	Size in octets of the TDD passed to the CA	Input
Data-ID	Integer	Identifier of the data group, returned by the CA if requested	Input/Output
Status	Integer	Return code; a value of zero indicates successful operation	Output

7.2.3 Function EPutData

This function allows the actual transfer of data (address lists and/or data files) from the LA to the CA. Depending on the implementation the data can either be files of buffers.

7.2.3.1 Purpose

The group of files transferred within one function call relates to one previous Request TDD that was transmitted to the CA by means of a previous EPutTDD function call.

The group is identified by the Data-ID that the CA returned as a result of the previous EPutTDD function call.

7.2.3.2 Behaviour

This function is invoked by the LA after the corresponding Request TDD was transmitted to the CA.

The LA shall build a data structure (Data-Descriptor) that describes where the files/buffers to be conveyed to the CA are located. The specification of the data structure is language-dependent but cross platform compatibility ensures that the binary layout does not depend on the platforms or languages.

The LA may transfer all the data in one shot or in multiple calls. The parameter Next indicates whether the function call is the last in the data group or that more function calls relevant to that data group are to follow.

7.2.3.3 Parameters

See Table 40.

TABLE 40/T.611

Parameters of the EPutData function

Parameter	Data type	Comment	Direction
Connection-ID	Integer	Identifier (handle) of the LA-CA data channel	Input
CA-ID	Integer	Identifier of the CA	Input
Data-ID	Integer	Identifier of the data group, returned by the CA	Input
Data-Descriptor	Memory address	Memory address of a data structure specifying which files/buffers have to be conveyed to the CA	Input
Next	Boolean	Indicator stating whether more data will be conveyed in a subsequent call to the function EPutData or that no more data is to follow (end of the data group)	Input
Status	Integer	Return code; a value of zero indicates successful operation	Output

7.2.4 Function EPollTDD

The EPollTDD function asks the CA how many Response TDDs are waiting to be handled by the requesting LA. The EPollTDD returns the number of Response TDDs pending and the type and size of the first Response TDD that will be returned by the next call to the EGetTDD function.

The EPollTDD function furthermore allows to select the type of TDD that is to be returned by the CA.

7.2.4.1 Purpose

The EPollTDD function allows to poll the CA in order to know how many Response TDDs are pending for retrieval.

7.2.4.2 Behaviour

The function allows the LA to select the type of Response TDD to be polled by giving a specific TDD-type parameter. If the LA specifies no TDD-type (TDD-type set to zero), the function selects, if available, any type of TDD and returns the selected type. The TDD selected, if any, then may be retrieved by a subsequent EGetTDD function call.

The EPollTDD function also returns the size of the Response TDD that will be returned, if any. The CA then should allocate enough storage space to receive the Response TDD returned by the next call to EGetTDD. The size given by the CA is an indication of the minimum buffer size that will hold the Response TDD.

Furthermore, the amount of pending Response TDDs – of the selected type – is returned. If the count returned is zero, no Response TDDs – of the selected type – are pending.

When the CA has many Response TDDs available, it chooses the one it will return first. This Response TDD is the TDD that will be transmitted to the LA at the next EGetTDD function call emitted by the same LA.

When no Response TDD is available for the requesting LA, the TDD-count is set to the value zero, in which case the TDD size returned is set to zero as well.

If the TDD that was submitted is erroneous or unknown, e.g. the <TDD Header> is missing, then the function may return the original Request TDD. In this case the CA shall set the returned TDD-type to zero and the return status to a non-zero value.

The TDD-type values assigned are shown by Table 41.

TABLE 41/T.611

Assignment of TDD-types

Response TDD	TDD-type	Response TDD	TDD-type
No TDD-type[a)]	00_{HEX}	DISPATCH Response	35_{HEX}
SENDACK Response	10_{HEX}	PREVIEW Response	36_{HEX}
RECEIVE Response	20_{HEX}	PRINT Response	40_{HEX}
COPY Response	30_{HEX}	CONVERT Response	41_{HEX}
DELETE Response	31_{HEX}	CHECK Response	42_{HEX}
CANCEL Response	32_{HEX}	EXTEND Response	50_{HEX}
PURGE Response	33_{HEX}	NATIONAL Response	60_{HEX}
RESCHEDULE Response	34_{HEX}	PRIVATE Response	70_{HEX}
a) Used as selector (on input) 00_{HEX} stands for "all TDD-types"; as return value it stands for unknown or erroneous TDD was submitted, or no Response TDD available.			

7.2.4.3 Parameters

See Table 42.

TABLE 42/T.611

Parameters of the EPollTDD function

Parameter	Data type	Comment	Direction
Connection-ID	Integer	Identifier (handle) of the LA-CA data channel returned by the ELogin function	Input
CA-ID	Integer	Identifier of the CA	Input
TDD-type	Integer	Selects/returns the type of Response TDD expected/available	Input/Output
TDD-size	Integer	Size of the next Response TDD ready to be retrieved	Output
TDD-count	Integer	Count of Response TDDs pending. Zero indicates: no response pending	Output
Status	Integer	Return code; a value of zero indicates successful operation	Output

7.2.5 Function EGetTDD

The function EGetTDD shall be supported by the CA. It may be invoked by a LA. EGetTDD allows to retrieve a Response TDD and get a data group identifier to retrieve the accompanying data files/buffers.

7.2.5.1 Purpose

EGetTDD allows the LA to retrieve a Response TDD into memory and obtain a handle to a group of data files/buffers. The handle is useful for all TDD types that involve data files/buffers, e.g. Receive or Trace:COPY.

7.2.5.2 Behaviour

The LA specifies the memory address where the CA shall copy a Response TDD that is available for the LA.

The CA shall return to the LA the Response TDD that was qualified by the previous EPollTDD function emitted by the same LA. The LA shall have prepared a recipient Response TDD area in its internal structures. The invocation of a EGetTDD function by a LA shall always be preceded by a EPollTDD function call. On call, the LA shall pre-set the TDD size parameter to the size of its response area. On return, the TDD size parameter shall hold the size of the Response TDD the CA provided.

If the LA invokes two or more consecutive EGetTDD functions (without an intermediate EPollTDD function call) the CA may return invalid information.

The EGetTDD function also returns the Data-ID identifier as a handle to receive the accompanying data files (with a subsequent EGetData function call).

If the Data-ID parameter value is not zero, then data files need to be retrieved by a subsequent call to the EGetData function. If the Data-ID parameter is zero, then no additional data is to be retrieved from the CA.

7.2.5.3 Parameters

See Table 43.

TABLE 43/T.611

Parameters of the EGetTDD function

Parameter	Data type	Comment	Direction
Connection-ID	Integer	Identifier (handle) of the LA-CA data channel returned by the ELogin function	Input
CA-ID	Integer	Identifier of the CA	Input
TDD-location	Memory address	Specifies the location where the Response TDD shall be stored	Input
TDD-size	Integer	Size of the buffer reserved for storing the Response TDD	Input/Output
Data-ID	Integer	Identifier of the data group, if any	Output
Status	Integer	Return code; a value of zero indicates successful operation	Output

7.2.6 Function EGetData

This function allows the transfer of data files (or buffers) from the CA to the LA. It is called by the LA.

7.2.6.1 Purpose

This function is used to retrieve the data from the CA after receiving a Response TDD (with the EGetTDD function) that specifies that data is to be retrieved (parameter Data-ID is not zero).

7.2.6.2 Behaviour

Immediately after the EGetTDD, the LA shall call the EGetData in order to receive the corresponding data (files or buffers). The CA returns a data descriptor containing information about the received data. The CA also returns whether additional calls to EGetTDD should be performed by the LA to retrieve further pertaining data.

The group of files transferred within one function call relates to one Response TDD (through a EGetTDD function call). This group is identified by the Data-ID parameter (LA provided).

7.2.6.3 Parameters

See Table 44.

TABLE 44/T.611

Parameters of the EGetData function

Parameter	Data type	Comment	Direction
Connection-ID	Integer	Identifier (handle) of the LA-CA data channel returned by the ELogin function	Input
CA-ID	Integer	Identifier of the CA	Input
Data-ID	Integer	Identifier of the data group	Input
Data-Descriptor	Memory address	Storage address of a descriptor for the data to be retrieved, passed to the CA on input and filled by the CA on output	Input/Output
Next	Boolean	States whether additional data should be retrieved	Output
Status	Integer	Return code; a value of zero indicates successful operation	Output

7.2.7 Function ESetAlarm

The function ESetAlarm can optionally be supported by the CA. In this case, it can optionally be invoked by a LA. If the ESetAlarm function is used by a LA, that LA shall supply an AlarmHandler. An AlarmHandler is a function which receives the subsequent alarms generated by the CA.

A CA shall declare the support of the ESetAlarm function in the ICE.

7.2.7.1 Purpose

This function allows the LA to register to the connected CA the type of alarm events the LA supports. This function shall not be invoked if the CA does not understand the alarm mechanism. Whether a CA understands the alarm mechanism may be obtained from the ALARM element of the ICE (see clause 9).

The ESetAlarm function tells the CA that it can give an alarm to the LA by invoking the AlarmHandler supplied.

The LA may temporarily disable an alarm event by disabling the corresponding address of the AlarmHandler, e.g. by setting the address to NULL. Restoring the alarm event for that alarm type is performed by enabling the address of the AlarmHandler, e.g. by setting the address to the actual entry point of the AlarmHandler.

The alarm events defined are shown in Table 45.

7.2.7.2 Behaviour

The CA supporting the ESetAlarm function shall record the location of the AlarmHandler function assigned by the LA. In the minimum, the CA can record as many AlarmHandler locations as there are different logged-in LAs. If an event registered to a particular LA occurs, the CA shall send the alarm to the particular LA by invoking its AlarmHandler.

TABLE 45/T.611

Alarm events for the ESetAlarm function

Event name	Comment
ASYNC_RESPONSES	The LA will not poll the CA. The CA should send an alarm each time a Response TDD is available
QUEUE_FULL	The CA will not accept any further TDDs unless the CA is polled
DOCUMENT_RECEIVED	A new document can be received
CONNECTION_LOST	The LA-CA connection is lost
SEND_SUCCESS	A document was successfully sent
SEND_FAILED	A document failed to be sent
CORRUPTED_TDD	The CA received a corrupted/unrecognizable TDD
SEND_EVENT	A send event occurred in the CA
RECEIVE_EVENT	A receive event occurred in the CA
CA_WILL_STOP	The CA will no longer process requests. Logout immediately
ALARMS_UNAVAILABLE	Alarms are no longer available
TDD_RESP_AVAILABLE	A TDD response is available. The LA should poll the CA as soon as possible
NOTE – Some alarms need additional data to give accurate information about the event.	

7.2.7.3 Parameters

See Table 46.

TABLE 46/T.611

Parameters of the ESetAlarm function

Parameter	Data type	Comment	Direction
Connection-ID	Integer	Identifier (handle) of the LA-CA data channel returned by the ELogin function	Input
CA-ID	Integer	Identifier of the CA	Input
Alarm-event	Integer	Combination of the types of alarms the LA may react to	Input
AlarmHandler	Memory address	Information addressing the entry point of the AlarmHandler. This entry point shall be called by the CA when one of the registered events occurs This parameter as well as the AlarmHandler itself is platform-dependent	Input
Status	Integer	Return code; a value of zero indicates successful operation	Output

7.2.7.4 AlarmHandler Function

Depending on the platform, the alarm triggered may have parameters that detail the reason for the alarm.

The AlarmHandler function defines a mechanism that allows a CA to alert the LA that some Response TDDs are available. The use of this optional mechanism can improve the flow control between LAs and CAs on heavily loaded systems.

The AlarmHandler is implemented by the LA. The CA calls that function when it needs to trigger an alarm. The alarms the CA may trigger are specified by the LA by means of the ESetAlarm function.

Some alarms may have parameters. They are passed to the LA along with the function call.

Some alarms imply that the LA polls the CA after having received the alarm (e.g. when the CA has received a new document). If the CA does not poll the CA quickly enough, the CA may trigger the alarm repeatedly until the LA performs the expected action.

Since theAlarmHandler function is platform dependent, it is described in greater detail in the appropriate clauses of Part IV of this Recommendation.

7.2.8 Function EAbortData

The EAbortData function allows to cancel the ongoing transfer of data. It has the effect of canceling the corresponding TDD too.

7.2.8.1 Purpose

This function is invoked by the LA in order to cancel a data transfer started by EPutData or EGetData.

7.2.8.2 Behaviour

When aborting an EPutData data transfer, the CA shall destroy the corresponding Request TDD that was conveyed by the previous EPutTDD function call.

When aborting an EGetData data transfer, the CA shall retain the corresponding Response TDD that was retrieved by the previous EGetTDD function call.

Abortion of the data transfers in either direction may occur only between two EGetData or EPutData function calls respectively.

Since the LA is responsible for the data transfer between the LA and the CA, calling EAbortData after the last data block has no effect (the data transfer is completed anyway). For the same reason, calling EAbortData before the first data block has the effect of cancelling the whole data transfer and the corresponding TDD.

7.2.8.3 Parameters

See Table 47.

7.2.9 Function ELogout

The function ELogout shall be supported by the CA. It shall be invoked by the LA on completion of any LA-CA interchange of Request TDDs and responses.

7.2.9.1 Purpose

The function ELogout returns to the LA a status that states whether the LA-CA interaction has finished orderly.

7.2.9.2 Behaviour

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

TABLE 47/T.611

Parameters of the EAbortData function

Parameter	Data type	Comment	Direction
Connection-ID	Integer	Identifier (handle) of the LA-CA data channel returned by the ELogin function	Input
CA-ID	Integer	Identifier of the CA	Input
Data-ID	Integer	Data group identifier	Input
Status	Integer	Return code; a value of zero indicates successful operation	Output

7.2.9.3 Parameters

See Table 48.

TABLE 48/T.611

Parameters of the ELogout function

Parameter	Data type	Comment	Direction
Connection-ID	Integer	Identifier (handle) of the LA-CA data channel returned by the ELogin function	Input
CA-ID	Integer	Identifier of the CA	Input
Status	Integer	Return code; a value of zero indicates successful operation	Output

7.3 Implementation of Basic Exchange Method Functions

The Basic Exchange Method Functions offer a functional interface, which has to be adapted to the underlying platform in a well defined way in order to provide a binary compatible interface.

For this purpose this Recommendation defines two real exchange mechanisms:

1) the Primitive Exchange Method; and

2) the File Exchange Method.

The Primitive Exchange Method is fast, but platform dependent; the File Exchange Method is far less platform independent, but slower and lacks the ability to implement all features offered by the Basic Exchange Method Functions.

The platforms may impose the type of exchange method to support. See Part IV of this Recommendation.

7.3.1 Primitive Exchange Method

The Primitive Exchange Method implementation utilizes function calls. It thus requires from the CA to provide an entry point for each of the Basic Exchange Method Functions. To achieve binary compatibility, the provision of this entry point shall conform to the definitions made by this Recommendation for each platform (see Part IV of this Recommendation).

7.3.2 File Exchange Method

The File Exchange Method implementation requires that TDDs are exchanged via files. For this reason, common file directories have to be agreed upon and shared between the LA and the CA. Therefore, the CA shall declare the complete paths for these directories in the appropriate section of the ICE.

Three file directories or, more abstract, areas have to be configured:

- an input area for TDDs that are transferred from the LA to the CA;
- an area for Response TDDs;
- an area for jobs that could not be processed due to syntactical errors or other errors. The CA removes incorrect TDDs from the input area and places them in this area.

These three areas are referred to symbolically as COM_JOB, COM_ACK and COM_ERR respectively.

Because they are implementation-and installation-specific, the complete paths to these areas shall be configurable during installation of the CA.

If a CA has to support multiple LA connections, it is suggested that the CA sets up three "mother"-areas (directories) as described above and creates a sub-area (subdirectory) inside of each "mother"-area for each LA which may be connected. The name of each sub-area may be derived from the LA-ID of the connecting LAs.

Adaptation of the Basic Exchange Method Functions to the File Exchange Method is rather complex, since the Basic Exchange Method Functions have to be emulated by the adapter.

NOTE – A LA implementation may – although not encouraged by this Recommendation – also interface directly to the File Exchange Method without using Basic Exchange Method Functions. The disadvantage of this approach is that portability to other primitive based platforms is more difficult to obtain.

7.3.2.1 TDD Transfer

To issue a function request, the LA places a TDD as a file into the COM_JOB area.

Some implementations, mainly on single tasking platforms, also require that the LA executes a SYNC program. SYNC stands for synchronization. The SYNC program ensures that the CA receives control and can process TDDs. Whether there is a need for executing a SYNC is stated in the CA-Descriptor of the corresponding CA.

When the CA has finished processing the TDD, it places the Response-TDD in the COM_ACK response area and deletes the Request-TDD in the COM_JOB input area. The Response-TDD placed by the CA has the same name as the former Request-TDD.

The complete cycle of the job transfer is shown in Figure 11.

NOTE – It is not guaranteed that Response-TDDs will be provided in the same sequence as Request-TDDs. Processing is not necessarily sequential.

7.3.2.2 Error Handling

Request-TDDs not processable by the CA (due to syntactical errors, etc.) are copied to the COM_ERR area and deleted from the COM_JOB area. The keyword FATAL is inserted at the end of the Response TDD and an error message with the form [Num/Text] is generated in the appended parameter field.

7.3.2.3 Sync Mechanism

If a SYNC program execution is required by the CA, the LA shall proceed as follows:

- after a Request-TDD is stored, the LA shall issue a SYNC to the CA ("SyncJob" function);
- before the LA polls a Response-TDD, another SYNC to the CA shall be issued ("SyncAck" function).

Under DOS, OS/2, UNIX and UNIX-like operating system, the SYNC mechanism shall be implemented by executing the "APPLI/COM" program from the LA.

The "SyncJob" and "SyncAck" functions (see above) are encoded as follows:

SyncJob: *EXEC* APPLI/COM JOB

SyncAck: *EXEC* APPLI/COM ACK

EXEC stands for the platform specific command to launch an executable program.

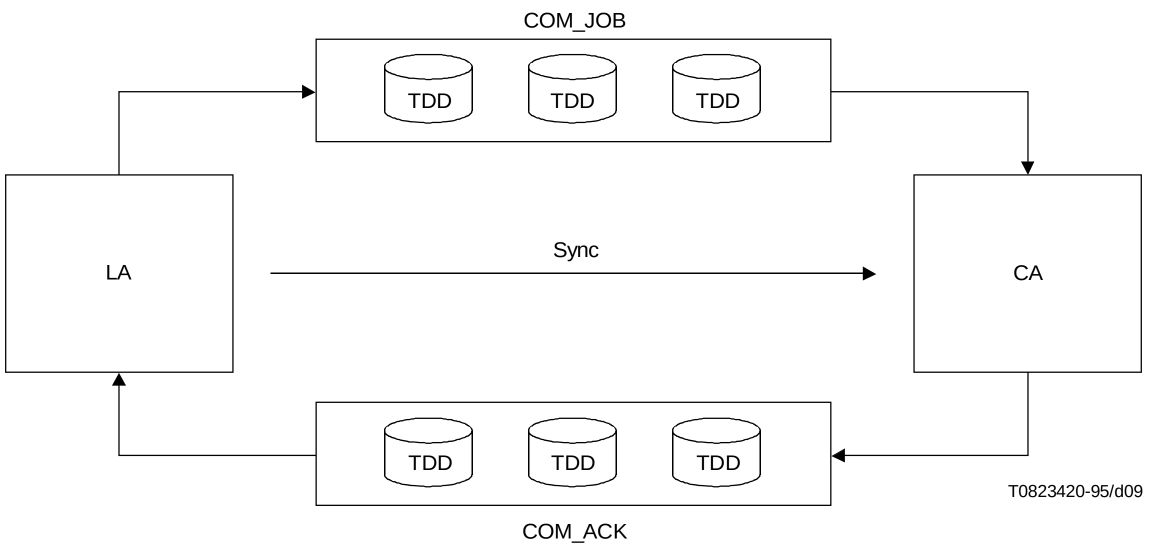


FIGURE 11/T.611

File Exchange Method–TDD Transfer

8 Transfer Formats

For proper operation, some requirements apply to the format of the Outgoing and Incoming Files exchanged between LA and CA. Incoming and Outgoing Files have a specific format which is not necessarily the same as the one used by the Transmission Files, e.g. a word processing file to be exchanged through the basic mode of the telefax service. In this case, format conversions are required (they shall be handled by the CA).

In order to exchange documents between LA and CA several Transfer Formats are defined by this Recommendation. The Transfer Formats apply to the Transfer Files defined in 2.1. These formats should not be confused with the format transmitted through the network (Transmission Formats) by the telecommunications services nor should they be confused with the format used by TTDs (TDD coding).

Several Transfer Formats are possible:

– text oriented Transfer Formats;

– graphic oriented Transfer Formats;

– the transparent Transfer Format;

– private Transfer Formats which are either text or graphic oriented.

This Recommendation defines:

– 3 text oriented Transfer Formats (APPLI/COM Extended ASCII, APPLI/COM Standard ASCII and the Teletex Format);

– 1 graphic oriented Transfer Format (APPLI/COM TIFF);

– the Transparent Format Transfer Format.

The Transparent Format may be used only if the document is to be transmitted as a binary file by the telecommunications service. In this case, no conversion of the contents of the file takes place thus the file will be transmitted unchanged.

This Recommendation is open for the support of private transfer formats. Hence other Transfer Formats (e.g. PostScript) may be implemented by CA manufacturers to adapt native formats of commonly used application programs (e.g word processors, data-bases or spreadsheets). To inform a LA about this support, the CA shall place the keyword ADDCONV into the CA-Descriptor of the ICE (see clause 9).

A CA supporting an additional Transfer Format has to ensure that documents are treated properly for all services supported by the CA. Restrictions imposed by the service itself have to be treated in a transparent way. Anyway, the Transfer Formats supported by CA manufacturers shall be described in their related documentation.

Table 49 indicates how Transfer Formats shall be supported by the CA depending on the operating system and the telecommunications service the CA offers.

TABLE 49/T.611

List of Transfer Formats

Transfer Format	Convert-id	Means
APPLI/COM Extended ASCII	ASCII437	ASCII Transfer Format as defined in 8.1.1. Use of this format is restricted to some services. See 8.4
	ASCII	ASCII Transfer Format as defined on the supporting CA. This Transfer Format may differ from the ASCII437 Transfer Format on those systems which do not support the same extended ASCII character set
APPLI/COM Standard ASCII	T.50	T.50 (IRA) Transfer Format as defined in 8.1.2. Use of this format is restricted to some services. See 8.4
Teletex Format	T.61	T.61 Transfer Format as defined in 8.2. Use of this format is restricted to some services. See 8.4
APPLI/COM TIFF	TIFF	TIFF Transfer Format as defined in 8.3. Use of this format is restricted to some services. See 8.4
Transparent Format	VOID	Stands for "no conversion to be done". Use is restricted for transparent transfer of document in various "file transfer" modes of the telecommunications services

The Transfer Formats defined in this Recommendation can be read and generated for the appropriate services under the appropriate operating systems by conforming CAs.

NOTE 1 – The text-oriented Transfer Formats for the telefax service are supported in the outgoing direction only.

Documents that are transferred to the CA in a text-oriented format shall be edited by the application in such a way that the format and character set correspond to the requirements of the service, i.e. set the "number of characters/line", "lines/page", "character pitch", "line spacing" and attributes such as "underline", "superscript" and "subscript".

The CA may reject documents with an incorrect format or character set.

NOTE 2 – If a LA wants to achieve service-independence the LA can use one of the "ASCII"-based formats for document transfer, since they are the only Transfer Formats that cover most services.

8.1 APPLI/COM Transfer Formats: Extended and Standard ASCII

The format codes shown in Table 50 are defined for the Transfer Formats APPLI/COM Extended ASCII and APPLI/COM Standard ASCII. Not all services permit all the format specifications listed (see 8.4). A CA is allowed to ignore ESC sequences. Furthermore, depending on the resident fonts used to perform ASCII to T.4 (or T.6) conversions, the CA is allowed to fold lines or rotate pages when required, except if this is disabled explicitly by the LA; in this case, if the CA may reject conversion or perform it downgraded.

TABLE 50/T.611

Format Effectors for APPLI/COM ASCII Transfer Formats

Format	Possible Values	HEX	ASCII	Default
Orientation	Portrait	1B 4F 30	ESC O 0	√
	Landscape	1B 4F 31	ESC O 1	
Pitch	10 Pitch	1B 50 30	ESC P 0	√
	12 Pitch	1B 50 31	ESC P 1	
	15 Pitch	1B 50 32	ESC P 2	
Line spacing	6 lines/inch (1 spacing)	1B 4C 30	ESC L 0	√
	4 lines/inch (1.5 spacing)	1B 4C 31	ESC L 1	
	3 lines/inch (2 spacing)	1B 4C 32	ESC L 2	
	12 lines/inch (0.5 spacing)	1B 4C 33	ESC L 3	
Attributes	Underline off	1B 55 30	ESC U 0	√
	Underline on	1B 55 31	ESC U 1	
	Superscript off	1B 41 30	ESC A 0	√
	Superscript on	1B 41 31	ESC A 1	
	Subscript off	1B 56 30	ESC V 0	√
	Subscript on	1B 56 31	ESC V 1	
	Boldface off	1B 42 30	ESC B 0	√
	Boldface on	1B 42 31	ESC B 1	
	Strike-out off	1B 53 30	ESC S 0	√
	Strike-out on	1B 53 31	ESC S 1	
	Italics off	1B 49 30	ESC I 0	√
	Italics on	1B 49 31	ESC I 1	
Text makeup	Fold lines disallowed	1B 54 30	ESC T 0	
	Fold lines allowed	1B 54 31	ESC T 1	√
	Rotate page disallowed	1B 52 30	ESC R 0	
	Rotate page allowed	1B 52 31	ESC R 1	√
	New line	0D 0A	CR LF	
	New page	0D 0C	CR FF	
Font selection	Select font number n	1B 43 'n'	ESC C 'n'	'n' = 0
NOTE – The fonts to be selected have to be declared in the ICE (see clause 9). The fonts are normally fixed width fonts, with a fixed character spacing. The conversion process shall ensure that characters shall neither overlap, nor be lost in the process. For large font sizes, the line spacing may increase and the formatting change. However, it is the CA responsibility to ensure that no contents is lost and text lines are legible.				

8.1.1 Character set of the APPLI/COM Extended ASCII Transfer Format

The characters supported by the APPLI/COM Extended ASCII Transfer Format for the Teletex, Telex and Telefax services are shown in Tables 51, 52 and 53[6]. This Transfer Format may be invoked in two ways:

– by specifying the parameter value "ASCII437" in the keyword "Convert" on those systems which implement the code page 437. These systems shall declare it in the ICE;

– by specifying the parameter value "ASCII" in the keyword "Convert" on those systems which implement the code page 437 as their native character set.

Otherwise, only the lower part of the table (i.e. octets from 20_{HEX} to $7E_{HEX}$) is guaranteed to be faithfully converted.

NOTE – The following tables shall be read by selecting first a column, then a row, i.e. character "A" is column 4, row 1.

8.1.2 Character set of the APPLI/COM Standard ASCII Transfer Format

The characters supported by the APPLI/COM Standard ASCII Transfer Format for the Teletex, Telex and Telefax services are shown in Tables 54, 55 and 56.

8.2 APPLI/COM Transfer Format: T.61

The APPLI/COM T.61 Transfer Format exactly corresponds to the ITU-T Recommendation T.61 for the Teletex service. Documents that the application transfers in this format are mapped to the Teletex service as it is. The format and codes shall be checked by the CA, but no conversion shall be performed. This format is suitable for applications that either already generate this format or are meant to transmit complex text layouts via the Teletex service.

8.3 APPLI/COM Transfer Format: TIFF

TIFF is a graphics oriented transfer format. The name TIFF stands for "Tagged Image File Format". The format of TIFF files includes the attributes that describe the image, such as resolution and dimensions, and are incorporated via tags in the TIFF file header. Because the information can be reached via tags, the program that generates TIFF files (TIFF writer) is not bound to a constant file structure, since a TIFF read program (TIFF reader) simply needs to know the algorithm necessary to locate the tags.

The TIFF Transfer Format offers the following features:

– it contains pixel information;

– it supports data-compression formats;

– it is independent of the hardware of the generating system (it works with any byte order);

– it is flexible due to its own tag structures.

Because of the fact that the number of possible tag combinations are quite high and not all tags defined are required for describing an image, several classes of TIFF formats have been formed during the course of TIFF’s development. For this reason, this Recommendation gives a definition of the profile taken as a basis for an interface-compatible TIFF file.

As a TIFF writer, a CA can generate 4 classes of files:

– TIFF standard format, here known as class 1. This is the default class for a CA TIFF writer.

– TIFF format with compression value 2, here known as class 2. Support of this format is an optional feature of the CA.

– TIFF format with ITU group 3 compression, here known as class 3. This format is supported by all CAs that serve the group 3 Telefax service.

– TIFF format with the ITU group 4 compression, here known as class 4. This format is supported by CAs that serve the group 4 Telefax service.

[6] The character set of the APPLI/COM Extended ASCII Transfer Format is a subset of the IBM-PC character set and is therefore highly suitable for implementations using these systems.

TABLE 51/T.611

Character Set of APPLI/COM Extended ASCII Transfer Format for the Telefax Service group 3 and 4 (corresponds to the ASCII 437 character set)

Hex	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
00			SP	0	@	P	‘	p	Ç	É	á	░	└	╨	α	≡
01			!	1	A	Q	a	q	ü	æ	í	▒	┴	╤	β	±
02			"	2	B	R	b	r	é	Æ	ó	▓	┬	╥	Γ	≥
03			#	3	C	S	c	s	â	ô	ú	│	├	╙	π	≤
04		¶	$	4	D	T	d	t	ä	ö	ñ	┤	─	╘	Σ	⌠
05		§	%	5	E	U	e	u	à	ò	Ñ	╡	┼	╒	σ	⌡
06			&	6	F	V	f	v	å	û	ª	╢	╞	╓	µ	÷
07			’	7	G	W	g	w	ç	ù	º	╖	╟	╫	τ	≈
08			(	8	H	X	h	x	ê	Ÿ	¿	╕	╚	╪	Φ	°
09			)	9	I	Y	i	y	ë	Ö	⌐	╣	╔	┘	Θ	•
0A			*	:	J	Z	j	z	è	Ü	¬	║	╩	┌	Ω	·
0B			+	;	K	[	k	{	ï	¢	½	╗	╦	█	δ	√
0C			,	<	L	\	l	|	î	£	¼	╝	╠	▄	∞	^{n}
0D			-	=	M	]	m	}	ì	¥	¡	╜	═	▌	∅	^{2}
0E			.	>	N	^	n	~	Ä	Pt	«	╛	╬	▐	∈	■
0F			/	?	O	_	o		Å	ƒ	»	┐	╧	▀	∩	

T0823430-95/d10

Conversion direction	Action performed
Outgoing	All characters shown in the table are accepted into the Teletex service character set
Incoming	The Transfer Format is not generated in this direction [except in such configurations where the CA is capable of OCR (Optional Character Recognition)]

NOTE – The “BACKSLASH” ($5C_{HEX}$) and “BRACES” ($7B_{HEX}$ and $7D_{HEX}$) characters, which are often used under DOS and OS/2, do not belong to the basic Teletex character set. Therefore, they must not be used in documents sent via this service.

TABLE 52/T.611

Character Set of APPLI/COM Extended ASCII Transfer Format for the Teletex service

Hex	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
00			SP	0	@	P	‘	p	Ç	É	á					
01			!	1	A	Q	a	q	ü	æ	í				β	±
02			"	2	B	R	b	r	é	Æ	ó					
03			#	3	C	S	c	s	â	ô	ú					
04		¶	$	4	D	T	d	t	ä	ö	ñ					
05		§	%	5	E	U	e	u	à	ò	Ñ					
06			&	6	F	V	f	v	å	û	ª				µ	÷
07			'	7	G	W	g	w	ç	ù	º					
08			(	8	H	X	h	x	ê	Ÿ	¿					°
09			)	9	I	Y	i	y	ë	Ö						•
0A			*	:	J	Z	j	z	è	Ü					Ω	·
0B			+	;	K	[	k		ï	¢	½					
0C			,	<	L		l	|	î	£	¼					
0D			-	=	M	]	m		ì	¥	¡					²
0E			.	>	N	^	n	~	Ä		«					
0F			/	?	O	_	o		Å		»					

T0823440-95/d11

Conversion direction	Action performed
Outgoing	All characters shown in the table are accepted into the Teletex service character set
Incoming	In the receive direction some characters can come from the service that are not contained in the table. These shall fall back onto characters that look similar to the original or by the character «?» ($3F_{HEX}$). The character FA_{HEX} is not generated in the receive direction anyway. The character $F9_{HEX}$ is generated instead

TABLE 53/T.611

Character Set of APPLI/COM Extended ASCII Transfer Format for the telex service

Hex	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
00			SP	0		P		p								
01				1	A	Q	a	q								
02				2	B	R	b	r								
03				3	C	S	c	s								
04				4	D	T	d	t								
05				5	E	U	e	u								
06				6	F	V	f	v								
07			’	7	G	W	g	w								
08			(	8	H	X	h	x								
09			)	9	I	Y	i	y								
0A				:	J	Z	j	z								
0B			+		K		k									
0C			,		L		l									
0D			-	=	M		m									
0E			.		N		n									
0F			/	?	O		o									

T0823450-95/d12

Conversion direction	Action performed
Outgoing	All characters shown in the table are accepted into the Telex service character set. Uppercase letters are interpreted as lowercase letters
Incoming	In the receive direction letters are always encoded in lowercase (Codes 61_{HEX} to $7A_{HEX}$). Uppercase letters are not generated

TABLE 54/T.611

Character Set of APPLI/COM Standard ASCII Transfer Format for the telefax service group 3 and 4

Hex	00	10	20	30	40	50	60	70
00			SP	0	@	P	‘	p
01			!	1	A	Q	a	q
02			"	2	B	R	b	r
03			#	3	C	S	c	s
04			$	4	D	T	d	t
05			%	5	E	U	e	u
06			&	6	F	V	f	v
07			’	7	G	W	g	w
08			(	8	H	X	h	x
09			)	9	I	Y	i	y
0A			*	:	J	Z	j	z
0B			+	;	K	[	k	{
0C			,	<	L	\	l	|
0D			-	=	M	]	m	}
0E			.	>	N	^	n	~
0F			/	?	O	_	o	

T0823460-95/d13

Conversion direction	Action performed
Outgoing	All characters shown in the table are accepted into the Telefax group 3 and group 4 service character set
Incoming	The Transfer Format is not generated in this direction

NOTE – The “BACKSLASH” ($5C_{HEX}$) and “BRACES” ($7B_{HEX}$ and $7D_{HEX}$) characters, which are often used under DOS and OS/2, do not belong to the Teletex character set. Therefore, they must not be used in documents sent via this service.

TABLE 55/T.611

Character set of the APPLI/COM Standard ASCII Transfer Format for the teletex service

Hex	00	10	20	30	40	50	60	70
00			SP	0	@	P	‘	p
01			!	1	A	Q	a	q
02			"	2	B	R	b	r
03			#	3	C	S	c	s
04			$	4	D	T	d	t
05			%	5	E	U	e	u
06			&	6	F	V	f	v
07			’	7	G	W	g	w
08			(	8	H	X	h	x
09			)	9	I	Y	i	y
0A			*	:	J	Z	j	z
0B			+	;	K	[	k	
0C			,	<	L		l	|
0D			-	=	M	]	m	
0E			.	>	N	^	n	~
0F			/	?	O	_	o	

T0823470-95/d14

Conversion direction	Action performed
Outgoing	All characters shown in the table are accepted into the Teletex service character set
Incoming	In the receive direction, characters can come that are not contained in the above table. Such characters may be replaced by the character “?” ($3F_{HEX}$)

TABLE 56/T.611

Character set of APPLI/COM Standard ASCII Transfer Format for the Telex service

Hex	00	10	20	30	40	50	60	70
00			SP	0		P		p
01				1	A	Q	a	q
02				2	B	R	b	r
03				3	C	S	c	s
04				4	D	T	d	t
05				5	E	U	e	u
06				6	F	V	f	v
07			’	7	G	W	g	w
08			(	8	H	X	h	x
09			)	9	I	Y	i	y
0A				:	J	Z	j	z
0B			+		K		k	
0C			,		L		l	
0D			-	=	M		m	
0E			.		N		n	
0F			/	?	O		o	

T0823480-95/d15

Conversion direction	Action performed
Outgoing	All characters shown in the table are accepted into the Telex service character set. Uppercase letters are interpreted as lowercase letters
Incoming	In the receive direction letters are always encoded in lowercase (Codes 61_{HEX} to $7A_{HEX}$). Uppercase letters are not generated

8.3.1 APPLI/COM TIFF Profile

The tags recognized by a CA and their handling are summarized in Table 57.

TABLE 57/T.611

APPLI/COM TIFF Profile table

Tag		TIFF-Reader		TIFF-Writer			
HEX	Name	Accepted value (dec)	Default value (dec)	Class 1 value (dec)	Class 2 value (dec)	Class 3 value (dec)	Class 4 value (dec)
FE	NewSubfileType (Note 1)	"0", "2"	"0"	"0"	"0"	"0"	"0"
FF	SubfileType (Note 1)	"0", "2"	"0"	"0"	"0"	"0"	"0"
100	ImageWidth	’Width’ (Note 2)	Reject	’Width’	’Width’	’Width’	’Width’
101	ImageLength	’Length’ (Note 3)	Reject	’Length’	’Length’	’Length’	’Length’
102	BitsPerSample	"1"	"1"	"1"	"1"	"1"	"1"
103	Compression	"1", "2", "3", "4", "32773" (Note 4)	"1"	"1"	"2"	"3"	"4"
106	Photometric-Interpretation	"0", "1"	"0"	"0"	"0"	"0"	"0"
107	Thresholding	Ignore		None			
108	CellWidth	Ignore		None			
109	CellLength	Ignore		None			
10A	FillOrder	"1", "2" (Note 5)	"1"	"1"	"1"	"2"	"2"
10D	DocumentName	Ignore		None			
10E	ImageDescription	Ignore		None			
10F	Make	Ignore		None			
110	Model	Ignore		None			
111	StripOffsets	’Offset’	Reject	’Offset’	’Offset’	’Offset’	’Offset’
112	Orientation	"1"	"1"	"1"	"1"	"1"	"1"
115	SamplesPerPixel	"1"	"1"	"1"	"1"	"1"	"1"
116	RowsPerStrip	’Rows’	Reject	’Rows’	’Rows’	’Rows’	’Rows’
117	StripByteCounts	’Count’	Reject	’Count’	’Count’	’Count’	’Count’
118	MinSampleValue	"0"	"0"	"0"	"0"	"0"	"0"
119	MaxSampleValue	"1"	"1"	"1"	"1"	"1"	"1"
11A	XResolution	"300" (Note 6)	"300"	"300"	"300"	’X-res’	’X-res’
11B	YResolution	"300" (Note 6)	"300"	"300"	"300"	’Y-res’	’Y-res’

TABLE 57/T.611 (*end*)

APPLI/COM TIFF Profile table

Tag		TIFF-Reader		TIFF-Writer			
HEX	Name	Accepted value (dec)	Default value (dec)	Class 1 value (dec)	Class 2 value (dec)	Class 3 value (dec)	Class 4 value (dec)
11C	PlanarConfiguration	"1"	"1"	"1"	"1"	"1"	"1"
11D	PageName	Ignore		None			
11E	XPosition	Ignore		None			
11F	YPosition	Ignore		None			
120	FreeOffsets	Ignore		None			
121	FreeByteCount	Ignore		None			
122	GrayResponseUnit	Ignore		None			
123	GrayResponseCurve	Ignore		None			
124	Group3Options	"0", "4" (Note 7)	"0"	"0"	"0"	"4"	"4"
125	Group4Options	"0", "4" (Note 7)	"0"	"0"	"0"	"4"	"0"
128	ResolutionUnit	"2"	"2"	"2"	"2"	"2"	"2"
129	PageNumber	’Page’	Reject	’Page’	’Page’	’Page’	’Page’
12C	ColorResponseUnit	Ignore		None			
12D	ColorResponseCurve	Ignore		None			

NOTES

1 Both NewSubfileType and SubfileType tags should be accepted by the readers. The writer should generate the NewSubfileType tag only because it has superseded the SubfileType tag.

2 The following applies: ImageWidth//XResolution <= 215 mm. If the quotient is exceeded, the document can be rejected. Hence, if the XResolution is 204 dpi (G3 fax standard), the value 1728 for ImageWidth is not exceeded.

3 ImageLength is processed if the following applies to the quotient ImageLength/YResolution < 297 mm. This corresponds to the length of a DIN A4 sheet of paper. The guarantee applies up to a resolution of 300 dpi. The processing of higher ImageLength values is an optional feature of the CA.

4 Support of Compression values:

- 1: (uncompressed) is guaranteed only if the resolution value (XResolution, YResolution) is 300 dpi or exactly corresponds to that of the fax service selected.
- 2: is an optional feature of the CA.
- 3 or 4: ITU group 3/4 compression is guaranteed only if the resolution value (XResolution, YResolution) exactly corresponds to the resolutions of the Telefax services selected.
- 32773: (pack-bits compression) is an optional feature of the CA.

5 The FillOrder value 2 (= 'reverse bit-order') is allowed only if the Compression value is 3 or 4.

6 For Compression values 3 and 4 the actual resolution value (XResolution, YResolution) in dpi that exactly corresponds to that of the Telefax service selected shall be specified. The default value of the reader does not apply since the X and Y resolution shall be applied. The LA is responsible for local conversion of received resolutions.

7 If the Compression value is 3, the value of the Group3Options is 4; otherwise, it is 0. Other combinations shall be rejected.

Table 58 summarizes the rules applying to the TIFF Profile table.

TABLE 58/T.611

Rules for TIFF Profile table

Column	Rule
Tag	Identifies the name of the tag and its value in hexadecimal notation. If a tag not listed here is encountered by a TIFF reader, the reader may reject the TIFF file
Reader – Accepted value	Summarizes the accepted values. Only the values listed are guaranteed to be accepted by a CA. Other values may be rejected by a CA and thus shall be avoided by a generating LA. If a value states "ignore", the tag is ignored by the reader
Reader – Default value	Shows the default value applied if the tag is not specified. If a value states "reject", the TIFF file shall be rejected if the tag is absent. If a value states "ignore", no default is applicable, since the tag is ignored by the reader
Writer	The writer columns contain the values generated by the appropriate CA TIFF writer class. If a value states "none", no tag and value is generated in this class

8.4 Service Constraints applying to Transfer Formats

Due to the nature of the telecommunications services selected, several constraints are applied to the transfer formats. These constraints are summarized in Table 59.

TABLE 59/T.611

Transfer Format Constraints

Telecommunications Service	Allowed Transfer Format	Constraints
Telex	ASCII, T.50	No format specifications but the text makeup is allowed. All other specifications shall be ignored. There are also constraints in the character set
Teletex	ASCII, T.50, T.61	Only the Teletex specific format specifications are allowed. All other specifications shall be either ignored or the file shall be rejected. There are also constraints in the character set
	VOID	Only allowed in conjunction with a TFT (Telematic File Transfer) type selection
Telefax (G3/G4)	ASCII, T.50[a)]	All formats and attributes are allowed. If a CA does not support a specific attribute or format, it is allowed to ignore it
	TIFF	Only allowed if none of the TFT (Telematic File Transfer) types was selected
	VOID	Only allowed in conjunction with a TFT (Telematic File Transfer) type selection
[a)] Only applicable in outgoing direction.		

9 ICE

Before a LA can start interacting with a CA, the LA needs to know which CA it requires (based on criteria like the type of telecommunication services supported) and how to "talk" to that CA. Since the interface described in this Recommendation aims at multiple platforms, the mechanism prescribed to obtain information from CAs prior to interacting with them is based on reading configuration information. This configuration information is named the Interface Configuration Environment (ICE).

A CA provider shall supply this configuration information, so that LAs can get information about a specific CA and the access to that CA.

The part of the ICE which gives information about a specific CA is called the CA-Descriptor. Depending on whether the system is a standalone system or utilizes a Local Area Network (LAN), information about the reachable CAs can be found in different places. Since multiple CAs can be reachable by a LA, there is a need for centralizing the locations of the individual CA-Descriptors in a "Master ICE" file. The location of the Master ICE file itself is defined on a per-platform basis (see Annex B).

Therefore, on a given system, a LA shall first read the Master ICE file to discover which CAs it can reach and the location of those CAs' CA-Descriptors. Then the LA reads the CA-Descriptor of the CA it wants to utilize. Finally, the LA logs in to that CA by using the information found in that CA's CA-Descriptor.

To illustrate these concepts, three typical configurations can be considered:

a) a standalone system supporting multiple LAs and multiple CAs;

b) a LAN-based system where LAs and CAs are located on different machines, and a file server is present;

c) a LAN-based system where LAs and CAs are located on different machines, and no file server is present;

In case a), the Master ICE is located in a specific file path (the precise location depends on the platform, see Annex B), and the Master ICE lists all the CAs on that system with the location of their respective CA-Descriptor (see Figure 12).

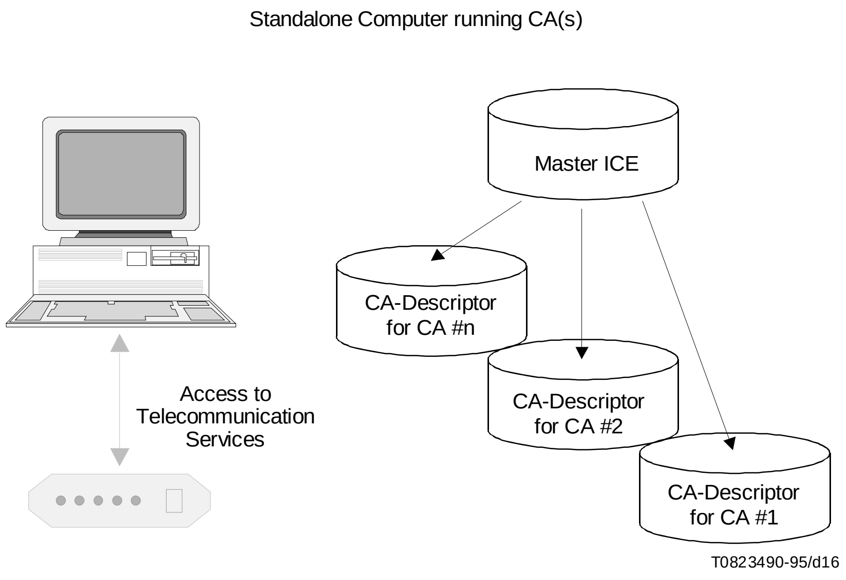


FIGURE 12/T.611

Standalone System supporting multiple CAs

In case b), the Master ICE is located on the file server, in a file path that all workstations can reach. The Master ICE lists all the CAs on the LAN and gives the absolute paths of their respective CA-Descriptor (which can be located on other machines than the file server itself) (see Figure 13).

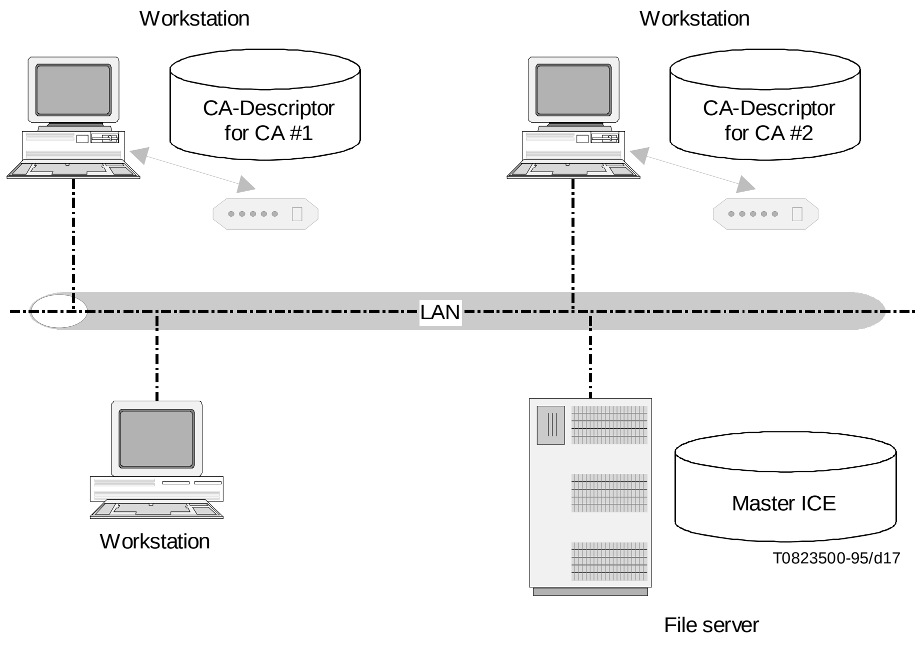


FIGURE 13/T.611

LAN based System with File service

In case c), since no file server can centralize the Master ICE file for the whole system, a Master ICE file is located on each machine that supports one or more CAs, at a fixed location as in case a). The location of the CA-Descriptors is recorded in the Master ICE with paths relative to that machine (see Figure 14).

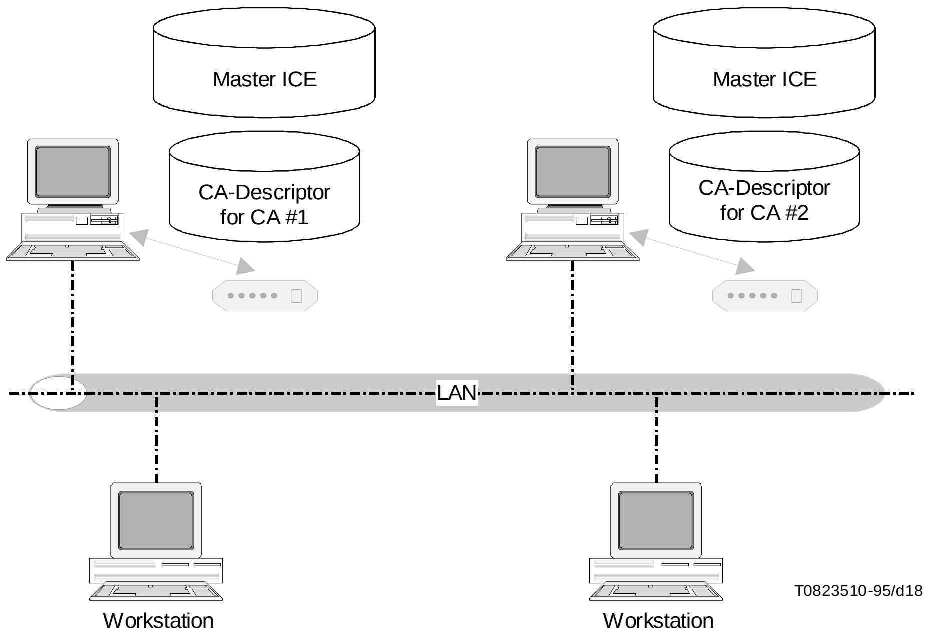


FIGURE 14/T.611

Peer to peer LAN system

9.1 Presentation of the ICE

The ICE is presented using readable text coding, as it is described for TDDs in 6.4.1. Except on those systems where the native coding of characters is EBCDIC (Code-ID = "E"), IRA coding is used (Code-ID = "I"). The ICE – Master ICE and CA-Descriptors – shall be presented in form of lines using the IRA character repertoire and shall be formatted as described for TDDs in 6.4.1.

9.2 Gaining access to the ICE Information

Access to the Master ICE shall be accomplished by means of a "file access method" which is common to most operating systems. In turn, all of the available access methods to obtain CA-Descriptor information for an individual CA shall be defined within the Master ICE. The access methods may be "file-based" or may take advantage of more dynamic methods which may be available within a particular operating environment (e.g DLLs for Windows and executable files or device drivers for DOS).

9.3 Master ICE

The Master ICE file lists all the CAs that a LA can reach. It contains information about the CA identifier, the supported communication services, the type of exchange mechanism supported and the location of the CA's CA-Descriptor where further information can be obtained.

9.3.1 Formal Description

This subclause describes the syntax of the Master ICE. The first syntax element in the Master ICE is the <Master ICE Header>. No other element, including SPACE and TABULATION format effectors are allowed before.The detailed syntax, described in BNF-based grammar is shown below (for a description of the BNF-based grammar see A.1):

```
<Master ICE>  :=              <Master ICE Header> <CA Entries>

<Master ICE Header>  :=       "I*APPLI/COM*" <Version> "*ITU-T*MASTER_ICE"
                              -- for definition of <ICE Header> see also 6.3.2.3

<Version>  :=                 "1994"
                              -- <Version> denotes year of approval of this Recommendation

<CA Entries>  :=              "#" <CA Entry> {"#" <CA Entry>}

<CA Entry>  :=                <CA Identifier> <CA Product Info> <CA EM>
                              <CA Service> {<CA Service>}
                              <CA ICE Location> {<CA ICE Location>}

<CA Identifier>  :=           "CA-ID" ":" NUMERIC-STRING
                              -- identifies the CA

<CA Product Info>  :=         "APPLI/COM" ":" STRING(SIZE(1..255))
                              -- Information about the product, CA manufacturer-provided

<CA EM>  :=                   "EM" ":" "FILE" | "PRIMITIVE"
                              -- exchange mechanism supported by the CA
                              -- refer to 7.3

<CA Service>  :=              "SERVICE" ":" <Service-id-parameter>
                              -- telecommunication services supported by the CA
                              -- refer to 6.4.5.1 and 10.4

<CA ICE Location>  :=         "ACCESS" ":" PATH
                              -- points to the location of the CA-Descriptor file or driver
```

9.3.2 Configuration

The Master ICE may be configured either manually by means of an "editor" software during the installation of a particular CA or dynamically by means of an appropriate "driver" software provided by the CA manufacturer[7)].

The ICE shall be configured so that it takes into account all possible communications applications accessible from within a system[8)].

9.4 CA-Descriptor

The CA-Descriptor contains the information relevant to a given CA. It is composed of sections that group logically related entries.

9.4.1 Formal Description

This subclause describes the syntax of the CA-Descriptor. The first syntax element in the CA-Descriptor is the <CA-Descriptor Header>. No other element, including SPACE and TABULATION format effectors are allowed before. The detailed syntax, described in BNF-based grammar is shown below:

<CA-Descriptor> :=	<CA-Descriptor Header> <CA-Descriptor-component>
<CA-Descriptor Header> :=	"I*APPLI/COM*" <Version> "*ITU-T*ICE" -- *for definition of <CA-Descriptor Header> see also 6.3.2.3*
<Version> :=	"1994" -- *<Version> denotes year of approval of this Recommendation*
<CA-Descriptor-component> :=	"#" <Section> {<Section>}
<Section> :=	<Section-Parameter> {<Section-Parameter>}
<Section-Parameter> :=	<Parameter-Name> ":" <Parameter-Value>
<Parameter-Name> :=	STRING(SIZE(1..16))
<Parameter-Value> :=	STRING(SIZE(1..255))

9.4.2 Configuration

The CA-Descriptor information shall be provided by the CA-Manufacturer in the form of a file or via a dynamic means such as a device driver, executable file or DLL. The access method(s) for the CA-Descriptor shall be defined in the Master ICE as part of the installation procedure for each CA. The CA-Descriptor may be configured either manually by means of an "editor" software during the installation of a particular CA or dynamically by means of an appropriate "driver" software provided by the CA manufacturer. However, a LA relying on the information from a CA-Descriptor regarding a particular CA shall not assume that the CA is active at that very moment (loaded and running). To be certain that a particular CA is really active, the LA shall login to that CA. Only when login succeeds shall the CA be considered active. For information about the login procedure refer to 7.1 and 7.2.1.

9.5 CA-Descriptor Components

Tables 60, 61 and 62 describe the relevant keyword and the corresponding parameter values of the CA-Descriptor. Subclause 6.4.4 gives complementary information about the coding of the parameter values.

[7)] Since access to files and to operating system device drivers is similar on popular operating systems, the Master ICE itself may be implemented statically as a file or dynamically in form of an operating system device driver.

[8)] It is not necessary for a CA to run within the same physical equipment as the LA does. The CA must only provide access by means of a device driver or similar.

9.5.1 Parameters of the GENERAL Section

See Table 60.

TABLE 60/T.611

CA-Descriptor information items independent of operating system or exchange mechanism

Keyword[a)]	Parameter	Interpretation
ALIAS	"yes" | "no"	Specifies whether the CA supports alias names (user friendly names)
CODEPAGE*	STRING	Specifies the additional code pages for the extended ASCII character sets the CA supports. String indicates the number of the code page (e.g. "850")
CODING*	Code-ID	Specifies which TDD encoding scheme is supported by the CA. See 63 for the supported values that can be specified. If the Code-ID "E" has to appear in the ICE, its binary value shall be set in accordance to the code presentation chosen for the ICE itself, i.e. if the Code-id of the ICE Header is "I" (APPLI/COM Standard ASCII) then the binary value of the "E" shall be coded as 45_{HEX}
CONVCHK*	<Convert-id-parameter>	Declares which transfer formats are supported in the Submit TDD function CONVERT and/or CHECK. For the syntax of the <Convert-id-parameter> see also 6.4.5.3
COUNTRY	STRING	Specifies the country for which the CA is configured. Used to register country specific features like gaining access to the conversion facility or accessing the black list of dialling numbers. The value to be placed in the parameter is to be taken from ITU-T Recommendation T.35. It has to be presented as a decimal counted, numeric string, i.e. "154" for the Seychelles
DRF	"yes" | "no"	Boolean-parameter; states whether the CA supports the "Dispatch Received Files" facility
EXTEND*	<keyword>	Provides the possibility for extensions to the Recommendation. Can only be implemented as formal changes to the Recommendation. All the CA-supported keywords shall be listed
FC	"A" | "B"	States which Functional Class the CA supports
FONT0	NUMERIC-STRING	A digit in the range "1" .. "9" to specify the number of declared fonts. Fonts are described starting from identifier "Font1" up to "Font 9" (see below)
FONTx*	STRING "," NUMERIC-STRING	The name of the font followed by the pitch (in characters per inch). The "x" value ranges from 1 to 9
NATIONAL*	<keyword>	Provides the possibility for national extensions to the Recommendation. Can only be implemented with the approval of national Administrations. (All supported keywords shall be listed)
PRINT*	<Printer-id-parameter>	Declares which printers could be addressed by the CA in the Submit TDD function PRINT. For the syntax of the <Printer-id-parameter> see also 6.4.4.10
PRIVATE*	<keyword>	Provides the possibility for private extensions to the Recommendation. (All supported keywords shall be listed)
RECORD*	<keyword> "," NUMERIC-STRING	Gives the complete list of CA-Record field names supported by the CA, in the order they are found in the file resulting from the Trace:COPY function. The CA shall state the keyword followed by – and separated by comma – the length the field will have in the resulting file. See also Table 23
SUBMIT*	"PRINT" | "CONVERT" | "CHECK"	Declares which functions are supported in the Submit TDD function. This keyword shall be repeated as many times as required

[a)] A "*" (star) at the end of a keyword indicates that this keyword may be repeated.

9.5.2 Parameters of the EM Section

See Table 61.

TABLE 61/T.611

CA-Descriptor information items applying for the file exchange mechanism

Keyword	Parameter	Interpretation
EM	"file" | "primitive"	Exchange Method used to interchange TDDs between LAs and CAs. "file" and "primitive" are the supported values (see 7.3 for further details)
SYNC	"yes" | "no"	Indicates whether the CA is "sync-driven". See section 7.3 for further details
F_JOB_Q	PATH	Specifies the path of the TDD request files. See 7.3 for further details
F_ACK_Q	PATH	Specifies the path of the TDD response files. See 7.3 for further details
ERROR_Q	PATH	Specifies the path of the TDD response files relating to errors. See 7.3 for further details
ALARM	"yes" | "no"	This Boolean-parameter states whether the CA supports the SetAlarm function

9.5.3 Parameters of the OPERATING SYSTEM Section

See Table 62.

9.5.4 Parameters of the SERVICE Section

See Table 63.

More service specific information items are described in the related clauses of Part II of this Recommendation.

10 Functional Classes and Profiles

10.1 Functional Class A

Functional Class A (FCA) specifies a set of Transfer Formats and specifies the interactions between LAs and CAs in order to send and receive documents.

FCA requires the support of:

– the <SendTDD> and the <SendAckTDD> that allows a LA to send a document through a CA;

– the <ReceiveTDD> that allows a LA to retrieve documents received by a CA.

In addition, the LA may select various CA-offered options like the use of additional keywords ("ADDKEYS") or the provision of the submit functionality ("SUBMIT").

10.2 Functional Class B

In addition to FCA features, Functional Class B (FCB) provides the complete Trace functionality.

FCB is a superset of FCA and gives further integration of communications functions to the users’ applications.

TABLE 62/T.611

CA-Descriptor information items applying to the operating system

Keyword[a)]	Parameter	Interpretation
ENVIRON*	"MSDOS" | "WINDOWS" | "UNIX" | "OS2" | "MacOS"	This keyword specifies the operating environment of the CA. If a CA supports several environments, the ICE shall contain as many ENVIRON keyword instances as the number of different operating systems supported
DRIVER	PATH	Name of the driver that must be opened to initiate dialogues with the CA. See Annex E for further details
INT	HEX,HEX	Indicates the interrupt number. Two hexadecimal numbers; the first specifies the multiplex number, the second the program code number. If the interrupt is not multiplexed, then the second hex number shall not be specified
LIB	"yes" | "no"	This Boolean-parameter states whether the CA is a static library (LA must be linked to it)
LIB-NAME*	PATH	Path(s) of the library(ies) (used in conjunction with the LIB keyword)
DLL	"yes" | "no"	Dynamic Link Library. See Annex E. The ‘DLL-NAME’ keyword shall be supported only if the DLL exchange mechanism is supported
DLL-NAME*	PATH	Path(s) of the DLL file(s) (used in conjunction with the DLL keyword)
DDE	"yes" | "no"	Dynamic Data Exchange mechanism. In the WINDOWS environment if the application supports the DDE exchange mechanism, it shall specify "yes". See Annex E. The next three keywords shall be included in the ICE if the DDE mechanism is used
WIN-APP	STRING	Application Name (MsDos format) XXXXXXXX.XXX
SUBJECT*	STRING	All CA "Subjects" shall be mentioned (if any) otherwise leave empty (to be used with the DDE keyword)
ITEM*	STRING	All CA "Items" shall be mentioned (if any) otherwise leave empty (to be used with the DDE keyword)
a) A "*" (star) at the end of a keyword indicates that this keyword may be repeated.		

TABLE 63/T.611

CA-Descriptor information items applying to the service section

Keyword[a)]	Parameter	Interpretation
ADDCONV*	STRING (SIZE (1..8))	Lists additional Transfer Formats supplied by the CA. The names listed here shall be used to select the Transfer Formats at appropriate places
ADDKEYS*	<keyword>	Lists all the additional keywords supported by the CA. Only keywords classified as "+" in the TDD tables of clause 6 may be specified here
a) A "*" (star) at the end of a keyword indicates that this keyword may be repeated.		

10.3 Additional Functions

The Submit and Extend functionalities are additional facilities that should be declared in the ICE by the CA (see also 9.5).

10.4 Service Profiles

Support of all the features of a given telecommunication service may be impossible, due to the large number of functions to implement. On the other hand, offering too many options for the support of a given service leads to compliant but incompatible implementations.

To help reduce incompatibilities between different vendors' applications, this Recommendation supports the concept of Service Profiles. A Service Profile groups a defined set of service features that must be supported by the implementations claiming compatibility to this Recommendation.

The set of features to be supported for a given Service Profile is defined on a per-service basis.

An implementation may thus claim compatibility to a Service Profile and support additional features. It is therefore garanteed that two implementations (i.e. a CA and a LA) supporting the same Service Profile will be able to offer all the features mentioned in the Service Profile, and optionally offer additional features not mandatory for that Service Profile.

CAs shall declare in the Master ICE the Service Profiles supported (see 9.5).

PART II – SERVICE DEPENDENCIES

11 Service: Telefax Group 3

Besides the telefax service specific keywords for <SendTDD>, <SendAckTDD> and <ReceiveTDD>, there is one additional function defined: Poll - to poll a remote fax station for a document to be retrieved. This function shall be implemented using the <ExtendTDD>. The extension is called <PollExtension>

Together with the <PollExtension> also the <SendTDD> and the SendAckTDD> may be extended to perform a Poll operation while sending a document.

The extensions are described at appropriate places within this clause.

11.1 Service Specific Syntax Elements

```
<ServiceDependentKeywordsSend>  :=
                                ((<Recipient> [<PollSendExtension>]) | <RecipientSpec>)
                                ((<Document> <Convert> [<Type>] [From] [To]) | <DocumentSpec>)
                                [<SubAddress>] [<G3Speed>] [<GenCil>] [<HighRes>] [<UseEcm>]

<ServiceDependentKeywordsSendack>  :=
                                <Recipient> [<PollSendExtension>]
                                ((<Document> <Convert> [<Type>] [From] [To]) | <DocumentSpec>)
                                [<SubAddress>] [<G3Speed>] [<GenCil>] [<HighRes>] [<UseEcm>]

<ServiceDependentKeywordsReceive>  :=
                                [<Originator>]
                                ((<Document> <Convert> <Type>) | <DocumentSpec>)
                                [SubAddress] [<CvFax3>] [<G3Speed>]

<ExtendSubFunctionKeywords>  :=
                                <PollExtension>

<PollSendExtension>  :=         <DoPoll> <PollPassword> [<PollSelector>]

<PollExtension>  :=             <PollSubFunction> <Recipient> <PollPassword> [<PollSelector>] [<SendTime>]
                                [<ComId>] [<Minor>] [<Warning>]
```

See Table 64.

11.2 Text Based Encoding

11.2.1 Mapping of Keywords

See Table 65.

11.2.2 Encoding of Parameters

See also 6.4.4 for the encoding of non service dependent parameters used.

11.2.2.1 Service-id-parameter

The Service-id parameter is encoded as STRING set to the constant value "FX3".

Syntax:

```
<Service-id-parameter>  :=      "FX3"
```

TABLE 64/T.611

Additional syntax elements for Telefax Group 3

Syntax element	Purpose
<Convert>	Specifies the Transfer Format to be used
<CvFax3>	Specifies the desired Transfer Format for received files
<Document>, <DocumentSpec>	Specifies the document or the documents to be sent or being received
<DoPoll>	Specifies whether recipient shall be polled in a <SendTDD> or <SendAckTDD>
<G3Speed>	Specifies the modulation speed desired (input) or that was used (output)
<GenCil>	Specifies the insertion of a CIL when generating the outgoing fax
<HighRes>	Indicates the resolution
<Minor>	This syntax element returns an additional error code in the Response-TDD
<Originator>	Specifies the communications address of the originator
<PollPassword>	Specifies the password for polling remote device
<PollSelector>	Gives a handle to the recipient in order to poll for a specific document
<PollSubFunction>	Specifies the subfunction of the Poll <ExtendTDD>
<Recipient>, <RecipientSpec>	Specifies the communications address(es) of the recipient(s)
<SubAddress>	Specifies the Originator's sub-address
<To>	Transmission should finish on page number specified
<Type>	Specifies the Transmission Format used
<UseEcm>	ECM is desired (input) or was activated (output)
<Warning>	This syntax element returns an additional warning code in the Response-TDD

11.2.2.2 Type-id-parameter

The Type-id-parameter is encoded as STRING set to one of the following values:

"STD"	Basic Telefax G3 Service (MH)
"BTM"	Telematic File Transfer (TFT) of Telefax G3 Service: Basic Transparent Mode
"DTM"	Telematic File Transfer (TFT) of Telefax G3 Service: Document Transparent Mode
"EDI"	Telematic File Transfer (TFT) of Telefax G3 Service: Edifact
"BFT"	Telefax G3 Service: Binary File Transfer

NOTE – The Transmission Format STD (=Modified Hufman Code, MH) is defined per ITU-T Recommendation T.4, the TFT formats BTM, DTM and EDI are defined per ITU-T Recommendation T.571 and T.30 (Annexes) and the Transmission Format BFT is defined per ITU-T Recommendation T.434.

Syntax:

<Type-id-parameter> := "STD" | "BTM" | "DTM" | "BFT" | "EDI"

TABLE 65/T.611

Text Based Encoding of additional syntax elements for Telefax Group 3
(A ⏎ denotes the new line format effector)

Syntax Element	Keyword/Parameter Pair
<Convert>	"CONVERT" ":" <Convert-id-parameter> ⏎
<CvFax3>	"CVFAX3" ":" <Convert-id-parameter> ⏎
<Document>	"FILENAME" ":" <File-parameter> ⏎
<DocumentSpec>	"FILENAME" ":" "@" <File-of-filespec> ⏎
<DoPoll>	"DOPOLL" ":" <Boolean-parameter> ⏎
<From>	"FROM" ":" NUMERIC-STRING ⏎
<G3Speed>	"G3SPEED" ":" <G3-speed-parameter> ⏎
<GenCil>	"GENCIL" ":" <Boolean-parameter> ⏎
<HighRes>	"HIGHRES" ":" <Resolution-parameter> ⏎
<Minor>	"MINOR" ":" <Error-parameter> ⏎
<Originator>	"ADDRESS" ":" <Address-parameter> ⏎
<PollPassword>	"PASSWORD" ":" <Password-parameter> ⏎
<PollSelector>	"POLLSELECT" ":" <Poll-select-parameter> ⏎
<PollSubFunction>	"SUBFUNC" ":" "Poll" ⏎
<Recipient>	"ADDRESS" ":" <Address-parameter> ⏎
<RecipientSpec>	"ADDRESS" ":" "@" <File-of-addrspec> ⏎
<SubAddress>	"SUBADDR" ":" <Sub-address-parameter> ⏎
<To>	"TO" ":" NUMERIC-STRING ⏎
<Type>	"TYPE" ":" <Type-id-parameter> ⏎
<UseEcm>	"USEECM" ":" <Boolean-parameter> ⏎
<Warning>	"WARNING" ":" <Error-parameter> ⏎

11.2.2.3 Convert-id-parameter

The Convert-id-parameter is encoded as STRING set to one of the following values:

"ASCII"	APPLI/COM Extended ASCII (For outgoing files only)
"ASCII437"	APPLI/COM Extended ASCII (For outgoing files only)
"T.50"	APPLI/COM Standard ASCII (For outgoing files only)
"TIFF"	TIFF Transfer Format as defined in 8.3
"TIFF2"	(For incoming files only). APPLI/COM TIFF Class 2 as specified in 8.3
"TIFF3"	(For incoming files only). APPLI/COM TIFF Class 3 as specified in 8.3
"VOID"	No conversion to be done

NOTE – If a LA requests a Telefax document that has been received, it is possible to:

– obtain the document in APPLI/COM TIFF class 2 by specifying TIFF2;

– obtain the document in APPLI/COM TIFF class 3 by specifying TIFF3,

provided the CA is capable of generating that special TIFF format. If only TIFF is specified, the document will be delivered in the APPLI/COM TIFF default class (class 1). However, in the send direction it is sufficient to specify TIFF only (without number extension), since the compression information is contained in the Transfer Format itself.

The use of the Convert-id-parameter also depends on the Type-id selected. Table 66 depicts this.

TABLE 66/T.611

Permitted Convert-id assignements dependent on Type-id and traffic direction

Type-id	Convert-id for outgoing traffic	Convert-id for incoming traffic
STD	ASCII, ASCIIxxx[a)], T.50, TIFF	TIFF, TIFFx[b)]
BTM, DTM, BFT, EDI	VOID	

a) xxx stands for a code-page, declared in the ICE, e.g. ASCII437 if code-page 437 has been declared.

b) x stands for the TIFF class to be read, which is a value between 2.3.

Syntax:

<Convert-id-parameter> :=	<Convert-std> | <Convert-bin>
<Convert-std> :=	<Convert-std-in> | <Convert-std-out>
<Convert-std-in> :=	"TIFF" | "TIFF2" | "TIFF3" -- *incoming documents*
<Convert-std-out> :=	<Ascii> | "T.50" | "TIFF" -- *outgoing documents*
<Ascii> :=	"ASCII" | STRING ("ASCII" + <Code-page>)
<Code-page> :=	<digit> <digit> <digit>
<digit> :=	"0" | ... | "9"
<Convert-bin> :=	"VOID"

11.2.2.4 File-of-addrspec and Address-parameter

The File-of-addrspec parameter is encoded as PATH which points to a file containing Addrspec-parameters, which contain the Address-parameter.

The Address-parameter is encoded as STRING. The STRING shall contain the telephone number. If the phone number starts with a "!" then it may contain special characters that are treated as operators (or modifiers) rather than dial digits, as shown in Table 67 below.

The telephone number may be terminated by a "#" character (number sign) and may be followed by a subaddress (subaddress of the remote equipment). The encoding of the subaddress is defined in 11.2.2.7.

Alternatively an alias name may be given instead of the phone number, provided the alias is introduced with a "&" character. It is assumed that the CA knows how to decode the alias specified. This is typically used when the CA implements a phone book.

TABLE 67/T.611

Dial string modifiers for the PSTN

Input	Type	Explanation
"0" | ... | "9" | "*" | "#"	Dial digit	Dial without interpretation
"!"	Modifier	First character in the dialling sequence string: enter the "raw dialling mode"
";"	Modifier	Pause in the dialling process. Duration is CA dependent. Note that the ";" must be escaped by the "\" character, otherwise the CA may understand it as a comment introducer
":"	Modifier	Same as ";" but does not need to be escaped
,	Modifier	Pause dialling for 2 seconds
"T" | "t"	Modifier	Enforce "tone" dialling for subsequent digits
"P" | "p"	Modifier	Enforce "pulse" dialling for subsequent digits
"W" | "w"	Modifier	CA waits and listens for a 3-seconds, continuous dial tone
"@"	Modifier	CA listens for remote ringing signal followed by a 5-seconds silence. If remote answer is not detected, CA responds as defaulted
NOTE – The dial string modifiers may be subject to national administration agreements.		

Syntax:

```
<File-of-addrspec>  :=          PATH
                                --  The path points to a file containing one or more
                                --  <Addrspec-parameter>s

<Addrspec-parameter>  :=        <Address-parameter>
                                ["," <Cover-path> "," <Cover-conv> ["," [<Cover-type>] ["," <Nopgbrk>]]]

<Address-parameter>  :=         (<Phone-number> ["#" <Subaddress>]) |
                                ("!" <Dial-command> ["#" <Subaddress>]) |
                                ("&" <Alias>)
                                --  contains telephone dialing sequence or alias

<Phone-number>  :=              NUMERIC-STRING

<Dial-command>  :=              <Dial-operator> {<Dial-operator>}

<Dial-operator>  :=             "0" | ... | "9" | "*" | "#" | ";" | ":" | "," | "T" | "t" | "P" | "p" | "W" | "w" | "@"

<Subaddress>  :=                <Subaddress-parameter>
                                --  see 11.2.2.7

<Alias>  :=                     STRING

<Cover-path>  :=                PATH
                                --  Path to a file containing a cover page for specific addressee
```

<Cover-conv> :=	<Convert-id-parameter> -- *Specifies the Transfer Format of the cover page file*
<Cover-type> :=	<Type-id-parameter> -- *Specifies the Transmission Format of the cover page*
<Nopgbrk> :=	<Boolean-parameter> -- *If set to true ("Yes") then no page break will occur between the cover* -- *page and the main document*

11.2.2.5 File-of-filespec and File-parameter

The File-of-filespec parameter is encoded as PATH which points to a file containing Filespec-parameters, which contain the File-parameter. The File-parameter itself is also encoded as PATH, which points to the file transferred.

Syntax:

<File-of-filespec> :=	PATH -- *The path points to a file containing one or more* -- *<Filespec-parameter>s*
<Filespec-parameter> :=	<File-parameter> "," <File-conv> ["," <File-type>]
<File-parameter> :=	PATH -- *Path to the file transferred*
<File-conv> :=	<Convert-id-parameter> -- *Specifies the Transfer Format of the file.*
<File-type> :=	<Type-id-parameter> -- *Specifies the Transmission Format of the file*

11.2.2.6 G3-speed-parameter

The G3-speed-parameter is encoded as NUMERIC-STRING set to one of the following values:

"2400"	2400 bits per second
"4800"	4800 bits per second
"7200"	7200 bits per second
"9600"	9600 bits per second
"12200"	12200 bits per second
"14400"	14400 bits per second

Syntax:

<G3-speed-parameter> :=	"2400" | "4800" | "7200" | "9600" | "12200" | "14400"

11.2.2.7 Sub-address-parameter

The Sub-address-parameter is encoded as STRING that represents a sub-address. As defined per ITU-T Recommendation T.30 (1994) the STRING is restricted to digits ("0".."9") and the characters "+" and "–". The "_" (underline) character shall be translated into a IRA space character (20_{HEX}) by the CA. The length of the Sub-address-parameter is restricted to 20 octets.

Syntax:

<Sub-address-parameter> :=	<Subaddress-digit> {<Subaddress-digit>}
<Subaddress-digit> :=	"0" | ... | "9" | "*" | "#" -- *the length of the <Sub-address-parameter> is restricted to 20 octets.*

11.2.2.8 Resolution-parameter

The Resolution-parameter is encoded as NUMERIC-STRING set to one of the following values:

"0"	8 pels per mm horizontal, 3,85 lines per mm vertical (98 dpi vertical)
"1"	8 pels per mm horizontal, 7,7 lines per mm vertical (196 dpi vertical)
"2"	200 dpi horizontal, 200 dpi vertical (200 dpi)
"3"	300 dpi horizontal, 300 dpi vertical (300 dpi)
"4"	400 dpi horizontal, 300 dpi vertical (400 dpi)
"5"	8 pels per mm horizontal, 15,4 lines per mm vertical (392 dpi vertical)
"6"	16 pels per mm horizontal, 15,4 lines per mm vertical (392 dpi vertical)

Syntax:

```
<Resolution-parameter>  :=          "0" | ... | "6"
```

11.2.2.9 Password-parameter

The Password-parameter is encoded as STRING. As defined per ITU-T Recommendation T.30 (1994) the character-set of the STRING is restricted to digits ("0".."9") and the characters "+" and "–". The "_" (underline) character shall be translated into a IRA space character (20_{HEX}) by the CA.

Syntax:

```
<Password-parameter>  :=          <Password-digit> {<Password-digit>}

<Password-digit>  :=              "0" | ... | "9" | "*" | "#"
                                  --  the length of the <Password-parameter> is restricted to 20 octets.
```

11.2.2.10 Poll-select-parameter

The Poll-select-parameter is encoded as STRING. As defined per ITU-T Recommendation T.30 (1994) the character-set of the STRING is restricted to digits ("0".."9") and the characters "+" and "–". The "_" (underline) character shall be translated into a IRA space character (20_{HEX}) by the CA.

Syntax:

```
<Poll-select-parameter>  :=       <Poll-select-digit> {<Poll-select-digit>}

<Poll-select-digit>  :=           "0" | ... | "9" | "*" | "#"
                                  --  the length of the <Poll-select-parameter> is restricted to 20 octets.
```

11.3 Additional Functionality

11.3.1 Function: Send and SendAck

See Tables 68 and 69.

11.3.2 Function: Receive

See Table 70.

11.3.3 Function: Poll

This extension consists in giving a LA the ability to retrieve a series of documents posted by a remote fax device.

For instance, if user "John" sends documents to remote fax "Tom", and "Tom" has posted documents to be retrieved by "John", the "Polling" extension allows "John" to receive the documents from "Tom" in the same communication session.

The <ExtendTDD>, expanded with the <PollExtension> is used to specifically retrieve the documents posted by the remote device (see Table 71).

TABLE 68/T.611

Additional functionality of the <SendTDD> for Telefax Group 3

Syntax Element	C	T	I/O	Text Based Encoding			Comment
				Keyword	Parameter	Default	
<G3Speed>	B	o	I	G3SPEED	<G3-speed-parameter>	Highest speed available	Modulation speed. If the CA cannot return the modulation speed used, it shall blank the field
<GenCil>	+	o	I	GENCIL	<Boolean-parameter>	"Yes"	Asks the CA to generate a CIL in the transmitted file
<HighRes>	+	o	I	HIGHRES	<Resolution-parameter>	"0"	Enforces higher resolution when set to values greater than "0". See also 11.2.2.8
<SubAddress>	+	o	I	SUBADDR	<Sub-address-parameter>	–	Originator's sub-address
<UseEcm>	+	o	I	USEECM	<Boolean-parameter>	"Yes"	States whether Error Correction Mode shall be used
Send to one addressee							
<Recipient>	B	m	I	ADDRESS	<Address-parameter>	–	Specifies one recipient's call number
<DoPoll>	+	o	I	DOPOLL	<Boolean-parameter>	"No"	If "Yes" receiver is polled for sending a document
<PollPassword>	+	o	I	PASSWORD	<Password-parameter>	–	
<PollSelector>	+	o	I	SELECT	<Poll-select-parameter>	–	
Send to one or many addressees							
<RecipientSpec>	+	m	I	ADDRESS	"@" <File-of-addrspec>	–	Specifies a list of recipients
Send only one file							
<Convert>	B	m	I	CONVERT	<Convert-id-parameter>	–	States the Transfer Format of the outgoing file
<Document>	B	m	I	FILENAME	<File-parameter>	–	The single outgoing file to be transmitted, delivered under the Transfer Format specified by the Convert keyword
<Type>	B	o	I	TYPE	<Type-id-parameter>	"STD"	Specifies the document type to be sent
<From>	+	o	I	FROM	NUMERIC-STRING	First page	The first page to be actually sent; does only apply to text files
<To>	+	o	I	TO	NUMERIC-STRING	Last page	The last page to be actually sent; does only apply to text files
Send one or many files							
<DocumentSpec>	+	m	I	FILENAME	"@" <File-of-filespec>	–	Specifies a list of files; uses a special syntax

TABLE 69/T.611

Additional functionality of the <SendAckTDD> for Telefax Group 3

Syntax Element	C	T	I/O	Text Based Encoding			Comment
				Keyword	Parameter	Default	
<Recipient>	B	m	I/O	ADDRESS	<Address-parameter>	–	Specifies one recipient’s call number
<G3Speed>	B	o	I/O	G3SPEED	<G3-speed-parameter>	Highest speed available	Modulation speed. If the CA cannot return the modulation speed used, it shall blank the field
<GenCil>	+	o	I	GENCIL	<Boolean-parameter>	"Yes"	Asks the CA to generate a CIL in the transmitted file
<HighRes>	+	o	I	HIGHRES	<Resolution-parameter>	"0"	Enforces higher resolution when set to values greater than "0". See also 11.2.2.8
<SubAddress>	+	o	I	SUBADDR	<Sub-address-parameter>	–	Originator’s sub-address
<UseEcm>	+	o	I/O	USEECM	<Boolean-parameter>	"Yes"	States whether Error Correction Mode shall be used
<DoPoll>	+	o	I	DOPOLL	<Boolean-parameter>	"No"	if "Yes" receiver is polled for sending a document
<PollPassword>	+	o	I	PASSWORD	<Password-parameter>	–	
<PollSelector>	+	o	I	SELECT	<Poll-select-parameter>	–	
Send only one file							
<Convert>	B	m	I	CONVERT	<Convert-id-parameter>	–	States the Transfer Format of the outgoing file
<Document>	B	m	I	FILENAME	<File-parameter>	–	The single outgoing file to be transmitted, delivered under the Transfer Format specified by the Convert keyword
<Type>	B	o	I	TYPE	<Type-id-parameter>	"STD"	Specifies the document type to be sent
<From>	+	o	I	FROM	NUMERIC-STRING	First page	The first page to be actually sent; does only apply to text files
<To>	+	o	I	TO	NUMERIC-STRING	Last page	The last page to be actually sent; does only apply to text files
Send one or many files							
<DocumentSpec>	+	m	I	FILENAME	"@" <File-of-filespec>	–	Specifies a list of files; uses a special syntax

TABLE 70/T.611

Additional functionality of the <ReceiveTDD> for Telefax Group 3

Syntax Element	C	T	I/O	Text Based Encoding			Comment
				Keyword	Parameter	Default	
<CvFax3>	B	o	I	CVFAX3	<Convert-id-parameter>	"TIFF"	Transfer format desired by the LA
<Originator>	B	o	O	ADDRESS	<Address-parameter>	–	Specifies orginator's phone number
<G3Speed>	+	o	O	G3SPEED	<G3-speed-parameter>	–	Modulation speed. If the CA cannot return the modulation speed used, field will be left empty
<SubAddress>	+	o	I/O	SUBADDR	<Sub-address-parameter>	–	Recipient's sub-address; if specified in the Request, used as a selector for the retrieval
Only one file received							
<Convert>	B	m	O	CONVERT	<Convert-id-parameter>	–	States the Transfer Format of the received file
<Document>	B	m	I/O	FILENAME	<File-parameter>	–	The filename may be pre-set by the LA on request. If only one file is received, the name shall be retained by the CA. If many files are received the name may be overwritten by the CA
<Type>	B	o	O	TYPE	<Type-id-parameter>	–	Specifies the document type received
Many files received							
<DocumentSpec>	+	m	O	FILENAME	"@" <File-of-filespec>	–	Specifies a list of files; uses a special syntax

11.4 CA-Descriptor Settings

A CA supporting the FX3 service shall specify the supported Type-IDs (Transmission Formats) in the CA-Descriptor (see 9.5)

For support of the Poll TDD, the EXTEND keyword of the CA-Descriptor has to be set accordingly. Table 72 depicts this.

TABLE 71/T.611

Syntax elements of <PollExtension> for Telefax Group 3

Syntax Element	C	T	I/O	Text Based Encoding			Comment
				Keyword	Parameter	Default	
<PollSubFunction>	B	m	I	SUBFUNC	"Poll"	–	CA shall generate a response
<Recipient>	B	m	I	ADDRESS	<Address-parameter>	–	Specifies one recipient’s call number
<PollPassword>	B	m	I	PASSWORD	<Password-parameter>	–	
<PollSelector>	B	o	I	SELECT	<Poll-select-parameter>		
<SendTime>	B	o	I	SENDTIME	<Send-time-parameter>	"IMMEDIATE"	CA shall actually process the request at the time specified
<ComId>	B	o	O	COMID	<Com-id-parameter>	–	Identification of the communication (computed by the CA)

TABLE 72/T.611

Additional CA-Descriptor settings for Telefax Group 3

Keyword	Parameter	CA declares that
FX3	"STD"	the Basic Telefax service (MH) is supported
FX3	"BTM"	Basic Transparent Mode is supported
FX3	"DTM"	Document Transparent Mode is supported
FX3	"BFT"	Binary File Transfer is supported
FX3	"EDI"	Electronic Data Interchange is supported
EXTEND	"Poll"	the Poll functionality is supported

12 Service: Telefax Group 4

12.1 Service Specific Syntax Elements

<ServiceDependentKeywordsSend> :=
(<Recipient> | <RecipientSpec>)
((<Document> <Convert> [<Type>] [<Name>] [<UserInfo>] [<Prolog>] [From] [To]) | <DocumentSpec>) [<SubAddress>] [<HighRes>]

<ServiceDependentKeywordsSendAck> :=
<Recipient>
((<Document> <Convert> [<Type>] [<Name>] [<UserInfo>] [<Prolog>] [<Cil>] [From] [To]) | <DocumentSpec>) [<SubAddress>] [<HighRes>]

<ServiceDependentKeywordsReceive> :=
[<Originator>]
((<Document> <Convert> <Type> [<Name>] [<UserInfo>] [<Prolog>] [<Cil>] [<FirstPg>]) | <DocumentSpec>) [<CvFax4>] [<SubAddress>]

See Table 73.

TABLE 73/T.611

Additional syntax elements for Telefax Group 4

Syntax Element	Purpose
<Cil>	Specifies the contents of the Call Identification Line as defined per ITU-T Recommendation F.200. See also G.1
<Convert>	Specifies the Transfer Format to be used
<CvFax4>	Specifies the desired Transfer Format for received files
<Document>, <DocumentSpec>	Specifies the document or the documents to be sent or being received
<FirstPg>	Specifies the number of the first received page
<From>	Transmission should start from the page number specified
<HighRes>	Indicates the resolution
<Name>	Assigns a name to the document to be sent or indicates the name of the received document as defined per ITU-T Recommendation T.571
<Originator>	Specifies the communications address of the originator
<Prolog>	Path to the "prolog" document. This is a document, coded according to ITU Recommendation T.61, which is prepended to the main document to be sent
<Recipient>, <RecipientSpec>	Specifies the communications address(es) of the recipient(s)
<SubAddress>	Specifies the Originator's sub-address
<To>	Transmission should finish on page number specified
<Type>	Specifies the Transmission Format used
<UserInfo>	Specifies a comment attached to the document to be sent/received. The comment is transmitted via the telecommunications service

12.2 Text Based Encoding

12.2.1 Mapping of Keywords

See Table 74.

TABLE 74/T.611

Text Based Encoding of additional syntax elements for Telefax Group 4
(A ⏎ denotes the new line format effector)

Syntax Element	Keyword/Parameter Pair
<Cil>	"CIL" ":" <Cil-parameter> ⏎
<Convert>	"CONVERT" ":" <Convert-id-parameter> ⏎
<CvFax4>	"CVFAX4" ":" <Convert-id-parameter> ⏎
<Document>	"FILENAME" ":" <File-parameter> ⏎
<DocumentSpec>	"FILENAME" ":" "@" <File-of-filespec> ⏎
<FirstPg>	"FIRSTPG" ":" NUMERIC-STRING ⏎
<From>	"FROM" ":" NUMERIC-STRING ⏎
<HighRes>	"HIGHRES" ":" <Resolution-parameter> ⏎
<Name>	"NAME" ":" STRING (SIZE(1..12)) ⏎
<Originator>	"ADDRESS" ":" <Address-parameter> ⏎
<Prolog>	"PROLOG" ":" <Path-parameter> ⏎
<Recipient>	"ADDRESS" ":" <Address-parameter> ⏎
<RecipientSpec>	"ADDRESS" ":" "@" <File-of-addrspec> ⏎
<SubAddress>	"SUBADDR" ":" <Sub-address-parameter> ⏎
<To>	"TO" ":" NUMERIC-STRING ⏎
<Type>	"TYPE" ":" <Type-id-parameter> ⏎
<UserInfo>	"USERINFO" ":" STRING (SIZE(1..12)) ⏎

12.2.2 Encoding of Parameters

See also 6.4.4 for the encoding of non service dependent parameters used.

12.2.2.1 Service-id-parameter

The Service-id parameter is encoded as STRING set to the constant value "FX4".

Syntax:

<Service-id-parameter> := "FX4"

12.2.2.2 Type-id-parameter

The Type-id-parameter is encoded as STRING set to one of the following values:

"STD"	Basic Telefax G4 Service (MR)
"DTM"	Telematic File Transfer (TFT) of Telefax G4 Service: Document Transparent Mode
"EDI"	Telematic File Transfer (TFT) of Telefax G4 Service: Edifact
"BFT"	Telefax G4 Service: Binary File Transfer

NOTE – The Transmission Format STD (=Modified Read Code, MR) is defined per ITU-T Recommendation T.6, the TFT formats DTM and EDI are defined per ITU-T Recommendation T.571 and the Transmission Format BFT is defined per ITU-T Recommendation T.434.

Syntax:

<Type-id-parameter> := "STD" | "DTM" | "BFT" | "EDI"

12.2.2.3 Convert-id-parameter

The Convert-id-parameter is encoded as STRING set to one of the following values:

"ASCII"	APPLI/COM Extended ASCII (For outgoing files only)
"ASCII437"	APPLI/COM Extended ASCII (For outgoing files only)
"T.50"	APPLI/COM Standard ASCII (For outgoing files only)
"T.61"	APPLI/COM Transfer Format T.61
"TIFF"	TIFF Transfer Format as defined in 8.3
"TIFF2"	(For incoming files only). APPLI/COM TIFF Class 2 as specified in 8.3
"TIFF4"	(For incoming files only). APPLI/COM TIFF Class 4 as specified in 8.3
"VOID"	No conversion to be done

NOTE – If a LA requests a Telefax document that has been received, it is possible to:

– obtain the document in APPLI/COM TIFF class 2 by specifying TIFF2;

– obtain the document in APPLI/COM TIFF class 4 by specifying TIFF4,

provided the CA is capable of generating that special TIFF format. If only TIFF is specified, the document will be delivered in the APPLI/COM TIFF default class (class 1). However, in the send direction it is sufficient to specify TIFF only (without number extension), since the compression information is contained in the Transfer Format itself.

The use of the Convert-id-parameter also depends on the Type-ID selected. Table 75 depicts this.

TABLE 75/T.611

Permitted Convert-id assignements dependent on Type-id and traffic direction

Type-id	Convert-id for outgoing traffic	Convert-id for incoming traffic
STD	ASCII, ASCIIxxx[a)], T.50, TIFF	TIFF, TIFFx[b)]
DTM, BFT, EDI	VOID	

[a)] xxx stands for a code-page, declared in the ICE, e.g. ASCII437 if code-page 437 has been declared.

[b)] x stands for the TIFF class to be read, the possible values are either 2 or 4.

Syntax:

```
<Convert-id-parameter>  :=     <Convert-std> | <Convert-bin> | <Convert-txt>

<Convert-std>  :=              <Convert-std-in> | <Convert-std-out>

<Convert-std-in>  :=           "TIFF" | "TIFF2" | "TIFF4"
                               --  incoming documents

<Convert-std-out>  :=          <Ascii> | "T.50" | "TIFF"
                               --  outgoing documents

<Convert-txt>  :=              <Ascii> | "T.50" | "T.61"

<Ascii>  :=                    "ASCII" | STRING ("ASCII" + <Code-page>)

<Code-page>  :=                <digit> <digit> <digit>

<digit>  :=                    "0" | ... | "9"

<Convert-bin>  :=              "VOID"
```

12.2.2.4 File-of-addrspec and Address-parameter

The File-of-addrspec parameter is encoded as PATH which points to a file containing Addrspec-parameters, which contain the Address-parameter.

The Address-parameter is encoded as STRING. The STRING consists of the recipient's terminal identifier as defined per ITU-T Recommentation F.184. The mnemonic part of the terminal identifier may be omitted or specified as a question mark ("?"), in which case no verification of the subscriber's address is performed. See also Annex G.2.

Alternatively an alias name may be given instead of the terminal identifier, provided the alias is introduced with a "&" character. It is assumed that the CA knows how to decode the alias specified. Typical use of this feature is when the CA implements a phone book.

Syntax:

```
<File-of-addrspec>  :=         PATH
                               --  The path points to a file containing one or more
                               --  <Addrspec-parameter>s

<Addrspec-parameter>  :=       <Address-parameter>
                               ["," <Cover-path> "," <Cover-conv> ["," [<Cover-type>] ["," <Nopgbrk>]]]

<Address-parameter>  :=        <Terminal-id> | ("&" <Alias>)

<Terminal-id>  :=              STRING
                               --  The string shall present the terminal-id as defined per
                               --  ITU-T Recommendation F.184

<Alias>  :=                    STRING

<Cover-path>  :=               PATH
                               --  Path to a file containing a cover page for specific addressee

<Cover-conv>  :=               <Convert-id-parameter>
                               --  Specifies the Transfer Format of the cover page file

<Cover-type>  :=               <Type-id-parameter>
                               --  Specifies the Transmission Format of the cover page

<Nopgbrk>  :=                  <Boolean-parameter>
                               --  If set to true ("Yes") then no page break will occur between the cover
                               --  page and the main document
```

12.2.2.5 File-of-filespec and File-parameter

The File-of-filespec parameter is encoded as PATH which points to a file containing Filespec-parameters, which contain the File-parameter. The File-parameter itself is also encoded as PATH, which points to the file transferred.

Syntax:

```
<File-of-filespec>  :=          PATH
                                --  The path points to a file containing one or more
                                --  <Filespec-parameter>s

<Filespec-parameter>  :=        <File-parameter> "," <File-conv> ["," [<File-type>]
                                ["," <Send-parameter> | <SendAck-parameter> | <Receive-parameter>] ]

<Send-parameter>  :=            [<File-name>] ["," <File-userinfo>]

<SendAck-parameter>  :=         [<File-name>] ["," [<File-userinfo>] ["," <File-cil>] ]

<Receive-parameter>  :=         [<File-name>] ["," [<File-userinfo>] ["," <File-cil>] ]

<File-parameter>  :=            PATH
                                --  Path to the file transferred

<File-conv>  :=                 <Convert-id-parameter>
                                --  Specifies the Transfer Format of the file.

<File-type>  :=                 <Type-id-parameter>
                                --  Specifies the Transmission Format of the file

<File-name>  :=                 STRING (SIZE(1..12))
                                --  Specifies the name of the file to be transmitted

<File-userinfo>  :=             STRING (SIZE(1..12))
                                --  Specifies additional user to user info to be transmitted

<File-cil>  :=                  Cil-parameter
                                --  See 12.2.2.6 below.
```

12.2.2.6 Cil-parameter

A string of 72 characters as defined per ITU-T Recommendation F.200, composed as follows (see also G.1):

Terminal identifier of recipient	Length: 24 characters
Separator	The character "/" ($2F_{HEX}$)
Terminal identifier of originator	Length: 24 characters
Separator	The character "/" ($2F_{HEX}$)
Local date and time of originator	Length: 14 characters, format: "YY-MM-DD-HH:MM"
Separator	The character "/" ($2F_{HEX}$)
Reference information	Length: 7 characters

Syntax:

```
<Cil-parameter>  :=             <Recipient-tid> "/" <Originator-tid> "/" <Local-time> "/" <Ref-info>

<Recipient-tid>  :=             STRING (SIZE(24))
                                --  The string presents the terminal-id as defined per
                                --  ITU-T Recommendation F.184

<Originator-tid>  :=            STRING (SIZE(24))
                                --  The string presents the terminal-id as defined per
                                --  ITU-T Recommendation F.184

<Local-time>  :=                <Date-time-parameter>

<Ref-info>  :=                  STRING (SIZE(7))
```

12.2.2.7 Sub-address-parameter

The Sub-address-parameter is encoded as NUMERIC-STRING that represents a sub-address.

Syntax:

```
<Sub-address-parameter>  :=        NUMERIC-STRING (SIZE(1..4))
```

12.2.2.8 Resolution-parameter

The Sub-address-parameter is encoded as NUMERIC-STRING set to one of the following values:

"0"	200 dpi
"1"	240 dpi
"2"	300 dpi
"3"	400 dpi
"4"	400 dpi

Syntax:

```
<Resolution-parameter>  :=         "0" | ... | "4"
```

12.3 Additional Functionality

12.3.1 Function: Send and SendAck

See Tables 76 and 77.

12.3.2 Function: Receive

See Table 78.

12.4 CA-Descriptor Settings

A CA supporting the FX4 service shall specify the supported Type-id (document Transmission Formats) in the CA-Descriptor (see 9.5) Table 79 below shows the possible assignments.

13 Service: Teletex

13.1 Service Specific Syntax Elements

```
<ServiceDependentKeywordsSend>  :=
                         (<Recipient> | <RecipientSpec>)
                         ((<Document> <Convert> [<Type>] [<Name>] [<UserInfo>] [<Prolog>] [From]
                         [To] [T61Options]) | <DocumentSpec>) [<SubAddress>]

<ServiceDependentKeywordsSendAck>  :=
                         <Recipient>
                         ((<Document> <Convert> [<Type>] [<Name>] [<UserInfo>] [<Prolog>] [<Cil>]
                         [From] [To] [T61Options]) | <DocumentSpec>) [<SubAddress>]

<ServiceDependentKeywordsReceive>  :=
                         [<Originator>]
                         ((<Document> <Convert> <Type> [<Name>] [<UserInfo>] [<Prolog>] [<Cil>]
                         [<FirstPg>]) | <DocumentSpec>) [<CvTtx>] [<SubAddress>]
```

See Table 80.

TABLE 76/T.611

Additional functionality of the <SendTDD> for Telefax Group 4

Syntax Element	C	T	I/O	Text Based Encoding			Comment
				Keyword	Parameter	Default	
<HighRes>	+	o	I	HIGHRES	<Resolution-parameter>	"0"	Enforces higher resolution when set to values greater than "0". See also 12.2.2.8
<SubAddress>	+	o	I	SUBADDR	<Sub-address-parameter>	–	Originator's sub-address
Send to one addressees							
<Recipient>	B	m	I	ADDRESS	<Address-parameter>	–	Specifies one recipient's call number
Send to one or many addressees							
<RecipientSpec>	+	m	I	ADDRESS	"@" <File-of-addrspec>	–	Specifies a list of recipients
Send only one file							
<Convert>	B	m	I	CONVERT	<Convert-id-parameter>	–	States the Transfer Format of the outgoing file
<Document>	B	m	I	FILENAME	<File-parameter>	–	The single outgoing file to be transmitted, delivered under the Transfer Format specified by the Convert keyword
<Type>	B	o	I	TYPE	<Type-id-parameter>	"STD"	Specifies the document type to be sent
<Name>	+	o	I	NAME	STRING (SIZE(1..12))	–	Assigns a "name" to the files to be sent. 12 characters maximum. It corresponds to the filename parameter in TFT (see ITU-T Rec. T.571)
<UserInfo>	+	o	I	USERINFO	STRING (SIZE(1..12))	–	Assigns a user comment to the document. This comment is transmitted with the document. 12 characters maximum
<Prolog>	+	o	I	PROLOG	PATH	–	Filename of the "control document" (full path)
<From>	+	o	I	FROM	NUMERIC-STRING	First page	The first page to be actually sent; does only apply to text files
<To>	+	o	I	TO	NUMERIC-STRING	Last page	The last page to be actually sent; does only apply to text files
Send one or many files							
<DocumentSpec>	+	m	I	FILENAME	"@" <File-of-filespec>	–	Specifies a list of files; uses a special syntax

TABLE 77/T.611

Additional functionality of the <SendAckTDD> for Telefax Group 4

Syntax Element	C	T	I/O	Text Based Encoding			Comment
				Keyword	Parameter	Default	
<Recipient>	B	m	I	ADDRESS	<Address-parameter>	–	Specifies one recipient's call number
<HighRes>	+	o	I	HIGHRES	<Resolution-parameter>	"0"	Enforces higher resolution when set > "0". See 12.2.2.8
<SubAddress>	+	o	I	SUBADDR	<Sub-address-parameter>	–	Originator's sub-address
Send only one file							
<Convert>	B	m	I	CONVERT	<Convert-id-parameter>	–	Transfer format of the outgoing file
<Document>	B	m	I	FILENAME	<File-parameter>	–	Single file to be transmitted, delivered under the Transfer Format specified by the Convert keyword
<Type>	B	o	I	TYPE	<Type-id-parameter>	"STD"	Specifies the document type to be sent
<Cil>	B	o	O	CIL	<Cil-parameter>	–	Call identification line built by the CA (see ITU-T Rec. F.200)
<Name>	+	o	I	NAME	STRING (SIZE(1..12))	–	Assigns a "name" to the files to be sent. Corresponds to the filename parameter in TFT (see ITU-T Rec. T.571)
<UserInfo>	+	o	I	USERINFO	STRING (SIZE(1..12))	–	Assigns a user comment to the document. This comment is transmitted with the document
<Prolog>	+	o	I	PROLOG	PATH	–	Filename of the "control document" (full path)
<From>	+	o	I	FROM	NUMERIC-STRING	First page	The first page to be actually sent; does only apply to text files
<To>	+	o	I	TO	NUMERIC-STRING	Last page	The last page to be actually sent; does only apply to text files
Send one or many files							
<DocumentSpec>	+	m	I	FILENAME	"@" <File-of-filespec>	–	Specifies a list of files; uses a special syntax

TABLE 78/T.611

Additional functionality of the <ReceiveTDD> for Telefax Group 4

Syntax Element	C	T	I/O	Text Based Encoding			Comment
				Keyword	Parameter	Default	
<Originator>	B	o	O	ADDRESS	<Address-parameter>	–	Specifies orginator's phone number
<CvFax4>	B	o	I	CVFAX4	<Convert-id-parameter>	"TIFF"	Transfer format desired by the LA
<SubAddress>	+	o	I/O	SUBADDR	<Sub-address-parameter>	–	Recipient's sub-address; if specified in the Request, used as a selector for the retrieval
Only one file received							
<Convert>	B	m	O	CONVERT	<Convert-id-parameter>	–	States the Transfer Format of the received file
<Document>	B	m	I/O	FILENAME	<File-parameter>	–	The filename may be pre-set by the LA on request. If only one file is received, the name shall be retained by the CA. If many files are received the name may be overwritten by the CA
<Type>	B	o	O	TYPE	<Type-id-parameter>	–	Specifies the document type received
<Cil>	B	o	O	CIL	<Cil-parameter>	–	Call identification line built by the CA (see ITU-T Rec. F.200)
<Name>	+	o	O	NAME	STRING (SIZE(1..12))	–	Returns the name of the incoming file (12 characters maximum)
<UserInfo>	+	o	O	USERINFO	STRING (SIZE(1..12))	–	Assigns a user comment to the document. This comment is transmitted with the document. 12 characters maximum
<FirstPg>	+	o	O	FIRSTPG	NUMERIC-STRING	–	Number of the first page received
<Prolog>	+	o	I	PROLOG	PATH	–	Filename of the Control document
Many files received							
<DocumentSpec>	+	m	O	FILENAME	"@" <File-of-filespec>	–	Specifies a list of files; uses a special syntax

TABLE 79/T.611

Additional CA-Descriptor settings for Telefax Group 4

Keyword	Parameter	CA declares that
FX4	"STD"	the Basic Telefax service is supported
FX4	"OPD"	the Operator Document transmission is supported
FX4	"MD"	the Monitor Document transmission is supported
FX4	"CTL"	Control Document transmission is supported
FX4	"DTM"	Document Transparent Mode is supported
FX4	"BFT"	Binary File Transfer is supported
FX4	"EDI"	Electronic Data Interchange is supported

TABLE 80/T.611

Additional syntax elements for the Teletex service

Syntax element	Purpose
<Cil>	Specifies the contents of the Call Identification Line as defined per ITU-T Recommendation F.200. See also G.1
<Convert>	Specifies the Transfer Format to be used
<CvTtx>	Specifies the desired Transfer Format for received files
<Document>, <DocumentSpec>	Specifies the document or the documents to be sent or being received
<FirstPg>	Specifies the number of the first received page
<From>	Transmission should start from the page number specified
<Name>	Assigns a name to the document to be sent or indicates the name of the received document as defined per ITU-T Recommendation T.571
<Originator>	Specifies the communications address of the originator
<Prolog>	Path to the "prolog" document
<Recipient>, <RecipientSpec>	Specifies the communications address(es) of the recipient(s)
<SubAddress>	Specifies the Originator’s sub-address
<T61Options>	Indicates the T.61 coding specific options
<To>	Transmission should finish on page number specified
<Type>	Specifies the Transmission Format used
<UserInfo>	Specifies a comment attached to the document to be sent/received. The comment is transmitted via the telecommunications service

13.2 Text Based Encoding

13.2.1 Mapping of Keywords

See Table 81.

TABLE 81/T.611

Text Based Encoding of additional syntax elements for the Teletex service
(A ↵ denotes the new line format effector)

Syntax Element	Keyword/Parameter Pair
<Cil>	"CIL" ":" <Cil-parameter> ↵
<Convert>	"CONVERT" ":" <Convert-id-parameter> ↵
<CvTtx>	"CVTTX" ":" <Convert-id-parameter> ↵
<Document>	"FILENAME" ":" <File-parameter> ↵
<DocumentSpec>	"FILENAME" ":" "@" <File-of-filespec> ↵
<FirstPg>	"FIRSTPG" ":" NUMERIC-STRING ↵
<From>	"FROM" ":" NUMERIC-STRING ↵
<Name>	"NAME" ":" (SIZE (1..12)) ↵
<Originator>	"ADDRESS" ":" <Address-parameter> ↵
<Prolog>	"PROLOG" ":" <Path-parameter> ↵
<Recipient>	"ADDRESS" ":" <Address-parameter> ↵
<RecipientSpec>	"ADDRESS" ":" "@" <File-of-addrspec> ↵
<SubAddress>	"SUBADDR" ":" <Sub-address-parameter> ↵
<T61Options>	"T61OPTS" ":" <T61-options-parameter> ↵
<To>	"TO" ":" NUMERIC-STRING ↵
<Type>	"TYPE" ":" <Type-id-parameter> ↵
<UserInfo>	"USERINFO" ":" STRING (SIZE (1..12)) ↵

13.2.2 Encoding of Parameters

See also 6.4.4 for the encoding of non service dependent parameters used.

13.2.2.1 Service-id-parameter

The Service-id parameter is encoded as STRING set to the constant value "TTX".

Syntax:

<Service-id-parameter> :=   "TTX"

13.2.2.2 Type-id-parameter

The Type-id-parameter is encoded as STRING set to one of the following values:

"STD"	Basic Teletex Service (T.61)
"OPD"	Basic Teletex Service: Operator Document
"MD"	Basic Teletex Service: Monitor Document
"CTL"	Basic Teletex Service: Control Document
"DTM"	Telematic File Transfer (TFT) of Teletex Service: Document Transparent Mode
"EDI"	Telematic File Transfer (TFT) of Teletex Service: Edifact
"BFT"	Teletex Service: Binary File Transfer

NOTE – The Transmission Format STD is defined per ITU-T Recommendation T.61, the document types OPD, MD and CTL are defined per ITU-T Recommendation T.62 (Annex E), the TFT formats DTM and EDI are defined per ITU-T Recommendation T.571 and the Transmission Format BFT is defined per ITU-T Recommendation T.434.

Syntax:

<Type-id-parameter> :=	"STD" | "OPD" | "MD" | "CTL" | "DTM" | "BFT" | "EDI"

13.2.2.3 Convert-id-parameter

The Convert-id-parameter is encoded as STRING set to one of the following values:

"ASCII"	APPLI/COM Extended ASCII
"ASCII437"	APPLI/COM Extended ASCII
"T.50"	APPLI/COM Standard ASCII
"T.61"	Basic Teletex Service coding as defined per ITU-T Recommendation T.61
"VOID"	No conversion to be done

The use of the Convert-id-parameter also depends on the Type-id selected. Table 82 depicts this.

TABLE 82/T.611

Permitted Convert-id assignements dependent on Type-id

Type-id	Permitted Convert-id for incoming and outgoing traffic
STD, OPD, MD, CTL	ASCII, ASCIIxxx[a)], T.50, T.61
DTM, BFT, EDI	VOID
a) xxx stands for a code-page, declared in the ICE, e.g. ASCII437 if code-page 437 has been declared.	

Syntax:

<Convert-id-parameter> :=	<Convert-txt> | <Convert-bin>
<Convert-txt> :=	<Ascii> | "T.50" | "T.61"
<Ascii> :=	"ASCII" | STRING ("ASCII" + <Code-page>)
<Code-page> :=	<digit> <digit> <digit>
<digit> :=	"0" | ... | "9"
<Convert-bin> :=	"VOID"

13.2.2.4 File-of-addrspec and Address-parameter

The File-of-addrspec parameter is encoded as PATH which points to a file containing Addrspec-parameters, which contain the Address-parameter.

The Address-parameter is encoded as STRING. The STRING consists of the recipient’s terminal identifier as defined per ITU-T Recommentation F.200. The mnemonic part of the terminal identifier may be omitted or specified as a question mark ("?"), in which case no verification of the subscriber’s address is performed. See also G.2.

Alternatively the presence of the call number as raw dialling number may be indicated by placing an exclamation point ("’!") in front of the first digit. This means the calling equipment must dial the digits that follow the exclamation point as is, without interpretation.

Furthermore an alias name may be given instead of the terminal identifier, provided the alias is introduced with a "&" character. It is assumed that the CA knows how to decode the alias specified.

Syntax:

<File-of-addrspec> :=	PATH -- *The path points to a file containing one or more* -- *<Addrspec-parameter>s*
<Addrspec-parameter> :=	<Address-parameter>
<Address-parameter> :=	<Terminal-id> | ("!" <Dial-sequence>) | ("&" <Alias>)
<Terminal-id> :=	STRING -- *The string shall present the terminal-id as defined per* -- *ITU-T Recommendation F.200*
<Dial-sequence> :=	NUMERIC-STRING
<Alias> :=	STRING

13.2.2.5 File-of-filespec and File-parameter

The File-of-filespec parameter is encoded as PATH which points to a file containing Filespec-parameters, which contain the File-parameter. The File-parameter itself is also encoded as PATH, which points to the file transferred.

Syntax:

<File-of-filespec> :=	PATH -- *The path points to a file containing one or more* -- *<Filespec-parameter>s*
<Filespec-parameter> :=	<File-parameter> "," <File-conv> ["," [<File-type>] ["," <Send-parameter> | <SendAck-parameter> | <Receive-parameter>]]
<Send-parameter> :=	[<File-name>] ["," <File-userinfo>]
<SendAck-parameter> :=	[<File-name>] ["," [<File-userinfo>] ["," <File-cil>]]
<Receive-parameter> :=	[<File-name>] ["," [<File-userinfo>] ["," <File-cil>]]
<File-parameter> :=	PATH -- *Path to the file transferred*
<File-conv> :=	<Convert-id-parameter> -- *Specifies the Transfer Format of the file.*
<File-type> :=	<Type-id-parameter> -- *Specifies the Transmission Format of the file*
<File-name> :=	STRING (SIZE(1..12)) -- *Specifies the name of the file to be transmitted*

<File-userinfo> :=	STRING (SIZE(1..12)) -- *Specifies additional user to user info to be transmitted*
<File-cil> :=	Cil-parameter -- See 13.2.2.6

13.2.2.6 Cil-parameter

A string of 72 characters as defined per ITU-T Recommendation F.200, composed as follows (see also G.1):

Terminal identifier of recipient	Length: 24 characters
Separator	The character "/" ($2F_{HEX}$)
Terminal identifier of originator	Length: 24 characters
Separator	The character "/" ($2F_{HEX}$)
Local date and time of originator	Length: 14 characters, format: "YY-MM-DD-HH:MM"
Separator	The character "/" ($2F_{HEX}$)
Reference information	Length: 7 characters

Syntax:

<Cil-parameter> :=	<Recipient-tid> "/" <Originator-tid> "/" <Local-time> "/" <Ref-info>
<Recipient-tid> :=	STRING (SIZE(24)) -- *The string presents the terminal-id as defined per* -- *ITU-T Recommendation F.184*
<Originator-tid> :=	STRING (SIZE(24)) -- *The string presents the terminal-id as defined per* -- *ITU-T Recommendation F.184*
<Local-time> :=	<Date-time-parameter>
<Ref-info> :=	STRING (SIZE(7))

13.2.2.7 Sub-address-parameter

The Sub-address-parameter is encoded as NUMERIC-STRING that represents a sub-address.

Syntax:

<Sub-address-parameter> :=	NUMERIC-STRING (SIZE(1..4))

13.2.2.8 T61-options-parameter

For further study.

Syntax:

<T61-options-parameter> :=	STRING

13.3 Additional Functionality

13.3.1 Function: Send and SendAck

See Tables 83 and 84.

13.3.2 Function: Receive

See Table 85.

TABLE 83/T.611

Additional functionality of the <SendTDD> for the Teletex service

Syntax Element	C	T	I/O	Text Based Encoding			Comment
				Keyword	Parameter	Default	
<SubAddress>	+	o	I	SUBADDR	<Sub-address-parameter>	–	Originator's sub-address
Send to one addressee							
<Recipient>	B	m	I	ADDRESS	<Address-parameter>	–	Specifies one recipient's call number
Send to one or many addressees							
<RecipientSpec>	+	m	I	ADDRESS	"@" <File-of-addrspec>	–	Specifies a list of recipients
Send only one file							
<Convert>	B	m	I	CONVERT	<Convert-id-parameter>	–	Transfer format of the outgoing file
<Document>	B	m	I	FILENAME	<File-parameter>	–	Single file to be transmitted.
<Type>	B	o	I	TYPE	<Type-id-parameter>	"STD"	Specifies the document type to be sent
<Name>	+	o	I	NAME	STRING (SIZE(1..12))	–	Filename parameter in TFT (T.571)
<UserInfo>	+	o	I	USERINFO	STRING (SIZE(1..12))	–	Userinfo is transmitted with the document
<Prolog>	+	o	I	PROLOG	PATH	–	Path to control document (T.62)
<From>	+	o	I	FROM	NUMERIC-STRING	First page	The first page to be actually sent; does only apply to text files
<To>	+	o	I	TO	NUMERIC-STRING	Last page	The last page to be actually sent; does only apply to text files
<T61Options>	+	o	I	T61OPTS	<T61-options-parameter>	–	
Send one or many files							
<DocumentSpec>	+	m	I	FILENAME	"@" <File-of-filespec>	–	Specifies a list of files; uses a special syntax

TABLE 84/T.611

Additional functionality of the <SendAckTDD> for the Teletex service

Syntax Element	C	T	I/O	Text Based Encoding			Comment
				Keyword	Parameter	Default	
<Recipient>	B	m	I	ADDRESS	<Address-parameter>	–	Specifies one recipient's call number
<SubAddress>	+	o	I	SUBADDR	<Sub-address-parameter>	–	Originator's sub-address
Send only one file							
<Convert>	B	m	I	CONVERT	<Convert-id-parameter>	–	Transfer format of the outgoing file
<Document>	B	m	I	FILENAME	<File-parameter>	–	Single file to be transmitted, delivered under the Transfer Format specified by the Convert keyword
<Type>	B	o	I	TYPE	<Type-id-parameter>	"STD"	Specifies the document type to be sent
<Cil>	B	o	O	CIL	<Cil-parameter>	–	Call identification line built by the CA (see ITU-T Rec. F.200)
<Name>	+	o	I	NAME	STRING (SIZE(1..12))	–	Assigns a "name" to the files to be sent. Corresponds to the filename parameter in TFT (see ITU-T Rec. T.571)
<UserInfo>	+	o	I	USERINFO	STRING (SIZE(1..12))	–	Assigns an user comment to the document. This comment is transmitted with the document
<Prolog>	+	o	I	PROLOG	PATH	–	Filename of the "control document" (full path)
<From>	+	o	I	FROM	NUMERIC-STRING	First page	The first page to be actually sent; does only apply to text files
<To>	+	o	I	TO	NUMERIC-STRING	Last page	The last page to be actually sent; does only apply to text files
<T61Options>	+	o	I	T61OPTS	<T61-options-parameter>	–	
Send one or many files							
<DocumentSpec>	+	m	I	FILENAME	"@" <File-of-filespec>	–	Specifies a list of files; uses a special syntax

TABLE 85/T.611

Additional functionality of the <ReceiveTDD> for the Teletex service

Syntax Element	C	T	I/O	Text Based Encoding			Comment
				Keyword	Parameter	Default	
<Originator>	B	o	O	ADDRESS	<Address-parameter>	–	Specifies orginator's phone number
<CvTtx>	B	o	I	CVTTX	<Convert-id-parameter>	"T.61"	Transfer format desired by the LA
<SubAddress>	+	o	I/O	SUBADDR	<Sub-address-parameter>	–	Recipient's sub-address; if specified in the Request, used as a selector for retrieval
Only one file received							
<Convert>	B	m	O	CONVERT	<Convert-id-parameter>	–	States the Transfer Format of the received file
<Document>	B	m	I/O	FILENAME	<File-parameter>	–	Filename may be pre-set by the LA. If only one file is received, the path shall be retained by the CA. If many files are received the name may be overwritten by the CA
<Type>	B	o	O	TYPE	<Type-id-parameter>	–	Specifies the document type received
<Cil>	B	o	O	CIL	<Cil-parameter>	–	Call identification line built by the CA (see ITU-T Rec. F.200)
<Name>	+	o	O	NAME	STRING (SIZE(1..12))	–	Returns the name of the incoming file (12 characters maximum)
<UserInfo>	+	o	O	USERINFO	STRING (SIZE(1..12))	–	Assigns an user comment to the document. This comment is transmitted with the document
<FirstPg>	+	o	O	FIRSTPG	NUMERIC-STRING	–	Number of the first page received
<Prolog>	+	o	I	PROLOG	PATH	–	Filename of the Control document
Many files received							
<DocumentSpec>	+	m	O	FILENAME	"@" <File-of-filespec>	–	Specifies a list of files; uses a special syntax

13.4 CA-Descriptor Settings

A CA supporting the TTX service shall specify the supported Type-id (document Transmission Formats) in the CA-Descriptor (see 9.5) Table 86 below shows the possible assignments.

TABLE 86/T.611

Additional CA-Descriptor settings for the Teletex service

Keyword	Parameter	CA declares that
TTX	"STD"	the basic Teletex service (T.61) is supported
TTX	"OPD"	the Operator Document transmission is supported
TTX	"MD"	the Monitor Document transmission is supported
TTX	"CTL"	Control Document transmission is supported
TTX	"DTM"	Document Transparent Mode is supported
TTX	"BFT"	Binary File Transfer is supported
TTX	"EDI"	Electronic Data Interchange is supported

14 Service: Telex via Teletex

14.1 Service Specific Syntax Elements

```
<ServiceDependentKeywordsSend>  :=
                    (<Recipient> | <RecipientSpec>)
                    ((<Document> <Convert> [<Type>] [From] [To]) | <DocumentSpec>)
                    [<SubAddress>] [<Notify>]

<ServiceDependentKeywordsSendAck>  :=
                    <Recipient>
                    ((<Document> <Convert> [<Type>] [From] [To]) | <DocumentSpec>)
                    [<SubAddress>] [<Notify>]

<ServiceDependentKeywordsReceive>  :=
                    [<Originator>] ((<Document> <Convert> <Type>) | <DocumentSpec>)
                    [<CvTx>] [<SubAddress>]
```

TABLE 87/T.611

Additional syntax elements for the Telex via Teletex service

Syntax element	Purpose
<Convert>	Specifies the Transfer Format to be used
<CvTx>	Specifies the desired Transfer Format for received files
<Document>, <DocumentSpec>	Specifies the document or the documents to be sent or being received
<From>	Transmission should start from the page number specified
<Notify>	Specifies whether delivery notification is requested
<Originator>	Specifies the communications address of the originator
<Recipient>, <RecipientSpec>	Specifies the communications address(es) of the recipient(s)
<SubAddress>	Specifies the Originator’s sub-address
<To>	Transmission should finish on page number specified
<Type>	Specifies the Transmission Format used

14.2 Text Based Encoding

14.2.1 Mapping of Keywords

See Table 88.

TABLE 88/T.611

Text Based Encoding of additional syntax elements for the Telex via Teletex service
(A ⏎ denotes the new line format effector)

Syntax Element	Keyword/Parameter Pair
<Convert>	"CONVERT" ":" <Convert-id-parameter> ⏎
<CvTx>	"CVTX" ":" <Convert-id-parameter> ⏎
<Document>	"FILENAME" ":" <File-parameter> ⏎
<DocumentSpec>	"FILENAME" ":" "@" <File-of-filespec> ⏎
<From>	"FROM" ":" NUMERIC-STRING ⏎
<Notify>	"NOTIFY" ":" <Boolean-parameter> ⏎
<Originator>	"ADDRESS" ":" <Address-parameter> ⏎
<Recipient>	"ADDRESS" ":" <Address-parameter> ⏎
<RecipientSpec>	"ADDRESS" ":" "@" <File-of-addrspec> ⏎
<SubAddress>	"SUBADDR" ":" <Sub-address-parameter> ⏎
<To>	"TO" ":" NUMERIC-STRING ⏎
<Type>	"TYPE" ":" <Type-id-parameter> ⏎

14.2.2 Encoding of Parameters

See also 6.4.4 for the encoding of non service dependent parameters used.

14.2.2.1 Service-id-parameter

The Service-id parameter is encoded as STRING set to the constant value "TX".

Syntax:

<Service-id-parameter> := "TX"

14.2.2.2 Type-id-parameter

The Type-id-parameter is encoded as STRING set to the constant value "STD".

Syntax:

<Type-id-parameter> := "STD"

NOTE – The Transmission Format STD is defined per ITU-T Recommendation S.1.

14.2.2.3 Convert-id-parameter

The Convert-id-parameter is encoded as STRING set to one of the following values:

"ASCII"	APPLI/COM Extended ASCII
"ASCII437"	APPLI/COM Extended ASCII
"T.50"	APPLI/COM Standard ASCII

Syntax:

```
<Convert-id-parameter>  :=      <Ascii> | "T.50"

<Ascii>  :=     "ASCII" | STRING ("ASCII" + <Code-page>)

<Code-page>  :=    <digit> <digit> <digit>

<digit>  :=     "0" | ... | "9"
```

14.2.2.4 File-of-addrspec and Address-parameter

The File-of-addrspec parameter is encoded as PATH which points to a file containing Addrspec-parameters, which contain the Address-parameter.

The Address-parameter is encoded as STRING forming the call number. If a subscriber identification verification is to be performed, the answerback unit of the receiving terminal must be entered after the call number, separated by a "equal" sign ("=").

Alternatively an alias name may be given instead of the terminal identifier, provided the alias is introduced with a "&" character. It is assumed that the CA knows how to decode the alias specified.

NOTE – If it is intended to perform a subscriber identification verification, the complete answerback unit must be entered after the equal sign, and not just the alphabetic component of the receiver.

Syntax:

```
<File-of-addrspec>  :=          PATH
                                --  The path points to a file containing one or more
                                --  <Addrspec-parameter>s

<Addrspec-parameter>  :=        <Address-parameter>

<Address-parameter>  :=         <Telex-address> | ("&" <Alias>)

<Telex-address>  :=             STRING
                                --  The string shall present a valid telex address

<Alias>  :=                     STRING
```

14.2.2.5 File-of-filespec and File-parameter

The File-of-filespec parameter is encoded as PATH which points to a file containing Filespec-parameters, which contain the File-parameter. The File-parameter itself is also encoded as PATH, which points to the file transferred.

Syntax:

```
<File-of-filespec>  :=          PATH
                                --  The path points to a file containing one or more
                                --  <Filespec-parameter>s

<Filespec-parameter>  :=        <File-parameter> "," <File-conv> ["," <File-type>]

<File-parameter>  :=            PATH
                                --  Path to the file transferred

<File-conv>  :=                 <Convert-id-parameter>
                                --  Specifies the Transfer Format of the file.

<File-type>  :=                 <Type-id-parameter>
                                --  Specifies the Transmission Format of the file
```

14.2.2.6 Sub-address-parameter

The Sub-address-parameter is encoded as NUMERIC-STRING that represents a sub-address.

Syntax:

<Sub-address-parameter> := NUMERIC-STRING

14.3 Additional Functionality

14.3.1 Function: Send and SendAck

See Tables 89 and 90.

TABLE 89/T.611

Additional functionality of the <SendTDD> for the Telex via Teletex service

Syntax Element	C	T	I/O	Text Based Encoding			Comment
				Keyword	Parameter	Default	
<Notify>	+	o	I	NOTIFY	<Boolean-parameter>	"Yes"	
<SubAddress>	+	o	I	SUBADDR	<Sub-address-parameter>	–	Originator’s sub-address
Send to one addressee							
<Recipient>	B	m	I	ADDRESS	<Address-parameter>	–	Specifies one recipient’s call number
Send to one or many addressees							
<RecipientSpec>	+	m	I	ADDRESS	"@" <File-of-addrspec>	–	Specifies a list of recipients
Send only one file							
<Convert>	B	m	I	CONVERT	<Convert-id-parameter>	–	Transfer format of the outgoing file
<Document>	B	m	I	FILENAME	<File-parameter>	–	Single file to be transmitted, delivered under the Transfer Format specified by the Convert keyword
<Type>	B	o	I	TYPE	<Type-id-parameter>	"STD"	Specifies the document type to be sent
<From>	+	o	I	FROM	NUMERIC-STRING	First page	The first page to be actually sent; does only apply to text files
<To>	+	o	I	TO	NUMERIC-STRING	Last page	The last page to be actually sent; does only apply to text files
Send one or many files							
<DocumentSpec>	+	m	I	FILENAME	"@" <File-of-filespec>	–	Specifies a list of files; uses a special syntax

TABLE 90/T.611

Additional functionality of the <SendAckTDD> for the Telex via Teletex service

Syntax Element	C	T	I/O	Text Based Encoding			Comment
				Keyword	Parameter	Default	
<Recipient>	B	m	I	ADDRESS	<Address-parameter>	–	Specifies one recipient's call number
<Notify>	+	o	I	NOTIFY	<Boolean-parameter>	"Yes"	
<SubAddress>	+	o	I	SUBADDR	<Sub-address-parameter>	–	Originator's sub-address
Send only one file							
<Convert>	B	m	I	CONVERT	<Convert-id-parameter>	–	Transfer format of the outgoing file
<Document>	B	m	I	FILENAME	<File-parameter>	–	Single file to be transmitted, delivered under the Transfer Format specified by the Convert keyword
<Type>	B	o	I	TYPE	<Type-id-parameter>	"STD"	Specifies the document type to be sent
<From>	+	o	I	FROM	NUMERIC-STRING	First page	The first page to be actually sent; does only apply to text files
<To>	+	o	I	TO	NUMERIC-STRING	Last page	The last page to be actually sent; does only apply to text files
Send one or many files							
<DocumentSpec>	+	m	I	FILENAME	"@" <File-of-filespec>	–	Specifies a list of files; uses a special syntax

14.3.2 Function: Receive

See Table 91.

14.4 CA-Descriptor Settings

A CA supporting the TX service shall specify the supported Type-id (document Transmission Formats) in the CA-Descriptor (see 9.5) Table 9.2 shows the possible assignments.

TABLE 91/T.611

Additional functionality of the <ReceiveTDD> for the Telex via Teletex service

Syntax Element	C	T	I/O	Text Based Encoding			Comment
				Keyword	Parameter	Default	
<Originator>	B	o	O	ADDRESS	<Address-parameter>	–	Specifies originator's phone number
<CvTx>	B	o	I	CVTX	<Convert-id-parameter>	"T.50"	Transfer format desired by the LA
<SubAddress>	+	o	I/O	SUBADDR	<Sub-address-parameter>	–	Recipient's sub-address; if specified in the Request, used as a selector for the retrieval
Only one file received							
<Convert>	B	m	O	CONVERT	<Convert-id-parameter>	–	States the Transfer Format of the received file
<Document>	B	m	I/O	FILENAME	<File-parameter>	–	The filename may be pre-set by the LA on request. If only one file is received, the name shall be retained by the CA. If many files are received the name may be overwritten by the CA
<Type>	B	o	O	TYPE	<Type-id-parameter>	–	Specifies the document type received
Many files received							
<DocumentSpec>	+	m	O	FILENAME	"@" <File-of-filespec>	–	Specifies a list of files; uses a special syntax

TABLE 92/T.611

Additional CA-Descriptor settings for the Telex via Teletex service

Keyword	Parameter	CA declares that
TX	"STD"	the basic service is supported

15 Service: Telex

15.1 Service Specific Syntax Elements

```
<ServiceDependentKeywordsSend>  :=
                                    (<Recipient> | <RecipientSpec>)
                                    ((<Document> <Convert> [<Type>] [From] [To]) | <DocumentSpec>)
                                    [<SubAddress>]

<ServiceDependentKeywordsSendAck>  :=
                                    <Recipient>
                                    ((<Document> <Convert> [<Type>] [From] [To]) | <DocumentSpec>)
                                    [<SubAddress>]

<ServiceDependentKeywordsReceive>  :=
                                    [<Originator>] ((<Document> <Convert> <Type>) | <DocumentSpec>)
                                    [<CvTlx>] [<SubAddress>]
```

See Table 93.

TABLE 93/T.611

Additional syntax elements for the Telex service

Syntax element	Purpose
<Convert>	Specifies the Transfer Format to be used
<CvTlx>	Specifies the desired Transfer Format for received files
<Document>, <DocumentSpec>	Specifies the document or the documents to be sent or being received
<From>	Transmission should start from the page number specified
<Originator>	Specifies the communications address of the originator
<Recipient>, <RecipientSpec>	Specifies the communications address(es) of the recipient(s)
<SubAddress>	Specifies the Originator’s sub-address
<To>	Transmission should finish on page number specified
<Type>	Specifies the Transmission Format used

15.2 Text Based Encoding

15.2.1 Mapping of Keywords

See Table 94.

15.2.2 Encoding of Parameters

See also 6.4.4 for the encoding of non service dependent parameters used.

15.2.2.1 Service-id-parameter

The Service-id-parameter is encoded as STRING set to the constant value "TLX".

Syntax:

```
<Service-id-parameter>  :=          "TLX"
```

TABLE 94/T.611

Text Based Encoding of additional syntax elements for the Telex service

(A ⏎ denotes the new line format effector)

Syntax Element	Keyword/Parameter Pair
<Convert>	"CONVERT" ":" <Convert-id-parameter> ⏎
<CvTlx>	"CVTLX" ":" <Convert-id-parameter> ⏎
<Document>	"FILENAME" ":" <File-parameter> ⏎
<DocumentSpec>	"FILENAME" ":" "@" <File-of-filespec> ⏎
<From>	"FROM" ":" NUMERIC-STRING ⏎
<Originator>	"ADDRESS" ":" <Address-parameter> ⏎
<Recipient>	"ADDRESS" ":" <Address-parameter> ⏎
<RecipientSpec>	"ADDRESS" ":" "@" <File-of-addrspec> ⏎
<SubAddress>	"SUBADDR" ":" <Sub-address-parameter> ⏎
<To>	"TO" ":" NUMERIC-STRING ⏎
<Type>	"TYPE" ":" <Type-id-parameter> ⏎

15.2.2.2 Type-id-parameter

The Type-id-parameter is encoded as STRING set to the constant value "STD".

Syntax:

<Type-id-parameter> := "STD"

NOTE – The Transmission Format STD is defined per ITU-T Recommendation S.1.

15.2.2.3 Convert-id-parameter

The Convert-id-parameter is encoded as STRING set to one of the following values:

"ASCII"	APPLI/COM Extended ASCII
"ASCII437"	APPLI/COM Extended ASCII
"T.50"	APPLI/COM Standard ASCII

Syntax:

<Convert-id-parameter> :=	<Ascii> | "T.50"
<Ascii> :=	"ASCII" | STRING ("ASCII" + Code-page>)
<Code-page> :=	<digit> <digit> <digit>
<digit> :=	"0" | ... | "9"

15.2.2.4 File-of-addrspec and Address-parameter

The File-of-addrspec parameter is encoded as PATH which points to a file containing Addrspec-parameters, which contain the Address-parameter.

The Address-parameter is encoded as STRING forming the call number. If a subscriber identification verification is to be performed, the answerback unit of the receiving terminal must be entered after the call number, separated by an "equal" sign ("=").

Alternatively an alias name may be given instead of the terminal identifier, provided the alias is introduced with a "&" character. It is assumed that the CA knows how to decode the alias specified.

NOTE – If it is intended to perform a subscriber identification verification, the complete answerback unit must be entered after the equal sign, and not just the alphabetic component of the receiver.

Syntax:

```
<File-of-addrspec>  :=          PATH
                                --  The path points to a file containing one ore more
                                --  <Addrspec-parameter>s

<Addrspec-parameter>  :=        <Address-parameter>

<Address-parameter>  :=         <Telex-address> | ("&" <Alias>)

<Telex-address>  :=             STRING
                                --  The string shall present a valid telex address

<Alias>  :=                     STRING
```

15.2.2.5 File-of-filespec and File-parameter

The File-of-filespec parameter is encoded as PATH which points to a file containing Filespec-parameters, which contain the File-parameter. The File-parameter itself is also encoded as PATH, which points to the file transferred.

Syntax:

```
<File-of-filespec>  :=          PATH
                                --  The path points to a file containing one ore more
                                --  <Filespec-parameter>s

<Filespec-parameter>  :=        <File-parameter> "," <File-conv> ["," <File-type>]

<File-parameter>  :=            PATH
                                --  Path to the file transferred

<File-conv>  :=                 <Convert-id-parameter>
                                --  Specifies the Transfer Format of the file

<File-type>  :=                 <Type-id-parameter>
                                --  Specifies the Transmission Format of the file
```

15.2.2.6 Sub-address-parameter

The Sub-address-parameter is encoded as NUMERIC-STRING that represents a sub-address.

Syntax:

```
<Sub-address-parameter>:=       NUMERIC-STRING
```

15.3 Additional Functionality

15.3.1 Function: Send and SendAck

See Table 95 and 96.

TABLE 95/T.611

Additional functionality of the <SendTDD> for the Telex service

Syntax Element	C	T	I/O	Text Based Encoding			Comment
				Keyword	Parameter	Default	
<SubAddress>	+	o	I	SUBADDR	<Sub-address-parameter>	–	Originator's sub-address
Send to one addressee							
<Recipient>	B	m	I	ADDRESS	<Address-parameter>	–	Specifies one recipient's call number
Set to one or many addressees							
<RecipientSpec>	+	m	I	ADDRESS	"@" <File-of-addrspec>	–	Specifies a list of recipients
Send only one file							
<Convert>	B	m	I	CONVERT	<Convert-id-parameter>	–	Transfer format of the outgoing file
<Document>	B	m	I	FILENAME	<File-parameter>	–	Single file to be transmitted, delivered under the Transfer Format specified by the Convert keyword
<Type>	B	o	I	TYPE	<Type-id-parameter>	"STD"	Specifies the document type to be sent
<From>	+	o	I	FROM	NUMERIC-STRING	First page	The first page to be actually sent; does only apply to text files
<To>	+	o	I	TO	NUMERIC-STRING	Last page	The last page to be actually sent; does only apply to text files
Send one or many files							
<DocumentSpec>	+	m	I	FILENAME	"@" <File-of-filespec>	–	Specifies a list of files; uses a special syntax

TABLE 96/T.611

Additional functionality of the <SendAckTDD> for the Telex via Teletex service

Syntax Element	C	T	I/O	Text Based Encoding			Comment
				Keyword	Parameter	Default	
<Recipient>	B	m	I	ADDRESS	<Address-parameter>	–	Specifies one recipient's call number
<SubAddress>	+	o	I	SUBADDR	<Sub-address-parameter>	–	Originator's sub-address
Send only one file							
<Convert>	B	m	I	CONVERT	<Convert-id-parameter>	–	Transfer format of the outgoing file
<Document>	B	m	I	FILENAME	<File-parameter>	–	Single file to be transmitted, delivered under the Transfer Format specified by the Convert keyword
<Type>	B	o	I	TYPE	<Type-id-parameter>	"STD"	Specifies the document type to be sent
<From>	+	o	I	FROM	NUMERIC-STRING	First page	The first page to be actually sent; does only apply to text files
<To>	+	o	I	TO	NUMERIC-STRING	Last page	The last page to be actually sent; does only apply to text files
Send one or many files							
<DocumentSpec>	+	m	I	FILENAME	"@" <File-of-filespec>	–	Specifies a list of files; uses a special syntax

15.3.2 Function: Receive

See Table 97.

TABLE 97/T.611

Additional functionality of the <ReceiveTDD> for the Telex via Teletex service

Syntax Element	C	T	I/O	Text Based Encoding			Comment
				Keyword	Parameter	Default	
<Originator>	B	o	O	ADDRESS	<Address-parameter>	–	Specifies orginator's phone number
<CvTlx>	B	o	I	CVTLX	<Convert-id-parameter>	"T.50"	Transfer format desired by the LA
<SubAddress>	+	o	I/O	SUBADDR	<Sub-address-parameter>	–	Recipient's sub-address; if specified in the Request, used as a selector for the retrieval
Only one file received							
<Convert>	B	m	O	CONVERT	<Convert-id-parameter>	–	States the Transfer Format of the received file
<Document>	B	m	I/O	FILENAME	<File-parameter>	–	The filename may be pre-set by the LA on request. If only one file is received, the name shall be retained by the CA. If many files are received the name may be overwritten by the CA
<Type>	B	o	O	TYPE	<Type-id-parameter>	–	Specifies the document type received
Many files received							
<DocumentSpec>	+	m	O	FILENAME	"@" <File-of-filespec>	–	Specifies a list of files; uses a special syntax

15.4 CA-Descriptor Settings

A CA supporting the TX service shall specify the supported Type-id (document Transmission Formats) in the CA-Descriptor (see 9.5) Table 98 shows the possible assignments.

TABLE 98/T.611

Additional CA-Descriptor settings for the Telex service

Keyword	Parameter	CA declares that
TX	"STD"	the basic service is supported

16 Service: E-Mail

This Recommendation allows sending or receiving information through E-Mail services. Full exhaustive control of E-Mail services through the interface is not intended.

The definitions made hereafter furthermore enable the access to the Message Handling System (MHS), as described in the ITU-T X.400-Series of Recommendations. In terms of MHS, the interface provides access to Interpersonal Messaging.

The implementation of the CA may rely on other E-Mail supporting interfaces as shown in Figure 15. The CA shown uses an underlying E-Mail interface to communicate with the E-Mail service provider. In general, the CA is free to use *any* E-Mail interface to access and control the E-Mail services.

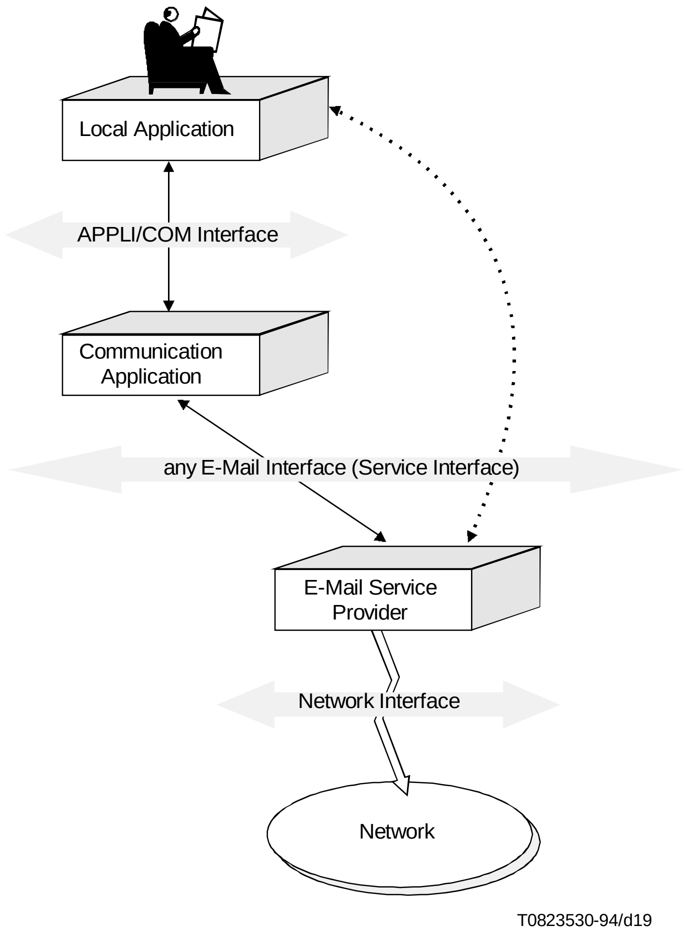


FIGURE 15/T.611

Model for E-Mail access

MHS and other E-Mail systems standardize the syntax and semantic of Interpersonal Messages (IPMs) as well as the syntax and semantic of Interpersonal Notifications (IPNs)[9] and Delivery Reports (DRs).

[9] For MHS, ITU-T Recommendation X.420 standardizes the syntax of IPMs and IPNs. In the version from 1984 of ITU-T Recommendation X.420 the notifications were called SR-UAPDU.

A recipient may create and send an IPN automatically or on request of the LA (explicit notification). The IPNs may either be:

- a Receipt Notification (RN): the IPM was received;
- a Non Receipt Notification (NRN): the IPM was not delivered; in this case, the undelivered IPM may be transmitted with the NRN.

Contrary to an IPN the DR is created by the message transfer system. The DRs are distinguished in:

- a Delivery Report (DR); or
- a Non-delivery Report (NDR).

In the Send and SendAck TDDs the LA may specify, which Interpersonal Notification (IPN) or Delivery Report (DR) shall be generated for each recipient. For more information see 16.2.2.6, 16.4 and 16.5.

MHS and other E-Mail systems allow complete IPMs - consisting of heading and body parts – to be transferred as body part in a message.

This Recommendation supports the transfer of such body parts in the Send (SendAck) and Receive TDDs through a specific Type-id-parameter (the Type-ID MESSAGE, see 16.2.2.2). Since the content of such a body part is encoded[10)], this Recommendation allows to decode and encode such a body part by use of the EncodeIPM and DecodeIPM TDDs.

In case of MHS the IPM is encoded according to ASN.1 BER.

The EncodeIPM and DecodeIPM TDDs shall be implemented using the <ExtendTDD>. The extensions are called:

- <EncodeIPMExtension>; and
- <DecodeIPMExtension> respectively.

The EncodeIPM and DecodeIPM TDDs are described in 16.3.3.

16.1 Service Specific Syntax Elements

Below the service dependent extensions of the generic TDD syntax referred to in 6.1 are shown:

```
<ServiceDependentKeywordsSend>  :=
                    <S-RecipientSpec> [<S-OriginatorSpec>]
                    ((<Document> <Convert> [<Type>]) | <DocumentSpec>)
                    <IpmId> [<Alternate>] [<ContType>] [<DiscloRec>] [<ExpiryTime>]
                    [<ImplicitConv>] [<Importance>] [<Language>] [<Priority>] [<RelatedSpec>]
                    [<ReplyId>] [<ReplyTime>] [<Sensitivity>] [<Subject>] [<UserInfo>]

<ServiceDependentKeywordsSendAck>  :=
                    <S-RecipientSpec> [<S-OriginatorSpec>]
                    ((<Document> <Convert> [<Type>]) | <DocumentSpec>)
                    <IpmId> [<Alternate>] [<ContType>] [<DiscloRec>] [<ExpiryTime>]
                    [<ImplicitConv>] [<Importance>] [<Language>] [<Priority>] [<RelatedSpec>]
                    [<ReplyId>] [<ReplyTime>] [<Sensitivity>] [<Subject>] [<UserInfo>]
                    [<MsgSubId>] [<SubmitTime>]

<ServiceDependentKeywordsReceive>  :=
                    <R-RecipientSpec> [<R-OriginatorSpec>]
                    ((<Document> <Convert> <Type>) | <DocumentSpec>)
                    <IpmId> [<RelatedSpec>] [<ContType>] [<ExpiryTime>] [<Forwarded>]
                    [<Importance>] [<IncompleteCopy>] [<Language>] [<Priority>] [<ReplyId>]
                    [<ReplyTime>] [<Sensitivity>] [<Subject>] [<SubmitTime>] [<UserInfo>]
```

10) In case of MHS the IPM is encoded according to ASN.1 BER.

<ExtendSubFunctionKeywords> :=	
	<EncodeIPMExtension> | <DecodeIPMExtension>
<EncodeIPMExtension> :=	<EncodeIPMSubFunction> ((<Document> <Convert> [<Type>]) | <DocumentSpec>) <Message> <S-RecipientSpec> [<S-OriginatorSpec>] <IpmId> [<ContType>] [<ExpiryTime>] [<Importance>] [<Priority>] [<RelatedSpec>] [<ReplyId>] [<ReplyTime>] <S-Originator> [<Sensitivity>] [<Subject>] [<UserInfo>]
<DecodeIPMExtension> :=	<DecodeIPMSubFunction> <Message> ((<Document> <Convert> <Type>) | <DocumentSpec>) <S-RecipientSpec> [<S-OriginatorSpec>] <IpmId> [<ContType>] [<ExpiryTime>] [<Forwarded>] [<Importance>] [<Language>] [<Priority>] [<RelatedSpec>] [<ReplyId>] [<ReplyTime>] [<Sensitivity>] [<Subject>] [<UserInfo>]

Table 99 describes the additional syntax elements for the E-Mail services. For a mapping of MHS service elements to the syntax elements used in this Recommendation refer to Table 112 in 16.6.

16.2 Text Based Encoding

16.2.1 Mapping of Keywords

See Table 100.

16.2.2 Encoding of Parameters

See also section 6.4.4 for the encoding of non service dependent parameters used.

16.2.2.1 Service-id-parameter

The Service-id parameter is encoded as STRING set to the constant value "EMAIL".

Syntax:

<Service-id-parameter> :=	"EMAIL"

16.2.2.2 Type-id-parameter

The Type-id-parameter is encoded as STRING set to one of the following values:

"STD"	IA5 Text body part
"TELETEX"	Teletex body part
"G3FAX"	Telefax Group 3 body part
"G4CLASS1"	Telefax Group 4 body part
"VIDEOTEX"	Videotex body part
"MESSAGE"	Body part contains an IPM (consisting of heading and bodyparts) as transferred by the E-Mail system
"BILATERAL"	Bilateral defined body part contents
"NATIONAL"	National defined body part contents
"ODA"	ODA body part

Syntax:

<Type-id-parameter> :=	"STD" | "TELETEX" | "G3FAX" | "G4CLASS1" | "VIDEOTEX" | "MESSAGE" | "BILATERAL" | "NATIONAL" | "ODA"[11].

11) For MHS ITU-T Recommendation X.420 defines a set of parameters related to each body part. These parameters are transferred by using the Filespec-parameter (see section 16.2.2.8).

TABLE 99/T.611

Additional syntax elements for E-Mail services

Syntax element	Purpose
<Alternate>	Specifies whether alternate recipient(s) are allowed
<ContType>	Specifies the content type of the message exchanged
<Convert>	Specifies the Transfer Format to be used
<DecodeIPMSubFunction>	Identifies the DecodeIPM Subfunction
<DiscloRec>	Recipients disclosure
<Document>, <DocumentSpec>	Specifies the document or the documents to be sent or being received (or to be exchanged with the EncodeIPM and DecodeIPM subfunctions)
<EncodeIPMSubFunction>	Identifies the EncodeIPM Subfunction
<ExpiryTime>	Date/Time of expiration of the message (Expiry Time)
<Forwarded>	Autoforwarded indication
<ImplicitConv>	Implicit conversion
<Importance>	Importance of the message contents
<IncompleteCopy>	Reflects the MHS incomplete copy indication
<IpmId>	Message ID
<Language>	Identification of the language used
<Message>	Path to the ASN.1 BER coded message of the EncodeIPM and DecodeIPM subfunctions
<MsgSubId>	Message submission identifier, returned by the E-Mail service provider after it has accepted the request
<Priority>	Priority of the message contents
<R-OriginatorSpec>, <S-OriginatorSpec>	Path to the file containing the address specifications of the originator, authorizing user(s) and reply-to user(s) of the message
<R-RecipientSpec>, <S-RecipientSpec>	Path to the file containing the address specifications of all intended recipients of the message, i.e. primary, copy and blind copy recipient(s)
<RelatedSpec>	Path name of the file containing the Relatedspecs of all messages related to the present one
<ReplyId>	ID of the message, for which this one is a response
<ReplyTime>	Date-Time by when the recipient(s) of the message should reply to the authorizing users (Reply Time)
<Sensitivity>	Sensitivity of the message contents
<Subject>	Subject the message is referring to
<SubmitTime>	Date-Time at which the E-Mail service provider processed the request (Message Submission Time)
<Type>	Specifies the Transmission Format used
<UserInfo>	User-provided optional envelope identifier that is forwarded along with the message. Can be used for user-specific identification purposes

TABLE 100/T.611

Text Based Encoding of additional syntax elements for E-Mail services

(A ⏎ stands for new line)

Syntax Element	Keyword/Paramete Pair
<Alternate>	"ALTERNATE" ":" <Boolean-parameter> ⏎
<ContType>	"CONT-TYPE" ":" <Content-type-parameter> ⏎
<Convert>	"CONVERT" ":" <Convert-id-parameter> ⏎
<DecodeIPMSubFunction>	"SUBFUNC" ":" "DecodeIPM" ⏎
<DiscloRec>	"DISCLO-REC" ":" <Boolean-parameter> ⏎
<Document>	"FILENAME" ":" <File-parameter> ⏎
<DocumentSpec>	"FILENAME" ":" "@" <File-of-filespec> ⏎
<EncodeIPMSubFunction>	"SUBFUNC" ":" "EncodeIPM" ⏎
<ExpiryTime>	"EXPIRYTIME" ":" <Date-time-parameter> ⏎
<Forwarded>	"FORWARDED" ":" <Boolean-parameter> ⏎
<ImplicitConv>	"IMPLICIT-CONV" ":" <Boolean-parameter> ⏎
<Importance>	"IMPORTANCE" ":" <Importance-parameter> ⏎
<IncompleteCopy>	"INC-COPY" ":" <Boolean-parameter> ⏎
<IpmId>	"IPM-ID" ":" <Ipm-id-parameter> ⏎
<Language>	"LANGUAGE" ":" <Language-id-parameter> ⏎
<Message>	"MESSAGE" ":" <Path-parameter> ⏎
<MsgSubId>	"MSG-SUB-ID" ":" <Msg-sub-id-parameter> ⏎
<Priority>	"PRIORITY" ":" <Priority-parameter> ⏎
<R-OriginatorSpec>	"ADDRESS" ":" "@" <File-of-originatorspec> ⏎
<R-RecipientSpec>	"RECIPIENT" ":" "@" <File-of-r-recipientspec> ⏎
<RelatedSpec>	"RELATED" ":" "@" <File-of-relatedspec> ⏎
<ReplyId>	"REPLYID" ":" <Ipm-id-parameter> ⏎
<ReplyTime>	"REPLYTIME" ":" <Date-time-parameter> ⏎
<S-OriginatorSpec>	"ORIGINATOR" ":" "@" <File-of-originatorspec> ⏎
<S-RecipientSpec>	"ADDRESS" ":" "@" <File-of-s-recipientspec> ⏎
<Sensitivity>	"SENSITIVITY" ":" <Sensitivity-parameter> ⏎
<Subject>	"SUBJECT" ":" <Subject-parameter> ⏎
<SubmitTime>	"SUBMITTIME" ":" <Date-time-parameter> ⏎
<Type>	"TYPE" ":" <Type-id-parameter> ⏎
<UserInfo>	"USERINFO" ":" <Userinfo-parameter> ⏎

16.2.2.3 Convert-id-parameter

The Convert-id-parameter is encoded as STRING set to one of the following values:

"ASCII"	APPLI/COM Extended ASCII
"ASCII437"	APPLI/COM Extended ASCII
"T.50"	APPLI/COM Standard ASCII
"T.61"	APPLI/COM Teletex format
"TIFF"	APPLI/COM TIFF
"VOID"	No conversion to be done
"PROBE"	Specifies an empty document (i.e. no document at all) for delivery testing procedures

The use of the Convert-id-parameter also depends on the Type-id selected. Table 101 depicts this.

TABLE 101/T.611

Permitted Convert-id assignements dependent on Type-id

Type-id	Convert-id for outgoing traffic	Convert-id for incoming traffic
STD	T.50	
TELETEX	ASCII, ASCIIxxx[a)], T.50, T.61	
G3FAX, G4CLASS1	ASCII, ASCIIxxx, T.50, TIFF	TIFF
VIDEOTEX, MESSAGE, BILATERAL, NATIONAL, ODA	VOID	
a) xxx stands for a code-page, declared in the ICE, e.g. ASCII437 if code-page 437 has been declared.		

Syntax:

<Convert-id-parameter> :=	<Convert-probe> | <Convert-std> | <Convert-ttx> | <Convert-img> | <Convert-bin>
<Convert-probe> :=	"PROBE" -- *Special mode, where no documents are transmitted; can only be sent*
<Convert-std> :=	"T.50"
<Convert-ttx> :=	<Ascii> | "T.50" | "T.61"
<Convert-img> :=	<Ascii> | "T.50" | "TIFF"
<Convert-bin> :=	"VOID"
<Ascii> :=	"ASCII" | STRING ("ASCII" + Code-page>)
<Code-page> :=	<digit> <digit> <digit>
<digit> :=	"0" | ... | "9"

16.2.2.4 MHS Addresses

MHS Addresses are referenced by the Originatorspec-parameter, R-Recipientspec-parameter and S-Recipientspec-parameter or by the Ipm-id-parameter (see 16.2.2.5, 16.2.2.6 and 16.2.2.11).

The syntax of the MHS Addresses, namely Mhs-or-descriptor, Mhs-or-name, Mhs-dl-name and Mhs-or-address, is specified below.

The MHS-or-address is encoded as STRING representing a compilation of address elements, each specifying a part of the address.

An address element is composed of an address field identifier, followed by a separator ("=" sign) and the address field attribute.

Table 102 below lists the address field identifiers predefined for MHS services within this Recommendation$^{12)}$.

NOTE – The addressing scheme of Table 102 can also be used for other E-Mail environments if appropriate. The creation of additional, not predefined address elements is possible using the formal BNF-style syntax description of the <Private-addr-element> below.

TABLE 102/T.611

Address field identifiers and attributes

Address field identifier	Address field attribute	Maximum attribute field length
C	CountryName	3
A	AdministrationDomainName	16
P	PrivateDomainName	16
O	OrganizationName	64
OUn (0 ≤ n ≤ 4)	OrganizationUnitName	32
Q	GenerationQualifier	3
S	Surname	40
G	GivenName	16
I	Initials	5
X121	X.121Address	16
T-ID	TerminalIdentifier	24
PD-BOX	PostalOfficeBoxAddress	60

Syntax:

```
-- Syntax of <Mhs-or-address> (referenced by other parameter syntaxes)

<Mhs-or-address>  :=
                 "/" <Mandatory-address-element> {"/" <Mandatory-address-element>}
                 {"/" <Optional-address-element>} {"/" <Private-addr-element>}
                 --  for use with MHS systems
                 --  at least one <Mandatory-address-element> shall be present

<Mandatory-address-element>  :=
                 ("C" "=" STRING (SIZE(1..3))) |          --  CountryName
                 ("A" "=" STRING (SIZE(1..16))) |         --  AdministrationDomainName
                 ("P" "=" STRING (SIZE(1..16))) |         --  PrivateDomainName
                 ("O" "=" STRING (SIZE(1..64))) |         --  OrganizationName
                 --  See also Table 102 for assignments

<Optional-address-element>  :=
                 ("OU"<n> "=" STRING (SIZE(1..32))) |     --  OrganizsationUnitName
                 ("Q" "=" STRING (SIZE(1..3))) |          --  GenerationQualifier
                 ("S" "=" STRING (SIZE(1..40))) |         --  Surname
```

12) Each address field identifier addresses an O/R-Name attribute as specified in ITU-T Recommendation X.402.

```
                            ("G" "=" STRING (SIZE(1..16))) |            -- GivenName
                            ("I" "=" STRING (SIZE(1..5))) |             -- Initials
                            ("X121" "=" STRING (SIZE(1..16))) |         -- X.121 Address
                            ("T-ID" "=" STRING (SIZE(1..24))) |         -- Terminal identifier
                            ("PD-BOX" "=" STRING (SIZE(1..60))) |       -- PostalOfficeBoxAddress
                            -- See also Table 102 for assignments

<n> :=                      "0" | ... | "4"

<Private-addr-element> :=   <Field-identifier> "=" <Field-attribute>
                            -- private or additional defined address element

-- Syntax of <Mhs-or-descriptor> (referenced by other parameter syntaxes)

<Mhs-or-descriptor> :=      (<Mhs-or-name> ["+" [<Mhs-free-name>] ["+" <Mhs-phone-number>]]) |
                            ("+" <Mhs-free-name> ["+" <Mhs-phone-number>])

<Mhs-free-name> :=          STRING (SIZE(1..64))

<Mhs-phone-number> :=       STRING (SIZE(1..32))

-- Syntax of <Mhs-or-name> (referenced by other parameter syntaxes)

<Mhs-or-name> :=            (<Mhs-or-address> ["+" <Mhs-dl-name>]) | ("+" <Mhs-dl-name>)

<Mhs-dl-name> :=            STRING (SIZE(1..64))
                            -- stands for Distribution List name according to ITU-T
                            -- Recommendation X.402
```

NOTE – The <Mhs-dl-name> is different from the <Alias> used in some syntaxes, since the alias is expanded locally (in the CA) whereas the <Mhs-dl-name> is expanded by the MTS.

16.2.2.5 File-of-originatorspec

The File-of-originatorspec construction specifies the originators of the message. It is encoded as PATH. The path given shall address a file containing one ore more Originatorspec-parameter(s).

Syntax:

```
<File-of-originatorspec> := PATH
                            -- The path points to a file containing one ore more
                            -- <Originatorspec-parameter>s

<Originatorspec-parameter> :=
                            <Addr-descriptor> | <Mhs-or-descriptor> "," <O-type>
                            -- for definition of <Mhs-or-descriptor> see 16.2.2.4

<Addr-descriptor> :=        STRING
                            -- Contents as required by the underlying (non-MHS) E-Mail system;
                            -- shall not contain "," characters

<O-type> :=                 "Authorizing" | "Originator" | "ReplyTo"
```

16.2.2.6 File-of-r-recipientspec and File-of-s-recipientspec

The File-of-r-recipientspec and File-of-s-recipientspec constructions specify the recipients of the message. They are encoded as PATH. The path given shall address a file containing one ore more R-recipientspec-parameter(s) or S-recipientspec-parameter(s) respectively.

Syntax:

```
<File-of-r-recipientspec> := PATH
                            -- The path points to a file containing one ore more
                            -- <R-recipientspec-parameter>s
```

```
<File-of-s-recipientspec>  :=      PATH
                                    --  The path points to a file containing one ore more
                                    --  <S-recipientspec-parameter>s

<R-recipientspec-parameter>  :=
                                    <Addr-descriptor> | <Mhs-or-descriptor>
                                    "," <R-type> ["," [<Reply>] ["," [<Notify>] ["," <Report>] ] ]
                                    --  Receiving side
                                    --  for definition of the <Mhs-or-descriptor> see 16.2.2.4

<S-recipientspec-parameter>  :=
                                    ("&" <Alias>) | <Addr-descriptor> | <Mhs-or-descriptor>
                                    "," <S-type> ["," [<Reply>] ["," [<Notify>] ["," <Report>] ] ]
                                    --  Sending side may specify an (local) alias instead of a descriptor
                                    --  for definition of the <Mhs-or-descriptor> see 16.2.2.4

<Alias>  :=                         STRING
                                    --  The Alias shall not contain "," or ";" characters!

<Addr-descriptor>  :=               STRING
                                    --  Contents as required by underlying (non-MHS) E-Mail system;
                                    --  shall not contain "," characters

<R-type>  :=                        "Primary" | "Copy" | "Blind" | "Intended"

<S-type>  :=                        "Primary" | "Copy" | "Blind"

<Reply>  :=                         "NoReply" | "Reply"
                                    --  NoReply is taken as default if not specified

<Notify>  :=                        "NoNotify" | "NotifyNotReceived" | "NotifyReceived" | "NotifyExplicit"
                                    --  NoNotify is taken as default if not specified

<Report>  :=                        "NoReport" | "BasicReport" | "ConfirmedReport"13)
                                    --  NoReport is taken as default if not specified
```

16.2.2.7 File-of-relatedspec

The File-of-relatedspec construction specifies IPM-IDs a message is related to. It is encoded as PATH. The path given shall address a file containing one ore more Relatedspec-parameter(s).

Syntax:

```
<File-of-relatedspec>  :=           PATH
                                    --  The path points to a file containing one ore more
                                    --  <Relatedspec-parameter>s

<Relatedspec-parameter>  :=         <Ipm-Id-parameter> "," <Relation>
                                    --  See 16.2.2.11 for definition of the Ipm-id-parameter

<Relation>  :=                      "Reference" | "Obsolete"
```

16.2.2.8 File-of-filespec and File-parameter

The File-of-filespec parameter is encoded as PATH which points to a file containing Filespec-parameters, which contain the File-parameter. The File-parameter itself is also encoded as PATH, which points to the file transferred.

If the special Convert-id-parameter "PROBE" is specified, the File-parameter may be omitted (see also 16.2.2.3).

[13] In case of MHS, "BasicReport" maps to non-delivery-report and "ConfirmedReport" maps to report of ITU-T Recommendation X.411.

Syntax:

<File-of-filespec> :=	PATH -- *The path points to a file containing one ore more* -- *<Filespec-parameter>s*
<Filespec-parameter> :=	<File-parameter> "," <File-conv> [["," <File-type>] ["," <Body-part-parameter>]]
<File-parameter> :=	PATH -- *Path to the file transferred*
<File-conv> :=	<Convert-id-parameter> -- *Specifies the Transfer Format of the file*
<File-type> :=	<Type-id-parameter> -- *Specifies the Transmission Format of the file*
<Body-part-parameter> :=	STRING -- *Syntax left open for further study*

16.2.2.9 Content-type-parameter

The Content-type-parameter is encoded as STRING set to one of the following values:

"mhsIPM84"	Interpersonal Message (IPM) according to P2 protocol as defined per ITU-T Recommendation X.420 (1984)
"mhsIPM88"	Interpersonal Message (IPM) according to P2 protocol as defined per ITU-T Recommendation X.420 (1988)
"mhsIPN84"	Interpersonal Notification (IPN) according to P2 protocol as defined per ITU-T Recommendation X.420 (1984)
"mhsIPN88"	Interpersonal Notification (IPN) according to P2 protocol as defined per ITU-T Recommendation X.420 (1988)
"mhsDR"	Delivery Report (DR) as defined per ITU-T Recommendation X.420
"cmcIPM"	Reserved for Interpersonal Messages (IPM) conforming to the Common Messaging Call E-Mail interface
"cmcIPN"	Reserved for Interpersonal Notifications (IPN) conforming to the Common Messaging Call E-Mail interface
"cmcDR"	Reserved for Delivery Reports (DR) conforming to the Common Messaging Call E-Mail interface
any other	Reserved for further study

Syntax:

<Content-type-parameter> :=	"mhsIPM84" | "mhsIPM88" | "mhsIPN84" | "mhsIPN88" | "mhsDR" | "cmcIPM" | "cmcIPN" | "cmcDR"

16.2.2.10 Importance-parameter

The Importance-parameter is encoded as NUMERIC-STRING set to one of the following values:

"0"	low
"1"	normal (default)
"2"	high

Syntax:

<Importance-parameter> :=	"0" | ... | "2"

16.2.2.11 Ipm-id-parameter

The Ipm-id-parameter is encoded as STRING.

In MHS based systems it is a construct of two components:

- a unique, user relative identifier (a local identifier);
- an optional O/R Address which identifies the user.

Syntax:

```
<Ipm-id-parameter>  :=          <Free-ipm-id> | <Mhs-ipm-id>
```

-- *for definition of the <Mhs-or-descriptor> see 16.2.2.4*

```
<Free-ipm-id>  :=               STRING
                                --  Contents as required by underlying (non-MHS) E-Mail system
                                --  The string shall not contain "," or ";" characters

<Mhs-ipm-id>  :=                <Local-ipm-id> ["," <Mhs-or-address>]
                                --  for use in MHS systems
                                --  for definition of the <Mhs-or-address> see 16.2.2.4

<Local-ipm-id>  :=              STRING (SIZE(1..64))
                                --  The string shall not contain "," or ";" characters
```

16.2.2.12 Language-id-parameter

The Language-id-parameter is encoded as STRING between 2 and 5 characters long. For assignment of the language code for MHS refer to ISO 639.2

Syntax:

```
<Language-id-parameter>  :=     STRING (SIZE(2..5))
```

16.2.2.13 Msg-sub-id-parameter

The Msg-sub-id-parameter is encoded as STRING. The Msg-sub-id-parameter identifies uniquely and unambiguously the message submission. It is generated by the E-Mail system[14].

Syntax:

```
<Msg-sub-id-parameter>  :=      <Free-sub-id> | <Mhs-sub-id>

<Free-sub-id>  :=               STRING
                                --  For use in non-MHS systems

<Mhs-sub-id>  :=                <Global-id> "," <Local-id>
                                --  For use in MHS systems

<Global-id>  :=                  "/" "C" "=" STRING (SIZE(1..3))       --  CountryName
                                 "/" "A" "=" STRING (SIZE(1..16))      --  AdministrationDomainName
                                ["/" "P" "=" STRING (SIZE(1..16))]     --  PrivateDomainName
                                --  see also definition of <Mhs-or-address>, 16.2.2.4

<Local-id>  :=                  STRING (SIZE(1..32))
```

16.2.2.14 Priority-parameter

The Priority-parameter is encoded as NUMERIC-STRING set to one of the following values:

"0"	normal (default)
"1"	non-urgent
"2"	urgent

[14] In case of MHS, the Msg-sub-id-parameter corresponds to the MPDUIndentifier (ITU-T Recommendation X.411 of 1984), renamed MTSIdentifier in the 1988 version of the ITU-T Recommendation X.411.

Syntax:

<Priority-parameter> := "0" | "1" | "2"

16.2.2.15 Subject-parameter

The Subject-parameter is encoded as STRING.

Syntax:

<Subject-parameter> := STRING (SIZE(1..128))

16.2.2.16 Sensitivity-parameter

The Sensitivity-parameter is encoded as NUMERIC-STRING set to one of the following values:

"0"	no sensitivity specified (default)
"1"	personal
"2"	private
"3"	company confidential

Syntax:

<Sensitivity-parameter> := "0" | ... | "3"

16.2.2.17 Userinfo-parameter

The Userinfo-parameter is encoded as STRING.

Syntax:

<Userinfo-parameter> := STRING (SIZE(1..16))

16.3 Interpersonal Messaging

16.3.1 Function: Send and SendAck

The Response TDD of the SendAck function shall be generated by the CA as soon as the document (the message) has been submitted to the E-Mail provider of the originator and the E-Mail provider has returned the <MsgSubId> to the CA. In other words: the submission is acknowledged, not the transmission of the message. See Tables 103 and 104.

16.3.2 Function: Receive

See Table 105.

16.3.3 Function: EncodeIPM and DecodeIPM

The EncodeIPM and DecodeIPM TDDs allow an LA to encode or decode a complete IPM - containing heading and body parts - into/from a "MESSAGE" type body part (see 16.2.2.2)

The encoding/decoding procedure requires that the LA provides a TDD containing syntax elements referring to the heading of the IPM to be encoded/decoded as well as syntax elements launching the encode/decode operation itself. Table 106 shows the EncodeIPM TDD extensions to the <ExtendTDD>, Table 107 shows the DecodeIPM TDD extensions.

TABLE 103/T.611

Additional functionality of the <SendTDD> for access to E-Mail services

Syntax Element	C	T	I/O	Text Based Encoding			Comment
				Keyword	Parameter	Default	
<IpmId>	B	m	I	IPM-ID	<Ipm-id-parameter>	–	Message identifier
<S-RecipientSpec>	B	m	I	ADDRESS	"@" <File-of-s-recipientspec>	–	Path of the file containing the Recipientspecs of all intended recipients
<S-OriginatorSpec>	B	m	I	ORIGINATOR	"@" <File-of-originatorspec>	–	Path name of the file containing the originator and authorising users
<Alternate>	B	o	I	ALTERNATE	<Boolean-parameter>	"No"	Alternate recipient(s) allowed
<ContType>	B	o	I	CONT-TYPE	<Content-type-parameter>	"mhsIPM84"	Content Type
<DiscloRec>	B	o	I	DISCLO-REC	<Boolean-parameter>	"No"	Recipient(s) disclosure
<ExpiryTime>	B	o	I	EXPIRYTIME	<Date-time-parameter>	–	Expiration of the message (Expiry Time)
<ImplicitConv>	B	o	I	IMPLICIT-CONV	<Boolean-parameter>	"Yes"	Implicit conversion
<Importance>	B	o	I	IMPORTANCE	<Importance-parameter>	"1"	Importance of the contents
<Language>	B	o	I	LANGUAGE	<Language-id-parameter>	–	Identification of the language used
<Priority>	B	o	I	PRIORITY	<Priority-parameter>	"0"	Priority of the contents

TABLE 103/T.611 *(end)*

Additional functionality of the <SendTDD> for access to E-Mail services

Syntax Element	C	T	I/O	Text Based Encoding			Comment
				Keyword	Parameter	Default	
<RelatedSpec>	B	o	I	RELATED	"@" <File-of-Relatedspec>	–	Path name of the file containing the Relatedspecs of all messages related to the present one
<ReplyId>	B	o	I	REPLYID	<Ipm-id-parameter>	–	ID of the message, for which this one is a response
<ReplyTime>	B	o	I	REPLYTIME	<Date-time-parameter>	–	Date-Time by when the recipient(s) of the message should reply to the authorizing users (Reply Time)
<Sensitivity>	B	o	I	SENSITIVITY	<Sensitivity-parameter>	"0"	Sensitivity of the contents
<Subject>	B	o	I	SUBJECT	<Subject-parameter>	–	
<Userinfo>	+	o	I	USERINFO	<Userinfo-parameter>	–	User-provided optional envelope identifier
Send only one file							
<Convert>	B	m	I	CONVERT	<Convert-id-parameter>	–	Transfer format of the outgoing file
<Document>	B	m	I	FILENAME	<File-parameter>	–	Single file to be transmitted, delivered under the Transfer Format specified by the Convert keyword
<Type>	B	o	I	TYPE	<Type-id-parameter>	"STD"	Specifies the document type to be sent
Send one or many files							
<DocumentSpec>	+	m	I	FILENAME	"@" <File-of-filespec>	–	Specifies a list of files; uses a special syntax

TABLE 104/T.611

Additional functionality of the <SendAckTDD> for access to E-Mail services

Syntax Element	C	T	I/O	Text Based Encoding			Comment
				Keyword	Parameter	Default	
<IpmId>	B	m	I	IPM-ID	<Ipm-id-parameter>	–	Message identifier
<S-RecipientSpec>	B	m	I	ADDRESS	"@" <File-of-s-recipientspec>	–	Path of the file containing the Recipientspecs of all intended recipients
<S-OriginatorSpec>	B	m	I	ORIGINATOR	"@" <File-of-originatorspec>	–	Path name of the file containing the originator and authorizing users
<Alternate>	B	o	I	ALTERNATE	<Boolean-parameter>	"No"	Alternate recipient(s) allowed
<ContType>	B	o	I	CONT-TYPE	<Content-type-parameter>	"mhsIPM84"	Content Type
<DiscloRec>	B	o	I	DISCLO-REC	<Boolean-parameter>	"No"	Recipient(s) disclosure
<ExpiryTime>	B	o	I	EXPIRYTIME	<Date-time-parameter>	–	Expiration of the message (Expiry Time)
<ImplicitConv>	B	o	I	IMPLICIT-CONV	<Boolean-parameter>	"Yes"	Implicit conversion
<Importance>	B	o	I	IMPORTANCE	<Importance-parameter>	"1"	Importance of the contents
<Language>	B	o	I	LANGUAGE	<Language-id-parameter>	–	Identification of the language used
<MsgSubId>	B	o	O	MSG-SUB-ID	<Msg-sub-id-parameter>	–	Message submission identifier. Returned by the E-Mail service provider after it has accepted the request
<Priority>	B	o	I	PRIORITY	<Priority-parameter>	"0"	Priority of the contents
<RelatedSpec>	B	o	I	RELATED	"@" <File-of-Relatedspec>	–	Path name of the file containing the Relatedspecs of all messages related to the present one
<ReplyId>	B	o	I	REPLYID	<Ipm-id-parameter>	–	ID of the message, for which this one is a response

TABLE 104/T.611 *(end)*

Additional functionality of the <SendAckTDD> for access to E-Mail services

Syntax Element	C	T	I/O	Text Based Encoding			Comment
				Keyword	Parameter	Default	
<ReplyTime>	B	o	I	REPLYTIME	<Date-time-parameter>	–	Date-Time by when the recipient(s) of the message should reply to the authorizing users (Reply Time)
<Sensitivity>	B	o	I	SENSITIVITY	<Sensitivity-parameter>	"0"	Sensitivity of the contents
<Subject>	B	o	I	SUBJECT	<Subject-parameter>	–	
<SubmitTime>	B	o	O	SUBMITTIME	<Date-time-parameter>	–	Date-time at which the request is processed by the E-Mail service provider
<Userinfo>	+	o	I	USERINFO	<Userinfo-parameter>	–	User-provided optional envelope identifier
Send only one file							
<Convert>	B	m	I	CONVERT	<Convert-id-parameter>	–	Transfer format of the outgoing file
<Document>	B	m	I	FILENAME	<File-parameter>	–	Single file to be transmitted, delivered under the Transfer Format specified by the Convert keyword.
<Type>	B	o	I	TYPE	<Type-id-parameter>	"STD"	Specifies the document type to be sent
Send one or many files							
<DocumentSpec>	+	m	I	FILENAME	"@" <File-of-filespec>	–	Specifies a list of files; uses a special syntax

TABLE 105/T.611

Additional functionality of the <ReceiveTDD> for access to E-Mail services

Syntax Element	C	T	I/O	Text Based Encoding			Comment
				Keyword	Parameter	Default	
<IpmId>	B	m	O	IPM-ID	<Ipm-id-parameter>	–	Message identifier
<R-RecipientSpec>	B	m	I	RECIPIENT	"@" <File-of-r-recipientspec>	–	Path of the file containing the Recipientspecs of all intended recipients
<R-OriginatorSpec>	B	m	I	ADDRESS	"@" <File-of-originatorspec>	–	Path name of the file containing the originator and authorizing users
<ContType>	B	o	O	CONT-TYPE	<Content-type-parameter>	–	Content Type
<ExpiryTime>	B	o	O	EXPIRYTIME	<Date-time-parameter>	–	Expiration of the message (Expiry Time)
<Forwarded>	B	o	O	FORWARDED	<Boolean-parameter>	–	Autoforwarded indication
<Importance>	B	o	O	IMPORTANCE	<Importance-parameter>	–	Importance of the contents
<Language>	B	o	O	LANGUAGE	<Language-id-parameter>	–	Identification of the language used
<Priority>	B	o	O	PRIORITY	<Priority-parameter>	–	Priority of the contents
<RelatedSpec>	B	o	I	RELATED	"@" <File-of-relatedspec>	–	Path name of the file receiving the Relatedspecs of all messages related to the present one
<ReplyId>	B	o	O	REPLYID	<Ipm-id-parameter>	–	ID of the message, for which this one is a response
<ReplyTime>	B	o	O	REPLYTIME	<Date-time-parameter>	–	Date-Time by when the recipient(s) of the message should reply to the authorizing users (Reply Time)
<Sensitivity>	B	o	O	SENSITIVITY	<Sensitivity-parameter>	"0"	Sensitivity of the contents
<Subject>	B	o	O	SUBJECT	<Subject-parameter>	–	

TABLE 105/T.611 *(end)*

Additional functionality of the <ReceiveTDD> for access to E-Mail services

Syntax Element	C	T	I/O	Text Based Encoding			Comment
				Keyword	Parameter	Default	
<SubmitTime>	B	o	O	SUBMITTIME	<Date-time-parameter>	–	Date-time at which the request is processed by the E-Mail service provider
<IncCopy>	+	o	O	INC-COPY	<Boolean-parameter>	–	Incomplete copy indicator
<Userinfo>	+	o	O	USERINFO	<Userinfo-parameter>	–	User-provided optional envelope identifier
Only one file received							
<Convert>	B	m	O	CONVERT	<Convert-id-parameter>	–	States the Transfer Format of the received file
<Document>	B	m	I/O	FILENAME	<File-parameter>	–	The filename may be pre-set by the LA on request. If only one file is received, the name shall be retained by the CA. If many files are received the name may be overwritten by the CA
<Type>	B	o	O	TYPE	<Type-id-parameter>	–	Specifies the document type received
Many files received							
<DocumentSpec>	+	m	O	FILENAME	"@" <File-of-filespec>	–	Specifies a list of files; uses a special syntax

TABLE 106/T.611

Additional syntax elements for the <EncodeIPMExtension> of E-Mail services

Syntax Element	C	T	I/O	Text Based Encoding			Comment
				Keyword	Parameter	Default	
Syntax elements belonging to the encode procedure itself (job description)							
<EncodeIPM-SubFunction>	B	m	I	SUBFUNC	"EncodeIPM"	–	
<Message>	B	m	I	MESSAGE	<Path-parameter>	–-	Path to file that will receive the encoded IPM
<Minor>	+	o	O	MINOR	<Error-parameter>	–	
<Warning>	+	o	O	WARNING	<Error-parameter>	–	
Only one body part shall be encoded							
<Convert>	B	m	I	CONVERT	<Convert-id-parameter>	–	Transfer format of the body part to be encoded
<Document>	B	m	I	FILENAME	<File-parameter>	–	Single body part to be encoded, delivered under the Transfer Format specified by the Convert keyword
<Type>	B	o	I	TYPE	<Type-id-parameter>	"STD"	Specifies the document type to be encoded
One or many body parts shall be encoded							
<DocumentSpec>	+	m	I	FILENAME	"@" <File-of-filespec>	–	Specifies a list of files; uses a special syntax
Syntax elements holding the heading information for the IPM to be encoded							
<IpmId>	B	m	I	IPM-ID	<Ipm-id-parameter>	–	Message identifier
<S-RecipientSpec>	B	m	I	ADDRESS	"@" <File-of-s-recipientspec>	–	Path of the file containing the Recipientspecs of all intended recipients
<S-OriginatorSpec>	B	m	I	ORIGINATOR	"@" <File-of-originatorspec>	–	Path name of the file containing the originator and authorizing users
<ContType>	B	o	I	CONT-TYPE	<Content-type-parameter>	"mhsIPM84"	Content Type
<ExpiryTime>	B	o	I	EXPIRYTIME	<Date-time-parameter>	–	Expiration of the message

TABLE 106/T.611 *(end)*

Additional syntax elements for the <EncodeIPMExtension> of E-Mail services

Syntax Element	C	T	I/O	Text Based Encoding			Comment
				Keyword	Parameter	Default	
<Importance>	B	o	I	IMPORTANCE	<Importance-parameter>	"1"	Importance of the contents
<Priority>	B	o	I	PRIORITY	<Priority-parameter>	"0"	Priority of the contents
<RelatedSpec>	B	o	I	RELATED	"@" <File-of-Relatedspec>	–	Path name of the file containing the Relatedspecs of all messages related to the present one
<ReplyId>	B	o	I	REPLYID	<Ipm-id-parameter>	–	ID of the message, for which this one is a response
<ReplyTime>	B	o	I	REPLYTIME	<Date-time-parameter>	–	Date-Time by when the recipient(s) of the message should reply to the authorizing users (Reply Time)
<Sensitivity>	B	o	I	SENSITIVITY	<Sensitivity-parameter>	"0"	Sensitivity of the contents
<Subject>	B	o	I	SUBJECT	<Subject-parameter>	–	
<Userinfo>	+	o	I	USERINFO	<Userinfo-parameter>	–	User-provided

TABLE 107/T.611

Additional syntax elements for the <DecodeIPMExtension> of E-Mail services

Syntax Element	C	T	I/O	Text Based Encoding			Comment
				Keyword	Parameter	Default	
Syntax elements belonging to the decode procedure itself (job description)							
<DecodeIPM-SubFunction>	B	m	I	SUBFUNC	"DecodeIPM"	–	
<Message>	B	m	I	MESSAGE	<Path-parameter>	–	Path to file that holds the IPM to be decoded
<Minor>	+	o	O	MINOR	<Error-parameter>	–	
<Warning>	+	o	O	WARNING	<Error-parameter>	–	
If only one body part is present							
<Convert>	B	m	O	CONVERT	<Convert-id-parameter>	–	States the Transfer Format of the decoded body part
<Document>	B	m	O	FILENAME	<File-parameter>	–	The filename may be pre-set by the LA on request. If only one body part is available, the name shall be retained by the CA. If many body parts are available the name may be overwritten by the CA
<Type>	B	o	O	TYPE	<Type-id-parameter>	–	Specifies the document type received
If many body parts are present							
<DocumentSpec>	+	m	I	FILENAME	"@" <File-of-filespec>	–	Specifies a list of files; uses a special syntax
Syntax elements receiving the heading information of the IPM to be decoded							
<IpmId>	B	m	O	IPM-ID	<Ipm-id-parameter>	–	Message identifier.
<R-RecipientSpec>	B	m	I	RECIPIENT	"@" <File-of-r-recipientspec>	–	Path of the file containing the Recipientspecs of all intended recipients

TABLE 107/T.611 *(end)*

Additional syntax elements for the <DecodeIPMExtension> of E-Mail services

Syntax Element	C	T	I/O	Text Based Encoding			Comment
				Keyword	Parameter	Default	
<R-OriginatorSpec>	B	m	I	ADDRESS	"@" <File-of-originatorspec>	–	Path name of the file containing the originator and authorizing users
<ContType>	B	o	O	CONT-TYPE	<Content-type-parameter>	–	Content Type
<ExpiryTime>	B	o	I	EXPIRYTIME	<Date-time-parameter>	–	Expiration of the message (Expiry Time)
<Forwarded>	B	o	O	FORWARDED	<Boolean-parameter>	–	Autoforwarded indication
<Importance>	B	o	O	IMPORTANCE	<Importance-parameter>	–	Importance of the contents
<Language>	B	o	O	LANGUAGE	<Language-id-parameter>	–	Identification of the language used
<Priority>	B	o	O	PRIORITY	<Priority-parameter>	–	Priority of the contents
<RelatedSpec>	B	o	I	RELATED	"@" <File-of-Relatedspec>	–	Path name of the file containing the Relatedspecs of all messages related to the present one
<ReplyId>	B	o	O	REPLYID	<Ipm-id-parameter>	–	ID of the message, for which this one is a response
<ReplyTime>	B	o	O	REPLYTIME	<Date-time-parameter>	–	Date-Time by when the recipient(s) of the message should reply to the authorizing users (Reply Time)
<Sensitivity>	B	o	O	SENSITIVITY	<Sensitivity-parameter>	"0"	Sensitivity of the contents
<Subject>	B	o	O	SUBJECT	<Subject-parameter>	–	
<Userinfo>	+	o	O	USERINFO	<Userinfo-parameter>	–	User-provided

16.4 Interpersonal Notification

An LA may transmit an Interpersonal Notification (IPN) with a Send TDD and may receive an IPN with the Receive TDD.

The IPN is treated by the CA as a special document, called the IPN document, i.e, the CA expects from the LA a specially prepared file containing the IPN parameters, when an IPN has to be sent. The CA shall convert the IPN document into the format required by the E-Mail system and transmit it to the recipient. Conversly, the CA shall present an IPN document to the LA when it receives an IPN through the network.

To differentiate between IPMs and IPNs (and DRs, see 16.5) the Content-type-parameter of the <ContType> syntax element is used (see also 16.2.2.9).

The IPN document shall conform to the same syntactical rules as those applying to TDDs.

The following subclauses describe the syntax and the text based encoding of the IPN document.

16.4.1 Syntax of IPN Document

The IPN Document is defined for MHS and conforms to following BNF-style syntax (see Table 108).

```
<IPN-Document>  :=          <NotifyType> <SubjectIPM> <ReceiptTime>
                            [<IPNOrigin>] [<PrefRecipient>] [<Reason>] [<AutoComment>] [<SupplmInfo>]
```

TABLE 108/T.611

Syntax elements of the IPN Document

Syntax element	Purpose
<AutoComment>	Comment of the originator of this notification, if the related IPM was auto-forwarded; only used for non-receipt-notifications of type NRN-FORW
<Reason>	Identifies why the IPM related to this notification was discarded: 0: ipm-expired 1: ipm-obsoleted 2: user-subscription-terminated only used for non-receipt-notifications of type NRN-DISC
<IPNOrigin>	Identifies the originator of this notification
<NotifyType>	Type of the notification: RN: receipt notification NRN-DISC: non-receipt-notification (discarded) NRN-FORW: non-receipt-notification (auto-forwarded) OTHER: other notification types
<PrefRecipient>	The preferred recipient of the IPM related to this notification
<ReceiptTime>	Identifies the time when the originator of this notification received the related IPM; only used for receipt notification (RN)
<SubjectIPM>	Identifies the IPM related to this notification
<SupplmInfo>	Supplementary information of the originating MTA; only used for receipt-notifications (RN)

16.4.2 Text based Encoding

16.4.2.1 Mapping of Keywords

See Table 109.

TABLE 109/T.611

Text Based Encoding of IPN syntax elements for E-Mail services

(A ⮰ stands for new line)

Syntax Element	Keyword / Parameter Pair
<AutoComment>	"AUTO-COMMENT" ":" STRING (SIZE(1..256)) ⮰
<Reason>	"REASON" ":" "0" | "1" | "2" ⮰
<IPNOrigin>	"IPN-ORIGIN" ":" <Mhs-or-descriptor> ⮰
<NotifyType>	"NOTIFY-TYPE" ":" "RN" | "NRN-DISC" | "NRN-FORW" | "OTHER" ⮰
<PrefRecipient>	"DISCLO-REC" ":" <Mhs-or-descriptor> ⮰
<ReceiptTime>	"RECEIPT-TIME" ":" <Date-time-parameter> ⮰
<SubjectIPM>	"SUBJECT-IPM" ":" "@" <Ipm-id-parameter> ⮰
<SupplmInfo>	"SUPPLM-INFO" ":" STRING (SIZE(1..256)) ⮰

16.4.2.2 Encoding of Parameters

For description of the text based encoding of:

– <Mhs-or-descriptor> refer to 16.2.2.4

– <Date-time-parameter> refer to 6.4.4.4

– <Ipm-id-parameter> refer to 16.2.2.11

16.5 Delivery Report

An LA may receive a Delivery Report (DR) with the Receive TDD.

The DR is treated by the CA as a special document, called the DR document. The CA shall present such a DR document to the LA when it receives a DR through the network.

To differentiate between IPMs and DRs (and IPNs, see 16.4) the Content-type-parameter of the <ContType> syntax element is used (see also 16.2.2.9).

The DR document delivered by the CA shall conform to exactly the same syntactical rules as applied to TDDs.

The following subclauses describe the syntax and the text based encoding of the DR document.

16.5.1 Syntax of DR Document

The DR Document is defined for MHS and conforms to following BNF-style syntax (see Table 110).

```
<DR-Document>  :=          <ReportType> <ActRecipient> <MsgSubId> [<ReceiveTime>] [<Reason>]
                           [<Diagnostic>] [<SupplmInfo>]
```

TABLE 110/T.611

Syntax elements of the DR Document

Syntax element	Purpose
<ActRecipient>	Identifies the actual recipient of the IPM related to this report
<Diagnostic>	Additional diagnostic information to the report as defined by ITU-T Recommendation X.411; only used for non-delivery reports (NDR)
<MsgSubId>	Identifies the IPM related to this report (MTSIdentifier)
<Reason>	Identifies why the IPM related to this report was not delivered as defined by ITU-T Recommendation X.411; only used for non-delivery reports (NDR);
<ReceiveTime>	Time at which the message was delivered (DeliveryTime); only used for delivery reports (DR)
<ReportType>	Type of the delivery report: DR: delivery report NDR: non-delivery report
<SupplmInfo>	Supplementary information of the originating MTA

16.5.2 Text based Encoding

16.5.2.1 Mapping of Keywords

See Table 111.

TABLE 111/T.611

Text Based Encoding of DR syntax elements for E-Mail services

(A ⏎ stands for new line)

Syntax Element	Keyword / Parameter Pair
<ActRecipient>	"ACT-RECIPIENT" ":" <Mhs-or-name> ⏎
<Diagnostic>	"DIAGNOSTIC" ":" NUMERIC-STRING ⏎
<MsgSubId>	"MSG-SUB-ID" ":" <Msg-sub-id-parameter> ⏎
<Reason>	"REASON" ":" NUMERIC-STRING ⏎
<ReceiveTime>	"RCVTIME" ":" <Date-time-parameter> ⏎
<ReportType>	"REPORT-TYPE" ":" "DR" | "NDR" ⏎
<SupplmInfo>	"SUPPLM-INFO" ":" STRING (SIZE(1..256)) ⏎

16.5.2.2 Encoding of Parameters

For description of the text based encoding of:

– <Mhs-or-name> refer to 16.2.2.4

– <Msg-sub-id-parameter> refer to 16.2.2.13

– <Date-time-parameter> refer to 6.4.4.4

16.6 Mapping of MHS service elements

Table 112 shows the mapping of MHS service elements as defined per ITU-T X.400-Series of Recommendation to the the syntax elements defined by this Recommendation. Listed are the corresponding keywords used in text based presentation of the TDDs and their support for a sending and/or receiving LA.

TABLE 112/T.611

Mapping of MHS service elements

MHS service element	Syntax element	Keyword	Sending LA	Receiving LA
Alternate Recipient Allowed Indication	<Alternate>	ALTERNATE	Supported	Not supported
Authorizing User's Indication	<R-OriginatorSpec> <S-OriginatorSpec>	ADDRESS ORIGINATOR	– Supported	Supported –
Auto-forwarded Message Indication	<Forwarded>	FORWARDED	Not supported	Supported
Blind Copy Recipients Indication	<R-RecipientSpec> <S-RecipientSpec>	RECIPIENT ADDRESS	– Supported	Supported –
Content Type Indication	<ContType>	CONT-TYPE	Supported	Supported
Conversion Prohibition	<ImplicitConv>	IMPLICIT-CONV	Supported	Not supported
Cross Referencing Indication	<RelatedSpec>	RELATED	Supported	Supported
Deferred Delivery Indication	<SendTime>	SENDTIME	Supported	Not supported
Delivery Report Request	<R-RecipientSpec> <S-RecipientSpec>	RECIPIENT ADDRESS	– Supported	Supported –
Delivery Time Indication	<ReceiveTime>	RCVTIME	Not supported	Supported
Disclosure of Other Recipients	<DiscloRec>	DISCLO-REC	Supported	Not supported
Expiry Date Indication	<ExpiryTime>	EXPIRYTIME	Supported	Supported
Forwarded IP Message	<Type>	TYPE	Supported	Supported
Grade of Delivery Selection	<Priority>	PRIORITY	Supported	Supported
Importance Indication	<Importance>	IMPORTANCE	Supported	Supported
Incomplete Copy Indication[a)]	<IncompleteCopy>	INC-COPY	Not supported	Supported
Intended Recipient Indication	<R-RecipientSpec>	RECIPIENT	Not supported	Supported
IP Message Indication	<IpmId>	IPM-ID	Supported	Supported

TABLE 112/T.611 *(end)*

Mapping of MHS service elements

MHS service element	Syntax element	Keyword	Sending LA	Receiving LA
Language Indication	<Language>	LANGUAGE	Not supported	Supported
Latest Delivery Indication[a)]	<LastTime>	LASTTIME	Supported	Not supported
Message Identification[a)]	<Userinfo>	USERINFO	Supported	Supported
Message Submission Identification	<MsgSubId>	MSG-SUB-ID	Supported[b)]	Not supported
Multi-Destination Delivery	<R-RecipientSpec> <S-RecipientSpec>	RECIPIENT ADDRESS	– Supported	Supported –
Multi-Part Body	implicit	n/a[c)]	Supported	Supported
Non-Receipt Notification	<R-RecipientSpec> <S-RecipientSpec>	RECIPIENT ADDRESS	– Supported	Supported –
Non-Delivery Report	<R-RecipientSpec> <S-RecipientSpec>	RECIPIENT ADDRESS	– Supported	Supported –
Obsoleting IPM Indication	<RelatedSpec>	RELATED	Supported	Supported
Originator Indication	<R-OriginatorSpec> <S-OriginatorSpec>	ADDRESS ORIGINATOR	– Supported	Supported –
Primary and Copy Recipients Indication	<R-RecipientSpec> <S-RecipientSpec>	RECIPIENT ADDRESS	– Supported	Supported –
Probe Indication	<Convert>	CONVERT	Supported	Not supported
Receipt Notification	<R-RecipientSpec> <S-RecipientSpec>	RECIPIENT ADDRESS	– Supported	Supported –
Reply IP Message Indication	<ReplyId>	REPLYID	Supported	Supported
Reply Recipients Indication	<R-RecipientSpec> <S-RecipientSpec>	RECIPIENT ADDRESS	– Supported	Supported –
Reply IPM Indication	<RelatedSpec>	RELATED	Supported	Supported
Reply Time Indication	<ReplyTime>	REPLYTIME	Supported	Supported
Sensitivity Indication	<Sensitivity>	SENSITIVITY	Supported	Supported
Subject Indication	<Subject>	SUBJECT	Supported	Supported
Submission Time Indication	<SubmitTime>	SUBMITIME	Supported[b)]	Supported

a) The service elements IncompleteCopyIndication, LatestDeliveryIndication and MessageIdentification are additional. Therefore they are not addressed in the profile definition.

b) The MessageSubmissionIdentification and the SubmissionTimeStamp is originated and maintained by the Message Transfer System (MTS). Both keywords are used in the <SendAckTDD> as output parameter.

c) The assignment of a keyword is not applicable.

Table 113 sumarizes the time stamps used within this Recommendation related to service elements of the message transfer system and the interpersonal messaging system.

These service elements, their meaning and the corresponding keywords of the text based encoding are shown in Table 113. Another column of this table indicates, if the time stamps may be specified by the sending LA (Send TDD) and if they may be received by an LA (Receive TDD).

TABLE 113/T.611

Summary of MHS Time-stamps

Service Element	Keyword	Send/receive	Meaning
Expiry Date Indication	EXPIRYTIME	s/r	This optional heading field of an IPM specifies the time when the authorizing user(s) consider the IPM to lose validity
Reply Time Indication	REPLYTIME	s/r	This optional heading field of an IPM specifies the time by when the authorizing user(s) request that any replies to the subject IPM be originated
Deferred Delivery Indication	SENDTIME	s/–	Specifies the time before which the message should not be delivered to the recipient(s); it may be generated by the originator of the message
Latest Delivery Indication	LASTTIME	s/–	Specifies the time after which the message should not be delivered to the recipient(s); it may be generated by the originator of the message
Delivery Time Indication	RCVTIME	–/r	Specifies the time at which the message was delivered to the recipient message transfer system user; it shall be generated by the message transfer system if the message was successfully delivered
Submission Time Indication	SUBMITTIME	s/r	Specifies the time at which the message transfer system accepts responsibility for the message; it shall be generated by the message transfer system
Receipt Time Indication	RECEIPT-TIME	s/r	Only used for receipt notification (RN); identifies the time when the originator of a notification has received the related IPM

16.7 E-Mail Profiles

This Recommendation provides access to E-Mail services as described in the ITU-T X.400-Series of Recommendation as well as to other E-Mail environments.

Powerful stored-and-forward telecommunication services make high demands on an interface specification. The Send, SendAck and Receive TDDs are enhanced by a large number of basic E-Mail specific elements respecting the philosophie of this Recommendation. Most of these addressed service elements are optional.

Enhanced features provided at the interface level for E-Mail access are:

– transmission and reception of notifications;

– reception of delivery reports;

– transmission and reception of E-Mail message(s) included in an E-Mail message (e.g. IPM as body part).

It is required that a CA implementation conforms to this Recommendation, even if not all possible features are supported. To improve the interoperability between LAs and CAs from different vendors, the concept of "profile" is introduced.

The profiles:

- X400-IPM;
- BasicX400-IPM;
- SimpleEMail.

define functional service subsets offered at the interface level described in this Recommendation[15)].

Together with the ICE and the functional classes the profile concept identifies the E-Mail service elements supported by a CA.

A CA may support one or more profiles. Moreover, implementers may describe and name their own profiles.

16.7.1 Service Profile X400-IPM

The X400-IPM profile comprises all X.400 service elements provided at the interface level of this Recommendation to support the conveyance of IPMs[16)] (see Table 114).

16.7.2 Service Profile BasicX400-IPM

The BasicX400-IPM profile specifies the basic functional subset of X.400 service elements provided at the interface level of this Recommendation to support the conveyance of IPMs (see Table 115).

16.7.3 Service Profile SimpleEMail

The SimpleEMail profile was introduced to enable access to E-Mail services which are not conforming to the open communication X.400-Series of Recommendation. It provides a minimal functional subset of E-Mail service elements provided at the interface level of this Recommendation (see Table 116).

16.8 CA-Descriptor Settings

A CA supporting E-Mail services shall specify in the CA-Descriptor the body parts (document Transmission Formats), the profile(s) and the Content-Types supported (see also 9.5).

For support of the EncodeIPM and DecodeIPM TDD, the EXTEND keyword of the CA-Descriptor has to be set accordingly. Table 117 depicts this.

17 Service: File Transfer

This Recommendation allows sending or receiving information through File Transfer services. However, full control of File Transfer services through the interface is not intended.

15) Other profiles like EDIM and EDIN are for further study.

16) The profile X400-IPM is specified according to the European Procurement Handbook for Open Systems (EPHOS), i.e. the profiles ENV 41201 and ENV 41202.

TABLE 114/T.611

Service Profile X400-IPM

Service element	Syntax element	Support of CA
Alternate Recipient Allowed Indication	<Alternate>	Mandatory
Authorizing User’s Indication	<R-OriginatorSpec>, <S-OriginatorSpec>	Mandatory
Auto-forwarded Message Indication	<Forwarded>	Mandatory
Blind Copy Recipients Indication	<R-RecipientSpec>, <S-RecipientSpec>	Mandatory
Content Type Indication	<ContType>	Mandatory
Conversion Prohibition	<ImplicitConv>	Mandatory
Cross Referencing Indication	<RelatedSpec>	Mandatory
Deferred Delivery Indication	<SendTime>	Mandatory
Delivery Report	<R-RecipientSpec>, <S-RecipientSpec>	Mandatory
Delivery Time Indication	<ReceiveTime>	Mandatory
Disclosure of Other Recipients	<DiscloRec>	Mandatory
Expiry Date Indication	<ExpiryTime>	Mandatory
Forwarded IP Message	<Type>	Mandatory
Grade of Delivery Selection	<Priority>	Mandatory
Importance Indication	<Importance>	Mandatory
Intended Recipient Indication	<R-RecipientSpec>	Mandatory
IP Message Indication	<IpmId>	Mandatory
Language Indication	<Language>	Mandatory
Message Submission Identification	<MsgSubId>	Mandatory
Multi-Destination Delivery	<R-RecipientSpec>, <S-RecipientSpec>	Mandatory
Multi-Part Body	Implicit	Mandatory
Non-Receipt Notification	<R-RecipientSpec>, <S-RecipientSpec>	Mandatory
Non-Delivery Report	<R-RecipientSpec>, <S-RecipientSpec>	Mandatory
Obsoleting IPM Indication	<RelatedSpec>	Mandatory
Originator Indication	<R-OriginatorSpec>, <S-OriginatorSpec>	Mandatory
Primary and Copy Recipients Indication	<R-RecipientSpec>, <S-RecipientSpec>	Mandatory
Probe Indication	<Convert>	Mandatory
Receipt Notification	<R-RecipientSpec>, <S-RecipientSpec>	Mandatory
Reply IP Message Indication	<ReplyId>	Mandatory
Reply Recipients Indication	<R-RecipientSpec>, <S-RecipientSpec>	Mandatory
Reply IPM Indication	<RelatedSpec>	Mandatory
Reply Time Indication	<ReplyTime>	Mandatory
Sensitivity Indication	<Sensitivity>	Mandatory
Subject Indication	<Subject>	Mandatory
Submission Time Indication	<SubmitTime>	Mandatory

TABLE 115/T.611

Service Profile BasicX400-IPM

Service element	Syntax element	Support of CA
Alternate Recipient Allowed Indication	<Alternate>	–
Authorizing User's Indication	<R-OriginatorSpec>, <S-OriginatorSpec>	Mandatory
Auto-forwarded Message Indication	<Forwarded>	–
Blind Copy Recipients Indication	<R-RecipientSpec>, <S-RecipientSpec>	Mandatory
Content Type Indication	<ContType>	Mandatory
Conversion Prohibition	<ImplicitConv>	–
Cross Referencing Indication	<RelatedSpec>	Mandatory
Deferred Delivery Indication	<SendTime>	Mandatory
Delivery Report	<R-RecipientSpec>, <S-RecipientSpec>	–
Delivery Time Indication	<ReceiveTime>	–
Disclosure of Other Recipients	<DiscloRec>	–
Expiry Date Indication	<ExpiryTime>	Mandatory
Forwarded IP Message	<Type>	Mandatory
Grade of Delivery Selection	<Priority>	Mandatory
Importance Indication	<Importance>	Mandatory
Intended Recipient Indication	<R-RecipientSpec>	–
IP Message Indication	<IpmId>	Mandatory
Language Indication	<Language>	Mandatory
Message Submission Identification	<MsgSubId>	–
Multi-Destination Delivery	<R-RecipientSpec>, <S-RecipientSpec>	Mandatory
Multi-Part Body	implicit	Mandatory
Non-Receipt Notification	<R-RecipientSpec>, <S-RecipientSpec>	–
Non-Delivery Report	<R-RecipientSpec>, <S-RecipientSpec>	–
Obsoleting IPM Indication	<RelatedSpec>	Mandatory
Originator Indication	<R-OriginatorSpec>, <S-OriginatorSpec>	Mandatory
Primary and Copy Recipients Indication	<R-RecipientSpec>, <S-RecipientSpec>	Mandatory
Probe Indication	<Convert>	–
Receipt Notification	<R-RecipientSpec>, <S-RecipientSpec>	–
Reply IP Message Indication	<ReplyId>	Mandatory
Reply Recipients Indication	<R-RecipientSpec>, <S-RecipientSpec>	Mandatory
Reply IPM Indication	<RelatedSpec>	Mandatory
Reply Time Indication	<ReplyTime>	Mandatory
Sensitivity Indication	<Sensitivity>	Mandatory
Subject Indication	<Subject>	Mandatory
Submission Time Indication	<SubmitTime>	–

TABLE 116/T.611

Service Profile SimpleEMail

Service element	Syntax element	Support of CA
Alternate Recipient Allowed Indication	<Alternate>	–
Authorizing User's Indication	<R-OriginatorSpec>, <S-OriginatorSpec>	Mandatory
Auto-forwarded Message Indication	<Forwarded>	–
Blind Copy Recipients Indication	<R-RecipientSpec>, <S-RecipientSpec>	Mandatory
Content Type Indication	<ContType>	Mandatory
Conversion Prohibition	<ImplicitConv>	–
Cross Referencing Indication	<RelatedSpec>	Mandatory
Deferred Delivery Indication	<SendTime>	–
Delivery Report	<R-RecipientSpec>, <S-RecipientSpec>	–
Delivery Time Indication	<ReceiveTime>	–
Disclosure of Other Recipients	<DiscloRec>	–
Expiry Date Indication	<ExpiryTime>	–
Forwarded IP Message	<Type>	–
Grade of Delivery Selection	<Priority>	Mandatory
Importance Indication	<Importance>	–
Intended Recipient Indication	<R-RecipientSpec>	–
IP Message Indication	<IpmId>	Mandatory
Language Indication	<Language>	–
Message Submission Identification	<MsgSubId>	–
Multi-Destination Delivery	<R-RecipientSpec>, <S-RecipientSpec>	Mandatory
Multi-Part Body	Implicit	Mandatory
Non-Receipt Notification	<R-RecipientSpec>, <S-RecipientSpec>	–
Non-Delivery Report	<R-RecipientSpec>, <S-RecipientSpec>	–
Obsoleting IPM Indication	<RelatedSpec>	–
Originator Indication	<R-OriginatorSpec>, <S-OriginatorSpec>	Mandatory
Primary and Copy Recipients Indication	<R-RecipientSpec>, <S-RecipientSpec>	Mandatory
Probe Indication	<Convert>	–
Receipt Notification	<R-RecipientSpec>, <S-RecipientSpec>	–
Reply IP Message Indication	<ReplyId>	Mandatory
Reply Recipients Indication	<R-RecipientSpec>, <S-RecipientSpec>	Mandatory
Reply IPM Indication	<RelatedSpec>	Mandatory
Reply Time Indication	<ReplyTime>	Mandatory
Sensitivity Indication	<Sensitivity>	Mandatory
Subject Indication	<Subject>	Mandatory
Submission Time Indication	<SubmitTime>	–

TABLE 117/T.611

Additional CA-Descriptor settings for E-Mail services

Keyword	Parameter	CA declares that
EMAIL	"STD"	text body parts are supported; "STD" maps to the appropriate basic alphabet of the selected E-Mail service
EMAIL	"TELETEX"	teletex body parts are supported
EMAIL	"VIDEOTEX"	videotex body parts are supported
EMAIL	"G3FAX"	telefax group 3 body parts are supported
EMAIL	"G4CLASS1"	telefax group 4 body parts are supported
EMAIL	"MIXEDMODE"	mixed mode body parts are supported
EMAIL	"MESSAGE"	IPM body parts are supported
EMAIL	"BILATERAL"[a)]	bilateral defined body parts are supported
EMAIL	"NATIONAL"	national defined body parts are supported
EMAIL	"ODA"	ODA body parts are supported
CONT-TYPE	"mhsIPMxx"	MHS IPM content protocol is supported (xx stands for the year of release of the related Recommendation, e.g. 84 for 1984)
CONT-TYPE	"mhsIPNxx"	MHS IPN content protocol is supported (the CA understands IPN documents as described in **Error! Not a valid result for table.**)
CONT-TYPE	"mhsDR"	MHS DR content protocol is supported (the CA understands DR documents as described in 16.5)
PROFILE	"X400-IPM"	the X400 profile is supported
PROFILE	"BasicX400-IPM"	the BasicX400 profile is supported
PROFILE	"SimpleEMail"	the SimpleEMail profile is supported
EXTEND	"EncodeIPM"	the EncodeIPM functionality is supported
EXTEND	"DecodeIPM"	the DecodeIPM functionality is supported

a) "Bilateral" is used for other body part types not mentioned (in this list).

NOTE – A CA may also support other or private E-Mail systems. In this case the type of E-Mail system should be reflected in the CONT-TYPE and PROFILE parameters (e.g "CONT-TYPE : cmcIPM" for support of CMC Interfaces, see 16.2.2.9).

17.1 Service Specific Syntax Elements

```
<ServiceDependentKeywordsSend>  :=
                                    (<Recipient> | <RecipientSpec>)
                                    ((<Document> <Convert> [<Type>]) | <DocumentSpec>)
                                    <Compress> <Environ> [<Password>] [<Name>]

<ServiceDependentKeywordsSendAck>  :=
                                    <Recipient>
                                    ((<Document> <Convert> [<Type>]) | <DocumentSpec>)
                                    <Compress> <Environ> [<Password>] [<Name>]
```

```
<ServiceDependentKeywordsReceive>  :=
                                        [<Originator>]
                                        ((<Document> <Convert> <Type>) | <DocumentSpec>)
                                        <Compress> <Environ> [<Password>] [<Name>]
```

See Table 118.

TABLE 118/T.611

Additional syntax elements for FT services

Syntax element	Purpose
<Compress>	Specifies the data compression format of the transferred file(s)
<Convert>	Specifies the Transfer Format to be used
<Document>, <DocumentSpec>	Specifies the document or the documents to be sent or being received
<Environ>	Specifies the operating system the file was created under by the originator
<Name>	Name the file has under the originating operating system
<Originator>	Specifies the communications address of the originator
<Password>	Password needed to access remote station
<Recipient>, <RecipientSpec>	Specifies the communications address(es) of the recipient(s)
<Type>	Specifies the Transmission Format used

17.2 Text Based Encoding

17.2.1 Mapping of Keywords

See Table 119.

TABLE 119/T.611

Text Based Encoding of additional syntax elements for FT services

(A ⮰ stands for new line)

Syntax element	Keyword/Parameter Pair
<Compress>	"COMPRESS" ":" <Compress-parameter> ⮰
<Convert>	"CONVERT" ":" <Convert-id-parameter> ⮰
<Document>	"FILENAME" ":" <File-parameter> ⮰
<DocumentSpec>	"FILENAME" ":" "@" <File-of-filespec> ⮰
<Environ>	"ENVIRON" ":" <Environ-parameter> ⮰
<Name>	"NAME" ":" <Name-parameter> ⮰
<Originator>	"ADDRESS" ":" <Address-parameter> ⮰
<Password>	"PASSWORD" ":" <Password-parameter> ⮰
<Recipient>	"ADDRESS" ":" <Address-parameter> ⮰
<RecipientSpec>	"ADDRESS" ":" "@" <File-of-addrspec> ⮰
<Type>	"TYPE" ":" <Type-id-parameter> ⮰

17.2.2 Encoding of Parameters

See also 6.4.4 for the encoding of non service dependent parameters used.

17.2.2.1 Service-id-parameter

The Service-id parameter is encoded as STRING set to the constant value "FT".

Syntax:

```
<Service-id-parameter>  :=          "FT"
```

17.2.2.2 Type-id-parameter

The Type-id-parameter is encoded as STRING set to the constant value "STD".

Syntax:

```
<Type-id-parameter>  :=             "STD"
```

17.2.2.3 Convert-id-parameter

The Convert-id-parameter is encoded as STRING set to the constant value "VOID".

Syntax:

```
<Convert-id-parameter>  :=          "VOID"
```

17.2.2.4 File-of-addrspec and Address-parameter

The File-of-addrspec parameter is encoded as PATH which points to a file containing Addrspec-parameters, which contain the Address-parameter.

The Address-parameter is encoded as STRING. The STRING shall contain the telephone number. If the phone number starts with a "!" then it may contain special characters that are treated as operators (or modifiers) rather than dial digits. The characters allowed are for further study.

Alternatively an alias name may be given instead of the phone number, provided the alias is introduced with an "&" character. It is assumed that the CA knows how to decode the alias specified.

Syntax:

```
<File-of-addrspec>  :=              PATH
                                    --  The path points to a file containing one ore more
                                    --  <Addrspec-parameter>s

<Addrspec-parameter>  :=            <Address-parameter>

<Address-parameter>  :=             <Phone-number> | ("!" <Dial-command>) | ("&" <Alias>)
                                    --  contains telephone number, dialing sequence or alias

<Phone-number>  :=                  NUMERIC-STRING

<Dial-command>  :=                  STRING
                                    --  for further study

<Alias>  :=                         STRING
```

17.2.2.5 File-of-filespec and File-parameter

The File-of-filespec parameter is encoded as PATH which points to a file containing Filespec-parameters, which contain the File-parameter. The File-parameter itself is also encoded as PATH, which points to the file transferred.

Syntax:

<File-of-filespec> :=	PATH -- *The path points to a file containing one ore more* -- *<Filespec-parameter>s*
<Filespec-parameter> :=	<File-parameter> "," <File-conv> ["," [<File-type>] ["," [<File-compress>] ["," <File-name>]]]
<File-parameter> :=	PATH -- *Path to the file transferred*
<File-conv> :=	<Convert-id-parameter> -- *Specifies the Transfer Format of the file*
<File-type> :=	<Type-id-parameter> -- *Specifies the Transmission Format of the file*
<File-compress> :=	<Compress-parameter> -- *Specifies the compression*
<File-name> :=	<Name-parameter> -- *Specifies the name of the file*

17.2.2.6 Compress-parameter

The Compress-parameter is encoded as STRING set to one of the following values:

"VOID"	No compression (default)
any other	Compression identifier

Syntax:

<Compress-parameter> :=	"VOID" | <Compress-identifier>
<Compress-identifier> :=	STRING -- e.g. "ZIP", "AZJ", "ARJ", "LZH" or "V42"

17.2.2.7 Environ-parameter

The Environ-parameter is coded as STRING set to one of the following values:

"MSDOS"	MS-DOS
"WINDOWS"	Windows
"UNIX"	Unix
"OS2"	OS/2
"MacOS"	Mac
any other	Reserved for further study

Syntax:

<Environ-parameter> := "MSDOS" | "WINDOWS" | "UNIX" | "OS2" | "MacOS"

17.2.2.8 Password-parameter

The Password-parameter is encoded as STRING.

Syntax:

<Password-parameter> := STRING

17.2.2.9 Name-parameter

The Name-parameter is encoded as STRING between 1 and 16 characters long.

Syntax:

<Name-parameter> := STRING (SIZE(1..16))

17.3 Additional Functionality

17.3.1 Function: Send and SendAck

See Tables 120 and 121.

TABLE 120/T.611

Additional functionality of the <SendTDD> for File Transfer

Syntax Element	C	T	I/O	Text Based Encoding			Comment
				Keyword	Parameter	Default	
<Environ>	B	m	I	ENVIRON	<Environ-parameter>	–	Operating system of sender
<Password>	B	o	I	PASSWORD	<Password-parameter>	–	Password for access of remote filesystem
Send to one addressee							
<Recipient>	B	m	I	ADDRESS	<Address-parameter>	–	Specifies one recipient's call number
Send to one or many addressees							
<RecipientSpec>	+	m	I	ADDRESS	"@" <File-of-addrspec>	–	Specifies a list of recipients
Send only one file							
<Convert>	B	m	I	CONVERT	<Convert-id-parameter>	–	States the Transfer Format of the outgoing file
<Document>	B	m	I	FILENAME	<File-parameter>	–	The single outgoing file to be transmitted, delivered under the Transfer Format specified by the Convert keyword
<Type>	B	o	I	TYPE	<Type-id-parameter>	"STD"	Specifies the document type to be sent
<Compress>	B	o	I	COMPRESS	<Compress-parameter>	"VOID"	Compression
<Name>	B	o	I	NAME	<Name-parameter>	–	Name of file under original environment
Send one or many files							
<DocumentSpec>	+	m	I	FILENAME	"@" <File-of-filespec>	–	Specifies a list of files; uses a special syntax

TABLE 121/T.611

Additional functionality of the <SendAckTDD> for File Transfer

Syntax Element	C	T	I/O	Text Based Encoding			Comment
				Keyword	Parameter	Default	
<Recipient>	B	m	I/O	ADDRESS	<Address-parameter>	–	Specifies one recipient's call number
<Environ>	B	m	I	ENVIRON	<Environ-parameter>	–	Operating system of sender
<Password>	B	o	I	PASSWORD	<Password-parameter>	–	Password for access of remote filesystem
Send only one file							
<Convert>	B	m	I	CONVERT	<Convert-id-parameter>	–	States the Transfer Format of the outgoing file
<Document>	B	m	I	FILENAME	<File-parameter>	–	The single outgoing file to be transmitted, delivered under the Transfer Format specified by the Convert keyword
<Type>	B	o	I	TYPE	<Type-id-parameter>	"STD"	Specifies the document type to be sent
<Compress>	B	o	I	COMPRESS	<Compress-parameter>	"VOID"	Compression
<Name>	B	o	I	NAME	<Name-parameter>	–	Name of file under original environment
Send one or many files							
<DocumentSpec>	+	m	I	FILENAME	"@" <File-of-filespec>	–	Specifies a list of files; uses a special syntax

17.3.2 Function: Receive

See Table 122.

TABLE 122/T.611

Additional functionality of the <ReceiveTDD> for File Transfer

Syntax Element	C	T	I/O	Text Based Encoding			Comment
				Keyword	Parameter	Default	
<Environ>	B	m	O	ENVIRON	<Environ-parameter>	–	Operating system of sender
<Password>	B	o	I	PASSWORD	<Password-parameter>	–	Password for access of remote filesystem
<Originator>	B	o	O	ADDRESS	<Address-parameter>	–	Specifies orginator's phone number
Only one file received							
<Convert>	B	m	O	CONVERT	<Convert-id-parameter>	–	States the Transfer Format of the received file
<Document>	B	m	I/O	FILENAME	<File-parameter>	–	The filename may be pre-set by the LA on request. If only one file is received, the name shall be retained by the CA. If many files are received the name may be overwritten by the CA
<Type>	B	o	O	TYPE	<Type-id-parameter>	–	Specifies the document type received
<Compress>	B	o	O	COMPRESS	<Compress-parameter>	"VOID"	Compression
<Name>	B	o	O	NAME	<Name-parameter>	–	Name of file under original environ
Many files received							
<DocumentSpec>	+	m	O	FILENAME	"@" <File-of-filespec>	–	Specifies a list of files; uses a special syntax

17.4 CA-Descriptor Settings

A CA supporting the File Transfer service shall specify the Type-id-parameter "STD" (see Table 123).

TABLE 123/T.611

Additional CA-Descriptor settings for File Transfer

Keyword	Parameter	CA declares that
FT	"STD"	the basic file transfer service is supported

PART III – BINARY ENCODING SCHEME

18 Generic C description

This clause describes the binary encoding scheme for the TDDs using a "generic" C-language description (see also A.2). The binary coding scheme requires the Code-ID set to "C" (see 6.3). The purpose of this description is to provide binary compatibility of TDDs between different vendors' applications using the binary encoding scheme.

The descriptions provided have to be adapted to the supporting platform to perform properly (see also Part IV of this Recommendation).

The following generic C data types are used to describe the TDDs. These data types shall be mapped onto their specific, platform-dependent C data type counterparts (see Table 124).

TABLE 124/T.611

Generic C data types

Generic C data type	Comment
INT16	A signed integer value, coded on 16 bits
UINT32	An unsigned integer value, coded on 32 bits
CHAR	A character, coded on 8 bits
BYTE	An octet, coded on 8 bits

18.1 Binary encoding of TDDs

The Request TDDs and Responses are represented by C structures. The components of those structures correspond to the syntax elements used in text-based encodings, as defined in clause 6 and the appropriate clauses of Part **Error! Reference source not found.** of this Recommendation.

The binary layout, described with C-Language structures, has been chosen so that elements of different data types are aligned to a 4 byte boundary.

Each TDD consists of, in order:

– a 4 octet wide binary header;

– a 16 octet wide TDD descriptor, which contains the offsets to four specific sections in the TDD layout; the offsets are computed from the base of the binary header.

The rest of the binary layout is made up by the four specific sections, which may appear in any order:

– *TDD section 1* – Contains the common input parameters, i.e. the ones described in clause 6.

– *TDD section 2* – Contains the service dependent input parameters, i.e. the ones described in the appropriate clause of Part II of this Recommendation.

– *TDD section 3* – Contains the common output parameters, i.e. the ones described in clause 6.

– *TDD section 4* – Contains the service dependent out parameters, i.e. the ones described in the appropriate clause of Part II of this Recommendation.

Because of this structure it is possible for applications to add or enhance parameters of a TDD with private parameters.

Figure 16 below depicts this structure.

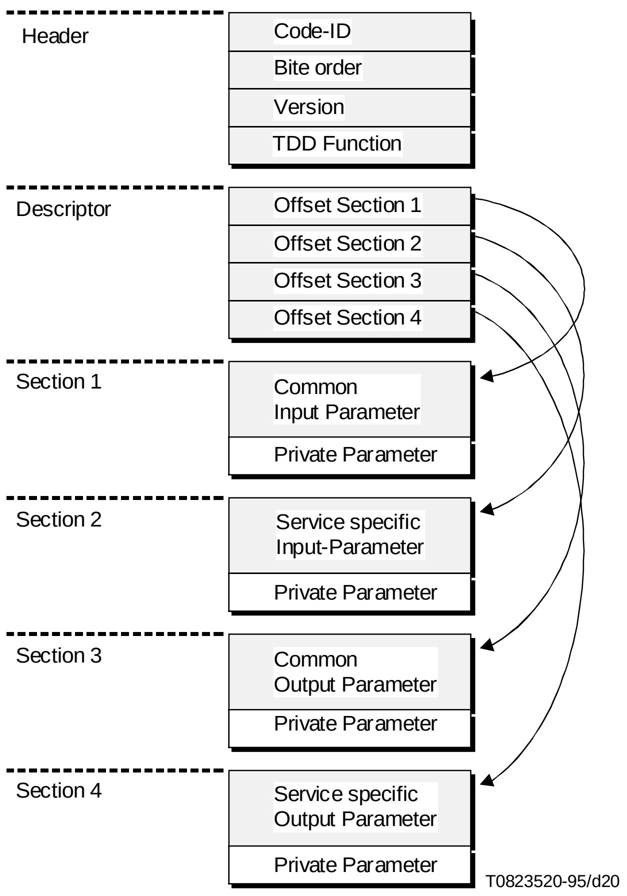


FIGURE 16/T.611

Binary Structure of TDD

18.1.1 TDD Header

```
/***
*          <TDD> syntax element
***/

struct tdd {
          struct tdd_header            hdr;                /* <TDD Header> syntax element, see below */
          struct tdd_descriptor        ofs;                /* Descriptor containing offsets, see below */
          };                                               /* Function dependent TDD structures ... */

/***
*          <TDD Header> syntax element
***/

struct tdd_header {
          BYTE_code_id;                /* Code-ID, constant set to 0x43 ('C') */
          BYTE_byte_order;             /* Optional constant value indicating byte order */
          BYTE_itu_version;            /* Constant indicating version of this Recommendation */
          BYTE_function;               /* Function code */
          };
```

```
/***
*          Offset descriptor
***/

struct tdd_descriptor {
           UINT32 cinp;              /* offset to section 1 (common input parameter) */
           UINT32 sinp;              /* offset to section 2 (service dependent input parameter) */
           UINT32 coup;              /* offset to section 3 (common output parameter) */
           UINT32 soup;              /* offset to section 4 (service dependent output parameter) */
           };
```

18.1.2 Service independent TDD structures

```
/***
*          <SendTDD> structure
***/

struct send_tdd_inp {                                  /* common input parms */
           La_id_type                la_id;            /* <LaId> */
           Req_id_type               req_id;           /* <ReqID> */
           Service_type              service;          /* <Service> */
           Date_time_type            sendtime;         /* <SendTime> */
           Date_time_type            lasttime;         /* <LastTime> */
           Comment_type              comment;          /* <Comment> */
           Userkey_type              userkey;          /* <UserKey> */
           };

/***
*          <SendackTDD> structure
***/

struct sendack_tdd_inp {                               /* common input parms */
           La_id_type                la_id;            /* <LaId> */
           Req_id_type               req_id;           /* <ReqID> */
           Service_type              service;          /* <Service> */
           Date_time_type            sendtime;         /* <SendTime> */
           Date_time_type            lasttime;         /* <LastTime> */
           Comment_type              comment;          /* <Comment> */
           Userkey_type              userkey;          /* <UserKey> */
           };

struct sendack_tdd_out {                               /* common output parms */
           Comid_type                comid;            /* <ComId> */
           Status_type               status;           /* <Status> */
           Error_type                error;            /* <Error> */
           Error_type                minor;            /* <Minor> */
           Error_type                warning;          /* <Warning> */
           };

/***
*          <ReceiveTDD> structure
***/

struct receive_tdd_inp {                               /* common input parms */
           La_id_type                la_id;            /* <LaId> */
           Req_id_type               req_id;           /* <ReqID> */
           Service_type              service;          /* <Service> */
           Boolean_type              deletedoc;        /* <Delete > */
           Comid_type                comid;            /* <Comid > */
           };
```

```
struct receive_tdd_out {                                /* common output parms */
          Comid_type              comid;                /* <ComId> */
          Status_type             status;               /* <Status> */
          Error_type              error;                /* <Error> */
          Error_type              minor;                /* <Minor> */
          Error_type              warning;              /* <Warning> */
          Service_type            service;              /* <Service> */
          Date_time_type          rcvtime;              /* <ReceiveTime> */
          };


/***
*         <CopyTDD> structure
***/


struct copy_tdd_inp {                                   /* common input parms */
          La_id_type              la_id;                /* <LaId> */
          Req_id_type             req_id;               /* <ReqID> */
          Comid_type              comid;                /* <ComId> */
          Req_id_type             reqref;               /* <ReqRef> */
          State_type              state;                /* <State> */
          Path_type               target;               /* <Target> */
          Layout_id_type          layout;               /* <Layout> */
          };


struct copy_tdd_out {                                   /* common output parms */
          Error_type              error;                /* <Error> */
          Error_type              minor;                /* <Minor> */
          Error_type              warning;              /* <Warning> */
          };


/***
*         <DeleteTDD> structure
***/


struct delete_tdd_inp {                                 /* common input parms */
          La_id_type              la_id;                /* <LaId> */
          Req_id_type             req_id;               /* <ReqID> */
          Comid_type              comid;                /* <ComId> */
          Req_id_type             reqref;               /* <ReqRef> */
          };


struct delete_tdd_out {                                 /* common output parms */
          Error_type              error;                /* <Error> */
          Error_type              minor;                /* <Minor> */
          Error_type              warning;              /* <Warning> */
          };


/***
*         <CancelTDD> structure
***/


struct cancel_tdd_inp {                                 /* common input parms */
          La_id_type              la_id;                /* <LaId> */
          Req_id_type             req_id;               /* <ReqID> */
          Comid_type              comid;                /* <ComId> */
          Req_id_type             reqref;               /* <ReqRef> */
          };
```

```
struct cancel_tdd_out {                                 /* common output parms */
        Error_type              error;          /* <Error> */
        Error_type              minor;          /* <Minor> */
        Error_type              warning;        /* <Warning> */
        };

/***
*       <PurgeTDD> structure
***/

struct purge_tdd_inp {                                  /* common input parms */
        La_id_type              la_id;          /* <LaId> */
        Req_id_type             req_id;         /* <ReqID> */
        Comid_type              comid;          /* <ComId> */
        Req_id_type             reqref;         /* <ReqRef> */
        State_type              state;          /* <State> */
        };

struct purge_tdd_out {                                  /* common output parms */
        Error_type              error;          /* <Error> */
        Error_type              minor;          /* <Minor> */
        Error_type              warning;        /* <Warning> */
        };

/***
*       <RescheduleTDD> structure
***/

struct reschedule_tdd_inp {                             /* common input parms */
        La_id_type              la_id;          /* <LaId> */
        Req_id_type             req_id;         /* <ReqID> */
        Comid_type              comid;          /* <ComId> */
        Req_id_type             reqref;         /* <ReqRef> */
        Address_type            address;        /* <Address> */
        Date_time_type          sendtime;       /* <SendTime> */
        Date_time_type          lasttime;       /* <LastTime> */
        };

struct reschedule_tdd_out {                             /* common output parms */
        Error_type              error;          /* <Error> */
        Error_type              minor;          /* <Minor> */
        Error_type              warning;        /* <Warning> */
        };

/***
*       <DispatchTDD> structure
***/

struct dispatch_tdd_inp {                               /* common input parms */
        La_id_type              la_id;          /* <LaId> */
        Req_id_type             req_id;         /* <ReqID> */
        Comid_type              comid;          /* <ComId> */
        La_id_type              newla;          /* <NewLa> */
        };

struct dispatch_tdd_out {                               /* common output parms */
        Error_type              error;          /* <Error> */
        Error_type              minor;          /* <Minor> */
        Error_type              warning;        /* <Warning> */
        };
```

```
/***
*           <PreviewTDD> structure
***/

struct preview_tdd_inp {                                 /* common input parms */
            La_id_type              la_id;               /* <LaId> */
            Req_id_type             req_id;              /* <ReqID> */
            Comid_type              comid;               /* <ComId> */
            Path_type               target;              /* <Target> */
            Convert_id_type         convert;             /* <Convert> */
            };

struct preview_tdd_out {                                 /* common output parms */
            Error_type              error;               /* <Error> */
            Error_type              minor;               /* <Minor> */
            Error_type              warning;             /* <Warning> */
            };

/***
*           <PrintTDD> structure
***/

struct print_tdd_inp {                                   /* common input parms */
            La_id_type              la_id;               /* <LaId> */
            Req_id_type             req_id;              /* <ReqID> */
            Path_type               filename;            /* <FileName> */
            Convert_id_type         informat;            /* <InFormat> */
            Printer_id_type         printer;             /* <Printer> */
            };

struct print_tdd_out {                                   /* common output parms */
            Error_type              error;               /* <Error> */
            Error_type              minor;               /* <Minor> */
            Error_type              warning;             /* <Warning> */
            };

/***
*           <ConvertTDD> structure
***/

struct convert_tdd_inp {                                 /* common input parms */
            La_id_type              la_id;               /* <LaId> */
            Req_id_type             req_id;              /* <ReqID> */
            Path_type               filename;            /* <FileName> */
            Path_type               target;              /* <Target> */
            Convert_id_type         informat;            /* <InFormat> */
            Convert_id_type         outformat;           /* <OutFormat> */
            };

struct convert_tdd_out {                                 /* common output parms */
            Error_type              error;               /* <Error> */
            Error_type              minor;               /* <Minor> */
            Error_type              warning;             /* <Warning> */
            };

/***
*           <CheckTDD> structure
***/
```

```
struct check_tdd_inp {                          /* common input parms */
        La_id_type              la_id;          /* <LaId> */
        Req_id_type             req_id;         /* <ReqID> */
        Path_type               filename;       /* <FileName> */
        Convert_id_type         check;          /* <Check> */
        };

struct check_tdd_out {                          /* common output parms */
        Error_type              error;          /* <Error> */
        Error_type              minor;          /* <Minor> */
        Error_type              warning;        /* <Warning> */
        };

/***
*       <ExtendTDD> structure
***/

struct extend_tdd_inp {                         /* common input parms */
        Subfunc_type            subfunc;        /* <SubFunction> */
        La_id_type              la_id;          /* <LaId> */
        Req_id_type             req_id;         /* <ReqID> */
        };

struct extend tdd out {                         /* common output parms */
        Error_type              error;          /* <Error> */
        Error_type              minor;          /* <Minor> */
        Error_type              warning;        /* <Warning> */
        };

/***
*       <NationalTDD> structure
***/

struct national_tdd_inp {                       /* common input parms */Subfunc_type
        subfunc;                /*                    /* <NationalFunction>, for further study */
        La_id_type              la_id;          /* <LaId> */
        Req_id_type             req_id;         /* <ReqID> */
        /*      ...                             <--- others are added here, for further study */
        };

struct national_c_out {                         /* common output parms */
        Error_type              error;          /* <Error> */
        Error_type              minor;          /* <Minor> */
        Error_type              warning;        /* <Warning> */
        /*      ...                             <--- others are added here, for further study */
        };

/***
*       <PrivateTDD> structure
***/

struct private_tdd_inp {                        /* common input parms */
        Subfunc_type            subfunc;        /* <PrivateFunction>, for further study */
        La_id_type              la_id;          /* <LaId> */
        Req_id_type             req_id;         /* <ReqID> */
        /*      ...                             <--- others are added here, for further study */
        };
```

```
struct private_tdd_out {                                /* common output parms */
          Error_type              error;                /* <Error> */
          Error_type              minor;                /* <Minor> */
          Error_type              warning;              /* <Warning> */
          /*        ...                                 <--- others are added here, for further study */
          };
```

18.1.3 Service Independent Constants and Type Definitions

```
/***
*         Constants used for fields of struct tdd_header
***/


/*        code_id field */
#define CODE_ID                 0x43              /* Code-ID, set to 'C' */


/*        byte_order field */
#define L_BYTEORDER                   0x4C              /* Little Endian (Intel, low-high) byte order */
#define B_BYTEORDER                   0x42              /* Big endian byte order */


/*        itu_version field */
#define ITU_VERSION             94                /* Current version (1994) */


/*        function field */
#define SENDACK_TDD             0x10              /* <SendAckFunction> */
#define SEND_TDD                0x11              /* <SendFunction> */
#define RECEIVE_TDD             0x20              /* <ReceiveFunction> */
#define COPY_TDD                0x30              /* <CopyFunction> */
#define DELETE_TDD              0x31              /* <DeleteFunction> */
#define CANCEL_TDD              0x32              /* <CancelFunction> */
#define PURGE_TDD               0x33              /* <PurgeFunction> */
#define RESCHEDULE_TDD          0x34              /* <RescheduleFunction> */
#define DISPATCH_TDD            0x35              /* <DispatchFunction> */
#define PREVIEW_TDD                   0x36              /* <PreviewFunction> */
#define PRINT_TDD               0x40              /* <PrintFunction> */
#define CONVERT_TDD             0x41              /* <ConvertFunction> */
#define CHECK_TDD               0x42              /* <CheckFunction> */
#define EXTEND_TDD              0x50              /* <ExtendFunction> */
#define NATIONAL_TDD            0x60              /* <NationalFunction> */
#define PRIVATE_TDD             0x70              /* <PrivateFunction> */


/***
*         Constants used for struct extend_tdd_inp
***/


/*        subfunction field */
#define POLL_FX3                0x0101            /* FX3: Poll */

#define ENCODE_IPM              0x2010            /* EMAIL: EncodeIPM */
#define DECODE_IPM              0x2011            /* EMAIL: DecodeIPM */


/***
*         Type definitions
*         Some types are service dependent. Their values might be restricted or expanded upon further.
***/
```

```
/*
**        Address_type
**        Some telecommunication services may constrain or expand this type furthermore.
*/

typedef CHAR Address_type[127+1];

/*
**        Boolean_type
**        Takes the values true or false
*/

typedef enum { false=0, true } Boolean_type;

/*
**        Comid type
*/

typedef CHAR Comid type[31+1];

/*
**        Comment type
*/

typedef CHAR Comment type[127+1];

/*
**        Convert_id_type
**        The (asciiz) strings assigned for text based encoding shall be used.
*/

typedef CHAR   Convert_id_type[15+1];

/*
**        Date_time_type and Send_time_type
**        The date and time specification follows closely the ANSI specification
**        The values "IMMEDIATE" and "URGENT" are set by specifying the seconds
**        part of the struct below (tm_sec) with (-1) or (-2)
*/

typedef struct {
        INT16             tm_sec;           /* seconds; used for "IMMEDIATE" and "URGENT"
        specification */
        INT16             tm_min;           /* minutes */
        INT16             tm_hour;          /* hours */
        INT16             tm_mday;          /* day of month */
        INT16             tm_mon;           /* month */
        INT16             tm_year;          /* full year */
        } Date_time_type;

#define Send_time_type                  Date_time_type
#define IMMEDIATE            (-1)                /* Set into tm_sec of above struct */
#define URGENT               (-2)

/*
**        Error_type
**        The definition of the Error type does not include the text part of the error message; If the text
**        part is needed, it shall be treated as a private extension!
*/

typedef UINT32 Error_type;
```

```
/*
**        La_id_type
**        Reference of an LA-ID. Presented as a string.
**        The purpose of the parameter is to identify the "owning" LA of a request.
*/

typedef CHAR La_id_type[15+1];

/*
**        Layout_id_type
**        Defines the layout of the <CopyTDD> target file.
*/

typedef enum {std=0, csv, tab } Layout_id_type;

/*
**        Path_type
**        Full path addressing a file of a directory. Stands for full path to document, file or directory.
**        Full path means: path given absolute, without relative components.
*/

typedef CHAR Path_type[255+1];

/*
**        Printer_id_type
**        ID of selected printer. Represented as a string. Depends on the supporting operating system.
**        The CA manufacturer shall state in documentation how to address printers.
*/

typedef CHAR Printer_id_type[127+1];

/*
**        Req_id_type
**        Reference of a Request ID. The parameter value is represented as a string encoded as implied
**        by the APPLI/COM header ID of the TDD. The purpose of the parameter is to identify the
**        relation of a response to a previous request. So the REQ-ID shall be unique within an LA. It is the
**        responsibility of the LA to ensure the REQ-ID is unique.
*/

typedef CHAR_Req_id_type[31+1];

/*
**        Service_type
**        Specifies the ITU-T Service to be used.
*/

typedef enum { fx3=1, fx4, ttx, tlx, tx, email, ft } Service_type;

/*
**        State_type
**        Specifies the state of the CA-Record. See also Error! Reference source not found..
*/

typedef enum { delayed=1, sending, sent, send_failed, reception, retrieved, receive_failed } State_type;

/*
**        Status_type
*/

typedef enum {unknown, positive, partial, negative } Status_type;
```

```
/*
**      Subfunc_type
*/


typedef UINT32 Subfunc_type;


/*
**      Type_id_type
**      Specifies the Type (Subtype) of a  telecommunications service. Each telecommunications
**      service has its own set of Type-ids. See appropriate clause of Part Error! Reference source not
found. of this Recommendation.

*/


typedef enum{ std=0, btm, dtm, bft, edi, opd, md, ctl, teletex, g3fax, g4class1, videotex, message, bilateral,
        national, oda } Type_id_type;


/*
**      Userkey_type
*/


typedef CHAR_Userkey_type[31+1];
```

18.1.4 FX3 dependent TDD Structures, Constants and Definitions

```
/***
*       FX3: Service specific structure for <SendTDD>
***/


struct send_fx3_inp {
        G3_speed_type           g3speed;        /* <G3Speed> */
        Boolean_type            gencil;         /* <GenCil> */
        G3_high_res_type        highres;        /* <HighRes> */
        G3_sub_address_type     subaddr;        /* <SubAddress> */
        Boolean_type            useecm;         /* <UseEcm> */
        Address_type            address;        /* <Recipient> or <RecipientSpec> */
        Boolean_type            dopoll;         /* <DoPoll> */
        Poll_password_type      password;       /* <PollPassword> */
        Poll_select type        select;         /* <PollSelector> */
        Path_type               filename;       /* <Document> or <DocumentSpec> */
        Convert_id_type         convert;        /* <Convert> */
        Type_id_type            type;           /* <Type> */
        INT16                   from;           /* <From> */
        INT16                   to;             /* <To> */
        };


/***
*       FX3: Service specific structures for <SendAckTDD>
***/
```

```
struct sendack_fx3_inp {
        G3_speed_type           g3speed;        /* <G3Speed> */
        Boolean_type            gencil;         /* <GenCil> */
        G3_high_res_type        highres;        /* <HighRes> */
        G3_sub_address_type     subaddr;        /* <SubAddress> */
        Boolean_type            useecm;         /* <UseEcm> */
        Address_type            address;        /* <Recipient> */
        Boolean_type            dopoll;         /* <DoPoll> */
        Poll_password_type      password;       /* <PollPassword> */
        Poll_select_type        select;         /* <PollSelector> */
        Path_type               filename;       /* <Document> or <DocumentSpec> */
        Convert_id_type         convert;        /* <Convert> */
        Type_id_type            type;           /* <Type> */
        INT16                   from;           /* <From> */
        INT16                   to;             /* <To> */
        };


struct sendack_fx3_out {
        Address_type            address;        /* <Recipient> */
        G3_speed_type           g3speed;        /* <G3Speed> */
        Boolean_type            useecm;         /* <UseEcm> */
        };


/***
*       FX3: Service specific structures for <ReceiveTDD>
***/


struct receive_fx3_inp {
        Convert_id_type         cvfax3;         /* <CvFax3> */
        G3_sub_address_type     subaddr;        /* <SubAddress> */
        Path_type               filename;       /* <Document> */
        };


struct receive_fx3_out {
        Address_type            address;        /* <Originator> */
        G3_speed_type           g3speed;        /* <G3Speed> */
        G3_sub_address_type     subaddr;        /* <SubAddress> */
        Path_type               filename;       /* <Document> or <DocumentSpec> */
        Convert_id_type         convert;        /* <Convert> */
        Type_id_type            type;           /* <Type> */
        };


/***
*       FX3: Structures for poll <ExtendTDD>
***/


struct poll_fx3_inp {
        Address_type            address;        /* <Address> */
        Poll_password_type      password;       /* <PollPassword> */
        Poll_select_type        select;         /* <PollSelector> */
        Date_time_type          sendtime;       /* <SendTime> (optional) */
        };


struct poll_fx3_out {
        Comid_type              comid;          /* <ComId> */
        };
```

```
/***
*       Constants to be used for specific fields of the Send, Sendack and Receive TDD.
*       Use of some definitions made in 18.1.3 are restricted. This applies to the following
*       definitions:
*
*               Service_type is restricted to:     fx3.
*               Type_id_type is restricted to:     std, btm, dtm, bft and edi.
*               Convert_id_type:                           See Table 78 for further information.
***/


/*
**      G3_speed_type
*/


typedef enum {bps2400=1, bps4800, bps7200, bps9600, bps12200, bps14400} G3_speed_type;


/*
**      G3_sub_address_type
*/


typedef CHAR G3_sub_address_type[23+1];


/*
**      G3_high_res_type
*/


typedef enum {g3dpi98=0, g3dpi196, g3dpi200, g3dpi300, g3dpi400, g3dpi392_8, g3dpi392_16 }
        G3_high_res_type;


/*
**      Poll_password_type
*/


typedef CHAR Poll_password_type[23+1]; */ only first 20 digits are valid */


/*
**      Poll_select_type
*/


typedef CHAR_Poll_password_type[23+1]; */ only first 20 digits are valid */
```

18.1.5 FX4 dependent TDD Structures, Constants and Definitions

```
/***
*         FX4: Service specific structure for <SendTDD>
***/

struct send_fx4_inp {
          G4_high_res_type        highres;        /* <HighRes> */
          G4_sub_address_type     subaddr;        /* <SubAddress> */
          Address_type            address;        /* <Recipient> or <RecipientSpec> */
          Path_type               filename;       /* <Document> or <DocumentSpec> */
          Convert_id_type         convert;        /* <Convert> */
          Type_id_type            type;           /* <Type> */
          String12_type           name;           /* <Name> */
          String12_type           userinfo;       /* <UserInfo> */
          Path_type               prolog;         /* <Prolog> */
          INT16                   from;           /* <From> */
          INT16                   to;             /* <To> */
          };

/***
*         FX4: Service specific structures for <SendAckTDD>
***/

struct sendack_fx4_inp {
          G4_high_res_type        highres;        /* <HighRes> */
          G4_sub_address_type     subaddr;        /* <SubAddress> */
          Address_type            address;        /* <Recipient> */
          Path_type               filename;       /* <Document> or <DocumentSpec> */
          Convert_id_type         convert;        /* <Convert> */
          Type_id_type            type;           /* <Type> */
          String12_type           name;           /* <Name> */
          String12_type           userinfo;       /* <UserInfo> */
          Path_type               prolog;         /* <Prolog> */
          INT16                   from;           /* <From> */
          INT16                   to;             /* <To> */
          };

struct sendack_fx4_out {
          Cil_type                                cil;    /* <Cil> */
          };

/***
*         FX4: Service specific structures for <ReceiveTDD>
***/

struct receive_fx4_inp {
          Convert_id_type         cvfax4;         /* <CvFax4> */
          G4_sub_address_type     subaddr;        /* <SubAddress> */
          Path_type               filename;       /* <Document> */
          Path_type               prolog;         /* <Prolog> */
          };

struct receive_fx4_out {
          Address_type            address;        /* <Originator> */
          G4_sub_address_type     subaddr;        /* <SubAddress> */
          Path_type               filename;       /* <Document> or <DocumentSpec> */
          Convert_id_type         convert;        /* <Convert> */
          Type_id_type            type;           /* <Type> */
          String12_type           name;           /* <Name> */
          String12_type           userinfo;       /* <UserInfo> */
          Cil_type                                cil;    /* <Cil> */
          INT16                   firstpg;        /* <FirstPg> */
          };
```

```
/***
*	Constants to be used for specific fields of the Send, Sendack and Receive TDD.
*	Use of some definitions made in 18.1.3 are restricted. This applies to the following
*	definitions:
*
*		Service_type is restricted to:	fx4.
*		Type_id_type is restricted to:	std, dtm, bft, edi, opd, md, and ctl.
*		Convert_id_type:			See Table 87 for further information.
***/

/*
**	String12_type
*/

typedef CHAR String12_type[15+1];

/*
**	G4_sub_address_type
*/

typedef CHAR G4_sub_address_type[7+1];

/*
**	G4_high_res_type
*/

typedef enum {g4dpi200=1, g4dpi240, g4dpi300, g4dpi400} G4_high_res_type;

/*
**	Cil_type
*/

typedef struct {
	CHAR	receiver_tid[24];
	CHAR	sep1;		/* = '/' (slash) */
	CHAR	sender_tid[24];
	CHAR	sep2;		/* = '/' (slash) */
	CHAR	datetime[14];
	CHAR	sep3;		/* = '/' (slash) */
	CHAR	refinfo[7];
	} Cil_type;		/* There is no terminating 0x00; Total length = 72 */
```

18.1.6 TTX dependent TDD Structures, Constants and Definitions

```
/***
*	TTX: Service specific structure for <SendTDD>
***/

struct send_ttx_inp {
	Ttx_sub_address_type	subaddr;	/* <SubAddress> */
	Address_type		address;	/* <Recipient> or <RecipientSpec> */
	Path_type		filename;	/* <Document> or <DocumentSpec> */
	Convert_id_type		convert;	/* <Convert> */
	Type_id_type		type;		/* <Type> */
	String12_type		name;		/* <Name> */
	String12_type		userinfo;	/* <UserInfo> */
	Path_type		prolog;		/* <Prolog> */
	INT16			from;		/* <From> */
	INT16			to;		/* <To> */
	T61_options_type	t61options;	/* <T61Options> */
	};
```

```
/***
*           TTX: Service specific structures for <SendAckTDD>
***/




struct sendack_ttx_inp {
            Ttx_sub_address_type      subaddr;         /* <SubAddress> */
            Address_type              address;         /* <Recipient> */
            Path_type                 filename;        /* <Document> or <DocumentSpec> */
            Convert_id_type           convert;         /* <Convert> */
            Type_id_type              type;            /* <Type> */
            String12_type             name;            /* <Name> */
            String12_type             userinfo;        /* <UserInfo> */
            Path_type                 prolog;          /* <Prolog> */
            INT16                     from;            /* <From> */
            INT16                     to;              /* <To> */
            T61_options_type};        t61options;      /* <T61Options> */
            };




struct sendack_ttx_out {
            Cil_type                                   cil;     /* <Cil> */
            };




/***
*           TTX: Service specific structures for <ReceiveTDD>
***/




struct receive_ttx_inp {
            Convert_id_type           cvttx;           /* <CvTtx> */
            Ttx_sub_address_type      subaddr;         /* <SubAddress> */
            Path_type                 filename;        /* <Document> */
            Path_type                 prolog;          /* <Prolog> */
            };




struct receive_ttx_out {
            Address_type              address;         /* <Originator> */
            Ttx_sub_address_type      subaddr;         /* <SubAddress> */
            Path_type                 filename;        /* <Document> or <DocumentSpec> */
            Convert_id_type           convert;         /* <Convert> */
            Type_id_type              type;            /* <Type> */
            String12_type             name;            /* <Name> */
            String12_type             userinfo;        /* <UserInfo> */
            Cil_type                                   cil;     /* <Cil> */
            INT16                     firstpg;         /* <FirstPg> */
            };
```

```
/***
*       Constants to be used for specific fields of the Send, Sendack and Receive TDD.
*       Use of some definitions made in 18.1.3 are restricted. This applies to the following
*       definitions:
*
*               Service_type is restricted to:      ttx.
*               Type_id_type is restricted to:      std, dtm, bft and edi.
*               Convert_id_type:                            See Table 94 for further information.
***/

/*
**      String12_type
**      See definition of FX4!
*/
/*
typedef CHAR String12_type[15+1];
*/

/*
**      Ttx_sub_address_type
*/
typedef CHAR Ttx_sub_address_type[7+1];
/*
**      Cil_type
**      See definition of FX4!
*/
/*
typedef struct {
        CHAR  receiver_tid[24];
        CHAR  sep1              /* = '/' (slash) */
        CHAR  sender_tid[24];
        CHAR  sep2              /* = '/' (slash) */
        CHAR  datetime[14];
        CHAR  sep3              /* = '/' (slash) */
        CHAR  refinfo[7];
        } Cil_type;
*/
/*
**      T61_options_type
*/

typedef CHAR T61_options_type[15+1];
```

18.1.7 TX dependent TDD Structures, Constants and Definitions

```
/***
*       TX: Service specific structure for <SendTDD>
***/

struct send_tx_inp {
        Boolean_type            notify;         /* <Notify> */
        Tx_sub_address_type     subaddr;        /* <SubAddress> */
        Address_type            address;        /* <Recipient> or <RecipientSpec> */
        Path_type               filename;       /* <Document> or <DocumentSpec> */
        Convert_id_type         convert;        /* <Convert> */
        Type_id_type            type;           /* <Type> */
        INT16                   from;           /* <From> */
        INT16                   to;             /* <To> */
        };
```

```
/***
*       TX: Service specific structures for <SendAckTDD>
***/


struct sendack_tx_inp {
        Boolean_type            notify;         /* <Notify> */
        Tx_sub_address_type     subaddr;        /* <SubAddress> */
        Address_type            address;        /* <Recipient> */
        Path_type               filename;       /* <Document> or <DocumentSpec> */
        Convert_id_type         convert;        /* <Convert> */
        Type_id_type            type;           /* <Type> */
        INT16                   from;           /* <From> */
        INT16                   to;             /* <To> */
        };


struct sendack_tx_out {};


/***
*       TX: Service specific structures for <ReceiveTDD>
***/


struct receive_tx_inp {
        Convert_id_type         cvtx;           /* <CvTx> */
        Tx_sub_address_type     subaddr;        /* <SubAddress> */
        Path_type               filename;       /* <Document> */
        };


struct receive_tx_out {
        Address_type            address;        /* <Originator> */
        Tx_sub_address_type     subaddr;        /* <SubAddress> */
        Path_type               filename;       /* <Document> or <DocumentSpec> */
        Convert_id_type         convert;        /* <Convert> */
        Type_id_type            type;           /* <Type> */
        };


/***
*       Constants to be used for specific fields of the Send, Sendack and Receive TDD.
*       Use of some definitions made in 18.1.3 are restricted. This applies to the following
*       definitions:
*
*               Service_type is restricted to:      tx.
*               Type_id_type is restricted to:      std
*               Convert_id_type is restricted to: "ASCII", "ASCIIxxx", "T.50"
***/


/*
**      Tx_sub_address_type
*/


typedef CHAR Tx_sub_address_type[7+1];
```

18.1.8 TLX dependent TDD Structures, Constants and Definitions

```
/***
*	TLX: Service specific structure for <SendTDD>
***/



struct send_tlx_inp {
	Tlx_sub_address_type	subaddr;	/* <SubAddress> */
	Address_type		address;	/* <Recipient> or <RecipientSpec> */
	Path_type		filename;	/* <Document> or <DocumentSpec> */
	Convert_id_type		convert;	/* <Convert> */
	Type_id_type		type;		/* <Type> */
	INT16			from;		/* <From> */
	INT16			to;		/* <To> */
	};



/***
*	TLX: Service specific structures for <SendAckTDD>
***/



struct sendack_tlx_inp {
	Tlx_sub_address_type	subaddr;	/* <SubAddress> */
	Address_type		address;	/* <Recipient> */
	Path_type		filename;	/* <Document> or <DocumentSpec> */
	Convert_id_type		convert;	/* <Convert> */
	Type_id_type		type;		/* <Type> */
	INT16			from;		/* <From> */
	INT16			to;		/* <To> */
	};



struct sendack_tlx_out {};



/***
*	TLX: Service specific structures for <ReceiveTDD>
***/



struct receive_tlx_inp {
	Convert_id_type		cvtlx;		/* <CvTx> */
	Tlx_sub_address_type	subaddr;	/* <SubAddress> */
	Path_type		filename;	/* <Document> */
	};
```

```
struct receive_tlx_out {
        Address_type            address;        /* <Originator> */
        Tlx_sub_address_type    subaddr;        /* <SubAddress> */
        Path_type               filename;       /* <Document> or <DocumentSpec> */
        Convert_id_type         convert;        /* <Convert> */
        Type_id_type            type;           /* <Type> */
        };

/***
*       Constants to be used for specific fields of the Send, Sendack and Receive TDD.
*       Use of some definitions made in 18.1.3 are restricted. This applies to the following
*       definitions:
*
*               Service_type is restricted to:      tlx.
*               Type_id_type is restricted to:      std
*               Convert_id_type is restricted to: "ASCII", "ASCIIxxx", "T.50"
***/

/*
**      Tlx_sub_address_type
*/

typedef CHAR Tlx_sub_address_type[7+1];
```

18.1.9 EMAIL dependent TDD Structures, Constants and Definitions

```
/***
*       EMAIL: Service specific structure for <SendTDD>
***/

struct send_email_inp {
        Ipm_id_type             ipm_id;         /* <IpmId> */
        Path_type               s_recipient;    /* <S-Recipient> */
        Boolean_type            alternate;      /* <Alternate> (optional) */
        Cont_type_type          cont_type;      /* <ContType> (optional) */
        Boolean_type            disclo_rec;     /* <DiscloRec> (optional) */
        Date_time_type          expirytime;     /* <ExpiryTime> (optional) */
        Boolean_type            implicit_conv;  /* <ImplicitConv> (optional) */
        Importance_type         importance;     /* <Importance> (optional) */
        Language_id_type        language;       /* <Language> (optional) */
        Priority_type           priority;       /* <Priority> (optional) */
        Path_type               related;        /* <Related> (optional) */
        Ipm_id_type             reply_id;       /* <ReplyId> (optional) */
        Date_time_type          replytime;      /* <ReplyTime> (optional) */
        Path_type               s_originator;   /* <S-Originator> (optional) */
        Sensitivity_type        sensitivity;    /* <Sensitivity> (optional) */
        Subject_type            subject;        /* <Subject> (optional) */
        Userinfo_type           userinfo;       /* <UserInfo> (optional) */
        };

/***
*       EMAIL:Service specific structures for <SendAckTDD>
***/

struct sendack_email_inp {
        Ipm_id_type             ipm_id;         /* <IpmId> */
        Path_type               s_recipient;    /* <S-Recipient> */
        Boolean_type            alternate;      /* <Alternate> (optional) */
        Cont_type_type          cont_type;      /* <ContType> (optional) */
        Boolean_type            disclo_rec;     /* <DiscloRec> (optional) */
        Date_time_type          expirytime;     /* <ExpiryTime> (optional) */
        Boolean_type            implicit_conv;  /* <ImplicitConv> (optional) */
        Importance_type         importance;     /* <Importance> (optional) */
        Language_id_type        language;       /* <Language> (optional) */
        Priority_type           priority;       /* <Priority> (optional) */
        Path_type               related;        /* <Related> (optional) */
        Ipm_id_type             reply_id;       /* <ReplyId> (optional) */
        Date_time_type          replytime;      /* <ReplyTime> (optional) */
        Path_type               s_originator;   /* <S-Originator> (optional) */
        Sensitivity_type        sensitivity;    /* <Sensitivity> (optional) */
        Subject_type            subject;        /* <Subject> (optional) */
        Userinfo_type           userinfo;       /* <UserInfo> (optional) */
        };


struct sendack_email_out {
        Msg_sub_id_type         msg_sub_id;     /* <MsgSubId> (optional) */
        Date_time_type          submittime;     /* <SubmitTime> (optional) */
        };


/***
*       EMAIL:Service specific structures for <ReceiveTDD>
***/


struct receive_email_inp {
        Path_type               r_originator;   /* <R-Originator> */
        Path_type               r_recipient;    /* <R-Recipient> */
        Path_type               related;        /* <Related> (optional) */
        };


struct receive_email_out {
        Ipm_id_type             ipm_id;         /* <IpmId> */
        Cont_type_type          cont_type;      /* <ContType> */
        Date_time_type          expirytime;     /* <ExpiryTime> */
        Boolean_type            forwarded;      /* <Forwarded> */
        Importance_type         importance;     /* <Importance> */
        Boolean_type            inc_copy;       /* <IncCopy> */
        Language_id_type        language;       /* <Language> */
        Priority_type           priority;       /* <Priority> */
        Ipm_id_type             reply_id;       /* <ReplyId> */
        Date_time_type          replytime;      /* <ReplyTime> */
        Sensitivity_type        sensitivity;    /* <Sensitivity> */
        Subject_type            subject;        /* <Subject> */
        Date_time_type          submittime;     /* <SubmitTime> */
        Userinfo_type           userinfo;       /* <UserInfo> */
        };


/***
*       EMAIL: Structure of file-record of R-Recipientspec
***/


struct r_recipientspec {
        X_name_type             x_name;
        R_recipient_type        r_type;
        Boolean_type            reply;
        Notify_type             notify;
        Report_type             report;
        };
```

```
/***
*         EMAIL: Structure of file-record of S-Recipientspec
***/

struct s_recipientspec {
          X_name_type                 x_name;
          S_recipient_type            s_type;
          Boolean_type                reply;
          Notify_type                 notify;
          Report_type                 report;
          };

/***
*         EMAIL: Structure of file-record of Originatorspec
***/

struct originatorspec {
          X_name_type                 x_name;
          Originator_type             o_type;
          };

/***
*         EMAIL: Structure of file-record of Relatedspec
***/

struct relatedspec {
          Ipm_id_type                 ipm_id;
          Relation_type               relation;
          };

/***
*         Constants to be used for specific fields of the Send, Sendack and Receive TDD.
*         Use of some definitions made in 18.1.3 are restricted. This applies to the following
*         definitions:
*
*                   Service_type is restricted to:    email.
*                   Type_id_type is restricted to:    std, teletex, g3fax, g4class1, videotex, message,
*                                                     bilateral, national
*                   Convert_id_type:                          See Table 113 for further information.
***/

/*
**        Cont_type_type
*/

typedef enum { mhsIPM84=0, mhsIPM88, mhsIPN84, mhsIPN88, mhsDR, cmcIPM, cmcIPN, cmcDR}
          Cont_type_type;

/*
**        Importance_type
*/

typedef enum { low = 0, normal, high } Importance_type;

/*
**        Ipm_id_type
*/

typedef struct {
          BYTE   id[63+1];
          BYTE   oraddress[511+1];
          } Ipm_id_type;
```

```
/*
**        Language_id_type
*/

typedef CHAR   Language_id_type[5+1];

/*
**        Msg_sub_id_type
*/

typedef CHAR   Msg_sub_id_type[67+1];

/*
**        Priority_type
*/

typedef enum { standard=0, nonurgent, urgent } Priority_type;

/*
**        Relation_type
*/

typedef enum { reference=0, obsolete } Relation_type;

/*
**        Subject_type
*/

typedef CHAR   Subject_type[127+1];

/*
**        Sensitivity_type
*/

typedef enum { none=0, personal, privateonly, companyconfidential } Sensitivity_type;

/*
**        Userinfo_type
*/

typedef CHAR   Userinfo_type[15+1];

/*
**        X_name_type
*/

typedef struct {
        Boolean_type              is_alias;
        CHAR                      addr_or_alias[511+1];
        } X_name_type;

/*
**        S_recipient_type
*/

typedef enum { sndprimary=1, sndcopy, sndblind } S_recipient_type;
```

```
/*
**       R_recipient_type
*/

typedef enum { recprimary=1, reccopy, recblind, recintended } R_recipient_type;

/*
**       Notify_type
*/

typedef enum { nonotify=1, notreceived, received, explicit } Notify_type;

/*
**       Report_type
*/

typedef enum { noreport=1, basic, confirmed } Report_type;

/*
**       Originator_type
*/

typedef enum { authorizing=1, originator, replyto } Originator_type;
```

18.1.10 FT dependent TDD Structures, Constants and Definitions

```
/***
*        FT: Service specific structure for <SendTDD>
***/

struct send_ft_inp {
         Environ_type        environ;       /* <Environ> */
         Password_type       password;      /* <Password> (optional) */
         Address_type        address;       /* <Recipient> or <RecipientSpec> */
         Path_type           filename;      /* <Document> or <DocumentSpec> */
         Convert_id_type     convert;       /* <Convert> */
         Type_id_type        type;          /* <Type> */
         Compress_type       compress;      /* <Compress> */
         Name_type           name;          /* <Name> (optional) */
         };

/***
*        FT: Service specific structures for <SendAckTDD>
***/

struct sendack_ft_inp {
         Environ_type        environ;       /* <Environ> */
         Password_type       password;      /* <Password> (optional) */
         Address_type        address;       /* <Recipient> */
         Path_type           filename;      /* <Document> or <DocumentSpec> */
         Convert_id_type     convert;       /* <Convert> */
         Type_id_type        type;          /* <Type> */
         Compress_type       compress;      /* <Compress> */
         Name_type           name;          /* <Name> (optional) */
         };

struct sendack_ft_out {};
```

```
/***
*         FT: Service specific structures for <ReceiveTDD>
***/


struct receive_ft_inp {
          Password_type           password;       /* <Password> (optional) */
          Path_type               filename;       /* <Document>  (optional) */
          };


struct receive_ft_out {
          Environ_type            environ;        /* <Environ> */
          Address_type            address;        /* <Originator> */
          Path_type               filename;       /* <Document> or <DocumentSpec> */
          Convert_id_type         convert;        /* <Convert> */
          Type_id_type            type;           /* <Type> */
          Compress_type           compress;       /* <Compress> */
          Name_type               name;           /* <Name> */
          };


/***
*         Constants to be used for specific fields of the Send, Sendack and Receive TDD.
*         Use of some definitions made in18.1.3 are restricted. This applies to the following
*         definitions:
*
*                   Service_type is restricted to:    ft
*                   Type_id_type is restricted to:    std
*                   Convert_id_type:                          std, void.
***/


/*
**        Compress_type
*         See also 17.2.2.6
*/


typedef CHAR Compress_type[7+1];


/*
**        Environ_type
*/


typedef enum { msdos=1, windows, unix, os2, macos } Environ_type;


/*
**        Password_type
*/


typedef CHAR Password_type[15+1];


/*
**        Name_type
*/


typedef CHAR  Name_type[15+1];
```

PART IV – PLATFORM DEPENDENCIES

19 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 Primitive Exchange Method (see 7.3.1). The Primitive Exchange Method provides direct access to the Basic Exchange Method functions described in 7.1. subclause 19.3 describes the implementation of the Primitive Exchange Method for each platform.

The implementation of the Primitive Exchange Method is optional for some platforms, since the more platform independent file exchange method might have been be chosen as default exchange method for specific platforms. subclause 19.2 lists the default assignements for the different platforms.

The binary encoding scheme provided in Part III of this Recommendation is *not* platform dependent. However, since for the description of this encoding a generic C-language has been used, the mapping of the generic C data types to the real data types used for a specific platform is given first in 19.1 below.

19.1 Mapping of Binary Coded TDD Data Types

See Table 125.

TABLE 125/T.611

Mapping of generic C data types for the various platforms

Platform	Real C type corresponding to generic C data type			
	INT16	UINT32	CHAR	BYTE
MS-DOS	short int	unsigned long int	char	unsigned char
WINDOWS	short int	dword	char	byte
Unix	short int	unsigned long int	char	unsigned char
OS/2	short int	unsigned long int	char	unsigned char
MacOS	For further study			

19.2 Default Exchange Method

See Table 126.

TABLE 126/T.611

Default Exchange Method assignements for the various platforms

Platform	Default Exchange Method
MS-DOS	File Exchange Method
WINDOWS	Primitive Exchange Method
Unix	For further study
OS/2	For further study
MacOS	For further study

19.3 Implementation of Primitive Exchange Method

The implementation of the Primitive Exchange Method shall be achieved by providing access to the Basic Exchange Method functions, which are described in general in 7.1. To ensure binary compatibility across a specific platform, the rules, data types and function prototypes specified in following subclauses for each platform shall be strictly respected.

To perform properly, the implementation has to rely on certain data structures and constants. The data structures and constants common to all implementations are shown below in C language description:

```
/*
**      This generic C language description uses certain generic types which are mapped to
**      real data types in the platform dependent descriptions further below.
**      The generic types used are: NUMBER, BUFPOINTER, LISTPOINTER and WORD.
*/


/***
*       The structure bem_data_buffer describes a buffer.
*       The types NUMBER and BUFPOINTER have to be mapped to each platform
***/


struct bem_data_buffer {
        NUMBER                  buffer_size;
        BUFPOINTER              buffer;
        };


/***
*       The structure bem_data_files describes a list of filenames.
*       The types NUMBER and LISTPOINTER have to be mapped to each platform
***/


struct bem_data_files {
        NUMBER                  number_of_files;
        LISTPOINTER             file_name[ ];
        };


/***
*       The structure bem_data_descriptor describes the data conveyed between LAs and CAs
*       (data files, address lists etc.). The data may be in files and/or in memory. The structure
*       bem_data_descriptor accounts for these situations.
*
*       When data are in memory, the structure bem_data_descriptor allows to actually convey that
*       data through the EPutData or EGetData functions. When data are in files, bem_data_descriptor
*       describes the actual file names that contain that data, i.e., the structure does not contain the data
*       itself.
*
*       The type WORD denotes a 16 bit unsigned integer, which has to be mapped to each platform
***/


struct bem_data_descriptor {
        WORD                    key;            /* key = 0 for data in buffer */
                                                /* key = 1 for data in file */
        WORD                    type;           /* type = 1 for "document lists" */
                                                /* type = 2 for "recipient lists", etc.*/
                                                /* used only in SEND, SENDACK and RECEIVE */
                                                /* ignored otherwise */
        union {
                struct bem_data_buffer buffer;  /* data in memory buffer */
                struct bem_data_files file;     /* data in file */
                } bof;                          /* buffer or file */
        };
```

```
#define BEM_DATA        struct bem_data_descriptor


/* Constants used for the "key" field of BEM_DATA */
#define   BUFFER_KEY              0          /* key: data in buffer */
#define   FILE_KEY                1          /* key: data in file */


/* Constants used for the "type" field of BEM_DATA */
#define   DOCUMENT_TYPE           1          /* type: document list */
#define   RECIPIENT_TYPE          2          /* type: recipient list (= address list) */
#define   ORIGINATOR_TYPE         3          /* type: originator list */
#define   RELATED_TYPE            4          /* type: related list */


/* Constants used for TDD Types, see also 7.2.4.2 */
#define   NO_TDD_RESPONSE                    0x00
#define   SENDACK_RESPONSE                   0x10
#define   RECEIVE_RESPONSE                   0x20
#define   COPY_RESPONSE                      0x30
#define   DELETE_RESPONSE                    0x31
#define   CANCEL_RESPONSE                    0x32
#define   PURGE_RESPONSE                     0x33
#define   RESCHEDULE_RESPONSE                0x34
#define   DISPATCH_RESPONSE                  0x35
#define   PREVIEW_RESPONSE                   0x36
#define   PRINT_RESPONSE                     0x40
#define   CONVERT_RESPONSE                   0x41
#define   CHECK_RESPONSE                     0x42
#define   EXTEND_RESPONSE                    0x50
#define   NATIONAL_RESPONSE                  0x60
#define   PRIVATE_RESPONSE                   0x70


/* Constants used for Alarm_types (may be or'ed), see also 7.2.7.1 */
#define   ASYNC_RESPONSES                    0x0001
#define   QUEUE_FULL                         0x0002
#define   DOCUMENT_RECEIVED                  0x0004
#define   CONNECTION_LOST                    0x0008
#define   SEND_SUCCESS                       0x0010
#define   SEND_FAILED                        0x0020
#define   CORRUPTED_TDD                      0x0040
#define   SEND_EVENT                         0x0080
#define   RECEIVE_EVENT                      0x0100
#define   CA_WILL_STOP                       0x0200
#define   ALARMS_UNAVAILABLE                 0x0400
#define   TDD_RESP_AVAILABLE                 0x0800
```

NOTE – Handling the Data-ID parameter in function EPutTDD for input and output. On input: the LA sets a non-NULL address if the LA needs to transfer data files in subsequent EPutData function calls. If the LA needs not to transfer files, then this parameter is set to NULL. On Output: if the input parameter was set to NULL, the CA does not change that value. Otherwise the CA computes a Data-ID and returns it at the address specified in the input.

19.3.1 MSDOS

Access to the Basic Exchange Method shall occur through a DOS interrupt mechanism. The following rules shall be respected by the calling LA:

– the interrupt number used by the CA shall be taken from the ICE (CA-Descriptor);

– if a multiplex scheme is used, the multiplex number shall be taken from the ICE (CA-Descriptor) and loaded into register AH before call;

– all parameters shall be transferred to the CA by pushing them onto the caller's stack. The stack shall have at least 256 bytes of headroom when calling;

– the parameters shall be pushed using C-call conventions (from right to left), pointers shall be pushed as double-word addresses (far pointers);

– the addressing of a desired function shall be achieved through a function code, also pushed onto the caller's stack as leftmost parameter;

– the return value shall be passed in the AX register;

– the segment registers (as well as the stack-pointer) shall be preserved by the CA during call; all other registers may be destroyed by the CA;

– the stack cleanup (popping of the calling parameters) shall be performed by the caller after function return (C-language calling convention).

To address the various functions the function codes defined in Table 127 shall be used.

TABLE 127/T.611

DOS function codes

Basic Exchange Method Function	Function code
ELogin ( )	1
EPutTDD ( )	2
EPutData ( )	3
EPollTDD ( )	4
EGetTDD ( )	5
EGetData ( )	6
ESetAlarm ( )	7
EAbortData ( )	8
ELogout ( )	9

19.3.1.1 C Function Prototypes and Definitions

```
#define STATUS          short int

#define BEM_DATA        struct bem_data_descriptor

struct bem_data_buffer {
        unsigned long           buffer_size;        /* = generic type NUMBER */
        unsigned char far *     buffer;             /* = generic type BUFPOINTER */
        };

struct bem_data_files {
        unsigned long           number_of_files;    /* = generic type NUMBER */
        char far *              file_name[ ];       /* = generic type LISTPOINTER */
        };
```

```
struct bem_data_descriptor {
        unsigned short int          key;                    /* = generic type WORD */
        unsigned short int          type;                   /* = generic type WORD */
        union {
                struct bem_data_buffer buffer;
                struct bem_data_files file;
                } bof;
        };

***/

*       ELogin ( )
*
*       Stack:  SP+18       | Connection_ID |   far pointer
*               SP+14       | CA_ID         |   far pointer
*               SP+10       | Selector      |   far pointer
*               SP+6        | Password      |   far pointer
*               SP+2        | Login_name    |   far pointer
*               SP -->      | 0x0001        |   word value; function code to identify ELogin ( )
***/

STATUS          ELogin (
                short int           function_code,          /* ELogin = 0x0001 */
                char far *          Login_name,
                char far *          Password,
                char far *          Selector,
                short int far *     CA_ID,
                short int far *     Connection_ID
                );


***/

*       EPutTDD ( )
*
*       Stack:  SP+12       | Data_ID       |   far pointer
*               SP+10       | TDD_size      |   word value
*               SP+6        | TDD_location  |   far pointer
*               SP+4        | CA_ID         |   word value
*               SP+2        | Connection_ID |   word value
*               SP -->      | 0x0002        |   word value; function code to identify EPutTDD ( )
*
*       If the address of Data_ID is set to NULL on call, CA will not provide the Data_ID and process
*       the TDD immediately, assuming the pathes given in the TDD are correct
***/

STATUS          EPutTDD (
                short int           function_code,          /* EPutTDD = 0x0002 */
                short int           Connection_ID,
                short int           CA_ID,
                unsigned char far * TDD_location,
                short int           TDD_size,
                unsigned long far * Data_ID
                );
```

```
***/
*          EPutData ( )
*
*          Stack:  SP+14      | Next          | word value
*                  SP+10      | Data          | far pointer
*                  SP+6       | Data_ID       | dword value
*                  SP+4       | CA_ID         | word value
*                  SP+2       | Connection_ID | word value
*                  SP -->     | 0x0003        | word value; function code to identify EPutData ( )
***/

STATUS          EPutData (
                short int              function_code,          /* EPutData = 0x0003 */
                short int              Connection_ID,
                short int              CA_ID,
                unsigned long          Data_ID,
                BEM_DATA far *         Data,
                short int              Next
                );

***/
*          EPollTDD ( )
*
*          Stack:  SP+14      | TDD_count     | far pointer
*                  SP+10      | TDD_size      | far pointer
*                  SP+6       | TDD_type      | far pointer
*                  SP+4       | CA_ID         | word value
*                  SP+2       | Connection_ID | word value
*                  SP -->     | 0x0004        | word value; function code to identify EPollTDD ( )
***/

STATUS          EPollTDD (
                short int              function_code,          /* EPollTDD = 0x0004 */
                short int              Connection_ID,
                short int              CA_ID,
                unsigned short far *   TDD_type,
                short int far *        TDD_size,
                short int far *        TDD_count
                );

***/
*          EGetTDD ( )
*
*          Stack:  SP+14      | Data_ID       | far pointer
*                  SP+10      | TDD_size      | far pointer
*                  SP+6       | TDD_location  | far pointer
*                  SP+4       | CA_ID         | word value
*                  SP+2       | Connection_ID | word value
*                  SP -->     | 0x0005        | word value; function code to identify EGetTDD ( )
***/
```

```
STATUS          EGetTDD (
                short int               function_code,          /* EGetTDD = 0×0005 */
                short int               Connection_ID,
                short int               CA_ID,
                unsigned char far *     TDD_location,
                short int far *         TDD_size,
                unsigned long far *     Data_ID
                );

***/
*       EGetData ( )
*
*       Stack:  SP+14    | Next          | far pointer
*               SP+10    | Data          | far pointer
*               SP+6     | Data_ID       | dword value
*               SP+4     | CA_ID         | word value
*               SP+2     | Connection_ID | word value
*               SP -->   | 0x0006        | word value; function code to identify EGetData ( )
***/

STATUS          EGetData (
                short int               function_code,          /* EGetData = 0x0006 */
                short int               Connection_ID,
                short int               CA_ID,
                unsigned long           Data_ID,
                BEM_DATA far *          Data,
                short int far *         Next
                );

***/
*       ESetAlarm ( )
*
*       Stack:  SP+8     | Alarm_handler | far pointer
*               SP+6     | Alarm_event   | word value
*               SP+4     | CA_ID         | word value
*               SP+2     | Connection_ID | word value
*               SP -->   | 0x0007        | word value; function code to identify ESetAlarm ( )
***/

void            FAR AlarmHandler (
                short int               Connection_ID,
                short int               CA_ID,
                unsigned short          Alarm_type,
                unsigned char far *     Parameter,              /* additional parameter buffer */
                short int               Length                  /* length of parameter buffer */
                );

STATUS          ESetAlarm (
                short int               function_code,          /* ESetAlarm = 0x0007 */
                short int               Connection_ID,
                short int               CA_ID,
                unsigned short          Alarm_event,
                void (* far             Alarm_handler) ( )
                );
```

```
***/
*        EAbortData ( )
*
*        Stack:  SP+6      | Data_ID        | dword value
*                SP+4      | CA_ID          | word value
*                SP+2      | Connection_ID  | word value
*                SP -->    | 0x0008         | word value; function code to identify EAbortData ( )
***/

STATUS      EAbortData (
            short int         function_code,      /* EAbortData = 0x0008 */
            short int         Connection_ID,
            short int         CA_ID,
            unsigned long     Data_ID
            );

***/
*        ELogout ( )
*
*        Stack:  SP+4      | CA_ID          | word value
*                SP+2      | Connection_ID  | word value
*                SP -->    | 0x0009         | word value; function code to identify ELogout ( )
***/

STATUS      ELogout (
            short int         function_code,      /* ELogout = 0x0009 */
            short int         Connection_ID,
            short int         CA_ID
            );
```

19.3.2 WINDOWS

To access the Basic Exchange Method functions, the CA provider shall offer a Dynamic Link Library (DLL), which opposes the Exchange Method functions described in the following subclause to the LA.

The Name of the DLL shall be stated at the appropriate place in the ICE by the configurator of the target platform. 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 parameter.

19.3.2.1 C Function Prototypes and Definitions

```
#include <windows.h>

#define BEM_DATA    struct bem_data_descriptor

struct bem_data_buffer {
        int                  buffer_size;        /* = generic type NUMBER */
        HANDLE               buffer;             /* = generic type BUFPOINTER */
        };

struct bem_data_files {
        int                  number_of_files;    /* = generic type NUMBER */
        LPSTR                file_name[1];       /* = generic type LISTPOINTER */
        };
```

```
struct bem_data_descriptor {
            word                key;            /* = generic type WORD */
            word                type;           /* = generic type WORD */
            union {
                        struct bem_data_buffer buffer;
                        struct bem_data_files file;
                        } bof;
            };

/***
*           ELogin ( )
***/

void FAR PASCAL ELogin (
                        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 */
                        );

/***
*           EPutTDD ( )
*
*           If the address of Data_ID is set to NULL on call, CA will not provide the Data_ID and process
*           the TDD immediately, assuming the pathes given in the TDD are correct
***/

void FAR PASCAL EPutTDD (
                        int             Connection_ID,  /* Connection ID */
                        int             CA_ID,          /* CA Identifier */
                        LPSTR           TDD_location,   /* TDD buffer text */
                        int             TDD_size,       /* size of the TDD text */
                        int far *       Data_ID,        /* Data identifier */
                        int far *       Status          /* 0 success, 1 fail */
                        );

/***
*           EPutData ( )
***/

void FAR PASCAL EPutData (
                        int             Connection_ID,  /* Connection ID */
                        int             CA_ID,          /* CA Identifier */
                        int             Data_ID,        /* Data identifier */
                        BEM_DATA far *  Data,           /* Data descriptor */
                        int             Next,           /* next boolean */
                        int far *       Status          /* 0 success, 1 fail */
                        );

/***
*           EPollTDD ( )
***/
```

```
void FAR PASCAL EPollTDD (
                int             Connection_ID,      /* Connection ID */
                int             CA_ID,              /* CA Identifier */
                word far *      TDD_type,           /* Type of next TDD */
                int far *       TDD_size,           /* Size of next TDD */
                int far *       TDD_count,          /* Count of waiting TDDs */
                int far *       Status              /* 0 success, 1 fail */
                );


/***

*       EGetTDD ( )

***/


void FAR PASCAL EGetTDD (
                int             Connection_ID,      /* Connection ID */
                int             CA_ID,              /* CA Identifier */
                LPSTR           TDD_location,       /* TDD buffer text */
                int far *       TDD_size,           /* size of the TDD text */
                int far *       Data_ID,            /* Data identifier */
                int far *       Status              /* 0 success, 1 fail */
                );


/***

*       EGetData ( )

***/


void FAR PASCAL EGetData (
                int             Connection_ID,      /* Connection ID */
                int             CA_ID,              /* CA Identifier */
                int             Data_ID,            /* Data identifier */
                BEM_DATA far *  Data,               /* Data descriptor */
                int far *       Next,               /* next boolean */
                int far *       Status              /* 0 success, 1 fail */
                );


/***

*       ESetAlarm ( )

***/


void FAR PASCAL AlarmHandler (
                int             Connection_ID,      /* Connection ID */
                int             CA_ID,              /* CA Identifier */
                word            Alarm_type,         /* Alarm type */
                HANDLE          Parameter,          /* additional parameter buffer */
                int             Length              /* length of parameter buffer */
                );

void FAR PASCAL ESetAlarm (
                int             Connection_ID,      /* Connection ID */
                int             CA_ID,              /* CA Identifier */
                word            Alarm_type,         /* Alarm event */
                FARPROC         Alarm_handler,      /* Alarm handler entry */
                int far *       Status              /* 0 success, 1 fail */
                );
```

```
/***

*           EAbortData ( )

***/


void FAR PASCAL EAbortData (
                int                 Connection_ID,          /* Connection ID */
                int                 CA_ID,                  /* CA Identifier */
                int                 Data_ID,                /* Data identifier */
                int far *           Status                  /* 0 success, 1 fail */
                );


/***

*           ELogout ( )

***/


void FAR PASCAL ELogout (
                int                 Connection_ID,          /* Connection ID */
                int                 CA_ID,
                int far *           Status                  /* 0 success, 1 fail */
                );
```

19.3.3 UNIX

For further study.

19.3.4 OS/2

For further study.

19.3.5 MacOS

For further study.

Annex A

Syntax for Presentation and Encoding

(This annex forms an integral part of this Recommendation)

A.1 BNF-style Syntax

In order to provide a generic description, a BNF-based syntax is used. The general BNF rules described below are employed throughout this Recommendation.

- A terminal token (leaf) is noted by a literal. If the literal cannot be distinghished from the literals used to describe the syntax, then the token shall be enclosed in double quotes (").
- Strings of characters enclosed in double quote characters denote terminal tokens that consist of the constant text formed by those strings.
- A non-terminal token (node) is noted by a literal delimited by the "<" (lower than) and ">" (greater than) characters.
- An optional token (or group of tokens) is delimited with the "[" and "]" characters.
- A group of tokens is delimited by the "(" and ")" characters. Groups can be nested.
- A group of tokens enclosed in "{" and "}" may be repeated 0, 1 or more times.
- The character "|" is used to separate alternative tokens (or groups of tokens).

– The string ":=" is used as the production delimiter.

– Tokens are separated by space or tabulation characters (or a combination of the two).

– The string "--" is used to introduce a comment in the BNF description. The comment finishes at the end of the line. BNF comments cannot be nested.

Applied to text based encodings:

– The terminal token STRING denotes a string of characters encoded as implied by the value of the Code-ID (see Table 14). The string of characters may be of a limited size, as stated by the token STRING(SIZE(xxx..yyy)). In this case, xxx denotes the minimum string length, yyy denotes the maximum string length.

– The special notation "STRING(xxxx + yyyy)" where "xxxx" and "yyyy" are character strings results in a string formed by the concatenation of "xxxx" and "yyyy".

– The terminal token PATH denotes a STRING containing only characters valid as absolute path specification for the underlying operating system.

– The terminal token NUMERIC-STRING denotes a string of characters representing a number. Only digits 0 to 9 may be used.

– The token ↵ indicates that a line break occurs. However, it is possible to include a comment in a TDD by ending the TDD line with a semicolon character, the comment itself, and the line break.

– The order implied by the production rules in the BNF description may not be the order in which the tokens have to be encoded. The additional rules for the encoding schemes specify the order, if any, that has to be respected.

A.2 C-Language Notation

The generic C-Language notation used is based on the current ANSI standard. So, if gathered in a file, the descriptions can be compiled by existing ANSI compilers. However, in the operating system dependent sections, application of the appropriate compilers is required. In general, the descriptions are based on following conventions:

– The first 16 characters of definitions and variable names are significant;

– Upper and lower case are differentiated.

Annex B

Location of the ICE

(This annex forms an integral part of this Recommendation)

The Interface Configuration Environment (ICE), represents a "global" source for all local applications (LAs) conforming to this Recommendation.

On the operating systems UNIX, MacOS and the operating systems belonging to the Microsoft family (OS/2, MS-DOS, etc.), the Master ICE is represented by a file[17)]. In a LAN computing environment such as Netware, Vines or LAN Manager, the Master ICE is also presented as a file. This file shall be opened and read by the application in order to extract the information about the CAs accessible within the system. The name of the file on the systems or networks mentioned above is "APPLICOM.ICE".

17) A "file" in this context means either a real operating system file or an operating system device driver, which behaves exactly as a file.

The Master ICE file is located differently on the above-mentioned systems. On some systems there exists an "environment variable"18), which may hold the path to the ICE. In those cases the variable is named "APPLICOM" (all capitals). The following algorithms for determining the ICE location shall be used (see Table B.1):

TABLE B.1/T.611

Determination of Master ICE location

Operating System	Algorithm
UNIX	First look in the Environment variable "APPLICOM". If there is no variable, then the ICE is located in the /dev subdirectory and named "APPLICOM_ICE"
MacOS	The ICE is located in the System folder
MS-DOS	Look in the Environment variable "APPLICOM". If this variable does not exist, this should be regarded as an error
Windows	File is named "ICE.INI" and located in the local windows directory of the LA system. If this file does not exist, look into file "WIN.INI" in the windows directory, section [APPLICOM_ICE]

In a LAN computing environment, the Master ICE file should be installed by the system administrator on a shared directory which is accessible by all potential LAs.

Individual CA-Descriptor files or drivers for CAs in a single user or LAN environment do not need to be installed on the same directory as the Master-ICE, but should be placed on directories accessible to all LAs.

The following naming convention shall be used for CA-Descriptor files or drivers:

CAnnnnAC.<ext>

where:

nnnn is a number in the range 1..9999

and:

<ext> := ("ini" | "exe" | "dll" | <other>)

"ini" shall be used for the case of a texte file, and <other> may be defined by the CA-Manufacturer based upon the naming conventions used within the target computing environment.

Access to the CA-Descriptor(s) for an individual CA shall be defined within the Master-ICE in the form of a path parameter (path plus file name). The default location for the CA-Descriptor if no path is defined shall be upon the same path as the Master-ICE.

18) Operating systems like UNIX or MS-DOS provide a so called "Environment" which consists of a set of ASCII strings applied to an ASCII represented variable.

Annex C

List of APPLI/COM Error Codes

(This annex forms an integral part of this Recommendation)

Table C.1 states the error codes which shall be used commonly by CA implementations conforming to this Recommendation.

TABLE C.1/T.611

List of error codes

Error code[a)]	Error text	Comment
0000	Success	No error – Successfull operation
0001-4999	Private use	Reserved for CA private use
5000	CA Interface Error	
5001	Unsupported Service specified	
5002	File "FILENAME" already exists	CA cannot create or write "FILENAME" file
5003	Error creating "TARGET"	CA cannot create or write "TARGET" file
5004	Error reading Address List	
5005	Error reading Document List	
5006	Error reading Document	
5007	Error writing Document	
5008	"Code-ID" not supported	
5009	Invalid "Connection-ID"	
5010	Invalid "CA-ID"	
5011-5499	Reserved	Reserved for further study
5500-5999	Private use	Reserved for CA private use
6000	Syntax Error	
6001	Unknown Function requested	
6002	TDD is empty	
6003	APPLI/COM Header is invalid	Invalid length or contents
6004	Parser Error (TDD line too long)	
6005	Parser Error (TDD line too short)	
6006	Multiple Occurencies of Keyword	
6007	Parameter has wrong Size	
6008	Keyword ERROR is missing	
6009	Mandatory Keyword is missing	
6010	Undefined Keyword	
6011	Conflicting Keywords	
6012	Parameter out of Range	
6013	Keywords out of Sequence	Order: FUNCTION, SUBFUNC, others...
6014	Keyword Separator missing	Colon (":") is missing
6015	Invalid "LA-ID"	

TABLE C.1/T.611 *(end)*

List of error codes

Error code[a)]	Error text	Comment
6016	Invalid "REQ-ID"	
6017	Invalid "COM-ID"	
6018	Invalid "Service-ID"	
6019	"Type-ID" does not fit to "Service-ID"	
6020	"Conversion-ID" does not fit to "Type-ID"	
6021	"TARGET" File missing	Applies to COPY TDD
6022	Invalid Code-ID	Allowed are: "A", "B", "C", "E", "I" or "P"
6023	Incompatible Interface Version	APPLI/COM Header Version does not fit
6024-6499	Reserved	Reserved for further study
6500-6999	Private use	Reserved for CA private use
7000	Hardware/System related Error	
7001-7499	Reserved	Reserved for further study
7500-7999	Private use	Reserved for CA private use
8000	Error during Document Conversion	
8001	Invalid Transfer Format	
8002	Unexpected End of Document	
8003	Unexpected Error accessing Document	
8004	Error during T.61 Conversion	
8005	Error during ASCII Conversion	
8006	Error during TIFF Conversion	
8007	Error during T.4 Conversion	
8008	Error during T.6 Conversion	
8009-8499	Reserved	Reserved for further study
8500-8999	Private use	Reserved for CA private use
9000	Transmission Error	
9001	Connection broken	
9002	Remote Destination busy	
9003	Connection set-up failed	
9006	Transmission Error (Transport Layer)	
9007	Transmission Error (Session Layer)	
9008	Remote Destination closed Connection	
9009	Remote Destination has call denied	
9010	Remote Destination not compatible	
9011	Transmission Failure – Document Error	
9012-9499	Reserved	Reserved for further study
9500-9999	Private use	Reserved for CA private use

a) Error codes are given in decimal.

Annex D

Examples of TTD Exchanges

(This annex does not form an integral part of this Recommendation)

D.1 A Sample Send Session

An LA-running on a MS-DOS (or Windows) based system - wants to send a document (which contains graphic information), say the document "c:\dtp\graphic1.tif" to an addressee via facsimile group 3 services. What the LA must do in sequence is:

- look for a CA which is capable of performing the facsimile group 3 service by inspecting the ICE and, if found, perform a Login to that CA by a Login function call (if not already done);
- prepare the document as an APPLI/COM TIFF file (if not already done);
- build a SEND TDD;
- hand over the TDD to the CA;
- repeatedly poll the CA (or waiting for get informed by the alarm function) until the Response-TDD becomes available;
- retrieve the Response-TDD to learn the status of the transmission;
- logout of the CA (or) perform other functions with the same CA.

Let us assume that the LA has already logged into the CA and the "c:\dtp\graphic1.tif" file is already prepared in TIFF format. Then the LA has to prepare the Send TDD in its memory. The LA uses the default coding (T.50) for preparation of the TDD. The TDD may look like shown in Figure D.1.

```
I*APPLI/COM*1994*ITU-T*

; Send a graphics document via facsimile group 3

; Fields into which a value may return are pre-set with underline (5F$_{HEX}$) characters.

FUNCTION    :   SendAck                      ; Send with response
LA-ID       :   myLA                         ; Name of LA
REQ-ID      :   g_0815                       ; Request-id, generated by LA
SERVICE     :   FX3                          ; Facsimile G3 service
ADDRESS     :   08154711                     ; Recipient
FILENAME    :   c:\dtp\graphic1.tif          ; Full path to document
CONVERT     :   TIFF                         ; Transfer format
COMID       :   ________________________     ; Unique CA ID (Response)
STATUS      :   ________________________     ; Status of transmission (Response)
ERROR       :   ________________________     ; Error occurred ? (Response)
```

FIGURE D.1/T.611

Once the TDD is prepared, the LA internally calls the basic exchange mechanism function PutTDD which hands over the TDD to the CA.

Then the LA repeatedly polls the CA using the basic exchange mechanism function PollTDD until the Response-TDD becomes available. Using the basic exchange mechanism function GetTDD the LA retrieves the TDD from the CA into its own memory and inspects the results. The Response-TDD could look like shown in Figure D.2.

```
I*APPLI/COM*1994*ITU-T*

; Send a graphics document via facsimile group 3
; Fields into which a value may return are pre-set with underline (5FHEX) characters.

FUNCTION     :  SendAck                    ; Send with response
LA-ID        :  myLA                       ; Name of LA
REQ-ID       :  g_0815                     ; Request-id, generated by LA
SERVICE      :  FX3                        ; Facsimile G3 service
ADDRESS      :  0498154711                 ; Recipient
FILENAME     :  c:\dtp\graphic1.tif        ; Full path to document
CONVERT      :  TIFF                       ; Transfer format
COMID        :  123456                     ; Unique CA ID (Response)
STATUS       :  +                          ; Status of transmission (Response)
ERROR        :  0000/Success               ; Error occurred ? (Response)
```

FIGURE D.2/T.611

As one can see from the STATUS and ERROR field, the transmission was successful.

D.2 A Sample Receive Session

An LA running on any operating environment wants to know if there are some documents to be received within a facsimile CA. What the LA must do in the sequence is:

- – look for a CA which is capable of performing the facsimile service by inspecting the ICE and, if found, Login into that CA by a Login function call;
- – prepare a receive TDD and hand it to the CA using PutTDD function;
- – repeatedly poll the CA (using the PollTDD function) (or waiting for get informed by a back-called alarm function) until the Response-TDD becomes available;
- – retrieve the Response-TDD (using the GetTDD function) to know the status of the reception;
- – logout of the CA or perform other functions.

The LA uses the default coding (T.50) for preparation of the receive TDD. The TDD may look like the shown in Figure D.3.

```
I*APPLI/COM*1994*ITU-T*

; Receive a T.4 document via facsimile group 3

; Fields into which a value may return are pre-set with underline (5F$_{HEX}$) characters.

FUNCTION    :   Receive                     ; Send with response
LA-ID       :   myLA                        ; Name of LA
REQ-ID      :   g_0816                      ; Request-id, generated by LA
SERVICE     :   FX3                         ; Facsimile G3 service
FILENAME    :   c:\file.ext                 ; Full path to document
Cvfax3      :   TIFF                        ; Desired transfer format
TypeID      :   ________________________    ; Status of the received file
ADDRESS     :   ________________________    ; Sender's addressee, filled by CA
CONVERT     :   ________________________    ; Transfer format
COMID       :   ________________________    ; Unique CA ID (Response)
STATUS      :   ________________________    ; Status of transmission (Response)
ERROR       :   ________________________    ; Error occurred ? (Response)
```

FIGURE D.3/T.611

Having prepared the TDD, the LA internally calls the basic exchange mechanism function PutTDD and hands over the TDD to the CA.

Having done this, the LA repeatedly polls the CA using the basic exchange mechanism function PollTDD until the Response-TDD becomes available. Using the basic exchange mechanism function GetTDD the LA retrieves the TDD from the CA into its own memory and inspects the results. The Response-TDD could look shown in Figure D.4.

```
I*APPLI/COM*1994*ITU-T*

; Receive a T.4 document via facsimile group 3
; Fields into which a value may return are pre-set with underline (5F$_{HEX}$) characters.

FUNCTION    :   Receive              ;Send with response
LA-ID       :   myLA                 ;Name of LA
REQ-ID      :   g_0816               ;Request-id, generated by LA
SERVICE     :   FX3                  ;Facsimile G3 service
FILENAME    :   c:\file.ext          ;Full path to document
Cvfax3      :   TIFF                 ;Desired transfer format
TypeID      :   STD                  ;Status of the received file
ADDRESS     :   033145782762         ;Sender's addressee, filled by CA
CONVERT     :   T.4                  ;Transfer format
COMID       :   000001               ;Unique CA ID (Response)
STATUS      :   +                    ;Status of transmission (Response)
ERROR       :   0000/Success         ;Error occurred ? (Response)
```

FIGURE D.4/T.611

As one can see from the STATUS and ERROR field, the reception was received by the CA.

D.3 A Sample Trace Session

An LA running on any operating environment could have information about the transitory of definitive state of a communication record (CA-Record). What the LA must do in the sequence is:

- look for a CA which is capable of performing any telecommunication service by inspecting the ICE and, if found, Login into that CA by a Login function call;
- prepare a Trace TDD and hand it to the CA using PutTDD function;
- repeatedly poll the CA (using the PollTDD function) – or wait to get informed by a back-called alarm function – until the Response-TDD becomes available;
- retrieve the Response-TDD (using the GetTDD function) to learn the status of the reception;
- logout of the CA or performing other functions.

The LA uses the default coding (T.50) for preparation of the receive TDD. The TDD may look like as shown in Figure D.5.

```
I*APPLI/COM*1994*ITU-T*


; Rectieve a list of all CA-Records in sending state

; Fields into which a value may return are pre-set with underline (5FHEX) characters.


FUNCTION      :   Copy                          ; Send with response

LA-ID         :   myLA                          ; Name of LA

REQ-ID        :   g_0816                        ; Request-id, generated by LA

State         :   sending                       ; CA-Records being processed

Target        :   c:\file.ext                   ; Full path to document

ERROR         :   _______________________       ; Error occurred ? (Response)
```

FIGURE D.5/T.611

Having prepared the TDD, the LA internally calls the basic exchange mechanism function PutTDD and hands over the TDD to the CA.

Having done this, the LA repeatedly polls the CA using the basic exchange mechanism function PollTDD until the Response-TDD becomes available. Using the basic exchange mechanism function GetTDD the LA retrieves all CA-Records in a particular state from the CA. The Response-TDD could look like shown in Figure D.6.

```
I*APPLI/COM*1994*ITU-T*


; Rectieve a list of all CA-Records in sending state

; Fields into which a value may return are pre-set with underline (5FHEX) characters.


FUNCTION      :   Copy                          ; Send with response

LA-ID         :   myLA                          ; Name of LA

REQ-ID        :   g_0816                        ; Request-id, generated by LA

State         :   sending                       ; CA-Records being processed

Target        :   c:\file.ext                   ; Full path to document

ERROR         :   0000/Success                  ; Error occurred ? (Response)
```

FIGURE D.6/T.611

As one can see from the ERROR field, the copy was received by the CA.

Annex E

Example of Interface Configuration Environment (ICE)

(This annex does not form an integral part of this Recommendation)

An example of a Master ICE configuration (located on an MS-DOS based machine as a file) is given in Figure E.1. The example shows a Master ICE which provides access to multiple CAs which support multiple services and access methods.

```
I*APPLI/COM*1994*ITU-T*MASTER_ICE


; Note that there may be one or more CAs referenced in the Master ICE and that each shall
; include the keywords APPLICOM; SERVICE; EM and ACCESS.
; A new CA-Entry is denoted by a "#" (number sign) as shown below.


#                                                       ; Beginning of new CA-Entry
APPLICOM:      Product1 (c) by DonaldDuck               ; CA Product and Manufacturer
SERVICE:       FX3                                      ; supports fax group 3 service
SERVICE:       EMAIL                                    ; supports EMAIL service
EM:            file                                     ; File exchange method
ACCESS:        CA1AC.INI                                ; File based CA-Descriptor


#                                                       ; Beginning of new CA- Header
APPLICOM:      Product2 (c) by MickeyMouse              ; CA Product and Manufacturer
SERVICE:       FX3                                      ; supports fax group 3 service
SERVICE:       TLX                                      ; supports telex service
SERVICE:       FT                                       ; supports file transfer service
EM:            primitive                                ; Primitive exchange method
CA-ID:         007                                      ; ID of CA
ACCESS:        CA2AC.INI                                ; File based CA-Descriptor
ACCESS:        CA2AC.EXE                                ; CA-Descriptor executable
ACCESS:        CA2AC.DLL                                ; CA-Descriptor DLL
```

FIGURE E.1/T.611

Sample Master ICE

An example of CA-Descriptors (located on a MS-DOS based machine in a file) is given in Figure E.2.

```
I*APPLI/COM*1994*ITU-T*ICE


; Note that there may be any type of configuration information which may be stored here.
; A new configuration information is always led in by a "#" (number sign) as shown below.


#                                                         ; Beginning of new configuration
APPLICOM:        Product1 (c) by Company XYZ              ;  CA Product & Manufacturer
FC:              A                                        ; APPLI/COM Functional Class
EM:              FILE                                     ; TDD Exchange Method
SYNC:            No                                       ; Not "sync" driven
CODING:          I                                        ; TDD Coding
F_JOB_Q:         c:\applicom\job                          ; Job Queue
F_ACK_Q:         c:\applicom\ack                          ; Acknowledge Queue (Response)
ERROR_Q:         c:\applicom\err                          ; Error Queue (Response)
TLX:             STD                                      ; Telex (without dialogue facility) supported
TX:              STD                                      ; Telex via Teletex service supported
TTX:             STD                                      ; Teletex service & Type options
TTX:             OPD                                      ; Teletex service & Type options
TTX:             CTL                                      ; Teletex service & Type options
TTX:             DTM                                      ; Teletex service & Type options
TTX:             EDI                                      ; Teletex service & Type options
FX3:             STD                                      ; Telefax service group 3
ADDKEYS:         LASTTIME                                 ; Additive Keywords
ADDKEYS:         SUBADDR                                  ; Additive Keywords

#                                                         ; Beginning of new configuration
APPLICOM:        Product2 (c) by Company ABC              ; CA Product & Manufacturer
FC:              B                                        ; APPLI/COM Functional Class
EM:              primitive                                ; TDD Exchange Method
ALARM:           yes                                      ; Callback supported
CODING:          I                                        ; TDD Coding
DRIVER:          applicom                                 ; Driver provided
FX3:             STD                                      ; Fax G3 service & Type options
FX3:             DTM                                      ; Fax G3 service & Type options
FX3:             BFT                                      ; Fax G3 service & Type options
SUBMIT:          CONVERT                                  ; Supports the Conversion Submissions
CONVCHK:         TIFF2                                    ; Conversions to/from TIFF2 can be asked
CONVCHK:         PCX                                      ; Conversions to/from PCX can be asked
ADDKEYS:         LASTTIME                                 ; Additive Keywords
ADDKEYS:         SPEED                                    ; Additive Keywords
ADDKEYS:         COMMENT                                  ; Additive Keywords
...
```

FIGURE E.2/T.611

Sample CA-Descriptors

Annex F

Exchange Method of 1992 Version

(This annex does not form an integral part of this Recommendation)

This annex lists the Exchange Method Functions of the 1992 version of this Recommendation, which are superseeded by the new, so called Basic Exchange Method Functions. Table F.1 below depicts this.

TABLE F.1/T.611

Exchange Method Comparison

Exchange Method Function (1992)	Basic Exchange Method Functions (this Recommendation)
Login ( )	ELogin ( )
PutTDD ( )	EPutTDD ( )
–	EPutData ( )
PollTDD ( )	EPollTDD ( )
GetTDD ( )	EGetTDD ( )
–	EGetData ( )
SetAlarm ( )	ESetAlarm ( )
–	EAbortData ( )
Logout ( )	ELogout ( )

F.1 Exchange Method Functions of 1992 version

F.1.1 Function Login

The function Login shall be supported by the CA. It shall be invoked by the LA before any LA-CA interchange of Request TDDs and responses.

F.1.1.1 Purpose

The function Login returns to the LA a "Connection-ID" that will be used all along the LA-CA interaction until the LA logs out.

The login function is the place where a CA may control access of an LA to it. This can be achieved by checking the Login-name and the Password given by the LA. However, to which extent control of the access rights is performed, is up to the CA implementation.

F.1.1.2 Behaviour

The CA checks the parameters of the Login call. If they match, it then generates a Connection-ID the LA shall use subsequently in PutTDD, PollTDD, GetTDD and Logout function calls. The LA shall wait for the return status to proceed: if the Connection-ID returned is set to Null (zero), it means the CA failed to connect the LA due to identification failures.

F.1.1.3 Parameters

The following parameters are required (see Table F.2).

TABLE F.2/T.611

Parameters of the Login function

Parameter Name	Structure	Comment	Direction
Login-Name	String	States the name of the LA user, used to connect an LA to a CA (differs from the LA-ID)	Input Parameter
Password	String	The LA gives its password so that the CA can identify the LA	Input Parameter
Connection-ID	Integer	Returned by the CA if the Login function succeeds. Otherwise (e.g. identification failed) the CA sets the value to NULL and the error code is given by the Status parameter (see below)	Output Parameter
Status	Integer	Return error code (0000 means success)	Output Parameter

F.1.2 Function PutTDD

The function PutTDD shall be supported by the CA. It may be invoked by an LA.

F.1.2.1 Purpose

The purpose of the PutTDD function is to transmit a Request TDD from an LA to a CA.

F.1.2.2 Behaviour

The CA copies the Request TDD carried by the PutTDD function into its internal structures. The result is composed of a status notified immediately to the requesting LA, and a REQ-ID reference.

The TDD is then parsed by the CA which will further process it as required by the nature of the TDD.

F.1.2.3 Parameters

The following parameters are required (see Table F.3).

TABLE F.3/T.611

Parameters of the PutTDD function

Parameter Name	Structure	Comment	Direction
Connection-ID	Integer	The connection identifier returned by the Login function	Input Parameter
TDD location	Memory address	Specifies where the LA’s TDD is located so that the CA can copy it into its own internal structure. When the function is complete, the LA’s TDD may be deleted or used for other purposes	Input Parameter
TDD size	Integer	Indicates the size of the TDD so that the CA can allocate enough internal resources to handle it	Input Parameter
Status	Integer	Acknowledges the PutTDD function. Return error code (0000 means success)	Output Parameter

F.1.3 Function PollTDD

The PollTDD function asks the CA how many Response TDDs are waiting to be handled by the requesting LA. The PollTDD returns the number of Response TDDs waiting, and the type and size of the first Response TDD that will be returned by the next call to the GetTDD function.

F.1.3.1 Purpose

The purpose of the PollTDD function is to prepare the LA for the handling of a possible Response TDD coming from a CA. It also gives an indication of the number of Response TDDs that are waiting to be handled by the LA.

F.1.3.2 Behaviour

When the CA has many Response TDDs available, it chooses the one it will return first. This Response TDD is the TDD that will be transmitted to the LA at the next GetTDD function call emitted by the same LA.

When no Response TDD is available for the requesting LA, the return status is set to the value zero, in which case the TDD size returned is set to zero. If the TDD that was submitted is erroneous or unknown, e.g. the <TDD Header> is missing, then the function returns the original Request TDD, and sets the return status to the value zero.

When a Response TDD is available, the LA shall allocate an empty TDD which will hold the copy of the Response TDD still under CA control. Table F.4 below shows the various TDD types defined.

TABLE F.4/T.611

Assignment of TDD types

Number	TDD type	Number	TDD type
0	Unknown or erroneous TDD was submitted, or no Response TDD available	4	SUBMIT Response
1	SEND Response	5	EXTEND Response
2	RECEIVE Response	6	NATIONAL Response
3	TRACE group Response	7	PRIVATE Response

F.1.3.3 Parameters

The following parameters are required (see Table F.5).

F.1.4 Function GetTDD

The function GetTDD shall be supported by the CA. It may be invoked by an LA.

F.1.4.1 Purpose

The purpose of the GetTDD function is to retrieve a Response TDD from a CA. The CA copies the Response TDD into an internal structure of the LA.

TABLE F.5/T.611

Parameters of the PollTDD function

Parameter Name	Structure	Comment	Direction
Connection-ID	Integer	The connection identifier returned by the Login function	Input Parameter
TDD size	Integer	Indicates the size of the next Response TDD so that the LA can prepare enough resources to handle it	Output Parameter
TDD type	Integer	Indicates which type of TDD the LA shall receive next. See above for values	Output Parameter
TDD count	Integer	Indicates the number of Response TDDs that are waiting to be retrieved by the LA. NULL means no TDD is waiting	Output Parameter
Status	Integer	Acknowledges the PollTDD function. Return error code (0000 means success)	Output Parameter

F.1.4.2 Behaviour

The LA indicates the location of an empty TDD where the CA shall copy a Response TDD that is available for the LA.

The CA shall return to the LA the Response TDD that was qualified by the previous PollTDD function emitted by the same LA. The LA shall have prepared a recipient Response TDD area in its internal structures. The invocation of a GetTDD function by an LA shall always be preceded by a PollTDD function call.

If the LA invokes two or more consecutive GetTDD functions (without an intermediate PollTDD function call) always the same TDD will be returned (the one indicated in the last PollTDD function return).

F.1.4.3 Parameters

The following parameters are required (see Table F.6).

TABLE F.6/T.611

Parameters of the GetTDD function

Parameter Name	Structure	Comment	Direction
Connection-ID	Integer	The connection identifier returned by the Login function	Input Parameter
TDD location	Memory address	Specifies where the CA can copy the Response TDD into the LA’s internal structure. When the function is complete, the CA’s Response TDD may be deleted or used for other purposes	Input/Output Parameter
TDD size	Integer	Indicates the size of the empty TDD handed to the CA	Input Parameter
Status	Integer	Acknowledges the GetTDD function. Return error code (0000 means success)	Output Parameter

F.1.5 Function SetAlarm

The function SetAlarm can optionally be supported by the CA. It can optionally be invoked by an LA. If the SetAlarm function is used by an LA, then that LA shall support the CallBackRoutine function. Support of the SetAlarm function is declared in the ICE.

F.1.5.1 Purpose

The purpose of the SetAlarm function is to declare to the CA the entry point of the CallBackRoutine function. The SetAlarm function indicates to the CA it can awake the LA by invoking the CallBackRoutine function. This function may be invoked only once during an LA/CA dialogue session.

F.1.5.2 Behaviour

The CA shall record the location of the CallBackRoutine function assigned by the LA. The CA can record as many CallBackRoutine locations as there are different logged-in LAs. The CA can then awake a particular LA by invoking its CallBackRoutine.

F.1.5.3 Parameters

The following parameters are required (see Table F.7).

TABLE F.7/T.611

Parameters of the SetAlarm function

Parameter Name	Structure	Comment	Direction
Connection-ID	Integer	The connection identifier returned by the Login function	Input Parameter
CallBackRoutine location	Memory address	Specifies the entry point of a particular LA CallBackRoutine function	Input Parameter
Status	Integer	Acknowledges the SetAlarm function. Return error code (0000 means success)	Output Parameter

F.1.6 Function CallBackRoutine

The CallBackRoutine function can optionally be supported by the LA. It can optionally be invoked by a CA. In order to be invoked by the CA, the LA shall have declared it to the CA by means of the SetAlarm function.

F.1.6.1 Purpose

The CallBackRoutine function defines a mechanism that allows a CA to alert the LA that some Response TDDs are available. The use of this optional mechanism can improve the flow control between LAs and CAs on heavily loaded systems.

Invocation of the CallBackRoutine by a CA does not guarantee that the LA will poll the CA within a certain delay. It only ensures that the LA will receive an alarm from a specific CA.

The CA may invoke repeatedly the CallBackRoutine function of a given LA if that LA does not poll the CA quickly enough.

There is only one CallBack Address allowed per LA. However, if an LA wishes to be called in dependency of a specific event, the LA has to implement this feature by itself within the back-called function.

F.1.6.2 Behaviour

The CallBackRoutine is designed to allow a logged-in CA to alert the LA that the CA needs to be polled via the PollTDD function. Thus the LA should poll the CA as soon as possible so that the CA does not enter an overflow error condition.

F.1.6.3 Parameter

The following parameter is required (see Table F.8).

TABLE F.8/T.611

Parameters of the CallBackRoutine function

Parameter Name	Structure	Comment	Direction
Connection-ID	Integer	The connection identifier returned by the Login function	Input Parameter

F.1.7 Function Logout

The function Logout shall be supported by the CA. It shall be invoked by the LA on completion of any LA-CA interchange of Request TDDs and responses.

F.1.7.1 Purpose

The function Logout returns to the LA a status that states whether the LA-CA interaction has finished orderly.

F.1.7.2 Behaviour

Before completing the LA-CA dialogue, the CA may (but is not obliged to) process all pending Request TDDs that were put out by that LA.

F.1.7.3 Parameters

The following parameters are required (see Table F.9).

TABLE F.9/T.611

Parameters of the Logout function

Parameter Name	Structure	Comment	Direction
Connection-ID	Integer	The connection identifier returned by the Login function	Input Parameter
Status	Integer	Returned by the CA. Indicates whether logout was orderly processed. Return error code (0000 means success).	Output Parameter

Annex G

Service Specific Information

(This annex does not form an integral part of this Recommendation)

G.1 Call Identification Line of the Telefax G4 and Teletex Service

The Call Identification Line (CIL) as defined by ITU-T Recommendation F.200 and is laid out as indicated in Figure G.1.

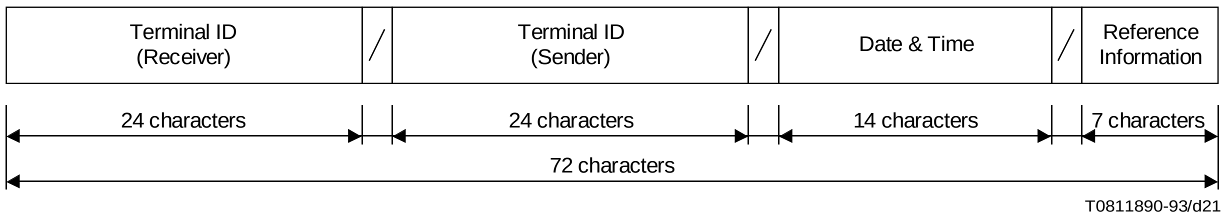


FIGURE G.1/T.611

Layout of the Call Identification Line; “/” is character code 2F$_{\mathrm{HEX}}$

G.2 Terminal ID

The Terminal ID is defined for the Teletex service per ITU-T Recommendation F.200 and for the Telefax Group 4 service per ITU-T Recommendation F.184. It is laid out as shown in Figure G.2. DNIC stands for Data Network Identification Code.

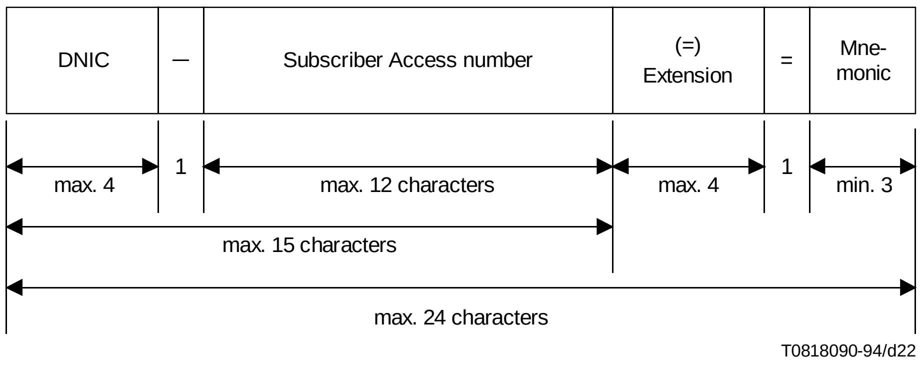


FIGURE G.2/T.611

Telefax G4 / Teletex Terminal ID

Annex H
Summary of Transfer and Transmission Formats

(This annex does not form an integral part of this Recommendation)

H.1 Transfer Formats related to Transmission Formats

See Table H.1

TABLE H.1/T.611

Summary of Transfer Formats

Service-id-parameter	Type-id-parameter	Allowed Convert-id-parameter for	
		Outgoing documents	Incoming documents
FX3	STD	ASCII, ASCIIxxx[a], T.50, TIFF	TIFF, TIFFx[b]
	BTM, DTM, BFT, EDI	VOID	
FX4	STD	ASCII, ASCIIxxx, T.50, TIFF	TIFF, TIFFx
	DTM, BFT, EDI	VOID	
TLX	STD	ASCII, ASCIIxxx, T.50, T.61	
TX	STD	ASCII, ASCIIxxx, T.50, T.61	
TTX	STD	ASCII, ASCIIxxx, T.50, T.61	
	OPD, MD, CTL	ASCII, ASCIIxxx, T.50, T.61	
	DTM, BFT, EDI	VOID	
EMAIL	STD	T.50	
	TELETEX	ASCII, ASCIIxxx, T.50, T.61	
5	G3FAX, G4CLASS1	ASCII, ASCIIxxx, T.50, TIFF	TIFFx
	VIDEOTEX, MESSAGE, BILATERAL, NATIONAL, ODA	VOID	
FT	STD	VOID	

[a] "xxx" in "ASCIIxxx" stands for the code-page declared in the ICE (e.g. "ASCII437").

[b] "x" stands for the TIFF class to be read, which equals a value between 2 and 4.

H.2 Transmission Formats related to Service

See Table H.2.

TABLE H.2/T.611

Summary of Transmission Formats (Document types)

Service-id	Type-id	Means
TLX	STD	Telex service (without dialogue facility)
TX	STD	Telex via TELETEX Conversion Facility
TTX	STD[a)]	Basic TELETEX (T.61)
	OPD[a)]	Basic TELETEX (T.61): Operator Document
	MD[a)]	Basic TELETEX (T.61): Monitor Document
	CTL[a)]	Basic TELETEX (T.61): Control Document
	DTM[b)]	Telematic File Transfer (TFT) of TELETEX Service: Document Transparent Mode
	BFT[c)]	Binary File Transfer of TELETEX Service
	EDI[b)]	Telematic File Transfer (TFT) of TELETEX Service: Edifact
FX3	STD	Basic Telefax G3 Service (MH)
	BTM[b)]	Telematic File Transfer (TFT) of Telefax G3 Service: Basic Transparent Mode
	DTM[b)]	Telematic File Transfer (TFT) of Telefax G3 Service: Document Transparent Mode
	BFT[d)]	Binary File Transfer of Telefax G3 Service
	EDI[b)]	Telematic File Transfer (TFT) of Telefax G3 Service: Edifact
FX4	STD	Basic Telefax G4 Service (MR)
	DTM[b)]	Telematic File Transfer (TFT) of Telefax G4 Service: Document Transparent Mode
	BFT[c)]	Binary File Transfer of Telefax G4 Service
	EDI[b)]	Telematic File Transfer (TFT) of Telefax G4 Service: Edifact
EMAIL	STD	IA5 Text body part
	TELETEX	Teletex body part
	G3FAX	Telefax Group 3 body part
	G4CLASS1	Telefax Group 4 body part
	VIDEOTEX	Videotex body part
	MESSAGE	body part contains an IPM (consisting of heading and bodyparts) as transferred by the E-Mail system.
	BILATERAL	Bilateral defined body part contents
	NATIONAL	National defined body part contents
	ODA	ODA body part
FT	STD	Basic File Transfer

a) Document type according to Annex E/T.62.

b) Telematic File Transfer (TFT) according to Recommendation T.571.

c) Binary File Transfer according to Recommendation T.434.

d) Binary File Transfer according to Recommendation T.434 and T.30 (Annexes).