INTERNATIONAL TELECOMMUNICATION UNION

CCITT

THE INTERNATIONAL
TELEGRAPH AND TELEPHONE
CONSULTATIVE COMMITTEE

T.611

(09/92)

TERMINAL EQUIPMENT AND PROTOCOLS FOR TELEMATIC SERVICES

PROGRAMMABLE COMMUNICATION INTERFACE (PCI) APPLI/COM FOR FACSIMILE GROUP 3, FACSIMILE GROUP 4, TELETEX AND TELEX SERVICES

Recommendation T.611

FOREWORD

The CCITT (the International Telegraph and Telephone Consultative Committee) is a permanent organ of the International Telecommunication Union (ITU). CCITT 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 Plenary Assembly of CCITT which meets every four years, establishes the topics for study and approves Recommendations prepared by its Study Groups. The approval of Recommendations by the members of CCITT between Plenary Assemblies is covered by the procedure laid down in CCITT Resolution No. 2 (Melbourne, 1988).

Recommendation T.611 was prepared by Study Group VIII and was approved under the Resolution No. 2 procedure on the 18th of September 1992.

CCITT NOTES

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

CONTENTS

Recommendation T.611

PROGRAMMABLE COMMUNICATION INTERFACE (PCI) APPLI/COM FOR FACSIMILE GROUP 3, FACSIMILE GROUP 4, TELETEX AND TELEX SERVICES

(1992)

1 Introduction

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

The general concepts of PCI are defined in Recommendation F.581[1] .

The APPLI/COM interface can be used in communication equipments allowing to participate in the Telefax Group 3, Telefax Group 4, Teletex and Telex services.

The purpose of this Recommendation is to define the messages used at the APPLI/COM interface in order to fulfill the requirements of these CCITT Telematic services.

The mechanism used for conveying these messages through the interface is called EM, for "Exchange Method". The principles of EM are shown in § 8.

The provision of the APPLI/COM interface is not mandatory for participating in a CCITT Telematic service.

The intention of this Recommendation is to help developers as much as possible by providing guidelines for the integration of CCITT services in their products.

This Recommendation covers:

- APPLI/COM Interface Requirements;
- General Definitions;
- General Principles;
- Application Features;
- Classes of implementation;
- Task Data Description (TDD);
- Exchange Method;
- Incoming/Outgoing Files;
- Conununications Control – CA-Record.

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

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.

1.1 *Normative References*

This Recommendation should be read in conjunction with the following Recommendations and International Standards:

- Recommendation F.200, *Teletex service*.
- Recommendation F.160, *General operational provisions for the international public facsimile services*.

[1] Presently at the stage of draft.

- Recommendation F.184, *Operational provisions for international facsimile service between subscriber stations with group 4 machines (Telefax 4).*
- Recommendation F.581[2)], *Service Recommendation for Programming Communication Interfaces.*
- 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 CCITT members codes.*
- Recommendation T.50, *International Reference Alphabet.*
- Recommendation T.51, *Latin Alphabet coded character sets for Telematic services.*
- Recommendation T.52[2)], *Non Latin coded character sets for Telematic services.*
- Recommendation T.61, *Character repertoire and coded character sets for the international Teletex service.*
- Recommendation T.62, *Control procedures for Teletex and G.4 facsimile services.*
- Recommendation T.434, *Specification of Binary file transfer (BFT).*
- Recommendation T.565, *Terminal characteristics for the Telematic File Transfer within the facsimile group 4 and Teletex apparatus.*
- Recommendation X.400, *Message handling system and service overview.*
- Recommendation X.209, *Specification of basic encoding rules for abstract syntax notation one (ASN.1).*
- ISO/IEC 9735:1990, *Electronic Data Interchange for Administration Commerce and Transport (EDIFACT) – Application level syntax rules.*

The following names are registered Trade Marks and belong to their respective holders:

MAC-OS	(APPLE COMPUTER, INC.)
MS-DOS	(MICROSOFT CORPORATION)
UNIX	(BELL LABS.)
TIFF	(ALDUS CORPORATION)
WINDOWS	(MICROSOFT CORPORATION)
OS/2	(MICROSOFT CORPORATION)

The versions of the Operating Systems mentioned in this Recommendation are:

MAC-OS	(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)

1.2 *Abbreviations and acronyms*

API	Application Programming Interface
APPLI/COM	Interface between local applications and communication applications

2) Presetly at the stage of draft.

ASCII	American Standard Code for Information Interchange
ASN.1	Abstract Syntax Notation One
BFT	Binary File Transfer
BNF	Backus Naur Form
CA	Communication Application
CIL	Call Identification Line
CR	Carriage Return
DCX	Multipage PCX File
DLL	Dynamic Link Library
DRF	Dispatch Received Files
EBCDIC	Extended Binary Coded Decimal Interchange Code
EM	Exchange Method
ESC	ESCape
FC	Functional Class
FCA	Functional Class A
FCB	Functional Class B
FF	Form Feed
FX3	Telefax Group 3
FX4	Telefax group 4
ICE	Interface Configuration Environment
IRA	International Reference Alphabet (Recommendation T.50)
IRV	International Reference Version (Recommendation T.50)
ISDN	Integrated Services Digital Network
LA	Local Application
LAN	Local Area Network
LF	Line Feed
PCI	Programmable Communication Interface
PCX	A variety of poster Information file used on personal computer software
PSTN	Public Switched Telephone Network
SP	SPace
TDD	Task Data Description
TFT	Telematic File Transfer
TIFF	Tagged Image File Format
TLX	Telex
TSR	Terminate and Stay Resident
TTX	Teletex
WAN	Wide Area Network

2 Scope

This Recommendation specifies the programming interface for accessing and administering the following Telematic services:

- Telefax 3;
- Telefax 4;
- Teletex;
- Telex.

It is be regarded as a high layer interface between local applications (LAs) and communication applications (CAs).

2.1 *APPLI/COM Interface Requirements*

The APPLI/COM interface shall meet the following requirements:

- 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, OS2, WINDOWS 3 and UNIX operating systems. Other operating systems will be considered on demand;
- the formal description of APPLI/COM interface messages is based on a BNF-like description, various coding schemes may be used to present the APPLI/COM 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 APPLI/COM interface can be used when several local applications and/or several communication applications are involved;
- be extensible and flexible.

3 General Definitions

3.1 *APPLI/COM Interface definitions*

The following definitions apply to the APPLI/COM interface.

3.1.1 **communication application (CA)**

A CA is provided by hardware and/or software allowing to participate in standardized services such as Teletex, Telefax Group 3, Telefax Group 4, Telex.[3)]

3.1.2 **local application (LA)**

It is an application able to generate files or documents and possibly able to manage communications dialogues. Such LAs can be word processors, spread sheets, gaphic editors, file editors, etc. The LA shall generate messages conforming to this Recommendation.

3.1.3 **task data description (TDD)**

A TDD describes the structure of the messages exchanged between an LA and a CA (not the way to effectively exchange them). A TDD request describes an exchange originated by an LA towards a CA. A TDD response describes an exchange originated by a CA towards an LA.

3) Other CCITT services such as X.400 are for further study.

3.1.4 exchange method (EM)

The Exchange Method describes how the TDDs are exchanged between a CA and an LA. This Recommendation defines a generic Exchange Method which shall be adapted to the target operating system. Local area network environments are supported by the generic Exchange Method.

3.1.5 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 included in the ICE.

3.1.6 interface configuration environment (ICE)

The ICE is a collection of CA-Descriptors. The 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.

3.2 *Files definitions*

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

3.3 *Transfer files*

The transfer files are those files exchanged between an LA and a CA. The format of these files is defined in § “9.1 Transfer formats”. (See Figure 1/T.611.)

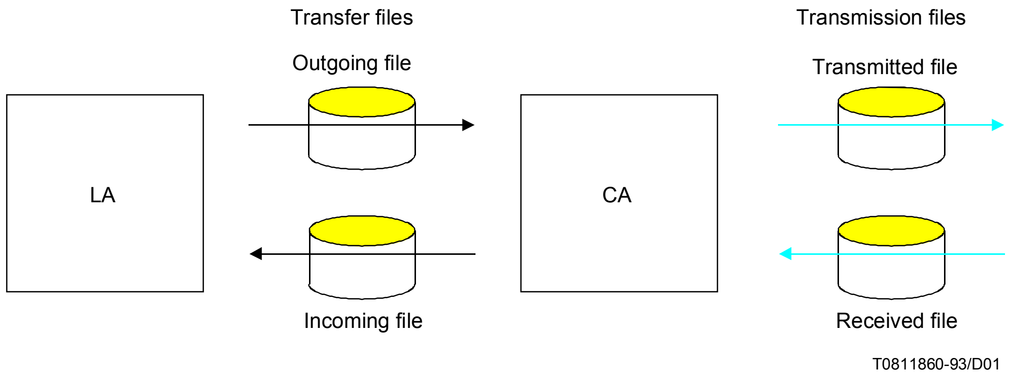


FIGURE 1/T.611

The distinct types of files defined by APPLI/COM

3.3.1 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* (see below).

3.3.2 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* (see below).

3.3.3 **transfer format**

The Transfer Format defines the structure of the transfer files. Depending on the telecommunication services used, some file formats are more suitable than others (e.g. International Alphabet No. 5, as defined in Recommendation T.50, is more suitable than T.4 for the Telex service).

3.4 *Transmission files*

The transmission files are files exchanged by a CA through the network. The format of these files is defined intrisically by the telecommunications service used (i.e. transmission files are in the T.4 format if the service used is Telefax Group 3 Normal Mode).

3.4.1 **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.

3.4.2 **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.

4 **General Principles**

4.1 *Client-Server Model*

The LA-CA interaction uses the “Client-Server” model. (See Figure 2/T.611.)

In this context:

– the CA is considered as a “server” providing telecommunication services to the LA;

– the LA is considered as the “client” of a CA using the telecommunication services provided by this CA.

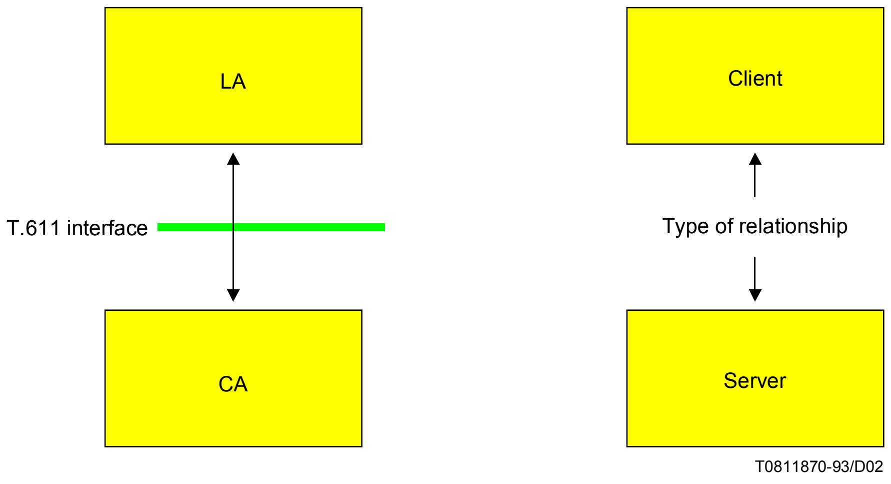


FIGURE 2/T.611

The relationship between LA and CA is modelled on the “Client-Server” model

Consequently:

– as a server, the CA must comply with one of the two functional classes defined in § 5, and with the exchange methods defined in this Recomendation;

– as a client, no constraint applies on the LA, with respect to the APPLI/COM 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 APPLI/COM 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.

The latter provides:

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

2) Conversion of documents from/into 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 exchanged through the APPLI/COM Interface*

Information to be accessed or exchanged through the interface is:

– 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 "TDD requests" and "TDD responses" which fall in one of the following categories:

– send (sending outgoing files);

– receive (receiving incoming files and CA alarms);

– trace (to follow up CA activities by accessing CA-Records);

– submit (subimission of tasks to the CA, like format conversions);

– national (to provide an extension mechanism for national based requirements);

– extend (to provide a standardized extension mechanism of the interface);

– private (to provide an extension mechanism for private needs).

4.3 *Method for information interchange*

As described with more detail in § 8, the exchange method is separated from the information to be exchanged (TDDs). Different implementations of the Exchange Methods are possible. All have in common the basic exchange method functions described in § 8. They rely on a "login" procedure which makes available 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 an LA and a CA.

4.4 *Identification means*

To cope with the various interchanges of information through the APPLI/COM interface, it is required to identify unambiguously the communicating entities and the communication events. These identifications provide the means to distinguish the various telecommunications events (COM-ID), the LAs logged in a given CA (LA-ID), and the requests generated by one LA toward a given CA (REQ-ID).

The following details these identifications.

4.4.1 *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 § 10.1).

A new COM-ID is generated by the CA when:

– A CA-record is generated; it is generated in two cases:
   1) an LA issues a "SEND" request to the CA; or
   2) an LA issues a "TRACE" request, function "Reschedule" to the CA;
– 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 dialog was terminated for any reason.

4.4.2 *Identification of LAs within a CA (LA-ID)*

4.4.2.1 *General*

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 § 10.1).

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

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.

4.4.2.2 *CA Access control*

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 request, function "Dispatch" may be offered to only one LA or to a set of different LAs, depending on the system configuration; the access to the TRACE request, function "Purge" 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 exercised by specific means that are beyond the scope of this Recommendation.

4.4.3 *Identification of LA requests (REQ-ID)*

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

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 Recommnendation. The REQ-ID is a field of the CA-Record (see § 10.1).

Recovery procedures that could be derived from the use of the REQ-ID are beyond the scope of this Recommendation (for further study).

4.4.4 *Reference to LA requests (REQ-REF)*

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

4.5 *Multiple LAs and multiple CAs*

Thanks to the APPLI/COM interface, multiple LAs may be connected to one or several communications applications (CAs). 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 setup 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 in to the communications application, it is free to use whatever service the CA provides, until the local application logs out from the CA.

4.5.1 *Step 1: Selection of a Communications Application*

The first step to be carried out by the local application during the interface setup procedure is to gain access to the interface configuration environment (ICE), which provides a list of communications applications (CA-Descriptors) accessible from within the system (see § 4.5). The CA-Descriptor is composed of items identifying the access method to a particular communications application and the capabilities of that application.

4.5.2 *Step 2: Login of a Local Application to a Communication Application*

Once the local application has selected an appropriate communications application from the ICE, it has to "login" to the selected CA. 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 no further access on behalf of that “Connection-ID” is 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.

4.6 *ICE*

The Interface Configuration Environment (ICE) lists all characteristics of the CAs available from within an equipment. Each CA has its own unique features. The ICE is the place where those features are collected. The ICE provides a standardized means for LAs to access original and/or extra features of CAs.

The ICE represents a global source for all LAs conforming to this Recommendation. It contains at the minimum the following set of APPLI/COM related configuration information, for each reachable CA:

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

– details of the Exchange method (Paths, Buffers, Service Addresses);

– TDD Coding;

– Class of Implementation;

– supported CCITT Services;

– CA facilities.

The collection of this information for one CA is called the “CA-Descriptor”. The ICE contains a list of CA-Descriptors.

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

The ICE is a logical file. The syntax and format of the ICE is described in § 6.3. The location of the ICE on a given system is described in Appendix II.

4.6.1 *Gaining access to the ICE*

Access to the ICE shall be accomplished by means of a **“**file access method” which is common to most operating systems. Hence, LA developers are not burdened with coding of operating system-dependent access methods.

4.6.2 *Configuration of the ICE*

The 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[4]. However, a LA relying on the information from the ICE regarding a particular CA shall not assume that the CA is active at that very moment (loaded & 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.

The ICE shall be configured so that it takes into account all possible communications applications accessible from within a system[5] .

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

[5] Note that 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.

4.6.3 *Presentation of the ICE*

IRA coding is used (except on those systems where the native coding of characters is EBCDIC). The ICE is presented in form of lines using the IRA character repertoire as shown in detail in Annex D.

4.7 *Submission principle*

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

– checking the conformance of a document format to specific service requirements;

– printing a document in accordance to the CCITT service through which it was (or will be) conveyed (with or without CIL, etc.);

– converting file formats [T.6, T.4, T.61, to or from transfer formats (see § 9.1)].

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

5 **Functional Classes**

5.1 *Introduction*

Two functional classes (FC) are defined. These classes are called functional class A (FCA) and functional class B (FCB). Functional class B is a strict superset of functional class A.

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 APPLI/COM interface provides access to additional CA features. These features are not essential with respects to the CCITT service 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.

5.2 *Functional Class A*

Functional class A specifies a set of transfer formats and specifies the interactions between LAs and CAs in order to send and receive documents.

FC A requires the support of:

– the “SEND” that allows a LA to send a document through a CA;

– the “RECEIVE” 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 funtionality (“SUBMIT”).

5.3 *Functional Class B*

In addition to FCA features, functional class B provides the complete TRACE functionality.

Functional class B is a strict superset of functional class A and gives further integration of communications functions to the users' applications.

5.4 *Additional Functions*

The functions “SUBMIT”, “EXTEND”, “NATIONAL” and “PRIVATE” are additional facilities that should be declared in the ICE by the CA.

6 Communications Application Features

6.1 *CA facilities*

6.1.1 *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 Recommmendation.

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

As this CA feature is optional, the support of the DRF shall be declared in the CA-Descriptor (see § 6.3).

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 § 7.4.3). The CA-Record is then assigned the LA-ID of the LA to which the received file is dispatched.

If the CA does not support the DRF, then the CA shall assign (by any appropriate means) the CA-Record to any LA using the LA-ID field.

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

6.1.2 *Alarms Support*

If the CA implements the Alarms Support feature in the Exchange Method, it means that the CA implements the SetAlarm function. This function is described in the Exchange Method; it allows a CA to "wake up" a given LA to return alarm events.

Support of the Alarms Support feature shall be stated in the ICE.

6.2 *Additional CA facilities*

This section is for further study.

6.3 *ICE*

6.3.1 *Introduction*

A given CA is described in the ICE by a set of ordered pieces of information. This set is called the CA-Descriptor.

Provision of the CA-Descriptor is mandatory for the CA. Any LA may rely on information included in the ICE. LAs cannot modify information contained in the ICE.

6.3.2 *Formal description of the ICE*

This section describes the notation used for the specification of the ICE. It utilizes a BNF-based grammar.

All CA-Descriptors are described following the rules stated below. The actual coding of the CA-Descriptors depends on the concrete syntax used; however the encoding shall follow the rules described here.

A CA-Descriptor is a collection of data organized logically as a record. The syntax is based on the use of "keywords" (see Annex A).

6.3.3 *Components*

Table 1/T.611 describes the possible <keyword> and the corresponding <parameter> values of the CA-Records. Subsection 7.5 gives complementary information about the meaning of the <parameter> values.

TABLE 1/T.611

CA-Descriptor information items

<keyword>[a)]	<parameter>	Interpretation
APPLICOM	String	Starts the definition of a CA-Descriptor. The keyword is followed by any string, in order to identify the manufacturer of the CA, for instance.
DRF	“yes”/“no”	States whether the CA supports the “Dispatch Received Files” facility.
EM	“file”/“primitive”	Exchange Method used to interchange TDDs between LAs and CAs. “file” or “primitive” are the supported values (see Annex E for further details)
CODING	Code-ID	Specifies which TDD encoding scheme is supported by the CA. See Annex C for the supported values that can be specified.
SYNC	“yes”/“no”	Indicates whether the CA is “Sync-driven”. Used only when EM specifies the “file” value; ignored otherwise. See Annex E for further details.
F_JOB_Q	Path	Specifies the path of the TDD request files. Used only when EM specifies the “file” value; ignored otherwise. See Annex E for further details.
F_ACK_Q	Path	Specifies the path of the TDD response files. Used only when EM specifies the “file” value; ignored otherwise. See Annex E for further details.
ERROR_Q	Path	Specifies the path of the TDD response files relating to errors. Used only when EM specifies the “file” value; ignored otherwise. See Annex E for further details.
DRIVER	Path	Name of the driver that must be opened to initiate dialogs with the CA. Used only when EM specifies the “primitive” value; ignored otherwise. See Annex E for further details.
COUNTRY	Octet as defined in Rec. T.35	Specifies the country where the CA is installed. Used to register country specific features like gaining access to the conversion facility or accessing the black list of dialing numbers.
ALARM	“yes”/“no”	States whether the CA supports the SetAlarm function.
FC	“A”/“B”	States which Functional Class the CA supports.
TLX	“STD”	Used only if the CA offers the Telex service. In this case, the value “STD” must be specified.

a) If the keyword name ends with a “*” then the keyword can be repeated.

TABLE 1/T.611 *(cont.)*

CA-Descriptor information items

<keyword>[a)]	<parameter>	Interpretation
TX	"STD"	Used only if the CA offers the Telex service through a Teletex gateway. In this case, the value "STD" must be specified.
TTX*	"STD" "OPD" "MD", "CTL" "DTM" "BFT" "EDI"	Used only if the CA offers the Teletex service. The <value> can include one or more options from the possible values listed besides. In the minimum, the value "STD" must be specified.
FX3*	"STD" "BTM", "DTM" "BFT"	Used only if the CA offers the Facsimile Group 3 service. The <value> can include one or more options from the possible values listed. In the minimum, the value "STD" must be specified.
FX4*	"STD" "OPD" "MD", "CTL" "DTM" "BFT" "EDI"	Used only if the CA offers the Facsimile Group 4 service. The <value> can include one or more options from the possible values listed. In the minimum, the value "STD" must be specified.
ADDKEYS*	keyword	Lists all the additional keywords supported by the CA. Only keywords classified as "+" in the TDD tables of § 7 may be specified here.
EXTEND*	keyword	Provides the possibility for extensions to the Recommendation. Can only be implemented as formal changes to the Recommendation. All CA supported keywords shall be listed.
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.
PRIVATE*	keyword	Provides the possibility for private extensions to the Recommendation. All supported keywords shall be listed.
SUBMIT*	"PRINT" "CONVERT" "CHECK"	Declares which functions are supported in the SubmitTDD function. This keyword shall be repeated as many times as required.
CONVCHK*	Convert-ID	Declares which transfer formats are supported in the SubmitTDD function "CONVERT/CHECK".
PRINT*	Print-ID	Declares which printers could be addressed by the CA in the Submit-TDD function:PRINT
RECORD*	keyword	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 manufacturer shall state in its documentation how the fields can be managed by LAs.

a) If the keyword name ends with a "*" then the keyword can be repeated.

TABLE 1/T.611 *(cont.)*

CA-Descriptor information items

<keyword>a)	<parameter>	Interpretation
ENVIRON*	"MS-DOS"/ "WINDOWS"/ "UNIX"/"OS2"/ "MAC-OS"	This keyword specifies the operating environment of the CA. If a CA supports several environments, the ICE should contain as many ENVIRON keyword instances as the number of different operating systems supported.
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.
DDE	"yes"/"no"	Dynamic Data Exchange mechanism. In the WINDOWS environment if the application supports the DDE exchange mechanism, it should specify "yes". Used only when EM specifies the "primitive" value; ignored otherwise. See Annex E. The next three keywords shall be included in the ICE of the DDE mechanism.
WIN-APP	string	Application Name(MS-DOS format) XXXXXXXX.XXX
SUBJECT*	string	All CA "Subjects" should be mentioned (if any) otherwise leave empty (to be used with the DDE keyword)
ITEM*	string	All CA "Items" should be mentioned (if any) otherwise leave empty (to be used with the DDE keyword)
DLL	"yes"/"no"	Dynamic Link Library Used only when EM specifies the "primitive" value; ignored otherwise. See Annex E. The DLL-NAME keyword should be supported only if the DLL exchange mechanism is supported.
DLL-NAME*	string	Path(s) of the DLL file(s) (used in conjunction with the DLL keyword).
LIB	"yes"/"no"	CA is a static library (LA must be linked to it).
LIB-NAME*	string	Path(s) of the library(ies) (used in conjunction with the LIB keyword).
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".

a) If the keyword name ends with a "*" then the keyword can be repeated.

7 Task Data Description (TDD)

7.1 *Introduction*

Proper operation of APPLI/COM 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 modes of information interchange are defined (see § 3 and Annex E).

One is a file interchange mode which may be easily implemented over a wide range of computers and operating systems but provides lower computer throughput.

The other is a primitive interchange mode using operating system dependent mechanisms to transport information and which provides higher throughput with reduced portability.

Due to this, the concept of Task Data Description (TDD) is introduced. 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. (See Figure 3/T.611.)

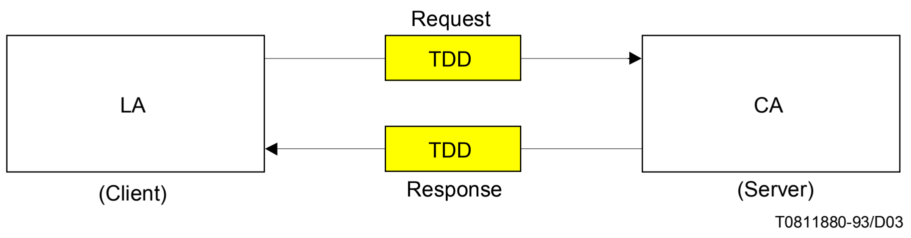


FIGURE 3/T.611
Tasks Data Description (TDDs)

The TDDs are independent of the information interchange mode 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 **TDD request**, generated by the LA toward the CA, describing the directive to be accomplished;

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

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

A CA may manage TDD requests in any order.

7.2 *TDD types*

Table 2/T.611 describes all the TDD types defined in this Recommendation. It specifies in particular whether a TDD response is generated by the CA for a given TDD request.

TABLE 2/T.611

TDD Types

TDD type	Response?	Comment
SEND	Yes if request requires one	Asks the CA to send one or more files through the network, using one CCITT service, to a set of recipients.
RECEIVE	Yes	Asks the CA to retrieve an incoming file resulting from a received file. The TDD response 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-Records belonging to a specific state. The action to be performed is described inside of 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 CCITT service specific rules, etc.
EXTEND	Yes	Provides the possibility for extensions to the Recommendation. Can only be implemented as formal changes to the Recommendation.
NATIONAL	Yes	Provides the possibility for national extensions to the Recommendation. Can only be implemented with the approval of national Administrations or authorities in charge of registering such extensions.
PRIVATE	Yes	Provides the possibility for private extensions to the Recommendation.

7.3 *TDD Formal description*

This section describes the notation used for the specification of the various TDD requests and responses. It utilizes a BNF-based grammar which is described in Annex A.

A TDD (request or response) is a collection of data organized logically as a record. The syntax is based on the use of “keywords”.

A TDD request has the same structure as the TDD response: it ensures that the TDD sizes will not change through the processing of the request by the CA. Some parameters of the request are thus “empty” (have no value specified) leaving room for the TDD response.

The following sections give details of the TDDs syntaxes.

Since the request and the corresponding response have the same format, each TDD is presented with all parameters, those applying for the request being intermixed with those applying for the response.

Subsection 7.4 gives information about the parameters that shall be used in the request and those that shall be used in the response. It also gives further details about the meaning of each parameter and value.

7.4 *Keywords and parameters of the TDDs*

The columns of the TDD tables are defined as follows (see Table 3/T.611):

TABLE 3/T.611

Explanation of the column headlines for the TDD tables

Column	Content
Class	Stands for the class of the keyword. "Basic" means 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.
Type	Specifies whether the keyword is mandatory ("m"), optional ("o") or conditional ("c"). When the keyword is optional, then the "Default" cell specifies a default value for the parameter. When the keyword is conditional, it means that the keyword is mandatory when the condition is met and forbidden when the condition is not met. The "Comment" cell specifies the condition.
Keyword	Gives the name of the keyword as it reads in the TDD. Case is NOT significant e.g. "Function" and "funcTioN" shall be interpreted equally. When the keyword name is followed by the "*" sign, it means that the keyword can be repeated many times in the same TDD. The "*" sign is not part of the keyword name, it thus shall not be specified.
Parameter	Lists the possible values for the parameter. The meaning of the information in that column is described in § 7.5.
Default	Lists the default value for the parameter (when applicable, i.e. when the "Type" cell reads "0")
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.
Comment	Gives short explanation for the use of the keywords and parameters. Also specifies the condition when the keyword type is "c" (see above).
Service applicability	Specifies to which CCITT service the keyword is applicable. If the CA encounters a keyword that is not applicable to a given service, the CA shall not reject the request (i.e. it shall ignore the keyword) if possible. The full explanation of the keywords states which way the CA shall behave in such situations.

Note – The following lists the general rules to be used for the assignment of default values:

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

– Keywords are divided into two categories: basic and "+" (additional). Additional 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 additional 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 4/T.611 explains how the “Type” and “Input/Output” classifications are used:

TABLE 4/T.611

Usage of “Type” and “Input/output” classifications

Type	Input/Output	The keyword
“m”	“i”	Must be specified in the Request; a parameter value must also be specified. In the Response, the keyword is also specified, with its parameter value unchanged.
	“o”	Must 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”	Must be specified in the Request; the parameter value must 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.
“c”	If the condition applies: handle the “c” as “m”. If the condition does not apply: the keyword may not be specified in the Request and in the Response.	

7.4.1 *SEND*

The SEND request is used to send one or more files to one or more recipients. The SEND response (if any) acts as an acknowledgment of a previous SEND Request. If the SEND Request specifies “send” in the parameter field of the “Function” keyword, the CA does not return any Response. If the SEND Request specifies “sendack” in the parameter field of the “Function” keyword, in which case it is not possible to use the “AddrList” keyword, the CA shall return a Response. For clarification purposes, two tables are provided: one for the “send” situation, the other for the “sendack” situation.

The SEND response is always understood as a completion status, i.e. if the request succeeds, the response shall be generated at the completion of the SEND request processing.

The SEND TDD belongs to Functional Classes A and B.

In order to send a single document with the “send” variation, in which case no TDD Response is generated by the CA, the TDD has to contain at minimum the following keywords: FUNCTION, LA-ID, REQ-ID, SERVICE, ADDRESS, FILENAME and CONVERT with the appropriate parameters.

It is furthermore possible to compose a list of documents to be sent to the recipient(s) by using the FILELIST keyword instead of the FILENAME keyword. The FILELIST keyword may be repeated. Due to the nature of the FILELIST keyword, the specification of the CONVERT and the optional TYPE keywords has to be done in accordance with the syntax of the FileSpec parameter as described in Annex A.

Sending a single document with the "sendack" variation involves at minimum the use of the keywords FUNCTION, LA-ID, REQ-ID, SERVICE, ADDRESS, FILENAME, CONVERT, STATUS and ERROR with the appropriate parameter values. STATUS and ERROR are used to store the status and error codes of the send event. Note that in this case it is not possible to use the ADDRLIST keyword, i.e. only one recipient may be specified.

In the case of the Facsimile group 3 and group 4 services, different resolutions may be specified. Table 5/T.611 summarizes these resolutions. This impacts the parameter values of the keyword "Highres" in the SEND TDD (both variations) (see Tables 6/T.611 and 7/T.611).

TABLE 5/T.611

Resolutions for the Facsimile services and HIGHRES keyword

HIGHRES parameter value	Fax Resolution (dpi)	
	Group 3	Group 4
0	98	200
1	196	240
2	200/400	300
3	300	400
4	400	400

7.4.2 *RECEIVE*

The RECEIVE request is designed to post a TDD that will be filled up by the CA (in the RECEIVE response).

The request specifies the telecommunications service, the storage area for the incoming file and the desired transfer format.

The recipient's sub-address can also be used as a selector.

The RECEIVE TDD belongs to Functional Classes A and B.

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

– either the LA retrieves what comes next without any preselection; in this case the LA must be able to handle the received document properly independently of the document transfer format;

– or the LA first inspects a copy of a list of received documents by using the TRACE:copy TDD 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 incorporates the "TRACE:copy" functionality.

In both cases the RECEIVE TDD has to contain at the minimum the keywords FUNCTION, LA-ID, REQ-ID, FILE-NAME, CONVERT, STATUS and ERROR with the appropriate parameter values. The keyword CONVERT is used to store the actual transfer format of the document inside the Response TDD; STATUS and ERROR 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 COM-ID in the RECEIVE Request in order to extract that document. (See Table 8/T.611.)

TABLE 6/T.611

Keywords and parameters for the SEND TDD ("sendack" variation)

Class	Type	Keyword	Parameter	Default	Input/ output	Comment	Service applicability tlx tx ttx fx3 fx4
Basic	m	Function	"sendack"		i	CA shall generate a SEND response.	x x x x x
Basic	m	LA-ID	LA-ID		i		x x x x x
Basic	m	REQ-ID	REQ-ID		i		x x x x x
Basic	m	Service	Service-id		i		x x x x x
Basic	o	Type	Type-id	"std"	i		x x x x x
Basic	m	Adress	Adress		i/o	Specifies one recipien't call number. See Annex B for the layout of the call numbers.	x x x x x
Basic	c	Filename	Path		i	The single outgoing file to be transmitted, delivered under the transfer format specified by the Convert keyword. If more outgoing files must be transmitted, use the Filelist keyword. **Condition:** only one file to send.	x x x x x
Basic	c	Convert	Convert-id		i	Used only if the Filename keyword is used – states the transfer format of the outgoing file. **Condition:** only one file is to be transmitted.	x x x x x
Basic	m	Status	status		o	Return status from the CA.	x x x x x
Basic	m	Error	error		o	The error returned by the CA.	x x x x x
Basic	o	COM-ID	com-id		o	Identification of the communication (computed by the CA).	x x x x x

TABLE 6/T.611 *(cont.)*

Keywords and parameters for the SEND TDD ("sendack" variation)

Class	Type	Keyword	Parameter	Default	Input/ output	Comment	Service applicability				
							tlx	tx	ttx	fx3	fx4
Basic	o	Sendtime	Date-time "immediate" "urgent"	"immediate"	i	CA shall actually process the request at the time specified. Process the request now (but yield scheduled Send requests that become due). Process the request urgently (even before other scheduled Send requests).	x	x	x	x	x
Basic	o	Cil	cil		o	Call identification line built by the CA (see Recommen-dation F.200).			x		x
+	o	GenCil	"yes"/"no"	"yes"	i	Asks the CA to generate a CIL in the transmitted file (done anyway in the Teletex and Fax Group 4 services).	x	x	x	x	x
+	o	Highres	"0", "1", "2", "3", "4"	"0"	i	Enforces higher resolution when set to values greater than "0". See Table 5/T.611.				x	x
+	o	Notify	"yes"/"no"	"yes"	i	Asks for explicit delivery notification (not used in all countries).		x			
+	o	Prolog	Path		i	Filename of the "control document" (full path).			x		x
+	c	FileList*	FileSpec		i	Specifies a list of files; uses a special syntax defined in Annex A. **Condition:** more than one file to send.	x	x	x	x	x
+	o	Minor	error		o		x	x	x	x	x
+	o	Warning	error		o		x	x	x	x	x

TABLE 6/T.611 *(cont.)*

Keywords and parameters for the SEND TDD ("sendack" variation)

Class	Type	Keyword	Parameter	Default	Input/ output	Comment	Service applicability tlx	tx	ttx	fx3	fx4
+	o	T61 Opts	T61options		i	Applicable only in the Filename keyword is used and the Convert keyword specifies a parameter set to "t61". It lists all the T.61-related description of the file features. See Annex A for syntax.			x		
+	o	Lasttime	Date-time	CA dependent	i	Limit time for processing the request expressed as an absolute time.	x	x	x	x	x
+	o	Comment	String		i		x	x	x	x	x
+	o	Userinfo	String		i	Assigns an user comment to the document. This comment is transmitted with the document. 12 characters maximum.			x		x
+	o	From	integer	first page	i	The first page to actually send; may be specified only if the Filename keyword is specified. Does not apply to binary files.	x	x	x	x	x
+	o	To	integer	last page	i	The last page to actually send; may be specified only if the Filename keyword is specified. Does not apply to binary files.	x	x	x	x	x
+	o	Subaddr	Sub-address		i	Originator's sub-address.	x	x	x		x
+	o	Name	String		i	Assigns a "name" to the files to be sent. 12 characters maximum, it corresponds to the filename parameter in TFT (see Recommen-dation T.565).			x		x

TABLE 6/T.611 *(cont.)*

Keywords and parameters for the SEND TDD ("sendack" variation)

Class	Type	Keyword	Parameter	Default	Input/ output	Comment	Service applicability tlx tx ttx fx3 fx4
+	o	Userkey	String		i	The request and the response shall contain the same parameter value so that the user can link them. The CA shall not interpret this parameter in any manner.	x x x x x
+	o	UseEcm	"yes"/"no"	"yes"	i/o	States whether Error Correction Mode shall be used.	x
Basic	o	G3Speed	14 400, 12 200, 9600, 7200, 4800, 2400	highest speed available	i/o	Modulation speed. If the CA cannot return the modulation speed used, it shall blank the field.	x

TABLE 7/T.611

Keywords and parameters for the SEND TDD ("send" variation)

Class	Type	Keyword	Parameter	Default	Input/ output	Comment	Service applicability tlx tx ttx fx3 fx4
Basic	m	Function	"send"		i	CA shall NOT generate a SEND response.	x x x x x
Basic	m	LA-ID	LA-ID		i		x x x x x
Basic	m	REQ-ID	REQ-ID		i		x x x x x
Basic	m	Service	Service-id		i		x x x x x
Basic	o	Type	Type-id	"std"	i		x x x x x
Basic	c	Address	Address		i	Specifies one recipient's call number. See Annex B for the dialing conventions. **Condition:** may not be used if the AddrList keyword is used.	x x x x x
Basic	c	Filename	Path		i	The single outgoing file to be transmitted, delivered under the transfer format specified by the Convert keyword. If more outgoing files must be transmitted, use the Filelist keyword. **Condition:** only one file to send.	x x x x x
Basic	c	Convert	Convert-id		i	Used only if the Filename keyword is used – states the transfer format of the outgoing file. **Condition:** may not be used in FileList keyword is used.	x x x x x
Basic	o	Sendtime	Date-time "immediate" "urgent"	"immediate"	i	CA shall actually process the request at the time specified. Process the request now (but yield scheduled Send requests that become due). Process the request urgently (even before other Scheduled Send requests).	x x x x x

TABLE 7/T.611 *(cont.)*

Keywords and parameters for the SEND TDD (“send” variation)

Class	Type	Keyword	Parameter	Default	Input/ output	Comment	Service applicability tlx	tx	ttx	fx3	fx4
+	o	Highres	“0”, “1”, “2”, “3”, “4”	“0”	i	Enforces higher resolution when set to values greater than “0”. See Table 5/T.611.				x	x
+	o	Notify	“yes”/“no”	“yes”	i	Asks for explicit delivery notification (not used in all countries).	x				
+	o	Prolog	Path		i	Filename of the “control document” (full path).			x		x
+	c	AddrList*	Address		i	Specifies a list of recipients. **Condition:** to send to more than one recipient.					
+	c	FileList*	FileSpec		i	Specifies a list of files. **Condition:** to send more than one file. Uses a special syntax described in Annex A.	x	x	x	x	x
+	c	T61 Opts	T61options		i	**Condition:** applicable only if the Filename keyword is used and the “Convert” keyword specifies a parameter value set to “t61”. It lists all the T.61-related description of the file features.			x		
+	o	Lasttime	Date-time	CA dependent	i	Limit time for processing the request expressed as an absolute time.	x	x	x	x	x
+	o	Comment	String		i		x	x	x	x	x
+	o	Userinfo	String		i	Assigns an user comment to the document. This comment is transmitted with the document. 12 characters maximum.			x		x

TABLE 7/T.611 *(cont.)*

Keywords and parameters for the SEND TDD ("send" variation)

Class	Type	Keyword	Parameter	Default	Input/ output	Comment	Service applicability				
							tlx	tx	ttx	fx3	fx4
+	o	From	integer	first page	i	The first page to actually send; may be specified only if the Filename keyword is specified. Does not apply to binary files.	x	x	x	x	x
+	o	To	integer	last page	i	The last page to actually send; may be specified only if the Filename keyword is specified. Does not apply to binary files.	x	x	x	x	x
+	o	Subaddr	Sub-address		i	Originator's sub-address.	x	x	x		x
+	o	Name	String		i	Assigns a "name" to the files to be sent. 12 characters maximum. It corresponds to the filename parameter in TFT (see Recommendation T.565).			x		x
+	o	Userkey	String		i	The request and the response shall contain the same parameter value so that the user can link them. The CA shall not interpret this parameter in any manner.	x	x	x	x	x
+	o	GenCil	"yes"/"no"	"yes"	i	Asks the CA to generate a CIL in the transmitted file (this is done anyway in the Teletex and Fax Group 4 services).	x	x	x	x	x
+	o	UseEcm	"yes/no"	"yes"	i	States whether Error Correction Mode shall be used.				x	
Basic	o	G3Speed	14 400, 12 200, 9600, 7200, 4800, 2400	highest speed available	i	Modulation speed. If the CA cannot return the modulation speed used, it shall blank the field.				x	

TABLE 8/T.611

Receive TDD keywords and parameters

Class	Type	Keyword	Parameter	Default	Input/ output	Comment	Service applicability				
							tlx	tx	ttx	fx3	fx4
Basic	m	Function	“receive”		i		x	x	x	x	x
Basic	m	LA-ID	LA-ID		i		x	x	x	x	x
Basic	m	REQ-ID	REQ-ID		e		x	x	x	x	x
Basic	o	Service	Service-id		i/o	If used in the Request, used as a selection criterion.	x	x	x	x	x
Basic	o	Type	Type-id		o		x	x	x	x	x
Basic	o	Address	Address		o	Specifies the originator’s call number. See Annex B for the layout of the call numbers.	x	x	x	x	x
Basic	m	Filename	Path		i	Specifies the file name of the incoming file.	x	x	x	x	x
Basic	m	Convert	Convert-id		o	Describes the actual transfer format of the incoming file returned by the CA.	x	x	x	x	x
Basic	o	Cvtlx	Convert-id	“T.50”	i	Transfer format desired by the LA.	x				
Basic	o	Cvtx	Convert-id	“T.50”	i	Transfer format desired by the LA.		x			
Basic	o	Cvttx	Convert-id	“T.61”	i	Transfer format desired by the LA.			x		
Basic	o	Cvfax3	Convert-id	“TIFF”	i	Transfer format desired by the LA.				x	
Basic	o	Cvfax4	Convert-id	“TIFF”	i	Transfer format desired by the LA.					x
Basic	m	Status	status		o	Return status from the CA.	x	x	x	x	x
Basic	m	Error	error		o	The error returned by the CA.	x	x	x	x	x
Basic	o	COM-ID	com-id		i/o	Identification of the communication (computed by the CA).	x	x	x	x	x
Basic	o	Cil	cil		o	Call identification line built by the CA (see Recommen- dation F.200).			x		x
+	o	Prolog	Path		i	Filename of the Control document.			x		x

TABLE 8/T.611 *(cont.)*

Receive TDD keywords and parameters

Class	Type	Keyword	Parameter	Default	Input/ output	Comment	Service applicability				
							tlx	tx	ttx	fx3	fx4
+	o	Delete	“yes”/“no”	“yes”	i	Received information becomes unavailable for the requesting LA after reception.	x	x	x	x	x
+	o	Minor	error		o		x	x	x	x	x
+	o	Warning	error		o		x	x	x	x	x
+	o	Name	String		o	Return the name of the incoming (12 characters max.).			x		x
+	o	Userinfo	String		o	Assigns an user comment to the document. This comment is transmitted with the document. 12 characters maximum.			x		x
+	o	RcvTime	Date-time		o	Time of the document reception by the CA.	x	x	x	x	x
+	o	Firstpg	integer		o	Number of the first page received.			x		x
+	o	Subaddr	Sub-address		i/o	Recipient’s sub-address; if specified in the Request, used as a selector for the retrieval.	x	x	x		x
+	o	G3Speed	14 400, 12 200, 9600, 7200, 4800, 2400		o	Modulation speed. If the CA cannot return the modulation speed, field will be left empty.				x	

7.4.3 *TRACE*

The TRACE request is used to manage the CA-Records. The TRACE is specific to Functional Class B.

The purpose of the TRACE TDD is to control the operation of a CA. The TRACE TDD operates on CA Records. It can perform the following actions:

– **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;

– **copy** any CA Records in any state into a file**;** this feature is useful to build what is commonly known as “journals” or “logs”;

- **cancel** any CA Record in the "sending" state; this feature allows the LA to interrupt and terminate the transmission that the CA is performing[6] .
- **delete** any CA Record in the "delayed" state in order to cancel it;
- **reschedule** any CA Record in the "failed" state to give the CA a chance to retransmit it; this feature facilitates the management of transmissions that failed because the recipient was busy or the Request was badly formed, for example;
- **dispatch** any CA Record in the "reception" state to assign it to the actual recipient LA.

It should be noted that all TRACE 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 SEND Request, the TDD shall contain at minimum the keywords: FUNCTION, LA-ID, REQ-ID, REQREF and ERROR. ERROR is used to store the error code of the operation in the Response TDD. If it is known, the LA may also specify the COM-ID) as a parameter to the COM-ID keyword instead of specifying the REQREF keyword. (See Table 9/T.611.)

TABLE 9/T.611

Keywords and parameters for the TRACE TDD (function "delete")

Class	Type	Keyword	Parameter	Default	Input/ output	Comment	Service applicability tlx tx ttx fx3 fx4
Basic	m	Function	"delete"		i	Deletes CA-Records in the "delayed" state.	x x x x x
Basic	m	LA-ID	LA-ID		i		x x x x x
Basic	m	REQ-ID	REQ-ID		i		x x x x x
Basic	m	Error	error		o	The error returned by the CA.	x x x x x
Basic	c	REQREF	REQ-ID		i	Reference to a previous REQ-ID. **Condition:** cannot be used together with the COM-ID keyword.	x x x x x
Basic	c	COM-ID	com-id		i	Identification of the communication computed by the CA. **Condition:** cannot be used with the REQREF keyword.	x x x x x
+	o	Minor	error		o		x x x x x
+	o	Warning	error		o		x x x x x

[6] It is not guaranteed that the cancel operation succeed because of the "batch" nature of the interface.

In order to get a copy of CA-Records referring to a specific state, the TDD contains at minimum the keywords:

– FUNCTION
– LA-ID
– REQ-ID
– STATE
– TARGET
– ERROR

The keyword STATE specifies the state of the CA-Records intended to be copied. The keyword TARGET specifies the file where the copy is to be performed by the CA. 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 Header-ID of the requesting TDD. The layout of the list is implied by the RECORDS entry of the ICE.

If it is known, the LA may also specify the COM-ID as a parameter to the COM-ID keyword instead of specifying the REQREF keyword. (See Table 10/T.611.)

TABLE 10/T.611

Keywords and parameters for the TRACE TDD (function "copy")

Class	Type	Keyword	Parameter	Default	Input/ output	Comment	Service applicability tlx tx ttx fx3 fx4
Basic	m	Function	"copy"		i	Copy all the CA-Records of a specific state in the file pointed out by the TARGET keyword.	x x x x x
Basic	m	LA-ID	LA-ID		i		x x x x x
Basic	m	REQ-ID	REQ-ID		i		x x x x x
Basic	m	State	state	"all"	i	State of the CA-RECORD. See § 10.	x x x x x
Basic	m	Target	Path		i	Destination file name.	x x x x x
Basic	m	Error	error		o	The error returned by the CA.	x x x x x
Basic	c	REQREF	REQ-ID	"all"	i	Reference to a previous REQ-ID. **Condition:** cannot be used together with the COM-ID keyword.	x x x x x
Basic	c	COM-ID	com-id	"all"	i	Identification of the communication computed by the CA. **Condition:** cannot be used with the REQREF keyword.	x x x x x
+	o	Minor	error		o		x x x x x
+	o	Warning	error		o		x x x x x

In order to cancel a SEND Request the TDD shall contain the keywords (at minimum):

– FUNCTION
– LA-ID
– REQ-ID
– REQREF
– ERROR

If it is known, the LA may also specify the COM-ID as a parameter to the COM-ID keyword instead of specifying the REQREF keyword.

Note that cancelling a SEND Request may not succeed due to the nature of the interface. (See Table 11/T.611.)

TABLE 11/T.611

Keywords and parameters for the TRACE TDD (function "cancel")

Class	Type	Keyword	Parameter	Default	Input/ output	Comment	Service applicability tlx tx ttx fx3 fx4
Basic	m	Function	"cancel"		i	Cancel a CA-Record generated by a previous request, specified by the COM-ID keyword.	x x x x x
Basic	m	LA-ID	LA-ID		i		x x x x x
Basic	m	REQ-ID	REQ-ID		i		x x x x x
Basic	m	Error	error		o	The error returned by the CA.	x x x x x
Basic	c	REQREF	REQ-ID		i	Reference to a previous REQ-ID. **Condition:** cannot be used together with the COM-ID keyword.	x x x x x
Basic	c	COM-ID	com-id		i	Identification of the communication computed by the CA. **Condition:** cannot be used with the REQREF keyword.	x x x x x
+	o	Minor	error		o		x x x x x
+	o	Warning	error		o		x x x x x

In order to purge CA Records from the CA, the TDD shall contain at minimum the keywords:

– FUNCTION
– LA-ID
– REQ-ID
– STATE
– ERROR

The keyword STATE specifies the state of the CA-Records to be purged (see § 10). The keyword ERROR store the error code of the operation in the Response TDD. (See Table 12/T.611.)

TABLE 12/T.611

Keywords and parameters for the TRACE TDD (function "purge")

Class	Type	Keyword	Parameter	Default	Input/ output	Comment	Service applicability tlx tx ttx fx3 fx4
Basic	m	Function	"purge"		i	Purge a CA-Record generated by a previous request, specified by the COM-ID keyword.	x x x x x
Basic	m	LA-ID	LA-ID		i		x x x x x
Basic	m	REQ-ID	REQ-ID		i		x x x x x
Basic	m	State	state	"all"	i	State of the CA-RECORD. See § 10.	x x x x x
Basic	m	Error	error		o	The error returned by the CA.	x x x x x
Basic	c	REQREF	REQ-ID	"all"	i	Reference to a previous REQ-ID. **Condition:** cannot be used together with the COM-ID keyword.	x x x x x
Basic	c	Com-ID	com-id	"all"	i	Identification of the communication computed by the CA. **Condition:** cannot be used with the REQREF keyword.	x x x x x
+	o	Minor	error		o		x x x x x
+	o	Warning	error		o		x x x x x

In order to reschedule a failed SEND Request, the TDD shall contain at minimum the keywords:

– FUNCTION

– LA-ID

– REQ-ID

– REQREF

– ERROR

If it is known, the LA may also specify the COM-ID as a parameter to the COM-ID keyword instead of specifying the REQREF keyword. (See Table 13/T.611.)

TABLE 13/T.611

Keywords and parameters for the TRACE TDD (function "reschedule")

Class	Type	Keyword	Parameter	Default	Input/ output	Comment	Service applicability tlx tx ttx fx3 fx4
Basic	m	Function	"reschedule"		i	Reschedule a CA-record generated by a previous SEND request.	x x x x x
Basic	m	LA-ID	LA-ID		i		x x x x x
Basic	m	REQ-ID	REQ-ID		i		x x x x x
Basic	o	Address	Address		i	Used for the "reschedule" function only; specifies the new recipient's address.	x x x x x
Basic	o	Sendtime	Date-time "immediate" "urgent"	"immediate"	i	Used for the "reschedule" function. Process the request at the time specified. Process the request now (but yield to scheduled Send requests that become due). Process the request urgently (do not yield to scheduled Send requests becoming due).	x x x x x
Basic	m	Error	error		o	The error returned by the CA.	x x x x x
Basic	c	REQREF	REQ-ID		i	Reference to a previous REQ-ID. **Condition:** cannot be used together with the COM-ID keyword	x x x x x
Basic	c	COM-ID	com-id		i	Identification of the communication computed by the CA. **Condition:** cannot be used with the REQREF keyword.	x x x x x
+	o	Lasttime	Date-time	CA dependent	i	Limit time for processing the request expressed as an absolute time.	x x x x x
+	o	Minor	error		o		x x x x x
+	o	Warning	error		o		x x x x x

In order to dispatch a received file to an LA, the TDD shall contain at minimum the keywords:

- FUNCTION
- LA-ID
- REQ-ID
- NEWLA
- COM-ID
- ERROR

The keyword NEWLA contains the LA-ID of the new LA then becomes owner of the received document and may retrieve it.

Note that the use of the dispatch function may be restricted. (See Table 14/T.611.)

TABLE 14/T.611

Keywords and parameters for the TRACE TDD (function "dispatch")

Class	Type	Keyword	Parameter	Default	Input/ output	Comment	Service applicability tlx tx ttx fx3 fx4
Basic	m	Function	"dispatch"		i	Dispatch documents associated to the CA-record designated by the COM-ID.	x x x x x
Basic	m	LA-ID	LA-ID		i		x x x x x
Basic	m	REQ-ID	REQ-ID		i		x x x x x
Basic	m	NewLA	LA-ID		i	Specifies the name of the new "owner" of the document.	x x x x x
Basic	m	Error	error		o	The error returned by the CA.	x x x x x
Basic	c	REQREF	REQ-ID		i	Reference to a previous REQ-ID. **Condition:** cannot be used together with the COM-ID keyword.	x x x x x
Basic	c	COM-ID	com-id		i	Identification of the communication computed by the CA. **Condition:** cannot be used with the REQREF keyword.	x x x x x
+	o	Minor	error		o		x x x x x
+	o	Warning	error		o		x x x x x

7.4.4 *SUBMIT*

SUBMIT is designed to request the CA some operations the LA could also provide. SUBMIT belongs to Functional Class “B”.

The purpose of the SUBMIT TDD 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 TDD supports the following features:

– print a document, given its path and format;

– a document, given its path, input format, output format and output file name;

– check the format of a document, given its path and the format against which it should be checked.

See Tables 15/T.611, 16/T.611 and 17/T.611.

TABLE 15/T.611

Keywords and parameters for the SUBMIT TDD (function “print”)

Class	Type	Keyword	Parameter	Default	Input/ output	Comment	Service applicability tlx tx ttx fx3 fx4
Basic	m	Function	“print”		i	Submit the printing task to the CA.	x x x x x
Basic	m	LA-ID	LA-ID				x x x x x
Basic	m	REQ-ID	REQ-ID		i		x x x x x
Basic	m	Filename	Path		i	Original file.	x x x x x
Basic	m	InFormt	Convert-id		i	Original format.	x x x x x
+	o	Printer	Printer-Id	Printer configured inside the CA	i	The ID of a CA-known printer.	x x x x x
Basic	m	Error	error		o	The error returned by the CA.	x x x x x
+	o	Minor	error		o		x x x x x
+	o	Warning	error		o		x x x x x

TABLE 16/T.611

Keywords and parameters for the SUBMIT TDD (function "convert")

Class	Type	Keyword	Parameter	Default	Input/ output	Comment	Service applicability tlx tx ttx fx3 fx4
Basic	m	Function	"convert"		i	Submit the converting task to the CA.	x x x x x
Basic	m	LA-ID	LA-ID		i		x x x x x
Basic	m	REQ-ID	REQ-ID		i		x x x x x
Basic	m	Filename	Path		i	Original file.	x x x x x
Basic	m	Target	Path		i	Target file.	x x x x x
Basic	m	InFormt	Convert-id		i	Original format.	x x x x x
Basic	m	OutFormt	Convert-id		i	Target format.	x x x x x
Basic	m	Error	error		o	The error returned by the CA.	x x x x x
+	o	Minor	error		o		x x x x x
+	o	Warning	error		o		x x x x x

TABLE 17/T.611

Keywords and parameters for the SUBMIT TDD (function "check")

Class	Type	Keyword	Parameter	Default	Input/ Output	Comment	Service applicability tlx tx ttx fx3 fx4
Basic	m	Function	"check"		i	Submit the checking task to the CA.	x x x x x
Basic	m	LA-ID	LA-ID		i		x x x x x
Basic	m	REQ-ID	REQ-ID		i		x x x x x
Basic	m	Filename	Path		i	Original file.	x x x x x
Basic	m	Check	Convert-id		i	Format to be checked against.	x x x x x
Basic	m	Error	error		o	The error returned by the CA.	x x x x x
+	o	Minor	error		o		x x x x x
+	o	Warning	error		o		x x x x x

7.4.5 *EXTEND*

The EXTEND is designed to gather "extended" features that are not vital for the functioning of the interface, but use of which becomes widespread or necessar. EXTEND is an additional facility. (See Table 18/T.611.)

TABLE 18/T.611

Minimum keywords and parameters for the EXTEND TDD

Class	Type	Keyword	Parameter	Default	Input/ output	Comment	Service applicability tlx tx ttx fx3 fx4
Basic	m	Function	"extend"		i		x x x x x
Basic	m	Subfunc[a)]			i		x x x x x
Basic	m	LA-ID	LA-ID		i		x x x x x
Basic	m	REQ-ID	REQ-ID		i		x x x x x
Basic	m	Error	error		o	The error returned by the CA.	x x x x x

a) Parameter values are for further study.

7.4.6 *NATIONAL*

The NATIONAL is designed to gather "national" features that are specific to each country. NATIONAL is an additional facility. (See Table 19/T.611.)

TABLE 19/T.611

Minimum keywords and parameters for the NATIONAL TDD

Class	Type	Keyword	Parameter	Default	Input/ output	Comment	Service applicability tlx tx ttx fx3 fx4
Basic	m	Function	"national"		i		x x x x x
Basic	m	Subfunc[a)]			i		x x x x x
Basic	m	LA-ID	LA-ID		i		x x x x x
Basic	m	REQ-ID	REQ-ID		i		x x x x x
Basic	m	Error	error		o	The error returned by the CA.	x x x x x

a) Parameter values are for further study.

7.4.7 *PRIVATE*

The PRIVATE is designed to gather "private" features that are specific to each manufacturer. PRIVATE is an additional facility. (See Table 20/T.611.)

TABLE 20/T.611

Minimum keywords and parameters for the PRIVATE TDD

Class	Type	Keyword	Parameter	Default	Input/ output	Comment	Service applicability tlx tx ttx fx3 fx4
Basic	m	Function	"private"		i		x x x x x
Basic	m	Subfunc[a)]			i		x x x x x
Basic	m	LA-ID	LA-ID		i		x x x x x
Basic	m	REQ-ID	REQ-ID		i		x x x x x
Basic	m	Error	error		o	The error returned by the CA.	x x x x x

a) Parameter values are for further study.

7.5 *Explanation of the <parameter> values*

Table 21/T.611 describes the interpretation of the column 4 (Value(s)) in the TDD tables.

TABLE 21/T.611

Interpretation of the Value(s) column

Syntax	Means
«xyz»	The value is a character string, encoded as implied by the APPLI/COM header ID of the TDD (see Annex C); or The value is a character string "xyz" (excluding the surrounding quotes) encoded as explained in Recommendation T.50 International Reference Version (or using EBCDIC on those systems where EBCDIC is the native character coding).
LA-ID	Reference of a LA-ID. Presented as a string encoded as implied by the APPLI/COM header ID of the TDD. The purpose of the parameter is to identify the "owning" LA of a request.
REQ-ID	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 REO-ID shall be unique within a LA. It is the responsibility of the LA to ensure the REQ-I D is unique.

TABLE 21/T.611 (cont.)

Interpretation of the Value(s) column

Syntax	Means
Address	One or many call numbers (depends on the service used). See Annex B for more information. This parameter field serves to transfer the call number to APPLI/COM. The different telecommunications services require different dialing sequences and traffic discrimination digits. Since APPLI/COM supports the Teletex, telex and Telefax services, the call number entry has been standardized in the form typical of each service. Additionally the raw dialing information (i.e. the digit/letter sequence) that is given when dialing a subscriber to the postal network can also be entered.
File	Full path for a file specification on the LA equipment.
FileSpec	A string composed of: the full path of the file, followed by the transfer format of the file, and optionally followed by the Type-Id of the file and finally an optional session identifier.
Date-time	A date and time specification following the convention: "YY-MM-DD-HH:MM" (year-month-day:hour-minutes).
Sub-address	Sub address digits.
String	A string encoded as implied by the value of the Code-Id (see below). The default encoding used is T.50 as defined in Recommendation T.50, International Reference Version.
Integer	A Recommendation T.50 string representing an integer value; e.g. the value 345 is encoded "3/3 3/4 3/5". (Or using EBCDIC equivalent coding on appropriate systems).
Path	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.
Code-ID	Identifies which encoding scheme is used (see "Annex C: APPLI/COM header").
error	The error parameter has the following layout: "nnnn"/"Text..." "nnnn" stands for a 4-digit, right-justified error number, filled to the left with "O"s. "0000" = success (no error). The value of the error number itself depends on the communications software used. "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 the plain text, the text is abbreviated as necessary. The plain text for error number "0000" is "Success".
Printer-Id	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.
Convert-Id	Conversion-ID. Specifies the Transfer format used for a document. Refer to § 7.5.3.
com-id	com-id stands for COMmunication-ID. See § 4.4.1.
Status	The parameter reflects the status of a send or receive event. lt can accept the following values: + positive, + – partially negative, – negative, ? unknown.

TABLE 21/T.611 (cont.)

Interpretation of the Value(s) column

Syntax	Means
State	Specifies the state of the CA Record. See § 7.5.4 and § 10.
CIL	CIL stands for “Call Identification Line”. See § 7.5.5.
Type-Id	Specifies the Type (Subtype) of a CCITT telecommunications service. Each telecommunications service has its own set of Type-ids. See § 7.5.2.
Service-Id	Specifies the CCITT Service to be used. See § 7.5.1.
Password	User’s password. Coding is secret.
NewLA	Reference of the new LA-ID the CA-Record has changed to during a TRACE “dispatch” function. Presented as a string encoded as implied by the APPLI/COM header id of the TDD.
T61options	Syntax for T61 options. See Annex A.

7.5.1 *APPLI/COM Service-id*

Service-id	Means
TLX	Telex service (native).
TX	Telex service using Teletex/Telex conversion Facility of the Teletex service.
TTX	Teletex service.
FX3	Telefax 3.
FX4	Telefax 4.

7.5.2 *APPLI/COM Type-id*

Service-id	Type-id	Means
TLX	STD	Basic Telex Service.
TX	STD	Telex via TELTEX Conversion Facility.
TTX	STD (Note 1) OPD (Note 2) MD (Note 3) CTL (Note 4) DTM (Note 5) BFT (Note 6) EDI (Note 7)	Basic TELETEX (T.61). Basic TELETEX (T.61): Operator Document. Basic TELETEX (T.61): Monitor Document. Basic TELETEX (T.61): Control Document. Telematic File Transfer (TFT) of TELETEX Service: Document Transparent Mode. Telematic File Transfer (TFT) of TELETEX Service: Binary File Transfer. Telematic File Transfer (TFT) of TELETEX Service: Edifact.
FX3	STD BTM (Note 8) DTM (Note 9) BFT (Note 10) EDI (Note 11)	Basic Telefax G3 Service (Rec. T.4). Telematic File Transfer (TFT) of Telefax G3 Service: Basic Transparent Mode. Telematic File Transfer (TFT) of Telefax G3 Service: Document Transparent Mode. Telematic File Transfer (TFT) of Telefax G3 Service: Binary File Transfer. Telematic File Transfer (TFT) of Telefax G3 Service: Edifact.
FX4	STD (Note 12) OPD (Note 13) MD (Note 14) CTL (Note 15) DTM (Note 16) BFT (Note 17) EDI (Note 18)	Basic Telefax G4 (Rec. T.6). Basic Telefax G4 Service: Operator Document. Basic Telefax G4 Service: Monitor Document. Basic Telefax G4 Service: Control Document. Telematic File Transfer (TFT) of Telefax G4 Service: Document Transparent Mode. Telematic File Transfer (TFT) of Telefax G4 Service: Binary File Transfer. Telematic File Transfer (TFT) of Telefax G4 Service: Edifact.

Note 1 – Document types according to Recommendation T.62, Annex E.

Note 2 – Document types according to Recommendation T.62, Annex E.

Note 3 – Document types according to Recommendation T.62, Annex E.

Note 4 – Document types according to Recommendation T.62, Annex E.

Note 5 – Telematic File Transfer (TFT) according to Recommendation T.565.

Note 6 – Telematic File Transfer (TFT) according to Recommendation T.434.

Note 7 – Telematic File Transfer (TFT) according to Recommendation T.565.

Note 8 – Telematic File Transfer (TFT) according to Recommendation T.434 and T.30 (Annexes).

Note 9 – Telematic File Transfer (TFT) according to Recommendation T.434 and T.30 (Annexes).

Note 10 – Telematic File Transfer (TFT) according to Recommendation T.434.

Note 11 – Telematic File Transfer (TFT) according to Recommendation T.434 and T.30 (Annexes).

Note 12 – Document types according to Recommendation T.62, Annex E.

Note 13 – Document types according to Recommendation T.62, Annex E.

Note 14 – Document types according to Recommendation T.62, Annex E.

Note 15 – Document types according to Recommendation T.62, Annex E.

Note 16 – Telematic File Transfer (TFT) according to Recommendation T.565.

Note 17 – Telematic File Transfer (TFT) according to Recommendation T.434.

Note 18 – Telematic File Transfer (TFT) according to Recommendation T.565.

7.5.3 *APPLI/COM Convert-id*

Convert-id	Means
ASCII437	ASCII transfer format as defined in § 9.1.1. of this Recommendation. Use of this format is restricted to some services. See § 9.
ASCII	ASCII transfer format as defined on the supporting CA. This transfer format may differ from the ASCII437 transfer format on those system which do not support the same extended ASCII character set.
T50	T.50(IRA) transfer format as defined in § 9.1 .1. of this Recommendation. Use of this format is restricted to some services. See § 9.
T61	T.61 transfer format as defined in § 9.1.2. of this Recommendation. Use of this format is restricted to some services. See § 9.
TIFF	TIFF transfer format as defined in § 9.1.3. of this Recommendation. Use of this format is restricted to some services. See Note below and § 9.
VOID	Stands for "no conversion to be done". Use is restricted for transparent transfer of document in various "file transfer" modes of the CCITT services.

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;

– 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.

7.5.4 *APPLI/COM States of CA-record*

State	Concerns	Means
"delayed"	Transmission	The record has not yet been processed by the CA.
"sending"	Transmission	The record is being processed by the CA for transmission.
"sent"	Transmission	The record has been successfully transmitted to the recipient.
"failed"	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"	Reception	The record has been received but not yet retrieved.
"retrieved"	Reception	The record has already been retrieved.

7.5.5 *APPLI/COM Call Identification Line*

The Call Identification Line is defined by Recommendation F.200 and is laid out as follows (See Figure 4/T.611.)

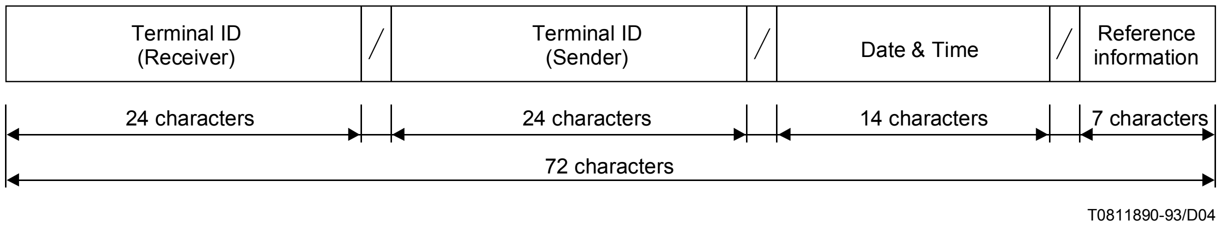


FIGURE 4/T.611

Layout of the APPLI/COM Call Identification Line;
"/" is character code 2F hex

7.6 *Error handling*

7.6.1 *Simple errors*

Keywords out of context (e.g. G3SPEED:9600 for the telex service) shall be ignored by the CA. Execution continues.

7.6.2 *Rejection*

All syntactical errors, i.e. unrecognized keywords, missing keywords classified as mandatory, conflicting keywords, multiple occurrences of a single keyword, parameters out of range within a TDD shall lead to rejection of the TDD. The CA shall not process the TDD any further. This rejection can be recognized immediately by looking at the status parameter of the PutTDD function described in § 8.4.

Any invalid outgoing file format shall also provoke the rejection of the corresponding TDD by the CA. The CA may also discard the outgoing file if appropriate.

7.6.3 *Error codes*

Error codes are counted decimal. They are comprised in the range 0000 to 9999 and subdivided in the following categories:

Error code	Meaning
0000	Success (no error) shall be supported by all CAs.
0001-4999	Reserved for CA private use.
5000-5999	APPLI/COM miscellaneous errors not fitting in other categories.
6000-6999	APPLI/COM syntax errors.
7000-7999	APPLI/COM resource and input/output errors (operating system and hardware related errors).
8000-8999	Conversion or transfer format related errors.
9000-9999	APPLI/COM service dependent errors (service signals and failures).

8 Exchange Method

This Recommendation defines an “abstract exchange method” between LAs and CAs. This exchange method can be realized by different means. Examples of recommended implementations are found in Annex E of this Recommendation.

The exchange method comprises various functional subsets; the following describes the “basic subset” called the Basic Exchange Method functions. Other functional subsets are for further study. The basic subset shall be supported by all CAs and LAs.

Note – The following has been designed in order to work with many configurations, like LA and CA being on the same equipment or CA being a communication server on a LAN. The only assumption is the presence of the ICE on each LA workstation.

In order to establish a communication channel between a LA and a CA that respects the client-server model, the functions described below shall be supported by all LAs and CAs.

8.1 *Overview of the Exchange Method*

Firstly, a LA needs to “login” to the requested CA. This login procedure is designed to ensure the LA-CA communication path is understood by the CA. No TDD exchange can take place before the login procedure is completed. By nature, the login procedure is a synchronous mechanism e.g. each action requires a response before action can take place.

Conversely, when a LA no longer needs to dialog with a CA any longer, it shall “logout” from this CA. The CA thus knows the LA-CA communications path is broken and thus will not use the CallBackRoutine function calls any longer. The logout mechanism is synchronous to ensure the LA-CA communications path is orderly broken.

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

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

– invoke the PutTDD function (described below).

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

– invoke the PollTDD function (described below);

– (if the POllTDD request returns that some TDD responses are available) Invoke the GetTDD function (described below).

When a LA needs to retrieve a TDD response, the LA shall:

– invoke the GetTDD function (described below).

When a LA receives an alarm from the CA via the “CallBackRoutine” (applicable only if the CA supports alarms as stated in the ICE (see § 6.3), and if the LA implements the “CallBackRoutine” function), it shall:

– invoke the PollTDD function (described below);

– invoke the GetTDD function (described below).

The PutTDD, PollTDD, GetTDD, SetAlarm, CallBackRoutine, Login and Logout functions are synchronous, i.e. the LA can proceed only when those functions have returned.

8.2 *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.

8.3 *Function Login*

The function Login shall be supported by the CA. It shall be invoked by the LA before any LA-CA interchange of TDD requests and responses.

8.3.1 *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.

8.3.2 *Behavior*

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.

8.3.3 *Parameters*

The following parameters are required:

Parameter name	Structure	Comment	Direction
Login-name	string	States the name of the LA user, used to connect an LA to a CA (differs form 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). See § 7.6.3.	Output parameter

8.4 *Function PutTDD*

The function PutTDD shall be supported by the CA. It may be invoked by a LA.

8.4.1 *Purpose*

The purpose of the PutTDD function is to transmit a TDD request from a LA to a CA.

8.4.2 *Behavior*

The CA copies the TDD request 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.

8.4.3 *Parameters*

The following parameters are required:

Parameter name	Structure	Comment	Direction
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 purporses.	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
Connection-ID	integer	The connection identifier returned by the Login function.	Input parameter
Status	integer	Acknowledges the PutTDD function. Return error code (0000 means success). See § 7.6.3.	Output parameter

8.5 *Function PollTDD*

The PollTDD function asks the CA how many TDD responses are waiting to be handled by the requesting LA. The PollTDD returns the num ber of TDD responses waiting, and the type and size of the first TDD response that will be returned by the next call to the GetTDD function.

8.5.1 *Purpose*

The purpose of the PollTDD function is to prepare the LA for the handling of a possible TDD response coming from a CA It also gives an indication of the number of TDD responses that are waiting to be handled by the LA.

8.5.2 *Behavior*

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

When no TDD response is available for the requesting LA, the return status is set to the "nothing to handle" value.

When a TDD response is available, the LA shall allocate an empty TDD which will hold the copy of the TDD response still under CA control.

The following numbers define the various TDD types:

– 1 for a SEND Response;

– 2 for a RECEIVE Response;

– 3 for a TRACE Response;

– 4 for a SUBMIT Response;

– 5 for an EXTEND Response

– 6 for a NATIONAL Response;

– 7 for a PRIVATE Response.

8.5.3 *Parameters*

The following parameters are required:

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 TDD response 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 TDD responses that are waiting to be retrieved by the LA. Null means no TDD is waiting.	Output parameter
Status	integer	Acknowledges the PutTDD function. Return error code (0000 means success). See § 7.6.3.	Output parameter

8.6 *Function GetTDD*

The function GetTDD shall be supported by the CA. It may be invoked by a LA.

8.6.1 *Purpose*

The purpose of the GetTDD function is to retrieve a TDD response from a CA. The CA copies the TDD response into an internal structure of the LA.

8.6.2 *Behavior*

The LA indicates the location of an empty TDD where the CA shall copy a TDD response that is available for the LA.

The CA shall return to the LA the TDD response that was qualified by the previous PollTDD function emitted by the same LA. The LA shall have prepared a recipient TDD response area in its internal structures. The invocation of a GetTDD function by a 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).

8.6.3 *Parameters*

The following parameters are required:

Parameter name	Structure	Comment	Direction
TDD location	memory address	Specifies where the CA can copy the TDD response into the LA's internal structure. When the function is complete, the CA's TDD response may be deleted or used for other purposes.	Input/Output parameter
Connection-ID	integer	The connection identifier returned by the Login function.	Input parameter
Status	integer	Acknowledges the GetTDD function. Return error code (0000 means success). See § 7.6.3.	Output parameter

8.7 *Function SetAlarm*

The function SetAlarm can optionally be supported by the CA. It can optionally be invoked by a LA. If the SetAlarm function is used by a LA, then that LA shall support the CallBackRoutine function Support of the SetAlarm function is declared in the ICE.

8.7.1 *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 a LA/CA dialogue session.

8.7.2 *Behavior*

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.

8.7.3 *Parameters*

The following parameters are required:

Parameter name	Structure	Comment	Direction
CallBackRoutine location	memory address	Specifies the entry point of a particular LA CallBackRoutine function.	Input parameter
Connection-ID	integer	The connection identifier returned by the Login function.	Input parameter
Status	integer	Acknowledges the SetAlarm function. Return error code (0000 means success). See § 7.6.3.	Output parameter

8.8 *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.

8.8.1 *Purpose*

The CallBackRoutine function defines a mechanism that allows a CA to alert the LA that some TDD responses 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.

8.8.2 *Behavior*

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.

8.8.3 *Parameters*

The following parameter is required:

Parameter name	Structure	Comment	Direction
Connection-ID	integer	The connection identifier returned by the Login function	Input parameter

8.9 *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 TDD requests and responses.

8.9.1 *Purpose*

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

8.9.2 *Behavior*

Before completing the LA-CA dialog, the CA may (but is not obliged to) process all pending TDD requests that were put out by that LA.

8.9.3 *Parameters*

The following parameters are required:

Parameter name	Structure	Comment	Direction
Status	integer	Returned by the CA. Indicates whether logout was orderly processed. Return error code (0000 means success). See § 7.6.3.	Output parameter
Connection-ID	integer	The connection identifier returned by the Login function.	Input parameter

9 Incoming/Outgoing Files

For proper operation, some requirements apply to the format of the outgoing and incoming files exchanged locally between LA and CA. Incoming and outgoing files have a specific format which is not necessarily the same as the one used for communications e.g. a word processing file to be exchanged through the basic mode of the Teletex service. In this case, conversions are required (they should be handled by the CA).

9.1 *Transfer formats*

In order to exchange documents between LA and CA several "document transfer formats" are defined by APPLI/COM. The transfer formats apply to the "transfer files" as defined in § 3. These formats should not be confused with the format transmitted through the network (transmission files) by the CCITT services nor should they be confused with the format used by TDDs (TDD Coding).

Several document 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.

APPLI/COM defines:

– 3 text oriented formats (APPLI/COM extended ASCII, T.50 and T.61);

– 1 graphic oriented format (TIFF);

– the transparent transfer format.

The transparent transfer format cannot be used unless the document is to be transmitted as a binary file by the CCITT service. In this case, no conversion of the contents of the file takes place thus the file will be transmitted unchanged.

APPLI/COM is open for the support of private transfer formats. Hence other transfer formats may be implemented by APPLI/COM CA munufacturers to adapt native formats of commonly used application programs (e.g. word processors, databases or spreadsheets). However, the transfer formats supported by CA manufacturers have to be stated in their related documentation. The support of those transfer formats shall also be stated explicitly in the ICE.

Table 22/T.611 indicates how transfer formats shall be supported by the CA depending on the Operating System and the CCITT service the CA offers:

TABLE 22/T.611

List of the transfer formats

Transfer format	Operating system	Teletex	TTX/Telex	Telex	Telefax 3	Telefax 4
APPLI/COM2 Extended ASCII	MS-DOS, OS/2	Yes	Yes	Yes	Yes (Note 1)	Yes (Note 1)
Standard ASCII (T.50)	MS-DOS, OS/2, UNIX, MAC-OS	Yes	Yes	Yes	yes (Note 1)	Yes (Note 1)
T.61	MS-DOS, OS/2, UNIX, MAC-OS	Yes	Yes			
APPLI/COM TIFF	MS-DOS, OS/2, UNIX, MAC-OS				Yes	Yes
Transparent	MS-DOS, OS/2, UNIX, MAC-OS	Yes (Note 2)	Yes (Note 2)		Yes (Note 2)	Yes (Note 2)

Note 1 – Only for outgoing documents.

Note 2 – Only if the CA supports binary transfer via the CCITT service.

The transfer formats defined in this Recommendation can be read and generated for the appropriate services under the appropriate operating systems by APPLI/COM-compatible CAs.

Note that the "text"-oriented transfer formats for the Telefax service are supported in the send direction only.

Documents that are transferred to APPLI/COM 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 atttributes such as "underline", "superscript" and "subscript".

Note – The CA shall reject documents with an incorrect format or character set.

Note – If a LA wants to achieve service-independence the LA should use one of the "ASCII"-based formats for document transfer, since they are the only transfer formats that cover all three services (with the exception of the Telefax service in the receive direction). These transfer formats are very easy to implement. It is only when texts with complex layout are sent / received by Teletex or graphics are sent /received by Telefax that the T.61 transfer format for Teletex or the TIFF transfer format for Telefax shall be implemented by the LA.

9.1.1 *Transfer formats APPLI/COM Extended ASCII and APPLI/COM T.50*

The following format codes are defined for the "ASCII" based transfer formats APPLI/COM Extended ASCII and APPLI/COM T.50:

Format	Possible values	Hex	ASCII	Default
Orientation	Portrait	1B 4F 30	ESC O0	X
	Landscape	1B 4F 30	ESC O1	
Pitch	10 Pitch	1B 50 30	ESC P0	X
	12 Pitch	1B 50 31	ESC P1	
	15 Pitch	1B 50 32	ESC P2	
Line spacing	6 lines/inch (1 spacing)	1B 4C 30	ESC L0	X
	4 lines/inch (1.5 spacing)	1B 4C 31	ESC L1	
	3 lines/inch (2.5 spacing)	1B 4C 32	ESC L2	
	12 lines/inch (2.5 spacing)	1B 4C 33	ESC L3	
Attributes	Underline off	1B 55 30	ESC U0	X
	Underline on	1B 55 31	ESC U1	
	Superscript off	1B 41 30	ESC A0	X
	Superscript on	1B 41 31	ESC A1	
	Subscript off	1B 56 30	ESC V0	X
	Subscript on	1B 56 31	ESC V1	
	Boldface off	1B 42 30	ESC B0	X
	Boldface on	1B 42 31	ESC B1	
	Strike-out off	1B 53 30	ESC S0	X
	Strike-out on	1B 53 31	ESC S1	
	Italics off	1B 49 30	ESC I0	X
	Italics on	1B 49 31	ESC I1	
Text makeup	Fold lines disallowed	1B 54 30	ESC T0	
	Fold lines allowed	1B 54 31	ESC T1	X
	Rotate page disallowed	1B 52 30	ESC R0	
	Rotate page allowed	1B 52 31	ESC R1	X
	New Line	0D 0A	CR LF	
	New page	0D 0C	CR FF	

Note that not all services permit all the format specifications listed (see § 9.1.4).

Note – 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 turn 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.

9.1.1.1 *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 23/T.611 to 25/T.611[7].

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.

[7] 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.

Otherwise, only the lower pan of the table (i.e. octets from 20hex to 7Fhex) is guaranteed to be faithfully converted.

Note – The Tables 23/T.611 to 25/T.611 shall be read by selecting first a column, then a row, i.e. character "A" is column 4, row 1.

TABLE 23/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	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0			SP	0	@	P	'	p	Ç	É	á	░	└	╨	α	≡
1			!	1	A	Q	a	q	ü	æ	í	▒	┴	╤	β	±
2			"	2	B	R	b	r	é	Æ	ó	▓	┬	╥	Γ	≥
3			#	3	C	S	c	s	â	ô	ú	│	├	╙	π	≤
4		¶	$	4	D	T	d	t	ä	ö	ñ	┤	─	╘	Σ	⌠
5		§	%	5	E	U	e	u	à	ò	Ñ	╡	┼	╒	σ	⌡
6			&	6	F	V	f	v	å	û	ª	╢	╞	╓	µ	÷
7			'	7	G	W	g	w	ç	ù	º	╖	╟	╫	τ	≈
8			(	8	H	X	h	x	ê	Ÿ	¿	╕	╚	╪	Φ	°
9			)	9	I	Y	i	y	ë	Ö	⌐	╣	╔	┘	Θ	∙
A	LF		*	:	J	Z	j	z	è	Ü	¬	║	╩	┌	Ω	·
B		ESC	+	;	K	[	k	{	ï	¢	½	╗	╦	█	δ	√
C	FF		,	<	L	\	l	|	î	£	¼	╝	╠	▄	∞	ⁿ
D	CR		-	=	M	]	m	}	ì	¥	¡	╜	═	▌	φ	²
E			.	>	N	^	n	~	Ä	Pt	«	╛	╬	▐	ε	■
F			/	?	O	_	o		Å	*f*	»	┐	╧	▀	∩	

T0811900-93/D05

Note – The "Backslash" ($5C_{Hex}$) and "Braces" ($7B_{Hex}$ and $7D_{Hex}$) character, 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.

Conversion direction	Action performed
Outgoing	All characters shown in Table 23/T.611 are accepted into the Telefax service character set.
Incoming	The transfer format is not generated in this direction (except such configurations where the CA is capable of OCR – Optical Character Recognition).

TABLE 24/T.611

Character Set of APPLI/COM Extended ASCII transfer format for the Teletex service

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

T0811910-93/D06

Conversion direction	Action performed
Outgoing	All characters shown in Table 24/T.611 are accepted into the Telefax service character set.
Incoming	The transfer format is not generated in this direction (except in such configurations where the CA is capable of OCR – Optical Character Recognition).

TABLE 25/T.611

Character Set of APPLI/COM Extended ASCII transfer format for the Telex service

Hex	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0			SP	0		P		p								
1				1	A	Q	a	q								
2				2	B	R	b	r								
3				3	C	S	c	s								
4				4	D	T	d	t								
5				5	E	U	e	u								
6				6	F	V	f	v								
7			'	7	G	W	g	w								
8			(	8	H	X	h	x								
9			)	9	I	Y	i	y								
A				:	J	Z	j	z								
B			+		K		k									
C			,		L		l									
D			-	=	M		m									
E			.		N		n									
F			/	?	O		o									

T0811920-93/D07

Conversion direction	Action performed
Outgoing	All characters shown in Table 25/T.611 are accepted into the Telex service character set.
Incoming	In the receive direction, letters are always encoded in lowercase (Codes 61Hex to 7AHex). Uppercase letters are not generared.

9.1.1.2 *Character Set of the T50 Transfer Format (Standard ASCII)*

The characters supported by the T50 transfer format for the Teletex, Telex and Telefax services are shown in Tables 26/T.611 to 28/T.611[8].

TABLE 26/T.611

Character Set of the APPLI/COM T.50 transfer format for the Telefax service Group 3 and 4

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

T0811930-93/D08

Conversion direction	Action performed
Outgoing	All characters shown in Table 26/T.611 are accepted into the Telefax group 3 and group 4 services character set.
Incoming	The transfer format is not generated in this direction.

[8] The T.50 character set is indicated with the US ASCII character set. It is defined as T.50 IRV (International Reference Version) per Recommendation T.50.

TABLE 27/T.611

Character Set of the APPLI/COM T.50 transfer format for the Teletex service

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

T0811940-93/D09

Note – The "Backslash" (5CHex) and "Curly Brackets" ($7B_{Hex}$ 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.

Conversion direction	Action performed
Outgoing	All characters shown in Table 27/T.611 are accepted into the Teletex service character set.
Incoming	In the receive direction, characters can come that are not contained in the Table 27/T.611. Such characters may be replaced by the character "?" (3FHex).

TABLE 28/T.611

Character Set of the APPLI/COM T.50 transfer format for the Telex service

Hex	0	1	2	3	4	5	6	7
0			SP	0		P		p
1			!	1	A	Q	a	q
2				2	B	R	b	r
3				3	C	S	c	s
4				4	D	T	d	t
5				5	E	U	e	u
6				6	F	V	f	v
7			'	7	G	W	g	w
8			(	8	H	X	h	x
9			)	9	I	Y	i	y
A				:	J	Z	j	z
B			+		K		k	
C			,		L		l	
D			-	=	M		m	
E			.		N		n	
F			/	?	O		o	

T0811950-93/D10

Conversion direction	Action performed
Outgoing	All characters shown in Table 28/T.611 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 61Hex to 7AHex). Uppercase letters are not generated.

9.1.2 *APPLI/COM transfer format T.61*

The APPLI/COM T.61 transfer format exactly corresponds to Recommendation T.61 for the Teletex service. Documents that the application transfers in this format are mapped to the Teletex service as is. The format and codes are checked but there is no conversion. This format is suitable for applications that either already generate this format or are meant to transmit complex text layouts via the Teletex service.

9.1.3 *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 include the attributes that describe the image, such as resolution and spread, 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 writes 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.

Due to 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 TIFFs development. For this reason, APPLI/COM gives a definition of the profile taken as a basis for an APPLI/COM-compatible TIFF file. This definition closely follows version 5 of the TIFF standard, the TIFF format class B.

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

9.1.3.1 *APPLI/COM TIFF Profile*

The tags recognized by a CA and their handling are summarized in the following TIFF Profile table (see Table 29/T.611).

TABLE 29/T.611

APPLI/COM TIFF profile table

Tag		Reader		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	PhotometricInterp.	0, 1	0	0	0	0	0
107	Thresholding	Ignore	Ignore	None	None	None	None
108	CellWidth	Ignore	Ignore	None	None	None	None
109	CellLength	Ignore	Ignore	None	None	None	None
10A	FillOrder	1, 2 (Note 5)	1	1	1	2	2
10D	DocumentName	Ignore	Ignore	None	None	None	None
10E	ImageDescription	Ignore	Ignore	None	None	None	None
10F	Make	Ignore	Ignore	None	None	None	None
110	Model	Ignore	Ignore	None	None	None	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'

TABLE 29/T.611 (*cont.*)

APPLI/COM TIFF profile table

Tag		Reader		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)
117	StripByteCount	'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'
11C	PlanarConfiguration	1	1	1	1	1	1
11D	PageName	Ignore	Ignore	None	None	None	None
11E	XPosition	Ignore	Ignore	None	None	None	None
11F	YPosition	Ignore	Ignore	None	None	None	None
120	FreeOffsets	Ignore	Ignore	None	None	None	None
121	FreeByteCount	Ignore	Ignore	None	None	None	None
122	GrayResponseUnit	Ignore	Ignore	None	None	None	None
123	GrayResponseCurve	Ignore	Ignore	None	None	None	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'

TABLE 29/T.611 (*fin*)

APPLI/COM TIFF profile table

Tag		Reader		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)
12C	ColorResponseUnit	Ignore	Ignore	None	None	None	None
12D	ColorResponseCurve	Ignore	Ignore	None	None	None	None

Note 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.

Note 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 Telefax standard**),** the value 1728for ImagcWidth shall not be exceeded**.**

Note 3 – APPLI/COM guarantees the processing of ImageLength if thefollowing 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.

Note 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 optionalfeature of the CA.

– 3/4 CCITT Group 3/4 compression is guaranteed only if the resolution value (XResolution, YResolution) exactly corresponds to the resolutions of the Telefax services selected 32 773 (pack-bits compression) is an optional feature of the CA.

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

Note 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.

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

The table below summarizes the rules applying to the TIFF Profile Table.

Column	Rule
Tag	Identifies the name of the tag and it's 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 acepted 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 not given. 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.

9.1.4 *Service Constraints applying to Transfer Formats*

Due to the nature of the CCITT services selected, several constraints are applied to the transfer formats. These constraints are summarized below:

CCITT 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 specifies 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 conjuction with a TFT (Telematic File Transfer) type selection.
Telefax (G3/G4)	ASCII, T.50 (véase la nota)	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.

Note – Only applicable in outgoing direction.

9.2 *Files descriptions*

To help correct processing of the incoming/outgoing file, it may be useful to provide more information than is already included in the file. For instance, the size of the pages or the resolution of a facsimile document are not found inside the document. As a consequence, the entity that processes the document may be obliged to scan it (case 1 of Figure 5/T.611).

Thanks to APPLI/COM, it is also possible to group this information inside the TDDs (case 2 of Figure 5/T.611). Figure 5/T.611 illustrates it in the case of the outgoing files, interchanged from the LA to the CA.

Conversely, when a LA receives an incoming file from the CA, the LA either reads the information contained in the TDD (case 2) when provided or must scan the incoming file (case 1).

The manufacturer shall state in its documentation which method is used to exchange the description of the outgoing/incoming files between LA and CA.

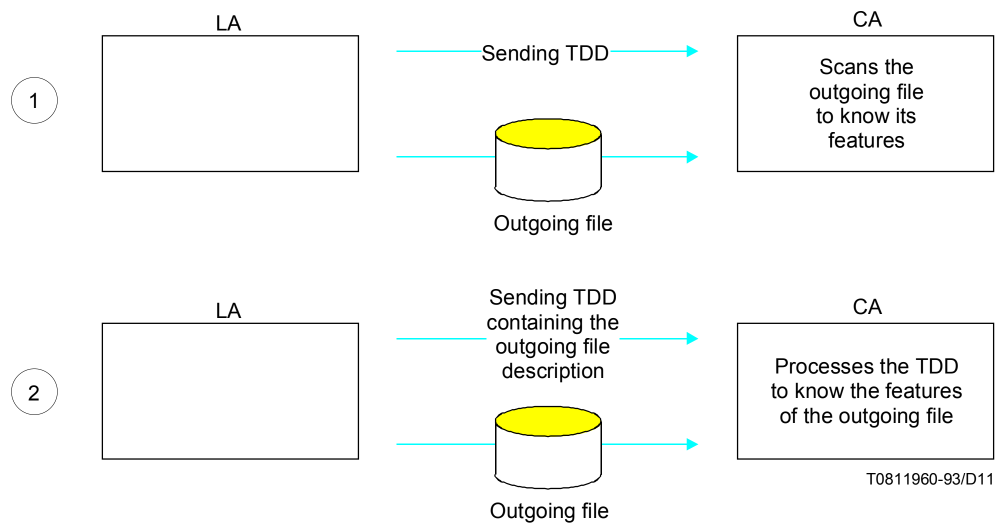
Note – However including the description inside the TDD is an additive feature of the CA; in any case, the description can be passed in the document itself as defined by the transfer formats (see § 7.1).

In case 2 of previous figure, the information describing the features of the incoming/outgoing file are contained in the TDD and are composed of the following parts:

– general information concerning the incoming/outgoing file (file name, file size, etc.);

– specific features of the communication (depending on the service, like resolution, page size, etc.);

– private information (up to the manufacturer providing the CA).



LA
CA
1
Sending TDD
Scans the outgoing file to know its features
Outgoing file
LA
CA
2
Sending TDD containing the outgoing file description
Processes the TDD to know the features of the outgoing file
Outgoing file
T0811960-93/D11

FIGURE 5/T.611

Description of files exchanged between LA and CA

10 Communications Control – CA-Record

Since the interchange mechanism between a LA and a CA is based on a “client-server” model, the LA always originates TDD requests to a CA. TDD responses from the CA are not spontaneous; the LA shall poll the CA in order to know whether any TDD responses 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 in 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 request, function “Reschedule” 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.

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

10.1 *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.

The following table shows the minimum list of CA-Record fields that any CA shall support:

Field name	Syntax	Purpose
COM-ID	COM-ID	Maintains the unique communication identifier assigned to the CA-Record.
LA-ID	La-id	Assigns the CA-Record tothe 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.

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 6 following states:

State name	Concerns	Means
"delayed"	Transmission	The record has not yet been processed by the CA. It waits until the CA moves it in the sending state.
"sending"	Transmission	The record is being 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 record has successfully been transmitted to the recipient.
"failed"	Transmission Reception	The record failed to be transmitted, (or) errors occurred during reception (or) an internal error occurred in the CA.
"reception"	Reception	The record is being received by the CA at that time, or, has been received but not retrieved by the recipient LA.
"retrieved"	Reception	The record 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 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 the DRF (Dispatch Received Files) facility, 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.

10.2 *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 identifier to each new CA-Record; it 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 DATE-TIME fields of the SEND request 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 the "sending" state at a given moment.

Figure 6/T.611 below illustrates these state transitions.

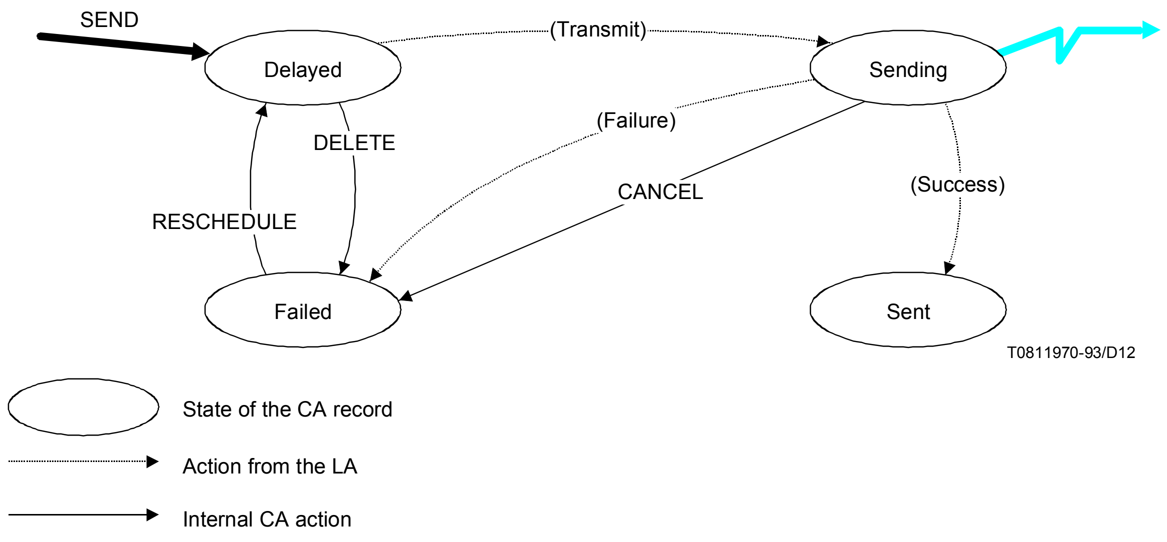


FIGURE 6/T.611

State transitions of the CA-Record (transmissions)

10.3 *Reception process – State transitions*

When a CA receives an incoming conununication from the network, it builds a CA-Record. This CA-Record is immediately 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 occurs while the reception is processed, the CA-Record is immediately switched to the "failed" state. Otherwise the CA-Record is switched to the "retrieved" state when an 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.

Fgure 7/T.611 ilustrates these state transitions.

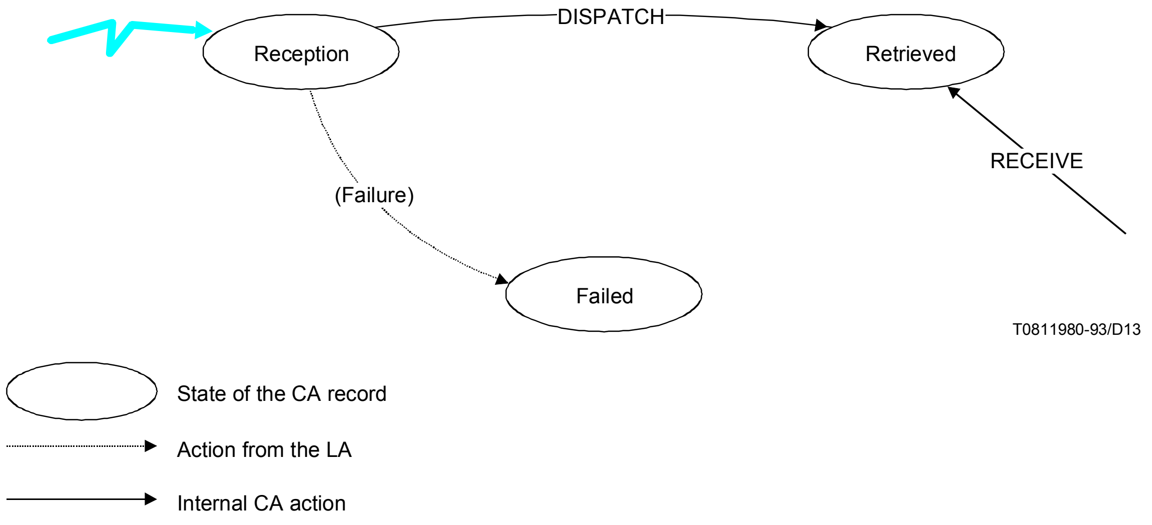


FIGURE 7/T.611

State transitions of the CA-Record (reception)

10.4 *Actions – Notation conventions*

For the sake of legibility, the following conventions apply for the notation of the various actions involving CA-Records.

Notation	Means
TRACE:DELETE rq	Request "TRACE", operation "DELETE".
Recipient LA	The LA to which an incoming call was dispatched.
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.

10.5 *Actions – Transmissions*

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

10.5.1 *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:

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 "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"

10.5.2 *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:

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"

10.5.3 *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:

Action	Originator	Purpose	Resulting state
TRACE:PURGE rq	Originating LA	To remove all the CA-Records being in the "sent" state.	(none)
TRACE:COPY rq	Originating LA	To build a logical file containing a copy of the list of the "sent" CA-Records.	"sent"

10.5.4 *Failed state – Transmissions*

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 to the reception process. See below for relevant information.

The following actions may apply to a CA-Record in the "failed" state:

Action	Originator	Purpose	Resulting state
TRACE:PURGE rq	Originating LA	To remove Ca-Records being in the "failed" state.	(none)
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"

10.6 *Actions – Receptions*

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

10.6.1 *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:

Action	Originator	Purpose	Resulting state
TRACE:DISPATCH rq	Any LA	To assign a LA-ID to a received CA-Record.	"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"

10.6.2 *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:

Action	Originator	Purpose	Resulting state
RECEIVE rq	Recipient LA	To retrieve all information relevant to the CA-Record (e.g. documents, originator, etc.)	"retrieved"
TRACE:PURGE rq	Any LA	To remove CA-Records which are in the "retrieved" state.	(none)
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"

10.6.3 *Failed state – Receptions*

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 to the transmission process. See § 10.5.4 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.

The table below details the actions that may apply on the CA-Records in the "failed" state:

Action	Originator	Purpose	Resulting state
TRACE:PURGE rq	Any LA		(none)
TRACE:COPY rq	Any LA	To build a logical file containing a copy of the list of the CA-Records in "failed" state. The LA sees "all" CA-Records in this state.	(none)

ANNEX A

(to Recommendation T.611)

Syntax used for describing APPLI/COM grammar

(Normative)

All objects defined in this Recommendation are described in a BNF like grammar. The grammar rules are given in the next paragraph and the punctuation elements are described in the Table A-1/T.611.

All lexical elements are delimited in a double quote sign and the grammar rules apply to the objects which are constituted of these terminal elements. These elements are to be considered as they are written.

Any object could be generated from a combination of the production rules combined with the terminal elements.

TABLE A-1/T.611

General syntax elements

Item	Means
<new definition>	Denotes a production rule.
“Terminal Element”	The string between the quotes shall be included in the concrete encoding (delimiting quotes shall not be present).
;	Denotes a separator. Can be implemented as the “semicolon” character or the “carriage return, line feed” combination of characters.
#	The “#” character shall be included in the encoding.
--	Denotes a comment in the syntax description (not in the encoding). A comment starts with “--” and finishes with either the end of the line or the next occurrence of “--”.
:=	Denotes the “assignment sign” following the production rule.
*	Denotes an item that can be repeated 0 or many times.
+	Denotes an item that can be repeated 1 or many times.
{xxx}	Everything between opening and closing brace is optional.
(xxx)	Denotes a group of syntax elements.
|	Denotes a “or” connector i.e. encoders shall choose between alternatives (“|” is not encoded).
STRING	Is a character string based on the character repertoire defined in the APPLI/COM header of the ICE.
SIZE(0...7)	Denotes a constraint on a string to be from 0 to 7 octets long.

Figure A-1/T.611 defines the general ICE grammar.

```
<ICE> :=                     <applicom-header> <end-of-line>+ <CA-Descriptor>+

<applicom-header> :=         -- see definition in Annex C
                             -- MUST be first element of file

<CA-Descriptor> :=           "#" <end-of-line>+ (<keyword-parameter-pair> <end-of-line>+ )+

<keyword-parameter-pair> :=  <keyword> ":" <parameter> ("," <parameter>)*

<keyword> :=                 STRING(SIZE(1..16))

<parameter> :=               STRING(SIZE(1..255))
                             -- if a semicolon character (“;”) is required inside the value
                             -- field, it shall be escaped by the backslash character (“\”)

<end-of-line> :=             {";" STRING(SIZE(0..255)) };
```

FIGURE A-1/T.611

Syntax of the ICE

Figure A-2/T.611 defines the general TDD grammar.

```
<tdd> :=                      <applicom-header> <end-of-line>+
                              <function> <end-of-line>+
                              <keyword-parameter-pair>+
                              -- relevant to the "function"
                              -- and the CCITT service used

<applicom-header> :=          -- see Annex C
                                                -- MUST be first element of file

<function> :=                 "FUNCTION:" ("Send" | "Receive" | "Send back" | "Purge" | "Delete" | "Extend" | ("Copy"
                              | "Cancel" | "Reschedule" | "Dispatch" | "Print" | "Convert" | "National" | "Private" |)
                              -- describes which TDD is invoked

<keyword-parameter-pair> :=   <keyword> ":" <parameter> <end-of-line>+
                              -- described by means of tables further on

<keyword>: =                  STRING (SIZE (1..16))

<parameter> :=                STRING (SIZE (1..255))
                              -- if a semicolon character (";") is required inside the value
                              -- field, it shall be escaped by the backslash character ("\")
<end-of-line> :=              {";" STRING(SIZE (0..255)) };
```

FIGURE A-2/T.611

Syntax of the TDDs (requests and responses)

The FileSpec parameter shall be structured as described in Figure A-3/T.611. This structure allows the description of multiple outgoing files by repetition of the FileSpec parameter, along with its related keyword "FILELIST". It allows multiple outgoing files, with their own transfer format, to be exchanged within one or many communication sessions (mainly used for the Teletex and Facsimile group 4).

```
<FileSpec> :=          <document> | "," <document> | * <end of line>

<document> :=          <filename> "(" <transfer-format>

                       | "," <type> | "," <session-number> || ")"

<filename> :=          Path
                       -- the full path of the file

<transfer-format> :=   Convert-id
                       -- the transfer format of the file
<type> :=              Type-id
                       -- the type of the file

<session-number> :=    integer
                       -- the number of the session in which to send the document
                       -- used for the Teletex service only
```

FIGURE A-3/T.611

Syntax of the FileSpec parameter

The T61 options parameter shall be structured as described in Figure A-4/T.611.

```
<T61options> :=  string
                 -- for further study
```

FIGURE A-4/T.611

Syntax of the T61options parameter

ANNEX B

(to Recommendation T.611)

Conventions for the notation of call numbers ("Address")

(Normative)

APPLI/COM interfaces support many CCITT services. Depending on national choices, CCITT services are provided on different network types (e.g. PSTN, ISDN, etc.).

Dialing sequences that must be carried out to reach an addressee depend on those network types, the CCITT service and on the location of the addressee. The notations of calling numbers usually reflect those dependencies and variations.

Dialing sequences result from the translation of that notation into a "raw dialing number" that include traffic discrimination digits.

This annex intends to standardize the call number (parameter of the "ADDRESS" keyword) for each CCITT service covered by the APPLI/COM interfaces.

APPLI/COM indicates the presence of the call number as raw dialing number 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).

B.1 *Teletex and Telefax Group 4 services*

Within these services, a "terminal ID" is defined. This identification is specified for these services in the parameter field of the "ADDRESS" keyword.

The terminal ID is defined in Recommendation F.200 (for Teletex) and Recommendation F.184 (Telefax Group 4).

When the mnemonic component is input, a subscriber identification verification is performed. The identification verification can be omitted by either entering a question mark in place of the mnemonic component, or leaving the mnemonic component empty.

B.2 *Telex service*

The call number is entered as parameter of the "ADDRESS" keyword. 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 ("=").

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.

B.3 *Telefax Group 3 service*

The call number is entered as a parameter of the keyword “ADDRESS”. The pure telephone dialing sequence is entered. If the dialing sequence starts with a “!” then it may contain special characters that are treated as operators (or “modifiers”) rather than dial digits (see Table B-1/T.611). It is not possible to perform a subscriber identification verification for this service.

TABLE B-1/T.611

Dial string modifiers for the PSTN

Digits	Type	Explanation
0...9	Dial digit	Dial withouth interpretation.
!	Operator	First character in the dialing sequence string: enter the “raw dialing mode”. Any other oposition: “hook flash”. Go on-hook (for a specific duration) and off-hook (for a specific duration).
;	Operator	Pause in the dialing process. Duration is CA dependent. Note that the “,” must be escaped by the “\” character, otherwise the CA may understand it as a comment introducer.
:	Operator	Same as “;”. Does not need to be escaped.
,	Operator	Pause dialing for 2 seconds.
T, t	Operator	Enforce “tone” dialing for subsequent digits.
P, p	Operator	Enforce “pulse” dialing for subsequent digits.
W, w	Operator	CA waits and listen for a 3 seconds, continuous dial tone.
@	Operator	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.

ANNEX C

(to Recommendation T.611)

APPLI/COM header

(Normative)

Within the TDDs and within the ICE the "APPLI/COM header" is always the first element specified. The "APPLI/COM header" is laid out as follows:

x*APPLI/COM*yyyy*CCITT"Reserved"

where:

x	stands for the "Code ID" (see Table C-1/T.611)
yyyy	reflects the date of the publication of the T.611 Recommendation (1992)
Reserved	is a string of 16 octets reserved for further extensions of this Recommendation
*	is a separator.

TABLE C-1/T.611

APPLI/COM header

Code ID Presentation	Code ID Value	Comment
A	41_{hex}	8 bit ASCII APPLI/COM Extended ASCII.
B	42_{hex}	Reserved for National Variations of IRA coding.
E	$C5_{hex}$	International EBCDIC Coding 8 bit.
I	49[a]$_{he}$x	7 bit ASCII, IRA as defined in Recomendation T.50. This coding is the APPLI/COM default.
P	50_{hex}	Private coding rules.
S	53_{hex}	ASN.1 BER.1 (Basic Encoding Rule number 1) as described in Recommendation X.209.

a) Ignore first bit.

ANNEX D

(to Recommendation T.611)

Location of the ICE

(Normative)

The Interface Configuration Environment (ICE), represents a "global" source for all local applications (LAs) conforming to this Recommendation.

On the operating systems UNIX, MAC-OS and the operating systems belonging to the Microsoft family (OS/2, MS- DOS, etc.) the ICE is represented by a file[9]. This file shall be opened and read by the application in order to extract the information about the APPLI/COM interface(s) accessible within the system during start-up. The name of the file on the systems mentioned above is "APPLICOM.ICE".

The file is located differently on the above-mentioned systems. On some systems there exists an "environment variable[10] ", which may hold the path to the ICE. In those cases the variable is named "APPLICOM" (stated in capital letters). Following algorithms for determining the ICE location shall be used (see Figure D-1/T.611).

TABLE D-1/T.611

Location of the ICE

Operating system	Algorithm
UNIX	First look in the Environment variable "APPLICOM". If there is no variable, then the ICE is contained in the /**dev** subdirectory and named "APPLICOM_ICE".
MAC-OS	The ICE is always contained in the **System** folder.
MS-DOS	Look in the Environment variable "APPLICOM". If there is no variable, 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].

[9] A "file" in this context means either a real operating system file or an operating system device driver, which behaves exactly as a file.

[10] Operating-Systems like UNIX or MS-DOS are providing a so called "Environment" which consists of a set of ASCII strings applied to a ASCII represented variable.

ANNEX E

(to Recommendation T.611)

Modes of Information Interchange

(Normative)

"Concrete description of the basic Exchange Mechanism functions"

In order to assist LA developers, the APPLI/COM basic exchange mechanism functions have been defined. They are built in a manner, which allows third party suppliers or CA providers to make available libraries for several computer languages. These libraries then may be linked to LA's code.

The basic exchange mechanism functions uncouple the real access method from the code of the LA. Interfacing to the real access method provided by the CA is done inside of the basic exchange mechanism functions. So, if a LA decides to interface with APPLI/COM through that functions, it stays independent of the real access method used.

The basic exchange mechanism functions roughly resemble the access functions of the APPLI/COM primitive access method. So they are bound to this access method with little overhead. Since the binding to the file access method is accomplished by emulating the primitive access method more overhead is involved.

The usage of the basic exchange method functions is shown in Figure E-1/T.611:

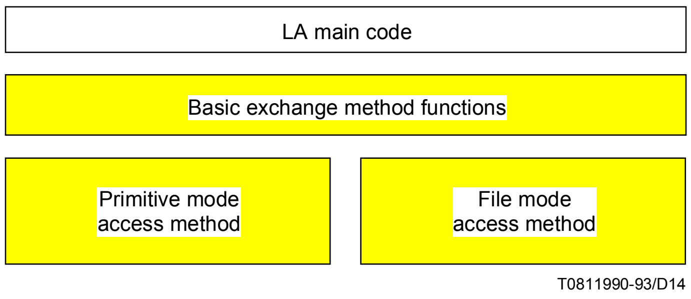


FIGURE E-1/T.611
Usage of the basic exchange method functions

The basic exchange mechanism functions are shown in detail below:

- Login Function;
- PutTDD Function;
- PollTDD Function;
- GetTDD Function;
- Logout Function;
- SetAlarm Function.

E.1 *File mode*

If a CA runs in file mode, three areas (paths or subdirectories) have to be configured inside of the CA for each possible LA which may connect to the CA:

- An input area for jobs that are transferred from the LA to the CA;
- An response area for processed jobs that are then placed in this area by the CA. In this case, processed means that the CA regards them as finished, but not necessarily executed successfully (e.g. documents not sent);
- An error area for jobs that could not be processed due to syntactical errors or other errors. The CA removes incorrect TDDs from the input area (see above) and places them in this area.

These three areas are refer-red to symbolically below as /COM_JOB, /COM_ACK and /COM_ERR.

Because they are implementation- and installation-specific, the complete paths to these areas must be designed so as to be configurable during Installation of the CA.

If a CA has to support multiple LA connections in file mode, 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. Hereby the name of each sub-area shall be derived from the LA-ID of the connecting LAs.

Note – The LA programmer is strongly urged to encapsulate the steps for job handling mentioned below in individual modules or functions, since the physical interface to APPLI/COM is located at these points and it is naturally this interface which is most likely to be affected by possible code changes due to porting. To achieve this, the best solution is lo rely on the basic exchange method functions interface described before.

E.1.1 *Job Transfer*

The LA generates the TDD. This TDD, which completely describes the details of the job by the keyword/parameter pairs, is then placed into the /COM_JOB area as a file.

The LA calls the commuunications software through a specific mechanism, called "SYNC" below. Depending on the type of communications software, SYNC can either compel an active task or can simply be triggered "pro forma".

Note that, if the basic exchange method functions are used, both steps mentioned (putting the TDD and performing a SYNC) we carried out by the PutTDD function.

The CA takes the envelope, "scans" the associated keywords and processes the job.

If a Response-TDD was requested by the LA and if the status is final, the CA generates the Response-TDD. Parameters for return fields (already filled with spaces) that were transferred are appropriately filled by the CA.

The CA places the Response-TDD in the /COM_ACK response area and deletes the Request-TDD in /COM_JOB input area.

The name of the Response-TDD is the same as that of the Request-TDD. However, this does not apply if the keyword REQ-ID (Request-id) stipulates another name for the formation of the Response-TDD name.

The LA now can access the Response-TDD and learn the status of the function call.

Note that, if the basic exchange method functions are used, the action of requesting and retrieving a Response-TDD are carried out through the PollTDD and GetTDD functions.

The complete circle of the job transfer is shown in Figure E-2/T.611.

Note – APPLI/COM does not guarantee that Response-TDDs will be provided in the same as request-TDDs. Processing is not necessarily sequential.

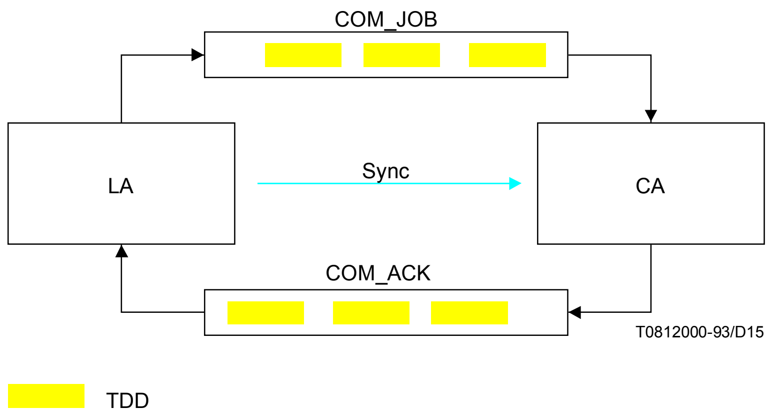


FIGURE E-2/T.611

How TDDs are interchanged with the File Method

E.1.2 *Error Handling*

Request-TDDs that cannot be processed 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 envelope and an error message with the form [Num/Text] is generated in the appended parameter field.

E.1.3 *Sync Mechanism*

In the real world of the APPLI/COM interface, there must be a mechanism that allows both the LA and the CA to execute their program code. This means that the two tasks must be able to work simultaneously or alternately, independent of whether there is a requester-server dependency or the tasks operate disassociated from one another. It is simply a question of the distribution of processor capacity.

Under multitasking and time-sharing operating systems such as OS/2 or UNIX, this possibility is offered by the operating system itself. In this case, both tasks can run (more or less) in parallel.

Under single-tasking operating systems such as MS-DOS, only one task can run at a time. If another task must be executed, the task that is running must either be terminated or suspended (initiate sub-task and terminate).

Since the APPLI/COM interface exists under various operating systems, it is not possible to have a general and open-ended way in which program control can be allocated to the CA. Nevertheless, APPLI/COM provides general specifications for the LA programmer so that the application be portable.

Each CA supplier has designed and disclosed "his own" SYNC mechanism. Under the DOS, OS/2, UNIX and UNIX-like operating systems, all CA suppliers implemented the SYNC call as an "EXEC" procedure call. Thus, in this case, the SYNC is implemented by calling the "APPLI/COM" program. Other mechanisms may be required for other systems. For the SYNC mechanisms for these operating systems, see the documentation of the individual supplier.

A short treatment of possible SYNC mechanisms is given below. Since the type of SYNC mechanism depends on the operating system, they are classified as follows:

E.1.3.1 *Multitasking operating systems*

In this case, the two tasks can run in parallel, a real SYNC is not absolutely necessary. The application can place its job envelope in the /COM_JOB area and poll the /COM_ACK area from time to time for a response envelope. The communications software polls the /COK_JOB area and places the processed response envelope in the /COM_ACK area.

In order to avoid the overhead caused by polling, some suppliers have implemented a more extensive mechanism such as UNIX-Pipes or Shared Memory.

E.1.3.2 *Single-tasking operating system: MS-DOS*

The way the SYNC mechanism operates, the communications software is called from the application (DOS "EXEC" system call) or the progams are called alternately via a batch job or program chaining (one program calls the other).

A simple solution is the manual, alternate "running" of the two programs.

Another (complex) solution is to relocate the COM program to the background and to call from the application via a software interrupt. A less complex method is to call the communications software from the application. When the communications software has finished, the application continues to run from the calling position.

The file mode is the mandatory exchange method that the CA running under the MS-DOS environment must support. The primitive mode can also be offered optionally.

E.1.3.3 *Other single-tasking systems*

In the case of networked single-tasking systems (e.g. DOS-LANs), there is basically the possibility of running the communications software on a PC (permanently) and running one or more applications from other PCs. This is the (rudimentary) equivalent of the first solution.

E.1.4 *Programming the SYNC Mechanism*

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).

Both SYNC calls shall be issued by the LA, independent of whether the CA requires them or not.

Under DOS, OS/2, UNIX and UNIX-like operating systems, the SYNC mechanism must be implemented by calling the "APPLICOM" program from the LA.

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

– "SyncJob"': "'EXEC" APPLICOM JOB

– "SyncAck": "EXEC" APPLICOM ACK

"EXEC" stands for the operating system specific calling of the EXEC procedure (under MS-DOS: "Load and Execute Program", Call #4B$_{\text{HEX}}$, Int 2I$_{\text{Hex}}$).

E.2 *Primitive mode*

Communication Applications are responsible to provide the right tools which allow their use by Local Applications, in a T.APPLI/COM fashion.

These tools may vary from one provider to the other. Therefore LAs may not predict which tool is implemented by a particular CA. LAs shall thus rely on the ICE file as explained in the body part of this Recommendation. What follows explains how to use the various exchange mechanisms based on a primitive approach, on different operating systems.

E.2.1 *MS-DOS environment*

Three possible mechanisms can be implemented under the MS-DOS environment. The first one relies on the provision of "libraries", the second is based on "software interrupts", and the third involves "system drivers".

However, the primitive mode is an option under the MS-DOS environment since the file mode is mandatory for all supporting CAs running under on MS-DOS.

E.2.1.1 *Libraries*

The CA provider implements the CA and gives a "library". The library collects the necessary function entry points giving service to the exchange mechanism.

The CA may itself be implemented as a program or a stand-alone application. Libraries do not impact on the underlying structure of the CA applications. The LA programmer does not see any difference between these two styles of implementation (see Figure E-3/T.611).

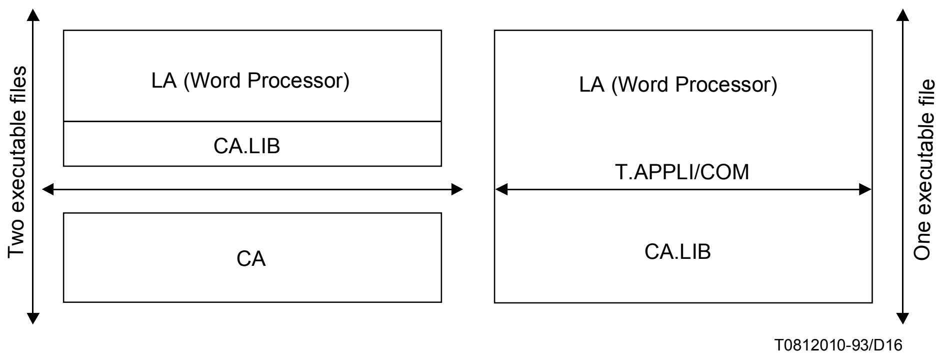


FIGURE E-3/T.611

Implementing a library hides the underlying structure to LAs

To some extent, the library approach can also hide from the LA programmer a CA implementation based on software interrupts or a system driver mechanism (which are both explained further on). The library method is therefore the recommended practice for CA manufacturers.

The following functions (using C language conventions and notation) shall be offered in the library[11].

```
typedef int Connection_Id;
Connection_Id Login (char *Login_name, char *Password);
int PutTDD (Connection_Id c_id, char *TDD_Location, int TDD_Size);
int PollTDD (Connection_Id c_id, int *TDD_Size, char *TDD_Type, int *TDD_Count);
int SetAlarm (Connection_Id c_id, int (*CallBackRoutine) ( ) );
int Logout (Connection_Id c_id);
```

In turn, the LA shall implement the function **CallBackRoutine**( ) if it wishes to use the **SetAlarm**( ) function (completion mechanism).

E.2.1.2 *Software interrupts*

Refer to Appendix I for more details.

[11] The syntax (case) of the function names is essential for interworking.

E.2.1.3 *Character Device Driver*

The CA provider offers a character-oriented device driver allowing any LA to use the CA. This Recommendation recommends the CA provider offer only three functions:

– open;

– IOCTL;

– close.

The LAs are thus invited to fetch the device name in the ICE, and use the functions stated above. The following describes how to invoke the services of such a driver. (See Figures E-4/T.611 to E-6/T.611.)

```
mov     dx,             seg DeviceName      ; DeviceName contains the name of the driver
mov     ds,             dx
mov     dx,             offset DeviceName
mov     al,             02h                 ; access mode: OPEN_ACCESS_READWRITE
mov     ah,             3Dh                 ; open the driver
int     21h                                 ; call DOS
jc      error_handling                      ; if Cy is set, then an error occured
mov     DriverHandle,   ax                  ; DriverHandle contains the name of the open driver
```

FIGURE E-4/T.611

Sample machine language code used to invoke the "Open" function

```
mov     bx,             DriverHandle
mov     ah,             3Eh                 ; close the driver
int     21h                                 ; call DOS
jc      error_handling                      ; if Cy is set, then an error occured
```

FIGURE E-5/T.611

Sample machine language code used to invoke the "Close" function

```
mov     bx,             DriverHandle
mov     cx,             MaxByte             ; number of bytes to send
mov     dx,             seg Buffer          ; Buffer contains data to transmit
mov     ds,             dx
mov     dx,             offset Buffer
mov     ax,             4403h               ; send data to driver
int     21h                                 ; call DOS
jc      error_handling                      ; If Cy is set, then an error occured
mov     ActualByte,     ax                  ; returns actual number of bytes transmitted
```

FIGURE E-6/T.611

Sample machine language code used to invoke the "IOCTL" function, in the case of a Request (from LA to CA)

E.2.2 *WINDOWS 3 Environment*

Various mechanisms are available under this environment:

– The librairies (static or dynamic). The CA provider offers a DLL library allowing any LA to access the services of the CA. LA must be compiled or programmed in order to use the DLL. Under the Windows 3 environment, the provision of a DLL-based CA is mandatory to comply with this Recommendation. Other exchange methods, such as those explained below, are optional. (See Figure E-7/T.611.)

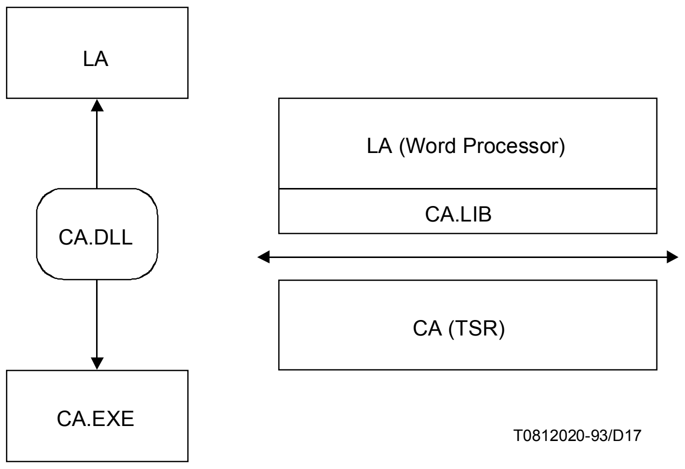


FIGURE E-7/T.611

CA.EXE is a WINDOWS application, CA (TSR) can be a DOS application

– Dynamic Data Exchange (DDE): The CA provider implements the DDE protocol. Any LA aware of the DDE protocol can use the ser vices of the CA, provided they understand the same subjects and topics as stated in the ICE. This uses the inter nal Windows messaging mechanisms and is one of the most efficient way to implement the client server model in this environment. (See Figure E-8/T.611.)

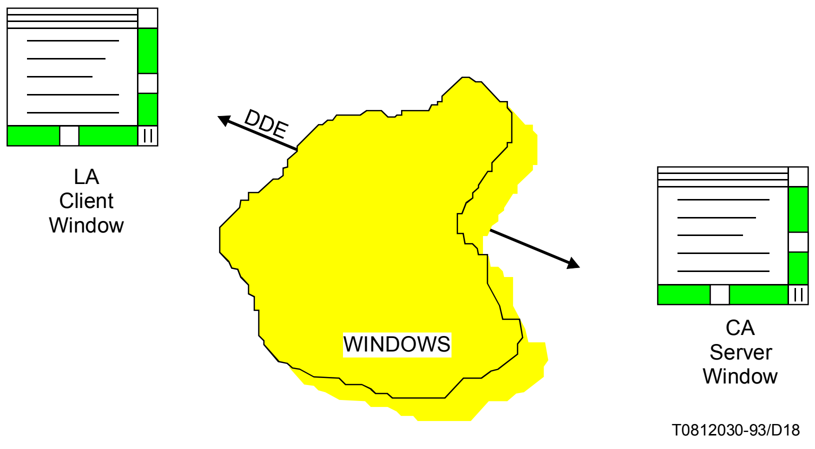


FIGURE E.8/T.611

Synopsis of the DDE protocol implementation

– WINDOWS Device Driver: The CA provider implements a “Device Driver” that allows any LA to access the CA. This mechanism is for further study.

APPENDIX I

(to Recommendation T.611)

Software interrupts for the MS-DOS environment

(Informative)

The CA provides an interrupt vector (which can be configured by software as stated in the ICE). This allows any LA to gain access to these entry points. This Recommendation recommends use of the multiplex interrupt 2Fh on DOS based systems. Interrupt 2Fh behaves like interrupt 21 h. Table I-1/T.611 describes the contents of the registers in order to activate the interrupt (See Figures I-2/T.611 and I-3/T.611.)

TABLE I-1/T.611

Register assignments for the interrupt mechanism

Parameter list	Register	Direction
Multiplex numbers[a)]	DS:AH	IN
Function Code (see Table I-2/T.611)	DS:AL	IN
TDD Buffer Handler (see Table I-3/T.611)	DS:BX	IN
Connection ID	DS:DX	IN/OUT
Status	DS:CH	OUT

a) This number shall be the same as the one declared in the ICE.

TABLE I-2/T.611

Function number

Function name	Function number
Login	00H
PutTDD	01H
PollTDD	02H
GetTDD	03H
SetAlarme	04H
Logout	05H

TABLE I-3/T.611

Contents of the TDD buffer

TDD buffer parameter list	Size in octets	Direction
TDD Size	2	IN/OUT
TDD Count	2	OUT
TDD Type	1	OUT
Login-name-size	2	IN
Login-name	Login-name-size	IN
Password-size	2	IN
Password	Password-size	IN
CallBackRoutine location	4	IN
TDD	TDD Size	IN

APPENDIX II

(to Recommendation T.611)

Examples of TDD exchanges

(Informative)

II.1 *A Sample Send Session using the "file" exchange method*

A LA – running on a MS-DOS (or Windows) based system – wants to send a document (which contains graphic information), say the document "c:\dtp\graphicl.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 a 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's assume the LA has already logged into the CA and the "c:\dtp\graphic l.tif" file is already prepared in TIFF format. Then the LA has to prepare the SEND TDD in it's memory. The LA uses the default coding (T50) for preparation of theTDD. The TDD may look like this:

```
I*APPLI/COM*1992*CCITT*
; Send a graphics document via facsimile group 3
; Note that the ";" leads in a comment
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:                             ; State of transmission (Response)
ERROR:                              ; Error occurred? (Response)
```

Note that fields into which a value may returned have to be already filled with blanks (20_{Hex}) or underline ($5F_{Hex}$) characters.

Once the TDD 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:

```
 I*APPLI/COM*1992*CCITT*
; Send a graphics document via facsimile group 3
; Note that the ";" leads in a comment
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\graphic 1.tif ; Full path to document
CONVERT:      TIFF                 ; Transfer format
COMID:        123456               ; Unique CA ID (Response)
STATUS:       \n                   ; State of transmission (Response)
ERROR:        0000/Success         ; Error occurred? (Response)
```

As one can see from the STATUS and ERROR field, the transmission was successful.

II.2 *A Sample Receive Session*

A 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;
- preparing 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;
- retrieving the Response-TDD (using the GetTDD function) to know the status of the reception;
- Logout of the CA or perfoming other functions.

The LA uses the default coding (T.50) for preparation of the receive TDD. The TDD may look like the following:

```
I*APPLI/COM*1992*CCITT*
; Receive a T.4 document via facsimile group 3
; Note that the ";" leads in a comment
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:                           ; Senders addressee, filled by CA
CONVERT:                           ; Transfer format
COMID:                             ; Unique CA ID (Response)
STATUS:                            ; State of transmission (Response)
ERROR:                             ; Error occurred? (Response)
```

Note that fields into which a value may returned have to be already filled with blanks (20_{Hex}) or underline ($5F_{Hex}$) characters.

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 in to its own memory and inspects the results. The Response-TDD could look like:

```
I*APPLI/COM*1992*CCITT*
; Receive a T.4 document via facsimile group 3
; Note that the ";" leads in a comment
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           ; Senders addressee, filled by CA
CONVERT:      T.4                    ; Transfer format
COMID:        000001                 ; Unique CA ID (Response)
STATUS:       +                      ; State of transmission (Response)
ERROR:        0000/Success           ; Error occurred? (Response)
```

As one can see from the STATUS and ERROR field, the reception was received by the CA.

II.3 *A Sample Trace Session*

A LA running on any operating environment could have information about the transitory or 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 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 learn the status of the reception;

– Logout of the CA or continue performing other functions.

The LA uses the default coding (T.50) for preparation of the receive TDD. The TDD may look like the following:

```
I*APPLI/COM*1992*CCITT*
; Retrieving a list of all CA-records in sending state
; Note that the ";" leads in a comment
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
Cvfax3:        TIFF                 ; Desired transfer format
COMID:                              ; Unique CA ID (Response)
ERROR:                              ; Error occurred? (Response)
```

Note that fields into which a value may returned have to be already filled with blanks (20_{Hex}) or underline ($5F_{Hex}$) characters.

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:

```
I*APPLI/COM*1992*CCITT*
; Retrieving a list of all CA-records in sending state
; Note that the ";" leads in a comment
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
Cvfax3:        TIFF                 ; Desired transfer format
COMID:         000002               ; Unique CA ID (Response)
ERROR:         0000/success         ; Error occurred? (Response)
```

As one can see from the ERROR field, the copy was received by the CA.

APPENDIX III

(to Recommendation T.611)

Example of ICE

(Informative)

An example of an ICE configuration (located on a MS-DOS based machine as a file) is given below:

```
I*APPLI/COM*1992*CCITT*ICE
; Note that there may be any type of configuration
; information which may be stored here. A new configuration
; information is always lead in by a "#" (number sign), as shown below after the empty line.

#                                                  ; Begin 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 native service supported
TX:          STD                                   ; Telex via Teletex service supp.
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                              ; Additional keywords
ADDKEYS:     SUBADDR                               ; Additional keywords
ADDKEYS:     SPEED                                 ; Additional keywords

#                                                  ; Begin of new Configuration
APPLICOM:    PRODUCT2 (c) bt 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:         BTM                                   ; 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                              ; Additional keywords
ADDKEYS:     SPEED                                 ; Additional keywords
ADDKEYS:     COMMENT                               ; Additional keywords
```

APPENDIX IV

(to Recommendation T.611)

Example of a CA Sample Implementation

(Informative)

This appendix shows how to utilize the T.611 interface on an existing popular private interface. The examples that follow describe how to send documents to multiple recipients, to receive a document and to read the log information generated by a Facsimile Group 3 card plugged into a personal computer.

The example involves the components that are described below (see Figure IV-1/T.611).

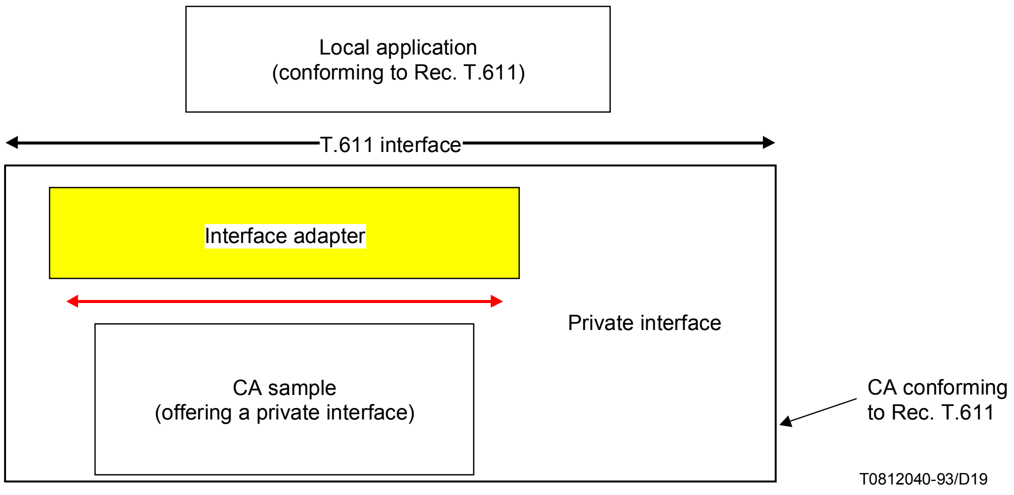


FIGURE IV-1/T.611

The LA sees an APPLI/COM conforming CA

The Local Application is a piece of software capable of generating T.611 function calls in order to send, receive and trace the CA. We also assume the LA handles the "primitive" based access method.

CA Sample is a preexisting Communications Application offering a private interface. It thus **does not** conform to this Recommendation. The idea is to implement additional software – "Interface Adapter" – so that the set consisting of the CA Sample and the Interface Adapter forms a conforming T.611 Communications Application.

The Interface Adaptor is software in charge of converting the LA requests into appropriate CA Sample function calls and returning appropriate responses to the LA. The following describes how the Interface Adapter can be implemented; first, we explain how the CA Sample works[12)].

The CA Sample is a Communications Application capable of sending and receiving facsimile group 3 documents. It also can return information about the progress of communications and provides the user with journals of events.

[12)] Details of the CA Sample are not described in this appendix. Only the specific information required to demonstrate operation is included here.

The CA Sample interface is based on two files:

– The "Control File" contains specific control information for a given Send or Receive request (e.g. who to call, when to call, etc.).

– The "Group File" contains information about the recipients of a given "Group Send" or "Group Polled Receive" when they concern more than one recipient.

The Control File is organized as displayed in Table IV-1/T.611.

TABLE IV-1/T.611

Structure of a control file as handled by CA sample

Parameter	Meaning	Set in ...
Request type	Send, Receive, Polled Send, Polled Receive, Group Send, Group Receive.	Request
LA-ID	Identifies the LA.	Request
Date and Time	Date and time scheduled for the Request, or date and time of the received event.	Request, response
Resolution	Fax 3 Resolution (not used if many recipients).	Request, response
Mode	Fax 3 Standard or "transparent" mode (not used if many recipients).	Request, response
Number of files	Number of files "to send" or "received".	Request, response
Offset of first file descriptor	Offset in this file of the first descriptor of files "to send" or "received".	Request, response
Phone number	Who to call (not used if many recipients).	Request
Group file name	File giving the list of the recipients (in case of many recipients only).	Request
Use cover page	Yes/no.	Request
Build a CIL	Yes/no.	Request, response
Cover page file	The path of a PCX file containing the cover page.	Request
Additional information	Non essential information (not described here).	
File descriptors	See below (as many file descriptors as the number of files to send/received).	Request, response

The "File descriptors" are appended at the end of the Control File. Each File Descriptor describes a single file that shall be transmitted in turn, in the same communication session. Table IV-2/T.611 summarizes the pieces of information that are required to describe a file to be sent or received.

TABLE IV-2/T.611

Contents of a file descriptor

Parameter	Meaning	Set in ...
File format	ASCII, PCX, DCX (in case of fax 3 transmission only).	Request, response
ASCII page format	Format of the ASCII page (in case of ASCII file only).	Request
File name	Complete path to the file.	Request, response
Page length	Only for fax 3 standard transmission.	Request, response
Non essential information	Additional pieces of information not described here.	Request, response

When addressing multiple recipients for a single request (Send of Group polled receive), the LA describes the recipients in a specific file named the “Group file”. The path of this file is stated in the “Group file name” parameter of the Control file. The format of the Group file consists of a sequence of records organized as shown in Table IV-3/T.611.

TABLE IV-3/T.611

Parameters contained in a record of the group file

Parameter	Meaning	Set in ...
Resolution	Resolution for fax 3 transmission.	Request, response
Mode	Fax 3 standard of transparent mode.	Request, response
Status	Transmission status.	Request, response
Dial number	Phone number call.	Request
Non essential information	Additional pieces of information not described here.	Request, response

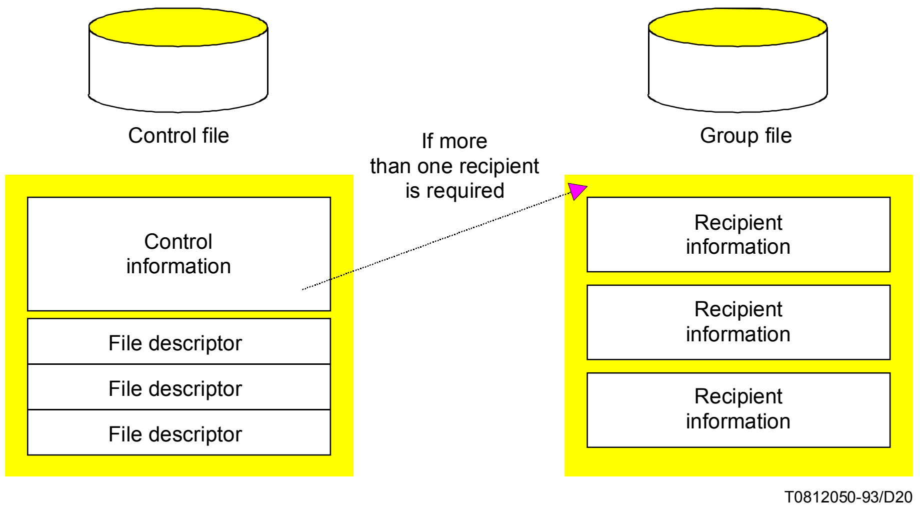
The CA Sample is capable of performing the following requests[13)].

TABLE IV-4/T.611

Requests supported by the CA sample

Send	The local computer transmits information to a remote device (fax machine or computer).
Receive	The local computer receives information from a remote device.
Polled send	The local computer waits for a remote device to call and then automatically sends information to it.
Polled receive	The local computer calls a remote device and receives information from it.
Group send	The local computer transmits the same information to multiple remote devices.
Group polled receive	The local computer calls multiple remote devices and receives information from them.

13) Functions in bold characters are used throughout this appendix.



Each request has one or more Control Files associated with it. A Control File contains information about an event that the CA Sample uses to schedule, execute, and report the status of the event. For example, the Control File for a Send request contains the phone number, date, and time information for the event.

To manage the events presented to it, the CA Sample places the Control Files associated with these events in queues. Three types of queues are supported: a Task Queue, a Receive Queue, and a Log Queue (see Table IV-5/T.611).

TABLE IV-5/T.611

Queues implemented by the CA sample

Task queue	Contains a list of pending events that the application running on the local computer has initiated. Pending events can be send, polled send, polled receive, group send, and group polled receive requests.
Receive queue	Contains a list of events that the local computer has received from a remote device.
Log queue	Contains a record of all the events that have been completed, aborted, or terminated with an error.

These event queues allow the CA Sample to schedule and monitor events. They also allow an application to check on the status of events.

To initiate an event, the LA must first create a Control File for the event. The application then submits the Control File for the event to the CA Sample and receives back a unique **event handle**. The LA uses the event handle to keep track of that particular event.

While an event is pending, the CA Sample keeps the Control File for the event in the Task Queue.

Once the event is completed, successfully or not, the CA Sample updates the status information in the event's Control File and moves the file to the Log Queue. The LA can then use the CA Sample interface functions to examine the Control Files in the log queue and determine the status of completed events.

For receive events, the CA Sample receives data from the remote device and stores it in a disk file. Once the receive event is completed, the CA Sample creates a Control File for the event and places copies of it in both the Receive Queue and the Log Queue. The application can then use CA Sample functions to examine the Control Files in the Receive and Log Queues and to open files of received data.

Group events are events in which the same activity is carried out between the local computer and several remote devices. For example, with a group send event, the local computer can send the same file (or files) to a group of several remote devices. Or, with a group polled receive, the local computer polls a group of remote devices to receive information from them.

When the Control File for a group event is submitted to the Task Queue, the CA Sample assigns an event handle to the group event. It also examines the Group File and assigns an event handle to each group member. This latter event handle is used to identify the instance of the group event (called a sub-event) that is processed for that group member.

Each time a sub-event is completed, the CA Sample creates a Control File in the Log Queue for that sub-event.

When all the sub-events associated with a parent event have been processed, the CA Sample moves the Control File for the parent event to the Log Queue. It also moves the Group File to the Log Queue.

The exchange mechanism implemented by the CA Sample is based on software interrupts. CA Sample uses interrupt 2Fh, with multiplex number 0CBh.

The CPU registers are filled up the following way, before any function call to the CA Sample. (See Table IV-6/T.611.)

TABLE IV-6/T.611

How to fill the registers before calling CA Sample functions

Register	Contains
AH = 0CBh	Multiplex number.
AL = function code	One of the function codes corresponding to the functions offered by CA Sample.
Others	... depend on the function invoked.

"Function" here means a utility routine. Other registers not listed above have a meaning depending on the utility routine invoked.

IV.1 *Examples of Interface Adapter Operation*

IV.1.1 *Transmission*

Let us suppose the LA wishes to send two documents ("C:\COVER.TIF" in the TIFF format and "C:\CONTENT.TXT" in the ASCII format) to two recipients in a T.611 fashion. The LA will thus build a SEND Request and convey it to the Interface Adapter.

Table IV-7/T.611 displays the contents of the SEND request built by the LA.

TABLE IV-7/T.611

Example of a sample SEND TDD

Function:	SendAck
LA_ID:	MyLA
REQ_ID:	1
Service:	Fx3
Status:	
Error:	
ComID	
CIL:	
SendTime:	91-10-01-15:06
AddrList:	C:\ADDRLIST\ADDRLIST.001
FileList:	C:\FILELIST\FILELIST.001

The AddrList and FileList keywords state here that the list of files and the list of recipients are found in files specified in their respective entry. The file "ADDRLIST.001" contains the following information:

```
!0##45782762;!0##45782222
```

This means that the LA wishes to call the phone numbers stated above (45782762 and 45782222 respectively). The "0##" prefix are often used as PABX prefixes.

The file "FILELIST.001" contains the following:

```
{[(COVER.TIF,TIFF) (CONTENT.TXT,ASCII)STD]}
```

When the Interface Adapter receives the SEND Request, it will in turn transform it into a CA Sample Interface function call by first generating a Control File as displayed in Table IV-8/T.611.

TABLE IV-8/T.611

Dump of the control file generated by the interface adapter

Parameter value	Means
04 "MyLA" "9110011506" 0000 00 02 xxxxx "c:\address.lst" "no" "yes"	Group send LA-ID Date and time to process the request No resolution stated because a group file is used No transmission mode stated because a group file is used 2 files to transmit Offset of the first file descriptor Path name of the group file Do not use a cover page Build a CIL
""	Empty: additional information not specified
First file descriptor	
"PCX" 0 ""C:\COVER.PCX" 00	File format[a)] Not applicable (file is not ASCII) File to send Page length is 11 inches (the default)
Second file descriptor	
"ASCII" 0 "C:\CONTENT.TXT" 00	File format Page format is 80 columns over 66 lines File to send Not applicable (file is ASCII)

a) The PCX format is the only graphics format accepted by the CA Sample. It is thus necessary to convert the file “CONVERT:TIF” into a PCX file named “CONVERT.PCX”. The handling of this conversion is the responsibility of the Interface Adaptor.

The file "C:\ADDRLIST\ADRLIST.001" is formatted as shown in Table IV-9/T.611.

TABLE IV-9/T.611

Contents of the C:\ADDRLIST\ADDRLIST.001 file

First recipient descriptor	
0 0 0 "0##45782762"	196 dpi resolution Standard fax 3 transmission Status Phone number to call
Second recipient descriptor	
0 0 0 0##45782762"	196 dpi resolution Standard fax 3 transmission Status Phone number to call

This example shows the operation of the Interface Adaptor to convert the SEND Request as issued by the LA into the Control file and the Group file as described above. This step accomplished, the Interface Adaptor may invoke the CA Sample internal routines to submit the Control file for further processing. It may do so by issuing a software interrupt 2F, indicating a multiplex number of 0CBh, with a function code 01 (which is designed to submit a task to the CA Sample).

IV.1.2 *Reporting*

When the LA wishes to check whether any inbound fax has arrived, it issues a TRACE:copy Request on the "reception" state. This lists all incoming facsimile documents that were handled by the CA Sample. The Request is displayed in Table IV-10/T.611.

TABLE IV-10/T.611

Contents of the TDD TRACE Request issued by the LA

Function:	Copy
LA_ID:	MyLA
REQ_ID:	2
State:	Reception
Error:	
Target:	C:\LOGS\RECEIVE.LOG
Com-ID:	

The Interface Adaptor will transform this TDD into CA Sample private function calls and will generate the RECEIVE.LOG file when returning.

The Interface Adaptor issues specific function calls to the CA Sample to read the Receive Queue and build up a series of CA Records into the file "C:\LOGS\RECEIVE.LOG". The Interface Adaptor is in charge of converting the entries found in the RECEIVE Queue into CA Records.

The Interface Adaptor would thus perform the steps below, shown in pseudo-code programming style:

```
Find first entry in RECEIVE QUEUE;
if (RECEIVE QUEUE is empty) then quit;
else {
        Open associated Control file;
        Read information of Control file;
        Convert into a CA Record;
        Close Control file;
        Open APPLI/COM Log file;
        Write CA record into APPLI/COM Log file;
        }
While (RECEIVE QUEUE is not empty)
        {
        Get next entry in RECEIVE QUEUE;
        Open associated Control file;
        Read information of control file;
        Convert into a CA Record;
        Close Control file;
        Write CA record into APPLI/COM Log file;
        }
Close APPLI/COM Log file;
Return APPLI/COM Log file handle to LA;
end.
```

The conversion of a Control file into a CA Record is a process beyond the scope of this appendix.

IV.1.3 *Reception*

LA now can consult the RECEIVE.LOG file and obtain a ComID which is added by the Interface Adaptor to all reformatted records. The LA now wishes to retrieve one of the received faxes, ComID set to 22. It thus builds a RECEIVE Request that will be managed by the Interface Adaptor (see Table IV-11/T.611).

TABLE IV-11/T.611

Contents of the RECEIVE Request constructed by the LA

Function:	Receive
LA_ID:	MyLA
REQ_ID:	3
Service:	Fx3
Address:	
Filename:	C:\RECEIVE\RECEIVE.TIF[a)]
Convert:	
Status:	
Error:	
CIL:	
ComID:	22
RcvTime:	

[a)] The LA wishes to receive a document in the TIFF format (because TIFF is the default format) and store it under the path name "C:\RECEIVE\RECEIVE.TIF".

The CA Sample spontaneously builds a Control file for every receive event – independently from the LA activity. The Interface Adaptor needs thus to map the ComID identifier to the appropriate Control file. This can be achieved by means of an internal table.

Note – The ComID identifier was built by the Interface Adaptor when the LA requested to obtain a list of the incoming faxes (see previous section).

For instance, let us suppose the Control file associated to ComID number 22 is as shown in Table IV-12/T.611.

TABLE IV-12/T.611

Control file generated by CA Sample on a RECEIVE event

Parameter	Means
void	Function
void	LA_ID
"9110011610"	Date and time the document was received
196 dpi	Resolution
STD	Standard fax 3 transmission mode
1	One file was received
xxxxx	Offset of its file descriptor
void	Phone number to call
void	Group file name
void	Use cover page
void	Build CIL
void	Cover page file path
void	Other non essential information items
First file descriptor	
PCX	File format
void	ASCII Page format
"C:\RECEIVE\RECEIVE.PCX"	Path of the received file
0	Page length (11 inches)
void	Other non essential information items

The Interface Adaptor then builds a RECEIVE Response as shown in Table IV-13/T.611.

TABLE IV-13/T.611

RECEIVE Response returned to the LA

Function:	Receive
LA_ID:	MyLA
REQ_ID:	3
Service:	Fx3
Address:	33147837634
Filename:	C:\RECEIVE\RECEIVE.TIF[a)]
Convert:	TIFF
Status:	0
Error:	0000/Success
CIL:	
ComID:	22
Rcv Time:	91-10-01-16:10

[a)] The interface adaptor converts the received “RECEIVE.PCX” file (PCX format) into the RECEIVE.TIF (TIFF format) as requested by the LA.

IV.2 *Example of Interface Configuration Environment (ICE) file for the Interface Adaptor*

This example ICE file could be used on the system using the Interface Adaptor as described in the previous sections of this appendix. (See Table IV-14/T.611.)

TABLE IV-14/T.611

Example ICE file for the interface adaptor

```
l*APPLI/COM*1992*CCITT*ICE
#
APPLICOM:                         Product ABC (c) by Company XYZ
DRF:                              Yes
EM:                               Primitive
ENVIRON:                          MS-DOS
INT:                              2F:D0
ENVIRON:                          WINDOWS 3.0
DDE:                              Yes
APPLICATION:                      ABC.EXE
SUBJECT:                          CA
ITEM:                             FX3
CODING:                           I
INT:                              2F-D1
COUNTRY:                          3D; France
ALARM:                            Yes
FC:                               B
```

Printed in Switzerland

Geneva, 1993