International Telecommunication Union

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

T.127
(08/2007)

SERIES T: TERMINALS FOR TELEMATIC SERVICES

Data protocols for multimedia conferencing

Multipoint binary file transfer protocol

ITU-T Recommendation T.127

ITU-T T-SERIES RECOMMENDATIONS

TERMINALS FOR TELEMATIC SERVICES

Facsimile – Framework	T.0–T.19
Still-image compression – Test charts	T.20–T.29
Facsimile – Group 3 protocols	T.30–T.39
Colour representation	T.40–T.49
Character coding	T.50–T.59
Facsimile – Group 4 protocols	T.60–T.69
Telematic services – Framework	T.70–T.79
Still-image compression – JPEG-1, Bi-level and JBIG	T.80–T.89
Telematic services – ISDN Terminals and protocols	T.90–T.99
Videotext – Framework	T.100–T.109
Data protocols for multimedia conferencing	**T.120–T.149**
Telewriting	T.150–T.159
Multimedia and hypermedia framework	T.170–T.189
Cooperative document handling	T.190–T.199
Telematic services – Interworking	T.300–T.399
Open document architecture	T.400–T.429
Document transfer and manipulation	T.430–T.449
Document application profile	T.500–T.509
Communication application profile	T.510–T.559
Telematic services – Equipment characteristics	T.560–T.649
Still-image compression – JPEG 2000	T.800–T.849
Still-image compression – JPEG-1 extensions	T.850–T.899

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

ITU-T Recommendation T.127

Multipoint binary file transfer protocol

Summary

ITU-T Recommendation T.127 defines a protocol to support the interchange of binary file data within an interactive conferencing or group working environment where the T.120 suite of standards is in use. It provides mechanisms to support simultaneous distribution of multiple files, selective distribution of files to a subset of participants and retrieval of files from remote sites. Provision is also made for remote directory access.

This revised version of T.127 introduces a number of clarifications to the previous version.

Source

ITU-T Recommendation T.127 was approved on 29 August 2007 by ITU-T Study Group 16 (2005-2008) under the ITU-T Recommendation A.8 procedure.

FOREWORD

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

The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1.

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

NOTE

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

Compliance with this Recommendation is voluntary. However, the Recommendation may contain certain mandatory provisions (to ensure e.g., interoperability or applicability) and compliance with the Recommendation is achieved when all of these mandatory provisions are met. The words "shall" or some other obligatory language such as "must" and the negative equivalents are used to express requirements. The use of such words does not suggest that compliance with the Recommendation is required of any party.

INTELLECTUAL PROPERTY RIGHTS

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

As of the date of approval of this Recommendation, ITU had not received notice of intellectual property, protected by patents, which may be required to implement this Recommendation. However, implementers are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database at http://www.itu.int/ITU-T/ipr/.

CONTENTS

Page

Page

ITU-T Recommendation T.127

Multipoint binary file transfer protocol

1 Scope

This Recommendation defines a protocol to support the interchange of binary files within an interactive conferencing or group working environment where the T.120 suite of standards is in use. It provides mechanisms which facilitate distribution and retrieval of one or more files simultaneously using the primitives provided by [ITU-T T.122] (Multipoint Communications Service). This Recommendation is designed to offer a versatile, light weight protocol which provides the core functionality to allow interworking between applications requiring a basic file transfer capability and also has the flexibility to meet the demands of more sophisticated applications. See Figure 1.

2 References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

[ITU-T H.221] ITU-T Recommendation H.221 (2004), *Frame structure for a 64 to 1920 kbit/s channel in audiovisual teleservices*.

[ITU-T T.35] ITU-T Recommendation T.35 (2000), *Procedure for the allocation of ITU-T defined codes for non-standard facilities*.

[ITU-T T.120] ITU-T Recommendation T.120 (2007), *Data Protocols for multimedia conferencing*.

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

[ITU-T T.123] ITU-T Recommendation T.123 (2007), *Network-specific data protocol stacks for multimedia conferencing*.

[ITU-T T.124] ITU-T Recommendation T.124 (2007), *Generic Conference Control*.

[ITU-T T.125] ITU-T Recommendation T.125 (1998), *Multipoint communication service protocol specification*.

[ITU-T T.434] ITU-T Recommendation T.434 (1999), *Binary file transfer format for the telematic services*.

[ITU-T V.42 *bis*] ITU-T Recommendation V.42 *bis* (1990), *Data compression procedures for data circuit-terminating equipment (DCE) using error correction procedures*.

[ITU-T X.680] ITU-T Recommendation X.680 (2002), *Information technology – Abstract Syntax Notation One (ASN.1) – Specification of basic notation*.

[ITU-T X.691] ITU-T Recommendation X.691 (2002), *Information technology – ASN.1 encoding rules: Specification of Packed Encoding Rules (PER)*.

[ISO/IEC 13239] ISO/IEC 13239:2002, *Information technology – Telecommunications and information exchange between systems – High-Level Data Link Control (HDLC) procedures*.

[ISO 8571-2] ISO 8571-2:1988, *Information processing systems – Open Systems Interconnection – File Transfer, Access and Management (FTAM) – Part 2: Virtual filestore definitions*.

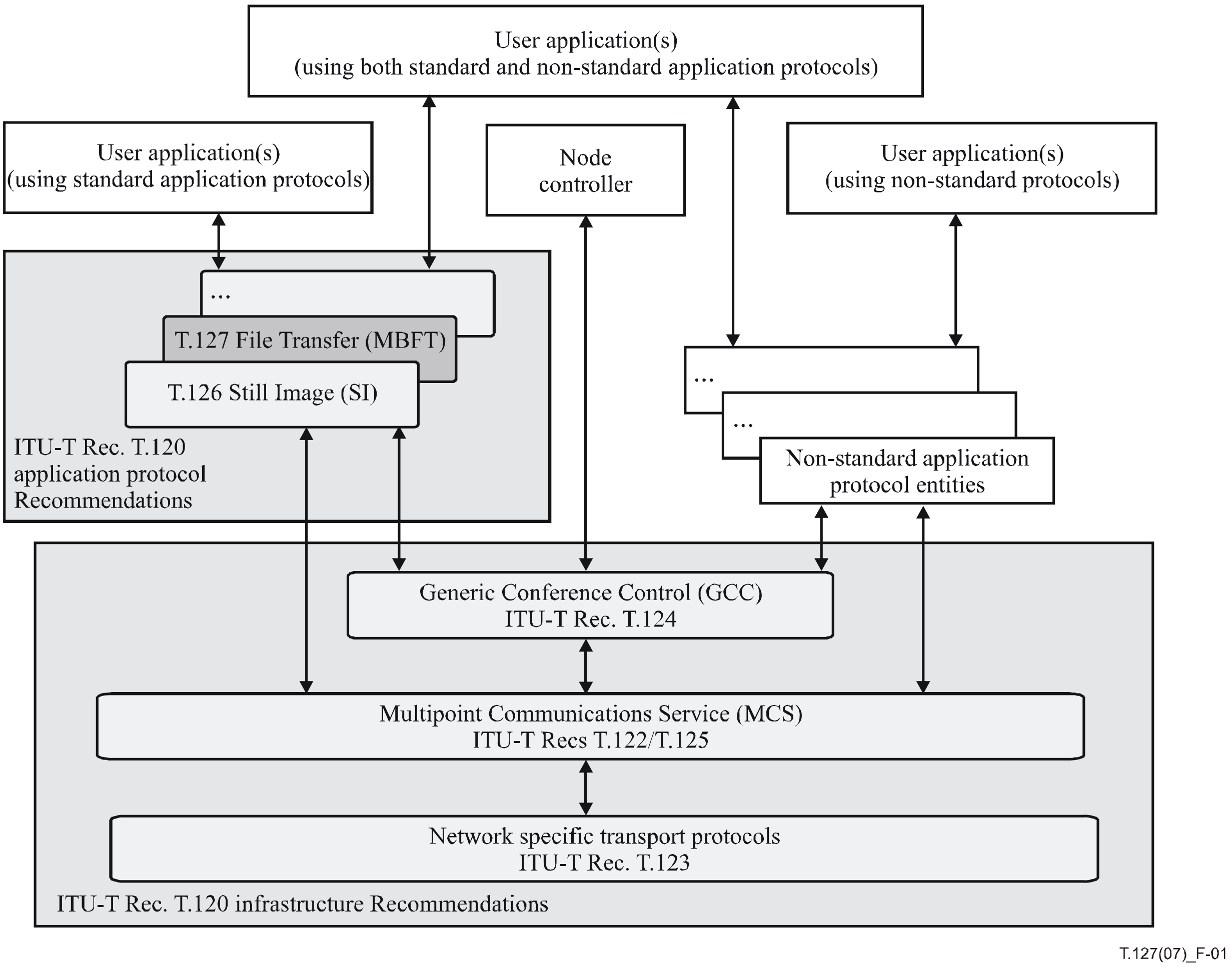

User application(s)
(using both standard and non-standard application protocols)
User application(s)
(using standard application protocols)
Node
controller
User application(s)
(using non-standard protocols)
...
T.127 File Transfer (MBFT)
T.126 Still Image (SI)
ITU-T Rec. T.120
application protocol
Recommendations
...
...
Non-standard application
protocol entities
Generic Conference Control (GCC)
ITU-T Rec. T.124
Multipoint Communications Service (MCS)
ITU-T Recs T.122/T.125
Network specific transport protocols
ITU-T Rec. T.123
ITU-T Rec. T.120 infrastructure Recommendations
T.127(07)_F-01

Figure 1 – Scope of T.127

3 Definitions

This Recommendation defines the following terms:

3.1 acknowledged data channel: An MCS channel on which files are distributed. Participants have the option of rejecting files offered on an acknowledged data channel. An acknowledged data channel may be *exclusive* (i.e., only the channel creator may send files on it), or *shared* (any participant may send files on it).

3.2 broadcast data channel: An MCS channel on which files are distributed. Participants must receive all files distributed on the channel, discarding the data locally if it is not required.

3.3 control channel: An MCS channel used for the management of file transactions.

3.4 file attributes: The name and other identifiable properties of a file.

3.5 FILE-REQUEST: The token used to ensure that there is at most one outstanding file request on the session control channel MBFT-CONTROL.

3.6 FILE-REQUEST(p): The token used to ensure that there is at most one outstanding file request on the sub-session control channel MBFT-CONTROL(p).

3.7 FILE-TRANSMIT: The token used to ensure that there is at most one file transfer in progress on the session broadcast data channel MBFT-DATA.

3.8 FILE-TRANSMIT(p): The token used to ensure that there is at most one file transfer in progress on the sub-session broadcast data channel MBFT-DATA(p).

3.9 FILE-TRANSMIT(n): The token used to ensure that there is at most one file transfer in progress on acknowledged data channel MBFT-DATA(n).

3.10 MBFT-CONTROL: The session control channel.

3.11 MBFT-CONTROL(p): A sub-session control channel, whose MCS Channel ID is p.

3.12 MBFT-DATA: The session broadcast data channel.

3.13 MBFT-DATA(p): A sub-session broadcast data channel, whose MCS Channel ID is p.

3.14 MBFT-DATA(n): An acknowledged data channel, whose MCS Channel ID is n.

3.15 non-standard capability: A capability that is outside the scope of this Recommendation. Non-standard capabilities must be negotiated before use.

3.16 session: A set of peer Application Protocol Entities.

3.17 standard capability: A capability that is defined within the scope of this Recommendation, but is not required for all MBFT implementations. Standard capabilities must be negotiated before use.

3.18 sub-session: A sub-group of peer Application Protocol Entities within a session.

4 Abbreviations

This Recommendation uses the following abbreviations:

ARM	Application Resource Manager
APE	Application Protocol Entity
ASE	Application Service Element
GCC	Generic Conference Control
GCCSAP	GCC Service Access Point
MBFT	Multipoint Binary File Transfer
MCS	Multipoint Communication Service
MCSAP	MCS Service Access Point
PDU	Protocol Data Unit

5 Introduction to multipoint file transfer

In order to support group activities such as meetings, conferences, etc. involving physically separated participants, there is a requirement to join together two or more locations. The term multipoint communication simply describes the interconnection of multiple terminals. Multipoint Binary File Transfer (MBFT) enables files to be exchanged interactively between participants within a multipoint environment through use of the underlying network independent Multipoint Communication Service (MCS).

Specifically, this Recommendation provides flexible and efficient mechanisms to support:

- Simultaneous distribution of multiple files.
- Broadcasting of files to all participants within a conference.
- Selective distribution of files to a subset of participants.
- Retrieval of files from remote sites.
- Partial retransmission of files following an interruption.
- Remote directory access.

6 Multipoint transfer of data – An overview

T.127 uses a control/data channel architecture to facilitate simultaneous transfer of one or more binary files. It enables files to be broadcast to all participants within a conference, or to be directed selectively to a subset of sites as a private file transfer. No restrictions are placed on the type of data being transmitted.

Two types of channels are used within T.127; *control channels* and *data channels*. Control channels are used for managing all aspects of the file transfer (offering files, requesting files), whereas data channels are used exclusively for the transfer of file data. Only one file can be transmitted on each data channel at a time, but additional data channels can be used to allow distribution of multiple files simultaneously. The number of data channels in use at any given time depends on the number of concurrent file transfers in progress.

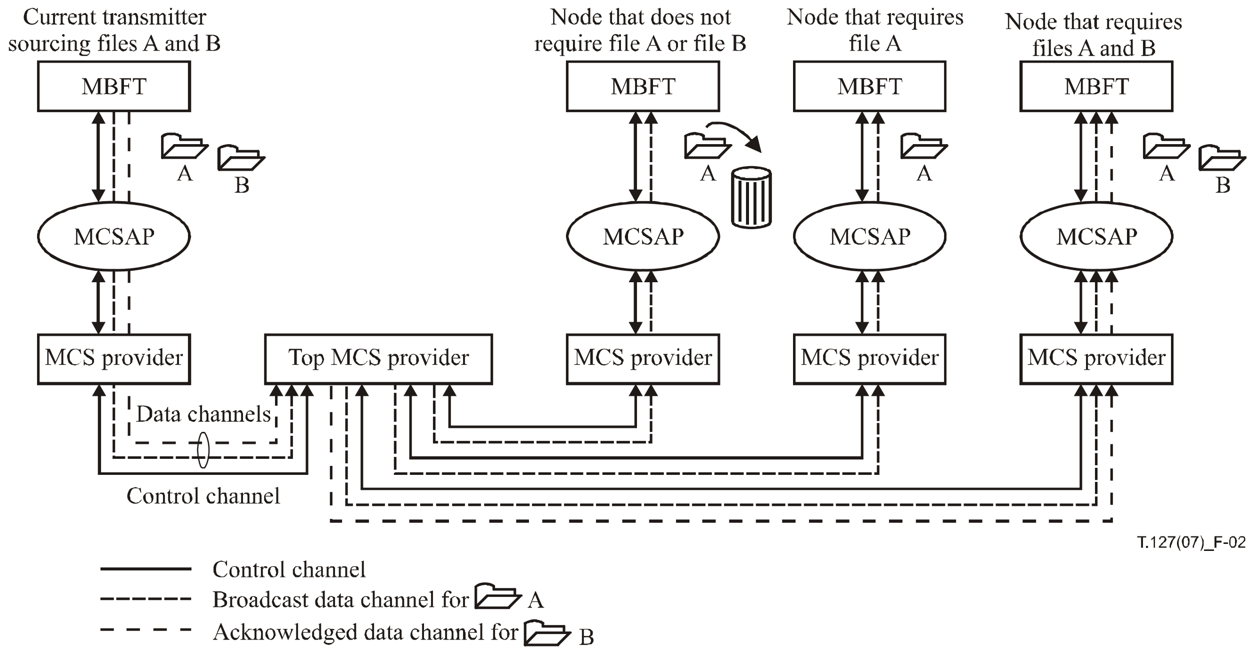
A group of file transfer applications communicating with each other are said to be participating in the same file transfer *session*. Each file transfer *session* requires a single control channel and one or more data channels for distribution of files to all participating applications.

T.127 supports two types of data channel: *broadcast* and *acknowledged*. If a transmitter wishes to mandate that all nodes receive a file it is offering, then it should use the broadcast data channel. All nodes must stay joined to the broadcast data channel for the duration of the file transfer session and are obliged to receive all files distributed on it; if a file is not required, receivers should discard it. If a transmitter wishes to give other nodes the option of rejecting a file, it should offer the file on an acknowledged data channel. In this case, each node must inform the transmitter of whether it requires the data or not, and only those which want the file join the data channel. Multiple concurrent file transfers are supported by use of acknowledged data channels.

Acknowledged data channels should be used if a transmitter considers that one or more of the parameters in the file header are essential to the operation of the application. For example, an application may require a pathname to be preserved by receivers for future reference. Key parameters are identified when offering the file for distribution; nodes which are unable to support all such parameters must reject the file.

The creator of an acknowledged data channel may be designate it to be *exclusive* (i.e., only the creator may send files on the channel) or *shared* (i.e., any participant may send files on it).

File transactions on the broadcast channel do not require any handshaking between transmitter and receivers as nodes are obliged to receive all files distributed on this channel. This minimizes latency at the start of file transfers for transactions on the broadcast data channel. Transactions on an acknowledged data channel incur some latency at the start of a file transfer, but may have a better overall performance by avoiding unnecessary distribution of data to sites that do not require it, particularly if such sites are on low bandwidth links. The choice of channel is at the transmitter's discretion and may depend on application, file size, network configuration and number of conference participants. See Figure 2.


Current transmitter sourcing files A and B
Node that does not require file A or file B
Node that requires file A
Node that requires files A and B
MBFT
MBFT
MBFT
MBFT
A
B
A
A
A
B
MCSAP
MCSAP
MCSAP
MCSAP
MCS provider
Top MCS provider
MCS provider
MCS provider
MCS provider
Data channels
Control channel
T.127(07)_F-02
Control channel
Broadcast data channel for A
Acknowledged data channel for B

NOTE 1 – All nodes attach to the control channel and broadcast data channel.
NOTE 2 – Nodes must receive files on the broadcast data channel, whether they require the data or not.
NOTE 3 – Nodes only join an acknowledged data channel if they wish to receive the file currently being offered on it.

Figure 2 – T.127 conference model

Selective distribution of files to a subset of nodes within a conference can be achieved by creating a private file transfer session, or by establishing a private file transfer *sub-session* within the existing session.

A sub-session follows the same model as a session by having a single control channel and one or more data channels. See Figure 3. However, it differs from a session in that it is not required to have a broadcast data channel. A sub-session has the same capability set as its parent session and its participants are selected from the participants of the parent session. Sub-sessions have no status within GCC and do not appear in the GCC-Application-Roster. A sub-session does not have a distinct Session ID, but instead operates with the Session ID of the main MBFT session. By avoiding the delay incurred by the enrollment process, sub-sessions allow private interactive file exchanges to be initiated in an expedient manner, at the same time conserving GCC and MCS resources.

Provision is made for sites to request a file from other nodes to allow information retrieval from databases, bulletin boards, etc. Sufficient information must be provided in the request to allow the sourcing site to uniquely identify the file required.

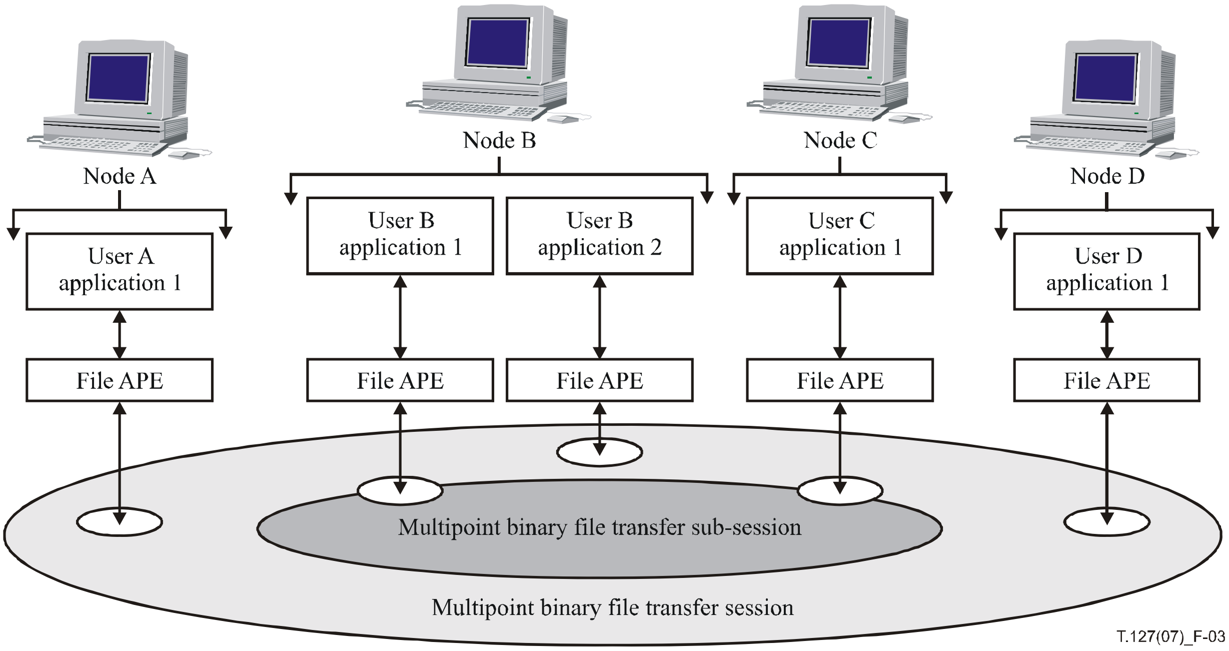


Figure 3 – Relationship between a sub-session and session

6.1 T.127 system model

An MBFT Session is characterized by the following attributes as illustrated in Figure 4.

- A single control channel.
- A single broadcast data channel.
- Zero or more acknowledged data channels.
- Zero or more private sub-sessions (to allow file exchange between a selected subset of conference participants).
- A Session ID.

Each sub-session has the following attributes:

- A single private control channel.
- Zero or one private broadcast data channels.
- Zero or more private acknowledged data channels.
- No individual Session ID (operates with the Session ID of the main MBFT session).

Each control channel has a FILE-REQUEST token associated with it (unless the channel creator requires an exclusive right to request files from other sites).

Each data channel has a FILE-TRANSMIT token associated with it (unless the channel creator requires an exclusive right to transmit files on that channel).

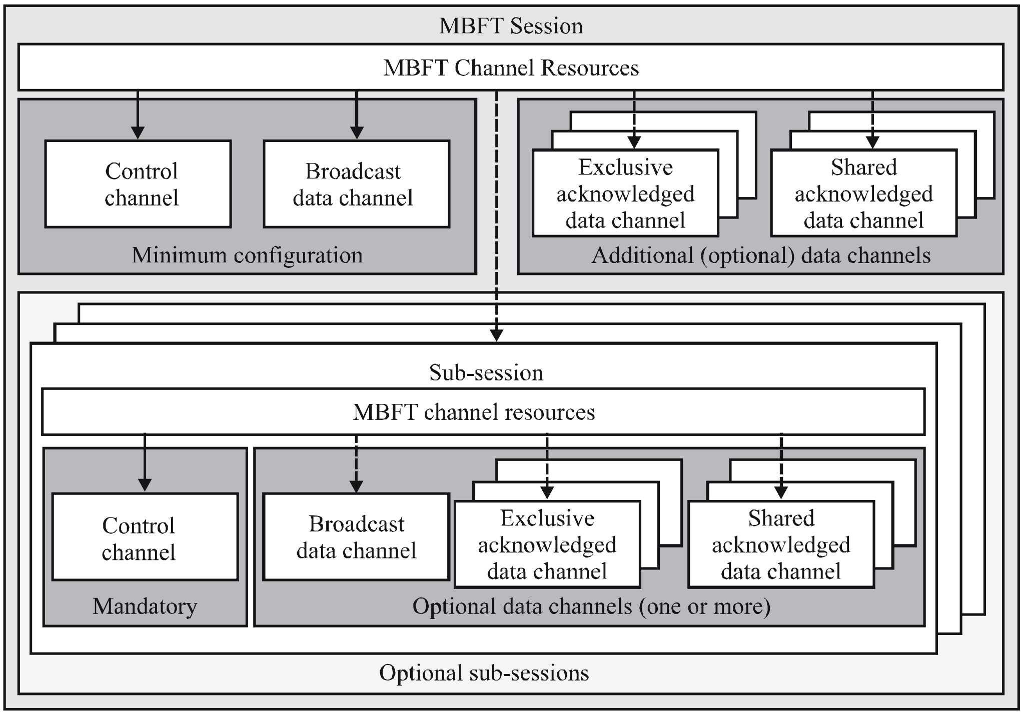


Figure 4 – T.127 channel model

6.2 Compression

Compression may be applied to files, subject to successful negotiation; by default files are uncompressed. Proprietary techniques may be identified using the formats specified in [ITU-T T.124], e.g., T.35 country code, nationally assigned code, manufacturer's code, non-standard capability code. Object identifiers may be acquired instead. *De facto* standard compression formats may also be identified via this mechanism. Note that compression applies only to the file data payload and not to the file header.

6.3 Priority

T.127 can be used as a background task performing bulk data transfer as well as a foreground task for immediate distribution of files. The mode to be used is selected by the transmitting site; medium priority should be used for fast data transfer and low for bulk data transfer. The priority must remain the same throughout the transmission of a file but can differ between successive transactions.

Management of file transactions on the control channel shall use high priority.

6.4 File preshipping

In order to minimize file transfer traffic during an interactive conference, files may be preshipped by convening a conference specifically to distribute conference material in advance. This can be an automated process and if receiving sites can identify those files which they wish to receive, file traffic can be kept to a minimum.

7 Baseline MBFT application

Applications wishing to support the file transfer protocol must be able to join the control channel and send or receive on the broadcast data channel. Table 1 identifies which PDUs must be supported.

Table 1 – Support of MBFT PDUs

MBFT PDU	File receive only APE		File transmit only APE		File transmit & receive APE	
	Send PDU	Receive PDU	Send PDU	Receive PDU	Send PDU	Receive PDU
File-Offer	–	M	M	M	M	M
File-Accept	M	–	–	M	M	M
File-Reject	M	–	M	M	M	M
File-Request	O	M	–	M	O	M
File-Deny	M	O	M	–	M	O
File-Error	O	–	–	O	O	O
File-Abort	O	–	O	M	O	M
File-Start	–	M	M	–	M	M
File-Data	–	M	M	–	M	M
Directory-Request	O	M	O	M	O	M
Directory-Response	M	O	M	O	M	O
MBFT-NonStandard	O	O	O	O	O	O
MBFT-Privilege-Request	O	O	M	O	M	O
MBFT-Privilege-Assign	O	M	O	M	O	M
Private-Channel-Join-Invite	O	M	O	M	O	M
Private-Channel-Join-Response	M	O	M	O	M	O
M Mandatory O Optional – Not Required						

8 Description of operation

A file transfer user application relies on the services of a File Transfer Application Protocol Entity (File APE) to communicate with peer applications at other nodes. The File APE has two components as shown in Figure 5: a File Transfer Application Resource Manager (File ARM) and a File Transfer Application Service Element (File ASE). The ARM provides generic functionality, common to all standardized application protocols, whilst the ASE provides functionality specific to this application protocol to enable interworking of file transfer applications. Note that this is a conceptual model and does not impose any constraints on the structure of actual implementations.

Each component is described in more detail below:

8.1 File transfer user application

This is the part of the file transfer application addressing those aspects which have no direct effect on interworking (e.g., user interface) and which may thus be product and platform specific. The influence of the user application is thus local to the site at which it is resident. As such it is outside the scope of this Recommendation. A user application relies on the services of a File Transfer Application Protocol Entity (APE) to communicate with peer applications at other nodes. It does not communicate with MCS or GCC; this is done by the File APE. The user application initiates a file transfer session via its File APE, specifying the application capabilities and session mode. Once the session has been established, all MBFT specific transactions are performed by the File APE on behalf of the user application.

8.2 File Transfer Application Resource Manager

The File Transfer Application Resource Manager (File ARM) is responsible for managing GCC and MCS resources on behalf of the File ASE. It provides the following services:

- Responding to indications from GCC (e.g., permission to enroll, invoke).
- Enrolling the File APE with GCC.
- Attaching to an MCS domain to obtain an MCS User ID for the File APE.
- Joining static channels.
- Identifying and joining multicast channels using the GCC Registry and MCS.
- Convening private channels and admitting peer File APEs to such channels.
- Joining any private channels to which the File APE has been admitted.
- Identifying and obtaining tokens from the GCC Registry.
- Deleting entries from the registry associated with any channel it may have created.
- Invoking peer File APEs at other nodes.
- Processing Application Roster reports to determine the negotiated Application Capability list and identity of peer File APEs.

8.3 File Transfer Application Service Element

The File Transfer Application Service Element (File ASE) provides file transfer functionality to the user application with resources obtained by the File ARM. Its operation is independent of the type (i.e., static or dynamic) and identity of tokens and channels passed to it. The user application specifies whether a broadcast or acknowledged data channel is required. For private transfers to a subset of the conference, a list of MBFT User IDs is also required.

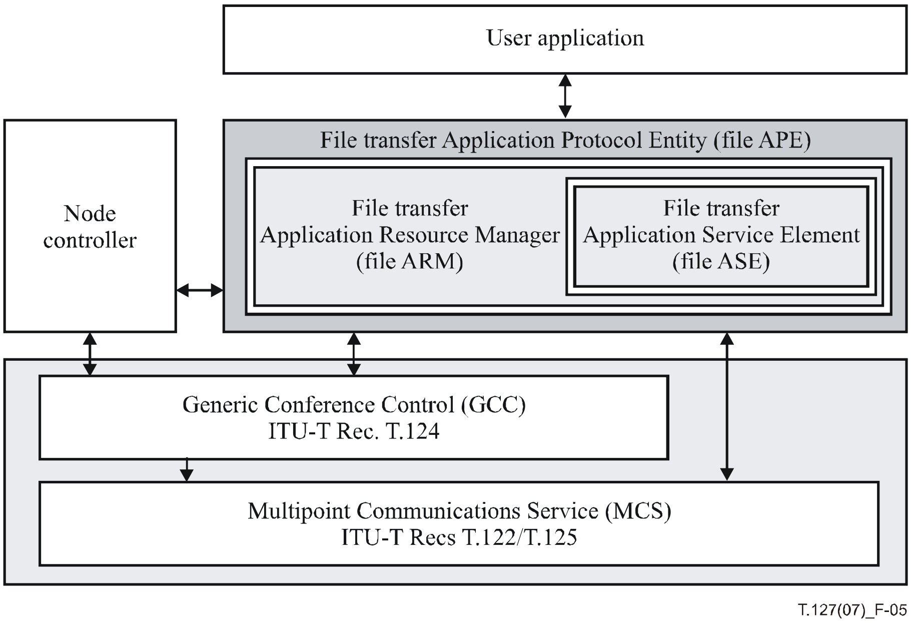

User application
File transfer Application Protocol Entity (file APE)
Node controller
File transfer
Application Resource Manager
(file ARM)
File transfer
Application Service Element
(file ASE)
Generic Conference Control (GCC)
ITU-T Rec. T.124
Multipoint Communications Service (MCS)
ITU-T Recs T.122/T.125
T.127(07)_F-05

Figure 5 – T.127 application model

The File ASE provides the following services:

- Sending and receiving MBFT PDUs.
- Grabbing and releasing tokens and determining token status using MCS.
- Responding to GCC-Conductor-Assign and Release indications.
- Issuing GCC-Conductor-Permission-Ask requests through the Node Controller.
- Responding to GCC-Conductor-Permission-Grant indications.

8.4 MBFT resources

A binary file transfer session uses *control channels* for management of file transfers and *data channels* for file distribution. Each control channel has one or more data channels associated with it; each data channel supports one file transfer at a time.

Every MBFT session has a session control channel (assigned the mnemonic MBFT-CONTROL) and a broadcast data channel (assigned the mnemonic MBFT-DATA), which must be joined by all applications participating in that session. The control channel is used to manage all file transfers on the broadcast data channel. All nodes are obliged to receive files transmitted on the broadcast data channel and discard them if the data is not required. This can lead to an unnecessary degradation in conference performance if any nodes which do not require the data are on low bandwidth links.

Use of *acknowledged data channels* (assigned the mnemonic MBFT-DATA(n), where n is the MCS Channel ID of the data channel) allows the simultaneous distribution of more than one file. Management of file transfers on these channels is done via the session control channel, but in this case nodes have the option of rejecting files offered to them. This ensures that files are only distributed to those nodes which require them, but at the expense of introducing some latency for each file transfer.

Selective distribution of files to a subgroup of participants within an existing session may be achieved by opening a *sub-session*. This consists of a private sub-session *control channel* (assigned the mnemonic MBFT-CONTROL(p), where p is the MCS Channel ID of the control channel). This is used to manage file transactions on zero or one *private broadcast data channels* (assigned the mnemonic MBFT-DATA(p), where p is the MCS Channel ID of the data channel) and zero or more *private acknowledged data channels* (assigned the mnemonic MBFT-DATA(n), where n is the MCS Channel ID of the data channel). A separate private control channel is required for each private subsession.

Each control channel may have a FILE-REQUEST token which is used to ensure that there is at most one outstanding file request on that channel at any instance. A File ASE requiring a file must grab this token before issuing the request and hold the token until it determines whether another node can supply the file. Dynamic control channels without a FILE-REQUEST token are permitted but only the creator of such a channel may issue file requests on that channel. The token for the session control channel MBFT-CONTROL is assigned the mnemonic FILE-REQUEST; the token for a sub-session control channel MBFT-CONTROL(p) is FILE-REQUEST(p).

Each data channel may have a FILE-TRANSMIT token which is used to ensure that there is only one file transfer in progress on that data channel at any instance. A transmitting File ASE grabs this token before offering a file for transmission and holds it for the duration of the file transfer, releasing the token after despatching the last block of file data. Dynamic data channels without a FILE-TRANSMIT token are permitted but only the creator of such a channel may send files on that channel. The token for the session broadcast data channel MBFT-DATA is assigned the mnemonic FILE-TRANSMIT, the token for a sub-session broadcast data channel MBFT-DATA(p) is FILE-TRANSMIT(p) whilst the token for acknowledged data channel MBFT-DATA(n) is FILE-TRANSMIT(n).

Together, these channels and tokens comprise the *resources* available to an MBFT session; they may be either static or dynamic. It is the responsibility of the File Transfer Application Resource Manager (File ARM) to determine the identity of these resources. For any given file transaction, the user application must specify the resources to be used by the File ASE. Static and dynamic resources are treated identically by the File ASE.

8.4.1 MBFT initialization

An MBFT session may be initiated locally by a user application or remotely through use of the GCC-Application-Invoke mechanism. In both cases, the parameters shown in Table 2 are passed to the File ARM. The action to be taken by the File ARM to identify the initial set of resources to be used for the session is determined by the session mode:

Static mode is used for unrestricted broadcasting of data to the conference. It is the simplest mode of operation as it uses predefined static channels and tokens. Applications can join and leave a static mode session at will. Although the File ASE may be employed by other application protocols simultaneously using different subsets of static channels and tokens, only one static mode of the File ARM is predefined by this Recommendation.

Multicast mode can be used for broadcasting of data when the static session is already in use. It is identical in function to static mode, but uses dynamic resources and so channel and token identities must be assigned via the GCC registry and MCS services by the creator of the multicast session (referred to as the *multicast creator*). All other participants (*multicast members*) may determine channel and token ids via the GCC Registry. Applications can join and leave a multicast session at will. There is no restriction on the number of multicast mode sessions in a conference.

Table 2 – File APE parameters

Parameter	Description
SessionMode	This parameter can have one of the following three values: *static*: This value indicates that the File APE should enroll using a Session Key composed of the MBFT object identifier and the sessionID parameter. It shall use the static predefined MBFT-CONTROL & MBFT-DATA channels and static FILE-TRANSMIT and FILE-REQUEST tokens. *dynamic multicast*: This value indicates that the File APE should enroll using a Session Key composed of the MBFT object identifier and the sessionID parameter. All channel and token resources are dynamic and are allocated by the creator of the multicast session using the MCS-CHANNEL-JOIN mechanism and GCC Registry mechanism respectively. Members of a multicast session determine token and channel ids via the GCC Registry. *dynamic private*: This value indicates that the File APE should enroll using a Session Key composed of the MBFT object identifier and the sessionID parameter. All token and channel resources are dynamic and are allocated by the Private Convenor File ARM using the GCC Registry mechanism and MCS-CHANNEL-CONVENE mechanism respectively. The File ARM creating the session may then admit to the MBFT-CONTROL and MBFT-DATA channels all peer File APEs whose MCS User Ids are present in the admitList protocol parameter. A Private Member File ARM must wait for its File APE to be admitted by the convenor of the private MCS channels before attempting to join them. Token identities are conveyed in-band by the channel convenor in the first transaction.

Table 2 – File APE parameters

Parameter	Description
sessionID	This parameter is used to differentiate the resources used by multiple sessions of the protocol that may be in existence simultaneously within the same MCS domain. The MCS Channel ID assigned to the MBFT-CONTROL channel is used as the Session ID, since this is guaranteed to be unique within the conference domain. The SessionID must be specified if the application wishes to participate in a static session or in an existing multicast or private session. It is omitted if the application wishes to create a new multicast or private session.

admitList	SessionMode = private and SessionID omitted
	List of GCC User IDs corresponding to the Nodes at which File APEs are to be admitted to the privately convened channels.
	else
	Omitted

Private mode is used for selective distribution of files to a subset of participants. It is the responsibility of the File ARM initiating the private session (referred to as the *private convenor*) to obtain tokens and channels using GCC and MCS respectively, and to admit peer File APEs (*private members*) to the channels. The identity of tokens is conveyed in-band, but may also be determined via the GCC Registry. There is no restriction on the number of private mode sessions in a conference. A private member File ARM must wait for its File APE to be admitted to the private channels and for the convenor to assign the tokens to be used for the session. Once the convenor leaves the private session, all remaining participants are expelled. Applications can only join a private session at the invitation of the session convenor.

In all cases the initial set of resources allows for the transfer of one file at a time. Concurrent transfer of multiple files is outlined in clause 8.6.

A File APE created in any mode must first establish a GCCSAP to allow it to communicate with the GCC provider at that node. When the node joins a conference, the GCC provider will issue a GCC-Application-Permission-to-Enroll with the Grant/Revoke flag set to Grant. The File ARM must then issue a GCC-Application-Enroll request regardless of whether the user application wishes to enroll at that time. If the user application does not wish to enroll, the File ARM must set the Enroll/Un-enroll flag in the GCC-Application-Enroll request to Un-enroll and specify the conference ID. No other parameters are required. The application may enroll at any time subsequently, unless permission is revoked by receipt of a GCC-Application-Permission-to-Enroll indication with the Grant/Revoke flag set to Revoke.

If the user application wishes to receive GCC-Application-Roster-Reports on all MBFT sessions in progress before deciding which session to participate in, the File ARM may enroll Inactive, specifying the Session Key without a Session ID. If the user application wishes to declare support of the MBFT protocol without consuming MCS resources, the File ARM may enroll Inactive without an MCS User ID.

When the user application undertakes to enroll active, the File ARM shall issue an MCS-Attach-User request to the MCS provider, using the Conference ID contained in the GCC-Application-Permission-to-Enroll indication as the Domain Selector. On receipt of a successful MCS-Attach-User confirm in response, the File ARM shall join the User ID channel indicated by issuing an MCS-Channel-Join request.

8.4.2 Static mode

After obtaining an MCS User ID, the File ARM shall join the MBFT static control and data channels by issuing two MCS-Channel-Join requests, specifying MBFT-CHANNEL-0 and MBFT-CHANNEL-1 as the respective channels to join. Once positive confirmation of joining these channels has been received, the File ARM shall enroll active by issuing a GCC-Application-Enroll request to the GCC provider, with the parameters specified in Table 3. The Active/Inactive flag shall be set to Active, the Session ID shall be specified as part of the Session Key, the Start-Up Channel shall be specified as Static and the full Application Capability list must be provided. See also Figure 6.

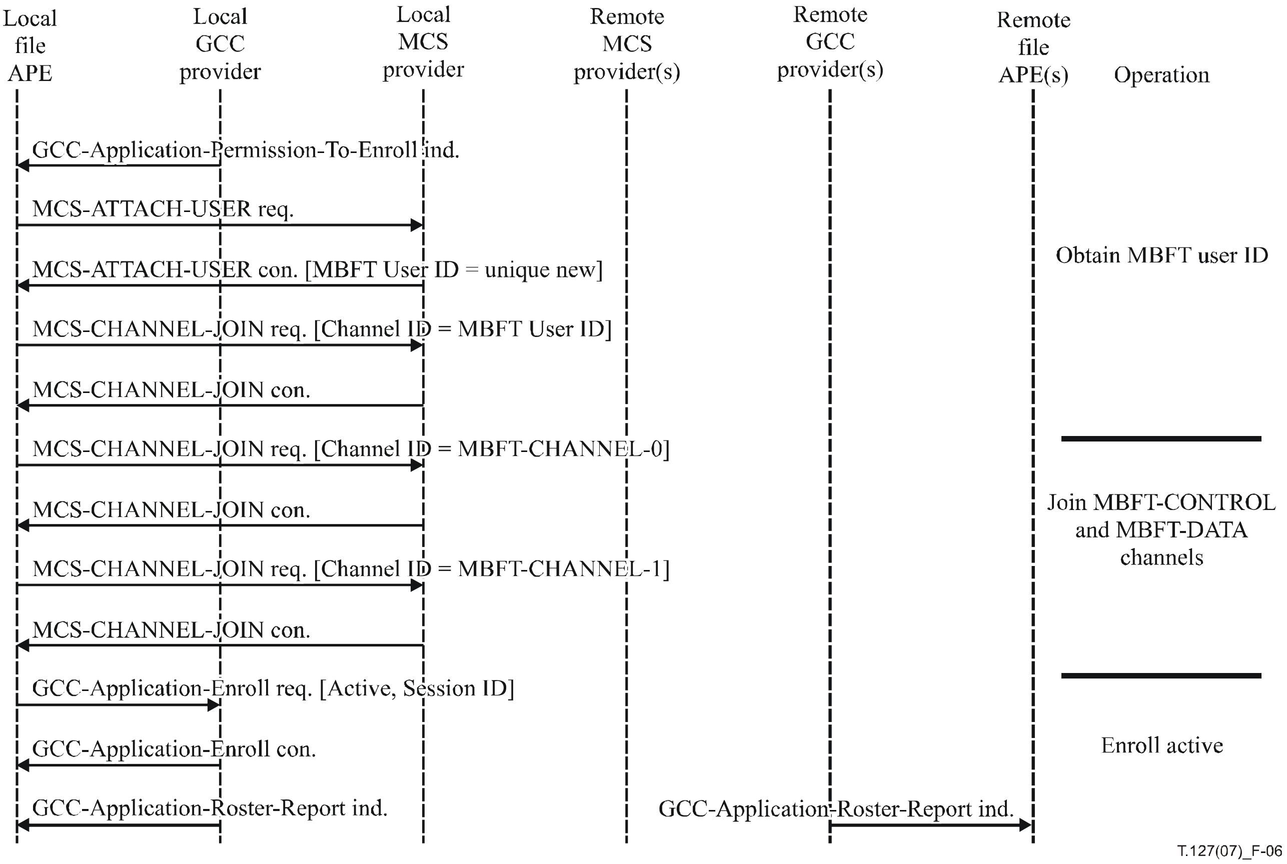

Local file APE
Local GCC provider
Local MCS provider
Remote MCS provider(s)
Remote GCC provider(s)
Remote file APE(s)
Operation
GCC-Application-Permission-To-Enroll ind.
MCS-ATTACH-USER req.
MCS-ATTACH-USER con. [MBFT User ID = unique new]
Obtain MBFT user ID
MCS-CHANNEL-JOIN req. [Channel ID = MBFT User ID]
MCS-CHANNEL-JOIN con.
MCS-CHANNEL-JOIN req. [Channel ID = MBFT-CHANNEL-0]
MCS-CHANNEL-JOIN con.
Join MBFT-CONTROL and MBFT-DATA channels
MCS-CHANNEL-JOIN req. [Channel ID = MBFT-CHANNEL-1]
MCS-CHANNEL-JOIN con.
GCC-Application-Enroll req. [Active, Session ID]
GCC-Application-Enroll con.
Enroll active
GCC-Application-Roster-Report ind.
GCC-Application-Roster-Report ind.
T.127(07)_F-06

Figure 6 – Static session protocol initiation sequence

Table 3 – Parameters for GCC-Application-Enroll Request

Parameter	Contents
Conference ID	Provided by GCC-Application-Permission-To-Enroll indication.
Session Key	{itu-t recommendation t 127 version(0) 1} and MBFT-SESSION-ID if this File APE parameter was specified.
Application User ID	Provided by MCS-Attach-User confirm.
Active/Inactive	Active when indicating that the File ARM has joined the MBFT-CONTROL and MBFT-DATA channels and determined any required MBFT token IDs. Inactive in the case of multicast or private modes when enrolling prior to joining the MBFT-CONTROL and MBFT-DATA channels.

Table 3 – Parameters for GCC-Application-Enroll Request

Parameter	Contents
Conducting Operation Flag	Set if the File APE is capable of becoming the session MBFT Conductor in conducted mode, i.e., if it can respond to MBFT-Privilege-Request PDUs. This flag must not be set if the Active/Inactive flag is set to Inactive.
Start-Up Channel	This parameter is dependent on the File APE parameters specified in Table 2: Static if SessionMode = Static Dynamic Multicast if SessionMode = Dynamic Multicast and sessionID omitted Dynamic Private if SessionMode = Dynamic Private and sessionID omitted Omitted otherwise
Non-Collapsing Capabilities List	No non-collapsing capabilities are specified by this protocol. This field may include non-standard non-collapsing capabilities specified by the user application.
Application Capability List	See Table 7; Omitted if Active/Inactive flag is set to Inactive.
Enroll/Unenroll	ENROLL

8.4.3 Multicast mode

After obtaining an MCS User ID, the File ARM shall examine the File APE sessionID parameter to determine whether it is to participate in an existing multicast session (as a multicast member) or to create a new one (as a multicast creator).

If the sessionID parameter is present, the File ARM shall attempt to join the session indicated by issuing a GCC-Application-Enroll request with the Active/Inactive flag set to Inactive and specifying the Session Key with the requisite Session ID. It shall then issue an MCS-Channel-Join request, specifying the Session ID of the chosen session as the Channel ID parameter. This channel is used as the MBFT-CONTROL channel. The File ARM must then identify and join the MBFT-DATA channel by issuing a GCC-Registry-Retrieve-Entry request, using the parameters given in Table 4. On receipt of the resulting GCC-Registry-Retrieve-Entry confirm, the File ARM shall examine the Registry Item parameter. The Channel ID contained in this parameter shall be used as the MBFT-DATA channel and the File ARM shall join it by issuing an MCS-Channel-Join request.

If it wishes to initiate a file transaction, the File ARM may then identify the FILE-TRANSMIT token to be used for the broadcast data channel by issuing a GCC-Registry-Retrieve-Entry request, using the parameters specified in Table 4. If the result parameter in the GCC-Registry-Retrieve-Entry confirm is 'entry not found', only the session creator is permitted to offer files on the broadcast data channel. The File ARM may also identify the FILE-REQUEST token to be used for the control channel by issuing a GCC-Registry-Retrieve-Entry request. If the result parameter in the GCC-Registry-Retrieve-Entry confirm is 'entry not found', only the session creator is permitted to request files on the control channel.

Table 4 – Parameters for GCC-Registry-Retrieve-Entry Request

Parameter	Contents
Conference ID	Provided by GCC-Application-Permission-To-Enroll indication
Registry Key	Registry Key formed as described in clause 8.4.5.

If the File ARM does not wish to initiate a file transaction, it does not need to identify the FILE-TRANSMIT token and FILE-REQUEST token during the enrollment process. Instead, it may determine the IDs of these tokens later from File-Offer or File-RequestPDUs issued by peer File ARMs on the MBFT-CONTROL channel.

Once all resources required for the session have been determined and channels joined, the File ARM shall issue a GCC-Application-Enroll request with the Active/Inactive flag set to Active, specifying the Session ID as part of the Session Key and providing the full Application Protocol Capability List. See Figures 7 and 8.

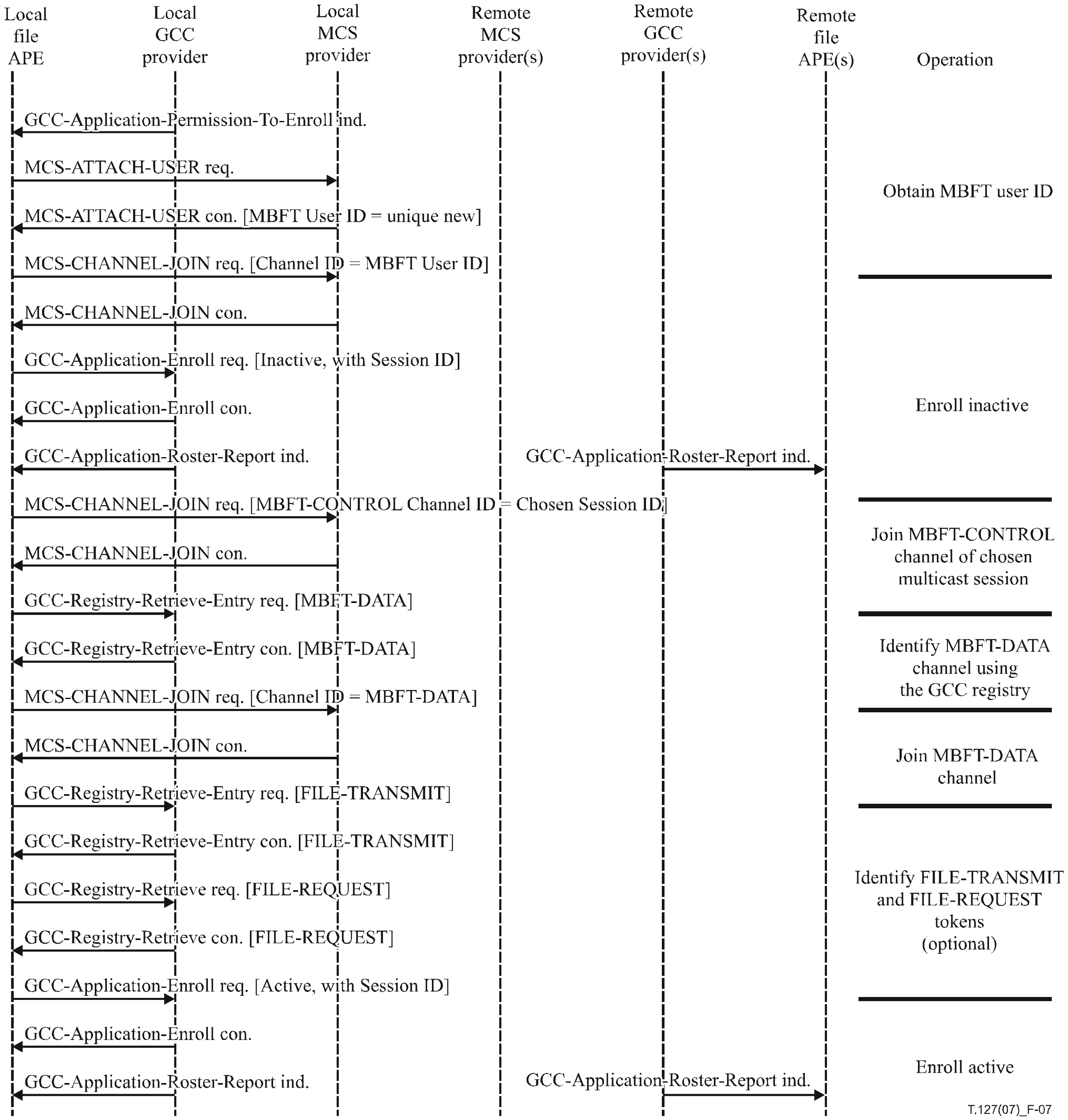

Local file APE
Local GCC provider
Local MCS provider
Remote MCS provider(s)
Remote GCC provider(s)
Remote file APE(s)
Operation
GCC-Application-Permission-To-Enroll ind.
MCS-ATTACH-USER req.
Obtain MBFT user ID
MCS-ATTACH-USER con. [MBFT User ID = unique new]
MCS-CHANNEL-JOIN req. [Channel ID = MBFT User ID]
MCS-CHANNEL-JOIN con.
GCC-Application-Enroll req. [Inactive, with Session ID]
GCC-Application-Enroll con.
Enroll inactive
GCC-Application-Roster-Report ind.
GCC-Application-Roster-Report ind.
MCS-CHANNEL-JOIN req. [MBFT-CONTROL Channel ID = Chosen Session ID]
Join MBFT-CONTROL channel of chosen multicast session
MCS-CHANNEL-JOIN con.
GCC-Registry-Retrieve-Entry req. [MBFT-DATA]
GCC-Registry-Retrieve-Entry con. [MBFT-DATA]
Identify MBFT-DATA channel using the GCC registry
MCS-CHANNEL-JOIN req. [Channel ID = MBFT-DATA]
MCS-CHANNEL-JOIN con.
Join MBFT-DATA channel
GCC-Registry-Retrieve-Entry req. [FILE-TRANSMIT]
GCC-Registry-Retrieve-Entry con. [FILE-TRANSMIT]
GCC-Registry-Retrieve req. [FILE-REQUEST]
Identify FILE-TRANSMIT and FILE-REQUEST tokens (optional)
GCC-Registry-Retrieve con. [FILE-REQUEST]
GCC-Application-Enroll req. [Active, with Session ID]
GCC-Application-Enroll con.
Enroll active
GCC-Application-Roster-Report ind.
GCC-Application-Roster-Report ind.
T.127(07)_F-07

Figure 7 – Multicast session protocol initiation sequence (member)

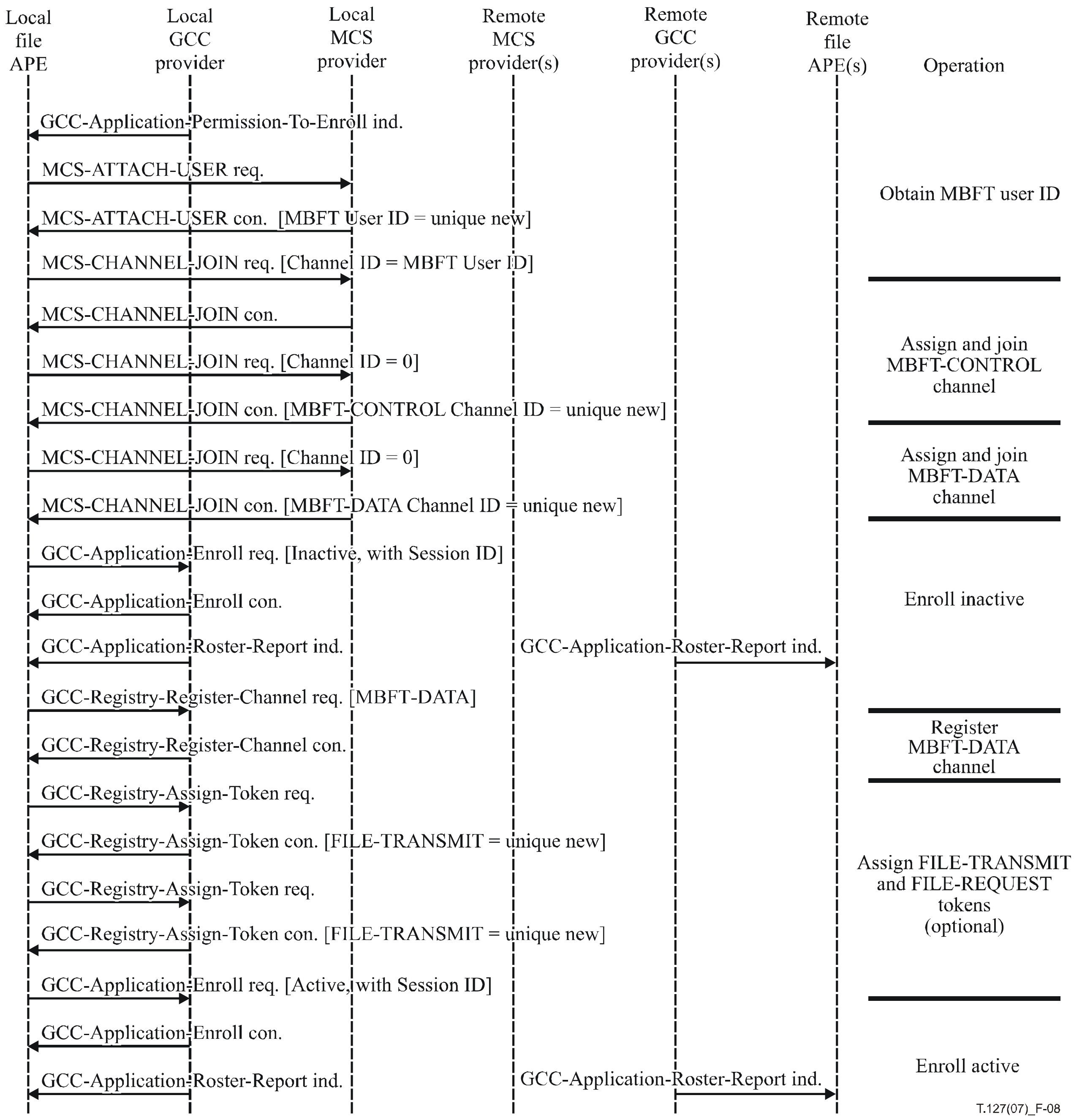

Local file APE
Local GCC provider
Local MCS provider
Remote MCS provider(s)
Remote GCC provider(s)
Remote file APE(s)
Operation
GCC-Application-Permission-To-Enroll ind.
MCS-ATTACH-USER req.
MCS-ATTACH-USER con. [MBFT User ID = unique new]
Obtain MBFT user ID
MCS-CHANNEL-JOIN req. [Channel ID = MBFT User ID]
MCS-CHANNEL-JOIN con.
MCS-CHANNEL-JOIN req. [Channel ID = 0]
MCS-CHANNEL-JOIN con. [MBFT-CONTROL Channel ID = unique new]
Assign and join MBFT-CONTROL channel
MCS-CHANNEL-JOIN req. [Channel ID = 0]
MCS-CHANNEL-JOIN con. [MBFT-DATA Channel ID = unique new]
Assign and join MBFT-DATA channel
GCC-Application-Enroll req. [Inactive, with Session ID]
GCC-Application-Enroll con.
Enroll inactive
GCC-Application-Roster-Report ind.
GCC-Application-Roster-Report ind.
GCC-Registry-Register-Channel req. [MBFT-DATA]
GCC-Registry-Register-Channel con.
Register MBFT-DATA channel
GCC-Registry-Assign-Token req.
GCC-Registry-Assign-Token con. [FILE-TRANSMIT = unique new]
GCC-Registry-Assign-Token req.
GCC-Registry-Assign-Token con. [FILE-TRANSMIT = unique new]
Assign FILE-TRANSMIT and FILE-REQUEST tokens (optional)
GCC-Application-Enroll req. [Active, with Session ID]
GCC-Application-Enroll con.
GCC-Application-Roster-Report ind.
GCC-Application-Roster-Report ind.
Enroll active
T.127(07)_F-08

Figure 8 – Multicast session protocol initiation sequence (creator)

If the File APE sessionID parameter is omitted, the File ARM shall attempt to create a new session. It shall first issue an MCS-Channel-Join request primitive with Channel ID = 0. The returned MCS-Channel-Join confirm, if successful, contains the assigned Channel ID which is to be used as the MBFT-CONTROL channel. This process is then repeated to assign and join the MBFT-DATA channel. The ARM shall then enroll inactive in this session by issuing a GCC-Application-Enroll request with the Active/Inactive flag set to Inactive, specifying the Session Key with the Session ID, but omitting the Application Capabilities List. Once the File ARM has received a GCC-Application-Roster Report containing an entry corresponding to the new session, the File ARM register the MBFT-DATA channel by issuing a GCC-Registry-Register-Channel request with the parameters given in Table 5.

Table 5 – Parameters for GCC-Registry-Register-Channel Request

Parameter	Contents
Conference ID	Provided by GCC-Application-Permission-To-Enroll indication.
Registry Key	Registry Key formed as described in clause 8.4.5.
Channel ID	Channel ID returned in MCS-Channel-Join confirm.

The File ARM may then assign FILE-TRANSMIT and FILE-REQUEST tokens by issuing two GCC-Registry-Assign-Token requests using the parameters given in Table 6. If the File ARM requires exclusive access to the broadcast data channel, it need not assign a FILE-TRANSMIT token. If it requires an exclusive right to request files, the File ARM need not assign a FILE-REQUEST token.

Table 6 – Parameters for GCC-Registry-Assign-Token Request

Parameter	Contents
Conference ID	Provided by GCC-Application-Permission-To-Enroll indication.
Registry Key	Registry Key formed as described in clause 8.4.5.

Once all resources have been successfully determined, the File ARM shall enroll actively by issuing a GCC-Application-Enroll request with the Active/Inactive flag set to Active, the Start-Up Channel specified as Dynamic Multicast and providing the full Application Protocol Capability List. The MBFT-CONTROL Channel ID is used as the Session ID in the Session Key.

8.4.4 Private mode

After obtaining an MCS User ID, the File ARM shall examine the File APE sessionID parameter to determine whether it is to participate in an existing private session (as a private member) or to create a new one (as a private convenor).

If the sessionID parameter is omitted, the File ARM shall attempt to create a new private session. The File ARM shall first issue an MCS-Channel-Convene request in order to obtain a private MBFT-CONTROL channel. If successful, the returned MCS-Channel-Convene confirm contains the ID of the channel allocated. The File ARM must then join this channel by issuing an MCS-Channel-Join request, specifying the channel id returned in the MCS-Channel-Convene confirm. This process must be repeated to assign and join a separate MBFT-DATA channel. The ARM shall then enroll inactive in the new session by issuing a GCC-Application-Enroll request with the Active/Inactive flag set to Inactive, specifying the Session Key with the MBFT-CONTROL channel ID as Session ID and omitting the Application Capabilities List.

The File ARM may then assign FILE-TRANSMIT and FILE-REQUEST tokens by issuing two GCC-Registry-Assign-Token requests using the parameters given in Table 6. If the File ARM requires exclusive access to the broadcast data channel, it need not assign a FILE-TRANSMIT token. If it requires an exclusive right to request files, the File ARM need not assign a FILE-REQUEST token.

The File ARM shall then enroll actively by issuing a GCC-Application-Enroll request with the Active/Inactive flag set to Active, the Start-Up Channel specified as Dynamic Private and providing the full Application Protocol Capability List. The MBFT-CONTROL Channel ID is used as the Session ID in the Session Key.

If there is no active peer File APE at one or more of the nodes that are to be invited to the private session, the File ARM shall issue a GCC-Application-Invoke request, specifying a list of GCC User IDs to be invited or NULL (indicating that all nodes in the conference are to be invited) as the

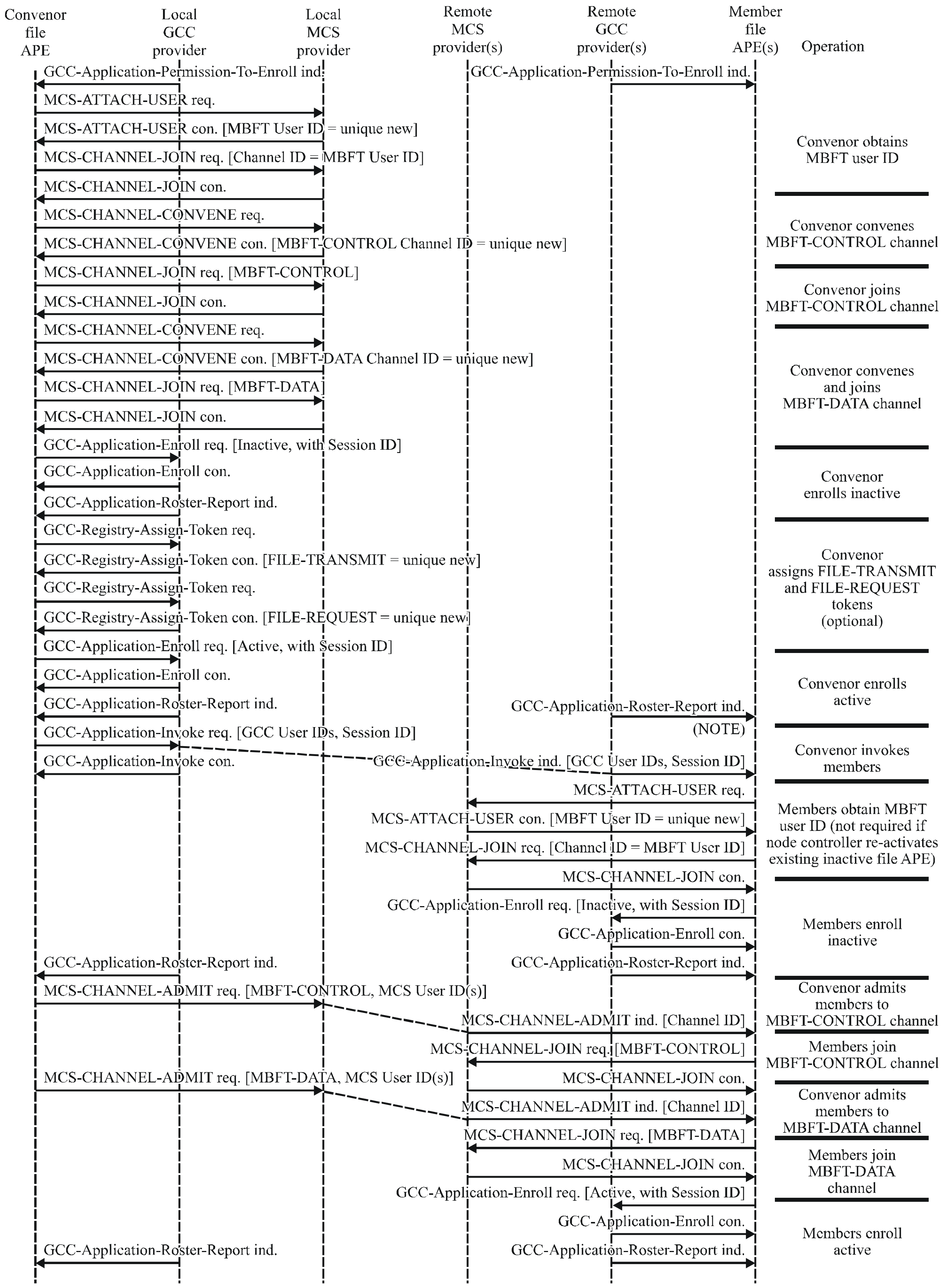
Destination Nodes parameter. It shall also specify Dynamic Private as the Start-Up Channel in the Application Protocol entry.

The convening File ARM must then wait until it has received a GCC-Application-Roster-Report indication containing the MCS User IDs of the File APEs to be invited to the private session (a timeout is recommended in case no MBFT application is initiated at one or more of the invited nodes). The File ARM shall then issue an MCS-Channel-Admit request for both the control and data channels, specifying the MCS User IDs of the File APEs to be invited as the list of MCS User IDs.

If the File APE sessionID parameter is present, the File ARM shall attempt to join the session indicated by issuing a GCC-Application-Enroll request with the Active/Inactive flag set to Inactive and specifying the Key with the requisite Session ID. It shall then wait until it has received an MCS-Channel-Admit indication from the File ARM at the node convening the private MBFT session. It then attempts to join the channel indicated in this primitive by issuing an MCS-Channel-Join request. A further MCS-Channel-Admit indication will be received from the private session convenor; the private member File ARM must also join the channel indicated in this primitive by issuing an MCS-Channel-Join request. To determine which of the two channels is to be used as the MBFT-CONTROL channel, the File ARM compares both Channel IDs with the Session ID. The Channel ID which matches the Session ID is the MBFT-CONTROL channel. The Channel ID that does not match the Session ID, but has the same channel manager User ID is the corresponding MBFT-DATA channel. When a private member File ARM is remotely invoked, the Session ID (and thus MBFT-CONTROL Channel ID) is obtained from the GCC-Application-Invoke indication.

Following receipt of a GCC-Application-Roster-Report indication containing the User ID of its File APE, the File ARM shall re-enroll with the Active/Inactive flag set to Active, including the Session ID as part of the Session Key and providing the full Application Protocol Capability List.

The identities of tokens to be used for the private session are conveyed by the convening File ASE when it initiates the first transaction on the control channel (i.e., in the File-OfferPDU or File-RequestPDU). Private members may individually access the GCC-Registry to determine token identities, but it must be emphasized that this approach is much less efficient. To identify the FILE-TRANSMIT token used with the broadcast data channel, the File ARM shall issue a GCC-Registry-Retrieve-Entry request, using the parameters specified in Table 4. If the result parameter in the GCC-Registry-Retrieve-Entry confirm is 'entry not found', only the session convenor is permitted to offer files on the broadcast data channel. The File ARM may also identify the FILE-REQUEST token to be used for the control channel by issuing a GCC-Registry-Retrieve-Entry request. If the result parameter in the GCC-Registry-Retrieve-Entry confirm is 'entry not found', only the session convenor is permitted to request files on the control channel. See Figure 9.


Convenor file APE
Local GCC provider
Local MCS provider
Remote MCS provider(s)
Remote GCC provider(s)
Member file APE(s)
Operation
GCC-Application-Permission-To-Enroll ind.
GCC-Application-Permission-To-Enroll ind.
MCS-ATTACH-USER req.
MCS-ATTACH-USER con. [MBFT User ID = unique new]
MCS-CHANNEL-JOIN req. [Channel ID = MBFT User ID]
MCS-CHANNEL-JOIN con.
Convenor obtains MBFT user ID
MCS-CHANNEL-CONVENE req.
MCS-CHANNEL-CONVENE con. [MBFT-CONTROL Channel ID = unique new]
Convenor convenes MBFT-CONTROL channel
MCS-CHANNEL-JOIN req. [MBFT-CONTROL]
MCS-CHANNEL-JOIN con.
Convenor joins MBFT-CONTROL channel
MCS-CHANNEL-CONVENE req.
MCS-CHANNEL-CONVENE con. [MBFT-DATA Channel ID = unique new]
MCS-CHANNEL-JOIN req. [MBFT-DATA]
MCS-CHANNEL-JOIN con.
Convenor convenes and joins MBFT-DATA channel
GCC-Application-Enroll req. [Inactive, with Session ID]
GCC-Application-Enroll con.
GCC-Application-Roster-Report ind.
Convenor enrolls inactive
GCC-Registry-Assign-Token req.
GCC-Registry-Assign-Token con. [FILE-TRANSMIT = unique new]
GCC-Registry-Assign-Token req.
GCC-Registry-Assign-Token con. [FILE-REQUEST = unique new]
Convenor assigns FILE-TRANSMIT and FILE-REQUEST tokens (optional)
GCC-Application-Enroll req. [Active, with Session ID]
GCC-Application-Enroll con.
GCC-Application-Roster-Report ind.
GCC-Application-Roster-Report ind.
(NOTE)
Convenor enrolls active
GCC-Application-Invoke req. [GCC User IDs, Session ID]
GCC-Application-Invoke con.
GCC-Application-Invoke ind. [GCC User IDs, Session ID]
Convenor invokes members
MCS-ATTACH-USER req.
MCS-ATTACH-USER con. [MBFT User ID = unique new]
MCS-CHANNEL-JOIN req. [Channel ID = MBFT User ID]
MCS-CHANNEL-JOIN con.
Members obtain MBFT user ID (not required if node controller re-activates existing inactive file APE)
GCC-Application-Enroll req. [Inactive, with Session ID]
GCC-Application-Enroll con.
GCC-Application-Roster-Report ind.
GCC-Application-Roster-Report ind.
Members enroll inactive
MCS-CHANNEL-ADMIT req. [MBFT-CONTROL, MCS User ID(s)]
MCS-CHANNEL-ADMIT ind. [Channel ID]
Convenor admits members to MBFT-CONTROL channel
MCS-CHANNEL-JOIN req.[MBFT-CONTROL]
MCS-CHANNEL-JOIN con.
Members join MBFT-CONTROL channel
MCS-CHANNEL-ADMIT req. [MBFT-DATA, MCS User ID(s)]
MCS-CHANNEL-ADMIT ind. [Channel ID]
Convenor admits members to MBFT-DATA channel
MCS-CHANNEL-JOIN req. [MBFT-DATA]
MCS-CHANNEL-JOIN con.
Members join MBFT-DATA channel
GCC-Application-Enroll req. [Active, with Session ID]
GCC-Application-Enroll con.
GCC-Application-Roster-Report ind.
GCC-Application-Roster-Report ind.
Members enroll active

NOTE – GCC-Application-Invoke ind is sent to the Node Controller.

T.127(07)_F-09

Figure 9 – Private session protocol initiation sequence

8.4.5 Forming Registry Keys

To determine the identity of a dynamic token or channel via the GCC Registry, a File ARM must form a registry key which consists of the Session Key for the current MBFT session and an MBFT Resource ID. Resource IDs are defined in Tables 29 and 30 for channels and tokens respectively.

8.5 MBFT capabilities

Capabilities negotiation for file transfer applications is done via the application enrollment mechanism. When a File ARM issues a GCC-Application-Enroll request with the Active/Inactive flag set to Active, it shall indicate the capabilities of its user application in the Application Capabilities List parameter of the request.

A File APE is notified of the capabilities available within its session by the receipt of a GCC-Application-Roster-Report indication which contains the Application Roster for that session. The Application Roster includes a list of nodes for which a peer File APE has enrolled. For each node, the list contains the GCC User ID of that node and the MCS User ID(s) of the peer File APE(s) at that node. The Application Roster has an instance number and contains flags to indicate whether File APEs have joined or left the session since the previous Application-Roster-Report was issued. It also contains a flag indicating whether the Application Capabilities List has been updated since the last roster and, if so, the new Capabilities List. If a File APE has newly enrolled, the Applications Capabilities List is updated, as this File APE does not have access to previous instances of the list.

The Application Capabilities List received in the GCC-Application-Roster-Report indication corresponds to the collapsed Application Capabilities Lists of all enrolled peer File APEs, i.e., it includes an entry for each capability which has been declared by any peer File APE. For each entry it includes the Capability ID, the number of peer File APEs (including the local one) which had advertised this capability and, for capabilities in the Unsigned Minimum class, the minimum value of the parameter among all peer File APEs which declared this capability. Table 7 defines the capability list elements for MBFT and the rules for collapsing each element. Note that certain capabilities are specified as being dependent on other capabilities. This means that the capability must not be included in the Application Protocol Capability List unless the capability on which it depends is also included.

Table 7 – Application capability list elements

Capability name default value and description	ID	Class	APE Count Rule	Capability value range	Dependency
Maximum file size (default: unlimited) Each File APE must specify the maximum file data payload in octets that it is capable of receiving.	1	Unsigned minimum	> 0	(65536..MAX)	–
Maximum data payload (default: 8 192) This is the maximum number of octets allowed in the data field of File-Start and File-Data PDUs.	2	Unsigned minimum	= ALL	(8193..65536)	–

Table 7 – Application capability list elements

Capability name default value and description	ID	Class	APE Count Rule	Capability value range	Depen-dency
V.42 *bis*-Compression (default: uncompressed) This capability is used to negotiate use of V.42 *bis* compression for file data. File APEs should assert this capability if they are able to receive V.42 *bis* compressed data.	3	Logical	= ALL	–	–
V.42 *bis*-Number-of-Codewords (default: 512) Specifies the total number of codewords to be used by the V.42 *bis* compression algorithm. This is an upper bound on V.42 *bis* parameter P1. V.42 *bis* does not impose an upper limit on its value.	4	Unsigned minimum	= D	(513..65535)	V.42 *bis*-Compression
V.42 *bis*-Max-String-Length (default: 6) Specifies the maximum string length input to the V.42 *bis* encoder. This is an upper bound on V.42 *bis* parameter P2.	5	Unsigned minimum	= D	(7..250)	V.42 *bis*-Compression
Non-standard capability (default: unspecified) This is used to negotiate non-standard functions, including non-standard compression techniques. Any number of these can appear in the Application Capability List, provided each has a unique Non-Standard Identifier. The interpretation of these capabilities is not defined in this Recommendation.	Non-Standard Identifier	–	–	–	–
APE Count Rule: > 0 – If the numberOfEntities parameter returned by the GCC-Application-Roster-Report indication for this capability is greater than zero, then the result of the unsigned minimum operation is established, otherwise the default value of the capability is established. = ALL – The numberOfEntities parameter returned by the GCC-Application-Roster-Report indication for this capability must equal the number of active File APEs enrolled in the current session, otherwise the default value of the capability is established. = D – If the numberOfEntities parameter returned by the GCC-Application-Roster-Report indication for this capability is equal to the numberOfEntities parameter for the capability on which it depends, then the result of the unsigned minimum operation is established, otherwise the default value of the capability is established.					

At any time while a File APE is enrolled in a conference, it may receive additional GCC-Application-Roster-Report indication, notifying it of a change in the roster contents. This may be due to new peer File APEs enrolling in the conference, peer File APEs leaving the conference, or peer File APEs having modified their enrollment information.

A File APE may change its Application Capabilities List at any time by re-enrolling. To do this, the File ARM issues a GCC-Application-Enroll request with the Enroll/Unenroll flag set to Enroll, including the revised Application Capabilities List, and providing all other parameters as included in the initial active enrollment. This may result in a change to the collapsed capability set, in which case all File APEs in the session will receive a GCC-Application-Roster-Report indication.

Any change in the collapsed capability set of an MBFT session shall not affect transactions already in progress. Changes shall take effect when the next transaction is initiated.

8.6 Support of additional concurrent file transfers

Two mechanisms are available to support simultaneous transmission of more than one file in a conference:

1) Create an additional MBFT session using the procedures defined above.

2) Create acknowledged data channels within the existing MBFT session.

For static and multicast modes, multicast or private channels may be used; for private mode, only private channels should be used. Particular care should be exercised in the use of multicast channels as such channels cease to exist when all joined users leave. It is thus recommended that File APEs only attempt to use a multicast channel if they have not left the channel since it was created or since the last transaction on the channel.

Each acknowledged data channel MBFT-DATA(n) requires its own FILE-TRANSMIT(n) token in order to manage file transactions on the channel, unless use is restricted to the channel creator.

8.6.1 Multicast channels

A File ARM wishing to use an acknowledged multicast data channel must first issue an MCS-Channel-Join request with Channel ID = 0. The returned MCS-Channel-Join confirm, if successful, contains the assigned Channel ID. If the data channel is to be a shared acknowledged data channel, the File ARM must then issue a GCC-Registry-Assign-Token request in order to obtain a FILE-TRANSMIT(n) token. The registry key is formed using the process described in clause 8.4.5. The Channel and Token IDs are conveyed to peer File APEs on the MBFT-CONTROL channel when the creator issues a File-Offer PDU (to transmit a file) or a File-Request PDU (to receive a file) for a file transaction on the new data channel.

8.6.2 Private channels

A File ARM wishing to use a shared acknowledged private data channel must first issue an MCS-Channel-Convene request. If successful, it shall join the channel assigned in the MCS-Channel-Convene confirm by issuing an MCS-Channel-Join request. The File ARM must then issue a GCC-Registry-Assign-Token request in order to obtain a FILE-TRANSMIT(n) token. The registry key is formed using the process described in clause 8.4.5.

A File ARM wishing to use an exclusive private data channel must first issue an MCS-Channel-Convene request. It does not need to join this channel (since it will never receive data on the channel), nor does it need to assign a token.

Once the resources have been successfully assigned, the File ARM shall issue an MCS-Channel-Admit request, specifying the MCS User IDs of the File APEs to be admitted to the channel as the list of MCS User IDs.

File APEs being admitted to the private data channel will receive an MCS-Channel-Admit indication, and must then attempt to join the channel indicated by issuing an MCS-Channel-Join request.

The FILE-TRANSMIT(n) Token ID (if applicable) is conveyed to peer File APEs on the MBFT-CONTROL channel when the creator issues a File-Offer PDU (to transmit a file) or a File-Request PDU (to receive a file) for a file transaction on the new data channel.

8.7 Selective file transfer

Two mechanisms are available to allow distribution of files to a subgroup of participants:

1) Convene a new private session.

2) Establish a private sub-session within the existing MBFT session.

The latter approach is outlined below. Once the communication path has been established, the mechanism used for file exchange is identical to that used for the initial communication path.

A File ARM wishing to initiate a private file transfer within an existing MBFT session must assign a private control channel and private data channel using MCS services. The File ARM first issues an MCS-Channel-Convene request for the control channel. If this is successful, it issues an MCS-Channel-Join request, specifying the Channel ID returned in the MCS-Channel-Convene confirm. This process is then repeated for the private data channel. The convening File ARM must then issue a Private-Channel-Join-InvitePDU to the MCS User ID of each File APE to be invited to the private channels, specifying the identity of the control and data channels as shown in Table 8. This PDU also indicates whether the data channel mode is broadcast or acknowledged.

Table 8 – Private-Channel-Join-InvitePDU

Parameter	Description
Control Channel ID	This identifies the private control channel to be joined
Data Channel ID	This identifies the private data channel to be joined
Mode	This flag is set to TRUE for broadcast mode, FALSE for acknowledged mode

The File ARM then invites other File APEs to join the control channel by issuing an MCS-Channel-Admit request, specifying the channel returned in the MCS-Channel-Convene confirm as Channel ID and including the list of File APE MCS User IDs to be invited to the private channel. This process is then repeated for the private data channel.

A File ARM which receives a Private-Channel-Join-InvitePDU with the Mode flag set to TRUE must attempt to join both the control and data channels indicated in the PDU when it receives the corresponding MCS-Channel Admit indications from the sub-session convenor. It shall then indicate whether both channels were successfully joined by sending a Private-Channel-Join-ResponsePDU to the channel convenor, with the Result parameter set accordingly.

A File ARM which receives a Private-Channel-Join-InvitePDU with the Mode flag set to FALSE must attempt to join the control channel indicated in the PDU when it receives the corresponding MCS-Channel Admit indication from the sub-session convenor. It may optionally join the acknowledged data channel at this time. The File ARM shall then indicate whether the control channels was successfully joined by sending a Private-Channel-Join-ResponsePDU to the channel convenor, with the Result parameter set accordingly.

Once the sub-session convenor has received a Private-Channel-Join-ResponsePDU from each of the File APEs invited to the sub-session, it may begin its file transactions. A timeout is recommended in case one or more of the File APEs fail to respond. See Table 9.

Table 9 – Private-Channel-Join-ResponsePDU

Parameter	Description
Channel ID	This identifies the private control channel which the File APE has been invited to join.
Result	If the Mode flag in the corresponding Private-Channel-Join-Invite PDU was set to TRUE, this parameter indicates whether or not the attempt to join the private control and broadcast data channel was successful. If the Mode flag in the corresponding Private-Channel-Join-Invite PDU was set to FALSE, this parameter indicates whether or not the attempt to join the private control channel was successful.

Once the convening File ARM has received Private-Channel-Join-ResponsePDU from all invited File APEs, it may allow its File ASE to use the channel. The File ARM should monitor the Application Roster in case any of the invited File APEs leave the session prematurely, and should have a timeout in case any of the File APEs fail to issue a Private-Channel-Join-ResponsePDU.

The convening File ARM may obtain a FILE-TRANSMIT(n) token and FILE-REQUEST(n) token at any time before the File ASE needs to use the private channels. This is done by issuing two GCC-Registry-Assign-Token requests in succession forming the registry key as described in clause 8.4.5.

Note that the first transaction must be initiated by the channel manager as the identity of any tokens to be used with the private channel is conveyed in-band in the first File-Offer or File-Request PDU.

NOTE – The convening file ARM should exercise caution when expelling members from or disbanding a private data channel, since MCS may expel receiving File APE's from the channel before all File-Data PDUs have been delivered.

8.8 Leaving an MBFT session

If an application wishes to leave an MBFT session, its File ARM shall issue a GCC-Application-Enroll request with the Enroll/Unenroll flag set to Unenroll. No other parameters are required.

If, at any time, the File ARM receives a GCC-Application-Roster-Report indication in which it is no longer included (i.e., its MBFT User ID is absent), the File ARM shall issue an MCS-Detach-User request immediately to detach from the specified conference. The application is no longer considered to be enrolled in the conference at this time but may attempt to re-enroll in the conference by issuing a GCC-Application-Enroll request.

If, at any time, the File ARM receives a GCC-Application-Permission-To-Enroll indication with the Grant/Revoke flag set to Revoke, it shall issue an MCS-Detach-User request immediately to detach from the specified conference. The application is no longer considered to be enrolled in the conference at this time and shall not attempt to re-enroll.

8.9 File exchange

This subclause considers MBFT transactions initiated by either a sending site or a receiving site. File transactions are performed by the File Transfer Application Service Element (File ASE).

8.9.1 Transmitter invoked operation

A File ASE wishing to transmit a file must first grab the FILE-TRANSMIT token associated with the data channel to prevent other File ASEs from attempting to send data to the channel at the same time. If there is no token associated with the data channel, then only the channel creator may attempt to send files on it. If the token is held by another File ASE (indicating that a file transfer is in progress), then the File ASE wishing to transmit may either wait until it is released, or attempt to use another data channel. Alternatively, the File ASE may request the token from the current transmitter by issuing an MCS-Token-Please request primitive, specifying the Token ID of the FILE-TRANSMIT token as Token ID. A File ASE receiving an MCS-Token-Please indication primitive whilst it is transmitting a file may choose to ignore it, or may offer the FILE-TRANSMIT token to the requesting File ASE at the end of the file transfer by issuing an MCS-Token-Give request, specifying the MCS User ID of the requesting File ASE as User id to receive token and the FILE-TRANSMIT Token ID as Token id. On receipt of an MCS-Token-Give indication, the requesting File ASE shall issue an MCS-Token-Give response primitive with result parameter 'rt-successful' or 'rt-user-rejected', depending on whether the File ASE still requires the token or not. If the result is 'rt-successful', the File ASE may assume that it has possession of the token.

Once the File ASE is in possession of the FILE-TRANSMIT token (or is the creator of an exclusive acknowledged data channel), it offers the file to all intended recipients on the control channel by issuing a File-OfferPDU. The contents of the File-OfferPDU are defined in Table 10. For file transfers on the broadcast data channel, the Ack flag must be set to FALSE to indicate that no acknowledgement is required. For file transfers on acknowledged data channels, the Ack flag must be set to TRUE to indicate that receiving File ASEs must signal whether or not they wish to accept the data. Sufficient information must be conveyed in the File Header parameter to uniquely identify the file and enable receivers to determine whether it is required. It is up to the transmitting File ASE to decide whether to transmit all or a subset of the T.434 parameters available for the file. It is recommended that any parameters deemed to be important to the application by the transmitter are included in the File-OfferPDU. If a transmitter wishes to verify whether receivers are able to support such parameters, it should use an acknowledged data channel for the transaction. The method of determining whether a file is needed is a local implementation issue; the application may determine automatically whether a file is already present, or may require interaction with an operator.

File ASEs receiving a File-Offer with the Ack flag set to TRUE and wishing to receive the offered file must join the data channel indicated before sending back a confirmation (File-Accept – Table 11) to the transmitter. File ASEs not wishing to receive the file reply with a negative acknowledgement (File-Reject – Table 12) do not join the data channel. Similarly, File ASEs which are unable to support one or more of the parameters included in the File Header should reject the offer. If a File ASE which does not require a file is joined to the data channel when it receives the File-Offer, it must leave the channel by issuing an MCS-Channel-Leave request. If a File ASE is unable to support further concurrent file transfers, it should either delay its response until the file transfer is complete or issue a File-Reject. Only File ASEs included in the Application Roster instance specified in the File-OfferPDU may respond, so that the transmitting File ASE knows how many acknowledgements to accept. File ASEs not included in the specified Application Roster instance must ignore the File-Offer.

File ASEs receiving a File-Offer with the Ack flag set to FALSE must be prepared to receive data on the broadcast data channel immediately. If the file is not required, the File ASE must either discard incoming data or leave the session.

Table 10 – File-OfferPDU

Parameter	Description
File Header	This parameter uses the T.434 file header structure and should include sufficient information to allow intending recipients to determine whether the file is required. File name and size are suggested as a minimum set.
Data Channel ID	This identifies the channel on which the file will be transmitted.
File Handle	This is a unique handle used to reference the file throughout the file transfer operation. It need only be locally unique at the transmitting File APE.
Roster Instance	Application Roster Instance Number as returned in the most recently received GCC-Application-Roster-Report indication. Only File APEs which are enrolled active in the identified roster may respond to this PDU. This field is only included if the Ack flag is set to TRUE. File APEs may need to wait to receive the corresponding roster report indication before they can generate a reply.
FILE-TRANSMIT Token ID	This identifies the token used to regulate file transmission on the channel specified by the Data Channel ID parameter. If the token is omitted, then only the creator of the data channel is permitted to send files on that channel. If a FILE-TRANSMIT token has been assigned to the data channel it must be included.
FILE-REQUEST Token ID	This identifies the token used to regulate file requests. If the token is omitted, then only the creator of the control channel is permitted to issue File-RequestPDUs. If a FILE-REQUEST token has been assigned to the control channel it must be included.
File Request Handle	This is a locally unique handle used to reference a file request. It shall be set to the same value as the handle in the corresponding File-Request PDU if the File-Offer is being generated in response to a File-Request, otherwise the handle is omitted.
MBFT User ID	This parameter is omitted unless the File-Offer is generated in response to a File-Request, in which case it is used to identify the File APE that issued the File-Request PDU.
Compression Specifier	This optional field specifies the compression technique (if any) applied to the data payload of File-Start and File-DataPDUs.
Compressed Filesize	This optional parameter is the number of octets in the data payload after compression has been applied.
Ack Flag	This flag is set to TRUE if the file is to be sent on an acknowledged data channel, requiring the File-OfferPDU to be acknowledged by all receiving File APEs. If the flag is set to FALSE, the file will be transmitted on the broadcast data channel and no acknowledgement is required.

Table 11 – File-AcceptPDU

Parameter	Description
File Handle	This is a unique handle used to reference the file throughout the file transfer operation. Its value is obtained from the File-Offer PDU.

Table 12 – File-RejectPDU

Parameter	Description
File Handle	This is a unique handle used to reference the file throughout the file transfer operation. Its value is obtained from the File-Offer PDU.
Reason	This parameter is used to inform the transmitter why the intended recipient is unable or does not wish to receive the file offered.

If the file is to be distributed on the broadcast data channel, the transmitting File ASE may start to send the file on this channel immediately after despatching the File-OfferPDU, otherwise it must wait until it has received an acknowledgement from all File ASEs joined to the control channel. In the latter case, the transmitter should monitor the application roster to determine which sites are currently connected, since File APEs may have left the control channel after the File-OfferPDU was issued. It is recommended that a timeout should be used to prevent File ASEs which fail to respond from locking the application. No timeout values are specified in this protocol specification; they should be sufficiently large to allow for network latencies and user response times.

Files are transported in a File-StartPDU followed by zero or more File-DataPDUs (see Tables 13 and 14). Note that the File-StartPDU contains a T.434 header. This is included to ensure that the entire file structure (header plus data) is transported as a single entity and also because the information provided in the File-OfferPDU may only be a subset of the header. The EOFflag is used to indicate the end of file.

Table 13 – File-StartPDU

Parameter	Description
File Header	This parameter uses the T.434 file header structure and should include all fields defined in the source file, irrespective of whether they were included in the preceding File-OfferPDU.
File Handle	This is a unique handle used to reference the file throughout the file transfer operation. It shall be set to the same value as the handle in the File-OfferPDU corresponding to this transfer.
EOF Flag	This flag is set to TRUE to indicate end of file.
CRC Flag	This indicates whether the last File-DataPDU (or File-StartPDU if the EOF flag is set to TRUE) contains a cyclic redundancy check.
Compression Specifier	This optional field specifies the compression technique (if any) applied to the data payload of File-Start and File-DataPDUs.
Compressed Filesize	Number of octets in the compressed file, excluding the header.
Data Offset	This parameter specifies the starting offset in octets from the beginning of the file data (so zero denotes the origin).
Data	The maximum number of octets of binary file data which can be included in this field is negotiated via the capability exchange mechanism.
CRC Check	A 32-bit cyclic redundancy check is included in the File-Start PDU if the CRC flag is set to TRUE and either a cumulative CRC is being used or the EOF flag is set to TRUE.

Table 14 – File-DataPDU

Parameter	Description
File Handle	This is a unique handle used to reference the file throughout the file transfer operation. It shall be set to the same value as the handle in the File-OfferPDU corresponding to this transfer.
EOF Flag	This flag is set to TRUE in the last File-DataPDU to indicate end of file.
Abort Flag	The abort flag is set to TRUE if the file transfer is terminated abnormally, either as a result of conductor intervention or at the transmitter's volition. No further File-DataPDUs will be sent in the current transmission after a File-DataPDU with the Abort flag set to TRUE.
Data	The maximum number of octets of binary file data which can be included in this field is negotiated via the capability exchange mechanism.
CRC Check	A 32-bit cyclic redundancy check is included in the File-Data PDU if the CRC flag was set to TRUE in the corresponding File-Start and either a cumulative CRC is being used or the EOF flag is set to TRUE.

Although the T.120-series provides reliable data transfer, some user applications may have an additional requirement to validate the data by using a Cyclic Redundancy Check (CRC). T.127 allows the possibility for transmitters to optionally include a CRC with the file data. This may either be a single CRC included in the last File-DataPDU (or File-Start PDU if the data can be transported in a single PDU), or a cumulative CRC included in the File-StartPDU and all subsequent File-DataPDUs. The presence of a CRC is indicated by setting the CRC flag in the File-StartPDU. Receivers may ignore CRCs if they are not required by the user application. The method of calculating the CRC is illustrated in Figure 10. Octets of data are input to the CRC calculation sequentially, low order bit first.

The CRC is reset at the start of each file transfer and is calculated from the file data payload of File-Start and File-Data PDUs using the 32-bit frame checking sequence defined in [ISO/IEC 13239]. If the file data is compressed, the CRC applies to the compressed data and not the source data.

If V.42 *bis* compression is applied to the file data payload, the encoder is initialized at the beginning of the file and processes data payload octets continuously until the end of the file. The encoder is not reset at the end of each PDU.

On completion of the transfer of each file, File ASEs must remain joined to both control and data channels to avoid the need to leave and re-join for subsequent operations. However, if a File ASE is joined to an acknowledged data channel and is subsequently offered a file which it does not require, it must leave the data channel before sending back a File-Reject.

When the transmitting node has despatched the last File-DataPDU, it should release the FILE-TRANSMIT token (if present) by issuing an MCS-Token-Release request to allow other sites to initiate file transmissions on the data channel. If the transmitter had previously received an MCS-Token-Please during the transfer, it may instead offer the token to the requesting File ASE by issuing an MCS-Token-Give request. If MCS-Token-Please indications have been received from more than one File ASE, it is a local matter to decide which to respond to.

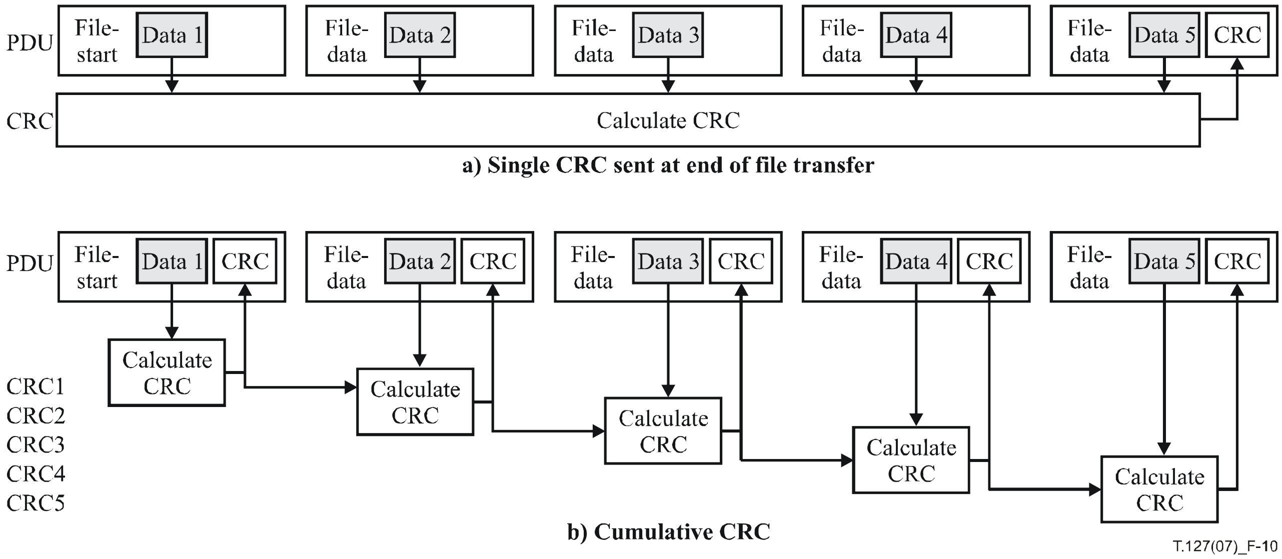


Figure 10 – Calculation of CRC

8.9.2 Receiver invoked operation

An optional MBFT feature is the ability to request transmission of a specific file (for file retrieval applications) or partial retransmission of a file (to allow recovery in the event of a site being disconnected during a file transfer). Partial file retransmission is particularly useful if a bulk file transfer is interrupted. As the protocol does not require each File-DataPDU to be acknowledged, it is the responsibility of the File ASE issuing the request to indicate how much of the file was successfully received.

To initiate a file request, a File ASE must first grab the FILE-REQUEST token before issuing a File-Request PDU. This ensures that there is at most one outstanding request on the control channel. If there is no FILE-REQUEST token, then only the channel creator may issue a File-Request. If the token is held by another File ASE, then the File ASE wishing to initiate a file request may ask for the token by issuing an MCS-Token-Please request, specifying the Token ID of the FILE-REQUEST token as Token id. The holder of the token may choose to give the FILE-REQUEST token to the requesting File ASE once it has determined whether its own file request can be satisfied. To do this, the token holder must issue an MCS-Token-Give request, specifying the MCS User ID of the requesting File ASE as User id to receive token and the FILE-REQUEST Token ID as Token id. On receipt of an MCS-Token-Give indication, the requesting File ASE shall issue an MCS-Token-Give response primitive with result parameter 'rt-successful' or 'rt-user-rejected', depending on whether the File ASE still requires the token or not. If the result is 'rt-successful', the File ASE may assume that it has possession of the token.

The contents of the File-RequestPDU are defined in Table 15. Sufficient information should be included in the request to enable recipients of the PDU to determine whether they are able to supply the file. For partial retransmission of a file, the File ASE must specify the number of octets successfully received prior to the file transfer interruption. A handle is included in the PDU so that responses can be correlated with the request. A FILE-TRANSMIT token must be included in the PDU to ensure that only one File ASE attempts to fulfil the request.

Table 15 – File-RequestPDU

Parameter	Description
File Header	This parameter uses the T.434 file header structure and should include sufficient information to allow transmitters to determine whether they are able to fulfil the file request. File name and size are suggested as a minimum set.
Data Channel ID	This identifies the channel on which the file must be transmitted. It may be either the broadcast data channel, an acknowledged data channel, or the MBFT User ID of the requesting File APE.
Request Handle	This is a unique handle used to reference the file request. It need only be locally unique at the requesting File APE.
Roster Instance	Application Roster Instance Number as returned in the most recently received GCC-Application-Roster-Report indication. Only File APEs which are enrolled active in the identified roster need to respond to this PDU. File APEs may need to wait to receive this roster report indication before they can generate a reply.
FILE-TRANSMIT Token ID	This identifies the token used to ensure that no more than one File APE can attempt to fulfil the file request. Its inclusion is mandatory.
FILE-REQUEST Token ID	This identifies the token used to regulate the issuing of file requests, so avoiding the need for other File ASEs to consult the registry. If the token is omitted, only the creator of the control channel is permitted to issue File-Request PDUs. If a FILE-REQUEST token has been assigned to the control channel it must be included.
Data Offset	This parameter specifies the starting offset in octets from the beginning of the file data (so zero denotes the origin).

The File-RequestPDU includes the identity of the data channel to be used so that the requesting File ASE can specify that:

- the file is to be sent exclusively to the requester on its File APE MCS User channel; or
- the file is to be broadcast to all File ASEs in the session on the broadcast data channel. In this case, all File ASEs are obliged to receive the data and discard it if it is not required; or
- the file is to be offered to all File ASEs in the session on an acknowledged data channel. In this case, File ASEs may choose whether they wish to receive the data.

So that the File ASE which issued the File-Request knows how many acknowledgements to accept, all File ASEs included in the Application Roster instance specified in the File-RequestPDU must respond, using the File-DenyPDU (see Table 16) if they are unable to supply the file. File ASEs not included in the specified Application Roster instance must ignore the File-Request.

Table 16 – File-DenyPDU

Parameter	Description
Request Handle	This is a unique handle used to reference the file request. It shall be set to the same value as the handle in the corresponding File-RequestPDU.
Reason	This parameter indicates to the requesting File APE why the file could not be provided.

A File ASE which is capable of providing the file must first grab the FILE-TRANSMIT token identified in the File-RequestPDU before offering the file for transmission on the data channel shown in the request, using File-Offer as if it had initiated the transaction. If the FILE-TRANSMIT token is not free, the File ASE shall respond with a File-Deny, specifying no Channel as the reason for refusal. The requesting File ASE may then make a further attempt, specifying a different data channel.

If all File ASEs respond with a File-Deny, then the requesting File ASE must relinquish the FILE-REQUEST token, otherwise the token is given up on receipt of the File-OfferPDU corresponding to the required file. Once a File ASE has fulfilled the request by sending a File-Offer, any sites which have not yet responded shall send a File-Deny, even if they are capable of sourcing the file. Further requests may be made by any File ASE on the control channel after the FILE-REQUEST token has been released.

The requesting File ASE should monitor the application roster to determine which File ASEs to expect responses from. It is recommended that a timeout be used to prevent the application being locked by sites which fail to respond. Any responses received after the timeout expires can be identified by the request handle.

8.10 Remote directory listing

This is an optional feature which allows a File ASE to obtain a directory listing of a remote site. It may be used in conjunction with the file request mechanism to provide a file retrieval service. A File ASE requiring a directory list issues a Directory-RequestPDU to the MBFT User Channel of the chosen remote site, using the parameters specified in Table 17.

Table 17 – Directory-RequestPDU

Parameter	Description
Pathname	This is the relative pathname of the directory for which a listing is required. If omitted or null, it is assumed that a listing of the default directory is requested.

A File ASE receiving a Directory-RequestPDU must respond by issuing a Directory-ResponsePDU to the originator's MBFT User Channel, using the parameters defined in Table 18. If the File ASE is able to fulfil the request, it includes a list of files and sub-directories contained in the directory identified in the Directory-RequestPDU. If it is unable or unwilling to provide a list, the File ASE should indicate the reason for refusal in the response. It should be noted that there is no requirement for the directory structure presented to remote sites to be the same as that presented locally.

Table 18 – Directory-ResponsePDU

Parameter	Description
Result	This indicates whether the File ASE is able to fulfil the request and, if not, gives the reason for refusal.
Pathname	This is the relative pathname of the directory being listed. If omitted or null, the listing is of the default directory.
Directory List	This parameter contains a list of files and sub-directories.

8.11 Conducted mode behaviour

MBFT is a conductor-aware application protocol, i.e., it has two modes of operation, one for non-conducted mode and one for conducted mode, as described in this subclause. The effect of conducted mode operation on file transactions is determined by the peer File ASE (i.e., a File ASE participating in the same MBFT session) at the node possessing the GCC-Conductor-Token if this ASE is capable of supporting the role of conductor. Each MBFT session has its own conductor, namely the File ASE at the conducting node participating in that session. If there is more than one File APE at the conducting node participating in a given MBFT session, then it is a local matter for

GCC at the conducting node to decide which File ASE adopts the role of conductor. When the conducting node does not have a peer File APE which supports the conductor role, the Node Controller at the conducting node proxies as the MBFT conductor. Support of MBFT conducted mode is mandatory. Note that whilst applications must be capable of operating in conducted mode, they do not have to support the MBFT conductor role.

Applications become aware of a GCC-Conference Conductor being assigned through receipt of a GCC-Conductor-Assign indication from the GCC provider. This includes the GCC User ID of the conducting node. Applications determine the identity of the conducting File ASE from the GCC Application Roster. No new file transactions (File-Offer or File-Request) are allowed after entering conducted mode until permission has been granted by the MBFT conductor. File transactions in progress when conducted mode is entered may continue to completion unless specifically aborted by use of the File-AbortPDU (see clause 8.12).

File ASEs at the conducting node may initiate transactions without taking any further action. Peer File ASEs at other sites must request permission to act from the MBFT conductor as detailed below. Note that File ASEs not wishing to initiate any file transactions (i.e., those which only wish to receive files offered to them) do not need to request permission from the MBFT conductor.

8.11.1 Peer File APE present at conducting node

Peer File ASEs at non-conducting nodes must request permission to initiate file transactions by sending an MBFT-Privilege-RequestPDU (Table 19) to the conducting node, specifying the privileges required. The following privileges are available:

- Privilege to source file transfers.
- Privilege to request file transfers.
- Privilege to create private channels.
- Privilege to use medium priority for file transfers.
- Privilege to issue File-AbortPDUs.
- Privilege to use non-standard extensions.

Table 19 – MBFT-Privilege-RequestPDU

Parameter	Description
File Transfer Privilege	This flag indicates whether the requesting File ASE wants permission to source file transfers.
File Request Privilege	This flag indicates whether the requesting File ASE wants permission to request files or retransmissions.
Private Channel Privilege	This flag indicates whether the requesting File ASE wants permission to create private channels.
Priority Privilege	This flag indicates whether the requesting File ASE wants permission to send medium priority File-DataPDUs.
Abort Privilege	This flag indicates whether the requesting File ASE wants permission to issue File-AbortPDUs.
Non-Standard Privilege	This flag indicates whether the requesting File ASE wants permission to use any negotiated non-standard extensions to the MBFT protocol.

On receipt of an MBFT-Privilege-RequestPDU, the MBFT conductor may respond by broadcasting an MBFT-Privilege-AssignPDU on the control channel. The PDU contains a list of privileges granted and the User ID of the requesting File ASE and enables File ASEs to determine which privileges have been assigned to their peers. The MBFT conductor does not need to send an MBFT-Privilege-AssignPDU if it does not wish to change the privileges of the requesting application. Note that the MBFT conductor may at any time revoke some or all of the privileges granted by issuing an MBFT-Privilege-AssignPDU with a revised privilege list for the designated node(s). Any transactions in progress may complete unless specifically aborted.

If the MBFT conductor receives a GCC-Application-Roster-Report indicating that new nodes have been added to the list of active File ASEs, it shall broadcast an MBFT-Privilege-AssignPDU on the control channel, specifying a list of File ASEs and their associated privileges (see Table 20). Entries in this list need only be made for those nodes which have one or more privileges assigned.

Table 20 – MBFT-Privilege-AssignPDU

Parameter	Description
Privilege List	This contains a list of one or more MBFT User IDs and the privileges assigned to them. These privileges are identified below.
File Transfer Privilege	This flag indicates whether the requesting File ASE has been granted permission to source file transfers.
File Request Privilege	This flag indicates whether the requesting File ASE has been granted permission to request files or retransmissions.
Private Channel Privilege	This flag indicates whether the requesting File ASE has been granted permission to create private channels.
Priority Privilege	This flag indicates whether the requesting File ASE has been granted permission to send medium priority File-DataPDUs. If this flag is set to FALSE, then only low priority can be used for file transfers.
Abort Privilege	This flag indicates whether the requesting File ASE has been granted permission to send File-AbortPDUs.
Non-Standard Privilege	This flag indicates whether the requesting File ASE has been granted permission to use any negotiated non-standard extensions to the MBFT protocol.

If there is a change in conductor without leaving conducted mode, all privileges are automatically revoked and File ASEs must apply to the new MBFT conductor to reacquire them. File ASEs shall consult the Application Roster to determine if there is a new MBFT Conductor for the session. If there is no MBFT conductor present at the new conducting node for that session, File ASEs must cease all transactions immediately. If there is an MBFT conductor present, any transactions in progress at the time of the change may continue unless specifically aborted by the new conductor.

If a File ASE receives a GCC-Conductor-Permission-Grant indication with the Permission flag set to TRUE it shall have all MBFT privileges granted. If the File ASE subsequently receives a GCC-Conductor-Permission-Grant indication with the Permission flag set to FALSE it shall revert to the privileges assigned to it by the MBFT Conductor.

If the File ASE receives a GCC-Application-Roster-Report indication in which the MBFT conductor is no longer present, all privileges are revoked and the File ASE shall cease all transactions immediately.

8.11.2 Peer File APE absent at conducting node

If there is no peer File APE at the conducting node capable of supporting the conductor role, a File ASE must direct requests to act, via its local Node Controller, to the Node Controller at the conducting node by issuing a GCC-Conductor-Permission-Ask request primitive. If permission is granted, then the File ASE will receive a GCC-Conductor-Permission-Grant indication with the permission flag set to TRUE. This enables the File ASE to act in an unrestricted manner (as in non-conducted mode) until further notice.

Permission to act may be revoked by the conducting node at any time. If the File ASE receives a GCC-Conductor-Permission-Grant indication with the permission flag set to FALSE, then it must cease all transactions with immediate effect. If the File ASE is transmitting a file it shall issue a final File-DataPDU containing no data and with the abort flag set to TRUE and relinquish the FILE-TRANSMIT token.

Note that a File ASE may receive an unsolicited successful GCC-Conductor-Permission-Grant indication in response to a GCC-Conductor-Permission-Grant request by its own Node Controller or another application protocol entity at that node or because the conductor decides to grant permission without having received a request.

8.12 Aborting a file transfer

If an individual File ASE is unable to receive a file or determines that the data is no longer required, it shall either disconnect from the data channel or continue to receive the incoming data and discard it.

In conducted mode, File ASEs may be granted permission by the MBFT conductor to issue File-AbortPDUs which can be used to order a transmitting File ASE to terminate a file transfer. After receiving a File-Abort PDU, the transmitter is obliged to set the abort flag to TRUE in the current File-DataPDU and then to cease transmission and relinquish the FILE-TRANSMIT token.

This mechanism can be used by the MBFT conductor to terminate any file transfers in progress on entering conducted mode.

The transmitter may cease transmission of its own volition by simply setting the abort flag in a File-DataPDU.

The File-AbortPDU includes a reason code to inform the transmitter why the abort request was generated. The transmission to be terminated can be selected by channel, MBFT User ID or file handle plus MBFT User ID. If no parameters are included, then transmissions on all channels (either static, multicast or private) by any File ASE must cease. If only the MBFT User ID entry is present, then the identified File ASE must cease transmission on all channels. If the MBFT User ID and file handle are present, then the specified file transaction must terminate. See Table 21.

Table 21 – File-AbortPDU

Parameter	Description
File Handle	This is a unique handle used to reference the file throughout the file transfer operation. Its value is obtained from the File-Offer PDU.
Data Channel ID	This identifies the channel on which file transfers must cease.
Transmitter User Channel ID	This identifies the File ASE being requested to cease file transmission.
Reason	This is used to inform the transmitter why it must terminate its current operation.

8.13 Diagnostics

Support is provided for exchange of diagnostics messages between sending and receiving sites by way of the File-ErrorPDU. The PDU contains three error parameters: an error type (indicating permanent error, transient error or informative message), an error identifier and an optional text message. The action to be taken on receipt of a File-ErrorPDU is not defined by this protocol. See Table 22.

Table 22 – File-ErrorPDU

Parameter	Description
File Handle	This is a unique handle used to reference the file throughout the file transfer operation and identifies the file affected. Its value is obtained from the File-OfferPDU.
Error type	This field provides an indication of the severity of the fault condition. Faults may be either permanent (causing all transactions to fail), transient (causing failure of the current transmission) or informative (current transaction completed successfully).
Error ID	This parameter is a fault code to identify the nature of the problem.
Error Text	This optional field allows File ASEs to convey a textual description of the fault condition to the transmitting File ASE.

8.14 Non-standard operations

This PDU allows any non-standard information to be transmitted, enabling applications to implement non-standard operations. Use of non-standard operations may be subject to negotiation of associated non-standard capabilities. MBFT-NonStandardPDUs may be sent at any time, but in conducted mode may only be sent by File APEs which have been granted the Non-Standard Privilege. See Table 23.

Table 23 – MBFT-NonStandardPDU

Parameter	Description
Data	Non-standard parameter. Use of this PDU is not defined in this Recommendation.

9 MBFT PDU Definitions

The structure of MBFT PDUs is specified as follows using the notation ASN.1 of [ITU-T X.680]. All MBFT PDUs shall be encoded for transmission by applying the Packed Encoding Rules of [ITU-T X.691] using the Basic Aligned variant. MBFT PDUs are encoded and placed in the data field of either MCS-Send-Data or MCS-Uniform-Send-Data primitives. The bit string generated by the ASN.1 encoding is placed in the OCTET STRING used by MCS in the order such that for each octet, the leading bit is placed in the most significant bit position and the trailing bit is placed in the least significant bit position.

```
MBFT-PROTOCOL {itu-t(0) recommendation(0) t(20) t127(127) version(0) 2
asn1Modules(2) mBFT-PROTOCOL(1)} DEFINITIONS AUTOMATIC TAGS ::=
BEGIN
IMPORTS  ChannelID,
     StaticChannelID,
     DynamicChannelID,
     UserID,
     TokenID,
     StaticTokenID,
     DynamicTokenID,
     Key,
     H221NonStandardIdentifier,
     NonStandardParameter,
     TextString
FROM GCC-PROTOCOL;
-- Export all symbols
-- Part 1: Message Components
Handle                  ::= INTEGER (0..65535)   -- 16-bit value
```

```
Contents-Type-Attribute ::= CHOICE
{
-- See Annex B/ISO 8571-2 for more information
       document-type [0] SEQUENCE
           {
           document-type-name  [1] Document-Type-Name,
           parameter [0] ISO-8571-2-Parameters OPTIONAL
           -- The actual types to be used for values of the parameter
           -- field are defined in the document-type-name.
           -- Currently, only UNSTRUCTURED TEXT and UNSTRUCTURED BINARY
           -- are supported.
           }
}

Document-Type-Name ::= OBJECT IDENTIFIER
ISO-8571-2-Parameters ::= SEQUENCE
{
        universal-class-number  [0] INTEGER OPTIONAL,
        maximum-string-length   [1] INTEGER OPTIONAL,
        string-significance     [2] INTEGER
        {variable (0), fixed (1), not-significant (2)} OPTIONAL
}
Entity-Reference ::= INTEGER
{
   no-categorisation-possible (0),
   initiating-file-service-user  (1),
   initiating-file-protocol-machine (2),
   service-supporting-the-file-protocol-machine   (3),
   responding-file-protocol-machine (4),
   responding-file-service-user  (5)
}
Filename-Attribute ::= SEQUENCE OF GraphicString
Access-Control-Attribute ::= CHOICE
{
   simple-password  [0] OCTET STRING,
         -- A simplified form of the access control syntax. Specifies
         -- one password for all types of access to the files and its
         -- attributes
   actual-values    [1] SET OF Access-Control-Element
         -- The semantics of this attribute are described in ISO 8571-2
}
Access-Control-Element ::= SEQUENCE
{
   action-list         [0] Access-Request,
   concurrency-access  [1] Concurrency-Access OPTIONAL,
   identity            [2] User-Identity OPTIONAL,
   passwords           [3] Access-Passwords OPTIONAL,
   ...
}
Access-Request ::= BIT STRING
{
   read             (0),
   insert           (1),
   replace          (2),
   extend           (3),
   erase            (4),
   read-attribute   (5),
   change-attribute (6),
   delete-file      (7)
}
Concurrency-Access ::= SEQUENCE
{
   read             [0] Concurrency-Key,
   insert           [1] Concurrency-Key,
   replace          [2] Concurrency-Key,
   extend           [3] Concurrency-Key,
   erase            [4] Concurrency-Key,
   read-attribute   [5] Concurrency-Key,
   change-attribute [6] Concurrency-Key,
   delete-file      [7] Concurrency-Key
}
Access-Passwords ::= SEQUENCE
{
   read-password    [0] Password,
   insert-password  [1] Password,
   replace-password [2] Password,
   extend-password  [3] Password,
   erase-password   [4] Password,
   read-attribute-password [5] Password,
   change-attribute-password [6] Password,
   delete-password  [7] Password
}
Password ::= CHOICE
{
   graphic-string   GraphicString,
   octet-string     OCTET STRING
}
Concurrency-Key ::= BIT STRING
{
   not-required     (0),
   shared           (1),
   exclusive        (2),
   no-access        (3)
}
Permitted-Actions-Attribute ::= BIT STRING
{
   -- Actions available
   read             (0),
   insert           (1),
   replace          (2),
   extend           (3),
   erase            (4)
}
Private-Use-Attribute ::= SEQUENCE
{
   manufacturer-values [0] EXTERNAL OPTIONAL
}
Protocol-Version ::= BIT STRING {version-1 (0)}
User-Identity ::= GraphicString
FileHeader ::= SEQUENCE
{
   protocol-version   [28] Protocol-Version DEFAULT {version-1},-- T.434 version
   filename           [0] Filename-Attribute OPTIONAL,
   permitted-actions  [1] Permitted-Actions-Attribute OPTIONAL,
   contents-type      [2] Contents-Type-Attribute OPTIONAL,
   -- DEFAULT {UNSTRUCTURED BINARY}
   -- not specifying this attribute implies that the data content of
   -- the file is unstructured binary
   storage-account            [3] GraphicString OPTIONAL,
   date-and-time-of-creation  [4] GeneralizedTime OPTIONAL,
   date-and-time-of-last-modification   [5] GeneralizedTime OPTIONAL,
   date-and-time-of-last-read-access    [6] GeneralizedTime OPTIONAL,
   -- 7 is reserved for date-and-time-of-last-attribute-modification
   identity-of-creator        [8] GraphicString OPTIONAL,
   identity-of-last-modifier  [9] GraphicString OPTIONAL,
   identity-of-last-reader    [10] GraphicString OPTIONAL,
   -- 11 is reserved for identity-of-last-attribute-modifier
   -- 12 is reserved for file-availability
   filesize            [13] INTEGER OPTIONAL,
   future-filesize     [14] INTEGER OPTIONAL,
   access-control      [15] Access-Control-Attribute OPTIONAL,
   -- the use of this attribute is for further study
   legal-qualifications     [16] GraphicString OPTIONAL,
   private-use         [17] Private-Use-Attribute OPTIONAL,
   structure           [18] OBJECT IDENTIFIER OPTIONAL,
   application-reference    [19] SEQUENCE OF GraphicString OPTIONAL,
   machine             [20] SEQUENCE OF GraphicString OPTIONAL,
   operating-system    [21] OBJECT IDENTIFIER OPTIONAL,
   recipient           [22] SEQUENCE OF GraphicString OPTIONAL,
   character-set       [23] OBJECT IDENTIFIER OPTIONAL,
   compression         [24] SEQUENCE OF GraphicString OPTIONAL,
   -- indicates an optional compression applied to the content
   -- octets of the file
   environment         [25] SEQUENCE OF GraphicString OPTIONAL,
   pathname            [26] SEQUENCE OF GraphicString OPTIONAL,
   user-visible-string      [29] SEQUENCE OF GraphicString OPTIONAL,
   ...
}
CompressionSpecifier ::= CHOICE
{
      v42bis-parameters   V42bis-Parameter-List,
      nonstandard-compression-parameters   SET OF NonStandardParameter,
      ...
}
V42bis-Parameter-List ::= SEQUENCE
{
      p1       INTEGER (512..65535),
      p2       INTEGER (6..250),
      ...
}
MBFTPrivilege ::= ENUMERATED
{
      file-transmit-privilege    (0),   -- Privilege to transmit files
      file-request-privilege     (1),   -- Privilege to request files
      create-private-privilege   (2),   -- Privilege to create private channels
      medium-priority-privilege  (3),   -- Privilege to use medium priority for
                                        -- file transfers
      abort-privilege            (4),   -- Privilege to issue File-AbortPDUs
      nonstandard-privilege      (5),   -- Privilege to use non-standard options
                                        -- subject to negotiation
      ...
}
DirectoryEntry ::= SEQUENCE
{
      subdirectory-flag   BOOLEAN,  -- TRUE for sub-directory entries
                                    -- FALSE for file entries
      attributes          FileHeader,
      ...
}
ErrorType ::= ENUMERATED
{
      informative      (0),   -- No recovery required
      transient-error  (1),   -- Current transfer failed,
                              -- future transfers may be OK
      permanent-error  (2),   -- Error occurs each time file
                              -- transfer is performed
      ...
}
ErrorID ::= INTEGER
```

```
{
       no-reason        (0),
       responder-error  (1),
       system-shutdown  (2),
       bft-management-problem (3),
       bft-management-bad-account (4),
       bft-management-security-not-passed  (5),
       delay-may-be-encountered  (6),
       initiator-error  (7),
       subsequent-error (8),
       temporal-insufficiency-of-resources (9),
       access-request-violates-VFS-security (10),
       access-request-violates-local-security  (11),
       conflicting-parameter-values  (1000),
       unsupported-parameter-values  (1001),
       mandatory-parameter-not-set   (1002),
       unsupported-parameter  (1003),
       duplicated-parameter   (1004),
       illegal-parameter-type (1005),
       unsupported-parameter-types   (1006),
       bft-protocol-error  (1007),
       bft-protocol-error-procedure-error  (1008),
       bft-protocol-error-functional-unit-error   (1009),
       bft-protocol-error-corruption-error (1010),
       lower-layer-failure (1011),
       timeout      (1013),
       invalid-filestore-password (2020),
       filename-not-found  (3000),
       initial-attributes-not-possible (3002),
       non-existent-file   (3004),
       file-already-exists (3005),
       file-cannot-be-created (3006),
       file-busy    (3012),
       file-not-available  (3013),
       filename-truncated  (3017),
       initial-attributes-altered (3018),
       bad-account  (3019),
       ambiguous-file-specification  (3024),
       attribute-non-existent (4000),
       attribute-not-supported (4003),
       bad-attribute-name  (4004),
       bad-attribute-value (4005),
       attribute-partially-supported (4006),
       bad-data-element-type  (5014),
       operation-not-available (5015),
       operation-not-supported (5016),
       operation-inconsistent (5017),
       bad-write    (5026),
       bad-read     (5027),
       local-failure (5028),
       local-failure-filespace-exhausted (5029),
       local-failure-data-corrupted  (5030),
       local-failure-device-failure  (5031),
       future-filesize-exceeded  (5032),
       future-filesize-increased (5034)

}
-- Part 2: PDU Messages
File-OfferPDU ::= SEQUENCE
{
       file-header      FileHeader,
       data-channel-id  ChannelID,
       file-handle      Handle,
       roster-instance  INTEGER (0..65535) OPTIONAL,
       file-transmit-token TokenID OPTIONAL,
       file-request-token  TokenID OPTIONAL,
       file-request-handle Handle OPTIONAL,
       mbftID              UserID OPTIONAL,
       compression-specifier CompressionSpecifier OPTIONAL,
       compressed-filesize INTEGER OPTIONAL,
       ack-flag            BOOLEAN,  -- True if acknowledgements required
       ...
}
File-AcceptPDU ::= SEQUENCE
{
       file-handle  Handle,
       ...
}

File-RejectPDU ::= SEQUENCE
{
       file-handle  Handle,
       reason       ENUMERATED
       {
          unspecified   (0),
          file-exists   (1),
          file-not-required(2),
          insufficient-resources  (3),
          transfer-limit   (4),   -- maximum no. of concurrent
                                  -- file transfers exceeded
          compression-unsupported (5),   -- algorithm identified
                                         -- in FileOffer not supported
          unable-to-join-channel (6),
          parameter-not-supported (7),   -- at least one File Header
                                         -- parameter is not supported
          ...
       },
       ...
}
File-RequestPDU ::= SEQUENCE
{
       file-header          FileHeader,
       data-channel-id      ChannelID,
       request-handle       Handle,
       roster-instance      INTEGER (0..65535),
       file-transmit-token  TokenID,
       file-request-token   TokenID OPTIONAL,
       data-offset          INTEGER,
       ...
}
File-DenyPDU ::= SEQUENCE
{
       request-handle  Handle,
       reason          ENUMERATED
       {
          unspecified   (0),
          file-not-present (1),
          insufficient-privilege  (2),
          file-already-offered(3),  -- File already being offered
                                    -- by another site
          ambiguous  (4),   -- Insufficient attributes to uniquely identify file
          no-channel (5),   -- No data channel available to fulfil the request
          ...
       },
     ...
}
```

```
File-AbortPDU ::= SEQUENCE
{
      reason                 ENUMERATED
      {
         unspecified         (0),
         bandwidth-required  (1),
         tokens-required     (2),
         channels-required   (3),
         priority-required   (4),
         ...
      },

      data-channel-id        ChannelID OPTIONAL,
      transmitter-user-id    UserID OPTIONAL,
      file-handle            Handle OPTIONAL,
      ...
}

File-StartPDU ::= SEQUENCE
{
      file-header  FileHeader,
      file-handle  Handle,
      eof-flag     BOOLEAN,  -- True if last packet of data
      crc-flag     BOOLEAN,  -- True if CRC present
      compression-specifier  CompressionSpecifier OPTIONAL,
      comp-filesize INTEGER OPTIONAL,
      data-offset  INTEGER,
      data         OCTET STRING (SIZE (0..65535)),
      crc-check    INTEGER (0..4294967295) OPTIONAL,
      ...
}
File-DataPDU ::= SEQUENCE
{
      file-handle  Handle,
      eof-flag     BOOLEAN,  -- True if last packet of data
      abort-flag   BOOLEAN,  -- True if file transfer is being aborted
      data         OCTET STRING (SIZE (0..65535)),
      crc-check    INTEGER (0..4294967295) OPTIONAL,
      ...
}
Directory-RequestPDU ::= SEQUENCE
{
      pathname     SEQUENCE OF GraphicString OPTIONAL,
      ...
}
Directory-ResponsePDU ::= SEQUENCE
{
      result       ENUMERATED
      {
         unspecified   (0),
         permission-denied(1),
         function-not-supported  (2),
         successful (3),
         ...
      },
      pathname     SEQUENCE OF GraphicString OPTIONAL,
      directory-list   SEQUENCE OF DirectoryEntry,
      ...
}
MBFT-Privilege-RequestPDU ::= SEQUENCE
{
      mbft-privilege   SET OF MBFTPrivilege,
      ...
}
```

```
MBFT-Privilege-AssignPDU ::= SEQUENCE
{
      privilege-list     SET OF SEQUENCE  -- One for each File APE
                                          -- with privileges
      {
         mbftID          UserID,
         mbft-privilege  SET OF MBFTPrivilege,
         ...
      },
      ...
}
Private-Channel-Join-InvitePDU ::= SEQUENCE
{
      control-channel-id  DynamicChannelID,
      data-channel-id     DynamicChannelID,
      mode                BOOLEAN,  -- True if broadcast
      ...
}

Private-Channel-Join-ResponsePDU ::= SEQUENCE
{
      control-channel-id   DynamicChannelID,
      result               ENUMERATED
      {
         unspecified          (0),
         unable-to-join-channel  (1),
         invitation-rejected     (2),
         successful           (3),
         ...
      },
      ...
}
File-ErrorPDU ::= SEQUENCE
{
      file-handle   Handle OPTIONAL,
      error-type    ErrorType,
      error-id      ErrorID,
      error-text    TextString OPTIONAL,
      ...
}
MBFT-NonStandardPDU ::= SEQUENCE
{
      data   NonStandardParameter,
      ...
}
-- Part 3: Messages sent using MCS-Send-Data or MCS-Uniform-Send-Data
MBFTPDU ::= CHOICE
{
      file-OfferPDU File-OfferPDU,
      file-AcceptPDU   File-AcceptPDU,
      file-RejectPDU   File-RejectPDU,
      file-RequestPDU  File-RequestPDU,
      file-DenyPDU  File-DenyPDU,
      file-ErrorPDU File-ErrorPDU,
      file-AbortPDU File-AbortPDU,
      file-StartPDU File-StartPDU,
      file-DataPDU  File-DataPDU,
      directory-RequestPDU    Directory-RequestPDU,
      directory-ResponsePDU   Directory-ResponsePDU,
      mbft-Privilege-RequestPDU  MBFT-Privilege-RequestPDU,
      mbft-Privilege-AssignPDU   MBFT-Privilege-AssignPDU,
      mbft-NonStandardPDU MBFT-NonStandardPDU,
        private-Channel-Join-InvitePDU   Private-Channel-Join-InvitePDU,
        private-Channel-Join-ResponsePDU Private-Channel-Join-ResponsePDU,
        ...
}
END
```

10 Use of the Multipoint Communication Service

All MBFT communication shall be through MCS as specified in [ITU-T T.122]. This clause details specific use of MCS services, channel allocation, token allocation and data priorities. Static tokens and channels assigned to MBFT are assigned in [ITU-T T.120].

10.1 Use of MCS data transmission services

Either Send-Data or Uniform-Send-Data may be used, at the discretion of the transmitter. Receivers must be capable of accepting both types of data. The channel and priority to be used for each PDU is defined in Table 25.

10.2 Channel allocation

In each MBFT session, two channels are provided to allow broadcasting of one file at a time. The control channel MBFT-CONTROL is used to manage all aspects of file transfer on the broadcast data channel MBFT-DATA. Each File APE also joins the User-ID channel allocated to it by MCS. Transfer of more than one file simultaneously within the same session requires acknowledged data channels [designated MBFT-DATA(n), where n is the MCS channel ID of the data channel], one per concurrent file transfer. The way in which channel ID values for MBFT-CONTROL and MBFT-DATA are determined depends on a session's mode of operation (static, multicast or private), as explained in clause 8.4.5. The method of determining channel IDs for MBFT-DATA(n) is dependent on whether multicast or private channels are used and is described in clauses 8.6.1 and 8.6.2 respectively.

A sub-session has a control channel MBFT-CONTROL(p) (where p is the MCS channel ID of the control channel). It may optionally have a single broadcast data channel MBFT-DATA(p) (where p is the MCS channel ID of the data channel) and zero or more acknowledged data channels MBFT-DATA(n). Channel ID values for MBFT-CONTROL(p) and MBFT-DATA(p) are determined by the process described in clause 8.7. See Table 24.

Table 24 – MBFT channel description

Mnemonic	Description
MBFT-CONTROL	Control channel for communication of file exchange management information between all File APEs within a session. File APEs must join and remain joined to this channel for the duration of the session.
MBFT-DATA	Broadcast data channel for distribution of file data to all File APEs within a session. File APEs must join and remain joined to this channel for the duration of the session.
MBFT-CONTROL(p) [e.g., MBFT-CONTROL(1001)]	Private control channel whose MCS Channel ID is (p). This channel is used for communication of file exchange management information between all File APEs within a sub-session. File APEs must join and remain joined to this channel for the duration of the sub-session.
MBFT-DATA(p) [e.g., MBFT-DATA(1002)]	Private broadcast data channel whose MCS Channel ID is (p). This channel is used for distribution of file data to all File APEs within a sub-session. The presence of a broadcast data channel within a sub-session is at the discretion of the convenor of the sub-session. If present, File APEs must join and remain joined to this channel for the duration of the sub-session.

Table 24 – MBFT channel description

Mnemonic	Description
MBFT-DATA(n) [e.g., MBFT-DATA(1003)]	Acknowledged data channel whose MCS Channel Id is (n). This channel is used for distribution of file data to File APEs within a session or sub-session. File APEs need only join the channel to receive files they require.
MBFT User ID channel	For communication from any File APE in a conference to a specific File APE at a particular node. Each File APE is identified by its MCS User ID.

Table 25 – MBFT channel usage

MBFT PDU	Channel	Priority
File-OfferPDU	MBFT-CONTROL MBFT-CONTROL(p)	High
File-AcceptPDU	MBFT User ID	High
File-RejectPDU	MBFT User ID	High
File-RequestPDU	MBFT-CONTROL MBFT-CONTROL(p)	High
File-DenyPDU	MBFT User ID	High
File-ErrorPDU	MBFT User ID	High
File-AbortPDU	MBFT-CONTROL MBFT-CONTROL(p)	High

File-DataPDU	MBFT-DATA
	MBFT-DATA(p)
	MBFT-DATA(n)
	MBFT User ID
		Medium, Low

File-StartPDU	MBFT-DATA
	MBFT-DATA(p)
	MBFT-DATA(n)
	MBFT User ID
		Medium, Low

Directory-RequestPDU	MBFT User ID	High
Directory-ResponsePDU	MBFT User ID	High
MBFT-Privilege-RequestPDU	Conductor MBFT User ID	High
MBFT-Privilege-AssignPDU	MBFT-CONTROL	High
Private-Channel-Join-InvitePDU	MBFT User ID	High
Private-Channel-Join-ResponsePDU	Channel Manager User ID	High
MBFT-NonStandardPDU	Not specified	Not specified

10.3 Token allocation

MBFT reserves two static tokens for its exclusive use in static mode: FILE-TRANSMIT and FILE-REQUEST. Dynamic equivalents of these tokens may be used for multicast and private modes.

The session broadcast data channel may have a FILE-TRANSMIT token to ensure that there is only one node transmitting data on the channel at a time. A File ASE intending to transmit a file must grab this token before offering the file to other participants using the File-OfferPDU. Once it has completed the file transmission (i.e., sent a File-StartPDU or File-DataPDU with the off Flag set to

TRUE), the transmitter should release the FILE-TRANSMIT token to allow other nodes to transmit on the data channel).

The session control channel may have a FILE-REQUEST token to ensure that there is, at most, one outstanding file request to be processed at any instance. A File ASE requesting a file must grab the FILE-REQUEST token before issuing a File-RequestPDU. It holds the token until it can determine whether another File ASE is able to fulfil the request, i.e., either one File ASE responds with a File-Offer, all File ASEs respond with a File-Deny, or a timeout expires. Once the FILE-REQUEST token is released, other File ASEs may attempt to request files by the same process.

Each acknowledged data channel may have a FILE-TRANSMIT token associated with it. The FILE-TRANSMIT token associated with an acknowledged data channel whose MCS Channel id is n [MBFT-DATA(n)] is assigned the mnemonic FILE-TRANSMIT(n) [e.g., FILE-TRANSMIT(1006)]. It is functionally identical to the FILE-TRANSMIT token associated with the broadcast data channel.

A sub-session optionally has a FILE-REQUEST(p) token associated with its MBFT-CONTROL(p) control channel. The FILE-REQUEST(p) token is functionally identical to the FILE-REQUEST token. If the sub-session has a broadcast data channel MBFT-DATA(p), this may optionally have a FILE-TRANSMIT(p) token associated with it. See Table 26.

Table 26 – MBFT token description

Mnemonic	Description
FILE-REQUEST	Token to ensure that there is at most one outstanding file request on the session control channel MBFT-CONTROL.
FILE-TRANSMIT	Token to ensure transmission of one file at a time on the session broadcast data channel MBFT-DATA.
FILE-REQUEST(p) [e.g., FILE-REQUEST(1001)]	Token to ensure that there is at most one outstanding file request on sub-session control channel MBFT-CONTROL(p).
FILE-TRANSMIT(p) [e.g., FILE-TRANSMIT(1002)]	Token to ensure transmission of one file at a time on sub-session broadcast data channel MBFT-DATA(p).
FILE-TRANSMIT(n) [e.g., FILE-TRANSMIT(1003)]	Token to ensure transmission of one file at a time on acknowledged data channel MBFT-DATA(n).

10.4 MCS services

MBFT assumes the MCS services indicated in Table 27. All primitives marked with an "M" are mandatory, those marked with an "O" are optional.

Table 27 – MCS services used by MBFT

Primitives	Use
MCS-Attach-User request	M
MCS-Attach-User confirm	M
MCS-Detach-User request	M
MCS-Detach-User indication	M
MCS-Channel-Join request	M
MCS-Channel-Join confirm	M
MCS-Channel-Leave request	M
MCS-Channel-Leave indication	M

Table 27 – MCS services used by MBFT

Primitives	Use
MCS-Channel-Convene request	O
MCS-Channel-Convene confirm	O
MCS-Channel-Disband request	O
MCS-Channel-Disband indication	M
MCS-Channel-Admit request	O
MCS-Channel-Admit indication	M
MCS-Channel-Expel request	O
MCS-Channel-Expel indication	M
MCS-Send-Data request	M
MCS-Send-Data indication	M
MCS-Uniform-Send-Data request	M
MCS-Uniform-Send-Data indication	M
MCS-Token-Grab request	M
MCS-Token-Grab confirm	M
MCS-Token-Give request	O
MCS-Token-Give indication	M
MCS-Token-Give response	M
MCS-Token-Give confirm	O
MCS-Token-Please request	O
MCS-Token-Please indication	M
MCS-Token-Release request	O
MCS-Token-Release confirm	O
MCS-Token-Release indication	M
MCS-Token-Test request	O
MCS-Token-Test confirm	O

11 Use of Generic Conference Control

MBFT assumes the services listed in Table 28. Primitives endorsed with an "M" are mandatory, those marked with an "O" are optional.

Table 28 – GCC primitives supported by MBFT

GCC primitive	Use
GCC-Application-Permission-To-Enroll indication	M
GCC-Application-Enroll request GCC-Application-Enroll confirm	M M
GCC-Application-Roster-Report indication	M
GCC-Registry-Retrieve-Entry request GCC-Registry-Retrieve-Entry confirm	O O
GCC-Registry-Register-Channel request GCC-Registry-Register-Channel confirm	O O
GCC-Registry-Assign-Token request GCC-Registry-Assign-Token confirm	O O
GCC-Conductor-Assign indication	M
GCC-Conductor-Release indication	M
GCC-Conductor-Permission-Ask request	O
GCC-Conductor-Permission-Grant request GCC-Conductor-Permission-Grant indication	O M

11.1 Resource IDs

Tables 29 and 30 define the construction of MBFT Resource IDs required for use of the GCC Registry. The registry is not required for the session MBFT-CONTROL channel as the MCS Channel ID can be obtained directly from the Session ID. It is also not required for private sub-session MBFT-CONTROL(p) channels as the MBFT-CONTROL(p) channel ID is conveyed to the selected participants by the private sub-session convenor. The registry may be used to identify the session broadcast data channel using the Resource ID "D0". Acknowledged multicast data channels are not placed in the registry, since the presence of a Registry entry does not guarantee the existence of the channel (see clause 8.6.2). Private data channels are not placed in the Registry since File ASEs are not permitted to join such channels until they receive an MCS-Channel-Admit from the channel convenor.

FILE-TRANSMIT(n) and FILE-TRANSMIT(p) Token IDs are constructed by expressing the MCS Channel ID of the associated MBFT-DATA(n) or MBFT-DATA(p) channel as a decimal numeric string without leading zeroes, prepending the single character "T", and encoding these characters into successive octets according to ITU-T Rec. T.50. FILE-REQUEST(p) Token Ids are constructed by expressing the MCS Channel ID of the associated MBFT-CONTROL(p) channel as a decimal numeric string without leading zeroes, prepending the single character "R", and encoding these characters into successive octets according to ITU-T Rec. T.50.

Table 29 – Channel determination

Mnemonic	Channel ID of static channels	Application registry resource ID for dynamic channels
MBFT-CONTROL	MBFT-CHANNEL-0	–
MBFT-DATA	MBFT-CHANNEL-1	D0
MBFT-CONTROL(p)	–	–
MBFT-DATA(p)	–	–
MBFT-DATA(n)	–	–

Table 30 – Token determination

Mnemonic	Token ID of Static Tokens	Application registry resource ID for dynamic tokens
FILE-REQUEST	MBFT-TOKEN-0	R0
FILE-TRANSMIT	MBFT-TOKEN-1	T0
FILE-REQUEST(p)	–	Rp (e.g., R1001)
FILE-TRANSMIT(p)	–	Tp (e.g., T1002)
FILE-TRANSMIT(n)	–	Tn (e.g., T1003)

Annex A

Static channel and token assignment

(This annex forms an integral part of this Recommendation)

The assignment of static resources (channels and tokens) is to be defined in [ITU-T T.120], but is included in this Recommendation pending the completion of [ITU-T T.120]. See Tables A.1 and A.2.

Table A.1 – Static channel ID assignments

Symbolic name	MCS Channel ID
MBFT-CHANNEL-0	9
MBFT-CHANNEL-1	10

Table A.2 – Static token ID assignments

Symbolic Name	MCS Token ID
MBFT-TOKEN-0	10
MBFT-TOKEN-1	11

Annex B

Object Identifier assignments

(This annex forms an integral part of this Recommendation)

Table B.1 lists the assignment of Object Identifiers defined for use by this Recommendation.

Table B.1 – T.127 object identifier assignment

Object Identifier Value	Description
{itu-t recommendation t 127 version(0) 2}	This Object Identifier is used to indicate the version of this Recommendation. At this time there is a single standardized version defined.

Appendix I

File Transfer Examples

(This appendix does not form an integral part of this Recommendation)

I.1 A number of examples of file transactions are given below to illustrate use of this Recommendation in different scenarios.

The following conventions are adopted in the figures:

Each PDU is followed by a Channel name enclosed in parentheses thus (). This indicates the channel on which the PDU is to be sent. Where appropriate, significant PDU parameters are appended in parentheses after the Channel ID.

Primitives are optionally followed by key parameters enclosed in square brackets thus [ ].

Note that the figures only denote transactions between a File APE, its peers and its local MCS and GCC providers. Transactions between the local MCS and GCC providers and the top MCS and GCC providers are omitted for clarity.

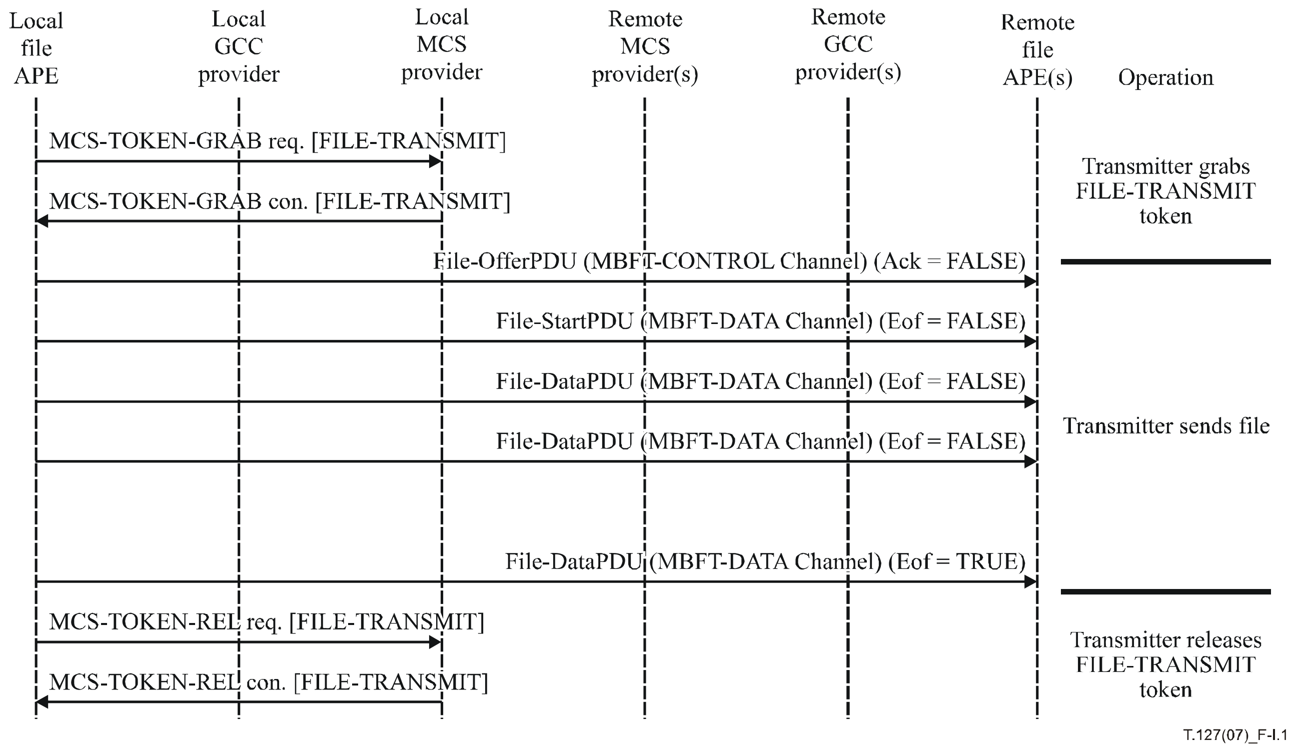
I.1.1 Figure I.1 illustrates distribution of a file on the broadcast data channel. All File APEs participating in the session are obliged to receive the data and discard it locally if it is not required.

I.1.2 Figure I.2 illustrates the creation and use of an acknowledged multicast data channel when at least one node wishes to receive the file being offered.

I.1.3 Figure I.3 illustrates an attempt to distribute a file on an acknowledged channel which is aborted because all receivers reject the file. It is assumed that the data channel and the associated FILE-TRANSMIT(n) token already exist.

I.1.4 Figure I.4 illustrates file distribution to a subset of the participants in the current session using a private sub-session. Participants have the option of rejecting files offered to them, and any site may attempt to request or offer files when the current transaction has finished.

I.1.5 Figure I.5 illustrates file distribution to a subset of the participants in the current session using a private sub-session. Participants have the option of rejecting files offered to them, but only the creator is permitted to transmit files. Note that the creator does not need to join the MBFT-DATA(n) channel, since it will never receive files.



Local file APE
Local GCC provider
Local MCS provider
Remote MCS provider(s)
Remote GCC provider(s)
Remote file APE(s)
Operation
MCS-TOKEN-GRAB req. [FILE-TRANSMIT]
MCS-TOKEN-GRAB con. [FILE-TRANSMIT]
Transmitter grabs FILE-TRANSMIT token
File-OfferPDU (MBFT-CONTROL Channel) (Ack = FALSE)
File-StartPDU (MBFT-DATA Channel) (Eof = FALSE)
File-DataPDU (MBFT-DATA Channel) (Eof = FALSE)
File-DataPDU (MBFT-DATA Channel) (Eof = FALSE)
Transmitter sends file
File-DataPDU (MBFT-DATA Channel) (Eof = TRUE)
MCS-TOKEN-REL req. [FILE-TRANSMIT]
MCS-TOKEN-REL con. [FILE-TRANSMIT]
Transmitter releases FILE-TRANSMIT token
T.127(07)_F-I.1

Figure I.1 – Broadcasting of files

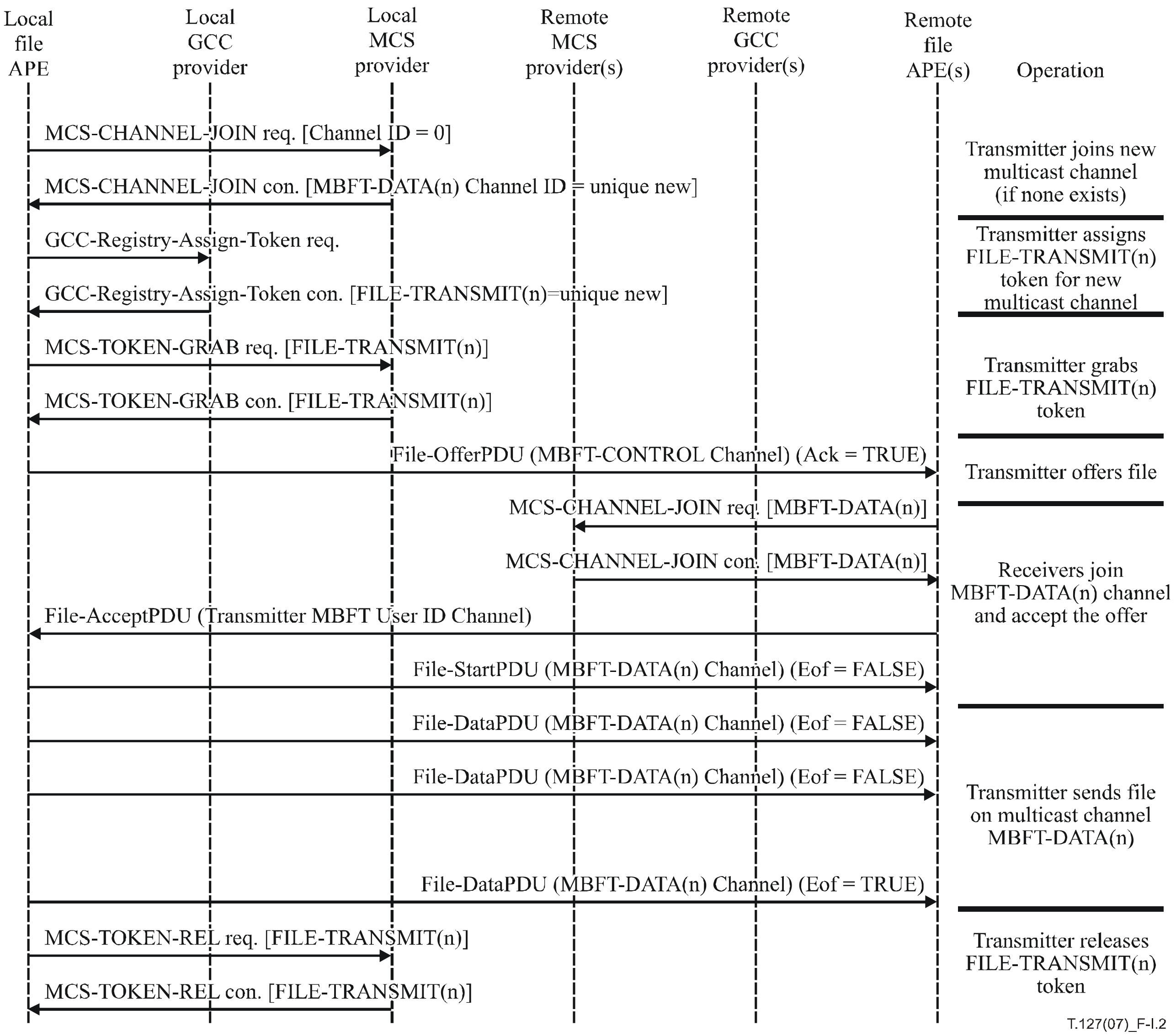

Local file APE
Local GCC provider
Local MCS provider
Remote MCS provider(s)
Remote GCC provider(s)
Remote file APE(s)
Operation
MCS-CHANNEL-JOIN req. [Channel ID = 0]
MCS-CHANNEL-JOIN con. [MBFT-DATA(n) Channel ID = unique new]
Transmitter joins new multicast channel (if none exists)
GCC-Registry-Assign-Token req.
GCC-Registry-Assign-Token con. [FILE-TRANSMIT(n)=unique new]
Transmitter assigns FILE-TRANSMIT(n) token for new multicast channel
MCS-TOKEN-GRAB req. [FILE-TRANSMIT(n)]
MCS-TOKEN-GRAB con. [FILE-TRANSMIT(n)]
Transmitter grabs FILE-TRANSMIT(n) token
File-OfferPDU (MBFT-CONTROL Channel) (Ack = TRUE)
Transmitter offers file
MCS-CHANNEL-JOIN req. [MBFT-DATA(n)]
MCS-CHANNEL-JOIN con. [MBFT-DATA(n)]
File-AcceptPDU (Transmitter MBFT User ID Channel)
Receivers join MBFT-DATA(n) channel and accept the offer
File-StartPDU (MBFT-DATA(n) Channel) (Eof = FALSE)
File-DataPDU (MBFT-DATA(n) Channel) (Eof = FALSE)
File-DataPDU (MBFT-DATA(n) Channel) (Eof = FALSE)
File-DataPDU (MBFT-DATA(n) Channel) (Eof = TRUE)
Transmitter sends file on multicast channel MBFT-DATA(n)
MCS-TOKEN-REL req. [FILE-TRANSMIT(n)]
MCS-TOKEN-REL con. [FILE-TRANSMIT(n)]
Transmitter releases FILE-TRANSMIT(n) token
T.127(07)_F-I.2

Figure I.2 – Successful acknowledged file transfer

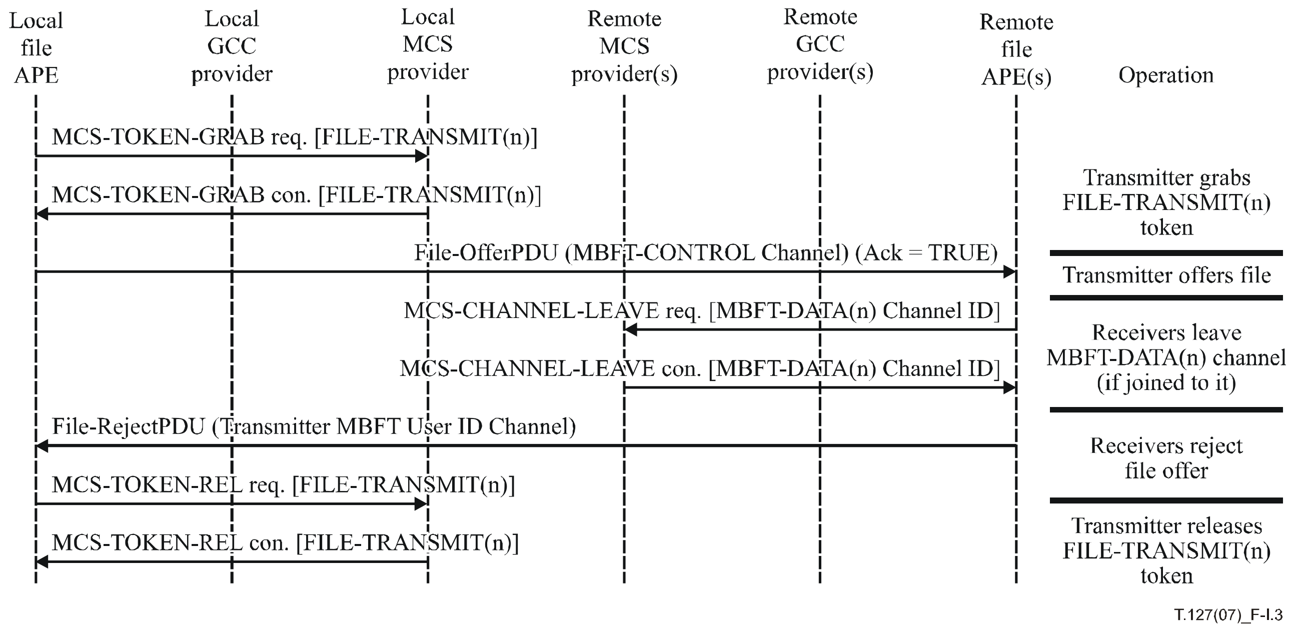

Local file APE
Local GCC provider
Local MCS provider
Remote MCS provider(s)
Remote GCC provider(s)
Remote file APE(s)
Operation
MCS-TOKEN-GRAB req. [FILE-TRANSMIT(n)]
MCS-TOKEN-GRAB con. [FILE-TRANSMIT(n)]
Transmitter grabs FILE-TRANSMIT(n) token
File-OfferPDU (MBFT-CONTROL Channel) (Ack = TRUE)
Transmitter offers file
MCS-CHANNEL-LEAVE req. [MBFT-DATA(n) Channel ID]
MCS-CHANNEL-LEAVE con. [MBFT-DATA(n) Channel ID]
Receivers leave MBFT-DATA(n) channel (if joined to it)
File-RejectPDU (Transmitter MBFT User ID Channel)
Receivers reject file offer
MCS-TOKEN-REL req. [FILE-TRANSMIT(n)]
MCS-TOKEN-REL con. [FILE-TRANSMIT(n)]
Transmitter releases FILE-TRANSMIT(n) token
T.127(07)_F-I.3

Figure I.3 – Rejected acknowledged file transfer

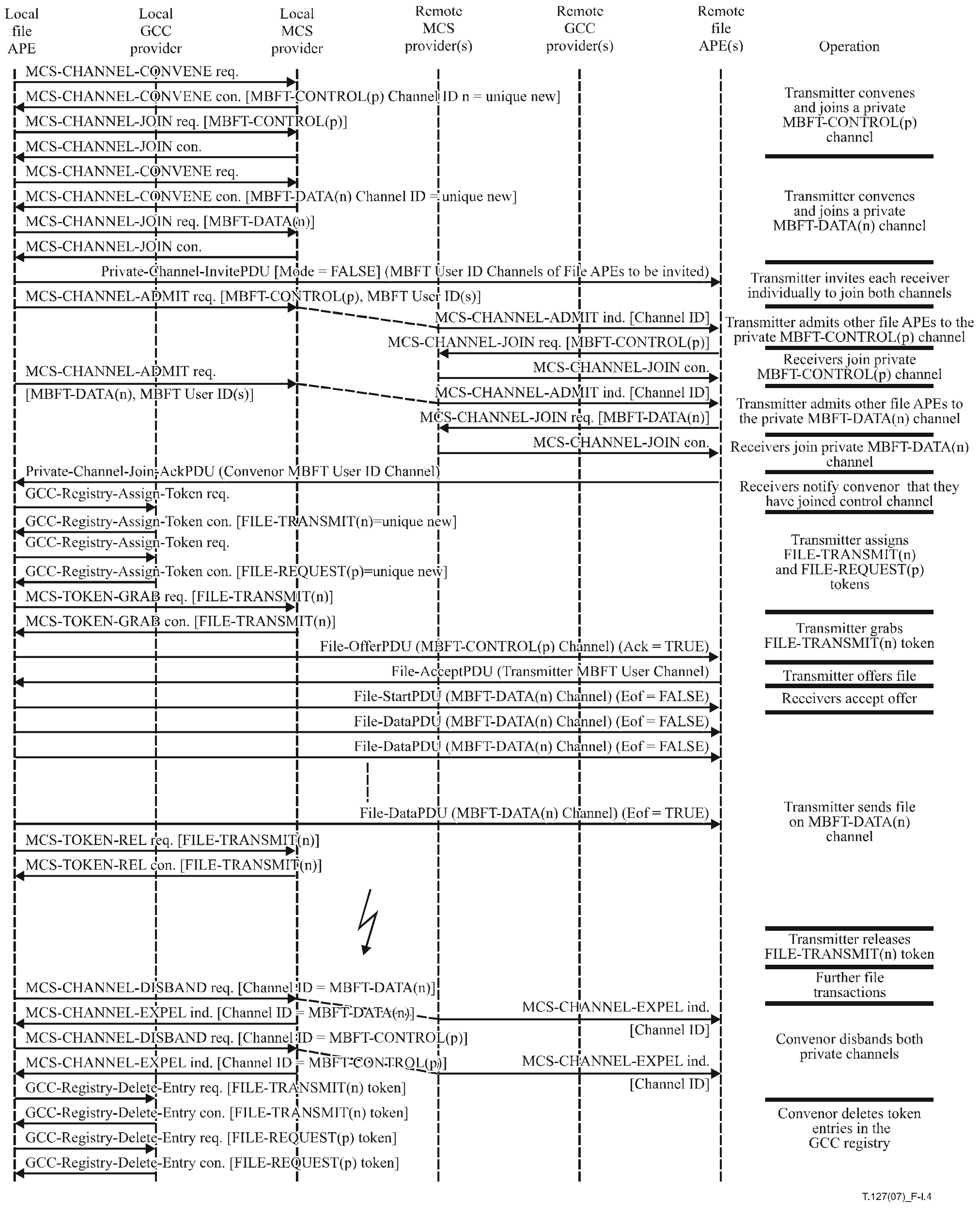

Local file APE
Local GCC provider
Local MCS provider
Remote MCS provider(s)
Remote GCC provider(s)
Remote file APE(s)
Operation
MCS-CHANNEL-CONVENE req.
MCS-CHANNEL-CONVENE con. [MBFT-CONTROL(p) Channel ID n = unique new]
MCS-CHANNEL-JOIN req. [MBFT-CONTROL(p)]
MCS-CHANNEL-JOIN con.
Transmitter convenes and joins a private MBFT-CONTROL(p) channel
MCS-CHANNEL-CONVENE req.
MCS-CHANNEL-CONVENE con. [MBFT-DATA(n) Channel ID = unique new]
MCS-CHANNEL-JOIN req. [MBFT-DATA(n)]
MCS-CHANNEL-JOIN con.
Transmitter convenes and joins a private MBFT-DATA(n) channel
Private-Channel-InvitePDU [Mode = FALSE] (MBFT User ID Channels of File APEs to be invited)
Transmitter invites each receiver individually to join both channels
MCS-CHANNEL-ADMIT req. [MBFT-CONTROL(p), MBFT User ID(s)]
MCS-CHANNEL-ADMIT ind. [Channel ID]
Transmitter admits other file APEs to the private MBFT-CONTROL(p) channel
MCS-CHANNEL-JOIN req. [MBFT-CONTROL(p)]
MCS-CHANNEL-JOIN con.
Receivers join private MBFT-CONTROL(p) channel
MCS-CHANNEL-ADMIT req.
[MBFT-DATA(n), MBFT User ID(s)]
MCS-CHANNEL-ADMIT ind. [Channel ID]
Transmitter admits other file APEs to the private MBFT-DATA(n) channel
MCS-CHANNEL-JOIN req. [MBFT-DATA(n)]
MCS-CHANNEL-JOIN con.
Receivers join private MBFT-DATA(n) channel
Private-Channel-Join-AckPDU (Convenor MBFT User ID Channel)
Receivers notify convenor that they have joined control channel
GCC-Registry-Assign-Token req.
GCC-Registry-Assign-Token con. [FILE-TRANSMIT(n)=unique new]
GCC-Registry-Assign-Token req.
GCC-Registry-Assign-Token con. [FILE-REQUEST(p)=unique new]
Transmitter assigns FILE-TRANSMIT(n) and FILE-REQUEST(p) tokens
MCS-TOKEN-GRAB req. [FILE-TRANSMIT(n)]
MCS-TOKEN-GRAB con. [FILE-TRANSMIT(n)]
Transmitter grabs FILE-TRANSMIT(n) token
File-OfferPDU (MBFT-CONTROL(p) Channel) (Ack = TRUE)
Transmitter offers file
File-AcceptPDU (Transmitter MBFT User Channel)
Receivers accept offer
File-StartPDU (MBFT-DATA(n) Channel) (Eof = FALSE)
File-DataPDU (MBFT-DATA(n) Channel) (Eof = FALSE)
File-DataPDU (MBFT-DATA(n) Channel) (Eof = FALSE)
File-DataPDU (MBFT-DATA(n) Channel) (Eof = TRUE)
Transmitter sends file on MBFT-DATA(n) channel
MCS-TOKEN-REL req. [FILE-TRANSMIT(n)]
MCS-TOKEN-REL con. [FILE-TRANSMIT(n)]
Transmitter releases FILE-TRANSMIT(n) token
Further file transactions
MCS-CHANNEL-DISBAND req. [Channel ID = MBFT-DATA(n)]
MCS-CHANNEL-EXPEL ind. [Channel ID = MBFT-DATA(n)]
MCS-CHANNEL-EXPEL ind.
[Channel ID]
MCS-CHANNEL-DISBAND req. [Channel ID = MBFT-CONTROL(p)]
MCS-CHANNEL-EXPEL ind. [Channel ID = MBFT-CONTROL(p)]
MCS-CHANNEL-EXPEL ind.
[Channel ID]
Convenor disbands both private channels
GCC-Registry-Delete-Entry req. [FILE-TRANSMIT(n) token]
GCC-Registry-Delete-Entry con. [FILE-TRANSMIT(n) token]
GCC-Registry-Delete-Entry req. [FILE-REQUEST(p) token]
GCC-Registry-Delete-Entry con. [FILE-REQUEST(p) token]
Convenor deletes token entries in the GCC registry
T.127(07)_F-I.4

Figure I.4 – Private sub-session file transfer using acknowledged data channel

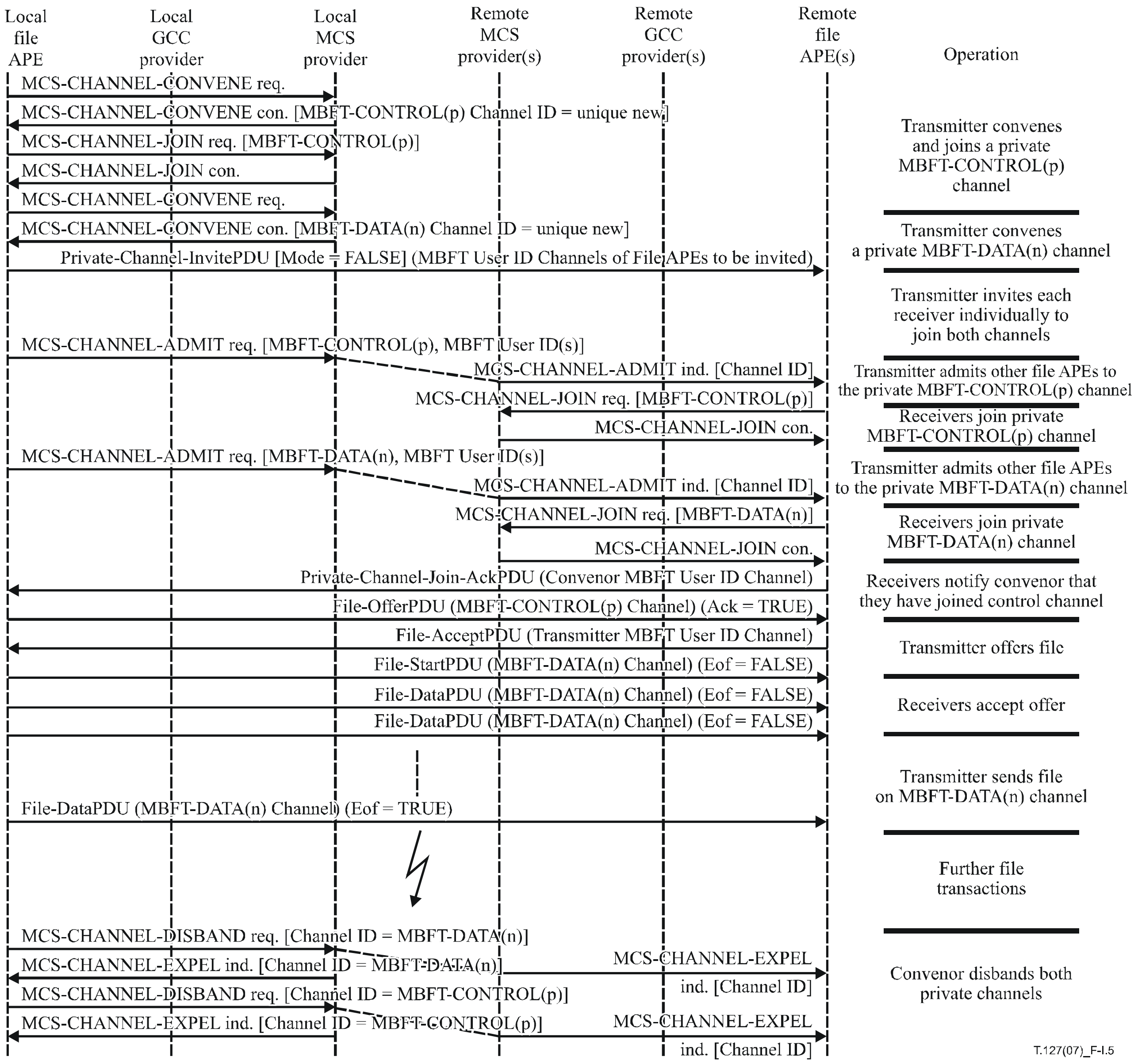

Local file APE
Local GCC provider
Local MCS provider
Remote MCS provider(s)
Remote GCC provider(s)
Remote file APE(s)
Operation
MCS-CHANNEL-CONVENE req.
MCS-CHANNEL-CONVENE con. [MBFT-CONTROL(p) Channel ID = unique new]
MCS-CHANNEL-JOIN req. [MBFT-CONTROL(p)]
MCS-CHANNEL-JOIN con.
Transmitter convenes and joins a private MBFT-CONTROL(p) channel
MCS-CHANNEL-CONVENE req.
MCS-CHANNEL-CONVENE con. [MBFT-DATA(n) Channel ID = unique new]
Transmitter convenes a private MBFT-DATA(n) channel
Private-Channel-InvitePDU [Mode = FALSE] (MBFT User ID Channels of File APEs to be invited)
Transmitter invites each receiver individually to join both channels
MCS-CHANNEL-ADMIT req. [MBFT-CONTROL(p), MBFT User ID(s)]
MCS-CHANNEL-ADMIT ind. [Channel ID]
Transmitter admits other file APEs to the private MBFT-CONTROL(p) channel
MCS-CHANNEL-JOIN req. [MBFT-CONTROL(p)]
MCS-CHANNEL-JOIN con.
Receivers join private MBFT-CONTROL(p) channel
MCS-CHANNEL-ADMIT req. [MBFT-DATA(n), MBFT User ID(s)]
MCS-CHANNEL-ADMIT ind. [Channel ID]
Transmitter admits other file APEs to the private MBFT-DATA(n) channel
MCS-CHANNEL-JOIN req. [MBFT-DATA(n)]
MCS-CHANNEL-JOIN con.
Receivers join private MBFT-DATA(n) channel
Private-Channel-Join-AckPDU (Convenor MBFT User ID Channel)
Receivers notify convenor that they have joined control channel
File-OfferPDU (MBFT-CONTROL(p) Channel) (Ack = TRUE)
File-AcceptPDU (Transmitter MBFT User ID Channel)
Transmitter offers file
File-StartPDU (MBFT-DATA(n) Channel) (Eof = FALSE)
File-DataPDU (MBFT-DATA(n) Channel) (Eof = FALSE)
File-DataPDU (MBFT-DATA(n) Channel) (Eof = FALSE)
Receivers accept offer
File-DataPDU (MBFT-DATA(n) Channel) (Eof = TRUE)
Transmitter sends file on MBFT-DATA(n) channel
Further file transactions
MCS-CHANNEL-DISBAND req. [Channel ID = MBFT-DATA(n)]
MCS-CHANNEL-EXPEL ind. [Channel ID = MBFT-DATA(n)]
MCS-CHANNEL-EXPEL ind. [Channel ID]
MCS-CHANNEL-DISBAND req. [Channel ID = MBFT-CONTROL(p)]
MCS-CHANNEL-EXPEL ind. [Channel ID = MBFT-CONTROL(p)]
MCS-CHANNEL-EXPEL ind. [Channel ID]
Convenor disbands both private channels
T.127(07)_F-I.5

Figure I.5 – Private sub-session file transfer on acknowledged data channel (convenor use only)

Appendix II

MBFT attributes

(This appendix does not form an integral part of this Recommendation)

A large number of parameters may be used to describe a file and its properties, as identified in Table II.1. The File-OfferPDU should include sufficient information to allow a site to determine whether a file is required and if it is capable of being received; filename and filesize are thus suggested as a minimum set, but no parameters are mandated. If insufficient (or no) information is provided and sites subsequently discover that they do not wish or are unable to continue receiving a file, they must disconnect from the data channel or discard the incoming data.

File-Start and File-DataPDUs are used to transport the entire file structure, including header information. Note that this may duplicate or supplement information contained in the File-OfferPDU.

File attributes are derived from [ITU-T T.434].

Table II.1 – MBFT file attributes

Attribute	Comments
Filename	A sequence of name components, the first element being the filename, and any subsequent components being concatenated to represent the file prefix.
Date and time of last modification	Expressed as GeneralizedTime
Compression	Applies only to file data
Filesize	Size in octets of the complete file
Protocol version	T.434 protocol version number
Permitted actions	Read, insert, replace, extend, erase
Contents type	Unstructured text, unstructured binary
Storage account	Identifies the accountable authority responsible for file storage charges
Date and time of creation	Expressed as GeneralizedTime
Date and time of last read access	Expressed as GeneralizedTime
Identity of creator	
Identity of last modifier	
Identifier of last reader	
Future filesize	Maximum file size in octets
Access control	For further study
Legal qualifications	Reflects the legal status of the file
Private use	Allows definition of proprietary attributes
Structure	Indicates the format of the data being transferred

Table II.1 – MBFT file attributes

Attribute	Comments
Application reference	Additional information describing various aspects of the environment the binary file transfer is originating from
Machine	
Operating system	
Environment	
Pathname	
User visible string	
Recipient	The final user destination of the binary file transfer
Character set	Identifies which international character set is to be used

SERIES OF ITU-T RECOMMENDATIONS

Series A	Organization of the work of ITU-T
Series D	General tariff principles
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media, digital systems and networks
Series H	Audiovisual and multimedia systems
Series I	Integrated services digital network
Series J	Cable networks and transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Construction, installation and protection of cables and other elements of outside plant
Series M	Telecommunication management, including TMN and network maintenance
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Telephone transmission quality, telephone installations, local line networks
Series Q	Switching and signalling
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	**Terminals for telematic services**
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	Data networks, open system communications and security
Series Y	Global information infrastructure, Internet protocol aspects and next-generation networks
Series Z	Languages and general software aspects for telecommunication systems

Printed in Switzerland
Geneva, 2008