INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

H.501

(03/2002)

SERIES H: AUDIOVISUAL AND MULTIMEDIA SYSTEMS

Mobility and Collaboration procedures – Overview of Mobility and Collaboration, definitions, protocols and procedures

Protocol for mobility management and intra/inter-domain communication in multimedia systems

ITU-T Recommendation H.501

ITU-T Recommendation H.501

Protocol for mobility management and intra/inter-domain communication in multimedia systems

Summary

The purpose of this Recommendation is to define messages and procedures for mobility management and for communication within and between domains of a mobile or non-mobile multimedia environment for the purpose of address resolution, user authentication, service data exchange, access authorization, call validation and usage reporting.

Source

ITU-T Recommendation H.501 was prepared by ITU-T Study Group 16 (2001-2004) and approved under the WTSA Resolution 1 procedure on 29 March 2002.

Keywords

Address resolution, administrative domains, authentication, authorization, H.225.0 Annex G, mobility management, multimedia systems, usage reporting.

CONTENTS

Page

ITU-T Recommendation H.501

Protocol for mobility management and intra/inter-domain communication in multimedia systems

1 Scope

This Recommendation describes a protocol for communication between logical elements of a multimedia packet network to allow the completion of calls to and from users managed by such logical elements. This protocol can be used in mobile and non-mobile environments for the purpose of address resolution, user authentication, service data exchange, access authorization, call validation and usage reporting. These capabilities enable the protocol to be used for mobility management in mobile environments.

The general procedure is for logical elements to exchange information regarding the location of users or endpoints, in the form of addresses each administrative domain can resolve. Addresses can be specified in a general manner or in an increasingly specific manner. Additional information allows elements within an administrative domain to determine the most appropriate administrative domain to serve as the destination for the call. Logical elements may control access to their exposed addresses, and require reports on the usage made during calls to those addresses.

Other Recommendations will specify how the protocol defined in this Recommendation is used by particular applications. It is not necessary for an application to implement the full protocol. The application can select from the messages and procedures those relevant to its requirements.

2 Normative 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.

– ITU-T Recommendation E.164 (1997), *The international public telecommunication numbering plan*.

– ITU-T Recommendation H.225.0 Version 4 (2000), *Call signalling protocols and media stream packetization for packet-based multimedia communication systems*.

– ITU-T Recommendation H.235 Version 2 (2000), *Security and encryption for H-series (H.323 and other H.245-based) multimedia terminals*.

– ITU-T Recommendation H.323 Version 4 (2000), *Packet-based multimedia communications systems*.

– ITU-T Recommendation X.680 (1997), *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation*.

– ITU-T Recommendation X.691 (1997), *Information technology – ASN.1 encoding rules: Specification of Packed Encoding Rules (PER)*.

– IETF RFC 2401 (1998), *Security Architecture for the Internet Protocol*.

– IETF RFC 2402 (1998), *IP Authentication Header*.

– IETF RFC 2406 (1998), *IP Encapsulating Security Payload (ESP)*.

– ISO 4217:2001, *Codes for the representation of currencies and funds*.

3 Definitions

This Recommendation defines the following terms:

3.1 administrative domain: An administrative domain is a collection of logical and physical entities administered by one administrative entity. An administrative domain can consist of one or more zones.

3.2 gatekeeper: A logical element that provides specific services (number translation, access control, etc.) to other entities within an administrative domain (see also ITU-T Rec. H.323).

3.3 logical element: An entity with defined functionality in a network. Logical elements do not impose any requirements on their provision; their functionality can be implemented in any suitable way in hard- or software.

3.4 peer element: A logical element which originates or terminates signalling messages defined in this Recommendation. Examples are H.225.0 Annex G border elements or H.323 gatekeepers.

3.5 zone: The subset of entities of an administrative domain under the control of a single gatekeeper.

4 Symbols and abbreviations

This Recommendation uses the following abbreviations:

AD	Administrative domain
DH	Diffie-Hellman key agreement protocol (ITU-T Rec. X.509)
DNS	Domain Name System
IP	Internet Protocol
OID	Object Identifier
PDU	Protocol Data Unit
RAS	Registration, Admission and Status protocol
SCN	Switched Circuit Network
TCP	Transmission Control Protocol
TPKT	Transport Packet
UDP	User Datagram Protocol
URL	Uniform Resource Locator
UTC	Universal Time Coordinated

5 Requirements

5.1 Transport requirements

Messages may be sent over an unreliable transport service (e.g. UDP) or a reliable transport service (e.g. TCP) to a well-known address. On IP networks, the well-known port *(2099)* should be used for both TCP and UDP, unless another port has been communicated to the sender. Elements shall listen on both TCP and UDP ports.

When messages are sent over the reliable transport service, whole messages shall be sent within the boundaries defined by the reliable transport protocol data unit (PDU). (In IP implementations, for instance as outlined in H.225.0 Appendix IV, this PDU is defined by TPKT; see Figure 1. Each H.501 PDU contains a single message defined in this Recommendation.)

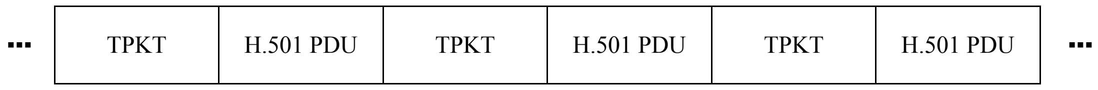

Figure 1/H.501 – TCP transport

When using an unreliable transport service, request messages may be retransmitted. The default value of the retransmission timer should be determined by an adaptive delay sensitive method (such as the one used by the TCP protocol). Exponential backoff shall be used for subsequent retransmissions. The number of retransmissions shall not exceed 5. Responses shall not be retransmitted.

In UDP IP implementations, messages shall also be prefixed with TPKT headers, to enable multiple messages per packet, see Figure 2. Each H.501 PDU contains a single message defined in this Recommendation. The UDP packet length field shall hold the total length of the payload, including all the messages and their TPKT headers.

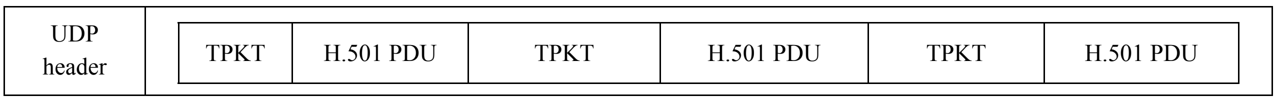

Figure 2/H.501 – UDP datagram format

5.2 Security considerations

When authentication, integrity, and encryption on the transport layer is desired for messages exchanged between peer elements, the operation of IP security shall be followed as described in IETF RFC 2401, including either, or both, of IETF RFC 2402, and IETF RFC 2406.

Where protection on the application layer is required, the procedures and constructs of ITU-T Rec. H.235 shall be utilized to support application-level security. Specifically, the token formats and authentication exchanges shall be used. Tokens and crypto-tokens received in response messages should be used in a subsequent related request.

5.3 Addressing conventions

In order to provide interoperability between domains, it is important that the addressing formats sent in signalling messages are understood by the receiving system. A peer element shall support all types of *AliasAddress* that an application may globally use.

At a minimum, the formats *email-id* and *partyNumber* (using *PublicNumber* with *PublicTypeOfNumber* of *internationalNumber*) shall be supported. When communicating with other peer elements, only the *email-id* and *partyNumber* types of *AliasAddress* should be used, unless there has been prior agreement on the use of other formats between the administrative domains concerned. For example, if a group of administrative domains have agreed on the interpretation of a private numbering plan, then these numbers may be used in messages amongst them.

In a mobile environment, the format *mobileUIM* may be used as a global user identification module upon agreement amongst the administrative domains concerned. This format is described in detail in ITU-T Rec. H.225.0.

5.4 Address templates and descriptors

An address template ("template", for short) defines a set of *AliasAddress* identifiers, pricing information to complete calls to those addresses, and the protocol to be used in reaching addresses in that set. An administrative domain advertises templates to indicate the calls it can resolve. Templates are grouped together by an identifier known as a "descriptor". Once a template is grouped by a descriptor, any change to a template under that descriptor implies a change to the descriptor "group". Template information may allow the aggregation of addressing information if the addressing scheme is arranged in some hierarchical or routable manner (for example, a given zone might handle 1303538*, meaning all telephone numbers that begin with 1303538).

NOTE – Since "*" is a meaningful character, the template actually includes an explicit field to indicate whether the address is a specific address or a wildcard address. These examples use "*" to indicate a wild card, but the actual representation in the template is through the explicit field. The "*" is not sent over the line if it indicates a wild card.

Template examples include:

"For 1 555 123 4567	send AccessRequest message to peer element A"
"For 1 555 987 *	send AccessRequest message to peer element B"
"For 1 555 987 6543	send Setup message to gateway X"
"For *@example.org	send AccessRequest message to peer element A"
"For 1	send AccessRequest message to peer element B"
"For private 31*	send AccessRequest message to peer element C"
"For 44 171 112*	doesn't exist"

6 Message definitions

This clause specifies the messages and elements of the protocol defined in this Recommendation. Annex A contains the corresponding ASN.1 definition according to ITU-T Rec. X.680. Messages shall be encoded using the basic aligned variant of packed encoding rules according to ITU-T Rec. X.691.

Each message contains a set of common fields in addition to the message-specific information. The common fields are:

Field	Description
sequenceNumber	Each request or update message contains a unique sequence number. The message sent in response to a request message (a confirmation or rejection message) uses the sequence number from the request message. Retransmitted messages shall have the same sequence number.
replyAddress	This is the address to which to send the reply to a request message. All request messages shall include a replyAddress except for cases where the address can be derived from the transport layer. On IP networks, if the sender of the request message is listening on the default port (*2099*), then the reply address need not be included. In such a case, the receiver obtains the transport address of the sender by appending default port (*2099*) to the IP address of the sender as received in the IP header of the request packet.[1]
version	Protocol version in use by the sender of this message.

[1] Peer elements are assumed not to be hidden behind network address translation (NAT) devices, thus it is not required to prefer the transport address over the replyAddress, as is the case for RAS messages.

annexGversion	Protocol version of H.225.0 Annex G (only present for backward compatibility; it shall indicate "Annex G V2").
hopCount	This defines the number of peer elements through which this message may propagate. When a peer element receives this message and decides that the message should be forwarded on to another peer element, it first decrements *hopCount*. If *hopCount* is then greater than 0, the peer element inserts the new hop count value into the message to be forwarded. If *hopCount* has reached 0, the peer element shall not forward the message. If the message is a request, the peer element should respond with a confirmation message with any applicable information. If no information is available, the peer element should respond with a rejection message.
integrityCheckValue	Provides improved message integrity/message authentication. The cryptographically based integrity check value is computed by the sender applying a negotiated integrity algorithm and the secret key upon the entire message. Prior to integrityCheckValue computation each byte of this field shall be set to zero. After computation, the sender puts the computed integrity check value in the integrityCheckValue field and transmits the message.
tokens	This is some data that may be required to allow the operation. The data shall be inserted into the message if available.
cryptoTokens	Encrypted tokens.
nonStandard	Non-standard information.
serviceID	This identifier identifies a particular service relationship session between two peer elements. Whenever a peer element receives a ServiceRequest message requesting the establishment of a new service relationship (which is indicated by the absence of the service ID field in the ServiceRequest message), it allocates a **globally** unique service ID and returns it to the sender of the ServiceRequest message in the ServiceConfirmation message. Once a service relationship has been established, the service ID is included in all subsequent messages with the peer element (e.g. UsageIndication, DescriptorIDRequest, DescriptorRequest, AccessRequest). This is used by the recipient peer element to check if it has a service relationship with the sender of the message.
genericData	Carries any generic data associated with the message. GenericData is described in ITU-T Rec. H.323/ITU-T Rec. H.225.0.
featureSet	Used to negotiate generic feature sets. The negotiation scheme is as described for RAS signalling in ITU-T Rec. H.323/ITU-T Rec. H.225.0.

6.1 Descriptor

The Descriptor is not a message, but is rather a message element used to label a set of templates.

The Descriptor contains the following information:

Field	**Description**
descriptorInfo	This holds a unique identifier for the descriptor and the time it was last changed (see Descriptor Information below).
templates	This is a set of templates that define the addresses this descriptor can resolve.

gatekeeperID	This is a text identifier that indicates the owner of the descriptor (i.e. the gatekeeper that created this message).

6.2 Descriptor information

Descriptor information uniquely identifies the descriptor and indicates the last time the descriptor changed.

Field	Description
descriptorID	This is a globally unique identifier used to identify this descriptor from among many possible descriptors.
lastChanged	This is the UTC date and time this descriptor was last changed.

6.3 Address template

The Address Template describes a set of one or more alias addresses. The Template is not a message, but is an element used as a building block for other elements. The Template consists of other structures, which are described in subsections.

Field	Description
pattern	This is a list of patterns (see Pattern below).
routeInfo	This is a list of route information for this template (see Route Information below).
timeToLive	This indicates the time, expressed in seconds, for which this template is valid.
supportedProtocols	Identifies the type of protocols that are supported by this template (e.g. voice, fax).
featureSet	Specifies the generic feature sets this addressTemplate supports and what generic features it needs and desires in a remote endpoint. Generic Feature information specified at this level applies to all routeInformation applicable to this AddressTemplate. The Generic Extensible Framework is described in ITU-T Rec. H.323/ITU-T Rec. H.225.0.

6.3.1 Pattern

The *pattern* structure appears in the Address Template. The *pattern* allows specification of an alias address, a wildcarded alias address, or a range of alias addresses: wildcarded addresses assume an hierarchically structured address space, while address ranges are only meaningful for address types that represent an ordered structure.

Field	Description
specific	This is a specific alias address.
wildcard	This is some hierarchical definition that represents possible expansions of the string. For E.164 numbers, this expansion is possible at the end of the number; for email addresses, the expansion is possible at the beginning. For example, if *wildcard* is "+1 303", the pattern could represent any number in the Denver, Colorado, USA area code.
range	This is a range of addresses, including the indicated start and end of range.

6.3.2 Route information

The route information structure found in the *template* (the *routeInfo* field) contains the following:

Field	Description
messageType	This indicates the type of message to send when attempting to resolve a specific address within this template. Possibilities are *sendAccessRequest*, *sendSetup*, or *nonExistent* (indicates that the address does not exist).
callSpecific	If set to TRUE, authorization is requested for each call to this route, implying that the AccessRequest message shall include the call information. This boolean field has meaning only when *messageType* is *sendAccessRequest*; otherwise, *callSpecific* shall be set to FALSE.
usageSpec	If present, this specifies the UsageIndication messages that shall be sent regarding the calls to this route.
priceInfo	This is a list of pricing information for this particular route (see Pricing information below). Note that multiple gateways with different pricing structures would be described in multiple *RouteInformation* structures.
contacts	This is contact information for the element that will accept the message as specified in the *messageType* field of *routeInfo*. The contact information may be provided as a list of possible contacts (see Contact information description below).
type	This indicates the type of endpoint that can serve the call. For gatekeeper routed cases, this indicates the types of endpoints served by the gatekeeper rather than the gatekeeper itself.
featureSet	Specifies the generic feature sets this route supports and what generic features it needs and desires in a remote endpoint. Feature information specified at this level applies to all *ContactInformation* applicable to this *RouteInformation* element. The Generic Extensible Framework is described in ITU-T Rec. H.323/ITU-T Rec. H.225.0.
circuitID	If present, this holds the SCN circuit information that applies to a specific call. Circuit information specified at this level applies to all *ContactInformation* applicable to this *RouteInformation* element.
supportedCircuits	If present, this holds circuit identifier values for the SCN circuits that are supported in a domain or zone for the destination pattern. This allows a peer element to advertise the support of destination circuit information to remote peer elements. Circuit information specified at this level applies to all *ContactInformation* applicable to this *RouteInformation* element.

6.3.3 Pricing information

Pricing information appears as an element in the route information structure (the *priceInfo* field). Pricing information is defined through the *PriceInfoSpec* and *PriceElement* structures.

The *PriceInfoSpec* structure contains the following fields:

Field	Description
currency	This is an ISO 4217 currency designator.
currencyScale	This is the number of places to shift the implied radix point to the left. For example, when *currency* is specified as USD, a *currencyScale* of 2 would indicate that the amount in *priceElement* is expressed in US cents.

validFrom	This is the UTC date and time from which this information is valid.
validUntil	This is the UTC date and time at which this information expires.
hoursFrom	This is the time of day when this rate starts.
hoursUntil	This is the time of day when this rate ends. It may be less than *hoursFrom*, indicating a rate which spans 0000.
priceElement	This is an optional list of *PriceElements* which sum to effect the pricing.
priceFormula	This is an optional string containing a pricing formula used as an alternative to the structured *PriceElement*.

The *PriceElement* structure contains the following fields:

Field	**Description**
amount	This is the meter increment. The meter increments once for each *quantum* or fraction of *quantum*.
quantum	This is the number of units for which *amount* applies. For example, a value of 60, with *units* in seconds, indicates that the call is priced per minute or fraction of minute. If the *units* field is set to either of *initial*, *minimum* or *maximum* values, then the *quantum* field is irrelevant, and its value shall be ignored by the recipient.
units	This is the type of unit in which quantum is expressed: • seconds – seconds of call duration. • packets – packets transmitted or received. • bytes – bytes transmitted or received. • initial – an initial connect charge. • minimum – a minimum call charge. • maximum – a maximum call charge.

6.3.4 Contact information

The contact information structure is an element of the route information structure (the *contacts* field).

Field	**Description**
transportAddress	This is the alias address (e.g. transport address or URL) to which to send the message specified in the messageType field of the RouteInformation structure. Whenever possible, a transport address shall be used.
priority	When multiple contacts are listed, the priority field specifies the order in which the multiple contacts should be tried. Contacts in the list can share a priority, for example if there is no preference on the order in which the contacts should be tried. A priority of 0 indicates the highest priority (first choice).
transportQoS	Indicates where the responsibility lies for resource reservation for a call made through this contact.
security	Security mechanism in descending order of preference to be used when communicating with contact.

accessTokens	This is a set of tokens that shall be passed in the message to this contact (Setup or AccessRequest). These tokens shall also be sent in subsequent UsageIndication messages pertaining to the calls using this template.
multipleCalls	This field has significance only when the value in the messageType field of the RouteInformation structure is sendSetup. If multipleCalls is TRUE, this indicates that the contact is capable of signalling multiple calls over a single call signalling connection. If FALSE, the contact does not have this capability.
featureSet	Specifies the generic feature sets that the entity associated with this ContactInformation element supports and what generic features it needs and desires in a remote endpoint. The Generic Extensible Framework is described in ITU-T Rec. H.323/ITU-T Rec. H.225.0.
circuitID	If present, this holds the SCN circuit information that applies to a specific call. Circuit information specified at this level applies to the contact related to this ContactInformation element.
supportedCircuits	If present, this holds circuit identifier values for the SCN circuits that are supported in a domain or zone for the destination pattern. This allows a peer element to advertise the support of destination circuit information to remote peer elements. Circuit information specified at this level applies to the contact related to this ContactInformation element.

6.4 Common structures

The structures defined in this clause appear in many of the messages.

6.4.1 AlternatePE

Field	**Description**
contactAddress	This is the alternate peer element's transport address (the address to which to send messages of this protocol).
priority	When multiple alternates are listed, the *priority* field specifies the order in which the multiple alternates should be tried. Alternates in the list can share a priority, for example if there is no preference on the order in which the alternates should be tried. A priority of 0 indicates the highest priority (first choice).
elementIdentifier	This alternate peer element uses this Unicode string as an identifier.

6.4.2 PartyInformation

This structure contains information about a party of the call (either source or destination).

Field	**Description**
logicalAddress	E-mail or E.164 formatted addresses that identify the party.
domainIdentifier	An alias address identifying the AD which originated or terminated the call. If multiple domains are involved in placing a call, then the domain that served as the call origination or termination from the sender's perspective should be stated.
transportAddress	This is the transport address of the endpoint.

endpointType	This indicates details about the endpoint type and capabilities.
userInfo	This is information regarding the user behind the call. See *UserInformation* below.
timeZone	This is the time zone of the party, as relevant for pricing purposes. If the originating party is a gateway, then the time zone of the gateway has to be conveyed. Described in seconds relative to UTC.

6.4.3 CallInformation

This structure contains information for identifying a specific call.

Field	**Description**
callIdentifier	This provides unique identification of the call. This shall be the *callIdentifier* associated with the same call as in RAS and call signalling messages.
conferenceID	This provides unique identification of the conference to which the call belongs. This shall be the *conferenceID* associated with the same call as in RAS and call signalling messages.
circuitID	If present, this holds the SCN circuit information that applies to the call.

6.4.4 UserInformation

This structure contains information for identifying the user represented by any party of the call.

Field	**Description**
userIdentifier	Alias address that uniquely identifies the user.
userAuthenticator	Encrypted tokens for secure authentication.

6.4.5 Usage specification

This element describes the required parameters needed to be reported in the UsageIndication messages. The calls for which this specification applies is determined by the context of the message containing the *UsageSpecification* element.

Field	**Description**
sendTo	Peer element to which the UsageIndication messages are to be sent. If the sender has a service relationship with that peer element, this is the element identifier returned in the ServiceConfirmation message.
when	Specifies the stages of the call, and the frequency, at which the indications should be sent: • never – stop sending messages. • start – when the call begins. • end – by the end of the call, or thereafter. • period – periodically, during the call lifetime. The period is measured in seconds. • failure – report failed call attempts.
required	A list of identifiers for fields that *must* be present in the UsageIndication messages. The sender of the usage information shall reject or ignore the message containing this message, if it cannot supply these fields.
preferred	A list of identifiers for fields that *should* be present in the UsageIndication messages.

sendToPEAddress	A resolvable address that, when resolved, specifies the address of a peer element to which UsageIndication messages shall be sent. If the resolution of this field results in more than one address (for example, in the case where a DNS query returns a list of addresses), the peer element shall send the UsageIndication messages to only one peer element from the list. If the peer element does not succeed in sending to one address, it may choose another address from the list and attempt to send the UsageIndication messages to the new address. The peer element may continue attempting each additional address in the list until it either receives a UsageIndicationConfirmation, a UsageIndicationRejection, or until there are no further addresses to attempt. Note that the "sendToPEAddress" field is different from the "sendTo" field in the UsageSpecification. The "sendTo" field is an identifier. It can be the identifier of a specific peer element (e.g. "border_element1"), or it can be an identifier that logically represents a set of peer elements (e.g. "border elements of my company"). The "sendToPEAddress" field resolves to one or more addresses.

6.4.6 Security Mode

This element describes a specific security profile to be used for peer to peer communication.

Field	Description
authentication	This indicates the authentication mechanism to be used. The authentication mechanism must be chosen from the set provided in the ServiceRequest message.
integrity	This indicates the integrity mechanism to be used. If present, all subsequent messages shall populate the *integrityCheckValue* field, in this case, the *AuthenticationMode* describes the way the secret keys are generated (DH exchange, or *a priori*).
algorithmOIDs	This indicates the encryption algorithms for the security mechanism.

6.5 Service relationship

6.5.1 Service Request

A peer element may send a ServiceRequest message to another peer element to establish a service relationship. The relationship defines the security mechanisms to be used between the peer elements and allows identification of alternate, or backup, peer elements. Note that the relationship is a one-way relationship. The security negotiated between the 2-peer elements is used for requests sent by the peer element that sent the ServiceRequest and for responses sent by the recipient of the ServiceRequest. Session keys may be generated during the process of service relationship establishment. The keys will be valid through the lifetime of the service relationship. Tokens may be used for that purpose, as defined in ITU-T Rec. H.235.

The recipient of the ServiceRequest may indicate alternate peer elements that the sender of ServiceRequest may try for backup service. Establishing a service relationship is an optional procedure, although a peer element's policy may require such a relationship.

A peer element may send a ServiceRequest message to a peer element with which it has an existing relationship, with the intent that the terms of the original relationship be terminated and replaced with the new terms. Service relationships may have limited time to live. A peer element may refresh the relationship by sending a new Service Request.

Field	Description
elementIdentifier	A string that identifies the peer element that sends the request.
domainIdentifier	The AD that requests the service relationship.
securityCapability	Set of security mechanisms that this peer element can support.
timeToLive	The suggested lifetime in seconds for the service relationship. If not present, infinite lifetime is assumed.
usageSpec	This specifies the usage information that the originating peer element requests the receiving peer element to send for all calls between the originating and receiving peer elements.

6.5.2 Service Confirmation

A peer element in receipt of a ServiceRequest message responds with a ServiceConfirmation message to indicate that it agrees to establish a service relationship. Every new service relationship is identified by a service identifier. Whenever a peer element receives a ServiceRequest message without a service ID, it allocates a unique service ID and returns it to the sender of the service request message in the ServiceConfirmation message. If the peer element already has a service relationship with the peer element that sent the ServiceRequest message, sending ServiceConfirmation indicates that the terms of the original relationship are terminated and replaced with the new terms. The ServiceConfirmation message shall contain the same service ID that was sent in the ServiceRequest message. A peer element that receives a ServiceRequest message containing a service ID that it does not recognize shall respond with a ServiceRejection message.

Field	Description
elementIdentifier	This is a string that identifies the peer element.
domainIdentifier	The AD that responds to the request.
alternates	This is a list of alternate peer elements that may be contacted in the event that this peer element fails to respond.
securityMode	This indicates the security mechanism to be used for this service relationship. The security mechanism must be chosen from the set provided in the ServiceRequest message.
timeToLive	The lifetime in seconds of the service relationship as determined by the serving peer element.
usageSpec	This specifies the usage information that the receiving peer element can support for all calls between the originating and receiving peer elements.

6.5.3 Service Rejection

A peer element in receipt of a ServiceRequest message responds with a ServiceRejection message to indicate that it declines to establish a service relationship. If the peer element already has a service relationship with the peer element that sent the ServiceRequest message, sending ServiceRejection indicates that the proposed new terms have been rejected, but the terms of the original relationship remain.

Field	Description
reason	This is the reason the peer element rejected the ServiceRequest. Choices are: • serviceUnavailable – This peer element is not currently available for service. • serviceRedirected – The list of alternate peer elements should be attempted. • security – This peer element cannot support any of the security mechanisms proposed in the ServiceRequest message. • continue – Indicates the subsequent ServiceRequest message be sent, in order to continue multiple stage key exchange processes. • undefined – The reason for rejecting the ServiceRequest does not match any of the other choices. • unknownServiceID – The serviceID field contained in the ServiceRequest message is not recognized by the peer element • cannotSupportUsageSpec – The peer element cannot comply with the proposed UsageSpecification. • neededFeature – Indicates that the request failed because the requesting entity did not support one or more needed generic features. The features that are needed are indicated in the neededFeatures field in the MessageCommonInfo field of the reply. • genericDataReason – Indicates that the real reason code is contained within the genericData field sent with this message. Note that the genericData may contain more than one reason code. • usageUnavailable – The peer element does not support usage reporting. • unknownUsageSendTo – The sendTo or sendToPEAddress specified in the proposed UsageSpecification could not be resolved.
alternates	This is a list of alternate peer elements that might be able to honour the ServiceRequest. If the *reason* is *serviceRedirected*, at least one alternate should be provided.

6.5.4 Service Release

Either peer element in a service relationship may terminate the relationship by sending the ServiceRelease message.

Field	Description
reason	This is the reason this peer element terminated the service relationship. Choices are: • outOfService – The peer element is going out of service. • maintenance – The peer element is being taken out of service for maintenance. • terminated – The peer element has decided to terminate the relationship. • expired – The time-to-live for the service relationship has elapsed.
alternates	This is a list of alternate peer elements that might be able to establish a service relationship.

6.6 Descriptor distribution

6.6.1 Descriptor Request

The DescriptorRequest message allows an entity to query a peer element for specific descriptors.

Field	Description
descriptorID	This identifies one or more particular descriptors requested by the sender of this message.

6.6.2 Descriptor Confirmation

The DescriptorConfirmation message is a peer element's positive response to a DescriptorRequest, when the peer element can interpret the request and implementation rules allow information exchange.

Field	Description
descriptor	This is(are) the descriptor(s) described above.

6.6.3 Descriptor Rejection

A peer element can reject a descriptor request for a variety of reasons.

Field	Description
reason	This is the reason the DescriptorRequest was rejected. Choices are: • packetSizeExceeded – The reply would exceed the maximum packet size, so the requester should send the request using a different transport mechanism (e.g. use TCP instead of UDP). • illegalID – The recipient of the DescriptorRequest has no record of the requested descriptor. • security – The DescriptorRequest did not meet the recipient's security requirements. • hopCountExceeded – The hop count reached zero and no information is available.

	• unavailable – The recipient cannot provide descriptors. Static or out-of-band provisioning method should be used. • noServiceRelationship – The recipient will exchange this information only after establishment of a service relationship. • undefined – The reason for rejecting the DescriptorRequest does not match the other choices. • neededFeature – Indicates that the request failed because the requesting entity did not support one or more needed generic features. The features that are needed are indicated in the neededFeatures field in the MessageCommonInfo field of the reply. • genericDataReason – Indicates that the real reason code is contained within the genericData field sent with this message. Note that the genericData may contain more than one reason code. • unknownServiceID – The serviceID field contained in the DescriptorRequest message is not recognized by the peer element
descriptorID	This identifies the specific descriptor for this response.

6.6.4 Descriptor ID Request

The DescriptorIDRequest allows an entity to query a peer element for the list of descriptor identifiers within the peer element's administrative domain.

6.6.5 Descriptor ID Confirmation

A DescriptorIDConfirmation message is a peer element's positive response to the DescriptorIDRequest message. A peer element in receipt of a DescriptorIDConfirmation message may send the DescriptorRequest message to request transmission of the descriptors.

Field	**Description**
descriptorInfo	This is a list of descriptor information, where each entry in the list uniquely identifies the descriptor and the time it last changed.

6.6.6 Descriptor ID Rejection

A peer element can reject a DescriptorIDRequest for a variety of reasons.

Field	**Description**
reason	This indicates the reason for rejecting the request. Choices are: • noDescriptors – This indicates that the peer element has no descriptors to report. • security – The DescriptorIDRequest did not meet the recipient's security requirements. • hopCountExceeded – The hop count reached zero and no information is available. • unavailable – The recipient cannot provide descriptors. Static or out-of-band provisioning method should be used. • noServiceRelationship – The recipient will exchange this information only after establishment of a service relationship. • undefined – The reason for rejecting the DescriptorIDRequest does not match the other choices.

- neededFeature – Indicates that the request failed because the requesting entity did not support one or more needed generic features. The features that are needed are indicated in the neededFeatures field in the MessageCommonInfo field of the reply.
- genericDataReason – Indicates that the real reason code is contained within the genericData field sent with this message. Note that the genericData may contain more than one reason code.
- unknownServiceID – The serviceID field contained in the DescriptorIDRequest message is not recognized by the peer element.

6.6.7 Descriptor Update

The DescriptorUpdate message is a peer element's notification that address information has changed. A peer element may also send the DescriptorUpdate message during initialization. A peer element in receipt of the DescriptorUpdate may request information from the element identified in the DescriptorUpdate.

Field	Description
sender	An element in receipt of the DescriptorUpdate may send a request to this address (e.g. transport address or URL).
updateInfo	This is a list of updates. Each entry in the list provides either the descriptor or the descriptor identifier that was updated. Each entry in the list also indicates whether the descriptor was changed, added or deleted.

6.6.8 Descriptor Update Acknowledgement

A peer element should acknowledge receipt of a DescriptorUpdate message by sending the DescriptorUpdateAck message. The sequence number used in the acknowledgement should be the same as the sequence number received in the DescriptorUpdate message. A peer element should not acknowledge a DescriptorUpdate message that arrives over multicast.

6.7 Address resolution

6.7.1 Access Request

A peer element can send an AccessRequest message to another peer element to ask for resolution of a specific alias address.

Field	Description
destinationInfo	This is the address to be resolved.
sourceInfo	This is information about the originating party of the call for which access is requested.
callInfo	This provides identification of the particular call for which access authorization is requested. If not present, then the request is for indefinite calls to the specified destinations.
usageSpec	This indicates the usage messages that the originating party requests the answering party to send regarding the call requested in this message. Applies only if *CallInfo* is present.

desiredProtocols	Identifies the type of protocols, in order of preference, the endpoint desires for its call (e.g. voice, fax). A resolving entity may use this field to locate an endpoint that also supports the protocol, giving consideration to the order of preference. The called entity shall ignore the supportedPrefixes field of this structure.

6.7.2 Access Confirmation

A peer element returns in the AccessConfirmation message the information requested in the AccessRequest message.

Field	Description
templates	This is a list of templates which match the attributes of the AccessRequest.
partialResponse	If TRUE, this message contains some fraction of the available information. The entire information was not sent because it would exceed the packet size. The entire information should be retrieved using another transport type (e.g. TCP).
supportedProtocols	Identifies the type of protocols that are supported (e.g. voice, fax).
serviceControl	As defined in ITU-T Rec. H.225.0 for use e.g. as described in Annex K/H.323.

6.7.3 Access Rejection

A peer element can reject an AccessRequest for a variety of reasons.

Field	Description
reason	This is the reason for rejecting the request. Choices are: • noMatch – The destination specified in the AccessRequest cannot be resolved. • packetSizeExceeded – The reply would exceed the maximum packet size, so the requester should send the request using a different transport mechanism (e.g. use TCP instead of UDP). • security – The AccessRequest did not meet the recipient's security requirements. • hopCountExceeded – The hop count reached zero and no information is available. • noServiceRelationship – The recipient will exchange this information only after establishment of a service relationship. • needCallInformation – Specific call information was not present in the request. • undefined – The reason for rejecting the AccessRequest does not match the other choices. • neededFeature – Indicates that the request failed because the requesting entity did not support one or more needed generic features. The features that are needed are indicated in the neededFeatures field in the MessageCommonInfo field of the reply. • genericDataReason – Indicates that the real reason code is contained within the genericData field sent with this message. Note that the genericData may contain more than one reason code. • destinationUnavailable – Destination was resolved but is unavailable.

	• aliasesInconsistent – Multiple aliases identify distinct destinations. • resourceUnavailable – One or more required resources are unavailable. • incompleteAddress – Destination cannot be distinctly identified. • unknownServiceID – The serviceID field contained in the AccessRequest message is not recognized by the peer element. • usageUnavailable – The peer element does not support usage reporting. • cannotSupportUsageSpec – The peer element cannot comply with the proposed UsageSpecification. • unknownUsageSendTo – The sendTo or sendToPEAddress specified in the proposed UsageSpecification could not be resolved.
serviceControl	As defined in ITU-T Rec. H.225.0 for use e.g. as described in Annex K/H.323.

6.8 Request in Process

A peer element may return the RequestInProgress message to indicate that the time required by the peer element to respond to a request may exceed normal expected response intervals. The sequence number shall be the same sequence number found in the request for which this message will be sent.

Field	Description
delay	The expected length of time, expressed in milliseconds, for the peer element to respond to the original request.
serviceControl	As defined in ITU-T Rec. H.225.0 for use e.g. as described in Annex K/H.323.

6.9 Non-standard messages

6.9.1 Non-Standard Request

The NonStandardRequest may be sent from a peer element to represent a request message not defined in this Recommendation. The non-standard information is carried in the *nonStandard* element of *MessageCommonInfo*.

6.9.2 Non-Standard Confirmation

The NonStandardConfirmation may be sent from a peer element in response to a NonStandardRequest message. The non-standard information is carried in the *nonStandard* element of *MessageCommonInfo*.

6.9.3 Non-Standard Rejection

The NonStandardRejection may be sent from a peer element in response to a NonStandardRequest message. The non-standard information is carried in the *nonStandard* element of *MessageCommonInfo*.

Field	Description
reason	This is the reason for rejecting the request. Choices are: • notSupported – The recipient understands that this is a NonStandardRequest, but does not understand or support the non-standard data.

	• noServiceRelationship – The recipient will exchange this information only after establishment of a service relationship.
	• undefined – The reason for rejecting the NonStandardRequest does not match the other choices.
	• neededFeature – Indicates that the request failed because the requesting entity did not support one or more needed generic features. The features that are needed are indicated in the neededFeatures field in the MessageCommonInfo field of the reply.
	• genericDataReason – Indicates that the real reason code is contained within the genericData field sent with this message. Note that the genericData may contain more than one reason code.
	• unknownServiceID – The serviceID field contained in the NonStandardRequest message is not recognized by the peer element.

6.10 Unknown Message Response

A peer element in receipt of a message it does not understand should respond to the transmitter with the UnknownMessageResponse message. The peer element should not use this message if some other message provides an appropriate response (for example, a DescriptorRejection would be the appropriate response to a DescriptorRequest with an illegal descriptor identifier).

Field	**Description**
unknownMessage	This is the contents of the unknown message.
reason	This is the reason the UnknownMessageResponse was used. Choices are: • notUnderstood – The message was not understood. • undefined – The reason for sending UnknownMessageResponse does not match any of the other choices.

6.11 Usage reports

6.11.1 Usage Request

Request the recipient to send UsageIndication messages concerning a specific call.

Field	**Description**
CallInfo	The call for which to send the Indication.
UsageSpec	Specifies when the indications should arrive, and what they should contain.

6.11.2 Usage Confirmation

The UsageConfirmation message is sent in response to a Usage Request message to indicate that the recipient accepted the request and will send usage indications.

6.11.3 Usage Rejection

The UsageRejection message is sent in response to a Usage Request message to indicate that the recipient rejected the request and will not send the usage indications subsequently.

Field	Description
reason	This is the reason why the peer element rejected the UsageRequest. Choices are: • invalidCall – The call specified in the UsageRequest is not a recognized call. • security – The UsageRequest did not meet the recipient's security requirements. • unavailable – The recipient does not have usage information for the requested call. • noServiceRelationship – The recipient will exchange this information only after establishment of a service relationship. • undefined – The reason for rejecting the UsageRequest does not match any of the other choices. • neededFeature – Indicates that the request failed because the requesting entity did not support one or more needed generic features. The features that are needed are indicated in the neededFeatures field in the MessageCommonInfo field of the reply. • genericDataReason – Indicates that the real reason code is contained within the genericData field sent with this message. Note that the genericData may contain more than one reason code. • unknownServiceID – The serviceID field contained in the UsageRequest message is not recognized by the peer element.

6.11.4 Usage Indication

Report call details and usage information. This message is sent with respect to the last *UsageSpecification* element received by the peer element concerning the call.

Field	Description
callInfo	The call for which the indication applies.
accessTokens	The access tokens for the call. These are the tokens that were received in the address template used for the call, and propagated in the AccessRequest/Setup message for the same call.
senderRole	The role of the sender of the indication: • originator – originating party. • destination – terminating party. • nonStandard – other.
usageCallStatus	The current status of the call: • preConnect. • callInProgress. • callEnded. • registrationLost.

srcInfo	E.164 or e-mail address of the calling party.
destAddress	E.164 or e-mail address for the called party.
startTime	The time the call started, in UTC format. Relevant only for calls that passed the set-up stage. For multiple media types used in the call, each media type should report a different startTime, corresponding to the time at which that media stream started. For periodic messages, startTime should correspond with the endTime of the previous message.
endTime	The time the call ended, in UTC format. Relevant only for ended calls. For multiple media types used in the call, each media type shall report a different endTime corresponding to the time at which that media stream ended. For periodic messages, endTime is the time which ends a reporting period.
terminationCause	The reason for the end of the call. Relevant only for ended calls.
usageFields	Set of fields of information. Each field is represented by a *UsageField* which can be a standard or non-standard. Standard UsageFields are for future study.

6.11.5 Usage Indication Confirmation

The UsageIndicationConfirmation message is sent in response to a UsageIndication message, indicating the recipient accepted the indication as reported.

6.11.6 Usage Indication Rejection

The UsageIndicationRejection message is sent in response to a UsageIndication message, indicating that the recipient rejected the indication and will ignore it.

Field	**Description**
reason	This is the reason why the peer element rejected the UsageIndication message. Choices are: • unknownCall – The call specified in the UsageIndication is not a recognized call. • incomplete – The UsageIndication did not contain all the information required by the UsageSpecification that applies to this UsageIndication. • security – The UsageIndication did not meet the recipient's security requirements. • noServiceRelationship – The recipient will exchange this information only after establishment of a service relationship. • undefined – The reason for rejecting the UsageIndication does not match any of the other choices. • neededFeature – Indicates that the request failed because the requesting entity did not support one or more needed generic features. The features that are needed are indicated in the neededFeatures field in the MessageCommonInfo field of the reply. • genericDataReason – Indicates that the real reason code is contained within the genericData field sent with this message. Note that the genericData may contain more than one reason code. • unknownServiceID – The serviceID field contained in the UsageIndication message is not recognized by the peer element.

6.12 Validations

6.12.1 Validation Request

A peer element that terminates a call can send a ValidationRequest message to another peer element to verify the validity of the origination of the call.

Field	Description
accessToken	Tokens received from the originator to prove access authorization for the call.
destinationInfo	Details about the destination of the call.
sourceInfo	This is information about the type of endpoint that originated the call.
callInfo	This provides identification of the particular call for which access authorization is requested.
usageSpec	If present, indicates a request from the peer element which is sending the message to receive usage indication regarding the validated call.

6.12.2 Validation Confirmation

Indicates that the call is validated. The requesting peer element may terminate the call. The validating peer element may indicate aliases to terminate the call.

Field	Description
destinationInfo	Alternative parameters for the destination to be used by the recipient peer element.
usageSpec	If present, indicates a request from the peer element which is sending the confirmation to receive usage indication regarding the validated call.

6.12.3 Validation Rejection

Indicates the call is not valid. The requesting peer element may not complete the call.

Field	Description
reason	This is the reason for rejecting the request. Choices are: • tokenNotValid – The access token supplied is not valid for the call. • security – The ValidationRequest did not meet the recipient's security requirements. • hopCountExceeded – The hop count reached zero and no information is available. • missingSourceInfo – The source information supplied was not sufficient to validate the call. • missingDestInfo – The source information supplied was not sufficient to validate the call. • noServiceRelationship – The recipient will exchange this information only after establishment of a service relationship. • undefined – The reason for rejecting the ValidationRequest does not match the other choices. • neededFeature – Indicates that the request failed because the requesting entity did not support one or more needed generic features. The features that are needed are indicated in the neededFeatures field in the MessageCommonInfo field of the reply.

	• genericDataReason – Indicates that the real reason code is contained within the genericData field sent with this message. Note that the genericData may contain more than one reason code.
	• unknownServiceID – The serviceID field contained in the ValidationRequest message is not recognized by the peer element.

6.13 Authentication

6.13.1 Authentication Request

A peer element can send an AuthenticationRequest message to another peer element to authenticate a user.

Field	**Description**
applicationMessage	Contains an application protocol message to be authenticated.

6.13.2 Authentication Confirmation

Indicates successful authentication. There are no specific fields defined for this message, the relevant information (tokens, cryptoTokens) is contained in the common part of the message.

6.13.3 Authentication Rejection

Indicates that authentication has failed.

Field	**Description**
reason	This is the reason for rejecting the request. Choices are:
	• security – The AuthenticationRequest did not meet the recipient's security requirements.
	• hopCountExceeded – The hop count reached zero and no information is available.
	• noServiceRelationship – The recipient will exchange this information only after establishment of a service relationship.
	• undefined – The reason for rejecting the AuthenticationRequest does not match the other choices.
	• neededFeature – Indicates that the request failed because the requesting entity did not support one or more needed generic features. The features that are needed are indicated in the neededFeatures field in the MessageCommonInfo field of the reply.
	• genericDataReason – Indicates that the real reason code is contained within the genericData field sent with this message. Note that the genericData may contain more than one reason code.
	• unknownServiceID – The serviceID field contained in the AuthenticationRequest message is not recognized by the peer element.
	• securityWrongSyncTime – The sender found a security problem with inappropriate timestamps. This could be due to a problem with the time server, lost synchronization or excessive network delay.
	• securityReplay – A replay attack has been encountered. This is the case when the same sequence number occurs more than once for a given timestamp.

- securityWrongGeneralID – Indicates a mismatch of the general ID in the message. This could be caused by wrong addressing.
- securityWrongSendersID – Indicates a mismatch of the sender's ID in the message. This could be caused by erroneous user's entry.
- securityMessageIntegrityFailed – The integrity/signature check failed. This could be caused by a wrong or mistyped password during the initial request or a wrong private/public key applied, or by an encountered active attack.
- securityWrongOID – Indicates any mismatch in token OIDs (clear or crypto token) or crypto algorithm OIDs. Different security algorithms/profiles are implemented.

Annex A

Message Syntax

```
H501-MESSAGES DEFINITIONS AUTOMATIC TAGS ::=
BEGIN

IMPORTS
AuthenticationMechanism,
TimeStamp,
ClearToken
     FROM H235-SECURITY-MESSAGES

AliasAddress,
TransportAddress,
ReleaseCompleteReason,
ConferenceIdentifier,
CallIdentifier,
CryptoH323Token,
CryptoToken,
EndpointType,
GatekeeperIdentifier,
GloballyUniqueID,
NonStandardParameter,
NumberDigits,
PartyNumber,
SupportedProtocols,
TransportQOS,
VendorIdentifier,
IntegrityMechanism,
ICV,
FeatureSet,
GenericData,
EnumeratedParameter,
ServiceControlSession,
CircuitInfo,
CircuitIdentifier
     FROM H323-MESSAGES;
```

```
Message ::= SEQUENCE
{
     body     MessageBody,
     common   MessageCommonInfo,
     ...
}

MessageBody ::= CHOICE
{
     serviceRequest                ServiceRequest,
     serviceConfirmation           ServiceConfirmation,
     serviceRejection              ServiceRejection,
     serviceRelease                ServiceRelease,
     descriptorRequest             DescriptorRequest,
     descriptorConfirmation        DescriptorConfirmation,
     descriptorRejection           DescriptorRejection,
     descriptorIDRequest           DescriptorIDRequest,
     descriptorIDConfirmation      DescriptorIDConfirmation,
     descriptorIDRejection         DescriptorIDRejection,
     descriptorUpdate              DescriptorUpdate,
     descriptorUpdateAck           DescriptorUpdateAck,
     accessRequest                 AccessRequest,
     accessConfirmation            AccessConfirmation,
     accessRejection               AccessRejection,
     requestInProgress             RequestInProgress,
     nonStandardRequest            NonStandardRequest,
     nonStandardConfirmation       NonStandardConfirmation,
     nonStandardRejection          NonStandardRejection,
     unknownMessageResponse        UnknownMessageResponse,
     usageRequest                  UsageRequest,
     usageConfirmation             UsageConfirmation,
     usageIndication               UsageIndication,
     usageIndicationConfirmation   UsageIndicationConfirmation,
     usageIndicationRejection      UsageIndicationRejection,
     usageRejection                UsageRejection,
     validationRequest             ValidationRequest,
     validationConfirmation        ValidationConfirmation,
     validationRejection           ValidationRejection,
     ...,
     authenticationRequest         AuthenticationRequest,
     authenticationConfirmation    AuthenticationConfirmation,
     authenticationRejection       AuthenticationRejection
}

MessageCommonInfo ::= SEQUENCE
{
     sequenceNumber           INTEGER (0..65535),
     annexGversion            ProtocolVersion, -- set to "H.225.0 Annex G V2"
     hopCount                 INTEGER (1..255),
     replyAddress             SEQUENCE OF TransportAddress OPTIONAL,
                              -- Must be present in request
     integrityCheckValue      ICV OPTIONAL,
     tokens                   SEQUENCE OF ClearToken OPTIONAL,
     cryptoTokens             SEQUENCE OF CryptoH323Token OPTIONAL,
     nonStandard              SEQUENCE OF NonStandardParameter OPTIONAL,
     ...,
     serviceID                ServiceID   OPTIONAL,
     genericData              SEQUENCE OF GenericData OPTIONAL,
     featureSet               FeatureSet OPTIONAL,
     version                  ProtocolVersion -- current H.501 protocol version
}
```

```
ServiceID ::= GloballyUniqueID


--
-- H.501 messages
--

ServiceRequest ::= SEQUENCE
{
     elementIdentifier       ElementIdentifier OPTIONAL,
     domainIdentifier        AliasAddress OPTIONAL,
     securityMode                 SEQUENCE OF SecurityMode OPTIONAL,
     timeToLive                   INTEGER (1..4294967295) OPTIONAL,
     ...,
     usageSpec                    UsageSpecification OPTIONAL
}

SecurityMode ::= SEQUENCE
{
     authentication          AuthenticationMechanism OPTIONAL,
     integrity                    IntegrityMechanism OPTIONAL,
     algorithmOIDs           SEQUENCE OF OBJECT IDENTIFIER OPTIONAL,
     ...
}


ServiceConfirmation ::= SEQUENCE
{
     elementIdentifier       ElementIdentifier,
     domainIdentifier        AliasAddress,
     alternates              AlternatePEInfo OPTIONAL,
     securityMode            SecurityMode OPTIONAL,
     timeToLive              INTEGER (1..4294967295) OPTIONAL,
     ...,
     usageSpec               UsageSpecification OPTIONAL
}


ServiceRejection ::= SEQUENCE
{
     reason                  ServiceRejectionReason,
     alternates              AlternatePEInfo OPTIONAL,
     ...
}

ServiceRejectionReason ::= CHOICE
{
     serviceUnavailable      NULL,
     serviceRedirected       NULL,
     security                NULL,
     continue                NULL,
     undefined               NULL,
     ...,
     unknownServiceID        NULL,
     cannotSupportUsageSpec  NULL, -- Cannot comply with proposed spec
     neededFeature           NULL,
     genericDataReason       NULL,
     usageUnavailable        NULL, -- Usage reporting not supported
     unknownUsageSendTo      NULL  -- Usage sendTo could not be resolved
}
```

```
ServiceRelease ::= SEQUENCE
{
    reason                  ServiceReleaseReason,
    alternates              AlternatePEInfo OPTIONAL,
    ...
}

ServiceReleaseReason ::= CHOICE
{
    outOfService            NULL,
    maintenance             NULL,
    terminated              NULL,
    expired                 NULL,
    ...
}



DescriptorRequest ::= SEQUENCE
{
    descriptorID            SEQUENCE OF DescriptorID,
    ...
}


DescriptorConfirmation ::= SEQUENCE
{
    descriptor              SEQUENCE OF Descriptor,
    ...
}


DescriptorRejection ::= SEQUENCE
{
    reason                  DescriptorRejectionReason,
    descriptorID            DescriptorID OPTIONAL,
   ...
}

DescriptorRejectionReason ::= CHOICE
{
    packetSizeExceeded      NULL,  -- use other transport type
    illegalID               NULL,  -- no descriptor for provided descriptorID
    security                NULL,  -- request did not meet security requirements
    hopCountExceeded        NULL,
    noServiceRelationship   NULL,
    undefined               NULL,
    ...,
    neededFeature           NULL,
    genericDataReason       NULL,
    unknownServiceID        NULL   -- The serviceID is not recognized by
                                   -- the peer element
}



DescriptorIDRequest ::= SEQUENCE
{
    ...
}
```

```
DescriptorIDConfirmation ::= SEQUENCE
{
    descriptorInfo     SEQUENCE OF DescriptorInfo,
    ...
}


DescriptorIDRejection ::= SEQUENCE
{
    reason             DescriptorIDRejectionReason,
    ...
}

DescriptorIDRejectionReason ::= CHOICE
{
    noDescriptors           NULL,  -- no descriptors to report
    security                NULL,  -- request did not meet security requirements
    hopCountExceeded        NULL,
    noServiceRelationship   NULL,
    undefined               NULL,
    ...,
    neededFeature           NULL,
    genericDataReason       NULL,
    unknownServiceID        NULL   -- The serviceID is not recognized by
                                   -- the peer element
}


DescriptorUpdate ::= SEQUENCE
{
    sender                  AliasAddress,
    updateInfo              SEQUENCE OF UpdateInformation,
    ...
}

UpdateInformation ::=   SEQUENCE
{
    descriptorInfo     CHOICE
      {
         descriptorID  DescriptorID,
         descriptor    Descriptor,
         ...
      },
    updateType         CHOICE
      {
         added         NULL,
         deleted       NULL,
         changed       NULL,
         ...
      },
    ...
}


DescriptorUpdateAck ::= SEQUENCE
{
    ...
}
```

```
AccessRequest ::= SEQUENCE
{
     destinationInfo    PartyInformation,
     sourceInfo         PartyInformation OPTIONAL,
     callInfo           CallInformation OPTIONAL,
     usageSpec          UsageSpecification OPTIONAL,
     ...,
     desiredProtocols   SEQUENCE OF SupportedProtocols OPTIONAL
}


AccessConfirmation ::= SEQUENCE
{
     templates               SEQUENCE OF AddressTemplate,
     partialResponse         BOOLEAN,
     ...,
     supportedProtocols      SEQUENCE OF SupportedProtocols OPTIONAL,
     serviceControl          SEQUENCE OF ServiceControlSession OPTIONAL
}


AccessRejection ::= SEQUENCE
{
     reason                  AccessRejectionReason,
     ...,
     serviceControl          SEQUENCE OF ServiceControlSession OPTIONAL
}

AccessRejectionReason ::= CHOICE
{
     noMatch                 NULL,  -- no template matched the destinationInfo
     packetSizeExceeded      NULL,  -- use other transport type
     security                NULL,  -- request did not meet security requirements
     hopCountExceeded        NULL,
     needCallInformation     NULL,  -- Call Information must be specified
     noServiceRelationship   NULL,
     undefined               NULL,
     ...,
     neededFeature           NULL,
     genericDataReason       NULL,
     destinationUnavailable  NULL,  -- Destination was resolved but is
                                    -- unavailable
     aliasesInconsistent     NULL,  -- Multiple aliases identify distinct
                                    -- destinations
     resourceUnavailable     NULL,  -- One or more required resources are
                                    -- unavailable
     incompleteAddress       NULL,  -- Destination cannot be distinctly
                                    -- identified
     unknownServiceID        NULL,  -- The serviceID is not recognized by
                                    -- the peer element
     usageUnavailable        NULL,  -- Usage reporting not supported
     cannotSupportUsageSpec  NULL,  -- Cannot comply with proposed spec
     unknownUsageSendTo      NULL   -- Usage sendTo could not be resolved
}



UsageRequest ::= SEQUENCE
{
     callInfo        CallInformation,
     usageSpec       UsageSpecification,
     ...
}
```

```
UsageConfirmation ::= SEQUENCE
{
    ...
}


UsageRejection ::= SEQUENCE
{
    reason    UsageRejectReason,
    ...
}


UsageIndication ::= SEQUENCE
{
    callInfo           CallInformation,
    accessTokens       SEQUENCE OF AccessToken OPTIONAL,
    senderRole         Role,
    usageCallStatus    UsageCallStatus,
    srcInfo            PartyInformation OPTIONAL,
    destAddress        PartyInformation,
    startTime          TimeStamp OPTIONAL,
    endTime            TimeStamp OPTIONAL,
    terminationCause   TerminationCause OPTIONAL,
    usageFields        SEQUENCE OF UsageField,
    ...
}

UsageField ::= SEQUENCE
{
    id                 OBJECT IDENTIFIER,
    value              OCTET STRING,
    ...
}


UsageRejectReason ::= CHOICE
{
    invalidCall              NULL,
    unavailable              NULL,
    security                 NULL,
    noServiceRelationship    NULL,
    undefined                NULL,
    ...,
    neededFeature            NULL,
    genericDataReason        NULL,
    unknownServiceID         NULL       -- The serviceID is not recognized by
                                        -- the peer element
}


UsageIndicationConfirmation ::= SEQUENCE
{
    ...
}


UsageIndicationRejection ::= SEQUENCE
{
    reason         UsageIndicationRejectionReason,
    ...
}
```

```
UsageIndicationRejectionReason ::= CHOICE
{
     unknownCall              NULL,
     incomplete               NULL,
     security                 NULL,
     noServiceRelationship    NULL,
     undefined                NULL,
     ...,
     neededFeature            NULL,
genericDataReason             NULL,
unknownServiceID              NULL      -- The serviceID is not recognized by
                                        -- the peer element
}

ValidationRequest ::= SEQUENCE
{
     accessToken         SEQUENCE OF AccessToken OPTIONAL,
     destinationInfo     PartyInformation OPTIONAL,
     sourceInfo          PartyInformation OPTIONAL,
     callInfo            CallInformation,
     usageSpec           UsageSpecification OPTIONAL,
     ...
}


ValidationConfirmation ::= SEQUENCE
{
     destinationInfo     PartyInformation OPTIONAL,
     usageSpec           UsageSpecification OPTIONAL,
     ...
}

ValidationRejection ::= SEQUENCE
{
     reason     ValidationRejectionReason,
     ...
}

ValidationRejectionReason ::= CHOICE
{
     tokenNotValid            NULL,
     security                 NULL,  -- request did not meet security requirements
     hopCountExceeded         NULL,
     missingSourceInfo        NULL,
     missingDestInfo          NULL,
     noServiceRelationship    NULL,
     undefined                NULL,
     ...,
     neededFeature            NULL,
     genericDataReason        NULL,
     unknownServiceID         NULL   -- The serviceID is not recognized by
                                     -- the peer element
}

RequestInProgress ::= SEQUENCE
{
     delay                    INTEGER (1..65535),
     ...,
     serviceControl           SEQUENCE OF ServiceControlSession OPTIONAL
}
```

```
NonStandardRequest ::= SEQUENCE
{
     ...
}

NonStandardConfirmation ::= SEQUENCE
{
     ...
}

NonStandardRejection ::= SEQUENCE
{
     reason        NonStandardRejectionReason,
     ...
}

NonStandardRejectionReason ::= CHOICE
{
     notSupported            NULL,
     noServiceRelationship   NULL,
     undefined               NULL,
     ...,
     neededFeature           NULL,
     genericDataReason       NULL,
     unknownServiceID        NULL -- The serviceID is not recognized by
                                  -- the peer element
}
UnknownMessageResponse ::= SEQUENCE
{
     unknownMessage     OCTET STRING,
     reason             UnknownMessageReason,
     ...
}

UnknownMessageReason ::= CHOICE
{
     notUnderstood      NULL,
     undefined          NULL,
     ...
}



AuthenticationRequest ::= SEQUENCE
{
     applicationMessage    ApplicationMessage, -- e.g. RAS message in
                                               -- ITU-T Rec. H.323
     ...
}

ApplicationMessage ::= OCTET STRING


AuthenticationConfirmation ::= SEQUENCE
{
     ...
}


AuthenticationRejection ::= SEQUENCE
{
     reason      AuthenticationRejectionReason,
     ...
}
```

```
AuthenticationRejectionReason ::= CHOICE
{
     security                     NULL,
     hopCountExceeded             NULL,
     noServiceRelationship        NULL,
     undefined                    NULL,
     neededFeature                NULL,
     genericDataReason            NULL,
     unknownServiceID             NULL,
     securityWrongSyncTime        NULL, -- time server problem or network delay
     securityReplay               NULL, -- replay attack encountered
     securityWrongGeneralID       NULL, -- wrong general ID
     securityWrongSendersID       NULL, -- wrong senders ID
     securityIntegrityFailed      NULL, -- integrity check failed
     securityWrongOID             NULL, -- wrong token OIDs or crypto alg OIDs
     ...
}



--
-- structures common to multiple messages
--

AddressTemplate ::= SEQUENCE
{
     pattern                 SEQUENCE OF Pattern,
     routeInfo               SEQUENCE OF RouteInformation,
     timeToLive              INTEGER (1..4294967295),
     ...,
     supportedProtocols      SEQUENCE OF SupportedProtocols OPTIONAL,
     featureSet              FeatureSet OPTIONAL
}

Pattern ::= CHOICE
{
     specific    AliasAddress,
     wildcard    AliasAddress,
     range       SEQUENCE
       {
       startOfRange     PartyNumber,
       endOfRange       PartyNumber
       },
     ...
}

RouteInformation ::= SEQUENCE
{
     messageType    CHOICE
       {
          sendAccessRequest  NULL,
          sendSetup          NULL,
          nonExistent        NULL,
          ...
       },
     callSpecific       BOOLEAN,
     usageSpec          UsageSpecification OPTIONAL,
     priceInfo          SEQUENCE OF PriceInfoSpec OPTIONAL,
     contacts           SEQUENCE OF ContactInformation,
     type               EndpointType OPTIONAL,
                        -- must be present if messageType = sendSetup
     ...,
     featureSet         FeatureSet OPTIONAL,
     circuitID          CircuitInfo OPTIONAL,
     supportedCircuits  SEQUENCE OF CircuitIdentifier OPTIONAL
}

ContactInformation ::= SEQUENCE
{
     transportAddress   AliasAddress,
     priority           INTEGER (0..127),
     transportQoS       TransportQOS OPTIONAL,
     security           SEQUENCE OF SecurityMode OPTIONAL,
     accessTokens       SEQUENCE OF AccessToken OPTIONAL,
     ...,
     multipleCalls      BOOLEAN OPTIONAL,
     featureSet         FeatureSet OPTIONAL,
     circuitID          CircuitInfo OPTIONAL,
     supportedCircuits  SEQUENCE OF CircuitIdentifier OPTIONAL
}

PriceInfoSpec ::= SEQUENCE
{
     currency           IA5String (SIZE(3)),    -- e.g. "USD"
     currencyScale      INTEGER(-127..127),
     validFrom          GlobalTimeStamp OPTIONAL,
     validUntil         GlobalTimeStamp OPTIONAL,
     hoursFrom          IA5String (SIZE(6)) OPTIONAL, -- "HHMMSS" UTC
     hoursUntil         IA5String (SIZE(6)) OPTIONAL, -- "HHMMSS" UTC
     priceElement       SEQUENCE OF PriceElement OPTIONAL,
     priceFormula       IA5String (SIZE(1..2048)) OPTIONAL,
     ...
}

PriceElement ::= SEQUENCE
{
     amount      INTEGER(0..4294967295), -- meter increment
     quantum     INTEGER(0..4294967295), -- each or part thereof
     units       CHOICE
       {
       seconds          NULL,
       packets          NULL,
       bytes            NULL,
       initial          NULL,
       minimum          NULL,
       maximum          NULL,
       ...
       },
     ...
}


Descriptor ::= SEQUENCE
{
     descriptorInfo     DescriptorInfo,
     templates          SEQUENCE OF AddressTemplate,
     gatekeeperID       GatekeeperIdentifier OPTIONAL,
     ...
}

DescriptorInfo ::= SEQUENCE
{
     descriptorID            DescriptorID,
     lastChanged             GlobalTimeStamp,
     ...
}
```

```
AlternatePEInfo ::= SEQUENCE
{
     alternatePE             SEQUENCE OF AlternatePE,
     alternateIsPermanent    BOOLEAN,
     ...
}

AlternatePE ::= SEQUENCE
{
     contactAddress          AliasAddress,
     priority                INTEGER (1..127),
     elementIdentifier       ElementIdentifier OPTIONAL,
     ...
}

AccessToken ::= CHOICE
{
     token          ClearToken,
     cryptoToken    CryptoH323Token,
     ...,
     genericData    GenericData
}

CallInformation ::= SEQUENCE
{
     callIdentifier     CallIdentifier,
     conferenceID       ConferenceIdentifier,
     ...,
     circuitID          CircuitInfo OPTIONAL
}


UsageCallStatus ::= CHOICE
{
     preConnect         NULL,     -- Call has not started
     callInProgress     NULL,     -- Call is in progress
     callEnded          NULL,     -- Call ended
     ...,
     registrationLost   NULL      -- Uncertain if call ended or not
}

UserInformation ::= SEQUENCE
{
     userIdentifier     AliasAddress,
     userAuthenticator  SEQUENCE OF CryptoH323Token OPTIONAL,
     ...
}


UsageSpecification ::= SEQUENCE
{
     sendTo      ElementIdentifier,
     when        SEQUENCE
       {
       never     NULL OPTIONAL,
       start     NULL OPTIONAL,
       end       NULL OPTIONAL,
       period    INTEGER(1..65535) OPTIONAL,    -- in seconds
       failures  NULL OPTIONAL,
       ...
       },
     required           SEQUENCE OF OBJECT IDENTIFIER,
     preferred          SEQUENCE OF OBJECT IDENTIFIER,
     ...,
     sendToPEAddress    AliasAddress OPTIONAL
}


PartyInformation ::= SEQUENCE
{
     logicalAddresses   SEQUENCE OF AliasAddress,
     domainIdentifier   AliasAddress OPTIONAL,
     transportAddress   AliasAddress OPTIONAL,
     endpointType       EndpointType OPTIONAL,
     userInfo           UserInformation OPTIONAL,
     timeZone           TimeZone OPTIONAL,
     ...
}


Role ::= CHOICE
{
     originator         NULL,
     destination        NULL,
     nonStandardData    NonStandardParameter,
     ...
}

TimeZone ::= INTEGER (-43200..43200)  -- number of seconds relative to UTC
                                      -- including DST if appropriate


TerminationCause ::= SEQUENCE
{
     releaseCompleteReason   ReleaseCompleteReason,
     causeIE                 INTEGER (1..65535) OPTIONAL,
     nonStandardData         NonStandardParameter OPTIONAL,
     ...
}



ProtocolVersion  ::=    OBJECT IDENTIFIER
     --   shall be set to
     --   {itu-t(0) recommendation(0) h(8) h-225-0(2250) annex(1) g(7)
     --   version(0) 2} in field annexGversion;
     --   {itu-t(0) recommendation(0) h(8) 501 version(0) 1}
     --   in field version


DescriptorID     ::=    GloballyUniqueID

ElementIdentifier  ::=  BMPString (SIZE(1..128))

GlobalTimeStamp    ::=  IA5String (SIZE(14))
                                 -- UTC,  in the form YYYYMMDDHHmmSS
                                 -- where YYYY = year, MM = month, DD = day,
                                 -- HH = hour, mm = minute, SS = second
                                 -- (for example, 19981219120000 for noon
                                 -- 19 December 1998)

--
-- REPOSITORY FOR APPLICATION SPECIFIC DATA --
--
-- H.225.0 Annex-G profile data
--
idAnnexGProfiles   INTEGER ::= 0
idAnnexGProfileA   INTEGER ::= 1
```

```
annexGProfileA EnumeratedParameter ::=
{
    id     standard:idAnnexGProfileA
}



END  -- of H501-MESSAGES
```

SERIES OF ITU-T RECOMMENDATIONS

Printed in Switzerland
Geneva, 2002