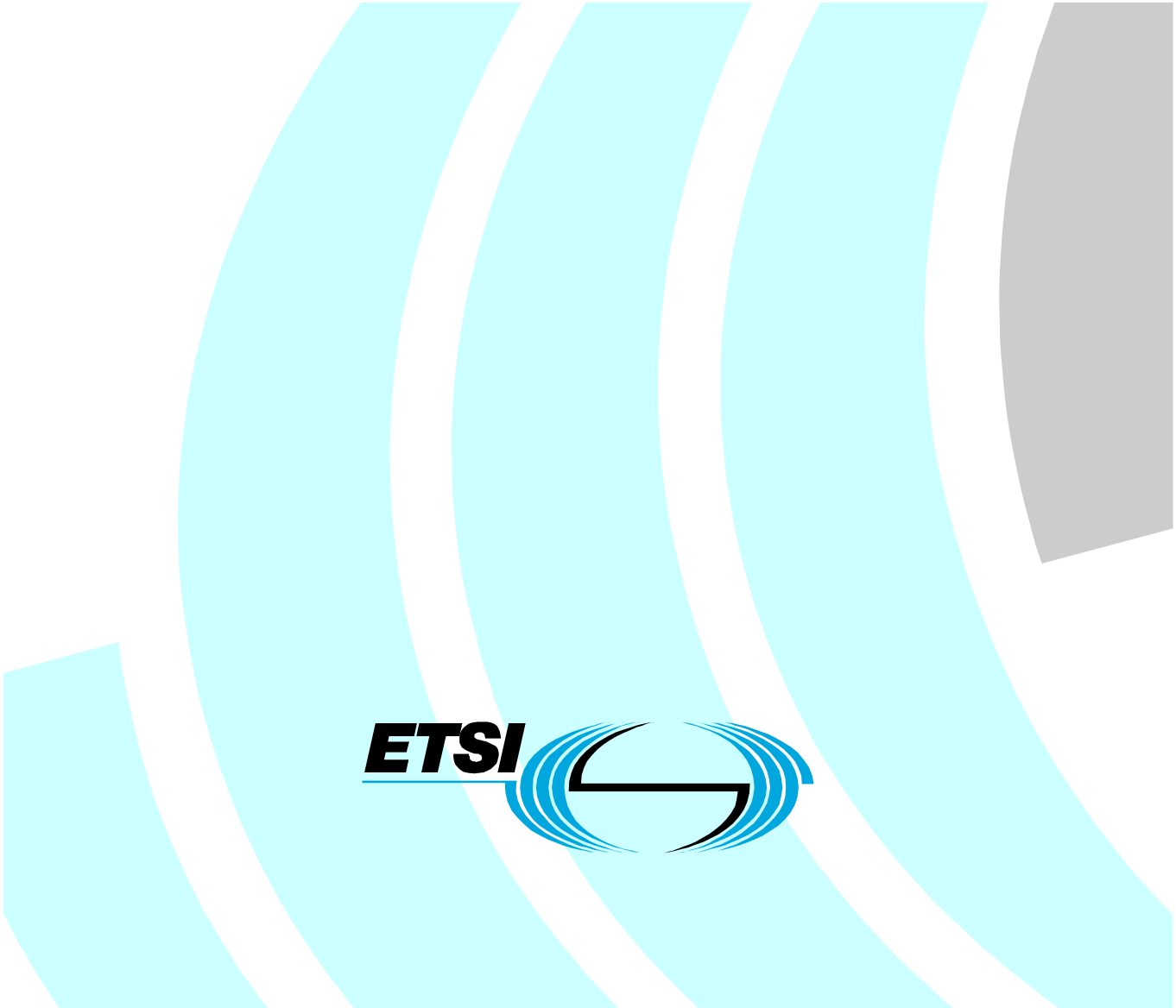
ETSI TS 102 108 V4.1.1 (2002-06)

Technical Specification

Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON) Release 4; H.248/MEGACO Profile for TIPHON reference point I3; InterConnect Function (ICF) control over reference point I3



Reference
DTS/TIPHON-03028R4

Keywords
architecture, configuration, H.248, internet, IP,
megaco, network, protocol, telephony, VoIP

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

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

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Contents

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://webapp.etsi.org/IPR/home.asp).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

This Technical Specification (TS) has been produced by ETSI Project Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON).

Introduction

The present document defines technology mappings of the meta-protocol to reference points in the transport plane. It specifies the technology mappings to reference point I3 as defined in TS 101 882 [2].

1 Scope

The present document describes a mapping of the meta-protocol for reference point I3, as described in annex D of TS 101 882 [2], to the ITU-T Recommendation H.248 [1]/IETF MEGACO protocol, to produce an interoperable profile (subset) of H.248/MEGACO augmented by the TPC Extension package defined in TS 101 332 [5].

It may be used for the control of equipment within an IP network such as Firewalls, Address Translators, Middleboxes and also other network edge devices.

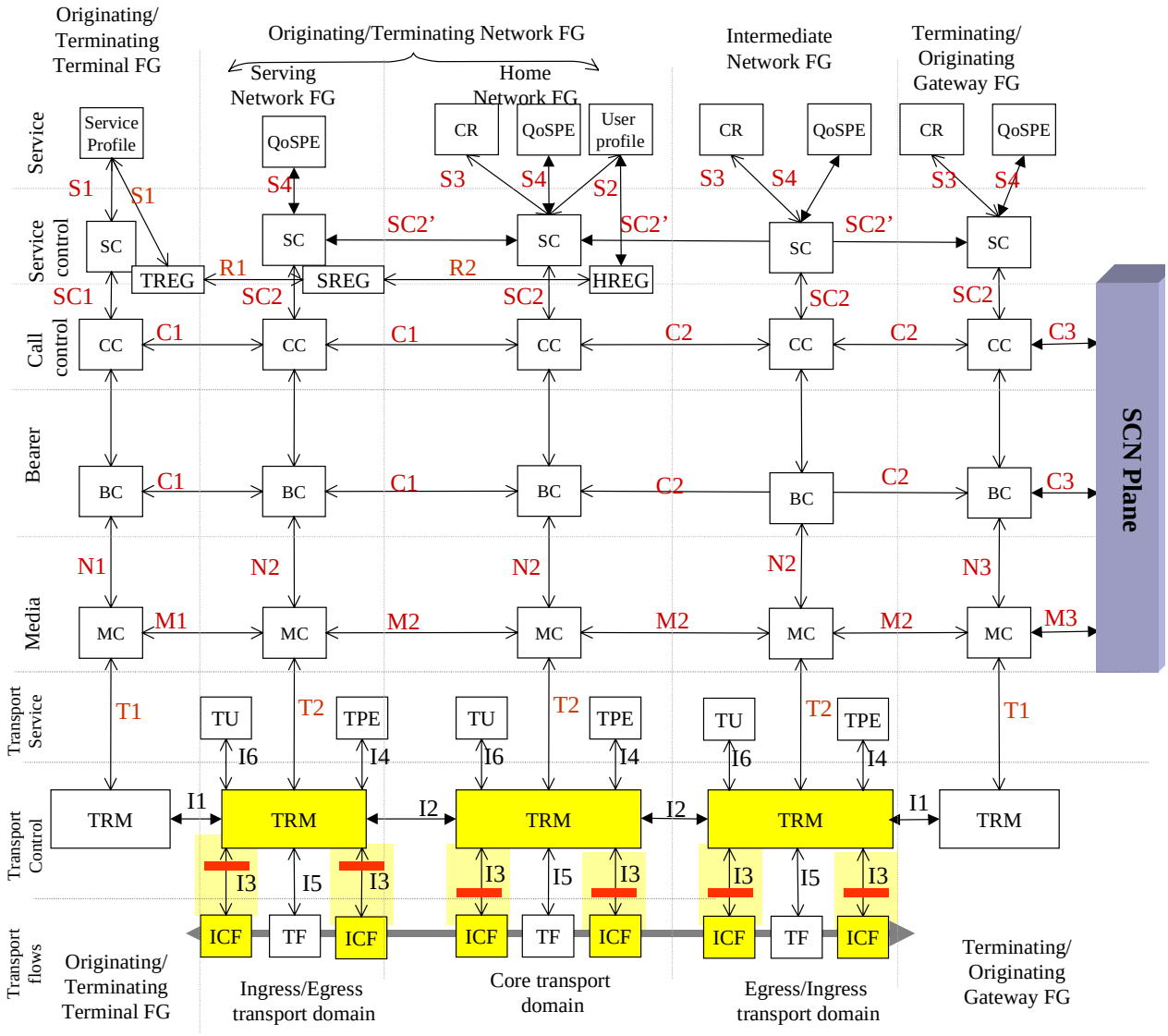


Figure 1a: TIPHON architecture with reference point I3 highlighted

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication and/or edition number or version number) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies.

[1] ITU-T Recommendation H.248 (2000): "Gateway control protocol".

[2] ETSI TS 101 882: "Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON) Release 3; Protocol Framework Definition; General (meta-protocol)".

[3] ETSI TS 101 314: "Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON) Release 3; Abstract Architecture and Reference Points Definition; Network Architecture and Reference Points".

[4] IETF RFC 2327 (April 1998): "SDP Session Description Protocol".

[5] ETSI TS 101 332: "Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON) Release 4; Interface Protocol Requirements Definition; TIPHON Extended H.248/MEGACO Package (EMP) Specification; ICF Control over Reference Point".

[6] IETF RFC 2205: "Resource ReSerVation Protocol (RSVP) - Version 1 Functional Specification".

[7] ETSI TS 101 329-3: "Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON) Release 3; End-to-end Quality of Service in TIPHON systems; Part 3: Signalling and control of end-to-end Quality of Service (QoS)".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the following terms and definitions apply:

charging: process of determining the amount of money a user shall pay for usage of a certain service

flow: single data stream, identified by a set of characteristic values (source address, source port, destination address, destination port, protocol number)

functional entity: entity in a system that performs a specific set of functions

functional group: collection of functional entities within a domain

NOTE: In TIPHON systems functional groups are used to structure the necessary functionality to offer IP telephony services across domains.

IP endpoint: device that originates or terminates the IP based part of a call

NOTE: Endpoints include H.323 clients, and IP telephony gateways.

IP network: packet transport network comprising one or more transport domains each employing the IP protocol

middlebox: firewall or NAT device which is coupled to a MIDCOM server, which offers the firewall/NAT services to clients

network: telecommunications network that provides telecommunications services

protocol: set of semantics, syntax and procedures which govern the exchange of information across an interface

Switched Circuit Network (SCN): telecommunications network, e.g. Public Switched Telephone Network (PSTN), Integrated Services Digital Network (ISDN), and General System for Mobile communications (GSM), that uses circuit-switched technologies for the support of voice calls

NOTE: The SCN may be a public network or a private network.

TIPHON compliant: entity that complies with the mandatory requirements identified in the TIPHON requirements documents together with compliance to the parts of the TIPHON specifications in which these requirements are embodied

3.2 Abbreviations

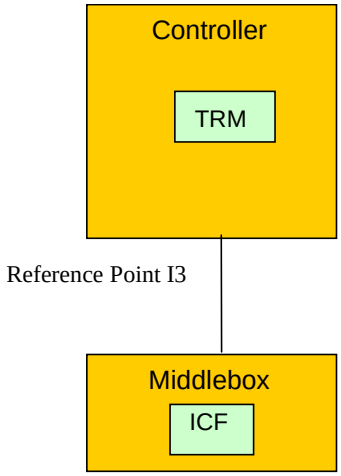
For the purposes of the present document, the following abbreviations apply:

EMP	Extended Megaco Package
GK	GateKeeper
ICF	InterConnect Function
IP	Internet Protocol
MC	Media Control
MB	MiddleBox
QoS	Quality of Service
RAS	Registration Admission and Status
RSVP	Resource reseVation Set-up Protocol
RTCP	RealTime Conferencing Protocol
RTP	RealTime transfer Protocol
SCN	Switched Circuit Networks
SDL	Specification and Description Language
TOS	Type Of Service
TRM	Transport Resource Manager

4 Scope of the H.248 ICF profile

The present document describes a profile of H.248 that mandates certain optional parts of the protocol and does not mandate certain other options. Here the H.248 protocol is used to convey only flow properties to edge devices (ICFs) in the transport plane. Used solely for this purpose, H.248/MEGACO, reduces to a small sub-set of the full protocol.

4.1 Use of the H.248 ICF profile



NOTE: This profile of H.248/MEGACO is used to control Middlebox devices in the transport plane. For the purposes of the present document a Middlebox is defined as a physical implementation of an InterConnect Function (ICF) TS 101 314 [3] and TS 101 329-3 [7]. A controller is defined as an implementation of control functionality. The control relationship between a Middlebox and its controller is shown in figure 1b.

Figure 1b: Entities involved in Control over the I3 Reference point

4.2 Functionality supported by the Profile

The I3 reference point controls the ICF functions in a transport domain and will be under the control of the operator or administrator of that domain. Instructions from the TRM (here the controller) to the ICF will relate to admission control in the case of firewalls, network addresses in the case of Address Translators or control signals for QoS mechanisms such as RSVP or DiffSDerv.

4.3 Service capabilities supported

TIPHON Release 4 service capabilities supported by the profile are basic bearer and QoS bearer set-up.

4.4 General note on compliance

Any options in the H.248 protocol not mentioned in the present document MAY be supported by the Middlebox or controller, however are outside the scope of the present document.

4.5 Error handling

General use of the error codes can be found in clause A.2.

If an H.248 information element is received (descriptor or parameter), which is not specified in this profile, the receiver shall ignore the information element and act as if the information element were not received.

If an unknown H.248 command or an information element not specified in this profile is received, the receiver shall ignore the command and send an appropriate error (443 - unsupported or unknown command or 444 - unsupported or unknown descriptor, 445 - unsupported or unknown property, 446 - unsupported or unknown parameter).

If an H.248 command with a mandatory information element missing is received, the receiver shall act as if the information element was received carrying the default values, or reject with the appropriate error message if there is no default value specified in the H.248 standard.

If an H.248 information element is received with syntactically invalid contents the receiver shall:

- if the information element is optional, ignore the information element; or
- if the information element is mandatory, act as if the information element was received correctly coded carrying the default values and reject/fail if there is no default value specified in the standard.

If an H.248 information element is received with a value not allowed within the context of the present document, the receiver shall:

- if the information element is optional, pass on, but otherwise ignore the information element; or
- if the information element is mandatory reject/fail.

NOTE: The security policy of an operator's network or the security policy implemented in a network element may override the error handling as described above.

4.6 Security

Security issues associated with the use of this profile are outside the scope of the present document. It is intended that they will be included in future releases of the present document.

5 Setting up of the controller-middlebox control interface

Clause 11 of ITU-T Recommendation H.248 [1] shall apply without restrictions. This may change in future releases of the present document.

6 Use of the profile for the establishment of transport flows

The basic messages of H.248 used in this profile are here defined as well as certain H.248 packages. This clause describes the messages used in the profile.

6.1 Process used

This clause provides the mapping of the flows in TS 101 882 [2] annex D, to H248 message flows. The parameter and error reason definitions in TS 101 882 [2] annex D are aligned to ensure interoperability of these values across the mappings of these reference points. In general each of the primitives defined on the T/I reference points in TS 101 882 [2] each map to one or more commands in H.248. General note on source filtering used in the scenarios SDP (see RFC 2327 [4]) specifies in clause B.2.1 a means for specifying unicast sessions. This provides the means to encode the originator MpoA. In clause B.2 it is shown how the recvonly and send only attribute are used for specifying the sender side behaviour of media flows.

6.2 Packages supported

The following packages shall be supported in this profile:

- Generic package (ITU-T Recommendation H.248 [1], clause E.1)
- Base Root package (ITU-T Recommendation H.248 [1], clause E.2)
- Network package (ITU-T Recommendation H.248 [1], clause E.11)
- EMP package TS 101 332 [5]
- If RTP statistics are to be supported, the RTP package (ITU-T Recommendation H.248 [1], clause E.12)

6.3 Required H.248/MEGACO messages and parameters

Annex A describes the codepoint mapping between H.248/MEGACO and the TIPHON meta-protocol TS 101 882 [2]. From this mapping follows that the following elements from H.248/MEGACO are to be supported.

6.3.1 Messages

- TransactionRequest.Add
- TransactionReply.Add
- TransactionRequest.Modify
- TransactionReply.Modify
- TransactionRequest.Subtract
- TransactionReply.Subtract

6.3.2 Parameters

- ContextId
- LocalDescriptor
- RemoteDescriptor
- TerminationID

6.3.3 Error values

- 411 – The transaction refers to an unknown ContextId
- 412 – No ContextIDs available
- 430 – Unknown TerminationID
- 432 – Out of TerminationIDs or No TerminationID available
- 432 – Put of terminationIDs or no terminationID available
- 433 – Termination ID is already in a contect
- 444 – Unsupported or unknown descriptor
- 510 – Insufficient resources
- 526 – Insufficient bandwidth

6.3.4 General notes on option use in H.248

The TIPHON semantics require that when multiple alternatives are returned by the Middlebox to the controller for a termination the Middlebox shall support whichever one the controller chooses. Failure to support parameters for a termination offered earlier is considered a fault.

H.248 does not have semantics as strict as those prescribed by TIPHON. To align the semantics, **ReserveGroup** needs to be set for all alternative transport flow descriptions that are requested from the Middlebox and that **ReserveValue** needs to be set for all request where the Middlebox is given the choice to provide flow descriptions.

The optional ModemDescriptor, MuxDescriptor. EventsDescriptor, DigitMapDescriptor, AuditDescriptor are not used in this profile.

A transaction level timer shall be provided.

The flows in TS 101 882 [2] use a **InvokingControllerReference**, is mapped to StreamID in H.248.

The contextID is a value that is local to the Middlebox-controller relationship and is not mapped to the meta protocol.

The TIPHON type **trReservedTransportReference** and **trEstablishedTransportReference** shall be mapped to the H.248 TerminationID.

Code point mapping for the **Local** and **Remote Descriptors** is provided in the annexes.

6.4 Default transport setup procedure

The transport plane allows reservation and allocation of resources for transport of a packet stream (e.g. to reserve processing capability in routers and on links between them). See TS 101 882 [2] for a full definition of the primitives involved .

NOTE: The following text is derived from the SDL in TS 101 882 [2]. Should there be discrepancies between TS 101 882 [2] and the present document, the text in TS 101 882 [2] shall take precedent.

The transport element shall establish the transport resources required to support the transport flows required. If so required, the transport elements shall establish a QoS controlled transport capability in accordance with the QoS parameters identified.

Step	TS 101 882 [2] Primitive	meaning
		MC layer decides it needs transport
1	**TransportReservationRequest**	asks the ICF/TRM to support it with a QoS-enabled transport stream and wants to know what parameters it can support there.
2	**TransportReservationConfirm**	the ICF/TRM acknowledges to the MC, providing the possible bearer characteristics for the MC to decide upon. The MC-entity hereby **commits** to provide the flows as identified in this message.
		The MC answers to BC….
3	**TransportEstablishementRequest**	the MC informs the ICF/TRM of the choice.
4	**TransportEstablishmentConfirm**	the ICF/TRM acknowledges the **establishment** of the transport flows.
5	**TransportReleaseRequest**	The ICF/TRM is requested to release the transport flows.
6	**TransportReleaseConfirm**	The ICF/TRM confirms the release of the transport flows.

6.5 Example flows for the use of this profile

Annexes C and D provide examples of the use of this profile for firewall control. The Middlebox can operate in several scenarios, these scenarios describe how the generic service offered over the T2 and I3 reference points can be used to achieve certain specific goals.

Annex C describes the following examples:

- C.1 Basic scenario;
- C.2 Basic scenario with address translation (the legacy mode);
- C.3 With address translation and with source delivery;
- C.4 Without address translation and with source filtering;
- C.5 With address translation and source filtering (the preferred TIPHON media mode);
- C.6 Explicit RTCP addressing;
- C.7 Signalling: Example flow for a H323 RAS control pin-hole.

Clause D.1 explains the option to mute the forward transport path until the call has been established.

6.6 QoS flows

The TIPHON semantics demand that a Middlebox provides QoS when the QoS parameters (maximum packet size, packet rate, delay budget, packet delay variation, packet loss) are set. The H.248 binary encoding supports the QoS parameters to be conveyed. Clause B.4 addresses the text encoding for this.

6.6.1 QoS on flows through the middlebox

If the QoS parameters (maximum packet size, packet rate, delay budget, packet delay variation, packet loss) are specified on the LocalDescriptors the middlebox is required to treat the incoming flow in accordance with the QoS parameters given.

6.6.1.1 QoS policing of incoming flows

If the MiddleBox (MB) is to police the rate of incoming packet flows. The Extended Megaco Package (EMP) value `tokenrate` shall be set to the meta-protocol value `packetRate.` If this value is set the middlebox shall exhibit the behaviour as specified by TS 101 332 [5], clause 4.1.2.

6.6.2 QoS on outgoing flows

Establishment of QoS on outgoing flows is (unfortunately) technology dependent.

6.6.2.1 ToS tagging of outgoing flows

If the middlebox supports this option, the controller shall set the ToS value as per clause B.3.1 on the RemoteDescriptor of the flow. The middlebox shall set this value on each of the outgoing packets of the flow.

6.6.2.2 RSVP

The RSVP flowspec values shall be derived from the QoS parameters (maximum packet size, packet rate, delay budget, packet delay variation, packet loss) when these values are specified on the RemoteDescriptors. The mapping between the TIPHON QoS values and the RSVP flowsspec shall be done in accordance with the RSVP specification (see RFC 2205 [6]).

6.7 ToS filtering on incoming flows

If the middlebox supports this option, the controller shall set the ToS value as per clause B.3.1 on the LocalDescriptor of the flow. The middlebox shall match the packets against this value.

7 Management operations

TIPHON release 3 does not prescribe any management operations on Middlebox. Behaviour regarding ServiceChange and Restart shall therefore be as specified by ITU-T Recommendation H.248 [1].

Annex A (normative): Mapping with meta-protocol

A.1 Code point mapping

This clause describes a generic mapping that is generally used for all primitives and parameters.

A.1.1 Primitive mappings

TIPHON primitives	**H.248 message types**
TransportReservationRequest	TransactionRequest.Add
TransportReservationConfirm	TransactionReply.Add
TransportEstablishmentRequest	TransactionRequest.Modify
TransportEstablishmentConfirm	TransactionReply.Modify
TransportReleaseRequest	TransactionRequest.Subtract
TransportReleaseConfirm	TransactionReply.Subtract
TIPHON primitives with data elements	**H.248 data elements**
TransportReservationRequest	TransactionRequest.Add
TransportHandleType (requested handle)	ContextId (see note 3)
SET OF *TransportDescriptorType* (rx)	LocalDescriptor
SET OF *TransportDescriptorType* (tx)	RemoteDescriptor
TransportReservationConfirm	TransactionReply.Add
TransportHandleType (requested handle)	ContextId (see note 3)
TransportStatusType	**See table of error messages**
SET OF TransportDescriptorWithHandle	LocalDescriptor, RemoteDescriptor
TransportEstablishmentRequest	TransactionRequest.Modify
TransportHandleType (reserved handle)	ContextId (see note 3)
TransportEstablishmentConfirm	TransactionReply.Modify
TransportHandleType (reserved handle)	ContextId (see note 3)
TransportStatusType	**See table of error messages**
TransportHandleType (established handle)	TerminationID
TransportReleaseRequest	TransactionRequest.Subtract
TransportHandleType (established handle)	TerminationID
TransportReleaseConfirm	TransactionReply.Subtract
TransportHandleType (established handle)	TerminationID
TransportStatusType	**See table of error messages**

A.1.2 Binary encoding

A.1.2.1 For an IP-based transport flow

TIPHON data elements	H.248 data elements
TransportDescriptorWithHandle	
TransportHandleType	ContextId (see note 3)
TransportDescriptorType (rx)	LocalDescriptor
TransportDescriptorType (tx)	RemoteDescriptor
TransportStatusType	**See table of error messages**
TransportHandleType	TerminationID
TransportDescriptorType	
GenericQoSDescriptorType	
SpecificQoSDescriptorType	<not supported>
MpoAType (originator)	<not supported>
MpoAType (destination)	
GenericQoSDescriptorType	
delay budget	PropDelay
Packet rate	Bitrate (shall be computed as follows: packetRate*MaximumPacketSize/100)
MaximumPacketSize	
Packet delay variation (jitter in milliseconds)	Jitterbuffer
packet loss	<not supported> the Middlebox shall support an adequately low number
IPv4Type	
Address	IPv6
Protocol	PortType (value =1 (UDP), RTP is implied)
Port	Port
IPv6Type	
Address	IPv4
Protocol	PortType (value = 1 (UDP), RTP is implied)
Port	Port
MpoAType	
iPv4	Clause C.6, tag value 6001
IPv6	Clause C.6, tag value 6002
MpoAProtocolType	
ENUMERATED (list from STD02)	See above

A.1.3 Text encoding

A.1.3.1 For an IP-based transport flow

TIPHON data elements	H.248 data elements
TransportDescriptorWithHandle	
TransportHandleType	ContextId (see note 3)
TransportDescriptorType (rx)	LocalDescriptor (for the contents see annex D)
TransportDescriptorType (tx)	RemoteDescriptor (for the contents see annex D)
TransportStatusType	**See table of error messages**
TransportHandleType	TerminationID

A.2 Error reasons

The following table provides a list of H.248 error reasons that a Middlebox shall use and their corresponding TIPHON meanings.

TIPHON reason	TIPHON diagnostic	H.248 error reason
RequestedHandle invalid		444 – Unsupported or unknown descriptor
RequestedHandle unavailable	CODEC unavailable QoS not available	510 – Insufficient resources 526 – Insufficient bandwidth
RequestedHandle does not exist		411 – The transaction refers to an unknown ContextId
Insufficient resources	no more bearers no more transport flows	412 – No ContextIDs available 432 – Out of TerminationIDs or No TerminationID available
TransportHandle does not exist		430 – Unknown TerminationID
TransportHandle invalid		433 – Termination ID is already in a contect
TransportHandle unavailable		432 – Put of terminationIDs or no terminationID available

Please note that TIPHON reference point T and I reject primitives usually pertain to a **TransportReference** within a **InvokingControllerReference**. H.248 error codes are uses as normal commands and hence apply to a particular stream or termination and hence carry the context ID and termination ID to which the error message pertains.

Annex B (normative): Meta-protocol to SDP mapping

SDP (see RFC 2327 [4]) is a description language for transport sessions. This mapping therefore has no states just the codepoints.

NOTE: Not all of these code points are used in this profile, e.g. all CODEC related parameters. These have been retained here however for completeness.

B.1 Mappable fields

The Protocol Version field (v = line) shall contain 0 as per page 8 of RFC 2327 [4].

The Origin field (o = line) shall be filled according to page 8 of RFC 2327 [4].

The Session Field (s = line) shall be filled according to page 9 of RFC 2327 [4].

The connection Field (c = line) shall contain the `address` value of the `destinationMpoA` in TS 101 882 [2].

- In the case of IPv4 and IPv6 addresses this shall take the form as prescribed on page 33 of RFC 2327 [4].
- Other cases are undefined in RFC 2327 [4] and may be extended in future versions of either RFC 2327 [4] or the present document.

The Bandwidth field (b = line) shall contain the value of `maxPacketSize` times `packetRate` divided by 8 `AS` shall be used as the bandwidth modifier.

The time filed (t = line) shall be filled according to page 14 of RFC 2327 [4]

The Transport field (m = line) shall be filled as follows:

- `<transport>` shall contain the type of transport flow. For Release 3 this shall be `audio` for audio flows and `control` for signalling flows.
- `<port>` shall contain `port` value of the `destinationMpoA` in TS 101 882 [2].
- `<transport>` shall contain the text RTP/AVP if RTCP flows are to be created by the middlebox and `UDP` for media without RTCP.
- `<fmt list>` shall be ignored by the middlebox.

Following attributes shall be used:

- `a=rtpmap:`

  `<payload type>` is identical to the value in <fmt list>

  `<encoding name>` shall be ignored by the middlebox

  `<clock rate>` equals the meta protocol values `packetRate` * `framesPerPacket` for this transport flow

  `<encoding parameters>` is filled per page 21 of RFC 2327

- `a=ptime: shall contain the meta protocol value of framesPerPacket`
- `a=fmtp:`

  `<format>` is identical to the value in <fmt list>

<format specific parameters> shall contain the value of the meta protocol field codecSpecificParameters

B.2 Specifying the OriginatorMpoA

SDP specifies in page 23 of RFC 2327 [4] a means for specifying unicast sessions. This provides the means to encode the
Originator MpoA. The recvonly and send only attribute are specifying the sender side behaviour of this message.

- The receive-only stream specifies the IP address and port where RTP should be sent to. Implicitly (port + 1) is the port where the RTCP should be sent to.
- The send-only stream specifies the IP address and port where RTP will be sent from. Implicitly (port + 1) is the port where the RTCP will be sent from.

EXAMPLE:

```
m=audio 1110 RTP/AVP 0
c=IN IP4 1.1.1.1
a=recvonly
m=audio 2220 RTP/AVP 0
c=IN IP4 2.2.2.2
a=sendonly
```

The sender of this SDP specifies that it wants to receive RTP packets on 1.1.1.1:1110 and send RTCP packets from 1.1.1.1:1111. It will send RTP packets from 2.2.2.2:2220 and receive RTCP packets on 2.2.2.2:2221.

NOTE: These semantics are compatible with current use of SDP. In the example above this leads to a scenario where a multihomed host receives RTP on 1.1.1.1 and sends the RTCP feedback on the stream from 2.2.2.2. The feedback on RTP that is sent from 2.2.2.2 will be received on 1.1.1.1. A more logical semantics would be that the RTCP is sent from (port in recvonly + 1) and received on (port in sendonly + 1). This way the RTP and related RTCP stream are neatly tied together.

B.3 Specifying QoS mechanism specific values

B.3.1 DiffServ Type of Service (ToS) value

When DiffServ is used its ToS value is provided in the meta-protocol value specificQoSDescriptor. It shall be encoded in SDP as follows:

a=X-ToS:<value>

NOTE: ToS is only valid for the vertical interfaces. This shall not be forgotten when this text is re-used for the upcoming general SDP profile deliverable.

B.3.2 RSVP

The RSVP flowspec values shall be derived from the SDP transport parameters in accordance with the RSVP specification (see RFC 2205 [6]).

B.4 Unmappable fields

At this time SDP has no provisions for the following meta-protocol values:

- DelayBudget,
- packetDelayVariation,
- packetLoss.

TIPHON intends to address all SDP related QoS issues and will register a coherent set of code points in SDP with IANA for these values.

Annex C (informative): TIPHON template usage scenarios

This clause provides template TIPHON messages for the profile on reference point I3.

The Middlebox can operate in several scenarios, these scenarios describe how the generic service offered over the T2 and I3 reference points can be used to achieve certain specific goals. The present document describes the following examples:

- C.1 Basic scenario.
- C.2 Basic scenario with address translation (the legacy mode).
- C.3 With address translation and with source delivery.
- C.4 Without address translation and with source filtering.
- C.5 With address translation and source filtering (the preferred TIPHON media mode).
- C.6 Explicit RTCP addressing.
- C.7 Signalling: Example flow for a H323 RAS control pin-hole.

C.1 Basic flow without address translation without source filtering

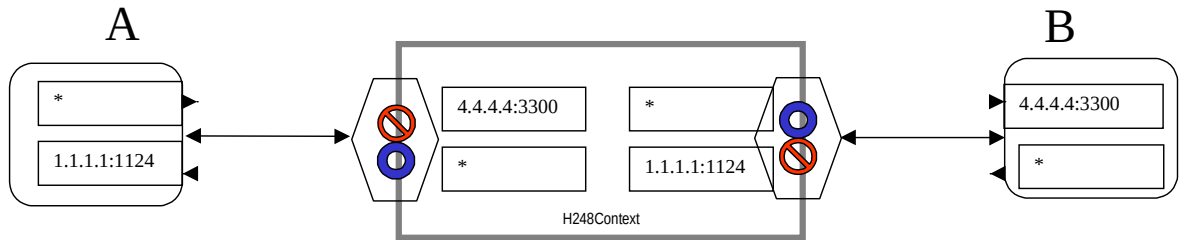


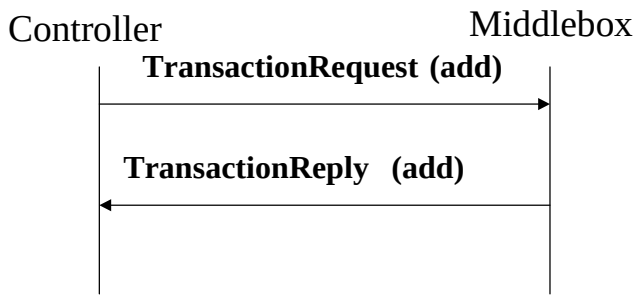
Figure C.1: Example H.248 context

Each termination sinks/sources one bi-directional stream with each a LocalDescriptor, RemoteDescriptor and a LocalControlDescriptor. The StreamIds are used to link the streams that are connected.

The following table provides the addresses and ports used in this figure:

Stream	StreamId	Address in LocalDescr Entering the middlebox	Addresses in RemoteDescr Leaving the middlebox.
A to B	1	**Source =*** **Sink = 4.4.4.4:3300**	**Source = *** **Sink = 4.4.4.4:3300**
B to A	1	**Source = *** **Sink = 1.1.1.1:1124**	**Source = *** **Sink = 1.1.1.1:1124**

The exchange above maps to the following H.248-EMP message flow.



1) Controller→Middlebox:

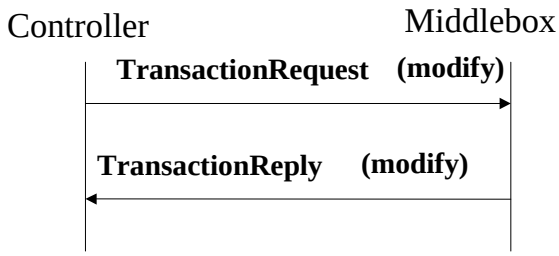
The controller must provide the Middlebox with enough information to a create sending and a receiving transport flow.

```
MEGACO/1 [Controller IP Address]:55555
Transaction = 1{
  Context = ${
            Add = ${
            TerminationState{
            EMP/iface=0 -- the A-side
            }
            Media{Stream = 1{
            Remote{
                   v=0
                   m= audio 1124 RTP/AVP 0
                   c=IN IP4 1.1.1.1
            }
          } /*of Stream*/
       } /*of Media*/
    } /*of ADD*/
            Add = ${
            TerminationState{
            EMP/iface=1 -- the B-side
            }
    Media{Stream = 1{
            Local{
                 v=0
                 m=audio 1124 RTP/AVP 0
                 c=IN IP4 1.1.1.1
            }
          } /*of Stream*/
       } /*of Media*/
    } /*of ADD*/
  }/* of context*/
} /*of transaction*/
```

2) Middlebox→Controller:

The Middlebox provides the information on the possible transport flows it can support. If the Middlebox can support multiple transport streams that satisfy the request, multiple local and remote Descriptors are to be returned.

```
MEGACO/1 [Middlebox IP Address]:55555
Reply = 1{
  Context = 3001{
            Add = 1234,
            Add = 1235
  }/* of context*/
} /*of transaction*/
```



3) Controller→Middlebox:

If there was a choice to be made, the controller has decided which `Local/Remote Descriptor` shall be used (probably after signalling with its peers) and collapses the choices in the terminations created. The terminations are modified by the controller to contain only the `Local/Remote \Descriptor`s required. If not provided earlier, additional information such as remote transport addresses are filled in.

```
MEGACO/1 [Controller IP Address]:55555


Transaction = 2{
  Context = 3001{
            Modify = 1234{
              Media{Stream=1{
            Local{
                  v=0
                  m=audio 3300 RTP/AVP 0
                  c=IN IP4 4.4.4.4
            }
          } /*of Stream*/
       } /*of Media*/
            Modify = 1235{
              Media{Stream=1{
            Remote{
                  v=0
                  m=audio 3300 RTP/AVP 0
                  c=IN IP4 4.4.4.4
            }
```

```
        } /*of Stream*/
      } /*of Media*/
    } /*of Modify*/
  }/* of context*/
} /*of transaction*/
```

4) Middlebox→Controller:

The Middlebox collapses the terminations to a usable state and transport starts to flow.

```
MEGACO/1 [Middlebox IP Address]:55555
Reply = 2{
  Context = 3001{
            Modify = 1234{}
            Modify = 1235{}

  }/* of context*/
} /*of transaction*/
```

5) Controller→Middlebox:

When the controller decides that the transport flows are to be terminated. Terminations are `subtract`ed in the normal fashion.

```
TransactionRequest.
  Subtract { originating and terminating sides }
```

6) MC→MC:

```
TransactionReply.Subtract
```

C.2 Basic scenario with address translation (the legacy mode)

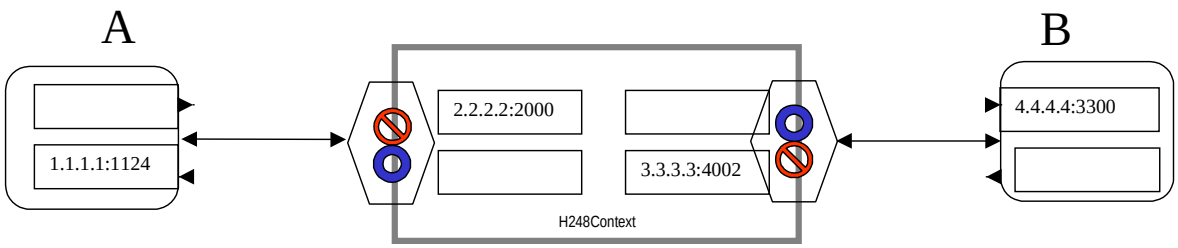


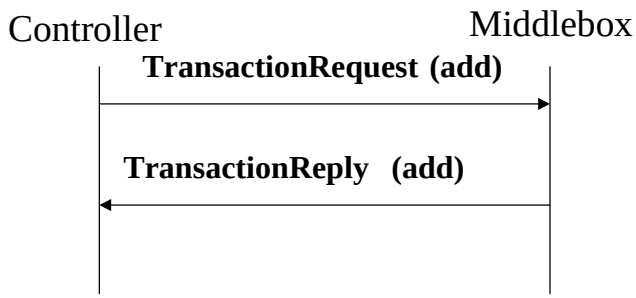
Figure C.2: Example H.248 context

Each termination sinks/sources one bi-directional stream with each a LocalDescriptor, RemoteDescriptor and a LocalControlDescriptor. The StreamIds are used to link the streams that are connected.

The following table provides the addresses and ports used in this figure:

Stream	StreamId	Address in LocalDescr Entering the media gateway	Addresses in RemoteDescr Leaving the media gateway
A to B	1	**Source =*:*** **Sink = 2.2.2.2:2000**	**Source = *:*** **Sink = 4.4.4.4:3300**
B to A	1	**Source = *:*** **Sink = 3.3.3.3:4002**	**Source = *:*** **Sink = 1.1.1.1:1124**

The meta protocol exchange above maps to the following H.248 message flow for this scenario.



1) MiddleboxController→Middlebox:

The MiddleboxController must provide the Middlebox with enough information to a create sending and a receiving media flow.

```
MEGACO/1 [MiddleboxController IP Address]:55555
Transaction = 1{
  Context = ${
            Add = ${
            Media{Stream = 1{
            Local{
              v=0
              m=audio $ RTP/AVP 0
              c=IN IP4 $ -- for the <A-Side dest IP address of Middlebox>

            }
            Remote{
                   v=0
                   m= audio 1124 RTP/AVP 0
                   c=IN IP4 1.1.1.1
            }
          } /*of Stream*/
        } /*of Media*/
     } /*of ADD*/
            Add = ${
     Media{Stream = 1{
            Local{
```

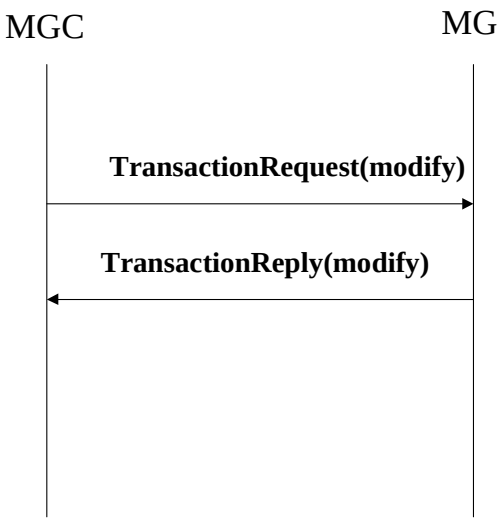
```
                v=0
                m=audio $ RTP/AVP 0
                c=IN IP4 $ -- <B-side dest IP address of Middlebox>
            }
            Remote{
                   v=0
                   m= audio $ RTP/AVP 0
                   c=IN IP4 $ -- <B-side source IP address of Middlebox>
            }
          } /*of Stream*/
        } /*of Media*/
      } /*of ADD*/
    }/* of context*/
} /*of transaction*/
```

2) Middlebox→MiddleboxController:

The Middlebox provides the information on the possible media flows it can support. If the Middlebox can support multiple media streams that satisfy the request, multiple local and remote Descriptors are to be returned.

```
MEGACO/1 [Middlebox IP Address]:55555
Reply = 1{
  Context = 3001{
            Add = 1234{
          Media{Stream = 1{
            Local{
              v=0
              m=audio 2000 RTP/AVP 0
              c=IN IP4 2.2.2.2
            }
          } /*of Stream*/
        } /*of Media*/
      } /*of ADD*/
            Add = 1235{
        Media{Stream = 1{
            Local{
                v=0
                m=audio 4002 RTP/AVP 0
                c=IN IP4 3.3.3.3
            }
          } /*of Stream*/
        } /*of Media*/
      } /*of ADD*/
```

```
   }/* of context*/
} /*of transaction*/
```



3) MiddleboxController→Middlebox:

If there was a choice to be made, the MiddleboxController has decided which `Local/Remote Descriptor` shall be used and collapses the choices in the terminations created. The terminations are modified by the MiddleboxController to contain only the `Local/Remote Descriptor`s required. If not provided earlier, additional information such as remote transport addresses are filled in.

```
MEGACO/1 [MiddleboxController IP Address]:55555
Transaction = 2{
   Context = 3001{
             Modify = 1235{
               Media{Stream=1{
             Local{
                  v=0
                  m=audio 4002 RTP/AVP 0
                  c=IN IP4 3.3.3.3
             }
             Remote{
                    v=0
                   m=audio 3300 RTP/AVP 0
                    c=IN IP4 4.4.4.4
             }
           } /*of Stream*/
         } /*of Media*/
      } /*of Modify*/
   }/* of context*/
} /*of transaction*/
```

4) Middlebox→MiddleboxController:

The Middlebox collapses the terminations to a usable state and media starts to flow.

```
MEGACO/1 [Middlebox IP Address]:55555
Reply = 2{
   Context = 3001{
             Modify = 1235{}
   }/* of context*/
} /*of transaction*/
```

5) MiddleboxController→Middlebox:

When the MiddleboxController decides that the media flows are to be terminated. Terminations are `subtract`ed in the normal fashion.

```
TransactionRequest.
  Subtract { originating and terminating sides }
```

6) MC→BC:

```
TransactionReply.Subtract
```

C.3 Flow with address translation without source filtering

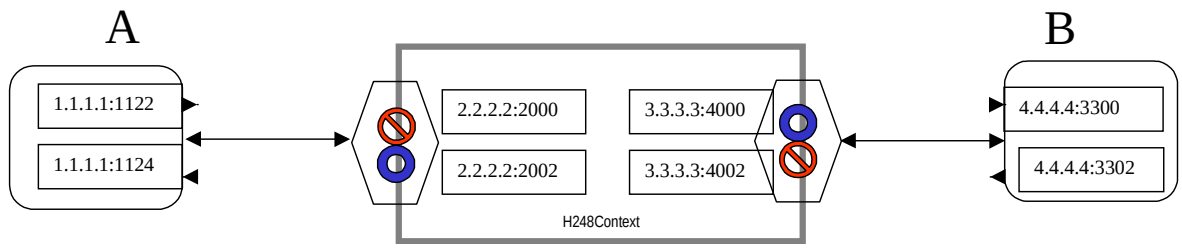


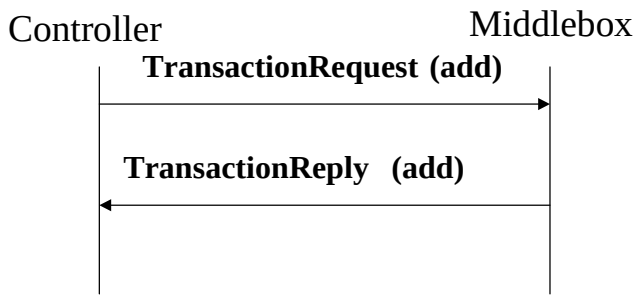
Figure C.3: Example H.248 context

Each termination sinks/sources one bi-directional stream with each a LocalDescriptor, RemoteDescriptor and a LocalControlDescriptor. The StreamIds are used to link the streams that are connected.

The following table provides the addresses and ports used in this figure:

Stream	StreamId	Address in LocalDescr Entering the middlebox	Addresses in RemoteDescr Leaving the middlebox
A to B	1	**Source =*:*** **Sink = 2.2.2.2:2000**	**Source =3.3.3.3:4000** **Sink = 4.4.4.4:3300**
B to A	1	**Source =*:*** **Sink =3.3.3.3:4002**	**Source = 2.2.2.2:2002** **Sink = 1.1.1.1:1124**

The exchange above maps to the following H.248-EMP message flow.



1) Controller→Middlebox:

The controller must provide the Middlebox with enough information to a create sending and a receiving transport flow.

```
MEGACO/1 [Controller IP Address]:55555
Transaction = 1{
  Context = ${
            Add = ${
            TerminationState{
            EMP/iface=0 -- the A-side
            }
            Media{Stream = 1{
            Local{
              v=0
              m=audio $ RTP/AVP 0
              c=IN IP4 $ -- for the <A-Side dest IP address of middlebox>
              a=recvonly
            }
            Remote{
                   v=0
                   m= audio 1124 RTP/AVP 0
                   c=IN IP4 1.1.1.1
                   a=recvonly
                   m=audio $ RTP/AVP 0
                   c=IN IP4 $
                   a=sendonly

            }
         } /*of Stream*/
       } /*of Media*/
    } /*of ADD*/
            Add = ${
            TerminationState{
            EMP/iface=1 -- the B-side
            }
    Media{Stream = 1{
```

```
            Local{
                 v=0
                 m=audio $ RTP/AVP 0
                 c=IN IP4 $ -- <B-side dest IP address of middlebox>
                 a=recvonly
            }
            Remote{
                   v=0
                   m= audio $ RTP/AVP 0
                   c=IN IP4 $ -- <B-side source IP address of middlebox>
                   a=sendonly
            }
          } /*of Stream*/
        } /*of Media*/
      } /*of ADD*/
    }/* of context*/
} /*of transaction*/
```

2) Middlebox→Controller:

The Middlebox provides the information on the possible transport flows it can support. If the Middlebox can support multiple transport streams that satisfy the request, multiple local and remote Descriptors are to be returned.

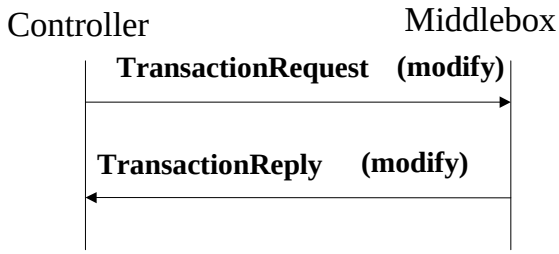
```
MEGACO/1 [Middlebox IP Address]:55555
Reply = 1{
  Context = 3001{
             Add = 1234{
          Media{Stream = 1{
            Local{
              v=0
              m=audio 2000 RTP/AVP 0
              c=IN IP4 2.2.2.2
              a=recvonly
              m=audio 1122 RTP/AVP 0
              c=IN IP4 1.1.1.1
              a=sendonly
            }
            Remote{
                   v=0
                   m= audio 1123 RTP/AVP 0
                   c=IN IP4 1.1.1.1
                   a=recvonly
                   m=audio 2002 RTP/AVP 0
```

```
                    c=IN IP4 2.2.2.2
                    a=sendonly

               }

          } /*of Stream*/
        } /*of Media*/
      } /*of ADD*/
            Add = 1235{
        Media{Stream = 1{
            Local{

                 v=0
                 m=audio 4002 RTP/AVP 0
                 c=IN IP4 3.3.3.3
                 a=recvonly
            }
            Remote{
                   v=0
                   m= audio 4000 RTP/AVP 0
                   c=IN IP4 3.3.3.3
              a=sendonly
            }
          } /*of Stream*/
        } /*of Media*/
      } /*of ADD*/
    }/* of context*/
} /*of transaction*/
```



3) Controller→Middlebox:

If there was a choice to be made, the controller has decided which `Local/Remote Descriptor` shall be used (probably after signalling with its peers) and collapses the choices in the terminations created. The terminations are modified by the controller to contain only the `Local/Remote Descriptor`s required. If not provided earlier, additional information such as remote transport addresses are filled in.

```
MEGACO/1 [Controller IP Address]:55555

Transaction = 2{
```

```
  Context = 3001{
            Modify = 1235{
              Media{Stream=1{
            Local{

                 v=0
                 m=audio 4002 RTP/AVP 0
                 c=IN IP4 3.3.3.3
                 a=recvonly
            }
            Remote{
                   v=0
                   m=audio 3300 RTP/AVP 0
                   c=IN IP4 4.4.4.4
                   a=recvonly
                   m= audio 4000 RTP/AVP 0
                   c=IN IP4 3.3.3.3
                   a=sendonly
            }

          } /*of Stream*/
        } /*of Media*/
      } /*of Modify*/
  }/* of context*/
} /*of transaction*/
```

4) Middlebox→Controller:

The Middlebox collapses the terminations to a usable state and transport starts to flow.

```
MEGACO/1 [Middlebox IP Address]:55555
Reply = 2{
  Context = 3001{
            Modify = 1235{}
  }/* of context*/
} /*of transaction*/
```

5) Controller→Middlebox:

When the controller decides that the transport flows are to be terminated. Terminations are `subtract`ed in the normal fashion.

```
TransactionRequest.
  Subtract { originating and terminating sides }
```

6) MC→MC:

```
TransactionReply.Subtract
```

C.4 Flow without address translation with source filtering

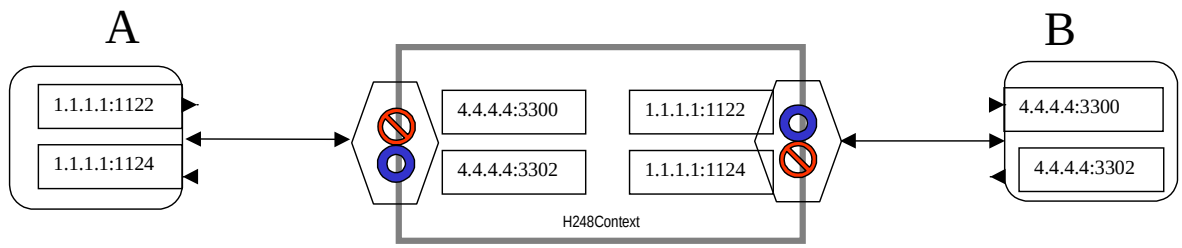


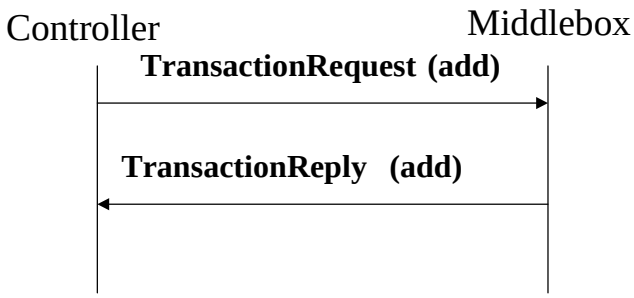
Figure C.5 is misleading since there are no 2.2.2.2 or 3.3.3.3. addresses in this example.

Each termination sinks/sources one bi-directional stream with each a LocalDescriptor, RemoteDescriptor and a LocalControlDescriptor. The StreamIds are used to link the streams that are connected.

The following table provides the addresses and ports used in this figure:

Stream	StreamId	Address in LocalDescr Entering the middlebox	Addresses in RemoteDescr Leaving the middlebox
A to B	1	**Source =1.1.1.1:1122** **Sink = 4.4.4.4:3300**	**Source = 1.1.1.1:1122** **Sink = 4.4.4.4:3300**
B to A	1	**Source = 4.4.4.4:3302** **Sink = 1.1.1.1:1124**	**Source = 4.4.4.4:3302** **Sink = 1.1.1.1:1124**

The exchange above maps to the following H.248-EMP message flow.



1) Controller→Middlebox:

The controller must provide the Middlebox with enough information to a create sending and a receiving transport flow.

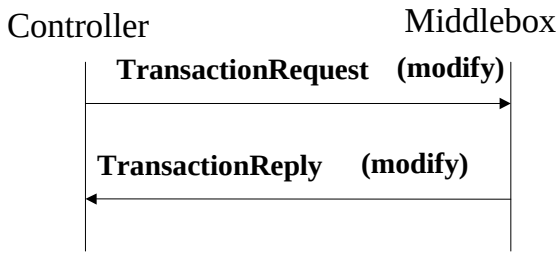
```
MEGACO/1 [Controller IP Address]:55555
Transaction = 1{
  Context = ${
            Add = ${
            TerminationState{
            EMP/iface=0 -- the A-side
            }
            Media{Stream = 1{
```

```
            Local{
              v=0
              m=audio 1122 RTP/AVP 0
              c=IN IP4 1.1.1.1
              a=sendonly
            }
            Remote{
                   v=0
                   m= audio 1124 RTP/AVP 0
                   c=IN IP4 1.1.1.1
                   a=recvonly
            }
          } /*of Stream*/
        } /*of Media*/
      } /*of ADD*/
            Add = ${
            TerminationState{
            EMP/iface=1 -- the B-side
            }
      Media{Stream = 1{
            Local{
                 v=0
                 m=audio 1124 RTP/AVP 0
                 c=IN IP4 1.1.1.1
                 a=recvonly
            }
            Remote{
                   v=0
                   m= audio 1122 RTP/AVP 0
                   c=IN IP4 1.1.1.1
                   a=sendonly
            }
          } /*of Stream*/
        } /*of Media*/
      } /*of ADD*/
    }/* of context*/
} /*of transaction*/
```

2) Middlebox→Controller:

The Middlebox provides the information on the possible transport flows it can support. If the Middlebox can support multiple transport streams that satisfy the request, multiple local and remote Descriptors are to be returned.

```
MEGACO/1 [Middlebox IP Address]:55555
Reply = 1{
  Context = 3001{
            Add = 1234,
            Add = 1235
  }/* of context*/
} /*of transaction*/
```



3) Controller→Middlebox:

If there was a choice to be made, the controller has decided which `Local/Remote Descriptor` shall be used (probably after signalling with its peers) and collapses the choices in the terminations created. The terminations are modified by the controller to contain only the `Local/Remote \Descriptor`s required. If not provided earlier, additional information such as remote transport addresses are filled in.

```
MEGACO/1 [Controller IP Address]:55555


Transaction = 2{
  Context = 3001{
            Modify = 1234{
              Media{Stream=1{
            Local{
                  v=0
                  m=audio 3300 RTP/AVP 0
                  c=IN IP4 4.4.4.4
                  a=recvonly
                  m= audio 1122 RTP/AVP 0
                  c=IN IP4 1.1.1.1
                  a=sendonly
            }
            Remote{
                 v=0
                 m=audio 1124 RTP/AVP 0
                 c=IN IP4 1.1.1.1
                 a=recvonly
```

```
                m=audio 3302 RTP/AVP 0
                c=IN IP4 4.4.4.4
                a=sendonly
            }
            Modify = 1235{
              Media{Stream=1{
            Local{
                v=0
                m=audio 1124 RTP/AVP 0
                c=IN IP4 1.1.1.1
                a=recvonly
                m=audio 3302 RTP/AVP 0
                c=IN IP4 4.4.4.4
                a=sendonly
            }
            Remote{
                   v=0
                   m=audio 3300 RTP/AVP 0
                   c=IN IP4 4.4.4.4
                   a=recvonly
                   m= audio 1122 RTP/AVP 0
                   c=IN IP4 1.1.1.1
                   a=sendonly
            }

         } /*of Stream*/
       } /*of Media*/
     } /*of Modify*/
  }/* of context*/
} /*of transaction*/
```

4) Middlebox→Controller:

The Middlebox collapses the terminations to a usable state and transport starts to flow.

```
MEGACO/1 [Middlebox IP Address]:55555
Reply = 2{
  Context = 3001{
            Modify = 1234{}
            Modify = 1235{}
  }/* of context*/
} /*of transaction*/
```

5) Controller→Middlebox:

When the controller decides that the transport flows are to be terminated. Terminations are `subtract`ed in the normal fashion.

```
TransactionRequest.
  Subtract { originating and terminating sides }
```

6) MC→MC:

```
TransactionReply.Subtract
```

C.5 Flow with address translation with source filtering

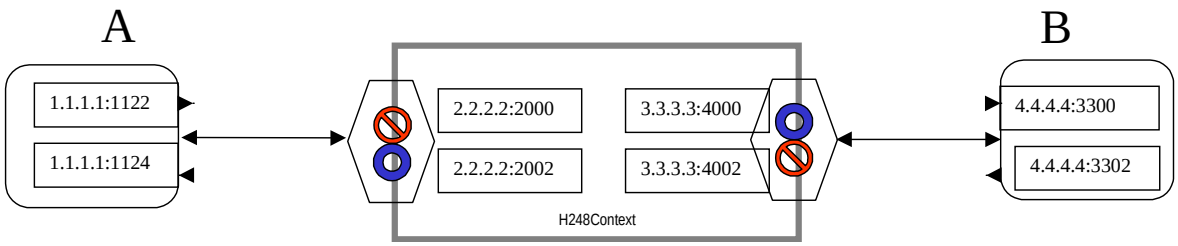


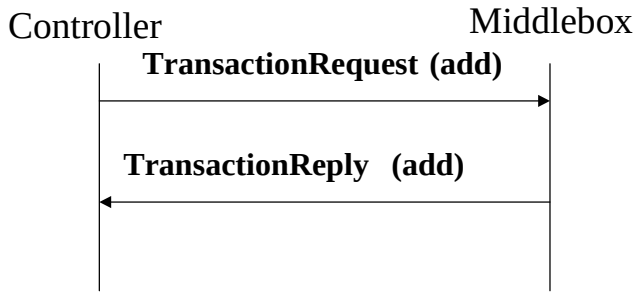
Figure C.5: Example H.248 context

Each termination sinks/sources one bi-directional stream with each a LocalDescriptor, RemoteDescriptor and a LocalControlDescriptor. The StreamIds are used to link the streams that are connected.

The following table provides the addresses and ports used in this figure:

Stream	StreamId	Address in LocalDescr Entering the middlebox	Addresses in RemoteDescr Leaving the middlebox
A to B	1	`Source = 1.1.1.1:1122` `Sink = 2.2.2.2:2000`	`Source = 3.3.3.3:4000` `Sink = 4.4.4.4:3300`
B to A	1	`Source = 4.4.4.4:3302` `Sink = 3.3.3.3:4002`	`Source = 2.2.2.2:2002` `Sink = 1.1.1.1:1124`

The exchange above maps to the following H.248-EMP message flow.



1) Controller→Middlebox:

The controller must provide the Middlebox with enough information to a create sending and a receiving transport flow.

```
MEGACO/1 [Controller IP Address]:55555
Transaction = 1{
  Context = ${
            Add = ${
            TerminationState{
            EMP/iface=0 -– the A-side
            }
            Media{Stream = 1{
            Local{
              v=0
              m=audio $ RTP/AVP 0
              c=IN IP4 $ -- for the <A-Side dest IP address of middlebox>
              a=recvonly
              m=audio 1122 RTP/AVP 0
              c=IN IP4 1.1.1.1
              a=sendonly
            }
            Remote{
                   v=0
                   m= audio 1124 RTP/AVP 0
                   c=IN IP4 1.1.1.1
                   a=recvonly
                   m=audio $ RTP/AVP 0
                   c=IN IP4 $
                   a=sendonly

            }
          } /*of Stream*/
        } /*of Media*/
    } /*of ADD*/
            Add = ${
            TerminationState{
            EMP/iface=1 -– the B-side
            }
    Media{Stream = 1{
            Local{
                 v=0
                 m=audio $ RTP/AVP 0
                 c=IN IP4 $ -- <B-side dest IP address of middlebox>
```

```
                a=recvonly
            }
            Remote{
                   v=0
                   m= audio $ RTP/AVP 0
                   c=IN IP4 $ -- <B-side source IP address of middlebox>
                   a=sendonly
            }
          } /*of Stream*/
        } /*of Media*/
      } /*of ADD*/
    }/* of context*/
} /*of transaction*/
```

2) Middlebox→Controller:

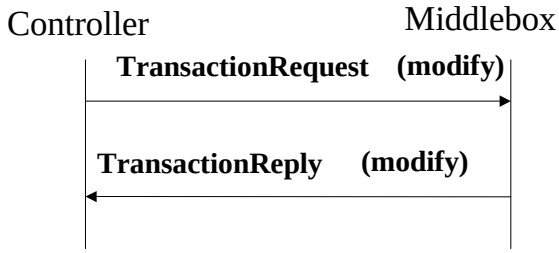
The Middlebox provides the information on the possible transport flows it can support. If the Middlebox can support multiple transport streams that satisfy the request, multiple local and remote Descriptors are to be returned.

```
MEGACO/1 [Middlebox IP Address]:55555
Reply = 1{
  Context = 3001{
             Add = 1234{
          Media{Stream = 1{
            Local{
              v=0
              m=audio 2000 RTP/AVP 0
              c=IN IP4 2.2.2.2
              a=recvonly
              m=audio 1122 RTP/AVP 0
              c=IN IP4 1.1.1.1
              a=sendonly
            }
            Remote{
                   v=0
                   m= audio 1123 RTP/AVP 0
                   c=IN IP4 1.1.1.1
                   a=recvonly
                   m=audio 2002 RTP/AVP 0
                   c=IN IP4 2.2.2.2
                   a=sendonly

            }
```

```
        } /*of Stream*/
      } /*of Media*/
    } /*of ADD*/
          Add = 1235{
      Media{Stream = 1{
          Local{

              v=0
              m=audio 4002 RTP/AVP 0
              c=IN IP4 3.3.3.3
              a=recvonly
          }
          Remote{
                v=0
                m= audio 4000 RTP/AVP 0
                c=IN IP4 3.3.3.3
            a=sendonly
          }
        } /*of Stream*/
      } /*of Media*/
    } /*of ADD*/
  }/* of context*/
} /*of transaction*/
```



3) Controller→Middlebox:

If there was a choice to be made, the controller has decided which `Local/Remote Descriptor` shall be used (probably after signalling with its peers) and collapses the choices in the terminations created. The terminations are modified by the controller to contain only the `Local/Remote \Descriptor`s required. If not provided earlier, additional information such as remote transport addresses are filled in.

```
MEGACO/1 [Controller IP Address]:55555

Transaction = 2{
  Context = 3001{
          Modify = 1235{
            Media{Stream=1{
          Local{
```

```
                v=0
                m=audio 4002 RTP/AVP 0
                c=IN IP4 3.3.3.3
                a=recvonly
                m=audio 3302 RTP/AVP 0
                c=IN IP4 4.4.4.4
                a=sendonly
            }
            Remote{
                   v=0
                   m=audio 3300 RTP/AVP 0
                   c=IN IP4 4.4.4.4
                   a=recvonly
                   m= audio 4000 RTP/AVP 0
                   c=IN IP4 3.3.3.3
                   a=sendonly
            }

         } /*of Stream*/
       } /*of Media*/
     } /*of Modify*/
   }/* of context*/
} /*of transaction*/
```

4) Middlebox→Controller:

The Middlebox collapses the terminations to a usable state and transport starts to flow.

```
MEGACO/1 [Middlebox IP Address]:55555
Reply = 2{
   Context = 3001{
             Modify = 1235{}
   }/* of context*/
} /*of transaction*/
```

5) Controller→Middlebox:

When the controller decides that the transport flows are to be terminated. Terminations are `subtracte`d in the normal fashion.

```
TransactionRequest.
  Subtract { originating and terminating sides }
```

6) MC→MC:

```
TransactionReply.Subtract
```

C.6 Explicit RTCP addresses with source filtering and address translation

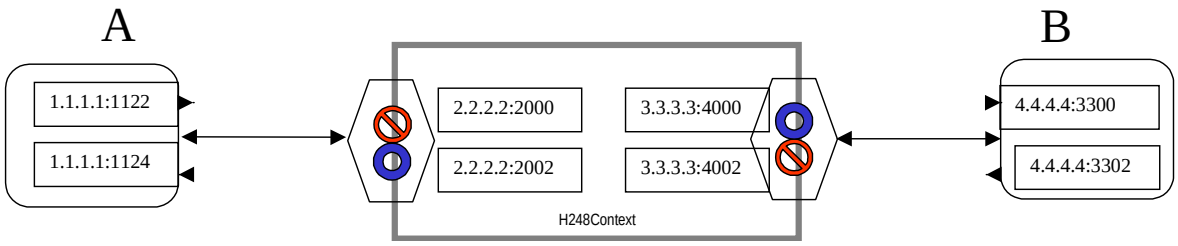


Figure C.6: Example H.248 context

Each termination sinks/sources one bi-directional stream with each a LocalDescriptor, RemoteDescriptor and a LocalControlDescriptor. The StreamIds are used to link the streams that are connected.

In previous scenarios the H.248 messages specified addresses and ports of the RTP streams. The Middlebox is supposed to implicitly open a pinhole for the associated RTCP streams. If port X is opened to receive RTP then implicitly port X + 1 is opened for RTCP, where X is an even number.

In this example we have explicit RTCP ports. The RTCP ports are the port numbers shown above prefixed with a "2". So the source RTCP port for A is 21222 and the sink port is 22000.

The Middlebox looks at the transport protocol sub-field of the m-line of SDP to decide whether an implicit RTCP pinhole should be opened. If the transport protocol is RTP/AVP then an implicit RTCP pinhole must be opened. If it is UDP then only a pinhole for the UDP stream must be opened.

The following table provides the addresses and ports used in this figure:

Stream	StreamId	Address in LocalDescr Entering the middlebox	Addresses in RemoteDescr Leaving the middlebox.
A to B (RTP)	1	**Source = 1.1.1.1:1122** **Sink = 2.2.2.2:2000**	**Source = 3.3.3.3:4000** **Sink = 4.4.4.4:3300**
B to A (RTP)	1	**Source =4.4.4.4:3302** **Sink = 3.3.3.3:4002**	**Source = 2.2.2.2:2002** **Sink = 1.1.1.1:1124**
A to B (RTCP)	2	**Source = 1.1.1.1:21122** **Sink = 2.2.2.2:22000**	**Source = 3.3.3.3:24000** **Sink = 4.4.4.4:23300**
B to A (RTCP)	2	**Source = 4.4.4.4:23302** **Sink = 3.3.3.3:24002**	**Source = 2.2.2.2:22002** **Sink = 1.1.1.1:21124**

1) Controller→Middlebox:

```
MEGACO/1 [Controller IP Address]:55555
Transaction = 1{
  Context = ${
            Add = ${
            TerminationState{
            EMP/iface=0 -- the A-side
```

```
  }
  Media{Stream = 1{ -- RTP stream
  Local{
    v=0
    m=audio $ UDP
    c=IN IP4 $ -- for the <A-Side dest IP address of middlebox>
    a=recvonly
    m=audio 1122 UDP
    c=IN IP4 1.1.1.1
    a=sendonly
  }
  Remote{
         v=0
         m= audio 1124 UDP
         c=IN IP4 1.1.1.1
         a=recvonly
         m=audio $ UDP
         c=IN IP4 $
         a=sendonly

  }
} /*of Stream*/
  Stream = 2{ -- RTCP stream
  Local{
    v=0
    m=control $ UDP RTCP
    c=IN IP4 $ -- for the <A-Side dest IP address of middlebox>
    a=recvonly
    m=control 21122 UDP RTCP
    c=IN IP4 1.1.1.1
    a=sendonly
  }
  Remote{
         v=0
         m= control 21124 UDP RTCP
         c=IN IP4 1.1.1.1
         a=recvonly
         m=control $ UDP RTCP
         c=IN IP4 $
         a=sendonly
  }
```

```
        } /*of Stream*/
      } /*of Media*/
    } /*of ADD*/
          Add = ${
          TerminationState{
          EMP/iface=1 -– the B-side
          }
    Media{Stream = 1{ -- RTP stream
          Local{
               v=0
               m=audio $ UDP
               c=IN IP4 $ -- <B-side dest IP address of middlebox>
               a=recvonly
          }
          Remote{
                 v=0
                 m= audio $ UDP
                 c=IN IP4 $ -- <B-side source IP address of middlebox>
                 a=sendonly
          }
        } /*of Stream*/
          Stream = 2{ -- RTCP stream
          Local{
               v=0
               m=control $ UDP RTCP
               c=IN IP4 $ -- <B-side dest IP address of middlebox>
               a=recvonly
          }
          Remote{
                 v=0
                 m= control $ UDP RTCP
                 c=IN IP4 $ -- <B-side source IP address of middlebox>
                 a=sendonly
          }
        } /*of Stream*/
      } /*of Media*/
    } /*of ADD*/
  }/* of context*/
} /*of transaction*/
```

1) Middlebox→Controller:

```
MEGACO/1 [Middlebox IP Address]:55555
Reply = 1{
  Context = 3001{
            Add = 1234{
          Media{Stream = 1{ -- RTP stream
            Local{
              v=0
              m=audio 2000 UDP
              c=IN IP4 2.2.2.2
              a=recvonly
              m=audio 1122 UDP
              c=IN IP4 1.1.1.1
              a=sendonly
            }
            Remote{
                   v=0
                   m= audio 1123 UDP
                   c=IN IP4 1.1.1.1
                   a=recvonly
                   m=audio 2002 UDP
                   c=IN IP4 2.2.2.2
                   a=sendonly

            }

          } /*of Stream*/
            Stream = 2{ -- RTCP stream
            Local{
              v=0
              m=control 22000 UDP RTCP
              c=IN IP4 2.2.2.2
              a=recvonly
              m=control 21122 UDP RTCP
              c=IN IP4 1.1.1.1
              a=sendonly
            }
            Remote{
                   v=0
                   m= control 21123 UDP RTCP
                   c=IN IP4 1.1.1.1
```

```
                a=recvonly
                m=control 22002 UDP RTCP
                c=IN IP4 2.2.2.2
                a=sendonly

          }
        } /*of Stream*/
      } /*of Media*/
    } /*of ADD*/
          Add = 1235{
          Media{Stream = 1{ -- RTP stream
          Local{
               v=0
               m=audio 4002 UDP
               c=IN IP4 3.3.3.3
               a=recvonly
          }
          Remote{
                 v=0
                 m= audio 4000 UDP
                 c=IN IP4 3.3.3.3
            a=sendonly
          }
        } /*of Stream*/
          Stream = 2{ -- RTCP stream
          Local{
               v=0
               m=control 24002 UDP RTCP
               c=IN IP4 3.3.3.3
               a=recvonly
          }
          Remote{
                 v=0
                 m= control 24000 UDP RTCP
                 c=IN IP4 3.3.3.3
            a=sendonly
          }
        } /*of Stream*/
      } /*of Media*/
    } /*of ADD*/
}/* of context*/
```

```
} /*of transaction*/
```

2) Middlebox→Controller:

```
MEGACO/1 [Middlebox IP Address]:55555
Reply = 1{
  Context = 3001{
            Add = 1234{
          Media{Stream = 1{ -- RTP stream
            Local{
              v=0
              m=audio 2000 UDP
              c=IN IP4 2.2.2.2
              a=recvonly
              m=audio 1122 UDP
              c=IN IP4 1.1.1.1
              a=sendonly
            }
            Remote{
                   v=0
                   m= audio 1123 UDP
                   c=IN IP4 1.1.1.1
                   a=recvonly
                   m=audio 2002 UDP
                   c=IN IP4 2.2.2.2
                   a=sendonly

            }

          } /*of Stream*/
            Stream = 2{ -- RTCP stream
            Local{
              v=0
              m=control 22000 UDP RTCP
              c=IN IP4 2.2.2.2
              a=recvonly
              m=control 21122 UDP RTCP
              c=IN IP4 1.1.1.1
              a=sendonly
            }
            Remote{
                   v=0
```

```
            m= control 21123 UDP RTCP
            c=IN IP4 1.1.1.1
            a=recvonly
            m=control 22002 UDP RTCP
            c=IN IP4 2.2.2.2
            a=sendonly

      }
    } /*of Stream*/
  } /*of Media*/
} /*of ADD*/
      Add = 1235{
      Media{Stream = 1{ -- RTP stream
      Local{
           v=0
           m=audio 4002 UDP
           c=IN IP4 3.3.3.3
           a=recvonly
      }
      Remote{
            v=0
            m= audio 4000 UDP
            c=IN IP4 3.3.3.3
        a=sendonly
      }
    } /*of Stream*/
      Stream = 2{ -- RTCP stream
      Local{
           v=0
           m=control 24002 UDP RTCP
           c=IN IP4 3.3.3.3
           a=recvonly
      }
      Remote{
            v=0
            m= control 24000 UDP RTCP
            c=IN IP4 3.3.3.3
        a=sendonly
      }
    } /*of Stream*/
  } /*of Media*/
```

```
    } /*of ADD*/
  }/* of context*/
} /*of transaction*/
```

As can be seen this signalling flow is quite similar to steps 1 and 2 in the flow in clause C.5. The change is the modification of RTP/AVP to UDP to prevent an implicit RTCP pinhole to be opened and the addition of stream 2 for the explicit RTCP flow. The remaining steps similar changes need to be made.

C.7 Signalling: example flow for a H323 RAS control pin-hole

The profile can not only be used for real-time stream setup but also for more management and security related issues like the enabling of signalling flows. This flow shows the sequence for setting up:

a) a general unidirectional RAS pin-hole that can be used for "GK Discovery", and

b) a specific RAS pin-hole that is set up due to receipt of a GK Discovery message (GRQ).

In the following figure, entity A is an H323 terminal and entity B is a GK that is sitting/hiding behind the middlebox. Initially, the GK will open up the general RAS pin-hole for GK Discovery messages. Assume that the terminal and the GK are sending/receiving on the same UDP port.

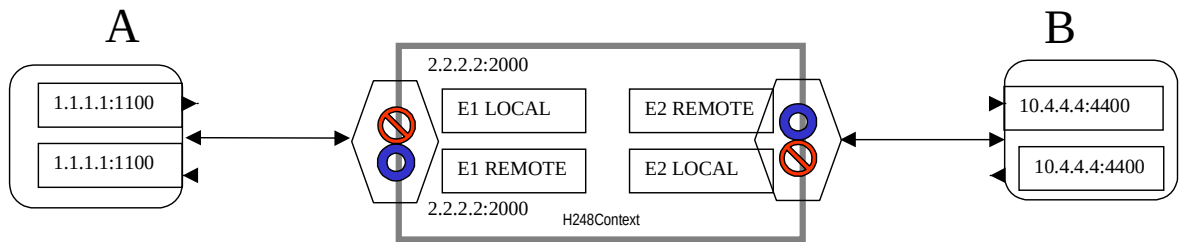


Figure C.7: General RAS pin-hole

The ephemeral terminations E1 and E2 have the following descriptors:

E1 LOCAL

```
v=0
m=control 2000 UDP RAS
c=IN IP4 2.2.2.2 /--"well known" GK Discovery address is 2.2.2.2:2000
a=recvonly        /-- it's unidirectional - incoming to MB
```

E2 REMOTE

```
v=0
m=control 4400 UDP RAS
c=IN IP4 10.4.4.4    /-- GK "real" address is 10.4.4.4:4400
a=recvonly    /-- it's unidirectional
```

On receipt of a GRQ, the GK decides to permit the user to register and so will provide him with an individual RAS address. Therefore a second pin-hole is set up with the following attributes:

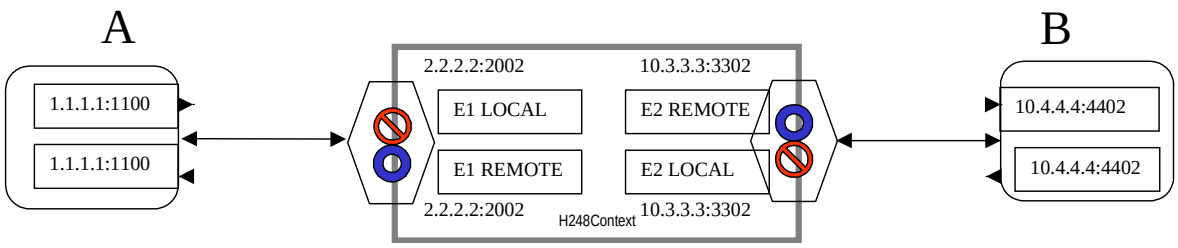


Figure C.8: Personal RAS pinhole

E1 LOCAL

```
v=0
m=control 2002 UDP RAS  /-- port chosen by MB and sent in GCF message
c=IN IP4 2.2.2.2 /-- external address of MB is 2.2.2.2
a=recvonly
m=control * UDP RAS /-- port unknown (will accept subsequent RAS messages
from any port)
c=IN IP4 1.1.1.1 /-- personal RAS messages come from A
a=sendonly
```

E1 REMOTE

```
v=0
m=control 1100 UDP RAS  /-- this is the port where A receives the GCF
                        /-- and subsequent messages signalling
c=IN IP4 1.1.1.1   /-- send messages to 1.1.1.1:1100
a=recvonly    /-- it's bi-directional
m=control 2000 UDP RAS
c=IN IP4 2.2.2.2   /-- GCF seems to be from 2.2.2.2:2000
a=sendonly
```

E2 LOCAL

```
v=0
m=control 1100 UDP RAS
c=IN IP4 1.1.1.1   /-- GK sends RAS messages to 1.1.1.1:1100
a=recvonly
m=control 4402 UDP RAS
c=IN IP4 10.4.4.4    /-- GK sends RAS messages from 10.4.4.4:4402
a=sendonly
```

E2 REMOTE

```
v=0
m=control 4402 UDP RAS
c=IN IP4 10.4.4.4  /-- send messages to 10.4.4.4:4402
a=recvonly
m=control 1100 UDP RAS
c=IN IP4 1.1.1.1
a=sendonly
```

Having sent the GCF, the second pin-hole is modified as follows :

E1 REMOTE

```
v=0
 m=control 1100 UDP RAS
c=IN IP4 1.1.1.1   /-- send messages to 1.1.1.1:1100
a=recvonly    /-- it's bi-directional
m=control 2002 UDP RAS
c=IN IP4 2.2.2.2   /-- Messages now come from 2.2.2.2:2002
a=sendonly
```

The 2nd RAS pin-hole may now be used for registration (RRQ). Note that there ought to be a TTL against the pin-hole on case no such RRQ is received.

Annex D (informative): Options on the basic scenarios

D.1 Transport activation option

TIPHON provides the option to mute the forward transport path until the call has been established.

This has the following change to the procedure as provided above. In step 1. Set **`LocalControl descriptor { Mode=SendOnly }`** to the protocol message.

Which then looks like:

```
Transaction = 1{
   Context = ${
             Add = ${
             TerminationState{
             EMP/iface=0 -– the A-side
             }
             Media{Stream = 1{
             LocalControl {
               Mode = SendOnly
             }
             Local{
v=0
               m=audio $ RTP/AVP 0
               c=IN IP4 $ -- <A-Side dest IP address of middlebox>
               a=recvonly
               m=audio <source port A-party> RTP/AVP 0
               c=IN IP4 <source IP address A-party>
               a=sendonly
             }
             Remote{
                    v=0
                    m= audio <dest port A-party> RTP/AVP 0
                    c=IN IP4 <destination IP address A=party>
                    a=recvonly
                    m=audio $ RTP/AVP 0
                    c=IN IP4 $ -- <A-side source address of middlebox>
                    a=sendonly

             }
          } /*of Stream*/
       } /*of Media*/
```

```
    } /*of ADD*/
           Add = ${
           TerminationState{
           EMP/iface=1 -– the B-side
           }
    Media{Stream = 1{
           LocalControl {
                Mode = ReceiveOnly
           }
           Local{
                v=0m=audio $ RTP/AVP 0
                c=IN IP4 $ -- <B-side dest IP address of middlebox>
                a=recvonly
           }
           Remote{
                  v=0
                  m= audio $ RTP/AVP 0
                  c=IN IP4 $ -- <B-side source IP address of middlebox>
                  a=sendonly
           }
         } /*of Stream*/
       } /*of Media*/
    } /*of ADD*/
  }/* of context*/
} /*of transaction*/
```

Between step 4 and 5 insert

The flow is set to duplex.

```
MEGACO/1 [Controller IP Address]:55555
Transaction = 3{
  Context = 3001{
           Modify= 1234{
           Media{Stream = 1{
           LocalControl{
             Mode=SendReceive,
           }
         } /*of Stream*/
       } /*of Transport*/
    } /*of Modify*/
           Modify= 1235{
```

```
            Media{Stream = 1{
            LocalControl{
               Mode=SendReceive,
            }
          } /*of Stream*/
        } /*of Transport*/
      } /*of Modify*/
   }/* of context*/
} /*of transaction*/
```

Generating the response:

```
MEGACO/1 [Middlebox IP Address]:55555
Reply = 3{
   Context = 3001{
            Modify = 1234,
            Modify = 1235
   }/* of context*/
} /*of transaction*/
```

As is shown in TS 101 882 [2], this shall happen after the call has been accepted by the remote party.

Annex E (informative): Bibliography

- ETSI TS 101 316: "Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON); Requirements for a protocol at reference point N; Media gateway controller to media gateway".

History

Document history		
V4.1.1	June 2002	Publication