Data-Over-Cable-Service-Interface Specifications
Modular-CMTS

Edge Resource Manager Interface Specification

CM-SP-ERMI-I02-051209

Issued Specification

Document Status Sheet

Document Control Number:	CM-SP-ERMI-I02-051209			
Document Title:	Edge Resource Manager Interface Specification			
Revision History:	D02 – 7/08/05 I01 – 8/05/05 I02 – 12/09/05			
Date:	December 9, 2005			
Status:	~~Work in Progress~~	~~Draft~~	Issued	~~Released~~
Distribution Restrictions:	~~Author Only~~	~~CL/Member~~	~~CL/ Member/ Vendor~~	Public

Key to Document Status Codes:

Work in Progress	An incomplete document, designed to guide discussion and generate feedback that may include several alternative requirements for consideration.
Draft	A document in specification format considered largely complete, but lacking review by Members and vendors. Drafts are susceptible to substantial change during the review process.
Issued	A stable document, which has undergone rigorous member and vendor review and is suitable for product design and development, cross-vendor interoperability, and for certification testing.
Released	A stable document, reviewed, tested and validated, suitable to enable cross-vendor interoperability.

Trademarks:

Contents

Figures

Tables

1 SCOPE

1.1 Introduction and Overview

This document specifies interfaces that are used by Edge QAM devices (EQAMs), Edge Resource Managers (ERMs) and M-CMTS cores within the context of a Modular Cable Modem Termination System (M-CMTS). This is one of several specifications that together define and specify a complete M-CMTS system (see Section 1.3). The basic architecture of a complete M-CMTS system is shown in Figure 1–1.

Three interfaces are specified in this document:

ERMI-1: A registration interface between an ERM and an EQAM. This interface is used to register and unregister EQAM resources (*i.e.,* QAM channels) with an ERM.

ERMI-2: A control interface between an EQAM and an ERM. This interface is used by an ERM to request QAM channel resource from an EQAM, and by an EQAM to deliver resources to an ERM.

ERMI-3: A control interface between an M-CMTS core and an ERM. This interface is used by the M-CMTS core to request specific QAM channel resources from the ERM, and by the ERM to respond to such requests with the location of QAM channel resources.

The interfaces specified in this document are shown in Figure 1–2 in Section 1.2.

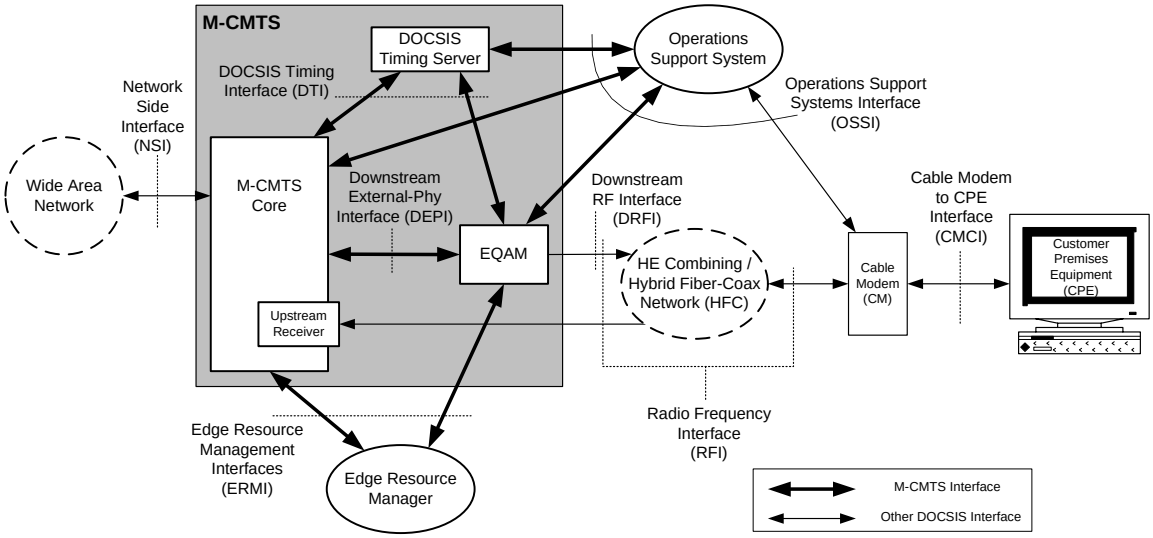

Figure 1–1 – M-CMTS Reference Architecture

1.2 Assumptions

In developing this specification, the following assumptions were made concerning the implementation and deployment of M-CMTS systems:

The system must function in the absence of an ERM.

In the absence of an ERM, all the interfaces to and from the ERM (*i.e.,* the interfaces specified in this document) are necessarily absent. Therefore, we assume that they will be replaced by a static configuration in which:

- The M-CMTS core is provisioned or otherwise configured with sufficient information to send data to EQAMs, and has sufficient knowledge of the resources available on each EQAM to access those resources without further information;

- The EQAMs are capable of co-operating with an M-CMTS core, even though they have not registered their resources with an ERM;
- The operator has configured the system in such a way that no unintended conflicts for resources arise.

The system will be compatible with Euro-DOCSIS [RFI2].

QAM channel identifiers (TSIDs) will be unique per head-end. In practice, this means that a TSID must be unique within the administrative domain that includes both the DOCSIS and VOD systems.

The system will be compatible with the Digital Set-top Gateway (DSG) specification [DSG].

A QAM channel resource will register with, and be controlled by a single ERM at any one time.

A QAM channel is used by one MAC Domain at any one time.

Only one CMTS may use the primary QAM channel (the QAM channel that carries the MAC control messages when DOCSIS channel-bonding is used).

Multiple ERMs may exist in a single system and the following are beyond the scope of this specification:

- Communication between ERMs;
- Resource aggregation using data from multiple ERMs;
- The method by which a session manager chooses the ERM with which to communicate.

A single ERM may manage both DOCSIS and VOD resources.

The M-CMTS system must function with DOCSIS 1.x and 2.0 cable modems (which do not provide feedback to the CMTS as to which downstream channels they can receive).

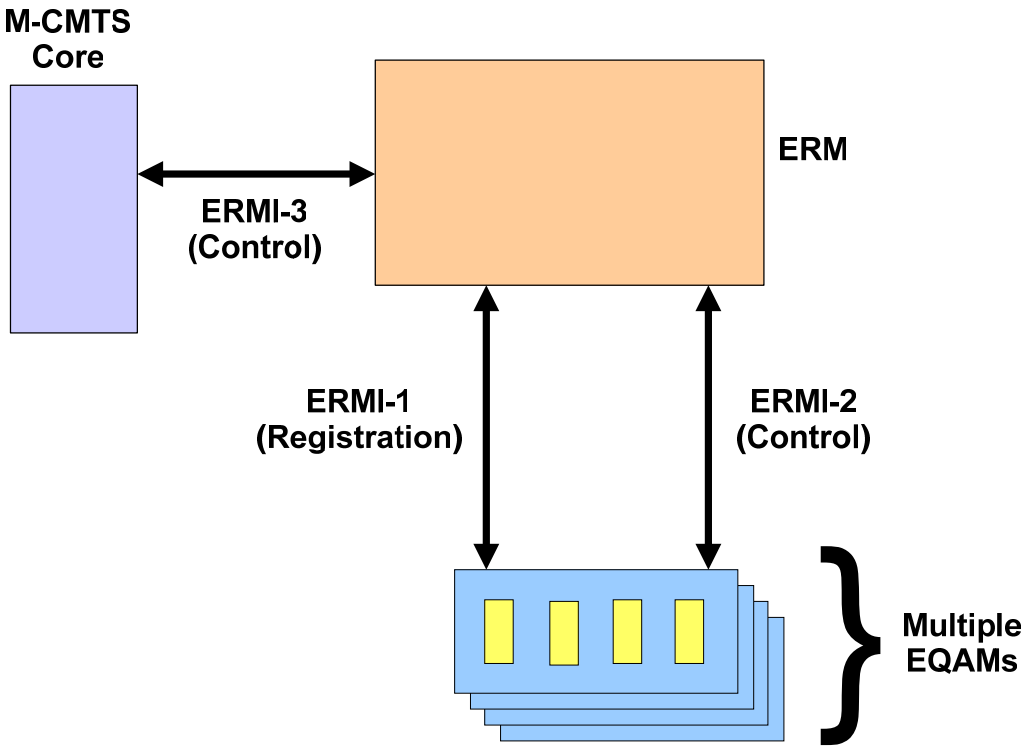

Figure 1–2 - ERMI Interfaces[1]

[1] Replaced this figure per ECN ERMI-N-05.0244-1 on 11/1/05.

1.3 Modular CMTS Interface Documents

A list of the documents in the Modular CMTS Interface Specifications family is provided in Table 1–1. For updates, please refer to http://www.cablemodem.com/specifications/.

Table 1–1 – M-CMTS Interface Specifications

Designation	Title
CM-SP-DEPI	Downstream External PHY Interface
CM-SP-DTI	DOCSIS Timing Interface
CM-SP-ERMI	Edge Resource Manager Interface
CM-SP-DRFI	Downstream Radio Frequency Interface

1.4 Requirements and Conventions

Throughout this document, the words that are used to define the significance of particular requirements are capitalized. These words are:

"MUST"	This word or the adjective "REQUIRED" means that the item is an absolute requirement of this specification.
"MUST NOT"	This phrase means that the item is an absolute prohibition of this specification.
"SHOULD"	This word or the adjective "RECOMMENDED" means that there may exist valid reasons in particular circumstances to ignore this item, but the full implications should be understood and the case carefully weighed before choosing a different course.
"SHOULD NOT"	This phrase means that there may exist valid reasons in particular circumstances when the listed behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.
"MAY"	This word or the adjective "OPTIONAL" means that this item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because it enhances the product, for example; another vendor may omit the same item.

2 REFERENCES

2.1 Normative References

[CMTS-NSI] Cable Modem Termination System – Network Side Interface Specification, SP-CMTS-NSI-I01-960702, July 2, 1996, Cable Television Laboratories, Inc.

[DEPI] Downstream External PHY Interface, CM-SP-DEPI-I02-051209, December 9, 2005, Cable Television Laboratories, Inc.

[DRFI] Downstream Radio Frequency Interface, CM-SP-DRFI-I02-051209, December 9, 2005, 2005, Cable Television Laboratories, Inc.

[DSG] DOCSIS Set-top Gateway (DSG) Interface Specification, CM-SP-DSG-I06-051209, December 9, 2005, Cable Television Laboratories, Inc.

[DTI] DOCSIS Timing Interface, CM-SP-DTI-I02-051209, December 9, 2005, Cable Television Laboratories, Inc.

[ISO 8601] ISO 8601:2004, Data elements and interchange formats – Information interchange – Representation of dates and times

[J.83] ITU-T Rec. J.83, (04/97) Digital multi-programme systems for television sound and data services for cable distribution.

[RFC 1123] STD 3, RFC 1123, Braden, R., "Requirements for Internet Hosts – Application and Support" October 1989, Internet Engineering Task Force.

[RFC 2068] RFC 2068, R. Fielding et al., “Hypertext Transfer Protocol – HTTP/1.1”, January 1997, Internet Engineering Task Force.

[RFC 2131] RFC 2131, R. Droms, “Dynamic Host Configuration Protocol”, March 1997, Internet Engineering Task Force.

[RFC 2326] RFC 2326, H. Schulzrinne; et al, Real Time Streaming Protocol (RTSP), April 1998, Internet Engineering Task Force.

[RFC 2373] RFC 2373, Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", July 1998, Internet Engineering Task Force.

[RFC 3219] RFC 3219, J. Rosenberg, H. Salama and M. Squire, "Telephony Routing over IP (TRIP)," January 2002, Internet Engineering Task Force.

[RFI2] DOCSIS 2.0 Radio Frequency Interface Specification, CM-SP-RFIv2.0-I10-051209, December 9, 2005, Cable Television Laboratories, Inc.

2.2 Reference Acquisition

- Cable Television Laboratories, Inc., 858 Coal Creek Circle, Louisville, CO 80027; Phone 303-661-9100; Fax 303-661-9199; Internet: http://www.cablelabs.com
- Internet Engineering Task Force (IETF), http://www.ietf.org
- International Telecommunication Union – Telecommunication Standardization Sector (ITU-T), http://www.itu.int/itu-t/
- International Organization for Standardization (ISO), http://www.iso.org

3 TERMS AND DEFINITIONS

This specification defines the following terms:

Edge QAM	A head-end or hub device that receives packets of digital video or data from the operator network. It re-packetizes the video or data into an MPEG transport stream and digitally modulates the transport stream onto a downstream RF carrier using QAM.
MAC domain	A grouping of layer 2 devices that can communicate with each other without using bridging or routing. In DOCSIS, a MAC domain is the group of CMs that are using upstream and downstream channels linked together through a MAC forwarding entity
Service Group	An HFC service group (also known as a service group) is a portion of an HFC access network used to deliver a set of services to a population of cable modems that share a common spectrum of RF channels.

4 ABBREVIATIONS AND ACRONYMS

This specification uses the following abbreviations and acronyms:

CM	Cable Modem
CMTS	Cable Modem Termination System
DEPI	DOCSIS External PHY Interface
DOCSIS	Data-Over-Cable-Service-Interface Specifications
DOCSIS-MPT	Traditional DOCSIS MPT Mode
DOCSIS-PSP	DOCSIS-Packet-Streaming-Protocol
DRRP	DOCSIS Resource Registration Protocol
DS	Downstream
DTI	DOCSIS Timing Interface
ERM	Edge Resource Manager
ERMI	Edge Resource Manager Interface(s)
FQDN	Fully Qualified Domain Name
FSM	Finite State Machine
HFC	Hybrid Fiber Coax
IP	Internet Protocol
M-CMTS	Modular Cable Modem Termination System
MAC	Media Access Control. Layer 2 of the ISO seven-layer model.
MPEG	Motion Picture Experts Group
MPEG-TS	Motion Picture Experts Group Transport Stream
PHY	Physical Layer. Used to refer to the downstream QAM transmitters and the upstream burst demodulators
PSP	Packet Streaming Protocol
QAM	Quadrature Amplitude Modulator (or Modulation)
RF	Radio Frequency
TRIP	Telephony Routing over IP
TSID	MPEG2 Transport Stream ID
UDP	User Datagram Protocol
URL	Uniform Resource Locator
VOD	Video On Demand

5 TECHNICAL OVERVIEW

5.1 Registration Interface

The registration interface (ERMI–1) allows EQAMs to register their QAM channel resources with an ERM.

5.1.1 Goals, Scope and Constraints

The Registration interface was designed to meet the following goals:

- Enable EQAMs to register their resources with an ERM
- Help the ERM to detect when failures occur in resources that it manages

The protocol used on this interface is based on TRIP [RFC 3219]. TRIP was designed to assist networks to locate voice-over-IP gateways and to route voice-over-IP traffic to an appropriate egress gateway. TRIP is a policy-driven protocol for advertising the reachability of telephony destinations between location servers, and for advertising attributes of the routes to those destinations.

Some EQAM resource registration protocols used in VOD systems are based on TRIP. Using a similar protocol to implement DOCSIS downstream resource allocation simplifies EQAM design.

When a VOD application needs a downstream QAM channel resource, it will request a downstream QAM channel resource from the ERM. When a DOCSIS application needs a downstream QAM channel resource, it will request a downstream QAM channel resource from the ERM.

5.1.1.1 Registering QAM Channels

Each ERM is responsible for managing the resources of one or more EQAMs. In order for an ERM to manage an EQAM's resources, it must first obtain an inventory of the resources associated with that EQAM. It must also obtain IP and/or RF addressing information associated with the EQAM and those resources in order to determine how to communicate with them. For example, an EQAM contains one or more QAM channels, each of which has associated RF properties (for example, the carrier frequency) and an RF address (the MPEG-2 Transport Stream ID (TSID)).

In order to allocate a particular QAM channel to a DOCSIS MAC domain, an ERM must know the following properties associated with that QAM channel:

- TSID
- QAM configuration, Capability, and QAM grouping
- Service Group
- Total available bandwidth

EQAMs use the Registration Interface to advertise available resources to an ERM. The ERM may use this information to populate a database of available resources.

5.1.2 Overall Architecture

The Registration Interface allows an EQAM to advertise to an ERM, information about the location and properties of resources under its control. The ERM may use this information to populate a database of the resources that have been reported to it by multiple EQAMs. The ERM may use this data to formulate responses to incoming requests for resources.

The Figure 5–1 shows the Registration component architecture. The Registration Interface between an ERM and an EQAM carries messages conforming to the DOCSIS Resource Registration Protocol (DRRP), as specified in Section 6.2.

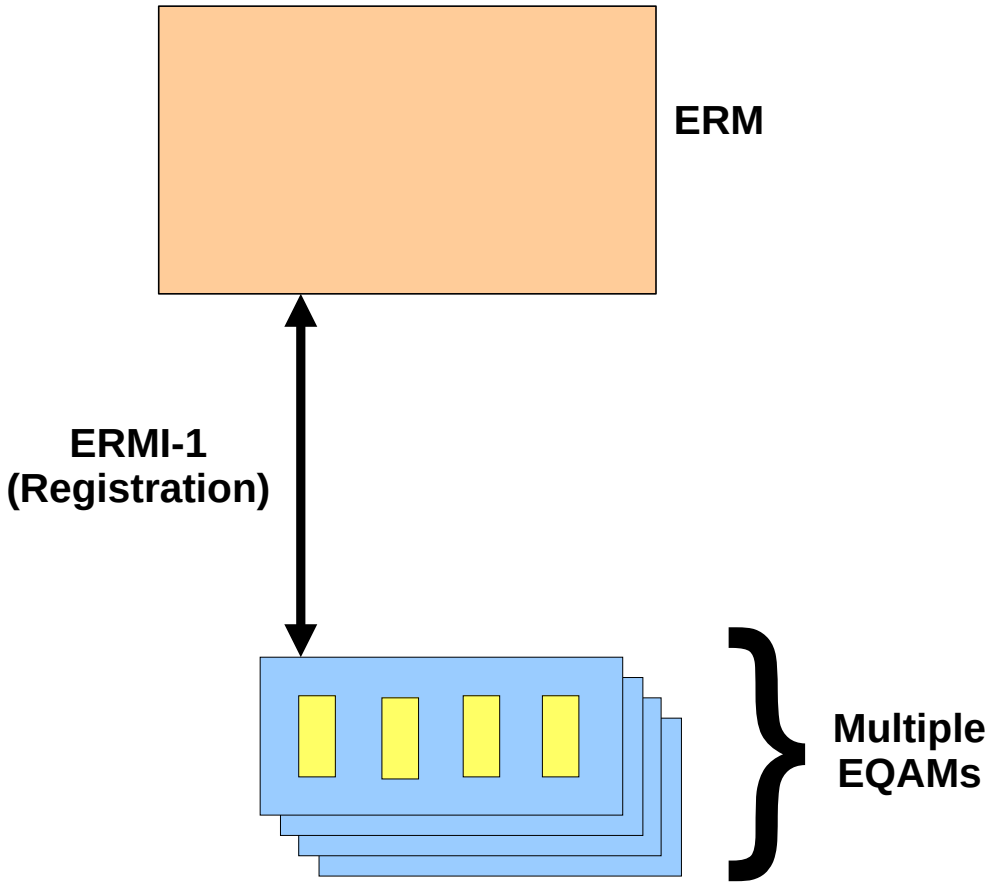

Figure 5–1 - Registration Interface and Components[2]

An ERM normally manages multiple EQAMs, as shown in the Figure 5–1. An EQAM requires an ERM IP address in order to contact the ERM. An EQAM may use a DNS query to obtain an ERM IP address. An EQAM may also be configured to communicate with a secondary ERM in the event that it can no longer communicate with the primary ERM. However, a single QAM channel is controlled by a single ERM at a time.

Messages sent using the DRRP protocol are carried over a TCP/IP connection that may be initiated by either the EQAM or the ERM.

In this specification, the term "DRRP node" (or simply "node") is sometimes used to refer to a generic entity that participates in exchanges of DRRP messages.

Because the device control interface for an EQAM is based on RTSP (see Section 7), the EQAM advertises an RTSP URL, in addition to the IP address. This RTSP URL is used to establish an RTSP transport connection with the EQAM. This allows the ERM to send device control messages to the EQAM to request QAM channel resources.

In normal use, an M-CMTS core requests a QAM channel resource from an ERM, (*i.e.,* during the process of creating a DOCSIS MAC domain (over ERMI-3)). The resources previously advertised by EQAMs (over ERMI-1) are used by the ERM to select a suitable QAM channel resource to meet the requirements of an incoming request. The ERM checks with the EQAM (over ERMI-2) that the resource is available, and that the ERM returns addressing information for that resource (over ERMI-3), to the requesting M-CMTS core.

[2] Replaced this figure per ECN ERMI-N-05.0244-1 on 11/1/05.

In addition to an initial advertisement, the EQAM sends additional advertisements whenever an attribute of a resource under its control changes. For example, if a QAM channel within an EQAM changes its carrier frequency, the EQAM will transmit the new information to the ERM. Also, if an EQAM detects a fatal failure in one of its QAM channels, it will inform the ERM.

5.1.3 DRRP Operation

This section describes how EQAMs use DRRP to register a QAM channel resource with an ERM and how ERMs use DRRP to obtain information about QAM channel resources, in order to build a database of available resources.

5.1.3.1 DRRP Addressing

A DRRP advertisement provides the mapping from an application-specific address field (in our case, a URL) of an advertised resource to an IP signaling address where control messages for that resource should be sent. The application-specific address identifies the individual resource being advertised.

When it boots, the EQAM advertises the application-specific addresses (*i.e.,* URLs) of its available QAM channel resources to the ERM. When an M-CMTS core sends a suitable request to the ERM, the ERM will select a QAM channel resource from its database. The ERM then sends an RTSP request to the EQAM to bind this QAM channel resource. After the EQAM receives a resource allocation request from the ERM, the EQAM verifies that the resource is available and that the QAM channel can be configured as requested. The EQAM then marks the resource as being in use and responds to the ERM request with an indication of success.

For each QAM channel resource that the EQAM wishes to advertise, it sends a DRRP ReachableRoute attribute to the ERM, embedded in a DRRP UPDATE message. To withdraw a QAM channel from service, the EQAM sends a WithdrawRoute attribute[3] (see Section 6.2.3.1).

The application-specific address passed in the DRRP Route attribute is the RTSP URL (*i.e.,* a URL whose protocol portion is rtsp:) of the particular resource that is being advertised.

DRRP advertisements include the IP address of the EQAM (to which RTSP signaling requests are sent) in a NextHopServer attribute. The syntax and semantics of the DRRP NextHopServer attribute are described in Section 6.2.3.3.

5.1.3.2 RTSP URLs

An RTSP URL identifies the protocol to be used to access the resource as RTSP, and contains two additional fields: hostname and abs-path, separated by a virgule: rtsp://hostname[:port]/abs-path.

The hostname field may contain either a FQDN or an IP address. In either case, it may include an optional TCP port number. The value identifies the address of the server that receives RTSP requests. In the context of the ERMI-2 interface, the ERM is an RTSP client and the EQAM is an RTSP server.

The abs-path field is used to further distinguish the entity upon which a subsequent RTSP request will operate. For example, suppose that an EQAM supports multiple QAM channels controlled by a single logical RTSP server in the EQAM. In order to control an individual QAM channel within an EQAM, an ERM must be able to identify the individual QAM channel to which a particular RTSP request applies. Because the hostname field in a URL identifies only a particular network device, it cannot identify an individual QAM channel within that device. Therefore, the abs-path field is used within an RTSP request to identify the individual destination QAM channel for the request.

Table 5–1 shows the URL format used by EQAMs when advertising QAM channels. In the table, the hostname field in the URL is represented by the string "*hostname*". The abs-path field contains the ASCII-encoded TSID of the QAM channel.

[3] The names of the Route, ReachableRoute and WithdrawRoute attributes are drawn from TRIP. Although this document retains these names, it may help the reader to think of them instead as Resource, UsableResource and WithdrawResource.

Table 5–1 - URL Formats for QAM Channels

Component	Sub-Component	URL Format	abs-path encoding
EQAM	QAM Channel	rtsp://*hostname[:port]*/<QAM ID>(*e.g.*, rtsp://192.168.0.2/1234)	QAM ID Value (which is the ASCII encoding of the TSID value for the QAM channel)

5.1.3.3 DRRP Timers

DRRP uses two timers, the Hold Timer and ConnectRetry Timer.

5.1.3.3.1 Hold Timer

Whenever a DRRP node receives a message, it resets and starts the Hold Timer. If the Hold Timer fires before it receives another message, then it sends a NOTIFICATION message which causes the DRRP connection to be closed. The duration of the Hold Timer, which is called the Hold Time, is negotiated by the DRRP nodes when the DRRP connection is established. KEEPALIVE messages should be sent at an interval of 1/3 of the Hold Time, in order to ensure that the Hold Timer does not fire.

5.1.3.3.2 ConnectRetry Timer

The ConnectRetry Timer is used during the process of establishing a DRRP connection. After the DRRP node initiates the TCP connection to the remote DRRP node, it starts the ConnectRetry Timer and waits for a response from the remote node. If the ConnectRetry Timer expires before receiving a response from the remote node, the DRRP node will retry to establish the TCP connection.

5.1.3.4 DRRP Attributes

In a DRRP UPDATE message, EQAMs may advertise the following DRRP attributes (for each QAM channel) to the ERM. Each of these attributes is specified in detail in Section 6.2.3.

- **DEPI Control Address**: This attribute identifies the destination IP address used by control packets to communicate with the QAM channel.
- **DOCSIS Capability**: This attribute identifies the type of configuration and DOCSIS modes in which the QAM channel is capable of operating.
- **NextHopServer**: This attribute identifies the EQAM to which signaling messages should be sent.
- **Port ID**: This attribute identifies the RF port of the QAM channel. (*i.e.,* the RF port to which the QAM channel is connected).
- **QAM Configuration**: This attribute identifies the current configuration of the QAM channel.
- **QAM ID**: This attribute identifies a QAM channel resource within the headend.
- **Reachable Routes**: This attribute identifies resources that are being advertised as available by the EQAM.
- **Service Status**: This attribute describes the current operational state of the QAM channel.
- **Total Bandwidth**: This attribute identifies the rate at which the QAM channel is capable of consuming data from the network.
- **WithdrawnRoutes**: This attribute identifies resources that are being withdrawn from service by the EQAM.

5.2 Resource Allocation Signaling

5.2.1 Resource Allocation Components and Interface

Figure 5-2 shows the components involved in the resource allocation interfaces.

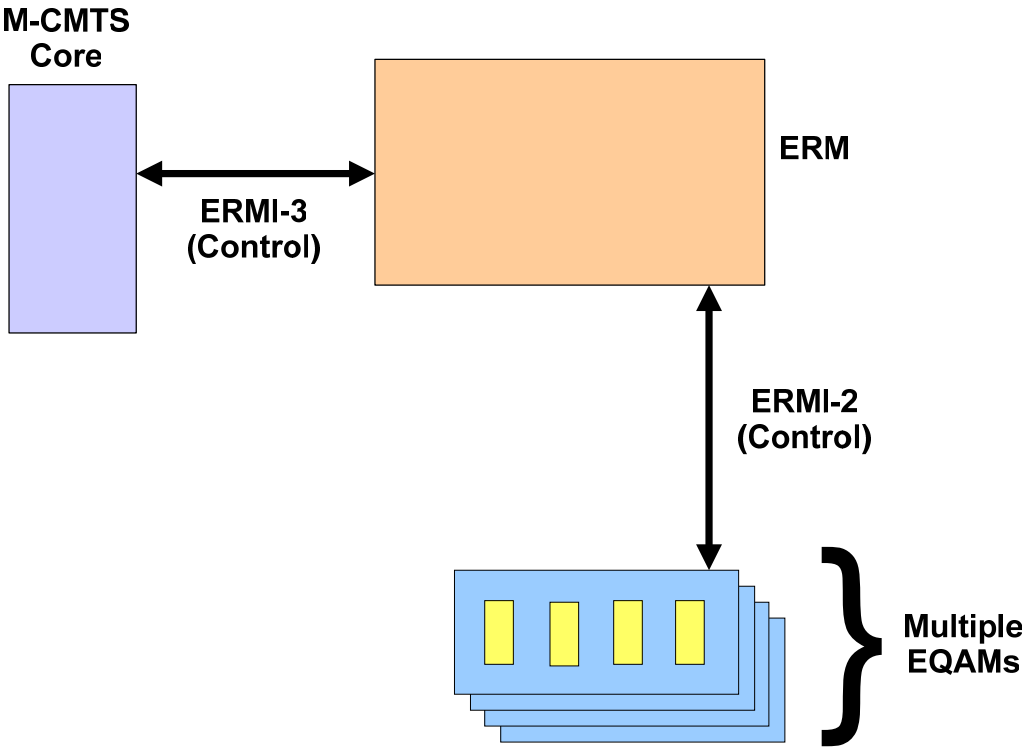

Figure 5-2 - Resource Allocation Interfaces and Components[4]

The M-CMTS core initiates a QAM resource transaction with the ERM when it requests or releases a QAM channel resource. When a MAC domain is being created, for example, the M-CMTS core provides the ERM with details of the desired service group, bandwidth, and QAM channel capability. The ERM then consults its database of available resources, verifies that the resources are available, and returns the contact information for an appropriate QAM channel, if one is available.

An EQAM is a device which has a pool of QAM channels that may be allocated to DOCSIS, or to other applications (such as video). A particular QAM channel may support only a subset of all possible DOCSIS capabilities. For example, some QAM channels may support only certain interleave settings; or some QAM channels may not support the DOCSIS PSP mode specified in [DEPI]. The capabilities of each individual QAM channel are advertised to the ERM when the EQAM advertises that particular QAM channel.

The ERM may apply operator-dependent policies when selecting a QAM channel. Such policies may take into consideration factors such as: QAM channel load balancing; whether DOCSIS bonded traffic may share a QAM channel with VOD traffic; and the existence of QAM channels that have been reserved for future DOCSIS traffic, etc.

[4] Replaced this figure per ECN ERMI-N-05.0244-1 on 11/1/05.

In an EQAM, QAM channels are physically connected to RF ports. A single RF port may be associated with multiple QAM channels. The RF port may impose limitations on the configuration of associated QAM channels. For example, all the QAM channels associated with a single RF port may be required to be configured identically. Although a QAM channel used by a video application and one used by DOCSIS may have the same carrier frequency and modulation type, they may have different interleave settings. To allow an individual QAM channel to switch between operating in a mode compatible with video and one compatible with DOCSIS, the EQAM should be able to control the configuration of a single QAM channel without affecting the configuration of any other QAM channels.

This document specifies two resource-allocation interfaces. ERMI-2 is an interface between an ERM and an EQAM, and is used to bind QAM channel resources selected by the ERM. ERMI-3 is an interface between an M-CMTS core and an ERM, and is used to request and return QAM channel resources.

5.2.2 Service Groups

The mapping between a QAM channel and its associated service group is determined by the physical wiring of the RF network. Figure 5-3 is a simplified depiction showing a small network of three QAM channels (QAMC1 through QAMC3), and three fiber nodes (FN1 through FN3). In Figure 5-3, QAMC1 and QAMC2 are cross-connected to FN1 and FN2. QAMC1 and QAMC2 therefore, belong to service group A. QAMC2 and QAMC3 are both connected to FN3. These two QAM channels therefore, belong to service group B. QAMC2 belongs to both service group A and to service group B.

A service group is identified by a list of QAM channels (or, an equivalent list of frequencies). Each QAM channel is identified by a QAM ID, which is a 16-bit MPEG Transport Stream ID (TSID), configured at the EQAM so as to be unique for each QAM channel in the head-end. TSIDs are inserted (inband) into video streams by QAM channels; it is convenient to use the same identifier for QAM channels in DOCSIS applications. In our example, QAMC1, QAMC2, and QAMC3 will be identified by TSID1, TSID2, and TSID3 respectively. Therefore, service group A is identified by the set {TSID1, TSID2}, and service group B by the set {TSID2, TSID3}.

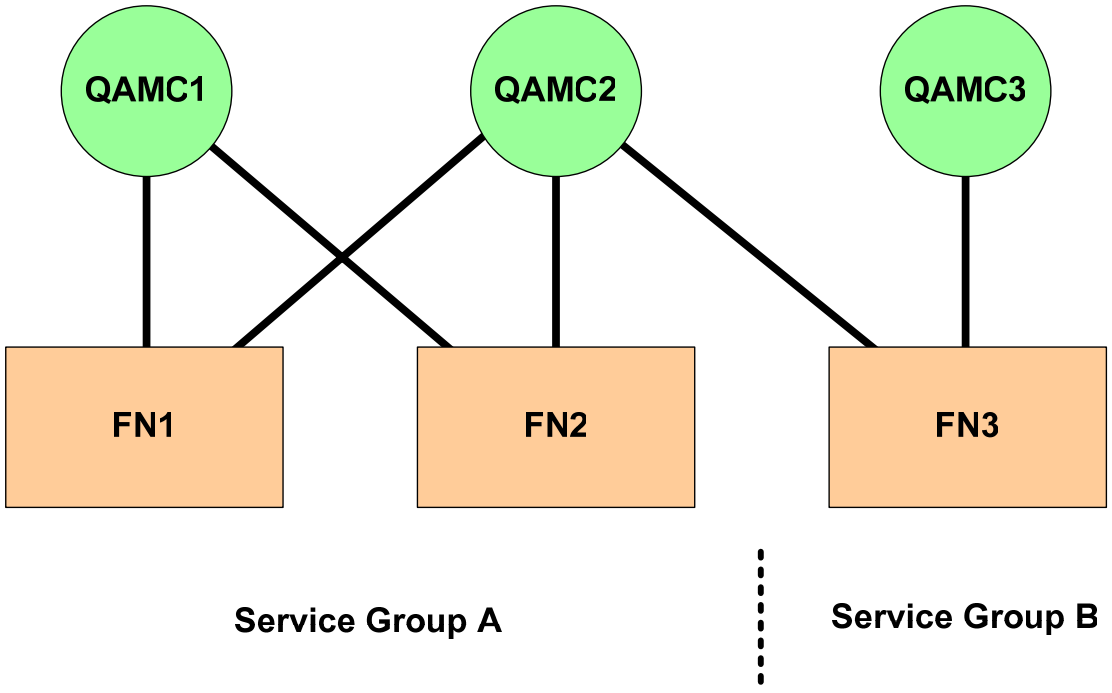

Figure 5-3 - Service Group Example5

[5] Replaced this figure per ECN ERMI-N-05.0244-1 on 11/1/05.

The mapping between QAM channel and service group can be learned either dynamically or through static configuration. In VOD applications, the subscriber's Set-Top Box scans the frequency spectrum to discover the list of visible QAM channels and their associated TSIDs. The list of visible TSIDs is passed to a VOD Session Manager. The Session Manager then forwards this list to the ERM. The method used to perform service group discovery in DOCSIS is outside the scope of this specification. If the service group information is not discovered automatically, it must be configured or otherwise made available to the M-CMTS core by some means not specified in this document.

The ERM allocates QAM channel resources based on the requested QAM ID list, bandwidth, and DOCSIS capability. The M-CMTS core includes a list of QAM IDs in its request for a resource; the ERM searches the list to find the "best" available QAM channel that meets the requirements for bandwidth and the DOCSIS channel capability and is consistent with local policy.

5.2.3 Signaling Protocol

The protocol used for the resource allocation interfaces (ERMI-2 and ERMI-3) is the Real Time Streaming Protocol, RTSP [RFC 2326]. RTSP is an application-level client/server protocol designed to control the delivery of real-time data. RTSP provides an extensible framework to enable controlled, on-demand delivery of such data. The protocol is intended to control multiple simultaneous data delivery sessions, to provide a means for choosing delivery channels, and to provide a way to choose among multiple delivery mechanisms.

The client initiates an RTSP session by sending a SETUP message containing a request for resources. The server allocates the resources for the session and replies by identifying a suitable resource that meets the client's request. In ERMI-2, the ERM is an RTSP client and the EQAM is an RTSP server. In ERMI-3, the M-CMTS core is an RTSP client and the ERM is an RTSP server.

An RTSP message is either a request or a response. A request contains an RTSP method, the object on which the method is operating, and any parameters necessary to further define the operation. Depending on the RTSP method, the direction of an RTSP request can be from client to server or *vice versa*.

In a VOD application, RTSP is extended with new methods and headers to support the VOD application. In the DOCSIS application specified in this document, additional headers are defined. Naturally, in order for an ERM/EQAM pair to function correctly in both VOD and DOCSIS applications, both devices must support the (different) RTSP extensions used by the two applications.

RTSP allows considerable variations in the capabilities supported by conformant implementations. This specification indicates the subset of RTSP requirements that must be supported by the RTSP components in the ERMI resource allocation interfaces (ERMI-2 and ERMI-3). In addition, extensions needed for the DOCSIS application are also specified herein.

5.2.4 Selecting an ERM

There may be multiple ERMs in the operator's head-end network. The M-CMTS core selects the correct ERM with which to communicate by means that are outside the scope of this document. For the simplest case, this can be achieved by static configuration. In a more dynamic network, ERMs may report resources to an aggregation point by means that are outside the scope of this document. Such an aggregation point may serve as a proxy to locate the correct ERM.

5.3 Static Partitioning

In current head-end networks, the QAM channel resources used by video applications and those used by DOCSIS applications are separate and distinct. In order to allow resources to be switched easily between these two types of application, common interfaces should be used within the ERM for registration, allocation, and control. In order to provide a relatively simple migration path, QAM channel resources may initially be partitioned by using static configuration.

Some EQAMs may control QAM channels that are capable of supporting both video and DOCSIS applications. It is permissible to provision statically, some of the QAM channels in the EQAM for video service and the rest for DOCSIS data applications.

5.3.1 Simplified Architecture for Static QAM Resource Sharing

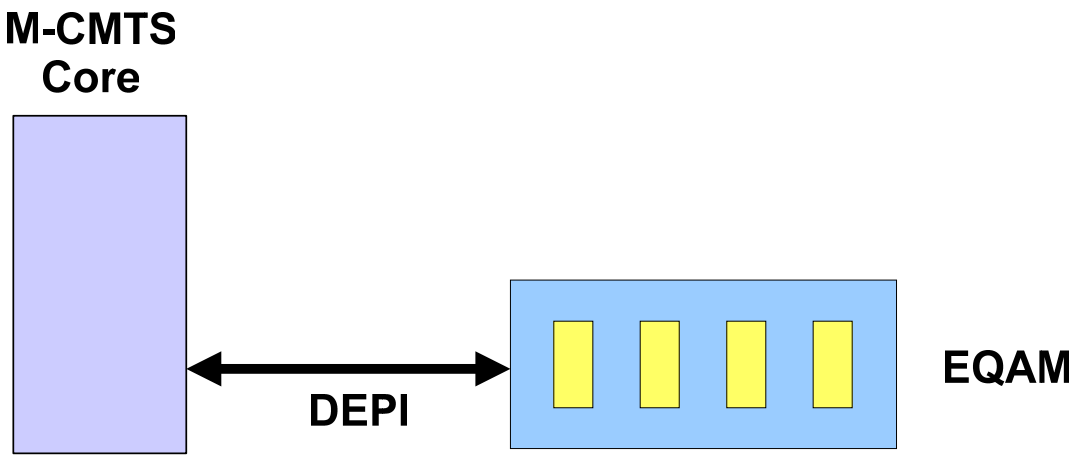

Figure 5–4 - Simplified Framework for Static QAM Partitioning

Figure 5–4 shows an architecture suitable for partitioning QAM channels statically between video and DOCSIS applications. In this simple architecture, the ERM is not used at all by DOCSIS applications.

The EQAM is configured so that only video QAM channels are advertised to the ERM (since the ERM is used only by video applications).

5.3.2 Operation

When the M-CMTS core needs a QAM channel resource, such as in the case of creating a new MAC domain, it selects a QAM channel from the appropriate service group. The M-CMTS core obtains the IP address of the QAM channel from its configuration database. The M-CMTS core then uses the DEPI protocol to establish a control channel to this IP address. Once this channel is in place, the M-CMTS core negotiates the physical settings for the QAM channel, as specified in [DEPI].

5.4 Device Configuration

The ERMI components, EQAM and M-CMTS core, must be properly configured for ERMI to function as intended. The configuration for dynamic signaling and static partioning is slightly different. The differences are noted below.

Each QAM channel in the EQAM is configured with:

- QAM ID;
- IP address, (this is the IP address used by the M-CMTS core to communicate with the QAM channel as specified by [DEPI]). UDP ports acceptable for DEPI data sessions communicating with this QAM channel can be optionally configured;
- Modulation type;
- Channel bandwidth;
- Interleave setting (I, J);
- J83 annex;
- Carrier frequency;
- Power;
- QAM capability;

- Resource allocation mode (ERM, or non-ERM);
- ERM IP address (only needed for dynamic signaling);

In the M-CMTS core, the following are configured:

- Mappings between QAM IDs and service groups;
- QAM ID for each QAM channel;
- IP address for each QAM channel (only needed for static portioning);
- IP address of ERM (only needed for dynamic signaling);
- QAM capability for each QAM channel;

as specified in the [DEPI] specification.

6 DOCSIS RESOURCE REGISTRATION PROTOCOL (DRRP) SPECIFICATION

6.1 Relationship with TRIP [RFC 3219]

DRRP shares many similarities with the protocol described in [RFC 3219]. It has similar procedures and a similar Finite State Machine for connection establishment. It also shares the same format for messages. DRRP nodes are also conformant TRIP nodes, although they perform only a subset of the messaging described in [RFC 3219].

DRRP supports four messages that may be exchanged between nodes: OPEN, UPDATE, NOTIFICATION, and KEEPALIVE:

- The OPEN message is used to initiate a DRRP connection between nodes. The OPEN message is used exactly as specified in [RFC 3219].
- The UPDATE message is used by an EQAM to advertise resources under its control to an ERM.
- The NOTIFICATION message is used by a DRRP node to inform the far end that an error has occurred. DRRP connections are closed after a NOTIFICATION message is sent or received. The NOTIFICATION message is used exactly as specified in [RFC 3219].
- DRRP includes a periodic bidirectional KEEPALIVE message whose frequency is negotiated by the two sides when the DRRP connection is established. The KEEPALIVE message is used as specified in [RFC 3219].

If the ERM does not receive a KEEPALIVE message within the agreed-upon period, it assumes that the EQAM has failed and updates its database accordingly, by marking the associated resources as no longer available. The ERM may try to re-establish the DRRP connection to that EQAM before removing that EQAM from its resource pool. The ERM can also discover a change in a QAM channel resource through receipt of an UPDATE message. Failures of QAM resources and indications of the availability of new QAM resources are conveyed to an ERM through the WithdrawnRoutes and ReachableRoutes attributes of the UPDATE message.

6.2 DRRP

This section documents the subset of [RFC 3219] used by the M-CMTS architecture, as well as the additional attributes that are used to advertise QAM channel resources. Instead of referring to the applicable sections of [RFC 3219], this section includes normatively, the relevant sections of [RFC 3219]. The resultant protocol is known as DRRP, the DOCSIS Resource Registration Protocol.

6.2.1 Establishing a DRRP Connection

DRRP messages MUST be carried over a TCP/IP connection. A DRRP node MUST listen on TCP port 6069 for incoming connections that will be used to carry DRRP traffic.

The DRRP connection may be initiated by either of the DRRP nodes.

A DRRP node MUST conform to the DRRP state machine specified in Section 6.2.7. A DRRP node begins in the [Idle] state and goes through several state transitions before reaching the [Established] state, after which point the EQAM SHOULD advertise its resources that are operational to the ERM.

6.2.2 Message Formats

Incomplete received DRRP messages MUST be ignored. The total size of a transmitted DRRP message MUST NOT exceed 4096 octets. The recipient MUST be able to process DRRP messages up to 4096 octets in size. Within a DRRP message, octets MUST be transmitted with most significant octet first and, within an octet, most significant bit first. This is commonly known as “network order”.

6.2.2.1 Message Header

Every DRRP message MUST begin with a 3-octet header, specified below. There may or may not be a data portion following the header, depending on the message type. The layout of the header fields is shown in Figure 6–1:

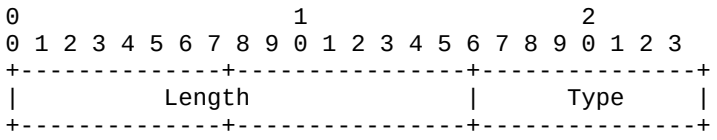

Figure 6–1 - DDRP Header Format

Length

This 2-octet unsigned integer indicates the total length of the message, including the header, in octets. Thus, it allows the recipient to locate the beginning of the next message in the message stream. The Length field MUST be present. The value of the Length field MUST be in the range $3 \leq \text{Length} \leq 4096$. The value may be further constrained by the message type. The stream of DRRP messages MUST NOT contain padding between messages.

Type

This 1-octet unsigned integer encodes the type of the message. The Type field MUST be present. The value of the field MUST be one of the values identified in Table 6–1:

Table 6–1 - DRRP Message Types

Type	DRRP message
1	OPEN
2	UPDATE
3	NOTIFICATION
4	KEEPALIVE

6.2.2.2 OPEN Message

The first DRRP message sent by a node MUST be either an OPEN message or a NOTIFICATION message sent in response to a received OPEN message.

If the OPEN message is acceptable to the recipient node, a KEEPALIVE message confirming the OPEN MUST be returned.

The minimum length of the OPEN message is 17 octets (including the message header). OPEN messages not meeting this minimum requirement are handled as defined in Section 6.2.4.2.

Following the fixed-size DRRP header, the OPEN message contains the following fields:

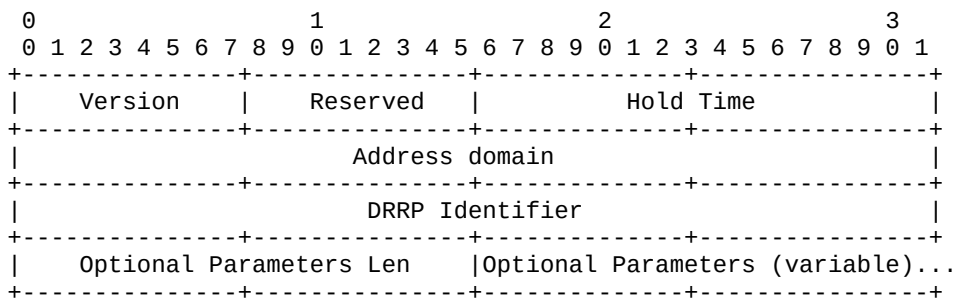

Figure 6–2 - DRRP OPEN Header

Version

This 1-octet unsigned integer indicates the protocol version of the message. The Version field MUST be present. The Version field MUST contain the value 1.

Hold Time

This 2-octet unsigned integer indicates the number of seconds that the sender proposes for the value of the Hold Timer. The Hold Time field MUST be present. Upon receipt of an OPEN message, a DRRP node MUST calculate the value of the Hold Timer, T_H, by using the smaller of its configured Hold Time (if any), and the value of the Hold Time field in the OPEN message. The value of the Hold Time field in an OPEN message MUST be either 0 or greater than two. The configured Hold Time (if any) MUST be either 0 or greater than two. A recipient SHOULD reject connections if the value of the Hold Time field does not meet local policy. The calculated value of T_H is the maximum number of seconds that may elapse between the receipt of successive KEEPALIVE and/or UPDATE messages (see Section 6.2.2.4 and Section 6.2.2.3).

Address domain

This 4-octet unsigned integer indicates the address domain number of the sender. The Address domain field MUST be present. Two DRRP nodes MUST have the same address domain in order to establish a DRRP connection, unless one has the reserved address domain number of zero.

The value of the Address domain field MUST be less than or equal to 255.

An Address domain field that contains the value 0 MUST be interpreted by the recipient to mean that the advertised address can be reached from any address domain.

DRRP Identifier

This 4-octet unsigned integer indicates the DRRP Identifier of the sender. The DRRP Identifier field MUST be present. The value of this field MUST uniquely identify this node within its address domain. A given node MAY set the value of its DRRP Identifier to an IPv4 address assigned to that node. The value of the DRRP Identifier of a DRRP node is configured on the node. The DRRP node MUST use the same value for the DRRP Identifier field in all OPEN messages sent to other nodes.

Optional Parameters Len

This 2-octet unsigned integer indicates the total length of the Optional Parameters field in octets. The Optional Parameters Len field MUST be present. If the value of this field is zero, the Optional Parameters field is absent.

Optional Parameters

This field contains optional parameters. Each parameter present MUST be encoded as a <Parameter Type, Parameter Length, Parameter Value> triple as described in Figure 6–3. The Optional Parameters field is optional; its presence depends on whether any optional parameters are being passed.

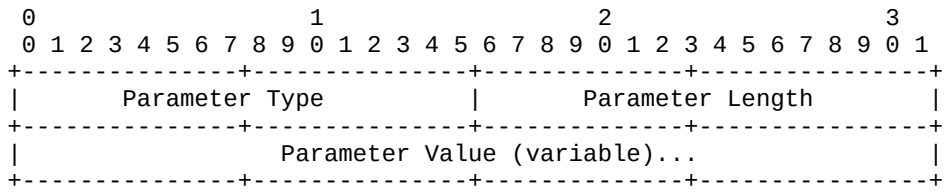

Figure 6–3 - Optional Parameter Encoding

Parameter Type

This is a 2-octet field that identifies the type of this parameter. The Parameter Type field MUST be present.

Parameter Length

This 2-octet unsigned integer contains the length of the Parameter Value field in octets. The Parameter Length field MUST be present.

Parameter Value

This is a variable length field that is interpreted according to the value of the Parameter Type field. The Parameter Value field is optional; its presence depends on the parameter being passed.

6.2.2.2.1 Open Message Optional Parameters

DRRP specifies a single optional parameter in the OPEN message: Capability Information.

6.2.2.2.1.1 Capability Information

The Capability Information parameter MUST identify its presence by setting the value of the Parameter Type field to 1.

The Capability Information parameter is used by a node to indicate the capabilities that it supports. Capability negotiation is specified in Section 6.2.6.

The parameter MUST contain one or more triples <Capability Code, Capability Length, Capability Value>, where each triple is encoded as shown in Figure 6–4:

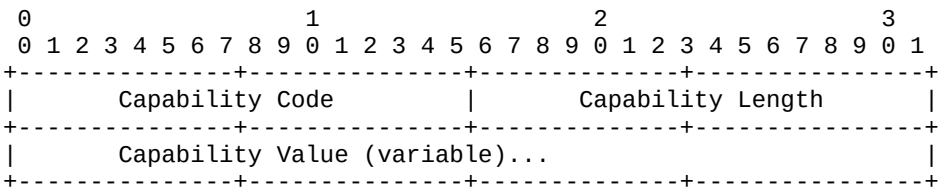

Figure 6–4 - Capability Optional Parameter

Capability Code

This is a 2-octet field, used to identify individual capabilities. The Capability Code field MUST be present.

Capability Length

This 2-octet unsigned integer contains the length of the Capability Value field in octets. The Capability Length field MUST be present.

Capability Value

This is a variable-length field that is interpreted according to the value of the Capability Code field and Capability Length field. A single capability, as identified by the value of its Capability Code field, may appear more than once in the Optional Parameters field.

The value of the Capability Code field MUST be one of the values identified in Table 6–2.

Table 6–2 - Capability Codes

Capability Code	Capability
1	Route Types Supported
2	Send Receive
32768	DRRP Version

6.2.2.2.1.2 Route Types[6] Supported

The Route Types Supported capability lists the route types supported by the transmitting node. A node MUST NOT use route types that are not supported by the remote node. If the route types supported by the remote node are not supported, a node SHOULD terminate the DRRP connection.

The format for a Route Type is shown in Figure 6–5:

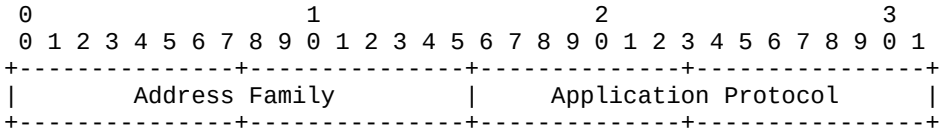

Figure 6–5 - Route Type Format

The Address Family and Application Protocol fields are specified in Section 6.2.3.1.1.1. The Address Family field identifies the address family of the resource within the ReachableRoutes attribute. The Address Family field MUST be present. The Application Protocol field identifies the application for which the resource may be used. The Application Protocol field MUST be present.

For example, a Route Type for DRRP could be <URL, ERMI>, indicating a URL for the ERMI resource signaling interface to the EQAM.

The Route Types Supported capability may contain multiple Route Types in the capability. The number of Route Types listed in the capability is limited only by the value of the Capability Length field.

6 As before, we maintain the nomenclature used in [RFC 3219].

6.2.2.2.1.3 Send Receive

This capability specifies the mode in which the DRRP node will communicate with the remote node. The possible modes are: Send Only, Receive Only, and Send Receive. The default mode is Send Receive.

In Send Only mode, a node transmits UPDATE messages to the remote node. If a node in Send Only mode receives an UPDATE message from the remote node, it MUST ignore that message.

In Receive Only mode, a DRRP node MUST NOT transmit an UPDATE message. In Send Only mode, a DRRP node MUST ignore any received UPDATE messages.

The Send Receive Capability field is a 4-octet unsigned integer. The value of the Send Receive Capability field MUST be one of the values identified in Table 6–3.

Table 6–3 - Send Receive Capability

Send Receive Capability	Meaning
1	Send Receive
2	Send Only
3	Receive Only

If a node discovers while processing an OPEN message that both it and its remote node are in Send Only mode or in Receive Only mode, it MUST send a NOTIFICATION message to the remote node to close the session. The error code in this NOTIFICATION message MUST be set to "Capability Mismatch" (see the DRRP error handling Section 6.2.4).

A node MUST send the same value of Send Receive Capability to all remote nodes. An ERM SHOULD advertise Send Receive mode if it supports both VOD and DOCSIS applications. An EQAM MUST advertise Send Only mode. DRRP component send receive capability is summarized in Table 6–4.

Table 6–4 - DRRP Component Send Receive Capability

Component	DRRP Send Receive Capability
EQAM	Send Only mode
ERM	Send Receive mode or Receive Only mode

6.2.2.2.1.4 DRRP Version

This is a 4-octet unsigned integer, representing the version of DRRP supported by the node. If a node receives an OPEN message containing a value for the DRRP Version field that it does not support, it MUST respond with a NOTIFICATION message. The error code in the NOTIFICATION message MUST be set to "Capability Mismatch" (see the DRRP error handling Section 6.2.4).

The value of the DRRP Version Capability field MUST be 1.

6.2.2.3 UPDATE Message Format

UPDATE messages are used to transfer information about resources between DRRP nodes. Information received in UPDATE messages is used to populate a database of available resources.

In particular, UPDATE messages are used to advertise and to withdraw resources from service. A single UPDATE message may simultaneously advertise and withdraw DRRP resources.

In addition to the DRRP header, the DRRP UPDATE may contain a list of so-called Routing Attributes, which MUST be formatted as shown in Figure 6–6. Routing Attributes MUST be contiguous in the message.

Figure 6–6 - DRRP UPDATE Format

The minimum length of an UPDATE message is 3 octets (*i.e.,* there are no mandatory attributes).

6.2.2.3.1 Routing Attributes

Each Routing Attribute is formatted as shown in Figure 6–7 and Figure 6–8.

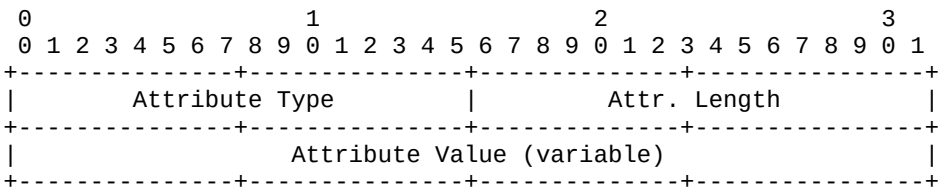

Figure 6–7 - Routing Attribute Format

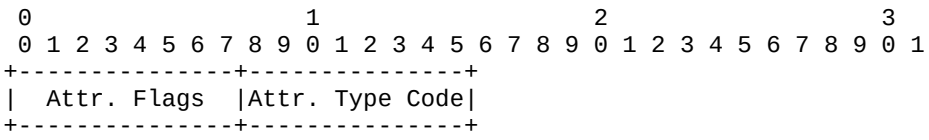

Figure 6–8 - Attribute Type Format

Attribute Type

A two-octet field that consists of two sub-fields: a one-octet Attribute Flags sub-field, followed by a one-octet Attribute Type Code sub-field. The Attribute Type field MUST be present. The Attribute Flags sub-field MUST be present. The Attribute Type Code sub-field MUST be present.

The value of the Attribute Type Code field identifies the type of the attribute. The allowable values are specified later in this section. If multiple attributes are present in an UPDATE message, they MUST be in increasing numerical order of the Attribute Type Code field. A particular value of this field MUST NOT appear more than once in a single UPDATE message. Attribute flags are used to control attribute processing when the attribute type is unknown, and are specified in Section 6.2.2.3.2.

DRRP uses TRIP as base protocol; it MUST conform to [RFC 3219] in its use of the Attribute Type Code. [RFC 3219] reserves the value of zero and defines attributes with codes between 1 and 11for this field. [RFC 3219] allows new attributes to be defined using vendor-specific codes 224 to 255. DRRP supports some of the attributes (ReachableRoutes, WithdrawnRoutes, and NextHopServer) defined in [RFC 3219]. DRRP defines more DRRP specific attributes using Attributes Type Codes between 224 and 255. These DRRP-supported attributes are further specified in Section 6.2.3.

Attributes not supported by DRRP MUST NOT be used.

Attribute Length

The Attribute Length is a two-octet unsigned integer that MUST contain the length of the Attribute Value field in octets.

Attribute Value

The remaining octets of the attribute represent the Attribute Value and are interpreted in accordance with the values of the Attribute Flags, Attribute Type Code, and Attribute Length fields. The supported attribute types, their values, and uses are defined in Section 6.2.3.

6.2.2.3.2 Attribute Flags

The Attribute Flags field is a one-octet field with the following structure:

Table 6–5 - Attribute Flag Field Bit Definition

Bit	Flag name
0 (most significant)	Well-known Flag
1 through 7	Reserved

Bit 0 MUST be unset. Messages received with bit 0 unset MUST be processed. Messages received with bit 0 unset MAY be processed. Note, per [RFC 3219], attributes are not well known if this bit is set.

Bits 1 through 7 MUST be zero on transmit. Bits 1 through 7 MUST be ignored on receipt.

6.2.2.4 KEEPALIVE Message Format

DRRP does not depend on any lower-level, keep-alive mechanism to determine whether remote nodes remain reachable.Instead, KEEPALIVE messages MUST be exchanged sufficiently often to ensure that the Hold Timer does not expire. The maximum time between the transmissions of successive KEEPALIVE messages SHOULD NOT be more than one third of the Hold Time. KEEPALIVE messages MUST NOT be sent more than once every 3 seconds.

If the negotiated Hold Time is zero, then KEEPALIVE messages MUST NOT be sent.

The KEEPALIVE message contains only a message header, so it has a length of 3 octets.

6.2.2.5 NOTIFICATION Message Format

A NOTIFICATION message MUST NOT be sent except in response to detection of an error. The TCP connection carrying the DRRP traffic MUST be torn down immediately following transmission or reception of a NOTIFICATION message.

In addition to the fixed-size DRRP header, the NOTIFICATION message contains Error Code and Error Subcode fields. The NOTIFICATION message contains a Data field if the Data field is specified for the specific values of Error Code and Error Subcode in the DRRP error handling (Section 6.2.4). The error report has the following format:

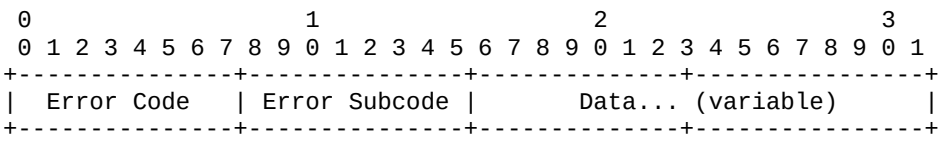

Figure 6–9 - DRRP NOTIFICATION Format

Error Code

This 1-octet unsigned integer indicates the type of the NOTIFICATION. This field MUST be present.

The value of the Error Code field MUST be one of the values identified in Table 6–6:

Table 6–6 - DRRP Error Code

Error Code	Name	Reference
1	Message Header Error	Section 6.2.4.1
2	OPEN Message Error	Section 6.2.4.2
3	UPDATE Message Error	Section 6.2.4.3
4	Hold Timer Expired	Section 6.2.4.5
5	Finite State Machine Error	Section 6.2.4.6
6	Cease	Section 6.2.4.7

Error Subcode:

This 1-octet unsigned integer provides more specific information about the nature of the reported error. The Error Subcode field MUST be present.

In the case of a Message Header Error, the value of the Error Subcode field MUST be one of the values identified in Table 6–7:

Table 6–7 - Message Header Error Subcodes

Error Subcode	Meaning
0	Unspecified error
1	Bad Message Length
2	Bad Message Type

In the case of an OPEN Message Error, the value of the Error Subcode field MUST be one of the values identified in Table 6–8.

Table 6–8 - OPEN Message Error Subcodes

Error Subcode	Meaning
0	Unspecified error
1	Unsupported Version Number
2	Bad Peer Address Domain
3	Bad DRRP Identifier
4	Unsupported Optional Parameter
5	Unacceptable Hold Time
6	Unsupported Capability
7	Capability Mismatch

In the case of an UPDATE Message Error, the value of the Error Subcode field MUST be one of the values identified in Table 6–9.

Table 6–9 - UPDATE Message Error Subcodes

Error Subcode	Meaning
0	Unspecified error
1	Malformed Attribute List
2	Unrecognized Well-known Attribute
3	Missing Well-known Mandatory Attribute
4	Attribute Flags error
5	Attribute Length error
6	Invalid Attribute

Data

This variable-length field is used to provide the reason for the NOTIFICATION. The contents of the Data field depend upon the Error Code and Error Subcode fields. The Data field MUST be present if the Data field format is specified for the particular values of the Error Code and Error Subcode fields (see the DRRP error handling Section 6.2.4).

The length of the Data field can be simply determined from the message length field:

Data Length = Message Length - 5

The minimum length of a NOTIFICATION message is 5 octets (including the message header).

6.2.3 DRRP Attributes

This section specifies the syntax and semantics of each DRRP UPDATE attribute listed in Table 6–10. The syntax of each attribute MUST conform to the descriptions in the following subsections:

Table 6–10 - DRRP Attribute Type Codes

DRRP Attribute	DRRP Type Code
WithdrawnRoutes	1
ReachableRoutes	2
NextHopServer	3
Total Bandwidth	234
Port ID	240
Service Status	241
QAM ID	246
DOCSIS Capability	247
QAM Channel Configuration	248
DEPI Control Address	249

6.2.3.1 WithdrawnRoutes

The WithdrawnRoutes attribute identifies zero or more routes that have been removed from service. When a QAM channel is out of service, the EQAM uses a WithdrawnRoutes attribute to inform the ERM that it must remove the resource from its database of available resources.

6.2.3.1.1 Syntax of WithdrawnRoutes

The WithdrawnRoutes attribute encodes a number of routes (which may be zero) in its value field. The format for individual routes is specified in Section 6.2.3.1.1.1 The WithdrawnRoutes attribute MUST list the individual routes sequentially and with no padding, as shown in Figure 6–10.

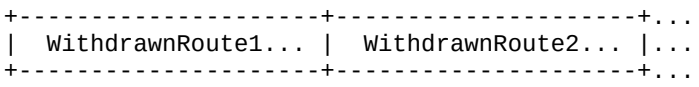

Figure 6–10 - WithdrawRoutes Format

6.2.3.1.1.1 Generic DRRP Route Format

The WithdrawRoutes and ReachableRoutes attributes share the same generic Route format. A DRRP route MUST be formatted as shown in Figure 6–11.

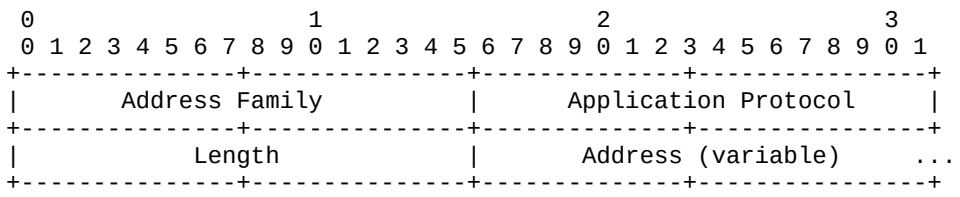

Figure 6–11 - Route Format for WithdrawRoutes and ReachableRoutes

Address Family

The Address Family field gives the type of address for the resource. The Address Family field MUST be present. The Address Family field MUST contain the value given in Table 6–11.

Table 6–11 - Values for Address Family

Value of Address Family field	Meaning
32769	URL

The syntax and use of the URL Address Family is described in Section 5.1.3.2.

Application Protocol

The Application Protocol field identifies the protocol for which the resource database is maintained. The Application Protocol field MUST be present. The Application Protocol field MUST contain the value given in Table 6–12:

Table 6–12 - Application Protocols Supported in DRRP

Application Protocol Code	Resource Allocation Interface	Protocol
32768	ERMI-2	RTSP

Length

This field is a 2-octet unsigned integer containing the length of the Address field, in octets. The Length field MUST be present.

Address

This is an address of the family type given by the Address Family field. The Address field MUST be present. The length of the Address field is variable and is given by the value contained in the Length field. If Address Family field contains 32769, the Address field MUST be a valid RTSP URL formatted as:

rtsp://hostname/abs_path

where the string hostname MUST be formatted as:

host[:port]

where

host =An FQDN; or

an IPv4 address using the textual representation defined in Section 2.1 of [RFC 1123]; or

an IPv6 address using the textual representation defined in Section 2.2 of [RFC 2373] and enclosed in "[" and "]" characters.

port = numerical value

The abs-path field MUST be the ASCII-encoded TSID of the QAM channel.

6.2.3.2 ReachableRoutes

The ReachableRoutes attribute identifies zero or more routes that have been placed in service.

6.2.3.2.1 Syntax of ReachableRoutes

The ReachableRoutes attribute encodes a number of routes (which may be zero) in its value field. Different QAM channel resources MUST be represented using different routes. The format for individual routes is specified in section 6.2.3.1.1.1. The ReachableRoutes attribute MUST list the individual routes sequentially and with no padding, as shown in Figure 6–12.

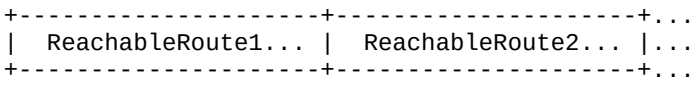

Figure 6–12 - ReachableRoutes Format

6.2.3.2.2 Resource Selection and ReachableRoutes

An EQAM advertises its QAM channel resource(s) using one or more ReachableRoutes attributes. When a QAM channel resource becomes available for use, the ReachableRoutes attribute is used to advertise the availability of that resource. The ERM uses the resources advertised in the ReachableRoutes attributes to populate its database of resources.

6.2.3.3 NextHopServer

The value of the NextHopServer attribute for protocol P and destination D indicates to the recipient the identity of the server to which messages of protocol P and with destination D are to be sent.

The NextHopServer attribute MAY identify the TCP port number for the next-hop signaling server. If the TCP port number is not identified, then the default port of the signaling protocol MUST be used.

The address identified by the NextHopServer attribute is specific to the set of destinations and application protocol identified in the ReachableRoutes attribute. This is not necessarily the address to which media (voice, video, *etc*.) will be transmitted.

A NextHopServer Attribute MUST be present if a ReachableRoutes or a WithdrawnRoutes Attribute is present.

6.2.3.3.1 NextHopServer Syntax

In order to support a variety of protocols, the identity of the next-hop server may be provided in any one of several formats (such as FQDN, IPv4, IPv6, etc.). The NextHopServer attribute includes the address domain number of the next-hop server, a length field, and a next-hop name or address.

The syntax for the NextHopServer attribute is shown in Figure 6–13.

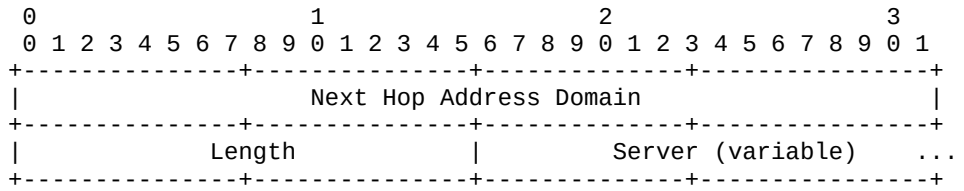

Figure 6–13 - NextHopServer Syntax

Next Hop Address Domain

The Next-Hop Address Domain field MUST be present. This 4-octet unsigned integer indicates the address domain number of the next-hop server.

Length

This is a two-octet unsigned integer containing the length of the Server field, in octets. The Length field MUST be present.

Server

This field identifies the next-hop server. The Server field MUST be present. The Server field MUST contain a string that conforms to the following syntax:

Server = host[:port]

where

host = An FQDN, or

an IPv4 address using the textual representation defined in Section 2.1 of [RFC 1123], or
an IPv6 address using the textual representation defined in Section 2.2 of [RFC 2373] and enclosed in "[" and "]" characters.

port = numerical value

6.2.3.4 QAM ID

The value of the QAM ID uniquely identifies a QAM channel within the head-end.

The syntax for the QAM ID attribute MUST be as shown in Figure 6–14:

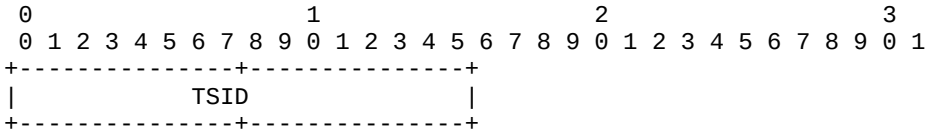

Figure 6–14 - QAM ID Syntax

The value of the QAM ID MUST be equal to the TSID of the QAM channel. (A TSID is a provisioned 16-bit value that is unique to each QAM channel in a head-end.) The QAM ID attribute MUST be present if a ReachableRoutes attribute is present.

6.2.3.5 DOCSIS Capability

The DOCSIS Capability attribute is used to advertise the ability of a QAM channel to support different types of DOCSIS operation. The DOCSIS Capability attribute MUST be present if a ReachableRoutes attribute is present.

The syntax for the DOCSIS Capability attribute MUST be as shown in Figure 6–15:

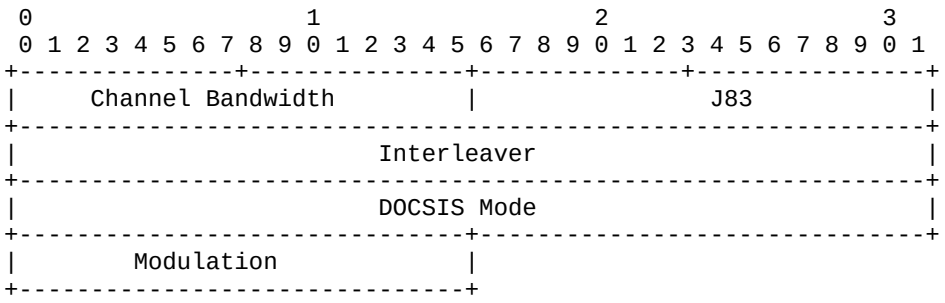

Figure 6–15 - DOCSIS Capability Format

The fields of the DOCSIS Capability attribute are further defined in Table 6–13 through Table 6–17. A bit value of 1 means that the corresponding capability is supported and a bit value of 0 means that the corresponding capability is not supported. Bit 0 is the most significant bit. All reserved bits MUST be set to 0.

The first bit in each field is the Lock bit. If the Lock bit is set, the referenced parameter MUST NOT be changed. If the Lock bit is not set, the referenced parameter MAY be changed. The next 7 bits comprise the QAM Group ID. If the referenced parameter is common to a group of QAM channels, the QAM Group ID MUST be a non-zero number that identifies the QAM group in an EQAM. Otherwise, the QAM Group ID MUST be set to 0.

Channel Bandwidth

The Channel Bandwidth field is used to describe channel bandwidth capabilities for the QAM channel.

Table 6–13 - QAM Channel Bandwidth Capability Bits

Channel Bandwidth Capability Bit	Description
0	Lock
1-7	QAM Group ID
8	6 MHz channel bandwidth
9	8 MHz channel bandwidth
10-15	Reserved

J83

The J83 field is used to describe which of the operational modes, defined in [J.83], are supported by the QAM channel.

Table 6–14 - J83 Capability Bits

J.83 Capability Bit	Description
0	Lock
1-7	QAM Group ID
8	Annex A
9	Annex B
10	Annex C
11-15	Reserved

Interleaver

The Interleaver field is used to identify the combinations of interleaver filter tap (I) and interleaver increment (J), as defined by [J.83], that are supported by the QAM channel.

Table 6–15 - QAM Interleaver Capability Bits

Interleaver Capability Bit	Description
0	Lock
1-7	QAM Group ID
8	I=8, J=16
9	I=16, J=8
10	I=32, J=4
11	I=64, J=2
12	I=128, J=1
13	I=128, J=2
14	I=128, J=3
15	I=128, J=4
16	I=128, J=5
17	I=128, J=6
18	I=128, J=7
19	I=128, J=8
20-31	Reserved

DOCSIS Mode

The DOCSIS Mode field is used to describe DOCSIS modes that are supported by the QAM channel.

Table 6–16 - DOCSIS Mode Capability Bits

DOCSIS mode Capability Bit	Description
0	Lock
1-7	QAM Group ID
8	Mixed Video/Data
9	Mixed DOCSIS mode
10-15	Reserved
16	Video
17	DOCSIS MPT
18	DOCSIS PSP
19-31	Reserved

Video mode (bit 16) is the operation mode for MPEG-2 transport video over QAM. DEPI modes, such as DOCSIS MPT and DOCSIS PSP are further defined in [DEPI]. The Mixed DOCSIS mode bit MUST be set only when DOCSIS MPT and DOCSIS PSP data can be freely mixed in a single QAM channel. The supported combination of DOCSIS modes is determined jointly by bits 9, 17, and 18. The Mixed Video/Data bit MUST be set only when the supported combination of DOCSIS modes can be mixed with video in the same QAM channel.

Modulation

The Modulation field is used to describe QAM modulation modes that are supported by the QAM channel.

Table 6–17 - QAM Modulation Capability Bits

Modulation Capability Bit	Description
0	Lock
1-7	QAM Group ID
8	64 QAM
9	256 QAM
10-15	Reserved

6.2.3.6 Total Bandwidth

The Total Bandwidth attribute is used to define the total amount of bandwidth that the downstream resource is capable of receiving (from the network), processing, and transmitting. The Total Bandwidth attribute MUST be present if a ReachableRoutes attribute is present.

The syntax for the Total Bandwidth attribute MUST be as in Figure 6–16.

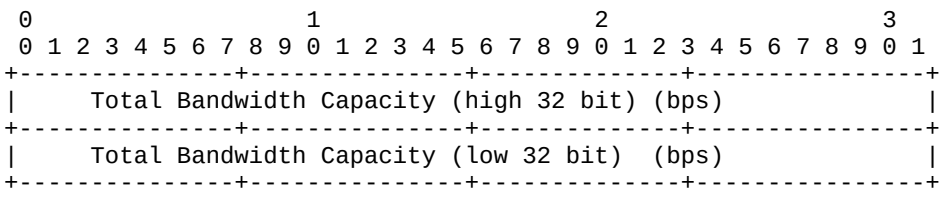

Figure 6–16 - Total Bandwidth Syntax

Total Bandwidth Capacity

This is an unsigned 64-bit integer that contains the bandwidth of the resource in units of bits per second.

6.2.3.7 DEPI Control Address

The DEPI Control Address attribute identifies the IP address that is used in the destination IP address field of control streams that are terminated at the QAM channel.

The DEPI Control Address attribute MUST be present if a ReachableRoutes attribute is present.

The syntax for the DEPI Control Address attribute is as shown Figure 6–17.

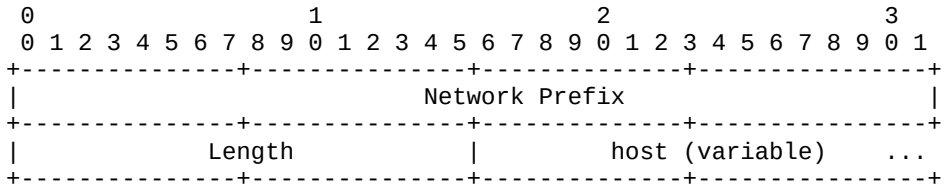

Figure 6–17 – DEPI Control Address Attribute

Network Prefix

The Network Prefix field contains the number of significant bits of the IP address used to identify a network. For IPv4, the value of the Network Prefix field MUST be less than or equal to 32. For IPv6, the value of the Network Prefix field MUST be less than or equal to 128. The Network Prefix field MUST be present.

Length

The Length field is an unsigned two-octet integer that contains the length of the host field, in octets. The Length field MUST be present.

host

The host field MUST be present. The host field MUST contain a string that represents:

- an FQDN, or
- an IPv4 address using the textual representation defined in Section 2.1 of [RFC 1123], or
- an IPv6 address using the textual representation defined in Section 2.2 of [RFC 2373] and enclosed in "[" and "]" characters

6.2.3.8 QAM Channel Configuration

The QAM Channel Configuration attribute is used by EQAMs to inform the ERM of the modulation frequency, modulation type, ITU-T J.83 operation mode, channel bandwidth, and interleaver parameters configured for an advertised QAM channel.

The QAM Channel Configuration attribute MUST be present if a ReachableRoutes attribute is present.

The syntax for the QAM Channel Configuration attribute is shown in Figure 6–18.

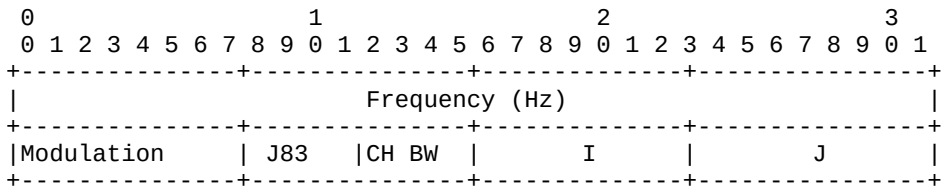

Figure 6–18 - QAM Configuration Attribute

Frequency

The Frequency field is a 32-bit unsigned integer containing the carrier frequency of the QAM channel, in hertz. The Frequency field MUST be present.

Modulation

The Modulation field is an 8-bit field that encodes the modulation type of the QAM channel. The Modulation field MUST be present. The Modulation field MUST contain a value from Table 6–18.

Table 6–18 - QAM Modulation Types

Modulation Value	Description
0	64-QAM
1	256-QAM

J83

The J83 field is a 4-bit field that encodes the J83 mode (see [J.83]) that the QAM channel is using. The J83 field MUST be present. The J83 field MUST contain a value from Table 6–19.

Table 6–19 - QAM J83 Modes

J83	Description
0	Annex A
1	Annex B
2	Annex C

CH BW

The CH BW field is a 4-bit field that encodes the bandwidth supported by the QAM channel. The CH BW field MUST be present. The CH BW field MUST contain a value from Table 6–20.

Table 6–20 - Channel Bandwidth Types

Channel Bandwidth	Description
0	6 MHz channel
1	8 MHz channel

I, J

The interleaver filter tap (I) and interleaver increment (J) values MUST be one of the pairs as defined in Table 6–15. The I and J fields MUST be present.

6.2.3.9 Port ID

The Port ID attribute identifies the RF port to which the QAM channel is attached. The Port ID attribute MUST be present if a ReachableRoutes attribute is present.

The Port ID attribute is a 32-bit value that MUST have the format in Figure 6–19:

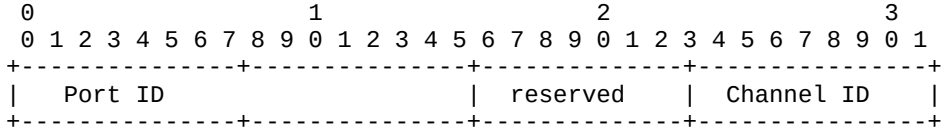

Figure 6–19 - Port ID Format

Port ID

The Port ID field contains a 16-bit identifier unique within the EQAM making this DRRP advertisement.

Channel ID

The Channel ID field contains an 8-bit QAM channel identifier unique within the RF port.

6.2.3.10 Service Status

The Service Status attribute identifies the operational status of a QAM channel. The Service Status attribute MUST be present if a ReachableRoutes attribute is present.

The syntax for the Service Status attribute MUST be as shown in Figure 6–20.

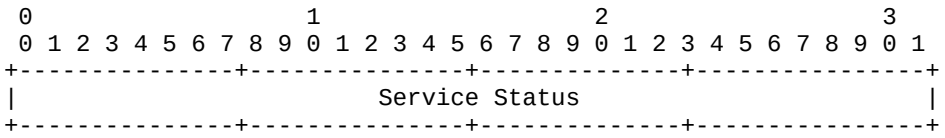

Figure 6–20 - Service Status Format

Service Status

The Service Status field contains a 32-bit unsigned integer that encodes the status of an EQAM. The value of the Service Status field MUST be taken from Table 6–21:

Table 6–21 - Service Status Values

Value	Meaning of Service Status
1	Operational – The device is operational. The resource manager is permitted to allocate resources from this device.
2	Shutting Down – The device is being shut down properly. The ERM SHOULD NOT allocate resources from this device for new sessions. When all sessions on this device have been torn down, the EQAM MUST advertise that its resources have been withdrawn from service (by using the DRRP Withdrawn-Routes attribute).
3	Standby – This is a redundant EQAM that may be used by the resource manager to replace a failed EQAM. Details of the operation of components during failover are not specified in this document.

6.2.4 DRRP Error Detection and Handling

This section and its subsections specify errors to be detected and the actions to be taken while processing DRRP messages.

If any of the conditions described in subsections below are detected:

1. A NOTIFICATION message with the indicated Error Code, Error Subcode, and Data fields MUST be sent;
2. The TCP connection carrying the DRRP messages MUST be torn down;
3. The ERM MUST treat any QAM channel resources previously advertised through this DRRP connection as being unavailable;

4. The EQAM SHOULD NOT disrupt active QAM channel resource reservations (QAM channel resources with active inbound data traffic).

Unless specified otherwise, the Data field of the NOTIFICATION message that is sent to indicate an error MUST be empty.

6.2.4.1 Errors in Message Headers

Errors detected while processing a Message Header MUST be indicated by sending a NOTIFICATION message with the Error Code field set to the value "Message Header Error". The checks in this section MUST be performed upon receipt of every DRRP message.

If the Length field of the message violates any of the requirements summarized in this section, then:

- the Error Subcode field MUST be set to "Bad Message Length"
- the Data field MUST contain the value from the erroneous Length field

The Length field requirements are:

- The Length field of the message header MUST NOT contain a value smaller than 3, nor greater than 4096;
- The Length field of an OPEN message MUST NOT contain a value smaller than 17;
- The Length field of an UPDATE message MUST NOT contain a value smaller than 3;
- The Length field of a KEEPALIVE message MUST contain the value 3;
- The Length field of a NOTIFICATION message MUST NOT contain a value smaller than 5.

If the Type field of the message header is not recognized, then:

- the Error Subcode field MUST be set to "Bad Message Type";
- the Data field MUST contain the value from the erroneous Type field.

6.2.4.2 Errors in OPEN Messages

Errors detected while processing an OPEN message MUST be indicated by sending a NOTIFICATION message with the Error Code field set to the value "OPEN Message Error". The checks in this section MUST be performed upon receipt of every OPEN message.

If the version number contained in the Version field of the received OPEN message is not supported, then:

- the Error Subcode field MUST be set to "Unsupported Version Number";
- the value of the Data field MUST be set to a 1-octet unsigned integer, indicating the largest supported version number.

If the value contained in the Address Domain field is unacceptable, then the Error Subcode field MUST be set to "Bad Address Domain".

If the value contained in the Hold Time field is unacceptable, then the Error Subcode field MUST be set to "Unacceptable Hold Time". Hold Time values of one and two seconds MUST be rejected.

If the value contained in the DRRP Identifier field is invalid, then the Error Subcode field MUST be set to "Bad DRRP Identifier". A DRRP identifier is four octets in length and can take any value. The recipient of an OPEN message MUST treat the contents of the DRRP Identifier field as invalid if it already has a DRRP connection in place with the same address domain and DRRP identifier.

If one of the Optional Parameters in the OPEN message is not recognized, the Error Subcode field MUST be set to "Unsupported Optional Parameter".

If the Optional Parameters of the OPEN message include Capability Information with any capability that has an unsupported capability type, or an unsupported capability value, the Error Subcode field MUST be

set to "Unsupported Capability". In this case, the unsupported capability (type and value) MUST be listed in the Data field of the NOTIFICATION message.

If the Optional Parameters of the OPEN message include Capability Information that does not match the recipient's capabilities, the Error Subcode field MUST be set to "Capability Mismatch". In this case, all the mismatched capabilities MUST be listed in the Data field of the NOTIFICATION message.

6.2.4.3 Errors in UPDATE Messages

Errors detected while processing an UPDATE message MUST be indicated by sending a NOTIFICATION message with the Error Code field set to the value "UPDATE Message Error". The checks in this section MUST be performed upon receipt of every UPDATE message.

If any recognized attribute has Attribute Flags that conflict with the Attribute Type Code, then:

- the Error Subcode field MUST be set to "Attribute Flags Error"
- the Data field MUST contain the erroneous attribute (type, length, and value)

If any recognized attribute has an Attribute Length that conflicts with the expected length (based on the Attribute Type Code), then:

- the Error Subcode field MUST be set to "Attribute Length Error"
- the Data field MUST contain the erroneous attribute (type, length and value)

If a required attribute is not present, then:

- the Error Subcode field MUST be set to "Missing Well-known Mandatory Attribute"
- the Data field MUST contain the Attribute Type Code of the missing attribute

If an attribute with the Well Known flag set to zero is not recognized, then:

- the Error Subcode field MUST be set to "Unrecognized Well-known Attribute"
- the Data field MUST contain the unrecognized attribute (type, length and value)

If any attribute has a value that is syntactically incorrect, undefined, or is an invalid value, then:

- the Error Subcode field MUST be set to "Invalid Attribute"
- the Data field MUST contain the incorrect attribute (type, length and value)

If any attribute appears more than once in the UPDATE message, the Error Subcode field MUST be set to "Malformed Attribute List".

6.2.4.4 Errors in NOTIFICATION Messages

Errors detected when processing NOTIFICATION messages SHOULD be logged to some error reporting and recording facility.

6.2.4.5 Hold Timer Expiration

If a system does not receive successive messages within the period required by the negotiated Hold Time, a NOTIFICATION message MUST be sent with the Error Code field set to the value "Hold Timer Expired".

6.2.4.6 Errors in the Finite State Machine

Errors detected by the DRRP Finite State Machine (see Section 6.2.7) (e.g., receipt of an unexpected event) MUST cause a NOTIFICATION message to be sent with the Error Code field set to the value "Finite State Machine Error".

6.2.4.7 Cease

In the absence of any fatal errors (defined in Section 6.2.4), a DRRP node SHOULD close its DRRP connection by sending a NOTIFICATION message with the Error Code field set to the value "Cease". The Cease NOTIFICATION message MUST NOT be used when a fatal error has been detected.

6.2.4.8 Connection Collision Detection

If two DRRP nodes try simultaneously to establish a DRRP connection to each other, a race condition exists, with the possibility that two parallel connections between these nodes might be created. We refer to this situation as a "connection collision".

Upon receipt of an OPEN message, the recipient MUST examine all its connections that are in the [OpenConfirm] state (see Section 6.2.7.5). If it finds a connection in the [OpenConfirm] state with the remote node that was the source of the OPEN message, it MUST cleanly tear down (*i.e.,* by transmitting a Cease NOTIFICATION) the connection with the lower numerical value of DRRP Identifier.

Upon receipt of an OPEN message, the recipient MAY examine connections in the [OpenSent] state if it knows the DRRP Identifier of the other node by means outside the protocol. If it finds a connection in the [OpenSent] state with the entity that was the source of the OPEN message, it MUST cleanly tear down (*i.e.,* by transmitting a Cease NOTIFICATION) the connection with the lower numerical value of DRRP Identifier.

6.2.5 Negotiating the DRRP Version

Nodes may negotiate the version of DRRP by making multiple attempts to open a DRRP connection, starting with the highest version number each supports. If an attempt to open a connection fails with the Error Code "OPEN Message Error" and the Error Subcode "Unsupported Version Number", then the node has available: the version number it tried; the version number that the remote node tried; the version number passed by the remote node in the NOTIFICATION message; and the version numbers that it supports. If the two nodes support one or more common versions, this will allow them to determine the highest version they have in common.

6.2.6 DRRP Capability Negotiation

A DRRP node MAY include the Capabilities Option in its OPEN message. A remote DRRP node that receives an OPEN message MUST NOT use any capabilities that were not included in the OPEN message when communicating with that node.

6.2.7 DRRP Finite State Machine

This section specifies DRRP operation in terms of a Finite State Machine (FSM). Implementations of DRRP nodes that conform to this specification MUST operate in accordance with the FSM described in this section. A DRRP node MUST implement an independent FSM for every DRRP connection.

The FSM MUST contain six states:

Table 6–22 - DRRP FSM States

State Name	Brief State Description
[Idle]	Initial state
[Connect]	TCP connection pending or open
[Active]	Listening for connection from remote node
[OpenSent]	An OPEN has been sent
[OpenConfirm]	Waiting for a KEEPALIVE or NOTIFICATION response to an OPEN
[Established]	DRRP connection ready for use

The FSM MUST contain 13 events:

Table 6–23 - DRRP FSM Events

Event Name	Brief Event Description
{DRRP Start}	The node is instructed to open a connection to a remote node
{DRRP Stop}	The node is instructed to end the DRRP session
{DRRP TCP connection open}	A TCP connection has been successfully created
{DRRP TCP connection closed}	The TCP connection has been closed
{DRRP TCP connection open failed}	An attempt to establish a TCP connection has failed
{DRRP TCP fatal error}	The established TCP connection has terminated unexpectedly
{ConnectRetry timer expired}	The ConnectRetry timer has fired
{Hold timer expired}	The Hold timer has fired
{Keepalive timer expired}	The Keepalive timer has fired
{Receive OPEN message}	An error-free OPEN message has been received
{Receive KEEPALIVE message}	An error-free KEEPALIVE message has been received
{Receive UPDATE message}	An error-free UPDATE message has been received
{Receive NOTIFICATION message}	An error-free NOTIFICATION message has been received

State transitions in the FSM MUST conform to Table 6–24 through Table 6–29. Following each table is text that specifies the details of each table.

The FSM MUST begin in the [Idle] state.

6.2.7.1 [Idle] State

Table 6–24 - DRRP FSM Transitions from [Idle]

Initial State	Event	Final State
[Idle]	{DRRP Start}	[Connect]
[Idle]	{DRRP Stop}	[Idle]
[Idle]	{DRRP TCP connection open}	[Idle]
[Idle]	{DRRP TCP connection closed}	[Idle]
[Idle]	{DRRP TCP connection open failed}	[Idle]
[Idle]	{DRRP TCP fatal error}	[Idle]
[Idle]	{ConnectRetry timer expired}	[Idle]
[Idle]	{Hold timer expired}	[Idle]
[Idle]	{Keepalive timer expired}	[Idle]

Initial State	Event	Final State
[Idle]	{Receive OPEN message}	[Idle]
[Idle]	{Receive KEEPALIVE message}	[Idle]
[Idle]	{Receive UPDATE message}	[Idle]
[Idle]	{Receive NOTIFICATION message}	[Idle]

Initially, the node is in the [Idle] state. In this state, the node ignores all incoming connections.

In response to the {DRRP Start} event (initiated by either the system or the operator), the node MUST:

- initialize DRRP resources;
- start the ConnectRetry timer;
- attempt to establish a TCP connection to the remote node;
- listen for a TCP connection from the remote node;
- enter the [Connect] state.

The value of the ConnectRetry timer MUST be sufficiently large to allow TCP initialization. The ConnectRetry timer value SHOULD be 120 seconds.

If a node detects an error when in some other state, it closes the TCP connection and changes its state to [Idle]. As Table 6–25 shows, transitioning from the [Idle] state requires receipt of the {DRRP Start} event. If such an event is generated automatically, persistent errors may result in flapping of the node. To avoid flapping when {DRRP Start} events are created automatically, whenever a node has transitioned to [Idle] because of an error, the time between automatically generated {DRRP Start} event MUST increase exponentially up to some maximum value. The initial value of the timer that generates the {DRRP Start} events SHOULD be 60 seconds. The value of the exponent MUST be at least two. The maximum value of the retry timer SHOULD be at least 900 seconds.

6.2.7.2 [Connect] State

Table 6–25 - DRRP FSM Transitions from [Connect]

Initial State	Event	Final State
[Connect]	{DRRP Start}	[Connect]
[Connect]	{DRRP Stop}	[Idle]
[Connect]	{DRRP TCP connection open}	[OpenSent]
[Connect]	{DRRP TCP connection closed}	[Idle]
[Connect]	{DRRP TCP connection open failed}	[Active]
[Connect]	{DRRP TCP fatal error}	[Idle]
[Connect]	{ConnectRetry timer expired}	[Connect]
[Connect]	{Hold timer expired}	[Idle]
[Connect]	{Keepalive timer expired}	[Idle]

Initial State	Event	Final State
[Connect]	{Receive OPEN message}	[Idle]
[Connect]	{Receive KEEPALIVE message}	[Idle]
[Connect]	{Receive UPDATE message}	[Idle]
[Connect]	{Receive NOTIFICATION message}	[Idle]

In this state, a DRRP node is waiting for a TCP protocol connection to be completed to a remote node, and is listening for inbound TCP connections from that node.

If the TCP connection succeeds, the node MUST:

- clear the ConnectRetry timer;
- send an OPEN message to the remote node;
- set its Hold Timer to a value of at least 240 seconds;
- start the Hold Timer;
- enter the [OpenSent] state.

If the attempt to open a TCP connection fails, (*e.g.,* because of a retransmission timeout), the node MUST:

- restart the ConnectRetry timer;
- continue to listen for a connection from the remote node;
- enter the [Active] state.

If the ConnectRetry timer expires, the node MUST:

- cancel any DRRP TCP connection to the remote node;
- restart the ConnectRetry timer;
- initiates a TCP connection to the remote node;
- continue to listen for a TCP connection from the remote node.

If an inbound TCP connection succeeds, the node MUST:

- clear the ConnectRetry timer;
- complete any necessary internal initialization;
- send an OPEN message to the remote node;
- set its Hold Timer to a value of at least 240 seconds;
- start the Hold Timer;
- enter the [OpenSent] state.

The {DRRP Start} event MUST be ignored.

In response to any other event (initiated by either the system or the operator), the node MUST:

- release all DRRP resources associated with the connection;
- enter the [Idle] state.

6.2.7.3 [Active] State

Table 6–26 - DRRP FSM Transitions from [Active]

Initial State	Event	Final State
[Active]	{DRRP Start}	[Active]
[Active]	{DRRP Stop}	[Idle]
[Active]	{DRRP TCP connection open}	[OpenSent]
[Active]	{DRRP TCP connection closed}	[Idle]
[Active]	{DRRP TCP connection open failed}	[Active]
[Active]	{DRRP TCP fatal error}	[Idle]
[Active]	{ConnectRetry timer expired}	[Connect]
[Active]	{Hold timer expired}	[Idle]
[Active]	{Keepalive timer expired}	[Idle]
[Active]	{Receive OPEN message}	[Idle]
[Active]	{Receive KEEPALIVE message}	[Idle]
[Active]	{Receive UPDATE message}	[Idle]
[Active]	{Receive NOTIFICATION message}	[Idle]

In this state, a DRRP node is listening for an inbound connection from the remote node, but is not in the process of initiating a connection to the remote node.

If an inbound attempt to create a TCP connection succeeds, the node MUST:

- clear the ConnectRetry timer;
- complete initialization;
- send an OPEN message to the remote node;
- set its Hold Timer to at least 240 seconds;
- start the Hold Timer;
- enter the [OpenSent] state.

If the ConnectRetry timer expires, the node MUST:

- restart the ConnectRetry timer;
- initiate a TCP connection to the remote node;
- continue to listen for a connection from the remote node;
- enter the [Connect] state.

In response to any other event (initiated by either the system or the operator), the node MUST:

- release all DRRP resources associated with the connection;
- enter the [Idle] state.

6.2.7.4 [OpenSent] State

Table 6–27 - DRRP FSM Transitions from [OpenSent]

Initial State	Event	Final State
[OpenSent]	{DRRP Start}	[OpenSent]
[OpenSent]	{DRRP Stop}	[Idle]
[OpenSent]	{DRRP TCP connection open}	[Idle]
[OpenSent]	{DRRP TCP connection closed}	[Active]
[OpenSent]	{DRRP TCP connection open failed}	[Idle]
[OpenSent]	{DRRP TCP fatal error}	[Idle]
[OpenSent]	{ConnectRetry timer expired}	[Idle]
[OpenSent]	{Hold timer expired}	[Idle]
[OpenSent]	{Keepalive timer expired}	[Idle]
[OpenSent]	{Receive OPEN message}	[Idle] or [OpenConfirm] (see text)
[OpenSent]	{Receive KEEPALIVE message}	[Idle]
[OpenSent]	{Receive UPDATE message}	[Idle]
[OpenSent]	{Receive NOTIFICATION message}	[Idle]

In this state, a node has sent an OPEN message to a remote node and is waiting for an OPEN message from that node.

When an OPEN message is received, the node MUST check all fields for conformance with this specification.

If the message header checking (see Section 6.2.4.1) or the OPEN message checking (see Section 6.2.4.2) detects an error or a connection collision (see Section 6.2.4.8), the node MUST:

- send a NOTIFICATION message;
- enter the [Idle] state.

If there are no errors in the OPEN message, the node MUST:

- send a KEEPALIVE message;
- set the KeepAlive timer, unless the Hold Time value is zero;
- set the Hold Timer to the negotiated Hold Time value, unless the Hold Time value is zero (see Section 6.2.2.2);

- enter the [OpenConfirm] state.

If the {DRRP TCP connection closed} event occurs, the node MUST:

- start the ConnectRetry timer;
- continue to listen for a connection from the remote node;
- enter the [Active] state.

If the Hold Timer expires, the node MUST:

- send a NOTIFICATION message with the Error Code “Hold Timer Expired”;
- enter the [Idle] state.

If the node receives a {DRRP Stop} event (initiated by either system or operator) the node MUST:

- send a NOTIFICATION message with the Error Code “Cease”;
- enter the [Idle] state;

The {DRRP Start} event MUST be ignored

In response to any other event, the node MUST:

- send a NOTIFICATION message with the Error Code “Finite State Machine Error”;
- enter the [Idle] state.

Whenever DRRP changes state from [OpenSent] to [Idle], it MUST close the TCP connection and release all resources associated with the connection.

6.2.7.5 [OpenConfirm] State

Table 6–28 - DRRP FSM Transitions from [OpenConfirm]

Initial State	Event	Final State
[OpenConfirm]	{DRRP Start}	[OpenConfirm]
[OpenConfirm]	{DRRP Stop}	[Idle]
[OpenConfirm]	{DRRP TCP connection open}	[Idle]
[OpenConfirm]	{DRRP TCP connection closed}	[Idle]
[OpenConfirm]	{DRRP TCP connection open failed}	[Idle]
[OpenConfirm]	{DRRP TCP fatal error}	[Active]
[OpenConfirm]	{ConnectRetry timer expired}	[Idle]
[OpenConfirm]	{Hold timer expired}	[Idle]
[OpenConfirm]	{Keepalive timer expired}	[OpenConfirm]
[OpenConfirm]	{Receive OPEN message}	[Idle]
[OpenConfirm]	{Receive KEEPALIVE message}	[Established]
[OpenConfirm]	{Receive UPDATE message}	[Idle]
[OpenConfirm]	{Receive NOTIFICATION message}	[Idle]

In this state, a node has sent an OPEN message to the remote node, received an OPEN message from that node, and sent a KEEPALIVE message in response to the OPEN message. The node is now waiting for a KEEPALIVE or a NOTIFICATION message in response to its OPEN message.

If the node receives a KEEPALIVE message, it MUST enter the [Established] state.

If the Hold Timer expires before a KEEPALIVE message is received, the node MUST:

- send a NOTIFICATION message with the Error Code “Hold Timer Expired”;
- enter the [Idle] state.

If the node receives a NOTIFICATION message, it MUST enter the [Idle] state.

If the Keepalive timer expires, the node MUST:

- send a KEEPALIVE message;
- restart the Keepalive timer.

If a disconnect notification is received from the underlying transport protocol (*i.e.,* TCP), the node MUST:

- tear down the TCP connection;
- restart the ConnectRetry timer;
- continue to listen for a connection from the remote node;
- enter the [Active] state.

If the node receives a {DRRP Stop} event (initiated by either system or operator) the node MUST:

- send a NOTIFICATION message with the Error Code “Cease”;
- enter the [Idle] state.

The {DRRP Start} event MUST be ignored.

In response to any other event, the node MUST:

- send a NOTIFICATION message with the Error Code “Finite State Machine Error”;
- enter the [Idle] state.

Whenever the FSM changes state from [OpenSent] to [Idle], the node MUST close the TCP connection and release all resources associated with the connection.

6.2.7.6 [Established] State

Table 6–29 - DRRP FSM Transitions from [Established]

Initial State	Event	Final State
[Established]	{DRRP Start}	[Established]
[Established]	{DRRP Stop}	[Idle]
[Established]	{DRRP TCP connection open}	[Idle]
[Established]	{DRRP TCP connection closed}	[Idle]
[Established]	{DRRP TCP connection open failed}	[Idle]
[Established]	{DRRP TCP fatal error}	[Idle]
[Established]	{ConnectRetry timer expired}	[Idle]
[Established]	{Hold timer expired}	[Idle]
[Established]	{Keepalive timer expired}	[Established]
[Established]	{Receive OPEN message}	[Idle]
[Established]	{Receive KEEPALIVE message}	[Established]
[Established]	{Receive UPDATE message}	[Idle] or [Established] (see text)
[Established]	{Receive NOTIFICATION message}	[Idle]

In this state, a node can exchange UPDATE, NOTIFICATION, and KEEPALIVE messages with the remote node.

If the negotiated Hold Timer is zero, no procedures are needed or used to keep alive a session with a remote node.

If the Hold Timer expires, the node MUST:

- send a NOTIFICATION message with the Error Code “Hold Timer Expired”;
- enter the [Idle] state.

If the Keepalive Timer expires, the node MUST:

- send a KEEPALIVE message;
- restart the Keepalive Timer.

If the Hold Timer is nonzero and the node receives an UPDATE or a KEEPALIVE message, it MUST restart its Hold Timer.

If the Hold Timer is nonzero, then every time that the node transmits an UPDATE message or a KEEPALIVE message, it MUST restart its Keepalive Timer.

If the node receives a NOTIFICATION message, it MUST enter the [Idle] state.

If the node receives an UPDATE message and the UPDATE message error handling procedure (see Section 6.2.4.3) detects an error, the node MUST:

- send a NOTIFICATION message;
- enter the [Idle] state.

If a {DRRP TCP fatal error} event occurs, the node MUST enter the [Idle] state.

If the node receives a {DRRP Stop} event (initiated by either system or operator) the node MUST:

- send a NOTIFICATION message with the Error Code “Cease”;
- enter the [Idle] state.

The {DRRP Start} event MUST be ignored.

In response to any other event, the node MUST:

- send a NOTIFICATION message with the Error Code “Finite State Machine Error”;
- enter the [Idle] state.

Whenever the FSM changes state from [Established] to [Idle], the node MUST close the TCP connection and release all resources associated with the connection.

7 RESOURCE CONFIGURATION AND ALLOCATION

7.1 RTSP protocol

RTSP [RFC 2326] is used for resource configuration and allocation. The terminology, message syntax, method definitions, and state machine definitions of RTSP are used in accordance with [RFC 2326]. Only a subset of all the required methods in [RFC 2326] is required by this specification.

In addition to the standard header definition specified in [RFC 2326], RTSP extensions specified in this document MUST be supported.

RTSP is a text and token-based protocol. The definitions in this specification MUST be considered case-sensitive unless otherwise noted. RTSP methods and headers MUST be terminated by a carriage return and a linefeed. White space MAY be inserted between tokens and between tokens and delimiters.

The M-CMTS core, the ERM, and the EQAM MUST use TCP as the transport-layer protocol. Implementations SHOULD listen on TCP port 554 for incoming RTSP connections. The TCP connection SHOULD be initiated by the RTSP client.

A persistent transport connection [RFC 2326] SHOULD be used.

An RTSP server MUST support request pipelining as specified in Section 9.1 of [RFC 2326]. A client MAY send multiple requests without waiting for each response. A server MUST send its response to those requests in the same order that the requests were received.

Figure 7–1 shows the role of M-CMTS components with regard of RTSP.

The M-CMTS core MUST be capable of acting as an RTSP client.

The ERM MUST be capable of acting as an RTSP client and as an RTSP server.

The EQAM MUST be capable of acting as an RTSP server.

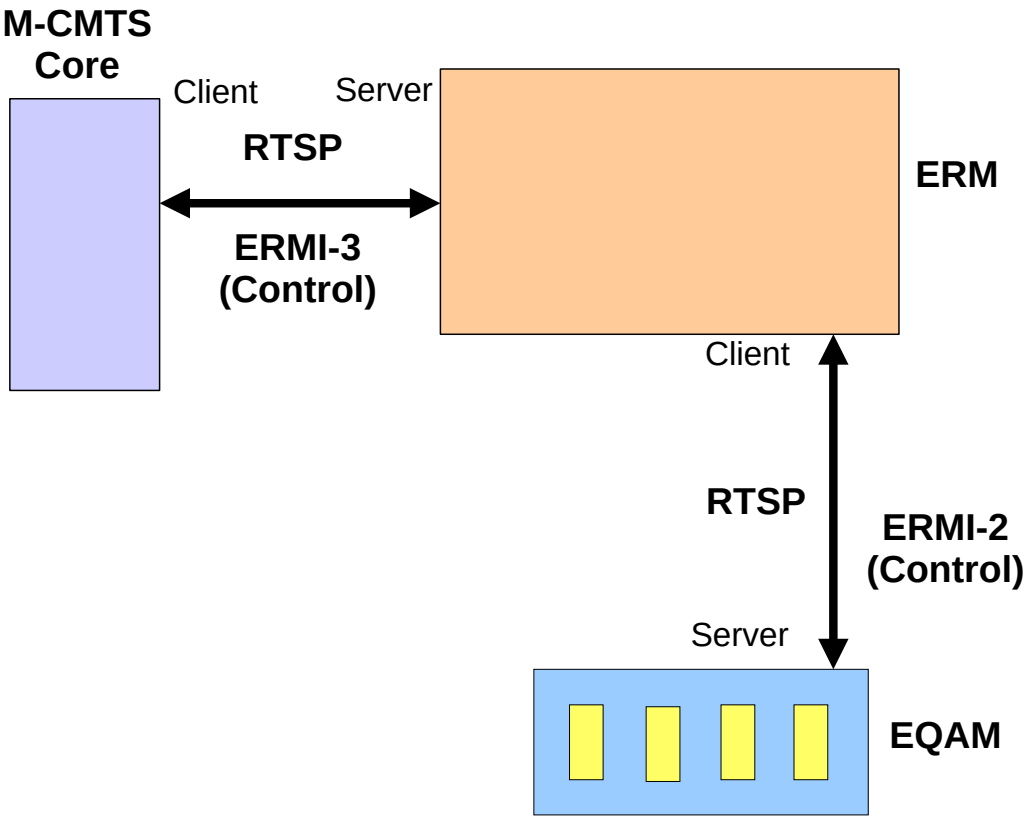

Figure 7–1 - RTSP Client - Server Connections

7.1.1 RTSP URL

Each RTSP request requires a URL. The URL MUST have the format:

rtsp://host[:port]/[abs_path]

The port field MUST be included if its value is other than 554.

For requests passed onto the resource allocation interface ERMI-3:

- the host field SHOULD contain the IP address or host name of the ERM;
- the abs_path field SHOULD NOT be present.

For requests passed onto the resource allocation interface ERMI-2:

- the host field SHOULD contain the IP address or host name of the EQAM;
- the abs_path field MUST identify the particular QAM channel that is to act on the request.

The ERM obtains the URL for each QAM channel through the registration interface (see Section 6). The URL obtained SHOULD be used by the ERM when sending an RTSP request to the EQAM over ERMI-2. For example, the ERM could use the URL "rtsp://172.10.2.3/1234", where 172.10.2.3 is the EQAM's signaling IP address and 1234 is the QAM channel TSID.

7.1.2 RTSP methods

The M-CMTS core, the ERM, and the EQAM MUST support the following RTSP methods:

- SETUP
- TEARDOWN
- PING
- GET PARAMETER

In addition, the M-CMTS core and the ERM MUST support the ANNOUNCE method.

The SETUP method is used to initiate sessions. This method includes headers to specify the resource requested for the session. A SETUP request is sent from a client to a server. If the resource allocation is successful, the server MUST send a response containing a session identifier and with Transport headers (see Section 7.1.5.1) that identify the allocated resource. RTSP uses a Session header (see section 12.37 of [RFC 2326]) to identify an RTSP session uniquely within a RTSP server. If the server fails to allocate the requested resources, the server MUST send a SETUP response with an appropriate error code.

The TEARDOWN method is used to request session teardown. The session identifier is given in the TEARDOWN request. The TEARDOWN request is sent from a client to a server. Upon receipt of a TEARDOWN request, the server MUST release all the resources related with the identified RTSP session and send an appropriate response to the client.

The GET_PARAMETER method is used by a RTSP client to get a parameter value from a RTSP server. Upon receipt of a GET_PARAMETER method, the server MUST find the parameter value and send the parameter value in GET_PARAMETER response. This specification extended the GET_PARAMETER method by defining a new parameter called session_list. The parameter session_list is specified in Section 7.1.5.4.

ANNOUNCE is an RTSP method that is used by a server to transfer information about the status of a session to a client in real time. An ANNOUNCE request is sent from server to client. An ANNOUNCE request MUST include a Notice header.

PING is a new RTSP method defined in this specification. A PING request is sent from client to server, and is used as a session keep-alive mechanism. The issuer of the request MUST include a Session header with the session ID of the session that is being kept alive. In addition, a CSeq header and a Require header MUST be included in a PING request. A PING response MUST carry only two headers: Session and CSeq. The Session header and the CSeq header are standard RTSP headers defined in [RFC 2326]. Required header usage is specified further in Section 7.1.4 of this specification. An example of a PING request and the corresponding response is given below.

A session keepalive PING request from the RTSP client to the server: [7]

```
PING rtsp://172.22.10.3 RTSP/1.0
CSeq: 123
Session: 1234567
Require: ermi
```

A corresponding response from the RTSP server to the client:

```
RTSP/1.0 200 OK
CSeq: 123
Session: 1234567
```

[7] Editorial changes to the format of this section per ECN ERMI-N-05.0244-1 on 11/1/05.

7.1.3 RTSP State Machine

Since support for only a limited set of methods is required by this specification, the client and server state machines are simple.

This section specifies the required RTSP operation in terms of a Finite State Machine (FSM). Implementations of clients and servers that conform to this specification MUST operate in accordance with the FSM described in this section. A client or server MUST implement an independent FSM for every RTSP session.

7.1.3.1 RTSP Server State Machine

The server FSM MUST contain two states:

Table 7–1 - RTSP Server FSM States

State Name	Brief State Description
[Idle]	Initial state for an RTSP session
[Ready]	The RTSP session is active

The server FSM MUST contain 3 events:

Table 7–2 - RTSP Server FSM Events

Event Name	Brief Event Description
{SETUP}	SETUP request received and processed successfully
{TEARDOWN}	TEARDOWN request received and processed successfully
{Session Timeout}	Session has timed out (see Section 7.1.6)

State transitions in the FSM MUST conform to Table 7–3.

The FSM MUST begin in the [Init] state.

Table 7–3 - RTSP Server State Machine

Initial State	Event	Final State
[Init]	{SETUP}	[Ready]
[Ready]	{TEARDOWN}	[Init]
[Ready]	{Session Timeout}	[Init] or [Ready] (see text)

When the server is in the [Init] state, if a SETUP request received by the server is processed successfully, the server MUST enter the [Ready] state; otherwise, the server MUST remain in the [Init] state.

When the server is in the [Ready] state, if a TEARDOWN request received by the server is processed successfully, the server MUST enter the [Init] state.

When server is in the [Ready] state, it MAY enter the [Init] state after detecting a session timeout.

7.1.3.2 RTSP Client State Machine

The client FSM MUST contain two states:

Table 7–4 - RTSP Client FSM States

State Name	Brief State Description
[Idle]	Initial state for an RTSP session
[Ready]	RTSP session is active

The client FSM MUST contain 2 events:

Table 7–5 - RTSP Client FSM Events

Event Name	Brief Event Description
{SETUP}	SETUP request sent and successful response (response code is 200) received
{TEARDOWN}	TEARDOWN request sent and successful response (response code is 200) received (or response timeout)

State transitions in the FSM MUST conform to Table 7–6.

The FSM MUST begin in the [init] state.

Table 7–6 - RTSP Client State Machine

Initial State	Event	Final State
[Init]	{SETUP}	[Ready]
[Ready]	{TEARDOWN}	[Init]

When the client is in the [Init] state, if instructed to establish a session, it will send out SETUP request with a Transport header that describes the resource requested. After sending a SETUP request and receiving a successful SETUP response with response code of 200, the client MUST enter the [Ready] state. Otherwise, it MUST remain in the [Init] state.

When the client is in the [Ready] state, after sending a TEARDOWN request and receiving a successful TEARDOWN response with response code of 200, the client MUST enter the [Init] state. If the response times out, the client SHOULD enter the [Init] state. Otherwise, it SHOULD remain in the [Ready] state.

7.1.4 RTSP headers

RTSP headers follow the RTSP base syntax specified in section 15.1 of RTSP [RFC 2326]. In RTSP, each header concludes with a carriage return and a line feed. The headers are separated from a (possibly empty) body with a carriage return and a linefeed.

The headers in Table 7–7 MUST be supported.

Table 7–7 - Supported RTSP Headers

Header	Direction	RTSP Method
CSeq	Request & Response	PING, SETUP, TEARDOWN, ANNOUNCE, GET_PARAMETER
Session	Request & Response	PING, SETUP(response), TEARDOWN, ANNOUNCE
Require	Request	PING, SETUP, TEARDOWN, ANNOUNCE, GET_PARAMETER
Transport	Request & Response	SETUP
DOCSIS-Notice	Request	ANNOUNCE
Content-Type	Request & Response	GET_PARAMETER
Content-Length	Request & Response	GET_PARAMETER

The value of the CSeq header is an RTSP request sequence number. Use of this header MUST conform to the requirements in section 12.17 of [RFC 2326].

The value of the Session header is an RTSP session identifier. Use of this header MUST conform to the requirements in section 12.37 of [RFC 2326].

The Require header is used to ensure that all the options are understood by both client and server. The Require header includes an option tag that represents the feature set supported. In ERMI, a new option tag (see section 7.1.5.3) is used to represent the requirements specified in this document. Use of this header MUST conform to the requirements in section 12.32 of [RTSP] a new "ermi" option tag defined in Section 7.1.5.3.

The Transport header format of the new transport type, DOCSIS/QAM, is specified in Section 7.1.5.1.

The DOCSIS-Notice header is further discussed in section 7.1.5.2. It identifies information sent from an RTSP server to a client in an ANNOUNCE request.

The Content-type header is defined in section 12.16 of [RFC 2326]. It MUST be included in GET_PARAMETER request and response. The format of Content-type MUST be:

Content-type: <content-type>
where <content-type> MUST be text/parameters

The Content-length header is defined in section 12.14 of [RFC 2326]. It MUST be included in GET_PARAMETER request or response. The Content-length header indicates the size of the message-body not including any headers, in decimal number of octets, sent to the recipient. The format of Content-length MUST be:

Content-length: <digits>
where <digits> is the ASCII representation of decimal number.

7.1.5 RTSP extensions

This section specifies the extensions to [RFC 2326] needed to support DOCSIS applications.

7.1.5.1 Transport Headers

Section 12.39 of [RFC 2326] specifies the Transport header. The Transport header is used to identify parameters associated with the transport of the media stream. The new transport type defined below MUST be supported by the M-CMTS core, the ERM, and the EQAM.

The Transport header is used in RTSP SETUP request and SETUP response messages. It indicates the type of transport being requested or granted for the session. This specification defines the transport specifier "DOCSIS/QAM", which is used to pass DOCSIS QAM channel transport parameters.

The DOCSIS/QAM Transport header MUST be included in the SETUP request from client to server. The values in a DOCSIS/QAM Transport header in a SETUP request indicate the desired resource parameters. The values in a DOCSIS/QAM Transport header in a SETUP response indicate the parameters of the selected resource.

In ERMI-2, the server (*i.e.,* the EQAM) MAY include a Transport header in the SETUP response only if the response code is 200. In ERMI-3, the server (*i.e.,* the ERM) MUST include a Transport header in the SETUP response if the response code is 200. The DOCSIS/QAM Transport header MUST NOT be included in the SETUP response if the response code is not 200.

Transport headers take the form (square brackets indicate optional fields):

Transport: <channel> [, <channel>]

where there is no limit to the number of comma-separated <channel> fields. The order in which multiple <channel> fields appear expresses a preference (most preferred first, least preferred last) for the order in which the requesting device desires QAM channels to be tested for availability and suitability. Parameter fields in the <channel> are separated by semicolons, with no space in between.

Each <channel> field takes the form:

DOCSIS/QAM; unicast

bit_rate=<bit_rate>

qam_id=<qam_id>

depi_mode=<depi_mode>

[;source=<source>]

[;server_port=<server-port>]

[;destination=<destination>]

[;modulation=<modulation>]

[;j83_annex=<j83_annex>]

[;taps=<taps>;increment=<increment>]

[;channel_width=<channel_width>]

Where:

<bit_rate> is the data rate in bits per second;

<qam_id> is the ASCII encoding of a QAM TSID that identifies a QAM channel in the head-end;

<depi_mode> MUST be one of the following modes, which are defined in [DEPI]:

- docsis_mpt
- docsis_psp

<source> is the IP address from which the data is streamed. If present, the value MUST be the IP address of the M-CMTS core.

<server-port> is the UDP port from which the media data will be streamed. If present, the value MUST be the UDP port on the M-CMTS core from which data is sent to the QAM channel.

<destination> is the IP address of the QAM channel.

<modulation> is the QAM modulation type. If this field is present, its value MUST be one of:

- 64
- 256

<j83_annex> is the desired QAM operation mode as defined in the Annex section of [J.83]. If this field is present, its value MUST be one of:

- Annex_A
- Annex_B
- Annex_C

<taps> and <increment> are interleaver parameters. The interleaver is defined in [J.83]. If the <taps> and <increment> fields are used, their values MUST be taken from Table 7–8.

Table 7–8 - Taps and Increments Supported

Taps	Increment
8	16
16	8
32	4
64	2
128	1
128	2
128	3
128	4
128	5
128	6
128	7
128	8

<channel_width> is the bandwidth of the QAM channel, in MHz. If this field is present, its value MUST be the entire bandwidth of the QAM channel and be one of:

- 6
- 8

The destination, modulation, j83_annex, taps, increment and channel_width fields in the Transport header are optional. If the client is ambivalent to the value of a particular parameter, then it SHOULD NOT send the corresponding optional field in a SETUP request. This increases the server's flexibility in choosing a QAM channel resource. If the client requires a particular value for an optional field, it MUST include the desired value for that optional field in the SETUP request.

If the Transport header is included in a SETUP response from server to client, it MUST include the following optional fields in the Transport header, if the SETUP response indicates success:

- destination
- modulation
- j83_annex
- taps
- increment
- channel_width

For each optional field, if that optional field was present in the request, the same field and value MUST be present in the response. If that optional field was not present in the request, the field MUST be present in the response with a value that reflects the current configuration of the QAM channel.

Two examples of valid DOCSIS/QAM transport headers are:

- Transport: DOCSIS/QAM; unicast; bit_rate=38800000; qam_id=1234; depi_mode=docsis_mpt
- Transport: DOCSIS/QAM; unicast; bit_rate=38800000; qam_id=32; depi_mode=docsis_psp; destination=172.22.25.1; modulation=256; j83_annex=Annex_B; taps=128; increment=1; channel_width=6

7.1.5.2 DOCSIS-Notice Header Extension

The DOCSIS-Notice header provides information sent from an RTSP server to a client in an ANNOUNCE request. The syntax of the DOCSIS-Notice header is:

DOCSIS-Notice: <notice-code>

The RTSP client and server MUST support the values of notice-code and code-description given in Table 7–9:

Table 7–9 - RTSP notice code

notice-code	Meaning
5701	Reclaim Session

When an ERM requests an M-CMTS core to release a QAM channel resource previously allocated by the M-CMTS core, the ERM MUST send an RTSP ANNOUNCE request to the M-CMTS core with a notice-code of 5701.

After the M-CMTS receives the ANNOUNCE request with a notice-code of 5701, if the M-CMTS core intends to release the QAM channel resource by sending a TEARDOWN request to the ERM, it SHOULD send an ANNOUNCE response with a response code of 200 OK. If the M-CMTS core does not intend to release the QAM channel resource at the current time, it SHOULD send the ANNOUNCE response with a response code of 503 Service Unavailable.

Other notice codes MUST NOT be used in an ANNOUNCE request.

7.1.5.3 Require header extension

An RTSP request MUST include a Require header with the option tag value “ermi”.

7.1.5.4 GET_PARAMETER method extension

After establishing a TCP connection to a RTSP server, a RTSP client SHOULD send GET_PARAMETER request to obtain a list of active RTSP sessions on the server that were initiated by the client before this TCP connection was established. The RTSP client SHOULD use this method to synchronize its session state with the RTSP server following a client reboot.

On receiving a GET_PARAMETER request for session list parameter, an RTSP server MUST identify the client using the client's IP address and retrieve all the active RTSP sessions IDs associated with this client. The server MUST send a GET_PARAMETER response with a list of these session IDs.

To request a session list, the content of the message body MUST be: session_list

The body of the response MUST have the format: session_list: <session_id>

where <session_id> is the RTSP session ID as defined in section 3.4 of [RFC 2326]. Multiple <session_id>s, MUST be separated by whitespace.

7.1.6 Keepalives and timeout

A PING request SHOULD be sent periodically from the client to the server. The session timeout value SHOULD be sent from server to client as specified as in [RFC 2326], section 12.37. The default session timeout MUST be 3 hours. The PING request interval MUST be less than the selected session timeout.

If the server does not receive any RTSP request in a period equal to the session timeout , the server MAY tear down the RTSP session and release the resources associated with the RTSP session. For ERMI-2, the QAM channel resource MUST also be released if the EQAM tears down the RTSP session. The EQAM SHOULD NOT tear down the RTSP session and release the resource if it knows that the data path corresponding to this RTSP session is still receiving data traffic.

The recipient of an RTSP request MUST transmit a response. The sender of an RTSP request MUST start an RTSP response timer immediately after sending the request. If the RTSP response timer expires before reception of the response, the sender SHOULD consider the request a failure (*i.e.,* it should act as if it had received an error response). The response timer SHOULD be set to 10 seconds. If a TEARDOWN request fails to receive a timely response, the client SHOULD release any resources associated with the session.

7.1.7 RTSP Response Code

RTSP clients MUST accept the response codes defined in Table 7–10. RTSP servers SHOULD use only the response codes defined in Table 7–10. The response code appears in the first line of RTSP response message as defined in section 7.1 of [RFC 2326]. The client is not required to examine the message text that follows the response code.

Table 7–10 -Supported RTSP Response Codes

Response Code	Message Text	Comment
200	OK	
404	Not Found	QAM not found
405	Method Not Allowed	
408	Request Timeout	
451	Invalid Parameter	
453	Not Enough Bandwidth	Insufficient QAM channel bandwidth
454	Session Not Found	
457	Invalid Range	
461	Unsupported Transport	
501	Not Implemented	
503	Service Unavailable	

Response Code	Message Text	Comment
505	RTSP Version Not Supported	
551	Option Not Supported	

7.1.8 Resource allocation operation

7.1.8.1 Resource Allocation

When the M-CMTS core needs to add a QAM channel resource to a DOCSIS MAC domain, it initiates an RTSP session with a SETUP request in order to obtain downstream QAM resources for the domain. Similarly, when the M-CMTS core wishes to remove a QAM channel resource from a DOCSIS MAC domain, it concludes the RTSP session by sending a TEARDOWN request to release that downstream QAM channel resource.

When setting up a session, the sequence of messages shown in Figure 7–2 is followed.

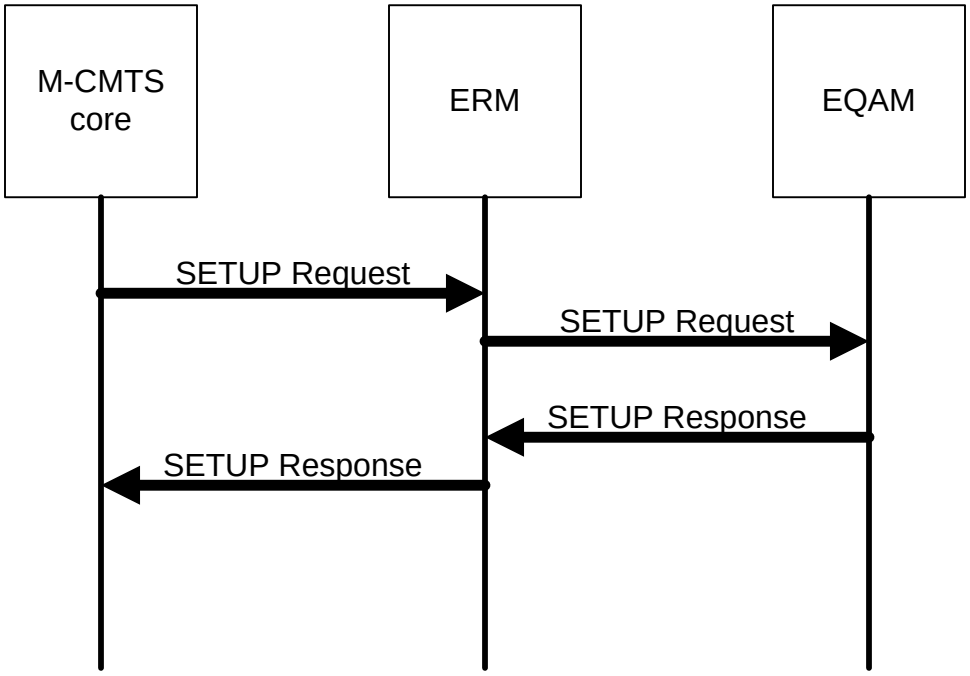

Figure 7–2 - RTSP SETUP Message Flow[8]

The M-CMTS core MUST include the following information in its SETUP request:

- A list of QAM IDs;
- QAM channel bit rate;
- QAM channel capability.

Each QAM channel MUST be represented by a single channel field in RTSP Transport header. The ERM MUST select a QAM channel from the list presented to it in the Transport header of the SETUP request.

The capabilities of the QAM channel are described in the Transport header. The ERM MUST NOT select a QAM channel that does not support all the listed capabilities.

[8] Replaced this figure per ECN ERMI-N-05.0244-1 on 11/1/05.

The DOCSIS capabilities listed in different Transport headers of a single RTSP request MUST be identical.

If the ERM cannot locate a QAM channel that satisfies all the capabilities listed in the Transport header, it MUST respond to the request with a SETUP response with an appropriate error code as defined in Section 7.1.7.

After a downstream QAM channel has been selected by the ERM, the ERM MUST send a SETUP request to the EQAM that contains the selected QAM channel. This request MUST contain at least the following information:

- QAM ID;
- QAM channel bit rate;
- QAM channel capability.

After the EQAM receives this RTSP SETUP request, it MUST verify that the downstream QAM channel resource is available. If the resource is available, it MUST send an RTSP SETUP response to the ERM with success status. Otherwise, it MUST return a SETUP response indicating failure. The EQAM MUST also respond with a failure notification if the requested QAM channel parameters indicated in Transport header cannot be supported without disrupting the operation of any other QAM channels in the EQAM. The EQAM MAY configure the QAM channel using the values indicated in the Transport header before it returns the SETUP response to the ERM. The EQAM MAY wait for DEPI operation to perform reconfiguration of QAM channel parameters. It MUST reconfigure the QAM channel (if necessary) before the QAM channel is used.

If a list of QAM IDs is given to the ERM in a SETUP request, the ERM MUST send a SETUP response with an appropriate response code as specified in Table 7–10. The ERM MUST try every QAM ID in the received list, in order, until it satisfies the request or it has tried every QAM ID in the list. If no QAM IDs are available to satisfy the request, the ERM MUST send response code 453 in the SETUP response.

If the ERM receives a SETUP response with a response code of 200 from the EQAM, it SHOULD send a SETUP response with a response code of 200 to the M-CMTS core. If the SETUP response from the ERM indicates success, the SETUP response to the M-CMTS core MUST include a Transport header with at least the following information:

- QAM ID;
- QAM channel IP address;
- QAM channel capabilities.

The M-CMTS core initiates a data session to the QAM channel, following the procedure defined in [DEPI].

The ERM SHOULD NOT send the SETUP response to the M-CMTS core before it receives a SETUP response from the EQAM, unless a timeout occurs or the ERM, after consulting its database of resources, cannot locate a suitable QAM channel.

7.1.8.2 Resource De-allocation

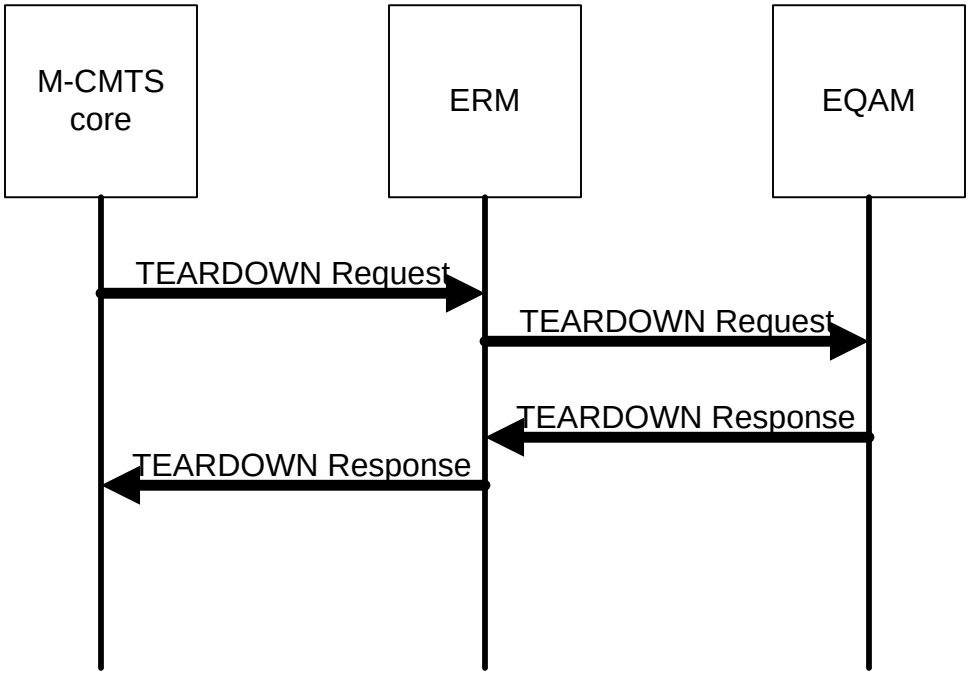

Figure 7–3 - RTSP TEARDOWN Message Flow[9]

If an MCMTS core removes a QAM channel resource from a MAC domain, it MUST send a TEARDOWN request to the ERM. This message MUST include the corresponding RTSP session ID obtained from the original SETUP response.

Upon receipt of a TEARDOWN request, the ERM MUST send a TEARDOWN request to the EQAM. The ERM SHOULD NOT send the TEARDOWN response to the M-CMTS core before it receives a TEARDOWN response from EQAM device, unless a timeout occurs.

If a QAM channel is re-allocated (*e.g.,* between applications that carry video and DOCSIS traffic), some of the channel configuration parameters may change as the channel switches between the two kinds of traffic. QAM channels SHOULD be configured by default so that their parameters match the requirements of video traffic because some video applications simply assume the default configuration is in effect. When a QAM channel is used for DOCSIS traffic, some of these settings could be changed over the DEPI interface (see [DEPI] for details). Therefore, in order to make this QAM channel available for VOD use after DOCSIS use, the QAM channel parameters MUST be restored by the EQAM to their prior values before an EQAM sends a TEARDOWN response to ERM to release a QAM channel resource. An EQAM MUST restore the QAM channel configuration only if the session being torn down is the only RTSP session in this QAM channel and no QAM channel dependencies exist that might affect other QAM channels that are in use.

7.1.9 Multiple QAM channels in MAC Domain

In a single DOCSIS MAC domain, there may be multiple downstream QAM channels. To support multiple QAM channels in a single MAC domain, the M-CMTS core MUST send resource requests for different QAM channels in the MAC domain to the ERM separately, one request for each QAM channel resource.

[9] Replaced this figure per ECN ERMI-N-05.0244-1 on 11/1/05.

7.1.10 Synchronization with DEPI control [DEPI]

[DEPI] defines a control interface between the M-CMTS core and the EQAM. This interface is used to negotiate session parameters for DOCSIS traffic. For example, the UDP port on which the EQAM accepts DOCSIS data for the session is negotiated over this interface.

If an M-CMTS application is not statically assigned QAM channel resources and must obtain them through the ERM via RTSP, then the M-CMTS core MUST NOT initiate a DEPI session prior to obtaining QAM channel resources for that session. Similarly, the M-CMTS core MUST NOT release QAM channel resources before tearing down the corresponding DEPI session. If DEPI session setup fails, the M-CMTS core SHOULD tear down the RTSP session to release the QAM channel resources.

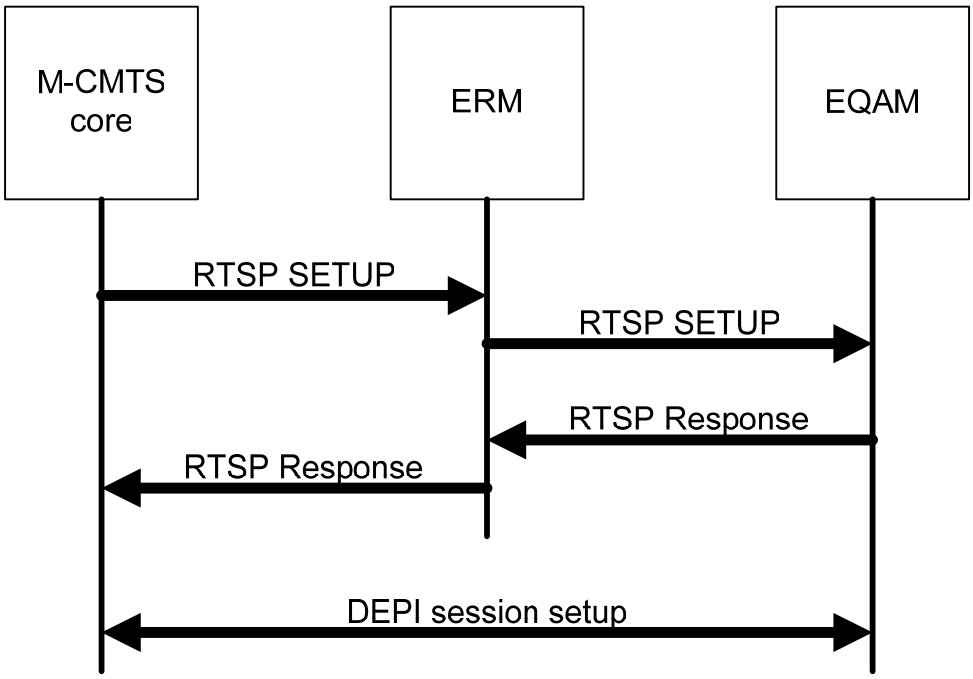

Figure 7–4 - Session Setup Sequence with DEPI[10]

[10] Replaced this figure per ECN ERMI-N-05.0244-1 on 11/1/05.

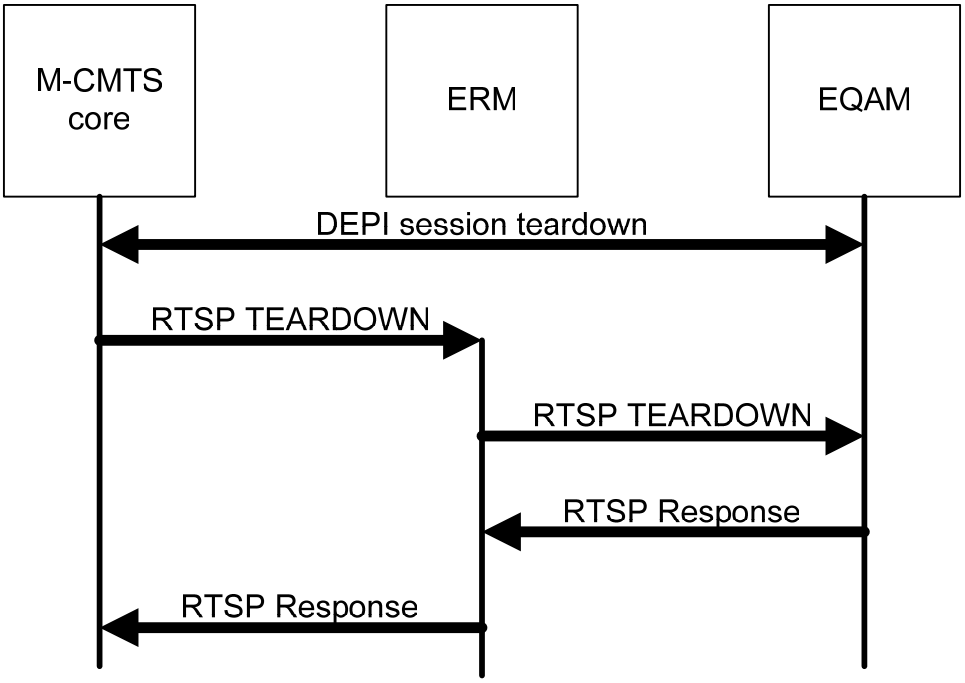

Figure 7–5 - Session Teardown Sequence with DEPI[11]

[DEPI] allows an M-CMTS core to change QAM channel physical settings directly. The EQAM checks to make sure that changing the setting of one QAM channel does not affect the operation of any other QAM channels in service. Because QAM channels are physically connected to RF ports, there may be limitations on how physical parameters can be changed.

For example, an RF port may require that all the associated QAM channels have the same modulation type. Suppose there are four QAM channels in the RF port and one QAM channel is already in service with a modulation type of QAM64. A new request (for a different QAM channel in this port) with modulation type QAM256 should be rejected. Otherwise, the operation of the existing QAM would be affected.

An M-CMTS core MUST use separate RTSP requests to obtain QAM channel resource for each DEPI data session.

[11] Replaced this figure per ECN ERMI-N-05.0244-1 on 11/1/05.

Appendix I Example DRRP Message Exchanges

This section provides an example of a DRRP message exchange between an EQAM and an ERM.

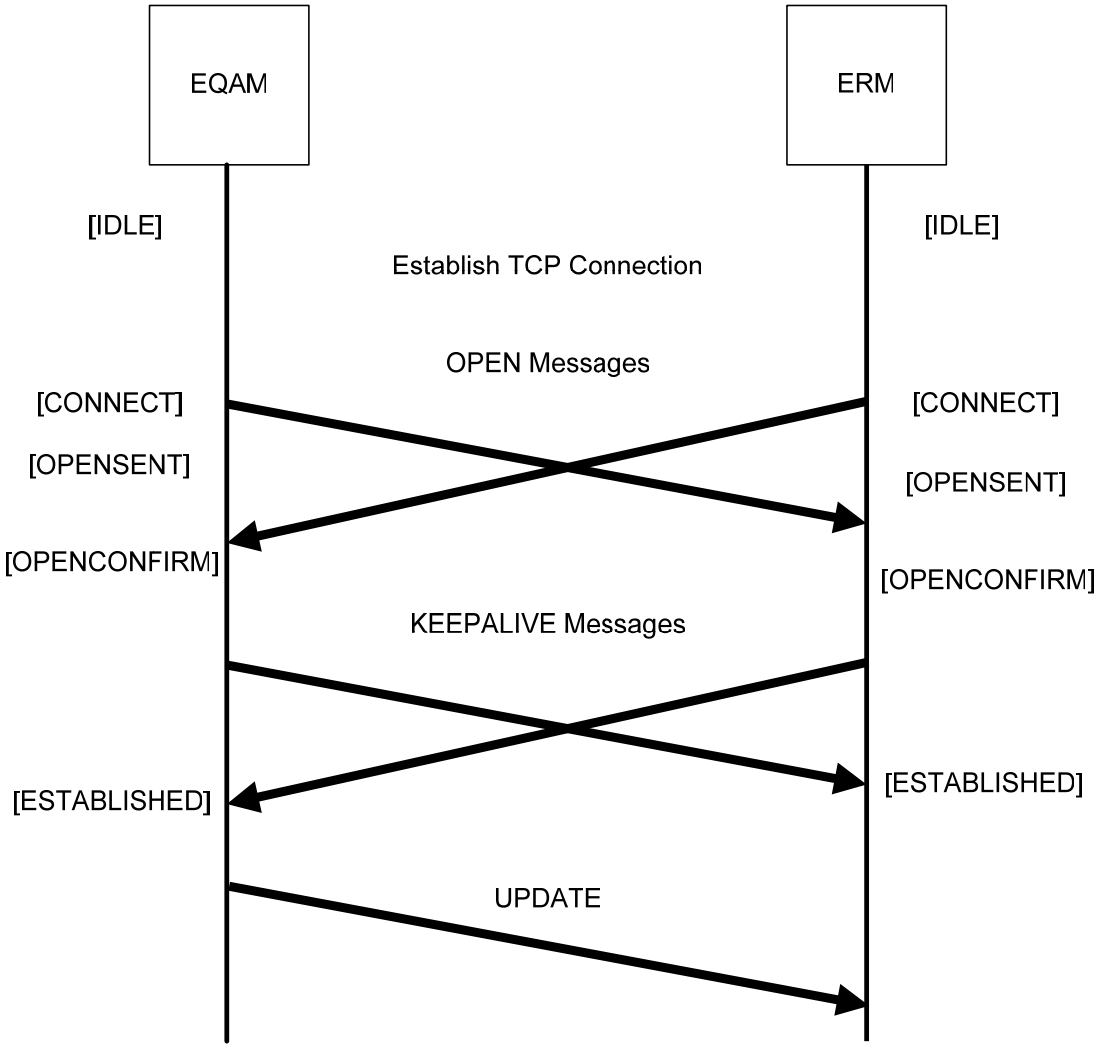

Figure I–1 - Example DRRP Connection Establishment[12]

[12] Replaced this figure per ECN ERMI-N-05.0244-1 on 11/1/05.

I.1 OPEN message

The first DRRP messages sent between the EQAM and the ERM are a pair of OPEN messages.

An example message from the EQAM to the ERM appears like this matrix:

	Field	Field Length	Field Value	Comment
Header	Length	2	45	Total length of this message
	Type	1	1	OPEN
OPEN	Version	1	1	
	Reserved	1	0	
	Hold Time	2	0	No Hold Time
	Address Domain	4	1	
	DRRP Identifier	4	1234	
	Optional Parameters Len	2	28	Length of optional parameters
	Parameter Type	2	1	Capability Information
	Parameter Length	2	24	Length of the parameter value
	Capability Code	2	1	Route Types Supported
	Capability Length	2	4	Length of capability
	Address Family	2	32769	Indicates that the address is an RTSP URL
	Application Protocol	2	32768	Indicates that the application protocol is RTSP
	Capability Code	2	2	Send Receive Capability
	Capability Length	2	4	Length of capability
	Send Receive Capability	4	2	Send Only mode
	Capability Code	2	32768	DRRP version

	Field	Field Length	Field Value	Comment
	Capability Length	2	4	Length of capability
	DRRP version	4	1	

When each device receives the OPEN, it responds with a KEEPALIVE.

I.2 KEEPALIVE message

The KEEPALIVE message comprises simply a DRRP header.

Field	Field Length	Field Value	Comment
Length	2	3	Total length of this message
Type	1	4	KEEPALIVE

I.3 UPDATE message

After the DRRP connection has been established, the EQAM reports its available resources to the ERM. To advertise available resources, the EQAM sends a separate UPDATE message for each QAM channel. After the initial UPDATE for a QAM channel, the EQAM sends further UPDATEs for that QAM channel only when the service status or configuration of that QAM channel changes.

In an UPDATE message, either the ReachableRoutes or the WithdrawnRoutes attribute is present. The ReachableRoutes attribute is used to advertise an operational QAM channel; the WithdrawnRoutes attribute is used to remove a QAM channel from service.

An example UPDATE message looks like this:

	Field	Field Length	Field Value	Comment
Header	Length	2	142	Total length of this message
	Type	1	2	UPDATE
UPDATE	Attr Flags	1	0x00	Well-known
	Attr Type Code	1	2	ReachableRoutes
	Attr Length	2	29	Length of the attribute
	Address Family	2	32769	RTSP URL
	Application Protocol	2	32768	ERMI

	Field	Field Length	Field Value	Comment
	Length	2	23	Length of address
	Address	23	rtsp://192.168.0.2/1234	URL of the QAM channel
	Attr Flags	1	0x00	Well-known
	Attr Type Code	1	3	NextHopServer
	Attr Length	2	17	Length of the attribute
	Next Hop Address Domain	4	1	
	Length	2	11	Length of address
	Server	11	192.168.0.2	IPv4 address
	Attr Flags	1	0x00	Well-known
	Attr Type Code	1	234	Bandwidth
	Attr Length	2	8	Length of the attribute
	Total Bandwidth	8	388000000	Bandwidth
	Attr Flags	1	0x00	Well-known
	Attr Type Code	1	240	Port ID
	Attr Length	2	4	Length of the attribute
	Port ID	2	0x1234	RF port ID = 0x1234
	Reserved	1	0	
	Channel ID	1	0	Channel ID = 0
	Attr Flags	1	0x00	Well-known
	Attr Type Code	1	241	Service status
	Attr Length	2	4	Length of the attribute
	Service Status	4	1	Operational
	Attr Flags	1	0x00	Well-known
	Attr Type Code	1	246	QAM ID

	Field	Field Length	Field Value	Comment
	Attr Length	2	2	Length of the attribute
	TSID	2	23	TSID = 23
	Attr Flags	1	0x00	Well-known
	Attr Type Code	1	247	DOCSIS Capability
	Attr Length	2	14	Length of the attribute
	Channel BW	2	0x0080	Lock=0, Group ID = 0, 6MHz only
	J83	2	0x00C0	Lock=0, Group ID = 0, Annex B only
	Interleaver	4	0x00F00000	Lock=0, Group ID=0, I=8, J=16, I=16, J=8, I=32, J=4, I=64, J=2
	DOCSIS mode	4	0x00406000	Lock=0, Group ID=0, Mixed DOCSIS mode=1, DOCSIS MPT, DOCSIS PSP
	Modulation	2	0x00C0	Lock=0, Group ID=0, 64QAM, 256QAM
	Attr Flags	1	0x00	Well-known
	Attr Type Code	1	248	QAM channel configuration
	Attr Length	2	8	Length of the attribute
	Frequency	4	550000000	550 MHz
	Modulation	1	1	256QAM
	Annex	4 bits	1	Annex B

	Field	Field Length	Field Value	Comment
	Ch	4 bits	0	6 MHz
	I	1	32	Interleaver
	J	1	4	Interleaver
	Attr Flags	1	0x00	Well-known
	Attr Type Code	1	249	DEPI Control IP Address
	Attr Length	2	17	Length of the attribute
	Network prefix	4	24	
	Length	2	11	Length of address
	Host	11	192.168.0.2	

I.4 NOTIFICATION message

The EQAM should send a NOTIFICATION to the ERM when it shuts down.

An example NOTIFICATION message looks like this:

	Field	Field Length	Field Value	Comment
Header	Length	2	5	Total length of this message
	Type	1	3	NOTIFICATION
NOTIFICATION	Error Code	1	6	Cease
	Error Subcode	1	0	

Appendix II Examples - RTSP

II.1 Session Setup

Suppose that an M-CMTS core wishes to create a MAC domain. It sends a SETUP request to request a QAM channel with a bit rate of 38 Mbps. The QAM channel is to be chosen from a service group, which is identified by a list of QAM channels. Suppose that the service group contains two QAM channels, with QAM IDs of 123 and 456. The DEPI mode for the session (see [DEPI]) is to be DOCSIS-MPT. The ERM's signaling IP address is 192.26.2.2. Then the SETUP request from the M-CMTS core to the ERM looks like this:

Line	Comments
SETUP rtsp://192.26.2.2 RTSP/1.0	Identifies this as an RTSP SETUP request
CSeq: 314	Sequence number
Require: ermi	The recipient must support this specification
Transport: DOCSIS/QAM; unicast; bit_rate=38000000; qam_id=456; depi_mode=docsis_mpt, DOCSIS/QAM; unicast; bit_rate=38000000; qam_id=123; depi_mode=docsis_mpt	The two channels in the service group
	Blank Line

The ERM searches its resource database, in the order in which the channels are presented in the received Transport header, and discovers that the QAM channel with QAM ID 456 is already in use but the channel with QAM ID 123 is available. The ERM therefore tentatively selects the latter QAM channel to satisfy the request, and sends a request to the EQAM for this resource.

In this request, the ERM specifies a bit rate of 38 Mbps. The ERM knows the EQAM's signaling IP address (172.2.2.2) through the service registration interface (ERMI-1).

Line	Comments
SETUP rtsp://172.2.2.2/123 RTSP/1.0	Identifies this as an RTSP SETUP request
CSeq: 101	Sequence number
Require: ermi	The recipient must support this specification
Transport: DOCSIS/QAM; unicast; bit_rate=38000000; qam_id=123; depi_mode=docsis_mpt	The resource being requested
	Blank line

The EQAM responds with a message to indicate that the QAM channel is indeed available. The session ID 47112344 was assigned to this RTSP session.

Line	Comments
RTSP/1.0 200 OK	Successful response to a request
CSeq: 101	Sequence number from the request
Session: 47112344	Session ID
	Blank line

The ERM now sends the SETUP response to the M-CMTS core, indicating that one of the requested resources is available for use. It fills in values for the Transport header from the information in its database that was passed to it from the EQAM at the time that the resource was registered.

Line	Comments
RTSP/1.0 200 OK	Successful response to a request
CSeq: 314	Sequence number from the request
Session: 47223344	Session ID
Transport: DOCSIS/QAM; unicast; bit_rate=3800000; qam_id=123; depi_mode=docsis_mpt; destination=172.2.2.2; modulation=256;taps=16; increment=8; J_83=Annex_B; channel_width=6	List of the parameters for this QAM channel
	Blank line

II.2 Session Teardown

When the M-CMTS deletes the DOCSIS MAC domain, it sends a TEARDOWN request to the ERM to release the QAM channel. The TEARDOWN message contains the RTSP session ID from the SETUP response:

Line	Comments
TEARDOWN rtsp://192.26.2.2 RTSP/1.0	Identifies this as an RTSP TEARDOWN request
CSeq: 316	Sequence number
Require: ermi	The recipient must support this specification
Session: 47223344	Session ID
	Blank line

The ERM now sends a corresponding TEARDOWN request to the EQAM:

Line	Comments
TEARDOWN rtsp://172.2.2.2/1234 RTSP/1.0	Identifies this as an RTSP TEARDOWN request
CSeq: 102	Sequence number
Require: ermi	The recipient must support this specification
Session: 47112344	Session ID
	Blank line

The EQAM release the QAM channel and sends a response to confirm its release:

Line	Comments
RTSP/1.0 200 OK	Successful response to a request
CSeq: 102	Sequence number from the request
Session: 47112344	Session ID from the request
	Blank line

The ERM sends a corresponding response to the M-CMTS core:

Line	Comments
RTSP/1.0 200 OK	Successful response to a request
CSeq: 316	Sequence number from the request
Session: 47223344	Session ID from the request
	Blank line

II.3 Session Keepalive

Session keepalive PING requests from the RTSP client to the RTSP server look like this:

Line	Comments
PING rtsp://172.22.10.3 RTSP/1.0	Identifies this as an RTSP PING request
CSeq: 123	Sequence number
Require: ermi	The recipient must support this specification
Session: 1234567	Session ID
	Blank line

PING responses from the RTSP server to the RTSP client look like this:

Line	Comments
RTSP/1.0 200 OK	Successful response to a request
CSeq: 123	Sequence number from the request
Session: 1234567	Session ID from the request
	Blank line

II.4 Get Parameter

The following is an example of GET_PARAMETER request from a RTSP client to a RTSP server to get a list of current active session ids.

Line	Comments
GET_PARAMETER rtsp://172.2.2.2/123 RTSP/1.0	Identifies this as an RTSP GET_PARAMETER request
CSeq: 130	Sequence number
Require: ermi	The recipient must support this specification
Content-length: 14	Length of content excluding headers
Content-type: text/parameters	Set content type
	Blank line
session_list	

The RTSP response from the server to the clients looks like this:

Line	Comments
RTSP/1.0 200 OK	Successful response to a request
CSeq: 130	Sequence number from the request
Content-length: 31	Length of content excluding headers
Content-type: text/parameters	Set content type
	Blank line
session_list: 1234567 1234568	List of session IDs for sessions between this client and this server

II.5 Session Announce

The following is an example ANNOUNCE request sent from the ERM to an M-CMTS core to ask the M-CMTS core to release the RTSP session:

Line	Comments
ANNOUNCE rtsp://172.2.2.2/123 RTSP/1.0	Identifies this as an RTSP ANNOUNCE request
CSeq: 130	Sequence number
Session: 1234567	Session ID
Require: ermi	The recipient must support this specification
DOCSIS-Notice: 5701	The server requests the client to teardown the session
	Blank line

If the M-CMTS core agrees to tear down the session, the response looks like this:

Line	Comments
RTSP/1.0 200 OK	Successful response to a request
CSeq: 130	Sequence number from the request
Session: 1234567	Session ID from the request
	Blank line

This is then followed by a TEARDOWN message from the M-CMTS core to the ERM (see section <Session Teardown>).

If, however, the M-CMTS core does not agree to tear down the session following receipt of the first message in this section, the response looks like this:

Line	Comments
RTSP/1.0 503 Service Unavailable	Unsuccessful response to the request
CSeq: 130	Sequence number from the request
Session: 1234567	Session ID from the request
	Blank line

Appendix III Use Cases

This appendix includes several possible examples of the use of ERMI and the EQAM, ERM and M-CMTS core devices in operator networks. These examples are not intended to be definitive, but are offered as guidance for implementers and operators.

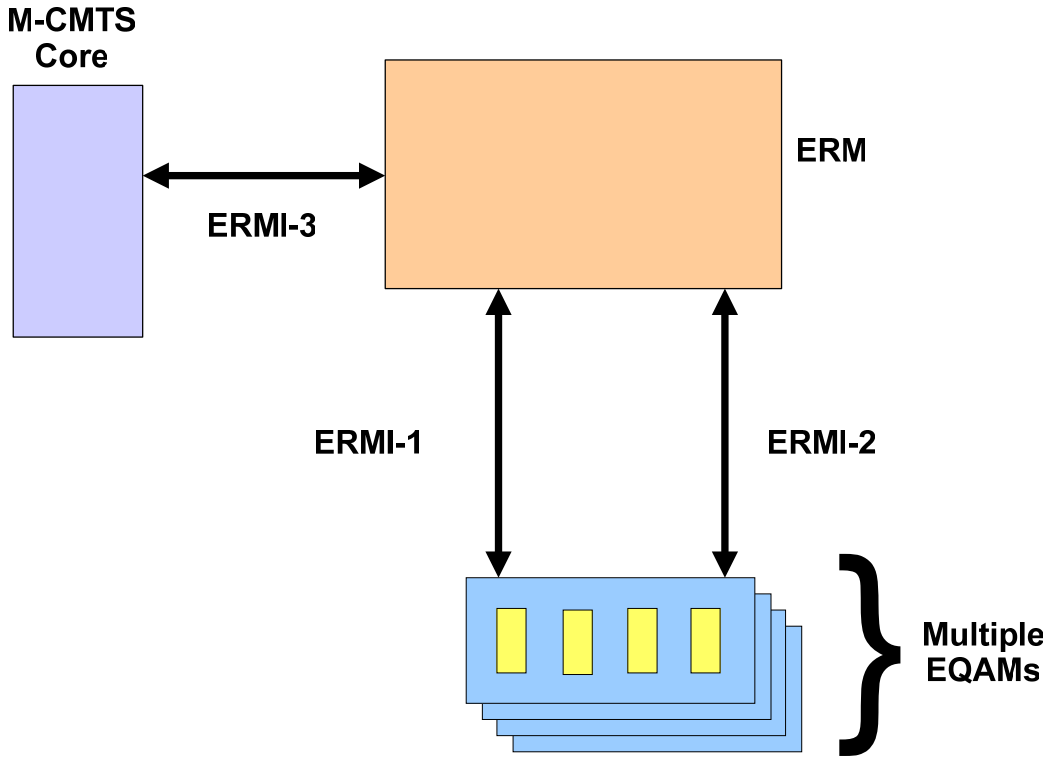

Figure III -1 - Use Case, base architecture[13]

III.1 Booting an EQAM

This section contains an example workflow that could plausibly be followed when a new EQAM device is attached to the operator network and powered up.

[13] Replaced this figure per ECN ERMI-N-05.0244-1 on 11/1/05.

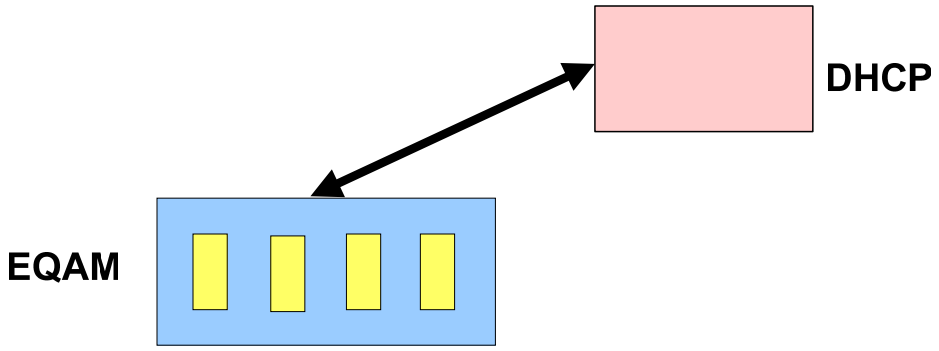

Figure III -2 - Booting an EQAM - DHCP

1. In order to be used on an IP network, the EQAM must first obtain an IP address. This address could be manually configured on to the EQAM; however, manual configuration tends to be error-prone, so most EQAMs will obtain their IP address via an automated boot-time protocol, typically the Dynamic Host Configuration Protocol (DHCP) [RFC 2131]. This specification does not define any particular procedure that must be followed by any network devices when obtaining their IP addresses.

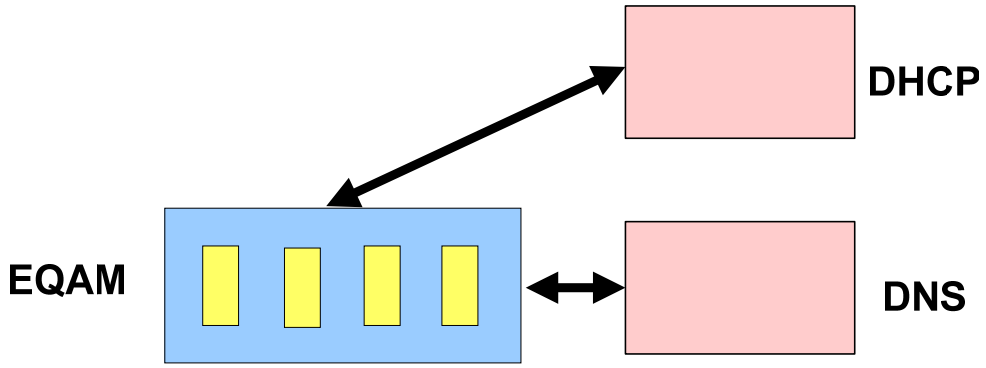

Figure III -3 - Booting an EQAM – DNS[14]

2. The EQAM must know a valid IP address that it can use to contact the ERM. This address could be manually configured on the EQAM; however, manual configuration tends to be error-prone, so most EQAMs will obtain an ERM IP address by some automatic means. Reasonable methods by which this could be achieved include the use of options in a DHCP lookup or using SRV records in a DNS lookup. This specification does not define any particular procedure that must be followed by the EQAM when it obtains the IP address of an ERM.

[14] Replaced this figure per ECN ERMI-N-05.0244-1 on 11/1/05.

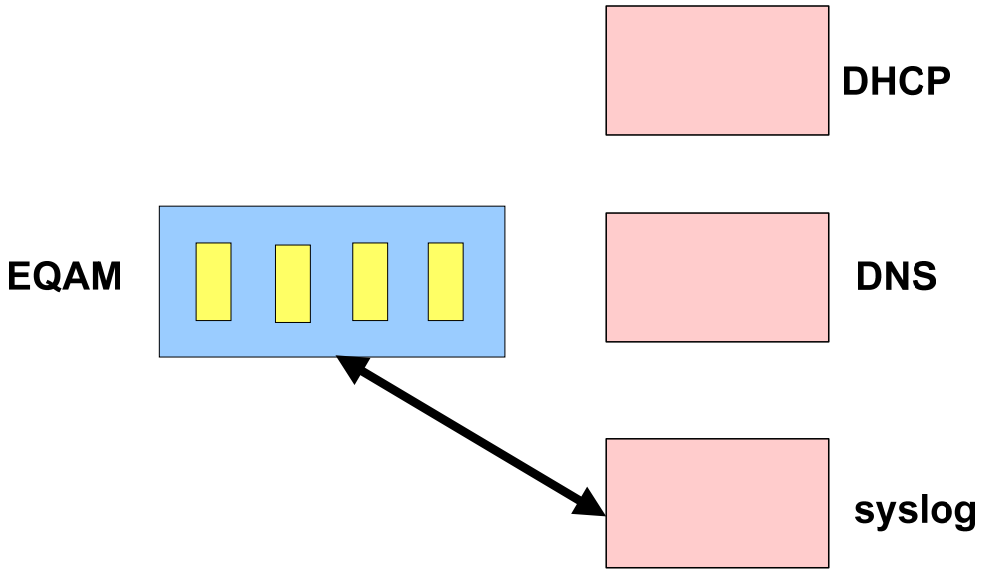

Figure III -4 - Booting an EQAM - syslog[15]

1. Once an EQAM has reached a vendor-defined "in service" state, it will typically report that fact to some system logging facility such as an SNMPv3 entity or a syslog server. This specification does not define how such registration takes place (nor does it require such registration).
2. The EQAM registers its presence and capabilities with the ERM over interface ERMI-1, using DRRP as specified in this document.
3. KEEPALIVES are sent between the EQAM and the ERM to maintain the DRRP connection.

III.2 The M-CMTS obtains a Downstream Resource

At some point, the M-CMTS core requires access to a new downstream QAM channel to provide data to a service group of modems.

1. The M-CMTS core creates a list of the QAM channels that are part of this service group. The method by which it knows the topology of the access network in sufficient detail to create a valid mapping between service groups and QAM channels is not specified in this document. Most likely, the mapping is pre-configured on to the M-CMTS core, either manually through a command-line interface or through a management protocol such as SNMPv3. The M-CMTS core would normally remove from this list any QAM channels that it knows will be unavailable (such as ones that it is already using).
2. The M-CMTS core must know a valid IP address that it can use to contact the ERM. This address could be manually configured on the M-CMTS core, or it could be obtained by some automatic means. Reasonable methods by which this could be achieved include the use of options in a DHCP lookup or using SRV records in a DNS lookup. This specification does not define any particular procedure that must be followed by the M-CMTS core when it obtains the IP address of an ERM.
3. The M-CMTS core sends an RTSP SETUP message to the ERM, giving the details of all the QAM channels in the service group.
4. The ERM examines the list of QAM channels in the SETUP request (which are listed in the order in which the M-CMTS core prefers them), and applies any local policy to the listed QAM channel, to select a "best" QAM channel that it believes to be available.
5. The ERM sends an RTSP SETUP request to the EQAM, identifying the selected QAM channel (only).

[15] Replaced this figure per ECN ERMI-N-05.0244-1 on 11/1/05.

6. The EQAM confirms that the chosen QAM channel is available, and returns a 200 OK response to the SETUP.
7. The ERM returns a 200 OK response to the M-CMTS core, and marks this QAM channel as being in use (*i.e.,* unavailable for use if another SETUP arrives and includes this QAM channel in its list). If the EQAM had returned a failure code other than 453 (indicating Insufficient QAM channel bandwidth), the ERM would have selected the "next-best" QAM channel from the list provided in the SETUP from the M-CMTS core and gone back two steps from here (to step 5) to send an RTSP SETUP request.

Once the M-CMTS core has received the 200 OK from the ERM, it is free to use the QAM channel.

III.3 The M-CMTS core releases a Downstream resource

When an MCMTS core has finished using a resource, it informs the ERM that it has done so, and the ERM returns it to the pool of resources that may be acquired by M-CMTS cores.

1. The M-CMTS core sends an RTSP TEARDOWN message to the ERM; this message contains the session ID that the M-CMTS core received from the ERM in the response to its initial SETUP request.
2. The ERM sends an RTSP teardown message to the EQAM; this message contains the session ID that the ERM received from the EQAM in the response to its initial SETUP request. The EQAM checks whether traffic is still passing on the QAM channel. Assuming that there is no such traffic, it resets the QAM channel configuration parameters to their original values and internally marks the QAM channel as available for use, so that a subsequent request to use it can succeed.
3. The EQAM sends a successful RTSP TEARDOWN response for the session to the ERM. The ERM marks the QAM channel as being available for subsequent SETUP requests.

The ERM sends a successful RTSP TEARDOWN response to the M-CMTS core

III.4 EQAM forces shutdown of a QAM channel

An EQAM may need to shutdown a QAM channel cleanly. Normally, a QAM channel should not be simply removed from service if it is passing traffic; however, this procedure can be used as part of a clean shutdown that will remove a QAM channel from service as soon as it becomes free.

1. The EQAM sends a DRRP UPDATE message to the ERM. This UPDATE message identifies the QAM channel that is being removed from service. The ERM will update its database to reflect the fact that this QAM channel is no longer available for use.
2. The EQAM tears down the DEPI control session to the M-CMTS core (see [DEPI] for details). The M-CMTS core now initiates the usual teardown sequence:
3. The M-CMTS core sends an RTSP TEARDOWN message to the ERM; this message contains the session ID that the M-CMTS core received from the ERM in the response to its initial SETUP request.
4. The ERM sends an RTSP teardown message to the EQAM; this message contains the session ID that the M-CMTS core received from the EQAM in the response to its initial SETUP request. The EQAM checks whether traffic is still passing on the QAM channel. Assuming that there is no such traffic, it resets the QAM channel configuration parameters to their original values and internally marks the QAM channel as available for use, so that a subsequent request to use it can succeed.
5. The EQAM sends a successful RTSP TEARDOWN response for the session to the ERM. The ERM marks the QAM channel as being available for subsequent SETUP requests.
6. The ERM sends a successful RTSP TEARDOWN response to the M-CMTS core

III.5 Broken connections

III.5.1 ERMI-1 transport connection broken

If the DRRP connection between the EQAM and the ERM unexpectedly goes down, the following series of events and messages would be reasonable.

1. The ERM detects the DRRP connection broken and does the following for each resource advertised through this DRRP connection
   - If there is no RTSP session using this resource, remove the resource from its database
   - If there is active RTSP sessions using this resource, mark this resource non-operational. The resource is removed from ERM database after all RTSP session timed out or torn down.
2. The EQAM establishes a new DRRP connection to the ERM.
3. The EQAM advertises its resources using DRRP UPDATE messages.

4. For each advertised resource, the ERM performs the following operations: Search in ERM database for this resource. If the resource does not exist, treat this resource as a new resource advertisement and mark it available for use. If the resource does exist in database, change the resource state to operational.

III.5.2 ERMI-2 transport connection broken

If the RTSP transport connection between the EQAM and the ERM unexpectedly goes down. The ERM attempts to establish a new transport connection to EQAM. Once a new transport connection has been created, both devices proceed as if the transport connection was never lost.

III.5.3 ERMI-3 transport connection broken

If the RTSP transport connection between the M-CMTS core and the ERM unexpectedly goes down, the M-CMTS core establishes a new transport connection to the ERM and both devices proceed as if the transport connection was never lost

III.6 Device failures

III.6.1 Complete EQAM failure

In the event that an EQAM completely fails and there is no automatic failover to an alternative EQAM, the M-CMTS core will detect that the DEPI session has been broken (see [DEPI]).

1. The M-CMTS core sends one or more RTSP TEARDOWN messages to the ERM, tearing down all RTSP sessions that were using QAM channels on the EQAM that has failed. These messages contain the session IDs that the M-CMTS core received from the ERM in the response to its initial SETUP requests.
2. DRRP connection will also fail in this situation. The procedure for handling DRRP connection failure should be followed.

III.6.2 Complete M-CMTS core failure

In the event that an M-CMTS core completely fails, the EQAM will detect that the DEPI session has been broken (see [DEPI]). The RTSP session between the ERM and the M-CMTS core will timeout (since PING messages are no longer being sent), which causes the ERM to teardown the RTSP sessions to this M-CMTS core. ERM also tears down RTSP sessions to EQAM to release all the resources used by this M-CMTS core.

III.6.3 Complete ERM failure

In the event that an ERM fails completely, the DEPI sessions between the M-CMTS core and the EQAM are unaffected.

The M-CMTS core should maintain the RTSP session to the ERM until the DEPI session ends.

The EQAM should maintain the RTSP session to the ERM until the DEPI session ends.

III.7 Device reboots

III.7.1 EQAM reboot

In the event that an EQAM reboots, it will behave exactly as it did on initial boot. The EQAM is not required or expected to maintain any non-configured state information across boots.

III.7.2 M-CMTS core reboot

In the event that an M-CMTS core reboots, the EQAM and the ERM will detect loss of the M-CMTS core and will proceed as in Section III.6.2. The EQAM does not release the QAM channel resources until the ERM instructs it to do so, following the timeout of the RTSP session with the M-CMTS core. The M-CMTS core is not required or expected to maintain any non-configured state information across boots.

1. When the M-CMTS core reboots, it sends a GET_PARAMETER RTSP request to the ERM.
2. The ERM responds by identifying the current RTSP sessions between these two devices. Depending on how quickly the M-CMTS core reboots and the values of the session time-outs, some of the sessions may have been torn down while it rebooted. Normally, it will explicitly tear down any sessions identified by the response to the GET_PARAMETER request.

III.7.3 ERM reboot

In the event that the ERM reboots, the EQAM and the M-CMTS core will detect loss of the ERM and will proceed as in sections <DRRP loss, ERMI-2 loss, ERMI-3 loss, ERM failure>. The ERM is not required or expected to maintain any non-configured state information across boots.

When the ERM reboots, it sends a GET_PARAMETER RTSP request to the EQAM. The EQAM responds by identifying the current RTSP sessions between these two devices. Depending on how quickly the ERM reboots and the values of the session time-outs, some of the sessions may have been torn down while it rebooted. Normally, it will explicitly tear down any sessions identified by the response to the GET_PARAMETER request.

Appendix IV Acknowledgements (Informative)

On behalf of the cable industry and our member companies, CableLabs would like to thank the following individuals for their contributions to the development of this specification.

Charles Bergren	CableLabs
Eduardo Cardona	CableLabs
Doc Evans	ARRIS International, Inc.
Xiaomei Liu	Cisco Systems, Inc.
Michael Mercurio	Camiant
Mike Patrick	Motorola
Sangeeta Ramkrishnan	Cisco Systems, Inc.
Dan Smith	BigBand Networks
Bruce Thompson	Cisco Systems, Inc.

We would particularly like to thank Xiaomei Liu and Doc Evans, who contributed extensively by serving as lead authors for the ERMI specification. We thank Dan Smith, Mike Patrick and Michael Mercurio who provided valuable thoughts and text. We would also like to acknowledge Bruce Thompson and Sangeeta Ramkrishnan for their work on the initial text. We thank Charles Bergren and Eduardo Cardona, who were the ERMI team leads. And finally many thanks go out to all the active members of the ERMI Team who contributed to this specification.

Appendix V Revision History (Informative)

V.1 Engineering Changes for CM-SP-ERMI-I02-051209

The following Engineering Changes are incorporated into CM-SP-ERMI-I02-051209:

ECN	ECN Date	Summary
ERMI-N-05.0244-1	9/28/05	Editorial changes, including making the figures editable.