



***Society of Cable
Telecommunications
Engineers***

**ENGINEERING COMMITTEE
Data Standards Subcommittee**

AMERICAN NATIONAL STANDARD

ANSI/SCTE 137-4 2017

**Edge Resource Manager Interface for Modular Cable
Modem Termination Systems**

NOTICE

The Society of Cable Telecommunications Engineers (SCTE) Standards and Operational Practices (hereafter called “documents”) are intended to serve the public interest by providing specifications, test methods and procedures that promote uniformity of product, interchangeability, best practices and ultimately the long term reliability of broadband communications facilities. These documents shall not in any way preclude any member or non-member of SCTE from manufacturing or selling products not conforming to such documents, nor shall the existence of such standards preclude their voluntary use by those other than SCTE members.

SCTE assumes no obligations or liability whatsoever to any party who may adopt the documents. Such adopting party assumes all risks associated with adoption of these documents, and accepts full responsibility for any damage and/or claims arising from the adoption of such documents.

Attention is called to the possibility that implementation of this document may require the use of subject matter covered by patent rights. By publication of this document, no position is taken with respect to the existence or validity of any patent rights in connection therewith. If a patent holder has filed a statement of willingness to grant a license under these rights on reasonable and nondiscriminatory terms and conditions to applicants desiring to obtain such a license, then details may be obtained from the standards developer. SCTE shall not be responsible for identifying patents for which a license may be required or for conducting inquiries into the legal validity or scope of those patents that are brought to its attention.

Patent holders who believe that they hold patents which are essential to the implementation of this document have been requested to provide information about those patents and any related licensing terms and conditions. Any such declarations made before or after publication of this document are available on the SCTE web site at <http://www.scte.org>.

All Rights Reserved
© Society of Cable Telecommunications Engineers, Inc.
140 Philips Road
Exton, PA 19341

DOCSIS® and M-CMTS™ are trademarks of Cable Television Laboratories, Inc. (CableLabs) and used in this document with permission.

Contents

1	SCOPE	10
1.1	INTRODUCTION AND OVERVIEW	10
1.2	ASSUMPTIONS	11
1.3	EQAM PROFILES	12
1.4	REQUIREMENTS AND CONVENTIONS	13
2	REFERENCES	14
2.1	NORMATIVE REFERENCES	14
2.2	INFORMATIVE REFERENCES	14
2.3	REFERENCE ACQUISITION	14
3	TERMS AND DEFINITIONS	15
4	ABBREVIATIONS AND ACRONYMS	16
5	TECHNICAL OVERVIEW	17
5.1	EDGE ARCHITECTURE OVERVIEW	17
5.2	REGISTRATION INTERFACE	19
5.2.1	Goals, Scope and Constraints	19
5.2.1.1	Registering QAM Channels	19
5.2.2	Overall Architecture	19
5.2.3	ERRP Operation	20
5.2.3.1	ERRP Addressing	20
5.2.3.2	RTSP URLs	21
5.2.3.3	ERRP Timers	21
5.2.3.4	ERRP Attributes	21
5.3	RESOURCE ALLOCATION SIGNALING	22
5.3.1	Resource Allocation Components and Interface	22
5.3.2	Signaling Protocol	23
5.3.3	Selecting an ERM	23
5.4	STATIC PARTITIONING	23
5.4.1	Simplified Architecture for Static QAM Resource Sharing	24
5.4.2	Operation	24
5.5	DEVICE CONFIGURATION	24
6	EDGE RESOURCE REGISTRATION PROTOCOL (ERRP)	26
6.1	RELATIONSHIP WITH TRIP [RFC 3219]	26
6.2	ERRP	26
6.2.1	Establishing a ERRP Connection	26
6.2.2	Message Formats	27
6.2.2.1	Message Header	27
	TABLE 6-1 - ERRP MESSAGE TYPES	27
6.2.2.2	OPEN Message	28
	TABLE 6-2 - CAPABILITY CODES	30
	TABLE 6-3 - SEND RECEIVE CAPABILITY	31
	TABLE 6-4 - ERRP COMPONENT SEND RECEIVE CAPABILITY	31
6.2.2.3	UPDATE Message Format	32
	TABLE 6-5 - ATTRIBUTE FLAG FIELD BIT DEFINITION	34
6.2.2.4	KEEPALIVE Message Format	34
6.2.2.5	NOTIFICATION Message Format	34

TABLE 6-6 - ERRP ERROR CODE.....	35
TABLE 6-7 - MESSAGE HEADER ERROR SUBCODES	35
TABLE 6-8 - OPEN MESSAGE ERROR SUBCODES	35
TABLE 6-9 - UPDATE MESSAGE ERROR SUBCODES.....	36
6.2.3 <i>ERRP Attributes.....</i>	36
TABLE 6-10 - ERRP ATTRIBUTE TYPE CODES.....	36
6.2.3.1 <i>WithdrawnRoutes.....</i>	37
TABLE 6-11 - VALUES FOR ADDRESS FAMILY	38
TABLE 6-12 - APPLICATION PROTOCOLS SUPPORTED IN ERRP	38
TABLE 6-13 - ROUTE NAME	39
6.2.3.2 <i>ReachableRoutes.....</i>	39
6.2.3.3 <i>NextHopServer</i>	39
6.2.3.4 <i>QAM Capability.....</i>	42
TABLE 6-14 - QAM CHANNEL BANDWIDTH CAPABILITY BITS	43
TABLE 6-15 - J83 CAPABILITY BITS	43
TABLE 6-16 - QAM INTERLEAVER CAPABILITY BITS.....	44
TABLE 6-17 - DOCSIS/VIDEO CAPABILITIES - BIT MAP.....	44
TABLE 6-18 - MODULATION CAPABILITY BITS	45
6.2.3.5 <i>Total Bandwidth.....</i>	45
6.2.3.6 <i>QAM Channel Configuration.....</i>	46
TABLE 6-19 - QAM TYPES.....	47
TABLE 6-20 - INTERLEAVER SETTINGS	47
TABLE 6-21 - QAM ANNEX MODES	48
TABLE 6-22 - CHANNEL BANDWIDTH TYPES	48
6.2.3.7 <i>Port ID.....</i>	48
6.2.3.8 <i>Service Status.....</i>	48
TABLE 6-23 - SERVICE STATUS VALUES	49
6.2.3.9 <i>CAS Capability</i>	49
TABLE 6-24 - POTENTIAL VALUES OF THE ENCRYPTION TYPE PARAMETER	50
TABLE 6-25 - POTENTIAL VALUES OF THE ENCRYPTION SCHEME PARAMETER	50
6.2.3.10 <i>Cost.....</i>	50
6.2.3.11 <i>Edge Input</i>	51
6.2.3.12 <i>Input Map</i>	52
6.2.3.13 <i>UDP Map.....</i>	53
6.2.3.14 <i>Max MPEG flows.....</i>	55
6.2.4 <i>ERRP Error Detection and Handling</i>	55
6.2.4.1 <i>Errors in Message Headers.....</i>	55
6.2.4.2 <i>Errors in OPEN Messages.....</i>	56
6.2.4.3 <i>Errors in UPDATE Messages</i>	56
6.2.4.4 <i>Errors in NOTIFICATION Messages</i>	57
6.2.4.5 <i>Hold Timer Expiration</i>	57
6.2.4.6 <i>Errors in the Finite State Machine.....</i>	57
6.2.4.7 <i>Cease</i>	57
6.2.4.8 <i>Connection Collision Detection</i>	57

6.2.5	<i>Negotiating the ERRP Version</i>	58
6.2.6	<i>ERRP Capability Negotiation</i>	58
6.2.7	<i>ERRP Finite State Machine</i>	58
TABLE 6–26 - ERRP FSM STATES		58
TABLE 6–27 - ERRP FSM EVENTS		58
6.2.7.1	[Idle] State.....	59
TABLE 6–28 - ERRP FSM TRANSITIONS FROM [IDLE]		59
6.2.7.2	[Connect] State.....	60
TABLE 6–29 - ERRP FSM TRANSITIONS FROM [CONNECT]		60
6.2.7.3	[Active] State.....	61
TABLE 6–30 - ERRP FSM TRANSITIONS FROM [ACTIVE]		61
6.2.7.4	[OpenSent] State.....	63
TABLE 6–31 - ERRP FSM TRANSITIONS FROM [OPENSENT]		63
6.2.7.5	[OpenConfirm] State.....	64
TABLE 6–32 - ERRP FSM TRANSITIONS FROM [OPENCONFIRM]		64
6.2.7.6	[Established] State.....	65
TABLE 6–33 - ERRP FSM TRANSITIONS FROM [ESTABLISHED]		65
6.3	ERRP MESSAGE EXAMPLES.....	67
6.3.1	<i>OPEN message</i>	68
TABLE 6–34 - EXAMPLE OPEN MESSAGE		68
6.3.2	<i>KEEPALIVE message</i>	68
TABLE 6–35 - EXAMPLE KEEPALIVE MESSAGE		69
6.3.3	<i>UPDATE message</i>	69
TABLE 6–36 - EXAMPLE UPDATE MESSAGE		69
6.3.4	<i>NOTIFICATION message</i>	71
TABLE 6–37 - EXAMPLE NOTIFICATION MESSAGE		71
7 RESOURCE CONFIGURATION AND PROVISIONING		72
7.1	TCP CONNECTION BEHAVIOR FOR RTSP.....	73
7.1.1	<i>Establishing the TCP socket</i>	73
7.1.2	<i>Connection timeout</i>	73
7.2	RTSP URL.....	74
7.3	RTSP METHODS.....	74
7.4	RTSP FINITE STATE MACHINE (FSM).....	75
7.4.1	<i>RTSP Server Finite State Machine</i>	75
TABLE 7–1 - RTSP SERVER FSM STATES		75
TABLE 7–2 - RTSP SERVER FSM EVENTS		75
TABLE 7–3 - RTSP SERVER STATE MACHINE		75
7.4.2	<i>RTSP Client State Machine</i>	76
TABLE 7–4 - RTSP CLIENT FSM STATES		76
TABLE 7–5 - RTSP CLIENT FSM EVENTS		76

TABLE 7-6 - RTSP CLIENT STATE MACHINE	76
7.5 SESSION IDENTIFIERS	76
7.6 RTSP HEADERS.....	77
TABLE 7-7 - SUPPORTED RTSP HEADERS	77
7.7 RTSP EXTENSIONS.....	78
7.7.1 <i>Data Representation</i>	78
7.7.2 <i>Base RTSP Syntax</i>	78
7.7.3 <i>RTSP Header Extensions</i>	78
TABLE 7-8 - RTSP HEADER EXTENSIONS.....	79
7.7.3.1 Extension: clab-ClientSessionId	79
7.7.3.2 Extension: clab-Notice.....	79
TABLE 7-9 - SUPPORTED CLAB-NOTICE CODES.....	80
7.7.3.3 Extension: clab-Reason.....	81
TABLE 7-10 - SUPPORTED TEARDOWN REASON CODES	82
7.7.3.4 Extension: clab-SessionGroup	82
7.7.3.5 Extension: clab-Priority	83
7.7.3.6 Extension: clab-SetupType	83
TABLE 7-11 - DESCRIPTION OF THE CLAB-SETUPTYPE HEADER	83
7.7.3.7 Extension: clab-PidRemap	83
TABLE 7-12 - CLAB-PIDREMAP EXTENSION	84
7.7.3.8 Extension: clab-MPTSMODE	84
7.7.3.9 Extension: clab-StatmuxGroup	84
7.7.4 <i>SETUP Transport Headers</i>	84
7.7.4.1 Transport Header Syntax.....	85
7.7.4.2 Transport header format –DOCSIS Data	87
7.7.4.3 Transport header format – Unicast video	89
7.7.4.4 Transport header format – Multicast video	90
7.7.4.5 Transport Header Use	91
7.8 RTSP ENTITY BODY.....	92
7.8.1 <i>Entity Body - text/xml</i>	92
7.8.2 <i>Entity Body - text/parameters</i>	92
7.8.2.1 Parameter: clab-session-list.....	93
7.8.2.2 Parameter: clab-connection-timeout.....	93
7.8.2.3 Parameter: clab-sessiongroup-list.....	94
7.9 SESSION KEEPALIVES AND MESSAGE TIMEOUT.....	94
7.9.1 <i>Session Keepalives and Timeout</i>	94
7.9.1.1 Session Timeout Value	94
7.9.1.2 Session Timeout Behavior	95
7.9.2 <i>Message Timeout</i>	95
7.10 RTSP RESPONSE CODE	95
TABLE 7-13 - SUPPORTED RTSP RESPONSE CODES.....	95
7.11 RTSP MESSAGE EXAMPLES	96
7.11.1 <i>SETUP Message Examples</i>	96
7.11.1.1 Session Setup for Unicast	96
7.11.1.2 Session Setup for Multicast	97
7.11.1.3 Setup for MPTS Sessions	98
7.11.1.4 Session Setup for M-CMTS.....	102
7.11.2 <i>TEARDOWN Message Examples</i>	104
7.11.2.1 Introduction	104
7.11.2.2 Message Headers	104
7.11.2.3 Session Teardown.....	104

7.11.3	<i>SET_PARAMETER Keepalive Message Examples</i>	105
7.11.3.1	Introduction	105
7.11.3.2	Interaction Diagram.....	105
7.11.3.3	Message Headers	105
7.11.3.4	Keepalive Interaction Scenario	105
7.11.3.5	Session Keepalive.....	106
7.11.4	<i>GET_PARAMETER Message Examples</i>	106
7.11.4.1	Introduction	106
7.11.4.2	Interaction Diagram.....	106
7.11.4.3	Message Headers	107
7.11.4.4	GET_PARAMETER Interaction Scenario	108
7.11.4.5	Get Parameter	108
7.11.5	<i>ANNOUNCE Message Examples</i>	109
7.11.5.1	Introduction	109
7.11.5.2	Interaction Diagram.....	109
7.11.5.3	Downstream Failure Message Header	109
7.11.5.4	ANNOUNCE Interaction Scenario.....	110
7.11.5.5	Session Announce.....	110
7.12	DOCSIS RESOURCE ALLOCATION OPERATION	111
7.12.1	Resource Allocation.....	111
7.12.2	Resource De-allocation	113
7.12.3	Multiple QAM channels in MAC Domain.....	113
7.12.4	Synchronization with DEPI control [DEPI]	113
ANNEX A	XML EXTENSIONS	116
A.1	ENCRYPTIONDATA DESCRIPTOR DEFINITIONS	116
A.2	XML SCHEMA DEFINITION	117
APPENDIX I	USE CASES	119
I.1	THE M-CMTS OBTAINS A DOWNSTREAM RESOURCE	119
I.2	THE M-CMTS CORE RELEASES A DOWNSTREAM RESOURCE	120
I.3	EQAM FORCES SHUTDOWN OF A QAM CHANNEL.....	120
I.4	BROKEN CONNECTIONS	121
I.4.1	ERMI-1 transport connection broken	121
I.4.2	ERMI-2 transport connection broken	121
I.4.3	ERMI-3 transport connection broken	121
I.5	DEVICE FAILURES.....	121
I.5.1	Complete EQAM failure	121
I.5.2	Complete M-CMTS Core failure.....	121
I.5.3	Complete ERM failure	122
I.6	DEVICE REBOOTS.....	122
I.6.1	EQAM reboot.....	122
I.6.2	M-CMTS Core reboot.....	122
I.6.3	ERM reboot.....	122
I.7	VIDEO ON DEMAND	122
I.8	SWITCHED DIGITAL VIDEO.....	123
I.8.1	Synchronous and Asynchronous Modes.....	123
APPENDIX II	DIGITAL PROGRAM INSERTION	125
II.1	BACKGROUND	125
II.2	INFORMATIVE TEXT	126
II.2.1	Treatment of Program numbers and PIDs.....	126

Figures

FIGURE 1–1 - M-CMTS REFERENCE ARCHITECTURE.....	11
---	----

FIGURE 1–2 - ERMI INTERFACES	12
FIGURE 5–1 - RF TOPOLOGY	18
FIGURE 5–2 - REGISTRATION INTERFACE AND COMPONENTS	20
FIGURE 5–3 - RESOURCE ALLOCATION INTERFACES AND COMPONENTS	22
FIGURE 5–4 - SIMPLIFIED FRAMEWORK FOR STATIC QAM PARTITIONING	24
FIGURE 6–1 - DDRP HEADER FORMAT	27
FIGURE 6–2 - ERRP OPEN HEADER	28
FIGURE 6–3 - OPTIONAL PARAMETER ENCODING.....	29
FIGURE 6–4 - CAPABILITY OPTIONAL PARAMETER	30
FIGURE 6–5 - ROUTE TYPE FORMAT.....	30
FIGURE 6–6 - ERRP UPDATE FORMAT.....	32
FIGURE 6–7 - ROUTING ATTRIBUTE FORMAT	33
FIGURE 6–8 - ATTRIBUTE TYPE FORMAT	33
FIGURE 6–9 - ERRP NOTIFICATION FORMAT	34
FIGURE 6–10 - WITHDRAWROUTES FORMAT	37
FIGURE 6–11 - ROUTE FORMAT FOR WITHDRAWROUTES AND REACHABLEROUTES.....	38
FIGURE 6–12 - REACHABLEROUTES FORMAT.....	39
FIGURE 6–13 - NEXTHOPSERVER SYNTAX.....	40
FIGURE 6–14 - NEXTHOPSERVERALTERNATES SYNTAX.....	41
FIGURE 6–15 - QAM NAMES ATTRIBUTE SYNTAX.....	42
FIGURE 6–16 - FIBER NODES ATTRIBUTE SYNTAX	42
FIGURE 6–17 - QAM CAPABILITY FORMAT	43
FIGURE 6–18 - TOTAL BANDWIDTH SYNTAX	45
FIGURE 6–19 - AVAILABLE BANDWIDTH ATTRIBUTE.....	46
FIGURE 6–20 - QAM CONFIGURATION ATTRIBUTE.....	46
FIGURE 6–21 - PORT ID FORMAT	48
FIGURE 6–22 - SERVICE STATUS FORMAT	49
FIGURE 6–23 - CAS CAPABILITY ATTRIBUTE SYNTAX	50
FIGURE 6–24 - COST ATTRIBUTE SYNTAX.....	51
FIGURE 6–25 - EDGE INPUT ATTRIBUTE SYNTAX	51
FIGURE 6–26 - INPUT MAP ATTRIBUTE	52
FIGURE 6–27 - UDP MAP ATTRIBUTE SYNTAX	54
FIGURE 6–28 - MAX MPEG FLOWS ATTRIBUTE SYNTAX	55
FIGURE 6–29 - EXAMPLE ERRP CONNECTION ESTABLISHMENT.....	67
FIGURE 7–1 - RTSP CLIENT - RTSP SERVER RELATIONSHIPS.....	73
FIGURE 7–2 - KEEPALIVE INTERACTION DIAGRAM	105
FIGURE 7–3 - GET_PARAMETER INTERACTION DIAGRAM.....	106
FIGURE 7–4 - ANNOUNCE INTERACTION DIAGRAM	109
FIGURE 7–5 - RTSP SETUP MESSAGE FLOW	111
FIGURE 7–6 - RTSP TEARDOWN MESSAGE FLOW.....	113
FIGURE 7–7 - SESSION SETUP SEQUENCE WITH DEPI.....	114
FIGURE 7–8 - SESSION TEARDOWN SEQUENCE WITH DEPI	114
FIGURE I–1 - USE CASE, BASE ARCHITECTURE	119
FIGURE I–2 - DYNAMIC SESSION LIFECYCLE.....	123
FIGURE I–3 - MESSAGE FLOW FOR SYNCHRONOUS MODE	124
FIGURE I–4 - MESSAGE FLOW FOR ASYNCHRONOUS MODE.....	124
FIGURE II–1 - DIGITAL PROGRAM INSERTION DIAGRAM.....	125

Tables

TABLE 6–1 - ERRP MESSAGE TYPES	27
TABLE 6–2 - CAPABILITY CODES	30

TABLE 6-3 - SEND RECEIVE CAPABILITY	31
TABLE 6-4 - ERRP COMPONENT SEND RECEIVE CAPABILITY	31
TABLE 6-5 - ATTRIBUTE FLAG FIELD BIT DEFINITION.....	34
TABLE 6-6 - ERRP ERROR CODE.....	35
TABLE 6-7 - MESSAGE HEADER ERROR SUBCODES.....	35
TABLE 6-8 - OPEN MESSAGE ERROR SUBCODES	35
TABLE 6-9 - UPDATE MESSAGE ERROR SUBCODES.....	36
TABLE 6-10 - ERRP ATTRIBUTE TYPE CODES.....	36
TABLE 6-11 - VALUES FOR ADDRESS FAMILY	38
TABLE 6-12 - APPLICATION PROTOCOLS SUPPORTED IN ERRP	38
TABLE 6-13 - ROUTE NAME.....	39
TABLE 6-14 - QAM CHANNEL BANDWIDTH CAPABILITY BITS	43
TABLE 6-15 - J83 CAPABILITY BITS.....	43
TABLE 6-16 - QAM INTERLEAVER CAPABILITY BITS	44
TABLE 6-17 - DOCSIS/VIDEO CAPABILITIES - BIT MAP.....	44
TABLE 6-18 - MODULATION CAPABILITY BITS.....	45
TABLE 6-19 - QAM TYPES	47
TABLE 6-20 - INTERLEAVER SETTINGS	47
TABLE 6-21 - QAM ANNEX MODES	48
TABLE 6-22 - CHANNEL BANDWIDTH TYPES.....	48
TABLE 6-23 - SERVICE STATUS VALUES.....	49
TABLE 6-24 - POTENTIAL VALUES OF THE ENCRYPTION TYPE PARAMETER	50
TABLE 6-25 - POTENTIAL VALUES OF THE ENCRYPTION SCHEME PARAMETER	50
TABLE 6-26 - ERRP FSM STATES	58
TABLE 6-27 - ERRP FSM EVENTS	58
TABLE 6-28 - ERRP FSM TRANSITIONS FROM [IDLE].....	59
TABLE 6-29 - ERRP FSM TRANSITIONS FROM [CONNECT].....	60
TABLE 6-30 - ERRP FSM TRANSITIONS FROM [ACTIVE]	61
TABLE 6-31 - ERRP FSM TRANSITIONS FROM [OPENSent]	63
TABLE 6-32 - ERRP FSM TRANSITIONS FROM [OPENCONFIRM].....	64
TABLE 6-33 - ERRP FSM TRANSITIONS FROM [ESTABLISHED].....	65
TABLE 6-34 - EXAMPLE OPEN MESSAGE	68
TABLE 6-35 - EXAMPLE KEEPALIVE MESSAGE	69
TABLE 6-36 - EXAMPLE UPDATE MESSAGE	69
TABLE 6-37 - EXAMPLE NOTIFICATION MESSAGE.....	71
TABLE 7-1 - RTSP SERVER FSM STATES.....	75
TABLE 7-2 - RTSP SERVER FSM EVENTS	75
TABLE 7-3 - RTSP SERVER STATE MACHINE	75
TABLE 7-4 - RTSP CLIENT FSM STATES.....	76
TABLE 7-5 - RTSP CLIENT FSM EVENTS	76
TABLE 7-6 - RTSP CLIENT STATE MACHINE.....	76
TABLE 7-7 - SUPPORTED RTSP HEADERS	77
TABLE 7-8 - RTSP HEADER EXTENSIONS	79
TABLE 7-9 - SUPPORTED CLAB-NOTICE CODES	80
TABLE 7-10 - SUPPORTED TEARDOWN REASON CODES.....	82
TABLE 7-11 - DESCRIPTION OF THE CLAB-SETUPType HEADER	83
TABLE 7-12 - CLAB-PIDREMAP EXTENSION.....	84
TABLE 7-13 - SUPPORTED RTSP RESPONSE CODES.....	95

1 SCOPE

NOTE: This document is identical to SCTE 137-4 2010 except for informative components which may have been updated such as the title page, NOTICE text, headers and footers. No normative changes have been made to this document.

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 resources from an EQAM, and by an EQAM to acknowledge 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.

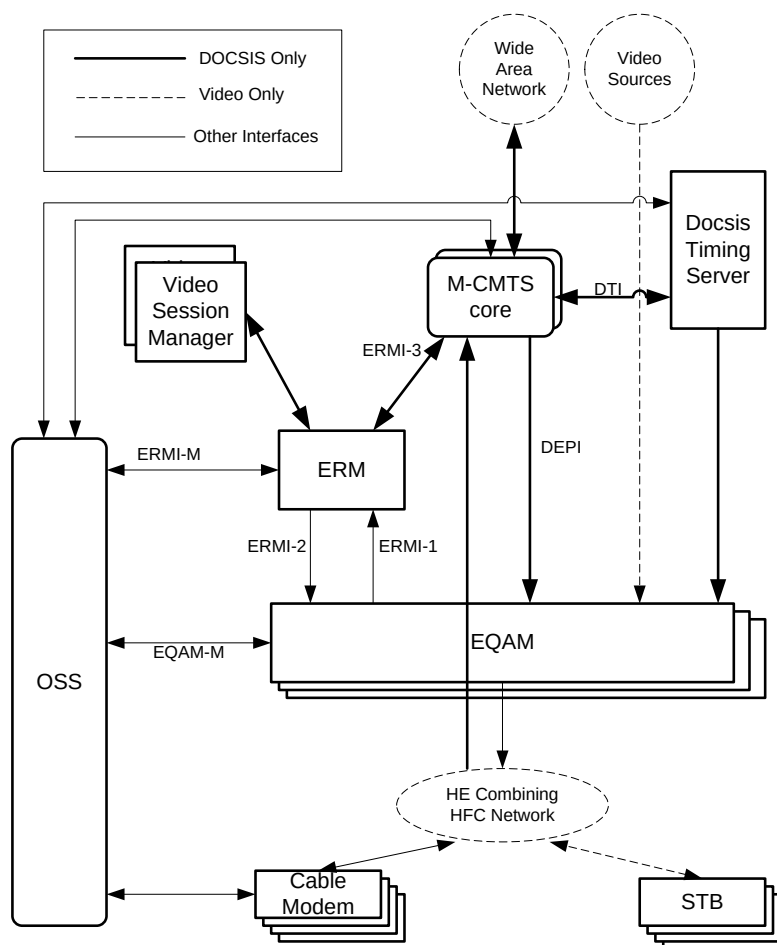


Figure 1-1 - M-CMTS Reference Architecture

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 cooperating 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. A QAM channel is also identified by a unique QAM name.

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 video resources. The interface between Video Session Managers and the ERM is out of scope.

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).

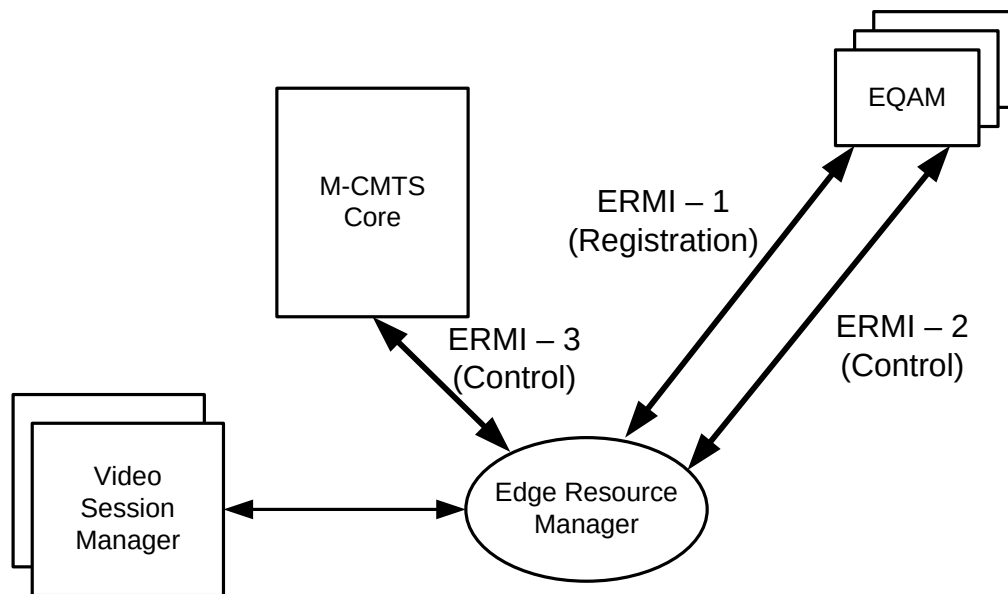


Figure 1-2 - ERM Interfaces

1.3 EQAM Profiles

An EQAM can be any of three designated profiles:

- Video EQAM Profile

An EQAM of this profile supports DOCSIS EQAM requirements applicable to delivering digital video and does not support DOCSIS M-CMTS EQAM requirements.

A video EQAM supporting only static UDP port mapping does not satisfy all necessary requirements to be designated as fitting this profile.

- M-CMTS EQAM Profile

An EQAM of this profile supports requirements applicable to an M-CMTS EQAM.

- Universal EQAM Profile

An EQAM of this profile supports requirements applicable to delivering digital video and supporting M-CMTS EQAM requirements. Output QAM channels can be flexibly allocated to digital video delivery or DOCSIS high speed data service. It is not required that the Universal EQAM be able to multiplex both digital video and DOCSIS data on the same QAM channel.

This specification contains requirements in support of all three EQAM profiles. To delineate the application of normative statements for requirements that are specific either to the support of digital video functionality or to the support of M-CMTS functionality, this specification explicitly indicates that such a requirement applies to the video profile or to the M-CMTS profile. The abovementioned indications notwithstanding, all EQAM requirement statements provided in this specification apply to the Universal EQAM profile.

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

The following documents contain provisions, which, through reference in this text, constitute provisions of this standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreement based on this standard are encouraged to investigate the possibility of applying the most recent editions of the documents listed below.

2.1 Normative References

- | | |
|------------|---|
| [DEPI] | ANSI/SCTE 137-2 2007, DOCSIS Downstream External PHY Interface for Modular Cable Modem Termination Systems. |
| [J.83] | ITU-T Recommendation. J.83, (04/97) Digital multi-programme systems for television sound and data services for cable distribution. |
| [RFC 1123] | IETF RFC 1123/STD 3, Braden, R., "Requirements for Internet Hosts – Application and Support" October 1989, Internet Engineering Task Force. |
| [RFC 2068] | IETF RFC 2068, R. Fielding et al., "Hypertext Transfer Protocol – HTTP/1.1", January 1997, Internet Engineering Task Force. |
| [RFC 2326] | IETF RFC 2326, H. Schulzrinne; et al, Real Time Streaming Protocol (RTSP), April 1998, Internet Engineering Task Force. |
| [RFC 4291] | IETF RFC 4291, Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", February 2006, Internet Engineering Task Force. |
| [RFC 3219] | IETF RFC 3219, J. Rosenberg, H. Salama and M. Squire, "Telephony Routing over IP (TRIP)," January 2002, Internet Engineering Task Force. |
| [VSI] | ANSI/SCTE 137-6 2010, Modular Headend Architecture Part 6: Edge QAM Video Stream Interface |

2.2 Informative References

- | | |
|------------|--|
| [DSG] | SCTE 106 2010, DOCSIS® Set-top Gateway (DSG) Specification. |
| [RFC 2131] | IETF RFC 2131, R. Droms, "Dynamic Host Configuration Protocol", March 1997, Internet Engineering Task Force. |
| [RFI2] | ANSI/SCTE 79-1 2009, DOCSIS 2.0 Part 1: Radio Frequency Interface. |

2.3 Reference Acquisition

- Internet Engineering Task Force (IETF), <http://www.ietf.org>
- International Telecommunication Union – Telecommunication Standardization Sector (ITU-T), <http://www.itu.int/itu-t/>

3 TERMS AND DEFINITIONS

This specification defines the following terms:

Edge Input Group	A set of EQAM input interfaces that have equivalent connectivity in the operator's Ethernet network.
Edge Resource Manager	A network element that manages the input and output resources of an EQAM via the protocols defined in this specification.
EQAM (or 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.
ERRP Node	An entity that participates in exchanges of ERRP messages; an ERM or an EQAM.
Input Map	A list of EQAM input interfaces that reach a particular QAM channel within the internal architecture of an EQAM.
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.
QAM Group	A set of QAM channels that reach a common set of set-top boxes.
Route	In the scope of ERRP, a QAM channel.
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 or set-top boxes that share a common spectrum of RF channels.
Transport-spec	A comma delimited element in an RTSP transport header.

4 ABBREVIATIONS AND ACRONYMS

This specification uses the following abbreviations and acronyms:

aBNF	augmented Backus-Naur Form
CA	Conditional Access
CAS	Conditional Access System
CM	Cable Modem
CMTS	Cable Modem Termination System
DEPI	DOCSIS External PHY Interface
DOCSIS®	Data-Over-Cable Service Interface Specifications
DOCSIS-MPT	DOCSIS MPEG Transport Stream mode of DEPI
DOCSIS-PSP	DOCSIS-Packet-Streaming-Protocol of DEPI
DS	Downstream
DSM-CC	Digital Storage Media Command and Control
EQAM	Edge QAM
ERM	Edge Resource Manager
ERMI	Edge Resource Manager Interface(s)
ERRP	Edge Resource Registration Protocol
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
MPTS	Multiple Program Transport Stream
PID	Packet Identifier
PSP	Packet Streaming Protocol
QAM	Quadrature Amplitude Modulator (or Modulation)
RF	Radio Frequency
RTSP	Real-Time Streaming Protocol
SDV	Switched Digital Video
SPTS	Single Program Transport Stream
SRM	Session Resource Manager
SSP	Session Setup Protocol
STB	Set top box
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

An EQAM is an edge device that receives packets of digital video or data from the IP network. It re-packetizes the video or data and delivers to the HFC network using QAM outputs. ERM is an edge resource manager that manages QAM resources on EQAMs.

Specifically, ERMI defines a mechanism for an ERM to discover EQAM resources via a registration interface. ERMI also defines resource allocation interfaces for an ERM to allocate QAM resources from an EQAM and for a CMTS core to allocate QAM resources from an ERM.

5.1 Edge Architecture Overview

An EQAM can be modeled as a device that has a number of IP input interfaces and a number of QAM channel outputs. The ERM learns the inputs and outputs from the registration interface.

Input interfaces advertised to an ERM are considered by the ERM as IP destinations that can be used to reach QAM channel outputs of the registering EQAM. Input interfaces advertised may be physical interface addresses or virtual interface addresses of the EQAM. An EQAM may support either any-to-any or partial internal connectivity between input interfaces and output QAM channels. When only partial connectivity is supported, the input to output mapping is advertised by the EQAM to the ERM via an "Input Map" for each QAM channel. The ERM might need to further subdivide the inputs of an EQAM (or of an Input Map in the case of partial internal connectivity) based on network connectivity external to the EQAM. A set of EQAM input interfaces that have equivalent connectivity in the operator's Ethernet network (i.e., source to EQAM) are called an Edge Input Group (EIG).

An ERM manages the EQAM output at the granularity of a QAM channel, though several QAM channels can be physically connected to a single RF port on the EQAM. QAM channels are grouped to TSID groups and QAM groups. TSID group is used to manage the configuration of QAM channels when there are hardware limitations on QAM channel independency. For example, QAM channels connected to the same RF port may be restricted to have a contiguous frequency assignment. QAM group is defined as a group of QAM channels that have equivalent connectivity in the RF topology (i.e., they reach the same set of fiber nodes).

The following figure illustrates EQAMs, their input and output relationships, and these network topology concepts. In the diagram, there are two EQAMs. The first EQAM has four input interfaces (G1, G2, G3, and G4) and four output QAM channels (Q1, Q2, Q3, and Q4), with any-to-any internal connectivity between inputs and outputs. This EQAM has its four input interfaces grouped into two Edge Input Groups EIG1 (G1 and G2) and EIG2 (G3 and G4) based on the Ethernet network topology. This EQAM has its four QAM channels grouped into two QAM groups QG1 (Q1 and Q2) and QG2 (Q3 and Q4) based on the RF topology. The second EQAM has four input interfaces (G5, G6, G7, G8) and four output QAM channels (Q5, Q6, Q7, Q8), but does not have any-to-any internal connectivity between inputs and outputs. In this EQAM, Q5 and Q6 are only reachable by G5 and G6, while Q7 and Q8 are only reachable by G7 and G8. This EQAM would advertise these input maps in registration. Even though the Ethernet network topology is common for all four input interfaces, and thus they are configured as a single Edge Input Group, the input maps result in the ERM restricting its input interface selection when assigning a stream to an output QAM channel. Since all four output QAM channels reach the same set of fiber nodes, they are considered to be a single QAM group (QG3).

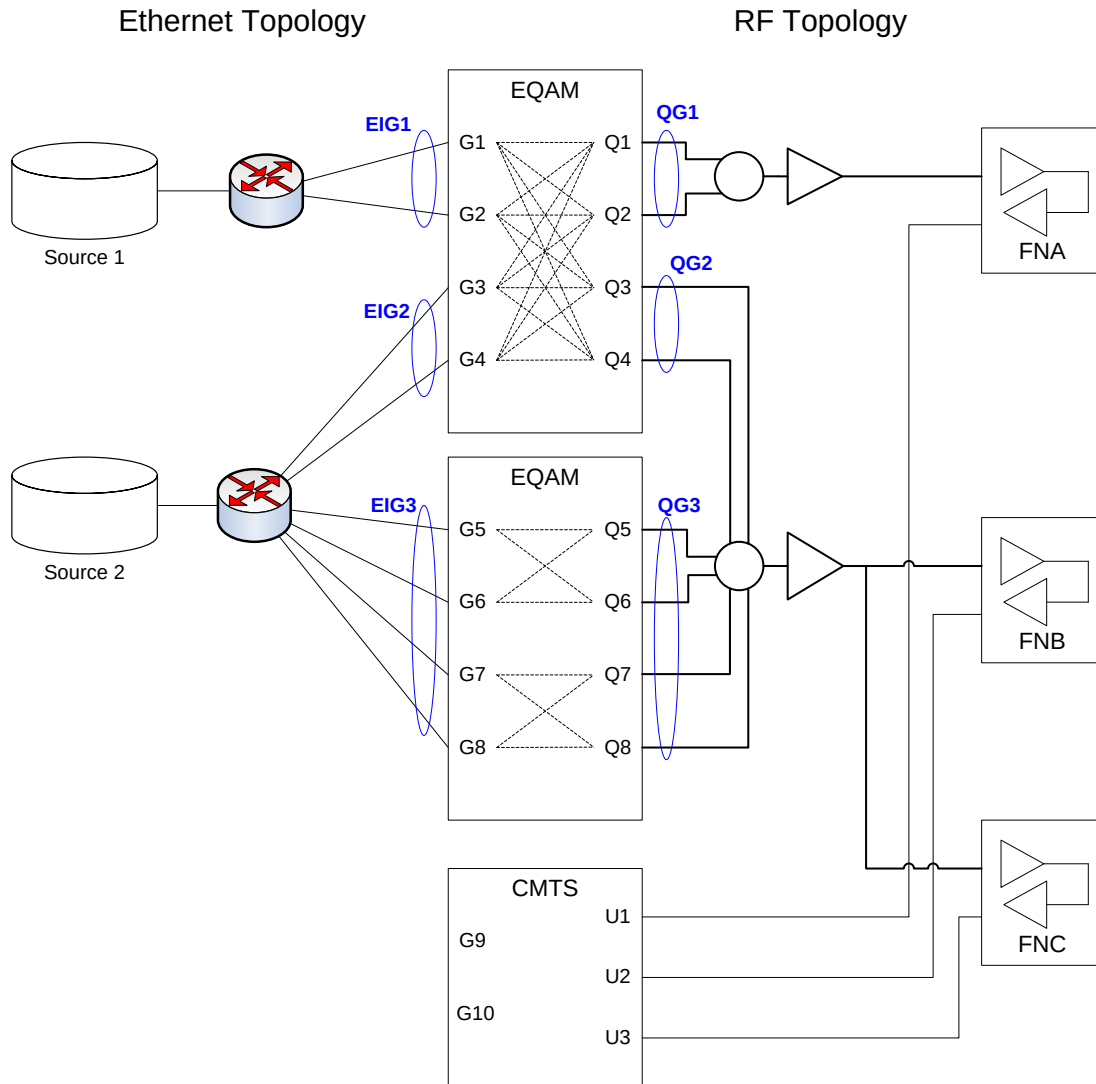


Figure 5-1 - RF Topology

ERMI facilitates the RF topology configuration and discovery feature introduced in DOCSIS3.0. In DOCSIS 3.0, fiber node is configured and used such that the downstream service groups and upstream service groups can be associated in MAC domains. Downstream service group is the complete set of downstream channels that could potentially reach a single Cable Modem or cable STB. Throughout this spec, service group refers to downstream service group.

A service group may span one or multiple fiber nodes. A service group may include one or multiple QAM groups. In the edge architecture shown in Figure 5-1, service group A (SGA) is feeding fiber node A. Service group SGA includes only a single QAM group QG1. Service group B (SGB) is feeding both fiber node B and fiber node C. Service group SGB includes QAM groups QG2 and QG3. In addition, it is also possible for a QAM group to belong to multiple service groups.

RF topology information is configured at EQAMs. QAM channel to QAM group mapping and QAM group to fiber node mapping are configured at EQAMs. An ERM learns the RF topology information via the registration interface. When a CMTS core requests edge resources from an ERM, the fiber node information is communicated between the CMTS core and the ERM. When a Video Session Manager requests edge resources from an ERM, the service group QAM list is communicated between the Video Session manager and the ERM.

Service groups are not directly managed by the EQAM and the ERM. Service groups may be managed directly by higher level entities such as the CMTS core or a video session manager.

5.2 Registration Interface

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

5.2.1 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.

TRIP is extended for this application to provide EQAMs with a mechanism to advertise reachability of QAM channels and to advertise capabilities and attributes of those QAM channels.

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.

5.2.1.1 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
- Fiber nodes
- 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.

5.2.2 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 5-2 shows the Registration component architecture. The Registration Interface between an ERM and an EQAM carries messages conforming to the Edge Resource Registration Protocol (ERRP), as specified in Section 6.2.

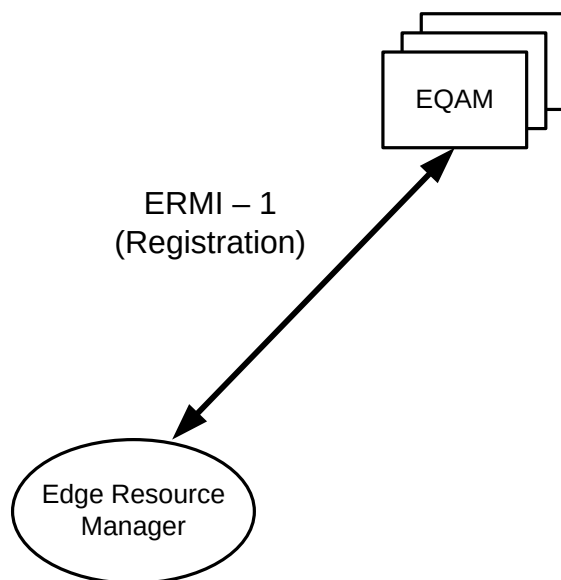


Figure 5-2 - Registration Interface and Components

An ERM normally manages multiple EQAMs, as shown in the Figure 5-2. 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 ERRP protocol are carried over a TCP/IP connection that may be initiated by either the EQAM or the ERM.

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

Because the device control interface for an EQAM is based on RTSP (see Section 6), 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 then the ERM returns addressing information for that resource (over ERMI-3) to the requesting M-CMTS Core.

In addition to an initial advertisement, the EQAM sends additional advertisements (over ERMI-1) 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.

5.2.3 ERRP Operation

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

5.2.3.1 ERRP Addressing

An ERRP 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 initializes, 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 allocate this QAM channel resource to the DOCSIS session. 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 ERRP ReachableRoute attribute to the ERM, embedded in a ERRP UPDATE message. To withdraw a QAM channel from service, the EQAM sends a WithdrawRoute attribute.¹

ERRP 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 ERRP NextHopServer attribute are described in Section 6.2.3.3.

5.2.3.2 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 slash: rtsp://hostname[:port]/.

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.

5.2.3.3 ERRP Timers

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

5.2.3.3.1 Hold Timer

Whenever an ERRP 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 ERRP connection to be closed. The duration of the Hold Timer, which is called the Hold Time, is negotiated by the ERRP Nodes when the ERRP 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.

5.2.3.3.2 ConnectRetry Timer

The ConnectRetry Timer is used during the process of establishing a ERRP connection. After the ERRP Node initiates the TCP connection to the remote ERRP 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 ERRP Node will retry to establish the TCP connection.

5.2.3.4 ERRP Attributes

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

¹ 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.

5.3 Resource Allocation Signaling

5.3.1 Resource Allocation Components and Interface

Figure 5-3 shows the components involved in the resource allocation interfaces.

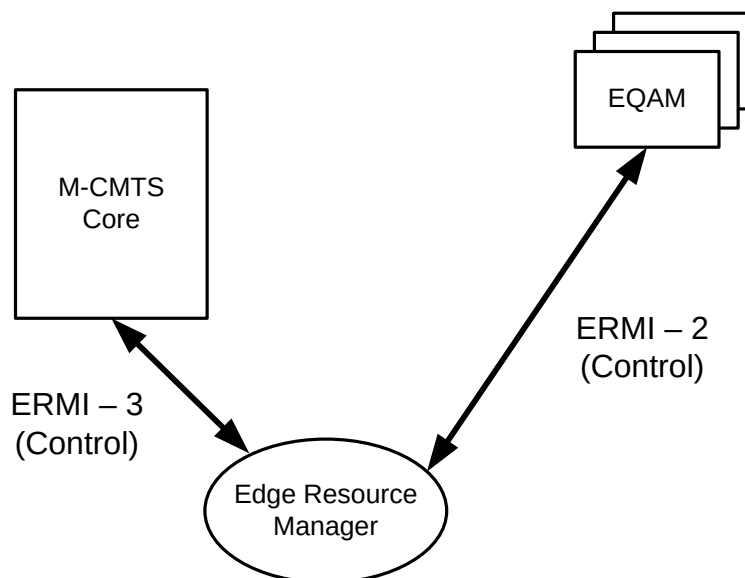


Figure 5-3 - Resource Allocation Interfaces and Components

The M-CMTS Core or Video Session Manager initiates a QAM resource transaction with the ERM when it requests or releases a QAM channel resource. When an M-CMTS requests a MAC domain to be 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, video, or both. 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 video (i.e., VOD, SDV etc.) traffic; and the existence of QAM channels that have been reserved for future DOCSIS traffic, etc.

In an EQAM, QAM channels are physically present on external 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 flow 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 flows 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 allocate 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. The interface between the ERM and Video Session Manager is beyond the scope of this document. When the CMTS core allocates QAM channel resources from the ERM, either explicit list of QAMs or fiber node information can be given to the ERM.

5.3.2 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 video applications, RTSP is extended with new methods and headers to support the video application. In the DOCSIS and video applications specified in this document, additional headers are defined. Naturally, in order for an ERM/EQAM pair to function correctly in both DOCSIS and video applications, all 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 ERM resource allocation interfaces (ERMI-2 and ERMI-3). In addition, extensions needed for the DOCSIS application are also specified herein.

5.3.3 Selecting an ERM

There may be multiple ERMs in the operator's head-end network. The M-CMTS Core or Video Session Manager 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.

5.4 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.

5.4.1 Simplified Architecture for Static QAM Resource Sharing

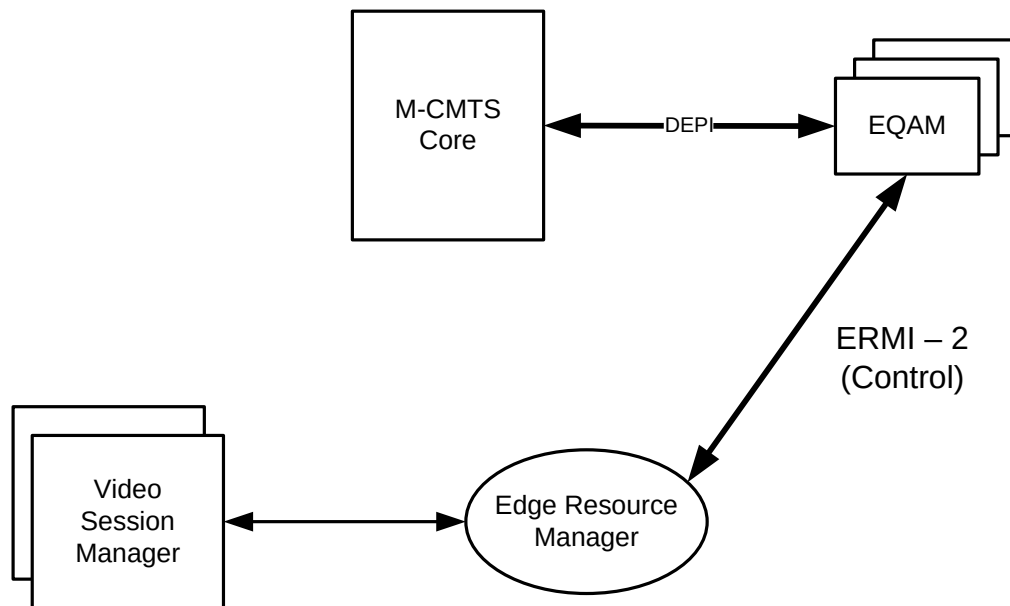


Figure 5-4 - Simplified Framework for Static QAM Partitioning

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).

5.4.2 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 EQAM 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].

5.5 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 partitioning is slightly different. The differences are noted below.

Each QAM channel in the EQAM is configured with:

- QAM TSID and QAM name;
- Input port IP addresses
- Fiber Node list
- Modulation type;
- Channel bandwidth;
- Interleaver 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, configuration depends on whether static partitioning or the dynamic signaling is used for QAM sharing. When static partitioning is used, QAM TSID, QAM IP address and QAM RF topology are configured. When dynamic signaling is used, ERM IP is configured. In addition, the M-CMTS Core may choose to either directly configure QAM RF topology or just configure the fiber node for RF topology.

6 Edge Resource Registration Protocol (ERRP)

Several requirements in this section are written for an "ERRP Speaker". The EQAM **MUST** implement the ERRP Speaker functionality. Some requirements are written for an "ERRP Listener". The ERM **MUST** implement the ERRP Listener functionality. Many of the requirements in this section apply to both ERRP Speakers and ERRP Listeners, and are written for an "ERRP Node". The EQAM **MUST** implement the ERRP Node functionality. The ERM **MUST** implement the ERRP Node functionality.

6.1 Relationship with TRIP [RFC 3219]

ERRP shares many similarities with the protocol described in TRIP, [RFC 3219]. While TRIP was originally developed for telephony services, a subset of the protocol can be used to deal with the resource manager selection problem addressed in this specification, specifically, TRIP has similar procedures and a similar Finite State Machine for connection establishment. TRIP also shares the same format for messages. ERRP Nodes are also conformant TRIP nodes, although they perform only a subset of the messaging described in [RFC 3219].

When TRIP is adapted to ERRP in a cable environment, data plane devices will be TRIP speakers, and resource managers will act as TRIP listeners.

ERRP supports four messages that may be exchanged between ERRP Nodes: OPEN, UPDATE, NOTIFICATION, and KEEPALIVE:

- The OPEN message is used to initiate a ERRP connection between ERRP 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 an ERRP Node to inform the far end that an error has occurred. ERRP connections are closed after a NOTIFICATION message is sent or received. The NOTIFICATION message is used exactly as specified in [RFC 3219].
- ERRP includes a periodic bidirectional KEEPALIVE message whose frequency is negotiated by the two sides when the ERRP 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 ERRP 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 ERRP

This section documents the subset of [RFC 3219] used in this architecture, as well as the additional attributes beyond [RFC 3219] 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 ERRP, the Edge Resource Registration Protocol.

6.2.1 Establishing a ERRP Connection

ERRP Nodes **MUST** use TCP/IP connections to carry ERRP messages. An ERRP Node **MUST** listen on TCP port 6069 for incoming connections that will be used to carry ERRP traffic.

The ERRP connection **MAY** be initiated by either of the ERRP Nodes (speaker or listener).

An ERRP Node **MUST** conform to the ERRP state machine specified in Section 6.2.7. An ERRP Node begins in the [Idle] state and goes through several state transitions before reaching the [Established] state. At that point the EQAM **SHOULD** advertise its operational resources to the ERM.

Typically, the EQAM (acting as a speaker) will be configured to establish the connection with ERM's. The ERM's, as listeners, will have no prior knowledge of the EQAM and therefore cannot establish the transport connection. KEEPALIVE messages are sent periodically to ensure adjacent peer ERRP Nodes are operational. Notification messages are sent in response to errors or special conditions. If a connection encounters an error condition, a Notification message is sent and the connection is closed.

6.2.2 Message Formats

ERRP Nodes MUST comply with the syntax and format of messages as outlined in the subsections below. Incomplete received ERRP messages MUST be ignored. An ERRP Node MUST not process a message until it is entirely received. An ERRP Node MUST NOT transmit ERRP messages that exceed 4096 octets. The recipient ERRP Node MUST be able to process ERRP messages up to 4096 octets in size. ERRP Nodes must transmit messages in "network order" so octets are transmitted most significant octet first and, within an octet, most significant bit first.

The smallest message that may be sent consists of an ERRP header without a data portion, or 3 octets.

6.2.2.1 Message Header

Every ERRP message begins 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:

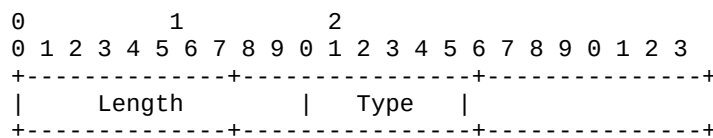


Figure 6–1 - DDRP Header Format

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 is mandatory. The value of the Length field is in the range $3 \leq \text{Length} \leq 4096$. The value may be further constrained by the message type. The stream of ERRP messages does not contain padding between messages.

Type

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

Table 6–1 - ERRP Message Types

Type	ERRP message
1	OPEN
2	UPDATE
3	NOTIFICATION
4	KEEPALIVE

6.2.2.2 OPEN Message

The first ERRP message sent by an ERRP 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 ERRP 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 ERRP header, the OPEN message contains the following fields:

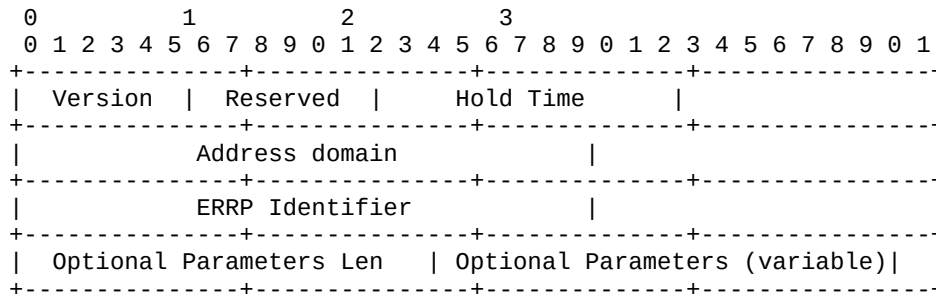


Figure 6-2 - ERRP OPEN Header

Version

This 1-octet unsigned integer indicates the protocol version of the message. The Version field is mandatory. The Version field contains the value 3.

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 is mandatory. Upon receipt of an OPEN message, an ERRP 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 is either 0 or greater than two; see Section 6.2.4.2 for error conditions. The configured Hold Time (if any) is either 0 or at least three seconds. 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.

Address domain

This 4-octet unsigned integer indicates the address domain number of the sender. The Address domain field is mandatory. Two ERRP Nodes **MUST** have the same address domain in order to establish an ERRP connection, unless one has the reserved address domain number of zero.

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

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

ERRP Identifier

This 4-octet unsigned integer indicates the ERRP Identifier of the sender. The ERRP Identifier field is mandatory. The value of this field uniquely identifies this ERRP Node within its address domain. An ERRP Node **MAY** set the value of its ERRP Identifier to an IPv4 address assigned to that ERRP Node. The value of the ERRP Identifier of an

ERRP Node is configured on the ERRP Node. The ERRP Node **MUST** use the same value for the ERRP Identifier field in all OPEN messages sent to other ERRP Nodes.

When comparing two ERRP identifiers, the ERRP Identifier is to be interpreted as a numerical 4-octet unsigned integer. It is recommended that the ERRP identifier is set to its IPv4 address as the default.

Optional Parameters Len

This 2-octet unsigned integer indicates the total length of the Optional Parameters field in octets. The Optional Parameters Len field is mandatory. If the value of this field is zero, the Optional Parameters field is absent. Note, this is not the same integer as 'Parameter Length' defined below.

Parameters

This field contains a list of parameters where each is encoded as a <Parameter Type, Parameter Length, Parameter Value> triplet as described in Figure 6–3. There are two mandatory parameters, Component Name and Streaming Zone, the rest are optional.

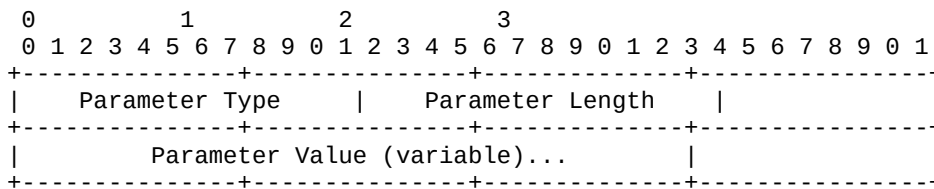


Figure 6–3 - Optional Parameter Encoding

Parameter Type

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

Parameter Length

This 2-octet unsigned integer contains the length of the Parameter Value field in octets. The Parameter Length field is mandatory.

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.

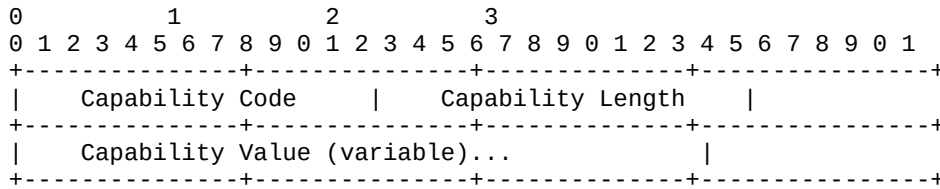
6.2.2.2.1 Open Message Parameters

6.2.2.2.1.1 Capability Information

If present, the Capability Information parameter identifies its presence by setting the value of the Parameter Type field to 1.

The Capability Information parameter is optional and is used by an ERRP Node to indicate to its peer ERRP Node the capabilities that it supports. Capability negotiation is specified in Section 6.2.6.

A valid Capability Information parameter contains one or more triplets <Capability Code, Capability Length, Capability Value>, where each triplet 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 is mandatory.

Capability Length

This 2-octet unsigned integer contains the length of the Capability Value field in octets. The Capability Length field is mandatory.

Capability Value

This is a variable-length field that is interpreted according to the value of the Capability Code 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 is one of the values identified in Table 6-2.

Table 6-2 - Capability Codes

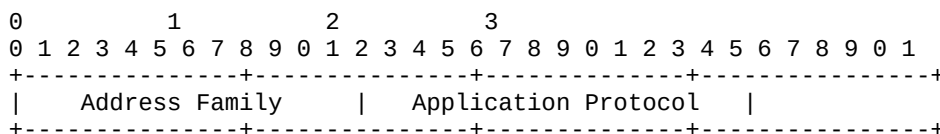
Capability Code	Capability
1	Route Types Supported
2	Send Receive
32768	ERRP Version

This specification reserves Capability Codes 32769-65535 for vendor-specific applications (these are the codes with the first bit of the code value equal to 1). This specification reserves value 0. Capability Codes (other than those reserved for vendor specific use) are controlled by IANA.

6.2.2.2.1.1.1 Route Types Supported

The Route Types Supported Capability Code lists the route types supported in this peering session by the transmitting ERRP Node. An ERRP Node speaker **MUST NOT** use route types that are not supported by its ERRP Node peer in any particular peering session. If the route types supported by an ERRP Node peer are not satisfactory, an ERRP Node speaker **SHOULD** terminate the peering session.

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. The Address Family field identifies the address family of the resource within the ReachableRoutes attribute. The Address Family field is mandatory. The Application Protocol field identifies the application for which the resource may be used. The Application Protocol field is mandatory.

For example, a Route Type for ERRP 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.

6.2.2.2.1.1.2 Send Receive

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

In Send Only mode, an ERRP Node speaker transmits UPDATE messages to its ERRP Node peer, but the ERRP Node peer **MUST NOT** transmit UPDATE messages to that ERRP Node speaker. If an ERRP Node speaker in Send Only mode receives an UPDATE message from its ERRP Node peer, it **MUST** discard that message, but no further action should be taken.

In Receive Only mode, the ERRP Node listener acts as a passive TRIP listener. In Receive Only mode, an ERRP Node **MUST NOT** transmit an UPDATE message though it receives and processes UPDATE messages from its ERRP Node peer. In Receive Only mode, an ERRP Node still transmits OPEN, KEEPALIVE, and NOTIFICATION messages. In Send Only mode, an ERRP Node still receives OPEN, KEEPALIVE, and NOTIFICATION messages. The Send Receive Capability field is a 4-octet unsigned integer. The value of the Send Receive Capability field is 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 an ERRP Node discovers while processing an OPEN message that both it and its remote ERRP Node are in Send Only mode or in Receive Only mode, it **MUST** send a NOTIFICATION message to the remote ERRP Node to close the session. The error code in this NOTIFICATION message is set to "Capability Mismatch" (see the ERRP error handling Section 6.2.4).

An ERRP Node **MUST** send the same value of Send Receive Capability to all remote ERRP Nodes. An ERM **SHOULD** advertise Send Receive mode if it supports both VOD and DOCSIS applications. An EQAM **MUST** advertise Send Only mode. ERRP component send receive capability is summarized in

Table 6–4.

Table 6–4 - ERRP Component Send Receive Capability

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

6.2.2.2.1.1.3 ERRP Version

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

The value of the ERRP Version Capability field is 1.

6.2.2.2.1.2 Streaming Zone

StreamingZone Name is a mandatory parameter when supporting video applications. It identifies its presence by setting the value of the Parameter Type field to 2. The capability is optional when signaling DOCSIS only resources. The value is to be set to the string that represents the StreamingZone Name i.e., <region>.<local name>. The length should be set to the length of the string and as such is variable. The Streaming Zone name represents the Streaming Zone within which the component operates.

The characters comprising the string are in the set within TEXT defined in section 15.1 of [RFC 2326]. Implementations must support minimum string lengths of 64; however, the composition of the string used is defined by implementation agreements specified by the service provider.

6.2.2.2.1.3 Component Name

Component Name is a mandatory parameter when supporting video applications. It identifies its presence by setting the value of the Parameter Type field to 3. The capability is optional when signaling DOCSIS only resources. The value is to be set to the string that represents the Component Name i.e., <region>.<local name>. The length should be set to the length of the string and as such is variable. The Component Name is the name of the component for which the data in the update message applies. For example, the Component Name would be the EQAM name in the case of the EQAM Registration Interface.

The characters comprising the string are in the set within TEXT defined in section 15.1 of [RFC 2326]. Implementations must support minimum string lengths of 64; however, the composition of the string used is defined by implementation agreements specified by the service provider.

6.2.2.2.1.4 Vendor Specific String

Vendor Specific String is an optional parameter. It identifies its presence by setting the value of the Parameter Type field to 4. The length should be set to the length of the string and as such is variable. The value could be anything the vendor of the component wishes to use to pass "innovation" hints to resource managers. Examples could include vendor and model name and number or enumerated list of functions.

The characters comprising the string are in the set within TEXT defined in section 15.1 of [RFC 2326]. Implementations must support minimum string lengths of 64; however, the composition of the string used is defined by implementation agreements specified by the service provider.

6.2.2.3 UPDATE Message Format

UPDATE messages are used to transfer information about resources between ERRP 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. A single UPDATE message may simultaneously advertise and withdraw ERRP resources.

In addition to the ERRP header, the ERRP UPDATE may contain a list of Routing Attributes, which are formatted as shown in Figure 6–6. Routing Attributes are contiguous (no padding) in the message.

```
+-----+-----+-----+-----+
| First Route Attribute | Second Route Attribute | ...
+-----+-----+-----+-----+
```

Figure 6–6 - ERRP 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

A variable length sequence of Routing Attributes is present in every UPDATE message. Each Routing Attribute is formatted as shown in Figure 6–7 and Figure 6–8.

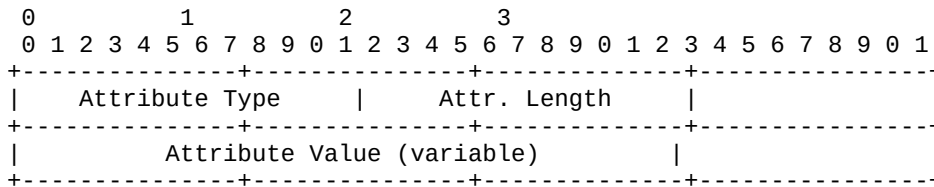


Figure 6–7 - Routing Attribute Format

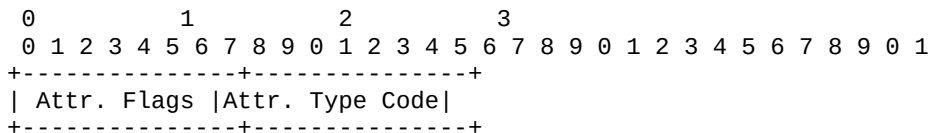


Figure 6–8 - Attribute Type Format

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 is mandatory. The Attribute Flags sub-field is mandatory. The Attribute Type Code sub-field is mandatory.

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 are in increasing numerical order of the Attribute Type Code field. A particular value of this field is not to 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.

ERRP uses TRIP as base protocol. ERRP Nodes MUST conform to [RFC 3219] in the use of the Attribute Type Code. [RFC 3219] reserves the value of zero and defines attributes with codes between 1 and 11 for this field. [RFC 3219] allows new attributes to be defined using vendor-specific codes 224 to 255 (these are the codes with the first three bits of the code equal to 1). Attribute Type Codes (other than those reserved for vendor specific use) are controlled by IANA. ERRP supports some of the attributes (ReachableRoutes, WithdrawnRoutes, and NextHopServer) defined in [RFC 3219]. ERRP defines more ERRP specific attributes using Attributes Type Codes between 224 and 255. These ERRP-supported attributes are further specified in Section 6.2.3.

Attributes not supported by ERRP MUST NOT be used by ERRP Nodes.

Attribute Length

The Attribute Length is a two-octet unsigned integer that contains 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

The high-order bit (bit 0) of the Attribute Flags octet is the Well-Known Bit. It defines whether the attribute is not well-known (if set to 1) or well-known (if set to 0). Messages received with bit 0 unset are processed. Messages received with bit 0 set may be processed.

Bits 1 through 7 are zero on transmit. Bits 1 through 7 are ignored on receipt.

6.2.2.4 KEEPALIVE Message Format

ERRP does not depend on any lower-level, keepalive mechanism to determine whether remote ERRP Nodes remain reachable. Instead, KEEPALIVE messages are exchanged sufficiently often to ensure that the Hold Timer does not expire. The maximum time between the transmissions of successive KEEPALIVE messages by an ERRP Node SHOULD NOT be more than one third of the negotiated Hold Time. An ERRP Node MUST NOT send KEEPALIVE messages more than once every 3 seconds.

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

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 by an ERRP Node except in response to detection of an error. The TCP connection carrying the ERRP traffic are torn down immediately following transmission or reception of a NOTIFICATION message.

In addition to the fixed-size ERRP 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 ERRP error handling (Section 6.2.4). The error report has the following format:

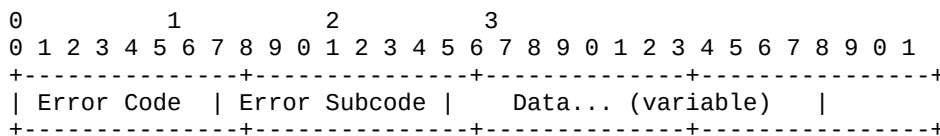


Figure 6-9 - ERRP NOTIFICATION Format

Error Code

This 1-octet unsigned integer indicates the type of the NOTIFICATION. This field is mandatory.

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

Table 6–6 - ERRP 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 is mandatory. Each Error Code may have one or more Error Subcodes associated with it. If no appropriate Error Subcode is defined, then a zero (Unspecific) value is used for the Error Subcode field.

In the case of a Message Header Error, the value of the Error Subcode field is 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 is 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 ERRP Identifier
4	Unsupported Optional Parameter

Error Subcode	Meaning
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 is 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 is mandatory if the Data field format is specified for the particular values of the Error Code and Error Subcode fields (see the ERRP error handling Section 6.2.4).

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

$$\text{Data Length} = \text{Message Length} - 5$$

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

6.2.3 ERRP Attributes

This section specifies the syntax and semantics of each ERRP UPDATE attribute listed in Table 6–10. ERRP Nodes using Attributes in message MUST use the syntax and format of the attributes listed in the following subsections.

Table 6–10 - ERRP Attribute Type Codes

Advertised Attributes	Type Code	Supported by ERM/EQAM
Withdrawn Route	1	Must
Reachable Route	2	Must
NextHopServer	3	Must
QAM Names	232	Must
CAS Capability	233	May

Advertised Attributes	Type Code	Supported by ERM/EQAM
Total Bandwidth	234	Must
Available Bandwidth	235	May
Cost	236	Should
Edge Input	237	Must
QAM Channel Configuration	238	Must
UDP Map	239	Must
Service Status	241	Should
Max MPEG Flows	242	May
NextHopAlternate	243	May
Output Port/Port ID	244	Must
Fibre Node	245	May
QAM Capability	247	Must
Input Map	249	May

There are no mandatory attributes in an ERRP message. However, there are conditional mandatory attributes. A conditional mandatory attribute is an attribute that is included in an UPDATE message if another attribute is included in that message. The EQAM implements the "Support" column of Table 6–10 as follows:

- The EQAM MUST support the 'Must' ERRP Attributes from Table 6–10.
- The EQAM SHOULD support the 'Should' ERRP Attributes from Table 6–10.
- The EQAM MAY support the 'May' ERRP Attributes from Table 6–10.

The two base attributes in ERRP are WithdrawnRoutes and ReachableRoutes. Their presence in an UPDATE message is entirely optional and independent of any other attributes. Attributes appear in the UPDATE message in increasing order of the Attribute Type Code.

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 lists the individual routes sequentially and with no padding, as shown in Figure 6–10.

```
+-----+-----+...
| WithdrawnRoute1... | WithdrawnRoute2... |...
+-----+-----+...
```

Figure 6–10 - WithdrawRoutes Format

Conditional Mandatory: False

Required Flags: Well-known

Attribute Type Code: 1

6.2.3.1.1.1 Generic ERRP Route Format

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

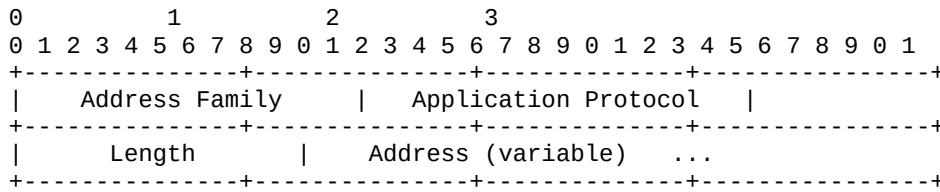


Figure 6–11 - Route Format for WithdrawRoutes and ReachableRoutes

Address Family

The Address Family field gives the type of address for the resource (route). The Address Family field is mandatory. The Address Family field contains the value given in Table 6–11.

Table 6–11 - Values for Address Family

Value of Address Family field	Meaning
32769	Video Name

The address family of Video Name in ERRP will contain a specific resource name such as QAM Group Name.

This specification reserves address family code 0. According to [RFC 3219], address family codes 1 – 32767 are administered by IANA. This specification reserves address family codes 32768, and 32770-65535 for vendor-specific applications (these are the codes with the first bit of the code value equal to 1. The values from Table 6–11 are purposefully in the vendor-specific range and can not be reused by a vendor in the context of using this specification.

Application Protocol

The Application Protocol field identifies the protocol for which the resource database is maintained. The Application Protocol field is mandatory. The Application Protocol field contains the value given in Table 6–12:

Table 6–12 - Application Protocols Supported in ERRP

Component	Application Protocol Code	Session Layer Interface	Protocol
Edge QAM	32766	pre-provisioned (static portmap)	None
	32768	Session Parameter Only (dynamic session)	RTSP
	32770	Session Parameter and provisioning (dynamic session)	RTSP

This specification reserves application protocol code 0. According to [RFC 3219], application protocol codes 1 – 32767 are administered by IANA. This specification reserves application protocol codes 32771-65535 for vendor specific applications. The values from Table 6–12 are purposefully in the vendor-specific range and can not be reused by a vendor in the context of using this specification.

Length

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