Data-Over-Cable Service Interface Specifications
Resource Management Interface

Resource Management Architecture and HTTP Specification

CM-SP-RMI-HTTP-I01-140520

ISSUED

DISCLAIMER

This document is furnished on an "AS IS" basis and neither CableLabs nor its members provides any representation or warranty, express or implied, regarding the accuracy, completeness, noninfringement, or fitness for a particular purpose of this document, or any document referenced herein. Any use or reliance on the information or opinion in this document is at the risk of the user, and CableLabs and its members shall not be liable for any damage or injury incurred by any person arising out of the completeness, accuracy, or utility of any information or opinion contained in the document.

CableLabs reserves the right to revise this document for any reason including, but not limited to, changes in laws, regulations, or standards promulgated by various entities, technology advances, or changes in equipment design, manufacturing techniques, or operating procedures described, or referred to, herein.

This document is not to be construed to suggest that any affiliated company modify or change any of its products or procedures, nor does this document represent a commitment by CableLabs or any of its members to purchase any product whether or not it meets the characteristics described in the document. Unless granted in a separate written agreement from CableLabs, nothing contained herein shall be construed to confer any license or right to any intellectual property. This document is not to be construed as an endorsement of any product or company or as the adoption or promulgation of any guidelines, standards, or recommendations.

Document Status Sheet

Document Control Number	CM-SP-RMI-HTTP-I01-140520			
Document Title	Resource Management Architecture and HTTP Specification			
Revision History	I01 – 05/20/2014			
Date	May 20, 2014			
Status	~~Work in Progress~~	~~Draft~~	**Issued**	~~Closed~~
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 generally public document that has undergone Member and Technology Supplier review, cross-vendor interoperability, and is for Certification testing if applicable. Issued Specifications are subject to the Engineering Change Process.
Closed	A static document, reviewed, tested, validated, and closed to further engineering change requests to the specification through CableLabs.

Trademarks

Contents

Figures

Tables

1 SCOPE

1.1 Overview

This specification is part of the DOCSIS® family of specifications developed by Cable Television Laboratories, Inc. (CableLabs). The Resource Management Interface (RMI) architecture is an evolution of the Edge Resource Manager Interface (ERMI) architecture which is intended to improve scalability, performance, and service velocity. The architecture adopts web services as the standard communications mechanism between components.

1.2 Purpose

This document specifies the usage of the HTTP protocol in the RMI system. This specification includes common elements, attributes, and data types. The document also provides a common list of the reason codes, response codes, and event codes used in RMI.

1.3 Scope

This specification details the usage of HTTP and web services. The information in this specification applies to all of the web service interfaces defined for RMI:

- Service Discovery and Registration (SDR) interface – A SDR-ERM interface between an Auto Discovery Server and an Edge Resource Manager (ERM), and a SDR-EDGE interface between an ERM and an Edge Device. This interface is used to provide resource status information to the ERM and Auto Discovery Server in order to effectively administer available Quadrature amplitude modulation (QAM) resources.
- Session Manager – ERM (SM-ERM) interface: This interface allows a Session Manager/M-CMTS Core to make resource requests on Edge Devices from the ERM that manages those devices.
- ERM – Edge Device (ERM-EDGE) interface: This interface allows an ERM to provision QAMs within an Edge Device and can be used to request that the Edge Device inspect and/or inject session-specific data.

The interfaces specified in the RMI documentation set are shown in Figure 1.

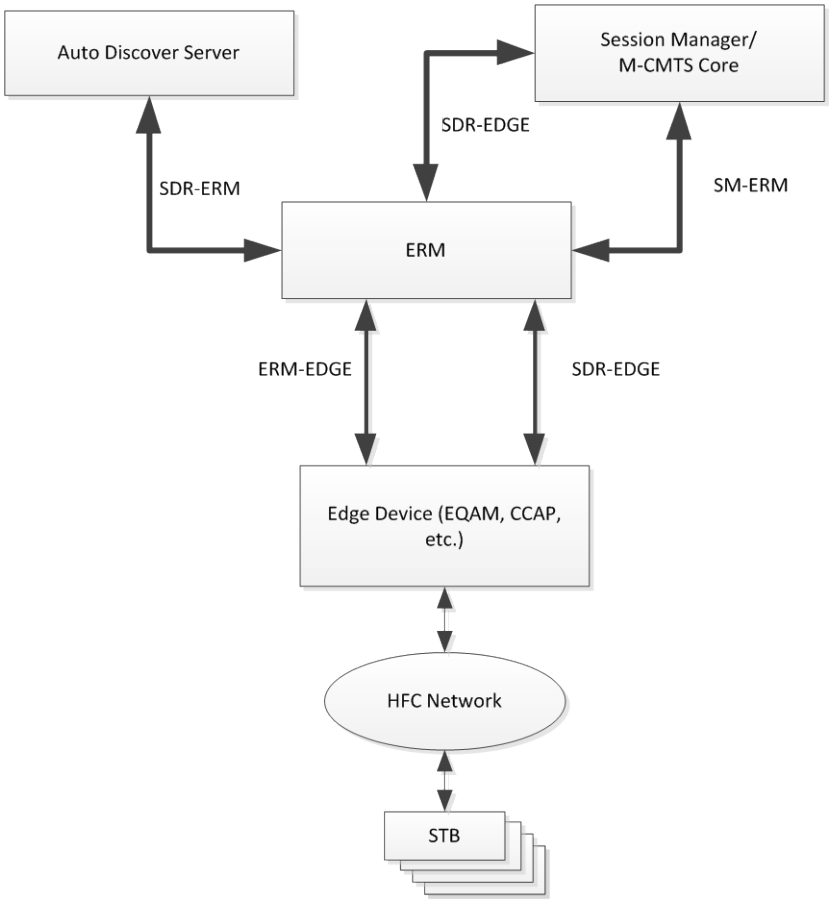

Figure 1 – Resource Management Interface Reference Architecture

1.4 Requirements

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

"MUST"	This word 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 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 means that this item is truly optional. For example, one vendor may choose to include the item because a particular marketplace requires it or because it enhances the product; another vendor may omit the same item.

This document defines many features and parameters, and a valid range for each parameter is usually specified. Equipment (CM and CMTS) requirements are always explicitly stated. Equipment must comply with all mandatory (MUST and MUST NOT) requirements to be considered compliant with this specification. Support of non-mandatory features and parameter values is optional.

1.5 Organization of Document

This specification is organized into the following sections:

Sections 1 – 4 contain the introduction, references, terms and abbreviations.

Section 5 provides an overview of the overall RMI architecture and scope of the individual interfaces.

Section 6 specifies usage of the HTTP Protocol.

Section 7 specifies common XML elements.

Section 8 specifies common response codes, event notification codes, and teardown reason codes used within the RMI.

2 REFERENCES

2.1 Normative References

There are no normative references in this document.

2.2 Informative References

This specification uses the following informative references.

[CCCP]	OpenCable CableCARD™ Copy Protection 2.0 Specification, OC-SP-CCCP2.0-I13-130418, April 18, 2013, Cable Television Laboratories, Inc.
[ITU J.83]	ITU-T Recommendation J.83 (12/07), Digital multi-programme systems for television, sound and data services for cable distribution, http://www.itu.int/rec/T-REC-J.83-200712-I
[RFC 2616]	IETF RFC 2616, Hypertext Transfer Protocol -- HTTP/1.1, June 1999.
[RMI-ERM-EDGE]	Edge Resource Manager - Edge Device Interface Specification, CM-SP-RMI-ERM-EDGE-I01-140520, May 20, 2014, Cable Television Laboratories, Inc.
[RMI-SDR]	Service Discovery and Registration Specification, CM-SP-RMI-SDR-I01-140520, May 20, 2014, Cable Television Laboratories, Inc.
[RMI-SM-ERM]	Session Manager – Edge Resource Manager Interface Specification, CM-SP-RMI-SM-ERM-I01-140520, May 20, 2014, Cable Television Laboratories, Inc.

2.3 Reference Acquisition

- Cable Television Laboratories, Inc., 858 Coal Creek Circle, Louisville, CO 80027; Phone +1-303-661-9100, Fax +1-303-661-9199; http://www.cablelabs.com/
- Internet Engineering Task Force (IETF): http://www.ietf.org/
- ITU-T Recommendations: http://www.itu.int/ITU-T/publications/recs.html

3 TERMS AND DEFINITIONS

This specification uses the following terms:

Advertisement	Messaging between an Edge Device and Edge Resource Manager that provides details of the QAM resources available on that device.
Auto Discovery Server	A network element that determines the service group of a client based on the narrowcast QAMs used to deliver Video on Demand (VOD) or Switched Digital Video (SDV) content to the client.
Edge Device	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.
Edge Input Group	A set of Edge Device 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 Edge Device via the protocols defined in this specification.
Input Map	A list of Edge Device input interfaces that reach a particular QAM channel within the internal architecture of an Edge Device.
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 reaches a common set of set-top boxes.
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.
Session Manager	A network element that manages the life cycle of video service sessions delivered to a subscriber

4 ABBREVIATIONS AND ACRONYMS

This specification uses the following terms:

3DES	Triple Data Encryption Standard
AC	Access Criteria
AES	Advanced Encryption Standard
APS	Analog Protection Signaling
CA	Conditional Access
CAS	Conditional Access System
CAT	Conditional Access Table
CCI	Copy Control Information
CPE	Customer Premises Equipment
CSA	Common Scrambling Algorithm
DES	Data Encryption Standard
DNS	Domain Name Server
DVB	Digital Video Broadcasting
EAS	Emergency Alert Service
ECM	Entitlement Control Message
ECMG	Entitlement Control Message Generator
EIG	Edge Input Group
EMM	Entitlement Management Message
ERM	Edge Resource Manager
ERMI	Edge Resource Manager Interface
eSTB	Embedded Set-Top Box
HFC	Hybrid Fiber-Coax
HTTP	HyperText Transfer Protocol
IGMP	Internet Group Management Protocol
IP	Internet Protocol
M-CMTS	Modular Cable Modem Termination System
MPTS	Multi-Program Transport Stream
NIT	Network Information Table
NPT	Normal Play Time
PAT	Program Association Table
PID	Packet Identifier
PSIP	Program and System Information Protocol
QAM	Quadrature amplitude modulation
RMI	Resource Management Interface
SDR	Service Discovery and Registration

SDV	Switched Digital Video
SM	Session Manager
SOP	Streaming Output Port
SPTS	Single-Program Transport Stream
SSM	Source-Specific Multicast
TCP	Transmission Control Protocol
TSID	Transport Stream ID
TTL	Time to Live
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
UTC	Universal Time Code
VOD	Video on Demand
XML	Extensible Markup Language

5 OVERVIEW AND THEORY OF OPERATIONS

An Edge Device receives packets of digital video or data from the Internet Protocol (IP) network. It re-packetizes the video or data and delivers to the Hybrid Fiber-Coax (HFC) network using QAM outputs. An ERM manages QAM resources on Edge Devices. A Session Manager works with the ERM to manage video sessions on Edge Devices.

RMI defines a mechanism for an ERM to discover Edge Device resources via the SDR interface. RMI also defines resource allocation interfaces for a Session Manager to direct an ERM to allocate QAM resources from an Edge Device.

5.1 Edge Architecture Overview

An Edge Device 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 SDR 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 Edge Device. Input interfaces advertised may be physical interface addresses or virtual interface addresses of the Edge Device. An Edge Device 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 Edge Device to the ERM via an "Input Map" for each QAM channel. The ERM might need to further subdivide the inputs of an Edge Device (or of an Input Map in the case of partial internal connectivity) based on network connectivity external to the Edge Device. A set of Edge Device input interfaces that have equivalent connectivity in the operator's Ethernet network (i.e., source to Edge Device) are called an Edge Input Group (EIG).

An ERM manages the Edge Device output at the granularity of a QAM channel, though several QAM channels can be physically connected to a single RF port on the Edge Device. QAM channels are grouped to QAM groups. A 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).

Figure 2 illustrates Edge Devices, their input and output relationships, and these network topology concepts. In the diagram, there are two Edge Devices. The first Edge Device 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 Edge Device 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 Edge Device 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 Edge Device 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 Edge Device, Q5 and Q6 are only reachable by G5 and G6, while Q7 and Q8 are only reachable by G7 and G8. This Edge Device would advertise these input maps through the SDR interface. 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 subscribers, they are considered to be a single QAM group (QG3). Also note that even though the Ethernet network topology of EIG1 and EIG2 are the same, they are in separate input groups because they are used by separate Edge Devices.

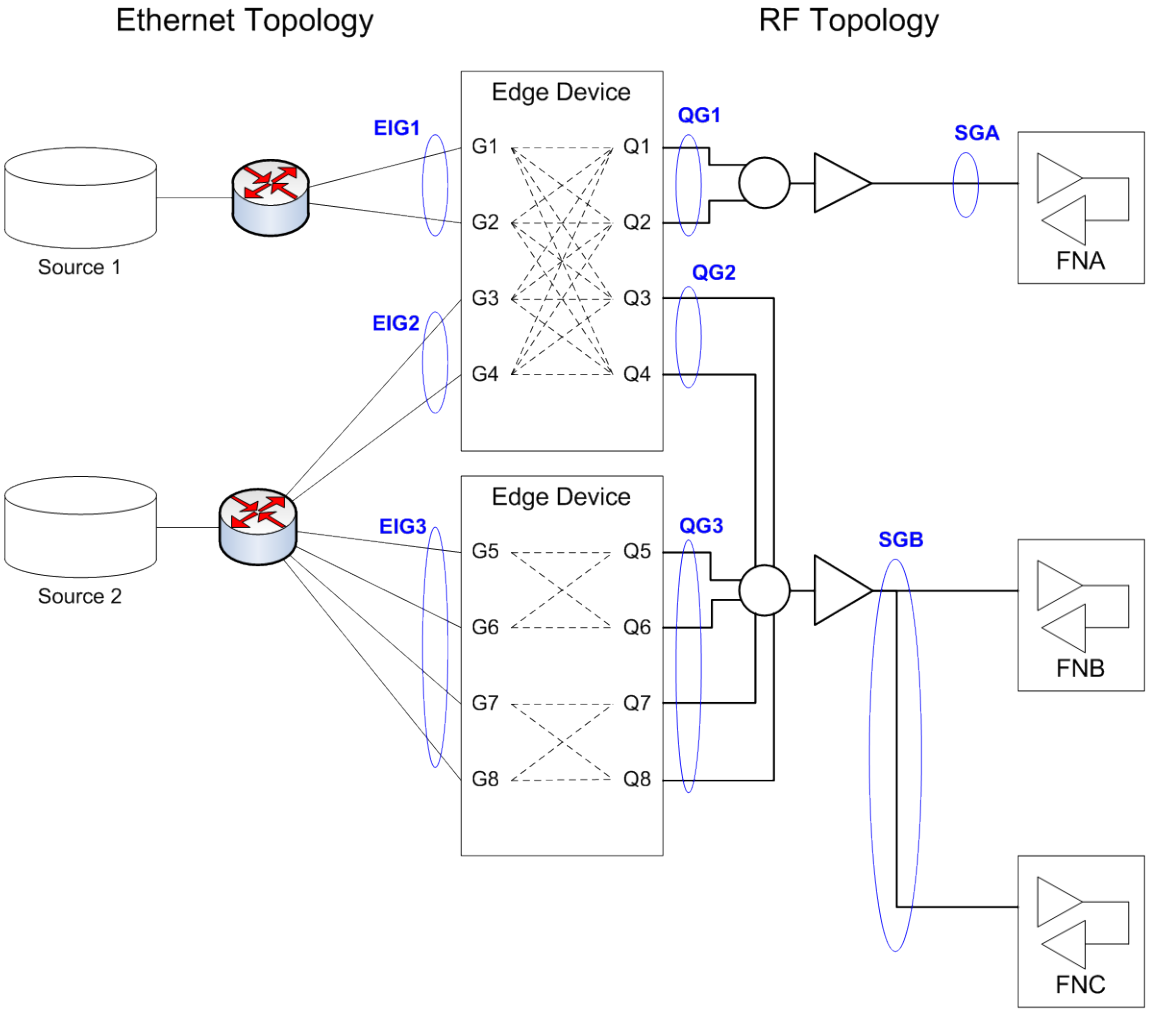

Figure 2 – RF Topology

RMI 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 these specifications, 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 2, 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 Edge Devices. QAM channel to QAM group mapping and QAM group to fiber node mapping are configured at Edge Devices. An ERM learns the RF topology information via the SDR interface. When a Session Manager requests edge resources from an ERM, the service group QAM list is communicated between the Session Manager and the ERM.

Service groups are not directly managed by the Edge Device and the ERM. Service groups may be managed directly by higher level entities such as a video Session Manager.

5.2 Service Discovery and Registration Interface

The Service Discovery and Registration (SDR) interface allows Edge Devices to register their QAM channel resources with an ERM (SDR-EDGE) and an ERM to register its resources with an Auto Discovery Server (SDR-ERM). The SDR interface is specified in [RMI-SDR].

5.2.1 Goals, Scope and Constraints

The SDR interface was designed to meet the following goals:

- Enable Edge Devices to register their resources with an ERM,
- Enable ERMs to register their managed resources with an Auto Discovery Server,
- Help the ERM to detect when failures occur in resources that it manages.

5.2.1.1 Registering QAM Channels

Each ERM is responsible for managing the resources of one or more Edge Devices. In order for an ERM to manage an Edge Device's resources, it must first obtain an inventory of the resources associated with that Edge Device. It must also obtain IP and/or RF addressing information associated with the Edge Device and those resources in order to determine how to communicate with them. For example, an Edge Device 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 addition the Auto Discovery Server tracks the resources available to multiple ERMs. The Auto Discovery Server collects QAM inventory information from the ERMs. This inventory specifies the QAM configuration information for each QAM managed by an ERM allowing the Auto Discovery Server to determine the QAMs available for a given service group.

Edge Devices use the SDR interface to advertise available resources to an ERM. The ERM uses this information to populate a database of available resources on the Auto Discovery Server.

5.2.2 Overall Architecture

The SDR interface allows an Edge Device 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 Edge Devices. The ERM may use this data to formulate responses to incoming requests for resources.

Figure 3 shows the SDR interface component architecture.

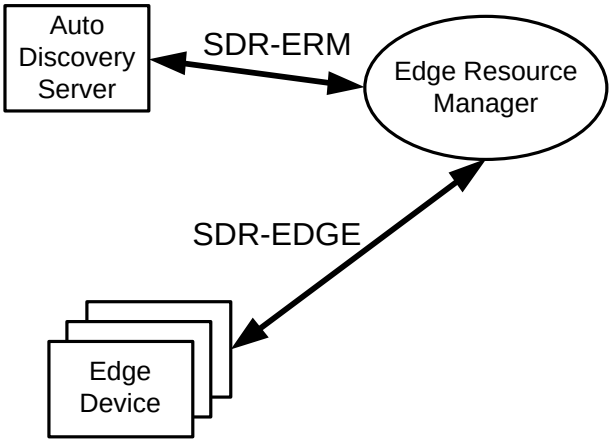

Figure 3 – SDR Interface and Components

An ERM normally manages multiple Edge Devices, as shown in Figure 3. An Edge Device requires an ERM IP address in order to contact the ERM. An Edge Device may use a Domain Name Server (DNS) query to obtain an ERM IP address. An Edge Device 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.

An Auto Discovery Server normally manages multiple ERMs. The Auto Discovery Server keeps track of the resources available to all of the ERMs for which it is responsible.

Messages sent using the HTTP protocol are carried over an IP connection and may be initiated by the Auto Discovery Server, the ERM, or the Edge Device.

Because the device control interface for an Edge Device is HTTP, the Edge Device registers a management URL, in addition to the IP address. This management URL is used to establish an HTTP connection between the ERM and the Edge Device. This allows the ERM to send device control messages to the Edge Device to request QAM channel resources.

In normal use, a Session Manager requests a QAM channel resource from an ERM, (i.e., during the process of setting up a session over the SM-ERM interface, describe in Section 5.3). The resources previously advertised by Edge Devices (over SDR-EDGE) are used by the ERM to select a suitable QAM channel resource to meet the requirements of an incoming request. The ERM directs the Edge Device (over ERM-EDGE) to setup the mapping of the services, and then the ERM returns addressing information for that resource (over SM-ERM) to the requesting Session Manager.

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

5.3 Resource Allocation Signaling

5.3.1 Resource Allocation Components and Interface

Figure 4 shows the components involved in the resource allocation interfaces.

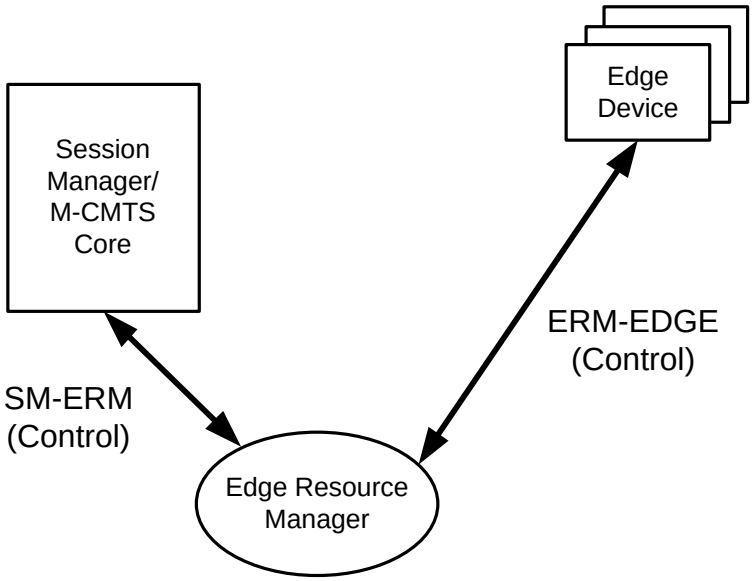

Figure 4 – Resource Allocation Interfaces and Components

The SM-ERM interface is specified in [RMI-SM-ERM]; the ERM-EDGE interface is specified in [RMI-ERM-EDGE]

The Video Session Manager or M-CMTS Core initiates a QAM resource transaction with the ERM when it requests or releases a QAM channel resource. When a Session Manager requests a VOD session to be setup, for example, the Session Manager provides the ERM with details of the session, bandwidth, and QAM channel. 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 Edge Device is a device which has a pool of QAM channels that may be allocated to DOCSIS, video, or both. The capabilities of each individual QAM channel are advertised to the ERM when the Edge Device advertises that particular QAM channel through the SDR interface.

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 Edge Device, QAM channels are physically present on external RF ports. A single RF port is 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 with the same physical-layer parameters. 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 Edge Device should be able to control the configuration of a single QAM channel without affecting the configuration of any other QAM channels.

The RMI specification set specifies two resource-allocation interfaces. ERM-EDGE is an interface between an ERM and an Edge Device and is used to allocate QAM channel resources selected by the ERM. SM-ERM is an interface between a Session Manager/M-CMTS Core and an ERM and is used to request and return QAM channel resources.

5.3.2 Signaling Protocol

The protocol used for the resource allocation interfaces (SM-ERM and ERM-EDGE) is HTTP. The protocol is intended to control multiple, simultaneous, asynchronous data delivery sessions, to provide a means for choosing delivery channels, and to provide a way to choose among multiple delivery mechanisms.

The Session Manager initiates an HTTP session by sending a SetupSession message containing a request for resources. The ERM in turn initiates an HTTP session with the Edge Device by sending a SetupSession message to the Edge Device to allocate the resources for the session. The Edge Device replies after identifying a suitable resource that meets the ERM's request. The resources are then sent by the ERM to the Session Manager in response to the original request. In SM-ERM, the Session Manager is an HTTP client and the ERM is an HTTP server. In ERM-EDGE, the ERM is an HTTP client and the Edge Device is an HTTP server.

HTTP allows considerable variations in the capabilities supported by conformant implementations. The RMI specification set indicates the subset of interface requirements that must be supported by the HTTP components in the RMI resource allocation interfaces (SM-ERM and ERM-EDGE).

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 specification set. 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.

6 HTTP PROTOCOL

6.1 Connection

In most cases, any device that interfaces with another device can initiate an HTTP request. Therefore, the HTTP request transmitter is considered the client and the HTTP request receiver is considered the server. The client always initiates the connection. Once the connection is established, that connection can be used for multiple requests, as discussed later in this specification. However, if a client initiates a request and the server has an HTTP request that it needs to transmit to the device (in effect, making the server the client in the context of the new message); a new connection MUST be established for the new HTTP request message.

Communication between the client and server MUST use the HTTP/1.1 protocol, as specified in [RFC 2616]. In HTTP/1.1, connections are persistent by default. The connection remains open unless either the client or server explicitly indicates the connection should close. The HTTP response includes the header *Connection: close* in order to indicate that the connection should be closed.

During periods when no messages are being exchanged, the client or server MAY close the connection to conserve resources. If the client does not transmit a message for 60 seconds, then the client MUST close the connection.

6.1.1 Connection Security

For communications between client and server components in the RMI system, it is expected that secure communication channels will be used. The methods and protocols used to secure the communications between entities are outside of the scope of this specification.

6.2 Request Messages

6.2.1 Use of HTTP Methods

A client MUST only use the GET or POST methods for HTTP requests. The client MUST use the GET method when retrieving existing information from the server. For example, when the client is requesting the state of a session a GET method would be used. The client MUST use the POST method to execute a task or action on the server other than retrieving existing information. For example, when the client is setting up or tearing down a session a POST method would be used. The client MAY use a POST method if the request includes many parameters or complex parameters. In that case, the parameters can be specified in Extensible Markup Language (XML) included in the POST operation. For example, the client specifies a query that includes a list of parameters.

The client MUST NOT use the PUT and DELETE HTTP methods.

6.2.2 URI Format

To allow for proxies to be inserted in any communication, the client MUST use the absolute format of the Uniform Resource Identifier (URI). The path of the URI identifies the application name and the method name. The format of the path is shown below.

```
path = "/" app_abbreviation "/" operation
```

The *app_abbreviation* is the abbreviated application name. Including this in the URI allows multiple applications to be sourced from the same server. The client MUST use the application abbreviations listed in Table 1 instead of component or application names.

Table 1 – Application Abbreviations

Component or Application Name	app_abbreviation
Auto Discovery Server	ds
Edge Device	edge
Edge Resource Manager	erm
Session Manager	sm

The *operation* indicates the task to perform. For Setup and Teardown POST operations, the client MUST use the "Setup" method and the "Teardown" method in the *operation*. This allows the method names to remain consistent with the RTSP methods. For Setup and Teardown operations, the client MUST use the HTTP POST method. The *app_abbreviation* indicates the interface on which these operations function. For example, "`/erm/Setup`" indicates the Session Manager is setting up a session on the ERM.

If a request is simply getting or setting a value, and it uses the GET or POST HTTP method, then the client SHOULD omit the operation name and simply indicate the data or information being retrieved or set. That is, the operation words "Get" and "Set" do not need to be included in the operation.

For example, consider the path required to retrieve streaming server configuration data (StreamingServerConfig). The ERM can issue an HTTP GET to retrieve the configuration details from an Edge Device it manages, or the Edge Device may issue an HTTP POST to send this configuration data to the ERM. In either case, the operation is simply the object name "EdgeDeviceConfig". Notice that it does not include "Get" or "Set".

The full HTTP headers for the EdgeDeviceConfig method are:

```
POST http://Device/erm/EdgeDeviceConfig HTTP/1.1
GET http://Device/edge/EdgeDeviceConfig HTTP/1.1
```

When the client uses the GET method, the URI MAY include an optional list of parameters, formatted according to the HTTP protocol as specified in [RFC 2616]. Note that data in the parameter list MUST be URL encoded.

The POST method does not include any parameters in the URI; parameters are specified in an XML Body.

6.2.3 HTTP Version

The client and server MUST implement HTTP version 1.1, as specified in [RFC 2616].

6.2.4 HTTP Request Headers

Compliance with HTTP 1.1 requires the use of the *Host* header. In addition, when a message body is present, the *Content-Length* header is also required. In this specification set, "`...`" is used as the value of this header, although an accurate count of characters in the message is required by HTTP 1.1.

The client MUST include the *User-Agent* header in all requests to aid in troubleshooting. The value of this header SHOULD describe the client application along with the version of the application.

When there is no message body, the client MUST use the *Accept* header. Use of the *Accept* header simplifies any future extension of media encodings for an interface.

When a message body is included in the request, the client MUST use the *Content-Type* header. When a message body is included, the client MAY use the *Accept* header.

The client MAY include other headers. The server MAY ignore non-required headers without changing the expected behavior.

An example HTTP request header is shown below (body not shown):

```
POST http://Device//erm/EdgeDeviceConfig/edge01.philly  HTTP/1.1
Host: Device
User-Agent: Example Streaming Server
Content-Type: application/xml
Content-Length: 1039
```

6.2.5 Message Body – XML

The message body for an HTTP POST request is XML. The root element of the XML MUST match the operation specified in the URI.

Each interface has its own namespace. This allows a "SetupSession" request to the ERM to have a different body than a "SetupSession" request to the Edge Device. The namespace used in the XML is shown below:

```
xmlns="urn:com:cablelabs:rmi:<interface>:2010:12:03"
```

where <interface> is the interface ID. For example, the namespace used for the ERM to Edge Device interface (ERM-EDGE) is:

```
xmlns="urn:com:cablelabs:rmi:erm-edge:2010:12:03"
```

There is no message body for an HTTP GET message.

6.2.6 GET and POST Request Message Examples

GET Request Message Example:

```
GET /edge/SessionList?SessionGroup="ERM1.GROUP4" HTTP/1.1
Host: Device
User-Agent: Example Device
Accept: application/xml
```

POST Request Message Example

```
POST /ds/QamInventory HTTP/1.1
Content-Type: application/xml
Content-Length: ...
<QamInventory
      xmlns="urn:com:cablelabs:rmi:sdr-erm"
      protocolVersion="1"
      componentName="erm02.philly"
      modelName="ABC Enterprises:ERM-1000"
      inventoryType="FULL" >
      <StreamingZone streamingZoneName="Philly">
            <ServiceGroup serviceGroupName="Philly.SG2">
                  <Qam qamName="Philly.110"
                        qamGroup="qg4.edge01.philly"
                        frequency="611000"
                        modulationMode="QAM256"
                        tsid="110"
                        serviceName="VOD" >
                  </Qam>
                  <Qam qamName="Philly.112"
                        qamGroup="qg4.edge01.philly"
                        frequency="617000"
                        modulationMode="QAM256"
                        tsid="112"
                        serviceName="VOD,SDV" >
                  </Qam>
            </ServiceGroup>
      </StreamingZone>
</QamInventory>
```

6.3 Response Messages

6.3.1 HTTP Status Code and Status Text

The *Status Code* and *Status Text* in the response indicate whether the server successfully processed the request or failed to process the request.

For a successful response, the server MUST return a *200 OK* status if the response includes a message body. If a successful response only includes an HTTP header, the server MUST return a *204 No Content* status.

The server MUST return a *307 Temporary Redirect* when the client has to be redirected to a different server to process the request. The server MUST specify the alternate server URL in the Location HTTP header.

The server MUST return one of the following status codes to indicate a client-side request error:

400 Bad Request – Malformed HTTP request

403 Forbidden – Client does not have sufficient privileges to execute the operation

404 Not Found – Item specified in the GET parameters is not found.

The server MUST return one of the following status codes to indicate a server-side error:

500 Internal Server Error – This is the most common type of error condition. It indicates that the server failed to process the request. The server MUST include the exact details of the error in the XML in the Response element.

501 Not Implemented – Request method is not supported

505 HTTP Version Not Supported

6.3.2 Caching Results

The server's default action MUST be to disable the caching of HTTP response messages. This is accomplished by including the *cache-control: no-cache* header in the HTTP response message. This action prevents intermediary HTTP servers from caching the results. However, the client application MAY cache the results as specified in the individual interfaces.

There are conditions when some results are cacheable at the protocol level on HTTP servers. These exception cases will be explicitly specified in the interface definitions. For these cases, the server MUST include the *Cache-Control: max-age* header in the HTTP response to specify the maximum amount of time that the HTTP response can be cached Time to Live (TTL value).

6.3.3 HTTP Response Headers

Compliance with HTTP 1.1 requires that the *Content-Length* header be provided when a message body is present. This is unchanged for the interfaces specified here. In this specification set, "`...`" is used as the value of this header, although an accurate count of characters in the message is required by HTTP 1.1.

To aid in troubleshooting, the server MUST include the *Server* header in all responses. The value of this header SHOULD describe the server application and the version of the application.

The server MUST include the *Cache-Control* header to ensure that any intermediate caches handle the response appropriately. The server MUST support the cache response directives *no-cache* and *max-age*.

When there is a message body, the server MUST use the *Content-Type* header. The server MAY include other headers. The client MAY ignore non-required headers without changing the expected behavior.

An example HTTP response example with XML body is shown here (body not shown):

```
HTTP/1.1 200 OK
Server: Example SM
Content-Type: application/xml
Content-Length: ...
Cache-Control: no-cache
```

An example HTTP response with no message body is shown here:

```
HTTP/1.1 204 No Content
Server: Example ODRM
Cache-Control: no-cache
```

6.3.4 Message Body – XML

The server MAY include XML in the message body in the response message to a successful request. The name of the root element of the XML depends on the request.

For an HTTP GET response, the root element MUST match the operation name specified in the URI of the HTTP GET request.

For an HTTP POST response, the root element MUST be the operation name specified in the URI of the HTTP POST request, with "Result" appended to the end. For example, if the URI of the POST is "/ds/QamInventory", then the root element of the XML in the response message body will be "QamInventoryResult". The addition of "Result" is used to differentiate the XML element in the response from the XML element in the request.

The namespace used for the XML document is specified in Section 6.2.5 Message Body – XML.

6.4 Message Flow

6.4.1 Request-Response Flow

The client connects to the server, sends an HTTP request, and waits for an HTTP response. The client MAY pipeline requests; that is, the client can send multiple requests on the same connection before receiving a response to the first request. The server MUST process all requests and return responses in the order they were sent.

Alternatively, the client MAY open multiple connections to the server and issue multiple requests in parallel. The server MUST process each request and send an HTTP response to the client on the same connection that the request was received.

Based on the HTTP headers, the client or server MAY close the connection or the connection can be left open for subsequent requests from the client.

The client MAY have several open connections to the server at any given time.

Figure 5 shows the message flow sequence between the client and server when the connection is closed after a single request-response pair.

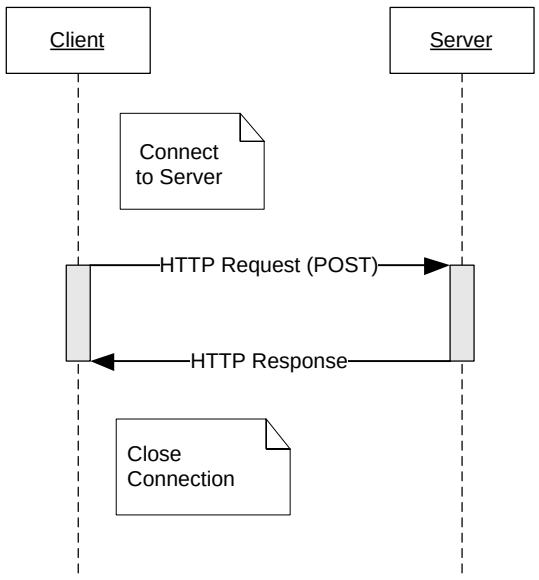

Figure 5 – Message Flow: Semi-Persistent Connection

Figure 6 shows the message flow sequence between the client and the server when the connection remains open between request-response pairs.

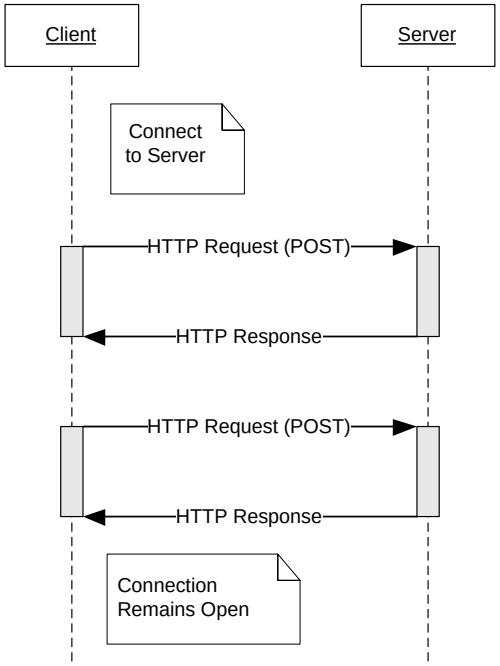

Figure 6 – Message Flow: Persistent Connection

6.4.2 Connection Lost

During the HTTP transaction, if the connection to the server is lost at any time prior to the client receiving a valid HTTP response message, the client MUST treat the request as failed. The client SHOULD immediately retry the request. If the retry fails, the client MAY discard the request or retry the request at a later time. The method of handling the failure is implementation specific, and can vary by importance of the request type.

In most cases, the server will have started processing the request prior to connection loss. If the connection to the client is lost, the server SHOULD finish processing the request. The server MUST NOT establish a connection with the client in order to return the response. It is up to the client to reconnect and retransmit the request.

6.4.3 Request Timeout

The client sets a timer when it sends an HTTP request. This timer represents the maximum amount of time that the client will wait for a response from the server. The client SHOULD set a timeout period of two (2) seconds.

If the client fails to receive an HTTP response before the timer expires, the client MUST treat the request as failed. The client MAY discard the request, immediately retry the request, or retry the request at a later time. The method of handling the failure is implementation specific, and may vary by importance of the request type.

7 COMMON XML ELEMENTS

7.1 Common Data Types

The following data types have been created to support the RMI HTTP protocol.

Table 2 – Common Data Types

Data Type Name	Base Type	Type Constraints	Description
NPT	xs:string		Represents a normalized play-time value in seconds.milliseconds decimal format, with a maximum of 3 digits in the fractional part. There must be at least one digit preceding and following the decimal point. The decimal value '0.0' is used to mean "beginning of stream". The value may be the string 'EOS', signifying End of Stream., Valid values include: 3579.123 2.2 0.025 24.0 EOS

7.2 CASInfo

The CASInfo element contains the 'casID' and 'casBlob' attributes. This is information generated by the CAS system and used by the encryptor to acquire the encryption data from the Entitlement Control Message Generator (ECMG).

Children: None

Table 3 – CASInfo Attribute Definitions

Attribute	Use	Data Type	Description
casID	Required	xs:string	The ID of the CAS system.
casBlob	Optional	xs:hexbinary	The casBlob is generated by the Conditional Access (CA) system for used by the encryptor and ECMG. Note that the requirement for this attribute in session setup is CA System dependent.

7.2.1.1 Example CASInfo XML

```
<CASInfo
      casId="47A0"
      casBlob=be074250cc5a11d98cd50800200c9a66>
</CASInfo>
```

7.3 EncryptControl

The EncryptControl element holds encryption control data for an encrypted stream. This information is sent to the encryption engine. If the stream is pre-encrypted, the encryptionType attribute value is set to PRE-ENCRYPTION and the Edge Device MUST NOT alter any encryption-associated attributes of the stream.

Children: None

Table 4 – EncryptControl Attribute Definitions

Attribute	Use	Data Type	Description
encryptionScheme	Conditionally Required	xs:string	Encryption scheme used by the encryption engine. Valid values: OTHER | DES | 3DES | AES | DVB-CSA | DVB-CSA3 This attribute is only required when the value of the encryptionType attribute is SESSION or NON-SESSION.
keyLength	Optional	xs:integer	Length of the key in bits. Limited to 4 digits. A value of zero (0) is used to indicate the default key length as specified by the reported Encryption Scheme.
blockStream	Optional	xs:boolean	Flag indicating whether the encryption engine should ensure that the ECM has been received from the ECMG and applied to the stream before allowing the stream to flow to the destination or output. Default value is "true".
encryptorOpaque	Optional	xs:string	encryptorOpaque holds private data used by the Encryptor which may be under CA license from the CA vendor.
encryptionGroupID	Optional	xs:string	This attribute may be used by the Encryptor to identify services with common parameters.
encryptionType	Required	xs:string	Indicates the type of encryption required for the session. Valid values: PRE-ENCRYPTION | BROADCAST | ON-DEMAND PRE-ENCRYPTION – the stream has had encryption applied before reaching the Edge Device and will be passed without modification. BROADCAST – the stream should be encrypted with the Access Criteria appropriately derived for broadcast video source content used in linear and SDV services. ON-DEMAND – the stream should be encrypted with the Access Criteria appropriately derived for on-demand video source content used in VOD services.

7.3.1 Example EncryptControl XML

7.3.1.1 Example EncryptControl: Encrypted Stream

```
<EncryptControl
      encryptionScheme="AES"
      keyLength="128"
      encryptorOpaque="a4a5d2f52g255c26g41d1ee651d2236f4f5g51a1b2b3c4d5e6f7g8
                       f9e213658745235aabd4e56f23"
      encryptionGroupID="123"
      encryptionType="BROADCAST">
</EncryptControl>
```

7.3.1.2 Example EncryptControl: Pre-Encrypted Stream

```
<EncryptControl
      encryptionType="PRE-ENCRYPTION">
</EncryptControl>
```

7.4 EventNotify

The EventNotify element is the root element for an event notification request message. This message is used to notify a resource manager or a session manager of a significant event.

Children:

SessionList (0..1)

MulticastTransport (0..1)

QamTransport (0..1)

UnicastTransport (0..1)

Table 5 – EventNotify Attribute Definitions

Attribute	Use	Data Type	Description
eventCode	Required	xs:integer	Event code ID of the event. See Section 8.2 for a list of valid codes and their meaning.
eventText	Required	xs:string	Text description of the event.
eventDate	Required	dateTime	Universal Time Code (UTC) Date/time when the event occurred, expressed in ISO 8601:2000 format.
npt	Optional	NPT	Normal Play Time (NPT) of the event, if the event is related to a single session. This attribute in only used for EventNotify messages that originate from the Streaming Server or ODRM.

The SessionList element is defined in Section 7.11. If the event is related to one or more sessions, the SessionList element is included with a list of impacted sessions. This allows for efficient notification of events that impact many sessions, such as QAM failure.

The MulticastTransport element is defined in Section 7.5. It is used to indicate which multicast stream was being delivered to the clients when a fail over to a redundant multicast stream occurs. See [RMI-ERM-EDGE].

The QamTransport element is defined in Section 7.7. If the event is related to one or more QAM transports, the QamTransport element is included with a list of impacted QAMs.

The UnicastTransport element is defined in Section 7.14. If the event is related to one or more unicast sessions, the UnicastTransport element is included with a list of impacted streams.

7.4.1 Example EventNotify XML: VOD Session Event

```
<EventNotify
      eventCode="5401"
      eventText="Downstream Failure"
      eventDate="2010-10-23T16:04:37Z" >
      <SessionList>
            <SessionId>be074250cc5a11d98cd50800200c9a66</SessionId>
      </SessionList>
</EventNotify>
```

7.4.2 Example EventNotify XML: IGMP Join Success Event

```
<EventNotify
      eventCode="2105"
      eventText="Stream Delivery Success"
      eventDate="2010-10-23T16:04:37Z" >
      <SessionList>
            <SessionId>be074250cc5a11d98cd50800200c9a66</SessionId>
```

```
      </SessionList>
      <MulticastTransport
            sourceAddress="10.32.2.17"
            groupAddress="224.2.1.103"
            recvAddress="10.40.1.23"
            groupPort="1901" >
      </MulticastTransport>
</EventNotify>
```

7.5 MulticastTransport

The MulticastTransport element specifies the details of a multicast stream. It indicates the multicast stream that is being delivered to the clients and is used when a session is established or when a fail over to a redundant source occurs.

Multiple MulticastTransport elements may be provided to specify multiple linear sources. All sources are required to have the same format and codec, although the bandwidth may be different for each. Multiple sources are provided for redundancy purposes. When multicast redundancy is used, the priority attribute is used to define which MulticastTransport element is the primary stream and which is the redundant stream – the higher number is the primary stream. If two MulticastTransport elements have the same priority value, hot-hot redundancy is implied. See the Multicast Redundancy section of [RMI-ERM-EDGE] for further details.

When using IGMP v3, the source address is required for each multicast stream. In case of IGMP v3, where source-specific multicast (SSM) is feasible, all multicast group addresses may be identical.

In the case of IGMP v2, IP traffic cannot be filtered using source IP address. Therefore, it is recommended that the multicast group address should be different for the redundant sources. The multicast group address can be the same as the redundant sources if they are delivered on different network paths and connected to different Edge Device inputs. The source IP address should be omitted or set to 0.0.0.0 if the ERM does not require inspection of a multicast source IP address.

Children: None

Table 6 – MulticastTransport Attribute Definitions

Attribute	Use	Data Type	Description
sourceAddress	Optional	xs:string	This is the source address of the device that is transmitting the multicast stream. This attribute is required when using IGMP v3.
groupAddress	Required	xs:string	This is the multicast group for this multicast. For example: 224.1.1.2
recvAddress	Conditiionally Required	xs:string	IP address of the Edge Device interface where a received multicast is bound. This attribute is not required in the SetupSession request from the Session Manager to the Edge Resource Manager, as the receive address is not yet known to the Session Manager. This attribute is required in the SetupSession request from the Edge Resource Manager to the Edge Device, and all successful SetupSession response messages.
groupPort	Required	xs:integer	UDP port number of the multicast service.
program	Conditionally Required	xs:integer	MPEG program number. The program attribute is required when demultiplexing a program from an input (MPTS) source. If this attribute is provided, regardless of transport stream type Multi-Program Transport Stream/ Single-Program Transport Stream (MPTS/SPTS), the supplied program number is required be present in the source transport stream; otherwise, the request will be rejected. If this attribute is not supplied in the SetupSession request, the system assumes a passthrough stream.

Attribute	Use	Data Type	Description
bitrate	Conditionally Required	xs:integer	Bit rate of this multicast stream in bits per second. This attribute is required if the bit rate is different than the bit rate advertised in the SetupSession request.
pid	Conditionally Required	xs:integer	Input elementary stream MPEG Packet Identifier (PID) number. The pid attribute is required only when routing an ancillary PID stream from a multicast IP source.
priority	Conditionally Required	xs:integer	This attribute is a number that identifies the preference order of this transport stream; higher number indicates a higher priority. It is used to order the multicast transport stream for the purpose of redundancy in the case of multiple multicast video sources. If two entries have the same "Priority", it implies Hot-Hot redundancy. Required if a redundant multicast stream is being configured.

7.5.1 Example MulticastTransport XML: SPTS Video Source

```
<MulticastTransport
      sourceAddress="10.30.1.24"
      groupAddress="224.2.1.103"
      recvAddress="10.40.1.21"
      groupPort="1901"
      program="5" >
</MulticastTransport>
```

7.5.2 Example MulticastTransport XML: PID Stream Source

```
<MulticastTransport
      sourceAddress="10.30.1.24"
      groupAddress="224.2.1.103"
      recvAddress="10.40.1.21"
      groupPort="1901"
pid="4501" >
</MulticastTransport>
```

7.5.3 Example MulticastTransport XML: MPTS Pass-Through Video Source

```
<MulticastTransport
      sourceAddress="10.30.1.24"
      groupAddress="224.2.1.103"
      recvAddress="10.40.1.21"
      groupPort="1901" >
</MulticastTransport>
```

7.5.4 Example MulticastTransport XML: MPTS Program Video Source

```
<MulticastTransport
      sourceAddress="10.30.1.24"
      groupAddress="224.2.1.103"
      recvAddress="10.40.1.21"
      groupPort="1901"
program="3" >
</MulticastTransport>
```

7.6 QamNameList

The QamNameList element provides a list of QamName elements.

Children: QamName (1..N)

The QamNameList element has no attributes; it simply contains a list of QAM names.

7.6.1 QamName

The QamName element contains a text string that defines the name of a QAM on an Edge Device. The name of the QAM is not an attribute of the element.

Table 7 – QamName Element Definition

Data Type	Description
xs:string	Unique name of the QAM. This is typically a string created by concatenating the following: <Streaming Zone> + "." + <transport stream ID (TSID)>

7.6.2 Example QamNameList XML

```
<QamNameList>
      <QamName>Philly.112</QamName>
      <QamName>Philly.113</QamName>
</QamNameList>
```

7.7 QamTransport

The QamTransport element specifies RF channel parameters that can be used for tuning by the client.

Children: None

Table 8 – QamTransport Attribute Definitions

Attribute	Use	Data Type	Description
qamName	Required	xs:string	Unique name of the QAM. This is typically a string created by concatenating the following: <Streaming Zone> + "." + <transport stream ID (TSID)>
frequency	Required	xs:integer	The center frequency of the QAM that is used for tuning. This value is in kHz.
mpegProgram	Conditionally Required	xs:integer	The output MPEG program number used for tuning. The mpegProgram attribute is required for the SetupSession request and response when multiplexing a specific program into the output multiplex. It is not required for MPTS pass through or when multiplexing an Ancillary PID Stream.
pid	Conditionally Required	xs:integer	Output elementary stream MPEG PID number. The pid attribute is required for the SetupSession request and response when routing an ancillary PID stream into the output multiplex.

Attribute	Use	Data Type	Description
pidType	Conditionally Required	xs:string	Output elementary stream type. Valid values include: FIXED | EAS | NET | EMM | PSIP FIXED – PID stream is mapped to the output stream without modification of MPEG tables. EAS – Input is an EAS stream that will be interleaved with the existing Program and System Information Protocol (PSIP) tables being transmitted on the PSIP PID. NET – PID stream is mapped to the output stream and mapped into the Program Association Table (PAT) table as the Network Information Table (NIT) program PID number. EMM – PID stream is mapped to the output stream and added to the Conditional Access Table (CAT) table. PSIP – PID stream replaces/updates the existing PSIP PID stream. The pidType attribute is required for the SetupSession request and response when routing an elementary PID stream into the output multiplex.
modulationMode	Required	xs:string	Indicates the modulation mode of the QamTransport. Valid values are: QAM64 | QAM256
tsid	Required	xs:integer	The transport stream ID of the MPTS carrying the video. The client uses this value to validate the stream after tuning.
annex	Optional	xs:string	Valid values are: A | B | C The annex attribute is optionally returned in the SetupSession response. This attribute is not used for SetupSession requests.
channelWidth	Optional	xs:integer	QAM channel width in MHz. Valid values are: 6 | 7 | 8 The channelWidth attribute is optionally returned in the SetupSession response. This attribute is not used for SetupSession requests.
symbolRate	Optional	xs:hexBinary	Annex B [ITU J.83] based systems: 0x004C4B40 Annex A [ITU J.83] based systems: Use the value defined for the delivery network. The symbolRate attribute is optionally returned in the SetupSession response. This attribute is not used for SetupSession requests.
interleaver	Optional	xs:string	Valid values include: I128J1 | 128J2 | I64J2 | I128J3 | I32J4 | I128J4 | I16J8 | I128J5 | I8J16 | I128J6 | I128J7 | I128J8 | I12J17 The interleaver attribute is optionally returned in the SetupSession response. This attribute is not used for SetupSession requests.

7.7.1 QamTransport Element XML Samples

7.7.1.1 Request Example: MPTS Pass Through

```
<QamTransport
      qamName="Philly.210">
</QamTransport>
```

7.7.1.2 Response Example: MPTS Pass Through with QAM Qualification

```
<QamTransport
      qamName="Philly.210"
      frequency="693000"
```

```
      modulationMode="QAM256"
      tsid="210">
</QamTransport>
```

7.7.1.3 Request Example: Program Multiplexing

```
<QamTransport
      qamName="Philly.210"
      mpegProgram="101">
</QamTransport>
```

7.7.1.4 Response Example: Program Multiplexing with QAM Qualification

```
<QamTransport
      qamName="Philly.210"
      frequency="693000"
      mpegProgram="101"
      modulationMode="QAM256"
      tsid="210" >
</QamTransport>
```

7.7.1.5 Request Example: Ancillary PID Stream Multiplexing

```
<QamTransport
      qamName="Philly.210"
      pid="8188"
      pidType="NET">
</QamTransport>
```

7.7.1.6 Response Example: Ancillary PID Stream Multiplexing with QAM Qualification

```
<QamTransport
      qamName="Philly.210"
      pid="8188"
            pidType="NET"
      frequency="693000"
      modulationMode="QAM256"
      tsid="210">
</QamTransport>
```

7.8 Response

The Response element indicates the results of an operation. The Response element only appears in the XML of a HTTP GET or POST response message. The client MUST provide a value in the responseCode attribute that specifies the result code of the operation. The client MAY include a value in the responseText attribute to specify detailed information about the result if the result was not a failure. If the result was a failure, the client MUST include a failure reason in the responseText attribute.

Children: None

Table 9 – Response Attribute Definitions

Attribute	Use	Data Type	Description
responseCode	Required	xs:integer	Integer value that indicates the result of the operation. A response code of 0 indicates no error.
responseText	Conditionally Required	xs:string	Detailed description of the results of the operation.Typically included in failures to specify details about the failure.

7.8.1 Response Element XML Example

```
<Response
      responseCode="772"
      responseText="Server Setup Failed – SOP Not Available">
</Response>
```

7.9 RightsMetadata

The RightsMetadata element is used by the Conditional Access System (CAS) to construct Access Criteria (AC). This element contains information about the content, content encryption requirements, client, and client decryption capabilities.

Children: None

Table 10 – RightsMetadata Attribute Definitions

Attribute	Use	Data Type	Description
providerId	Required	xs:string	The content provider ID of the asset required to be encrypted.
assetId	Required	xs:string	The content asset ID of the asset requiring encryption.
updateNumber	Optional	xs:integer	Version number associated with the asset ID. Used to indicate a change in the asset's terms from the previous version.
copyControl	Required	xs:string	Copy protection requirement for assets, as specified in "Copy Control Information (CCI)" in Section 9 of [CCCP]. Valid values are: NEVER | ONCE | FREELY | NO_MORE
aps	Required	xs:string	Analog Protection Signaling. Valid values: OFF | SB | 2LSB | 4LSB | UNDEFINED OFF – APS disabled (default) SB – APS On, split burst mode 2LSB – APS On, 2 line split burst mode 4LSB – APS on, 4 line split burst mode UNDEFINED – Allows APS to be set by application
cit	Required	xs:string	Constrained Image Trigger. Valid values: OFF | ON
rct	Required	xs:string	Redistribution Control Trigger. Valid values: OFF | ON
rightsExpiration	Optional	xs:datetime	Date and time that the content rights expire. Date/time specified in ISO 8601:2000 format. Time is specified in UTC.

7.9.1 RightsMetadata Sample XML

```
<RightsMetadata
      providerId="Warner Bros"
      assetId="abcd1234567890123456"
      updateNumber="1"
      copyControl="NEVER"
      aps="OFF"
      cit="ON"
      rct=”ON”
      rightsExpiration="2005-10-20T12:00:00Z">
```

```
</RightsMetadata>
```

7.10 SessionGroupList

The SessionGroupList element has no attributes; it serves as a container for a list of session group names.

Children: SessionGroup (1..N)

7.10.1 SessionGroup

The SessionGroup element holds the name of a Session Group, which can be used to associate sessions to services and signaling.

Children: None

Table 11 – SessionGroup Element Definition

Data Type	Description
xs:string	Name of the Session Group, which is used to associate sessions for session list and signaling.

7.10.2 SessionGroupList Example XML

```
<SessionGroupList>
      <SessionGroup>SM1.GROUP4</SessionGroup>
      <SessionGroup>SM1.GROUP6</SessionGroup>
      <SessionGroup>SM1.GROUP7</SessionGroup>
</SessionGroupList>
```

7.11 SessionList

The SessionList element has no attributes; it serves as a container for a list of session IDs. This list is used to report the name of the session in the response messages of many requests, such as GetSessionList and EventNotify.

Children: SessionId (0..N)

7.11.1 SessionId

This element contains the name of a session; it is used to report the name of one or more sessions in a response message.

Children: None

Table 12 – SessionId Element Definition

Data Type	Description
xs:string	Session Manager assigned session ID for the session.

7.11.2 SessionList Example XML

```
<SessionList>
      <SessionId>be074250cc5a11d98cd50800200c9a66</SessionId>
      <SessionId>c8ba47402f494a6d9037fdda7746ca29</SessionId>
      <SessionId>1301b42a229149909611c9584f833fe8</SessionId>
      <SessionId>78c5d9cefe0d45d5871e812a294caf05</SessionId>
</SessionList>
```

7.12 SessionStatus

The SessionStatus element specifies the current status for an active session.

The eventCode attribute indicates if an event or error has occurred. A value of 0 indicates that no error or session-related event has occurred. If an error or event occurs and the session is currently in that state, the eventCode will provide a value from Table 17 – Event Notify Codes in Section 8.2.

Children:

QamTransport (0..1)

MulticastTransport (0..1)

UnicastTransport (0..1)

EncryptControl (0..1)

Table 13 – SessionStatus Attribute Definitions

Attribute	Use	Data Type	Description
eventCode	Required	xs:integer	ID of the event as specified in Section 8.2 Event Notify Codes. A value of 0 (zero) indicates the session has no errors or events.
eventText	Conditionally Required	xs:string	Text description of the event. The client MUST supply this description if the eventCode is not 0 (zero).
eventDate	Conditionally Required	xs:datetime	Date/time when the error/event occurred. This is in ISO 8601:2000 format. The client MUST supply this value if the eventCode is not 0 (zero).

The QamTransport element is defined in Section 7.7. The MulticastTransport element is defined in Section 7.5. The UnicastTransport element is defined in Section 7.14. The EncryptControl element is defined in Section 7.3.

7.12.1 SessionStatus XML Examples

7.12.1.1 Example SessionStatus XML for a Failed Session

```
<SessionStatus
      eventCode="6000"
      eventText="Encryption Engine Failure"
      eventDate="2010-10-23T16:04:37Z" >
</SessionStatus>
```

7.12.1.2 Example SessionStatus XML for a Linear Session

```
<SessionStatus
      eventCode="6000"
      eventText="Encryption Engine Failure"
      eventDate="2010-10-23T16:04:37Z" >
      <MulticastTransport
            sourceAddress="10.30.1.24"
            groupAddress="224.2.1.103"
            recvAddress="10.40.1.21"
            groupPort="1901">
      </MulticastTransport>
      <EncryptControl
            casId="47A0"
            encryptionType="BROADCAST"
```

```
            encryptionScheme="AES"
            keyLength="128">
      </EncryptControl>
      <QamTransport
            qamName="Philly.210"
            frequency="693000"
            mpegProgram="101"
            modulationMode="QAM256"
            tsid="210">
      </QamTransport>
      <ContentId
            idType = "LIN"
            idValue = "12345678" >
      </ContentId>
</SessionStatus>
```

7.13 TargetClient

The TargetClient element specifies a unique identifier for the client that will receive the content (i.e., consumption device). The TargetClient can differ from the client that sent the SetupSession request. For example, the requesting client may be a mobile phone, while the client that will receive the video asset is a set-top box.

Although the attributes of the TargetClient element are optional, the client MUST include at least one attribute that uniquely identifies the consumption device from the following list:

- caMac
- hostMac
- accountNumber
- ipAddress

If the client has a conditional access MAC Address, the client MUST specify this MAC Address in the caMac attribute.

MAC addresses are represented as 12 lower-case hex digits that correspond to the 48-bit binary value, without embedded punctuation. Leading zeroes can be added as required.

Children: None

Table 14 – TargetClient Attribute Definitions

Attribute	Use	Data Type	Description
caMac	Optional	xs:hexBinary	Unique identifier of the target device. This is typically the conditional access Mac Address, but could be another entity such as the cable card.
hostMac	Optional	xs:hexBinary	Host (or "eSTB") MAC address, expressed in hexadecimal format.
cmMac	Optional	xs:hexBinary	Embedded cable modem MAC address, expressed in hexadecimal format.
accountNumber	Optional	xs:string	Subscriber account number.
profile	Optional	xs:string	The value used in navigation queries, with extension labels.
casAddress	Optional	xs:string	The address of the Conditional Access System within the Customer Premise Equipment (CPE). It is expected some systems will populate this instead of the ipAddress attribute when doing session setup. The casAddress can be an IPv4 address, IPv6 address, or host name. Examples: 10.20.30.40 or DNCS.
ipAddress	Optional	xs:string	IP address of the target device. The IP address may be IPv4 or IPv6.

Attribute	Use	Data Type	Description
ipPort	Optional	xs:int	The port to use when sending messages to the target device. This attribute can only be included if an ipAddress attribute is specified.

7.13.1 TargetClient Example XML

7.13.1.1 TargetClient with a Single MAC Address Provided

```
<TargetClient
      caMAC="c0c011223344"
      profile="A38,SCH,QAM,HD"
/>
```

7.13.1.2 TargetClient with Multiple MAC Addresses

```
<TargetClient
      caMAC="c0a011223344"
      hostMAC="d0e011223344"
      profile="B1.1,SCH,QAM,HD,SOC"
/>
```

7.13.1.3 TargetClient with IP Address

```
<TargetClient
      profile ="FSLAB,SCH,QAM,IPSTREAM,HD,SOC"
      ipAddress="10.24.101.43"
      ipPort="8080"
/>
```

7.14 UnicastTransport

The UnicastTransport element provides source and/or destination IP details for unicast streams. The source is typically a streaming server and the destination is typically an Edge Device. Note that other sources are also supported, such as a network encryptor.

Children: None

Table 15 – UnicastTransport Attribute Definitions

Attribute	Use	Data Type	Description
sourceAddress	Optional	xs:string	Source IP address for streamed content (i.e., streaming server output IP address). This may be an IPv4 or IPv6 address.
sourcePort	Optional	xs:integer	Source port number for streamed content.
sopName	Optional	xs:string	Name of the streaming output port, if the source is a streaming server.
sopGroup	Optional	xs:string	Name of the streaming output port group, if the source is a streaming server.
bitrate	Optional	xs: integer	Bit rate of this stream in bits per second.
destinationAddress	Conditionally Required	xs:string	Target IP address for streamed content or PID stream (i.e., Edge Device input IP address). This may be an IPv4 or IPv6 address. See the details of the SetupSession request for details on when this attribute is required.
destinationPort	Required	xs:integer	Target port number for streamed content or PID stream. See the details of the SetupSession request for details on when this attribute is required.

Attribute	Use	Data Type	Description
edgeInputGroup	Optional	xs:string	Name of the Edge Input Group if the destination is an Edge Device.
encryptorName	Optional	xs:string	Name of the selected encryptor if the destination is an encryptor.
encryptorGroup	Optional	xs:string	Name of the Encryptor Group if the destination is an encryptor.
transportProtocol	Optional	xs:string	Protocol used for transport. The default value is MP2T (MPEG-2 transport stream).
program	Conditionally Required	xs:integer	MPEG program number. The program attribute is required when demultiplexing a program from an input MPTS source. The program attribute should be zero (0) when specifying an SPTS input source.
lowerTransport	Optional	xs:string	The IP-layer transport protocol used for the stream. The default value is UDP. Valid values include: UDP | TCP (Uniform Datagran Protocol/Transmission Control Protocol)
lowerTransport	Optional	xs:string	The IP-layer transport protocol used for the stream. The default value is UDP. Valid values include: UDP | TCP
pid	Conditionally Required	xs:integer	Input elementary stream MPEG PID number. The pid attribute is only required when routing an ancillary PID stream from a unicast IP source.

7.14.1 UnicastTransport XML Examples

7.14.1.1 UnicastTransport for Edge Device Input

```
<UnicastTransport
      destinationAddress="10.200.1.23"
      destinationPort="5101"
      edgeInputGroup="Philly.eig8">
</UnicastTransport>
```

7.14.1.2 UnicastTransport for PID Stream

```
<UnicastTransport
      destinationAddress="10.200.1.23"
      destinationPort="5101"
      pid="8188”>
</UnicastTransport>
```

8 COMMON CODES

8.1 Response Codes

Table 16 lists common response codes for errors in the Resource Management Interface (RMI) system. These codes are used in the responseCode attribute of the Response element.

Table 16 – Response Codes

Code	Description
200	No Error
205	Reset Context - This result status code is used in response to a PLAY request where the server is able to play the content, but either the returned Range header or Scale header has been changed to comply with a restriction on a play list item. This code is not necessary if the Range header indicates a different value due to the precision necessary for a specific splice location and the Scale header indicates a different value due to the specific trick speeds that are supported by the content.
400	Bad Request – Request had missing or unexpected parameters.
403	Forbidden – User does not have the appropriate privileges to access the data.
404	Not Found – Requested data could not be found.
405	Method Not Allowed
408	Request Time Out
412	Pre-Condition Failed
415	Unsupported Content Type
451	Invalid Parameter
453	Insufficient Bandwidth
454	Session Not Found
455	Method Not Valid in this State
456	Header Field Not Valid
457	Invalid Range
458	Parameter is Read-Only
459	Aggregate Operation Not Allowed
460	Only Aggregate Operation Allowed
461	Unsupported Transport
462	Destination Unreachable
470	Rescan – QAM Scan Report includes invalid or unexpected data. Client needs to rescan the frequencies.
480	Session not set up. Hot-Hot multicast redundancy not supported by Edge Device.
500	Internal Server Error
501	Not Implemented
502	Bad Gateway
503	Service Unavailable
504	Gateway or Device Timeout
505	Protocol Version Not Supported
551	Option Not Supported
599	No Available Streaming Servers

Code	Description
600	Insufficient Encryption Resources
601	ECMG Not Available or Not Responding
602	Stream/Service identified for encryption processing indicates it is pre-encrypted content
603	ECMG Authentication Error
604	ECMG Session Setup Failure
610	Non-Valid Encryptor Certificate
620	Unexpected HTTP Status Code from ECMG
650	SM Setup Failed – No Response
651	SM Setup Failed – Unknown QAM or Service Group
652	SM Setup Failed – Invalid Request
653	SM Setup Failed – Internal Error
670	ERM Setup Failed – No Response
671	ERM Setup Failed – Invalid Request
672	ERM Setup Failed – QAM Bandwidth Not Available
673	ERM Setup Failed – Network Bandwidth Not Available
674	ERM Setup Failed – Program Not Available
675	ERM Setup Failed – Service Group Not Found
676	ERM Setup Failed – QAM Not Found
677	ERM Setup Failed – QAM Not available
678	ERM Setup Failed – Edge Device Not Available
679	ERM Setup Failed – Internal Error
800	Auto Discovery Failed – No Response Object is damaged
801	Auto Discovery Failed – Client Not Found Error processing MPEG data
802	Auto Discovery Failed – Service Group Not Found Object too large
803	Auto Discovery Failed – Invalid Request Out of disk bandwidth
804	Auto Discovery Failed – Internal Error Ingest Error

8.2 Event Notify Codes

Table 17 below lists common event codes used in the RMI system.

Table 17 – Event Notify Codes

Code	Description
2105	Stream Delivery Success – Edge Device may send this to the ERM when it successfully delivered a unicast or multicast stream.
5200	Server Resources Unavailable – Sent when there is an issue with availability of server resources.
5401	Downstream Failure - Sent when there is a downstream failure, e.g., QAM failure.

Code	Description
5402	Client Session Terminated – Sent as a notification that for some reason the client session was terminated by the server.
5403	Network Delivery Failure – SDV stream is lost and the Edge Device cannot fail over to an alternate stream source
5404	Unable to Join Multicast Group – Edge Device sends a notification with this reason code when an IGMP Join fails. If multiple source addresses were provided to the Edge Device, then an IGMP Join failure means that all sources failed.
5405	Input Port Failure – Edge Device input port failure.
5406	Switch Over to Redundant Multicast Source – Edge Device detects a loss of input of the SDV stream from the current multicast source and it fails over to a redundant multicast source.
5407	Stream Delivery Failure – Edge Device failed delivering a unicast stream.
5502	Internal Server Error – Sent as a notification that there was an issue with the server.
5602	Bandwidth Exceeded Limit
5700	Session In Progress – Sent by server when a session timer expires.
6000	Encryption Engine Failure/Fault
6001	Stream Bandwidth Exceeds that Available
6004	Downstream Destination Unreachable (i.e., ICMP Host/Port Unreachable, ARP Request Failure, etc.)
6005	Unable to Encrypt One or More Components
6006	ECMG Session Failure
6007	ECMG Re-authentication Error
6008	Encryption Control Word Stretching

8.2.1 Teardown Reason Codes

Table 18 below lists common teardown reason codes in the RMI system.

Table 18 – Teardown Reason Codes

Code	Description
200	User Stop
201	End of Stream
202	Beginning of Stream
203	Pause Timeout
204	No user activity
205	Settop capability mismatch
206	Insufficient Priority
207	Network Delivery Failure
400	Fail to Tune
401	Loss of Tune
403	Control Plane Communication Failure
404	Channel Failure
405	Server Failed to Respond
406	Trick Play Failed
407	Internal ODA Issue
408	Unknown

Code	Description
409	Network Resource Failure
410	TSID Mismatch
420	Settop Heartbeat Timeout
421	Settop Inactivity Timeout
422	Content Unavailable
423	Streaming Failure
424	QAM Failure
425	Volume Failure
426	Stream Control Error
427	Stream Control Timeout
428	Session List Mismatch
500	Internal Server Error
502	QAM Parameter Mismatch
550	Session Timeout
600	Encryption Error

Appendix I Acknowledgements

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.

Contributor	Company Affiliation	Contributor	Company Affiliation
Weidong Mao	Comcast	Niem Dang	Time Warner Cable
Doug Will	Comcast	Brian Floyd	Time Warner Cable
Glenn Babecki	CCAD	Ty Pearman	Comcast
Joe Solomon	Comcast		