Video Specifications
IP Multicast

IP Multicast Controller-Client Interface Specification

OC-SP-MC-EMCI-C01-161026

CLOSED

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 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	OC-SP-MC-EMCI-I02-160923			
Document Title	IP Multicast Controller-Client Interface Specification			
Revision History	I01 - 05/28/2015 I02 - 09/23/2016 C01 - 10/26/2016			
Date	October 26, 2016			
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 Video family of specifications developed by Cable Television Laboratories, Inc. (CableLabs) and published under the OpenCable License Agreement. The IP Multicast MC-EMC Interface Specification defines an interface identified in the IP Multicast Technical Report [IPM-TR]. The intent of this specification is to provide multi-vendor interoperability across this interface such that interoperable products can be brought to market which support Multicast-Assisted ABR (Adaptive Bit Rate).

The IP Multicast specifications primarily adopt web services as the standard communications mechanism between components.

1.2 Purpose

This document specifies the usage of the HTTP protocol in the IP Multicast 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 across the IP Multicast interface specifications.

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 IP Multicast:

- Multicast Controller to Embedded Multicast Client (mc-emc) interface: The mc-emc interface is defined between a Multicast Controller in an operator's back office and a Multicast Client embedded in a residential gateway. This interface is used to signal viewing-related activity and the list of content streams available via IP multicast.
- Multicast Controller to Multicast Server (mc-ms) interface: The mc-ms interface is defined between the Multicast Controller, which controls what streams are available on multicast and when, and the Multicast Server, which performs content retrieval and multicast delivery.

The interfaces defined in the IP Multicast reference architecture are shown in Figure 1.

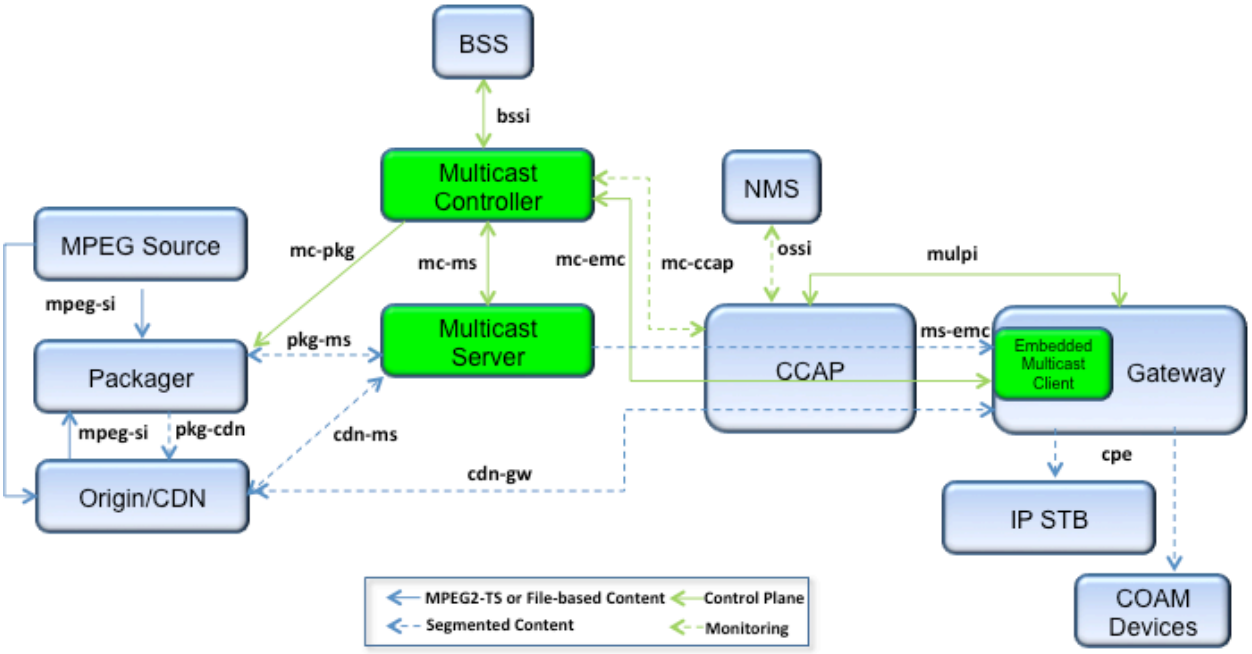

Figure 1 - IP Multicast Reference Architecture

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 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.

2 REFERENCES

2.1 Normative References

This specification uses the following normative references.

[MULPI]	MAC and Upper Layer Protocols Interface Specification, CM-SP-MULPIv3.1-I09-160602, June 2, 2016, Cable Television Laboratories, Inc.
[PCRE]	PERL Compatible Regular Expressions, http://www.pcre.org/original/doc/html/pcrepattern.html

2.2 Informative References

This specification uses the following informative references.

[IPM-TR]	IP Multicast Adaptive Bit Rate Architecture Technical Report, OC-TR-IP-MULTI-ARCH-C01-161026, October 26, 2016, Cable Television Laboratories, Inc.
[MS-EMC]	IP Multicast Server-Client Interface Specification, OC-SP-MS-EMCI-C01-161026, Cable Television Laboratories, Inc.
[RFC 2616]	IETF RFC 2616, Hypertext Transfer Protocol - HTTP/1.1, June 1999.
[RFC 3376]	IETF RFC 3376, Internet Group Management Protocol, Version 3, October 2002.
[RMI HTTP]	Resource Management Architecture and HTTP Specification, CM-SP-RMI-HTTP-I02-150528, May 28, 2015, 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/

3 TERMS AND DEFINITIONS

This specification uses the following terms:

Access Network	The HFC network between the Gateway and the CCAP.
Adaptive Bit Rate	A streaming video technique where Players select between multiple bit rate encodings of the same video stream.
Bonding Group	A logical set of DOCSIS® channels which support parallel transmission.
Companion Device	A video playback device which is not a television such as a tablet, smartphone or PC.
Converged Cable Access Platform	A system which provides DOCSIS and QAM-based video services to CMs, Gateways and set-top boxes.
Content Distribution Network	A network designed to minimizing latency by distributing network objects onto geographically diverse servers.
Embedded Multicast Client	The function embedded in the Gateway which joins multicast groups and receives multicast content.
Gateway	A customer premises device which facilitates delivery of video, data and other services.
Headend	The central location on the cable network that is responsible for injecting broadcast video and other signals in the downstream direction.
Home Network	A network within the subscriber premises which connects to the Access Network via the Gateway.
IP Multicast	A delivery mechanism whereby IP packets can be transmitted to/received from devices that have explicitly joined a multicast group.
Key Server	A server which provides keys as part of a DRM solution.
License Server	A server which checks authorization and provides licenses as part of a DRM solution.
Linear TV	A continuous content stream from a provider, e.g., a broadcast television network.
Multicast Controller	A device which controls what channels are provided via multicast.
Multicast Configuration Server	A server responsible for Gateway configuration. The configuration function is conceptually part of the Multicast Controller, but deployments can choose to utilize a separate server for this function.
Multicast Server	A device which delivers content via multicast.
Multicast-Active Stream	A stream, identified by a channelId and bitrate pair, that the Multicast Server is currently multicasting to a specific (S,G). Multicast-Active Streams match a URLRegex that is configured on a Gateway and are a subset of Multicast-Ready streams which are communicated in a ChannelMap element.
Multicast-Ready Stream	A stream that the Multicast Server might be multicasting currently or might be directed to multicast in the future. Multicast Ready Streams are communicated via ChannelMap objects. Multicast-Ready Streams match a URLRegex that is configured on a Gateway.
Packager	A device which takes continuous video streams, encodes them at different bit rates and breaks them into shorter duration segments.
Player	An application for playback of ABR video.
Serving Group	A set of receivers which all receive the same transmission of a given frequency band.
Stream	A series of video segments which contain the same video asset, typically at the same bit rate encoding.
Unicast	Delivery of IP packets to a single device.

4 ABBREVIATIONS AND ACRONYMS

This specification uses the following terms:

ABR	Adaptive Bit Rate
BSS	Business Support System
CCAP	Converged Cable Access Platform
CDN	Content Delivery Network
CM	Cable Modem
CMS	Content Management Server
COAM	Customer Owned and Managed
CPE	Customer Premises Equipment
DNS	Domain Name System
DOCSIS®	Data-Over-Cable Service Interface Specifications
EAS	Emergency Alert System
EAN	Emergency Action Notification
EMC	Embedded Multicast Client
FEC	Forward Error Correction
GW	Gateway
HD	High Definition
HDS	HTTP Dynamic Streaming
HLS	HTTP Live Streaming
HTTP	Hyper Text Transfer Protocol
IGMP	Internet Group Management Protocol
IP	Internet Protocol
IPsec	Internet Protocol Security
IP-STB	IP Set-top Box
IPv4	Internet Protocol Version 4
IPv6	Internet Protocol Version 6
JSON	JavaScript Object Notation
M-ABR	Multicast-Adaptive Bit Rate
MC	Multicast Controller
MCS	Multicast Configuration Server
MLD	Multicast Listener Discovery
MoCA	Multimedia over Coax Alliance
MPEG	Moving Picture Experts Group
MPEG-DASH	Moving Picture Experts Group Dynamic Adaptive Streaming over HTTP
MS	Multicast Server
MSS	Microsoft Smooth Streaming
NACK	Negative-Acknowledgement
NMS	Network Management System
NORM	NACK-Oriented Reliable Multicast

QAM	Quadrature Amplitude Modulation
QoS	Quality of Service
REST	Representational State Transfer
RTP	Real-time Transport Protocol
RTCP	RTP Control Protocol
RTSP	Real-Time Streaming Protocol
RTMP	Real-Time Messaging Protocol
SD	Standard Definition
SDV	Switched Digital Video
(S,G)	(Source IP Address, Group IP Address)
SNMP	Simple Network Management Protocol
STB	Set-top Box
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TR	Technical Report
UA	User Agent
UDP	User Datagram Protocol
UE	User Equipment
URI	Uniform Resource Identifier
Wi-Fi	Wireless Local Area Network
XML	eXtensible Markup Language

5 OVERVIEW AND THEORY OF OPERATIONS

Multicast ABR or, perhaps more accurately, Multicast-assisted ABR, is just that - a technique for using IP multicast to assist in the delivery of ABR video segments. It is really a network-layer efficiency mechanism which is transparent to ABR Players.

Multicast-ABR is more fully described in [IPM-TR] and the Functional Overview section of this specification.

5.1 Design Principles

This protocol was designed to provide centralized control of the streams available via multicast as well as the multicast streams watched (or, more precisely, joined) by individual Gateways. This was done to minimize complexity on the Gateway for reduced cost and increased service agility.

This protocol uses a web services architecture. The web services API structure is based on [RMI HTTP] as this protocol is used to deliver other video services.

5.2 Functional Overview

A typical M-ABR system can be thought of as a standard ABR video system, which uses a transparent caching proxy resident in the Gateway. That transparent cache can be filled either via unicast or multicast. This allows the Player to switch seamlessly between less-popular content only available on unicast and popular content available on multicast, as it is completely transparent to the Player whether the content is delivered to the Gateway via unicast or multicast. In fact, the system can switch seamlessly between unicast and multicast delivery of the same stream, as any content not delivered by multicast will be retrieved via unicast.

While this technology is referred to as "Multicast Adaptive Bit Rate (M-ABR)", it is important to note that individual multicast streams do not “adapt” their bit rates. Rather, the term is used to refer to the multicast delivery of video segment files to the Gateway, which subsequently delivers these segments via HTTP when they are requested by a streaming video Player. Each multicast stream only contains a single bit rate. The pre-filling of the Gateway’s cache is expected to result in the reliable receipt of fragments by the Player, such that the Player does not adapt and instead chooses to remain at that bit rate. However, for robustness, the manifest still generally contains reference to other bit rate encodings of the same content stream. These other bit rates can be provided on a separate multicast stream or may be available only via unicast retrieval.

The basic model for the retrieval of new content by a Player is shown in the following figure:

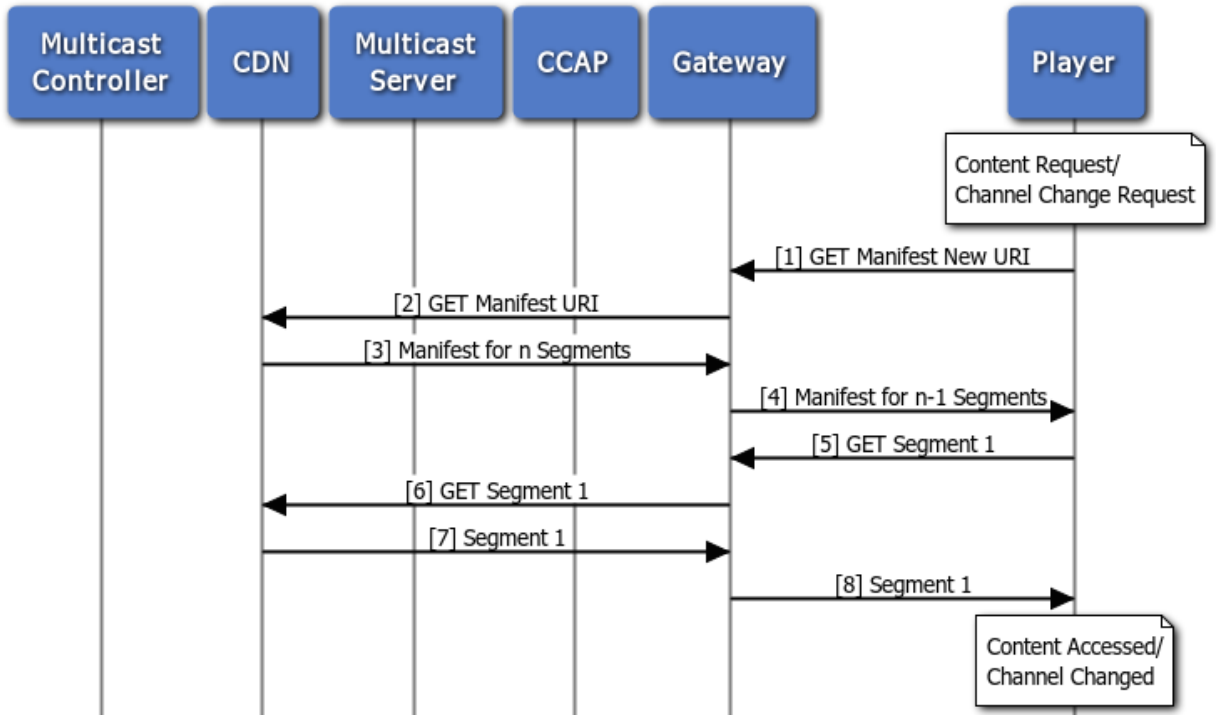

Figure 2 - Initial Unicast Retrieval of New Content

It is important to note that there are no multicast-related steps in this sequence diagram. This sequence is identical to the sequence which would occur in a unicast system with a transparent caching proxy – with one very small exception, between Step 3 and Step 4 the Gateway modifies the manifest by dropping the last segment from the list.[1] This way the Gateway always knows about more segments than the Player is aware of which allows the system time for segment delivery via multicast before the segment is requested by the Player.[2] While not important for this initial content delivery or a channel change, this is important feature for multicast delivery and will be explained in more detail later in this section.

It is also important to note that since the steps taken are identical to that for unicast, the performance of an M-ABR system is never worse than that of unicast and thus the QoE for the end user is also never worse than normal ABR retrieval. This applies to both initial content access and channel changes.

Figure 1 only shows the retrieval of the very first video segment file in the content stream. The following figure takes this one step further by showing two other aspects of this system:

The retrieval of multiple manifests and multiple segments as a video is watched.

The Gateway checking to see if a video segment is available in the cache before fetching the segment from the CDN.

Except for the fact that the Gateway's cache can be pre-filled by multicast delivery the system in the figure behaves just like a unicast transparent caching proxy would.

[1] Shortening the manifest which gets delivered to the Player can also be performed in the back office. Some operators have two versions of the manifest for a given content stream – a shortened one for Players (hereafter referred to as the Player Manifest) and a longer version for other components (hereafter referred to as the Server Manifest). Also note that the manifest can be shortened by more than one segment; this is particularly useful to ensure the Gateway stays ahead of the Player when the segments are of a short duration.

[2] Optimizations are also possible. For example, a Gateway might choose not to shorten a manifest if it knew it had all of the segments referenced by the manifest in its cache.

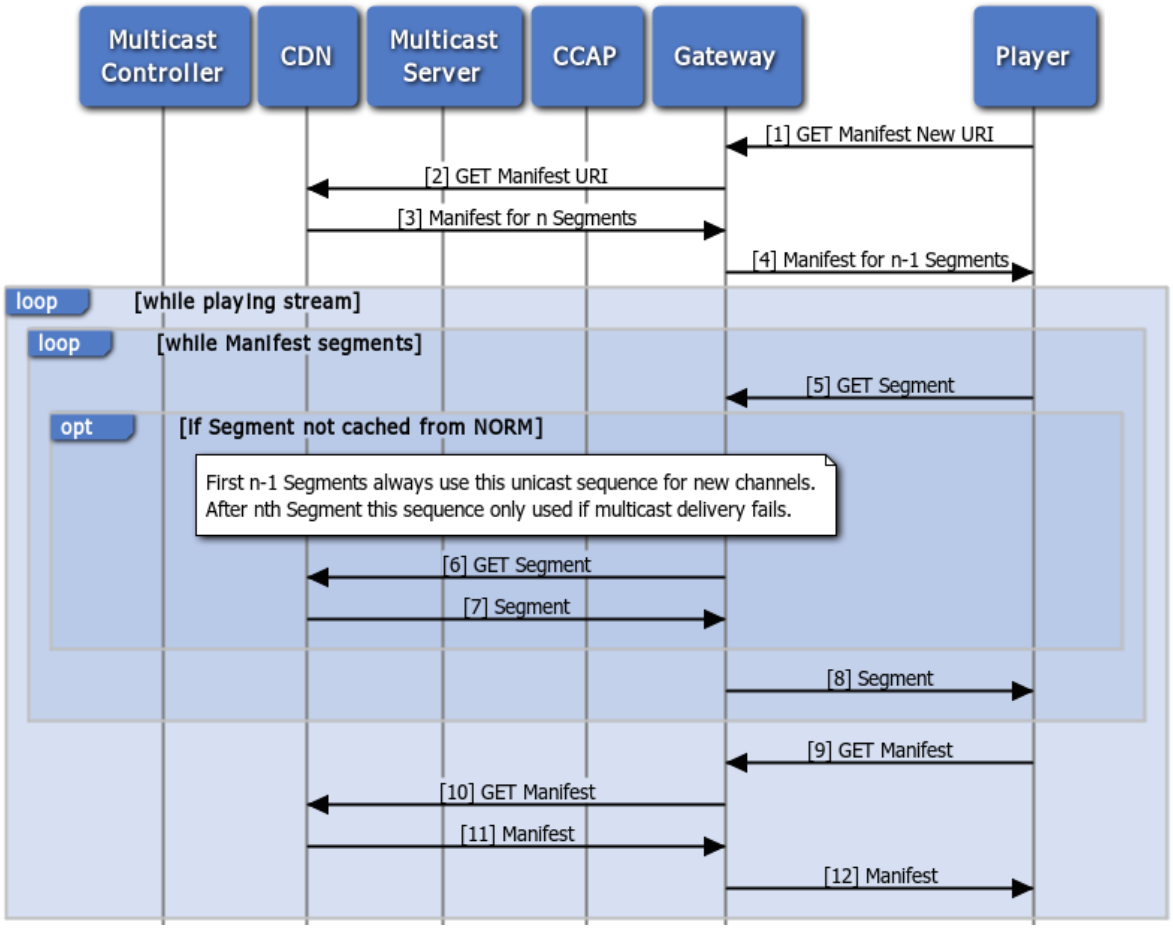

Figure 3 - Continuous Delivery of New Content

Thus, playback and channel change functions are virtually identical to unicast. Similarly, the performance in terms of channel change times and other QoE metrics is identical to unicast. Where the system differs from unicast is in the way the cache can be filled if a given content stream is available via multicast.

The following figure shows how the caching subsystem utilizes multicast to fill its cache and related signaling defined in this specification.

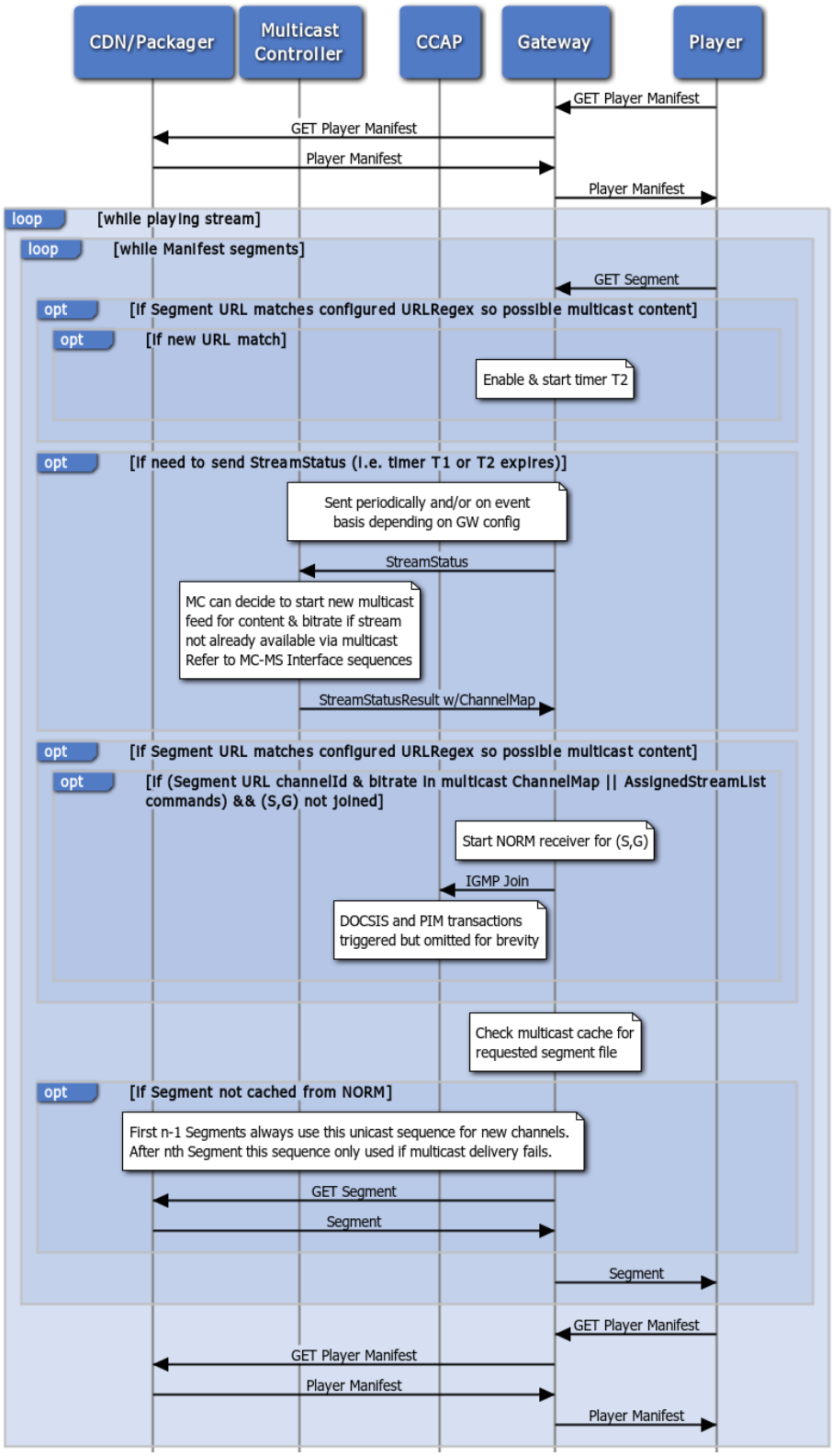

Figure 4 - Multicast Cache Filling

The unicast content delivery sequence and the multicast cache filling sequence are initiated by the same trigger – a Player request for new content. If the Gateway needs to send a StreamStatus message it sends one to the Multicast Controller. The Gateway is provisioned with a set of URL regular expressions that match configured linear video assets that can be delivered via Multicast-ABR. If any streams being requested by Players behind the Gateway match one of these URL regular expressions, the Gateway includes information about those streams in the StreamStatus message. Streams that match one of the configured URL regular expressions are streams that the Multicast Controller might be multicasting currently or might decide to multicast in the future; this document refers to these streams as Multicast-Ready Streams.

Multicast-Ready Streams being accessed by Players are of interest to the Multicast Controller and are therefore included by the Gateway in its StreamStatus message. Streams which do not match the URL regular expressions are either over-the-top streams from other sources or operator streams which are not available via multicast (thus, these streams are not included by the Gateway in its StreamStatus message).

Thus, a StreamStatus message indicating a Player accessing a Multicast-Ready Stream could potentially lead to the Multicast Controller deciding to offer this content on multicast via the Multicast Server, if this content is not already being multicast.

The Multicast Controller responds to a StreamStatus message with a StreamStatusResult message. The StreamStatusResult message can include a Channel Map of content available via multicast. This allows the Gateway to (a) determine if the representation (i.e., bit rate) and channelId requested by the Player is available via multicast and (b) identify the (S,G) for any available multicast. The StreamStatusResult message can also include a set of commanded multicast groups for the Gateway to join. If there are commanded groups then the Gateway starts a NORM (NACK-Oriented Reliable Multicast) receiver for any of the multicast groups which it is not already a member. However, if there are no commanded groups then the Gateway can start a NORM multicast receiver for any multicast groups which match the channelId and bit rate of content requested by one of the Players it services. Starting a NORM multicast receiver for a given multicast stream triggers an IGMP/MLD Join for that multicast group.

The IGMP/MLD Join triggers a number of DOCSIS MAC-layer events which are not shown in the diagram, but which are necessary for the Gateway's embedded CM to receive multicast traffic.

At this point the multicast receiver in the Gateway is available to start receiving video/audio segments via NORM. These segments are cached and made available via the transparent proxy to fulfill requests from the Player.

However, there is a race condition here – the Player is requesting segments sequentially (and potentially requesting segments faster than it is playing them) and the Gateway is also getting these segments delivered sequentially via multicast. How does the system increase the likelihood that there is a cache hit and ensure that segments have been delivered via multicast in advance of them being requested by the Player? This is where manifest manipulation comes into play. As mentioned previously, the Gateway (or Multicast Server) typically knows about at least one more segment than the Player. The system uses this to provide the Gateway with a timing advantage over the Player. The goal is for the Gateway to have segment n waiting in cache when the Player requests it while, simultaneously, the NORM multicast receiver for the stream is receiving segment n+1 and, thus, staying ahead of the Player's requests.

6 CONTENT LOCATION & MANIFESTS

6.1 Content Identification

The only signal from the Player to the Gateway that identifies the content to be streamed is the URL. Thus, the URL needs to include sufficient information to uniquely identify a stream which might be available via multicast.

An example URL for a video segment is:

```
http://linear.private.cableco.net/hls/AnEHD_HD_NAT_14710_0_6713276793419826123_HLS/format/hls/tra
ck/video/repid/root_video/bandwidth/2946000/frag/435173/asset/20141201T175607-01-543173live.ts
```

6.1.1 URL Encoding

Typically, the information sufficient to identify a stream which might be available via multicast includes – the domain of the server (i.e., is this operator provided content or not), a unique content identifier and a bit rate.

6.1.1.1 Domain

The M-ABR service is primarily designed to support operator provided content from an operator's CDN. Thus, the first check that a Gateway typically performs on a URL which might be of interest is on the domain/host of the URL being requested by a Player. If the URL matches one of the configured domains/hosts the URL will be processed further by the EMC. Otherwise the request is simply handled normally, independent of the EMC.

6.1.1.2 Channel Identifier/Linear Content Identifier

An identifier of the linear content itself (e.g., CNN, ESPNU, etc.) is also needed in the URL architecture designed by the operator. This can either be a human-readable, but unique string such as the channel name or it can be an encoding such as a numeric identifier.

6.1.1.3 Media Format

An identifier of the type of format of the Adaptive Bitrate is needed in the URL architecture designed by the operator. This should be a unique string such as HLS, DASH, etc.

6.1.1.4 Media Type

An identifier of the type of media is needed for content that is generated with separate audio and video tracks. This should be a unique string such as VIDEO, AUDIO, etc.

6.1.1.5 Representation ID or Bitrate

Finally, as a given content stream might only be available via multicast for certain representations, the representation also needs to be included in the URL. Instead of including the bit rate of the encoding itself in the URL, it is also possible to use a representation identifier. For example, if an operate provides video streams with multiple renditions or to represent multiple bit rates more abstractly (e.g., 1=lowest, 2=low, 3=medium, 4=high, 5=highest). This has the advantage of supporting renditions with the same bit rate or allowing the actual bit rate of the encoding to change independent of the URL. For example, using a Representation ID the highest encoded bit rate could change from 7 Mbps to 8 Mbps, but the Representation ID used in the URL could remain as "highest" (or 5, in the previous example).

6.1.2 URL Matching

PERL Compatible Regular Expressions (PCRE) are used to determine whether or not a stream is available via multicast. The domain, content identifier and bit rate are all compared to the URL via regular expressions. If there is a match against one of the Gateway's configured regular expressions, the content identifier and bit rate portions of the URL are then compared against the same fields in the channel map of available multicast content. The Gateway MUST support PCRE as defined in [PCRE].

The system is intended to be capable of functioning with just a single regular expression, but there may be reasons to support more than a single regular expression. For example, during a transition period between one URL format and another it might be desirable to have the Gateway support both the "old" URL style and the "new" URL style as content migrates from one system to the other. Thus, the system supports the option to configure more than one regular expression on the Gateway.

An example regular expression for matching against segment requests is:

```
http://linear\.private\.cableco\.net/hls/(?P<channelId>.*)/format/(?P<mediaformat>.*)/track/(?P<mediatype>.*)/repid/(?P<repid>.*)/bandwidth/.*)/frag/.*\.ts
```

Note this regular expression uses named capturing groups to extract the channelId, media format, media type and Representation ID from the URL.

6.2 Manifests

For the Gateway to be able to deliver segment files to the Player from its multicast cache, the Gateway needs to have the segment files before the Player requests them. Thus, to maximize the potential for a "cache hit" for multicast content, the Gateway needs to be at least one segment ahead of the Player in the ABR stream. To achieve this, either the Gateway trims segments from the manifest that it delivers to the Player or the system utilizes two different versions of the variant manifest – one version for Players and one version for Multicast Servers. In either case, the goal is the same – to keep the multicast delivery portion of the system a segment or two ahead of the unicast delivery portion of the system.

6.3 Multicast Zones

Given that cable operators perform local ad insertion, the same channelId and representation can still have multiple variations depending on the ad zone of the Gateway. Thus, the protocol communicates a multicast zoneId to the Gateway during configuration. The Gateway communicates this zoneId to the Multicast Controller (which may not be the same server as the configuration server) such that the Multicast Controller can provide the Gateway the proper Channel Map. The Multicast Controller can also use this information to help it decide what set of streams to direct the Multicast Server to provide via multicast.

Although this information is delivered to the Gateway, the Gateway itself does not generally utilize this information in its internal logic.

7 GATEWAY FUNCTIONALITY & PROTOCOL OPERATION

7.1 Gateway Configuration

To enable the Multicast ABR feature on a Gateway conformant with this specification, the Gateway needs to be provisioned with the IP address/DNS hostname of a Multicast Configuration Server (refer to Section 7.3.1). The Multicast Configuration Server address can be provisioned either through the DOCSIS CM configuration file or via TR-069.

If TLV TBD is present in the CM configuration file, the Gateway MUST utilize this address to send its initial ConfigReq message.

If, as part of TR-069 provisioning, the Gateway receives a MulticastConfigurationServerAddress (TBR) element, it MUST utilize this address to send its initial ConfigReq message, unless the Multicast Configuration Server was already configured via the CM configuration file, in which case this element is ignored.

7.2 Gateway HTTP Proxying & Caching

A fundamental requirement of the M-ABR system is that it is transparent to Players and requires no configuration on Players. Thus, the Gateway typically examines all GET requests and compares them against its configured URL regular expressions to determine if the request is relevant to M-ABR (i.e., whether or not the request is for a Multicast-Ready Stream). However, some solutions can exist to further optimize processing of requests at the gateway. If the gateway is capable of modifying its IP Tables to filter requests destined to specific IP addresses for further processing by the EMC, this can be enabled by providing a list of addresses in the Config Result message. Some systems exist whereby the Player can be redirected or otherwise manipulated to send requests directly to the Gateway itself. Therefore, while this is not the default mode, it is possible to disable transparent proxying on the Gateway. Unless proxy.transparency.disable is 'true', the Gateway MUST compare all GET requests that match the IP Tables against its configured URLRegex[3]. Matching a URLRegex can trigger various Gateway behaviors such as joining/leaving multicast groups and checking its internal cache for the requested segment file. The specific requirements for these behaviors are described elsewhere in this section.

The Gateway SHOULD support HTTP GETs with a RangeRequest for the specific missing portion of a segment file (e.g., a NORM FEC block). The Gateway MUST handle HTTP GETs with RangeRequests from the Player.

The Gateway receives video segments files per [MS-EMC]. The Gateway MUST cache at least one segment per multicast session. The Gateway MUST serve HTTP GET requests for segment files in its cache without sending an HTTP GET of its own. If the Gateway receives a request for a segment file (or portion thereof) which is not in its cache, the Gateway MUST send an HTTP GET for the missing portion of the file requested.

7.3 Protocol Operation

7.3.1 Configuration

As operators have a variety of policies and needs, this protocol was designed to be very flexible. There are a number of capabilities for which the Gateway informs the Multicast Controller and, in return, the Multicast Controller informs the Gateway of its set of operating parameters.

Similarly, different operators can have different IP Multicast deployment architectures. While, conceptually, this interface defines the communication between the Embedded Multicast Client in a Gateway and a Multicast Controller, protocol elements have been defined which allow operators to distribute aspects of Multicast Controller functionality across multiple servers. In particular, this specification defines a Multicast Configuration Server which is often the same server as the Multicast Controller, but can also be a separate server. The Multicast Configuration Server's functionality can be further distributed between an initial configuration server and a configuration refresh server. Again, the initial configuration server and the refresh server can be the same server or different servers.

[3] It is important to note that substantial performance optimizations exist. For example, the Gateway can extract address and port information from the URLRegex and utilize that information to reduce the number of GET requests which require full regex comparisons.

Thus, the specification allows operators to use one, two or three different server addresses for different aspects of the Multicast Controller functionality. Simple deployments likely use only a single server, but operators with larger deployments can choose to distribute this functionality across multiple servers if they prefer.

At startup the Gateway MUST send a ConfigReq (refer to Section 8.1.1) message to the Multicast Configuration Server address it was assigned via provisioning. If no Multicast Configuration Server address is assigned then the Multicast ABR feature is disabled on the Gateway.

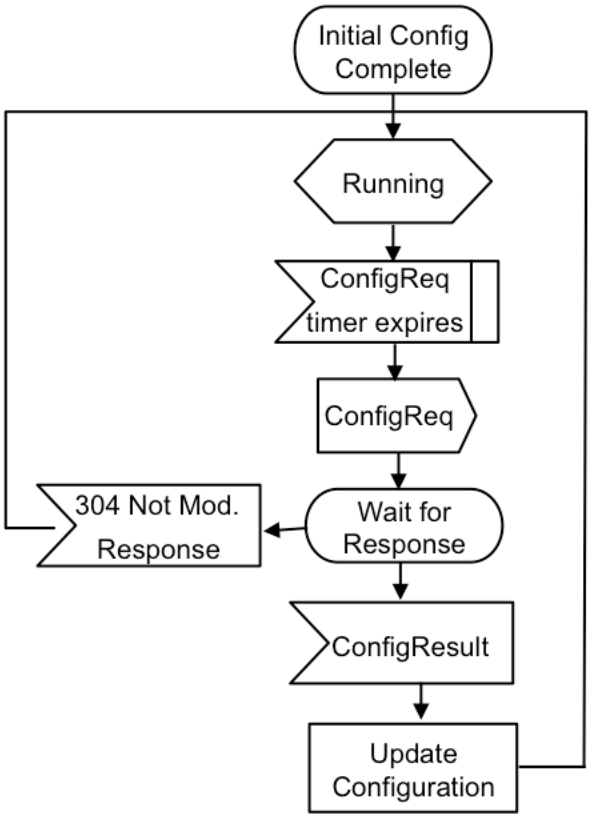

Figure 5 - Configuration State Machine

The Gateway MUST send a ConfigReq message every refresh.interval seconds after receiving its initial ConfigResult.

Upon receipt of a ConfigReq for a configuration which is new or has changed, the Multicast Configuration Server MUST send a ConfigResult (refer to Section 8.1.2) message containing the Gateway's current configuration. Upon receipt of a ConfigReq for a configuration which is unchanged, the Multicast Configuration Server MAY send a 304 Not Modified response. If the Gateway receives a 304 Not Modified response, it MUST continue using its current configuration.

As part of its ConfigResult the Multicast Configuration Server MAY include a refresh.server configuration element. If the refresh.server configuration element is included in the ConfigResult, the Gateway MUST utilize the refresh.server address for all subsequent ConfigReq messages. If the refresh.server configuration element is not included in the ConfigResult, the Gateway MUST utilize the Multicast Configuration Server for all subsequent ConfigReq messages.

The Gateway MAY support manifest manipulation (i.e., reducing the manifest by n segments). If the Gateway supports manifest manipulation it MUST set mfestManipSupport to "true" in its ConfigReq message. If the Gateway does not support manifest manipulation it SHOULD set mfestManipSupport to "false" in its ConfigReq message. If

a Gateway which does not support manifest manipulation receives a ConfigResponse with a non-zero mfest.segmentDrops value, the Gateway MUST log the error and continue operation.

The ConfigResult also includes the URL of the Multicast Controller assigned to this Gateway. The Gateway MUST send all other messages to this Multicast Controller.

As part of its ConfigResult the Multicast Configuration Server MAY include a multicast.channelMap.sourceAddress, multicast.channelMap.groupAddress and multicast.channelMap.port. This attribute indicates the (S,G) where ChannelMapMsgs are sent[4]. (Refer to the Channel Map Message section of the [MS-EMC] specification for the message definition.) Upon receipt of a ConfigResult message containing a multicast.channelMap.sourceAddress, multicast.channelMap.groupAddress and multicast.channelMap.port, the Gateway MUST join the corresponding (S,G). Upon joining this (S,G), the Gateway MUST replace any internally cached channel map with the most recent ChannelMapMsg sent to this address. (Note, the ChannelMap element can also be sent via the unicast StreamStatusResult message.) If any of the three multicast.channelMap configuration parameters is omitted, the Gateway MUST log an error, ignore the remaining multicast.channelMap parameters and continue operation.

7.3.2 Normal Operation

After completing configuration, the Gateway can start sending StreamStatus messages to the Multicast Controller. The transmission of StreamStatus message by the Gateway can be event-based (i.e. based on the event of a Player requesting a Multicast-Ready Stream, with an optional delay) and/or time-based (i.e., strictly based on time). Thus, as detailed in this and subsequent sections, the Gateway can function in one of three modes based on its configuration:

1. Time-Based – StreamStatus messages are strictly periodic and independent of Player requests. (Timer T1; streamStatus.interval > 0 and streamStatus.eventing.eventDelay < 0)
2. Event-Based – StreamStatus messages are triggered solely by Player requests for Multicast-Ready Streams. (Timer T2; streamStatus.interval <= 0 and streamStatus.eventing.eventDelay >=0)
3. Time- & Event-Based – Stream status messages are triggered either by events or by time depending on which protocol timer triggers first. (Timer T1 or T2; streamStatus.interval > 0 and streamStatus.eventing.eventDelay >= 0)

By default, the Gateway is in Time-Based mode and sends StreamStatus messages strictly periodically. However, to address different operational models the other modes are supported as well.

This section details the protocol requirements related to these configuration elements.

Upon receipt of its initial ConfigResponse, if the streamStatus.interval is > 0 then the Gateway MUST set timer T1 to streamStatus.interval seconds. If the streamStatus.interval is <= 0 then the Gateway MUST ignore timer T1. Upon the expiration of timer T1, the Gateway MUST send a StreamStatus message. Upon transmission of a StreamStatus message, the Gateway MUST set timer T1 to streamStatus.interval seconds, if timer T1 is in use. Refer to Figure 6.

When sending a StreamStatus (refer to Section 8.2.1) message, the Gateway MUST include the StreamState of all streams that match one of its configured URL regular expressions (URLRegex) and have been requested within streamStatus.active.threshold seconds. Streams that match the one of the configured URL regular expressions are streams that the Multicast Controller might be multicasting currently or might decide to multicast in the future, this document refers to these streams as Multicast-Ready Streams. When sending a StreamStatus message, the Gateway SHOULD include StreamHistory elements for Multicast-Ready Streams requested since the last StreamStatus message transmission. The Gateway MUST limit the number of StreamHistory elements to the streamStatus.history.depth count. If the Gateway has received request for more than streamStatus.history.depth Multicast-Ready Streams since its last StreamStatus transmission, the Gateway SHOULD include the StreamHistory elements for the streams with the largest number of requests. A value of zero for streamStatus.history.depth indicates that the history feature is disabled and the Gateway MUST NOT include any StreamHistory elements in its StreamStatus message. Zero is the default value of streamStatus.history.depth.

[4] While standard mechanisms are defined for the Multicast Controller to POST a ChannelMap to a Multicast Server for multicast transmission, it is possible that the Multicast Controller or other device multicasts the ChannelMapMsg to this (S,G) via NORM.

If the value of streamStatus.eventing.eventDelay is greater 0 then, after every Player request for an unjoined Multicast-Ready Stream, the Gateway MUST set timer T2 to streamStatus.eventing.eventDelay seconds. If timer T2 expires before another request from the same Player for an unjoined Multicast-Ready Stream or the streamStatus.eventing.eventDelay is equal to 0, then the Gateway MUST send the Multicast Controller a StreamStatus message. Upon transmission of a StreamStatus message, the Gateway MUST ignore timer T2 until it is set again by another request for an unjoined Multicast-Ready Stream.

The StreamStatusResult (refer to Section 8.2.2) and ChannelMapMsg messages can include a ChannelMap element indicating a set of Multicast-Active Streams (i.e., streams which are actively being transmitted to an (S,G) by the Multicast Server). The Gateway MUST store the most recent ChannelMap received via either message. The Gateway MUST use this ChannelMap for associating channelId-bit rate tuples with multicast addresses when joining the multicast group for a given multicast stream. The Gateway MUST use this ChannelMap for identifying streams which are no longer actively being transmitted via multicast and can be left (per the rules in Section 7.4).

The StreamStatusResult message can directly/indirectly control the multicast group membership of the Gateway as described in Section 7.4. Upon receipt of a StreamStatus message, if the set of Multicast-Active Streams is new or modified, the Multicast Controller MUST send a StreamStatusResult message. Upon receipt of a StreamStatus message, if its ChannelMap is unmodified, the Multicast Controller MAY send a 304 Not Modified response or omit the ChannelMap in its StreamStatusResult. If the Gateway receives a 304 Not Modified response or no ChannelMap, it MUST continue using its current ChannelMap.

7.4 Multicast Group Membership Control

The AssignedStreamList (refer to Section 8.2.2.1.3) is the set of multicast streams the Gateway is assigned to. The AssignedStreamList can be explicitly signaled by the Multicast Controller for centralized control of group membership or the Multicast Controller can allow the Gateway to participate in determining which multicast groups to maintain.

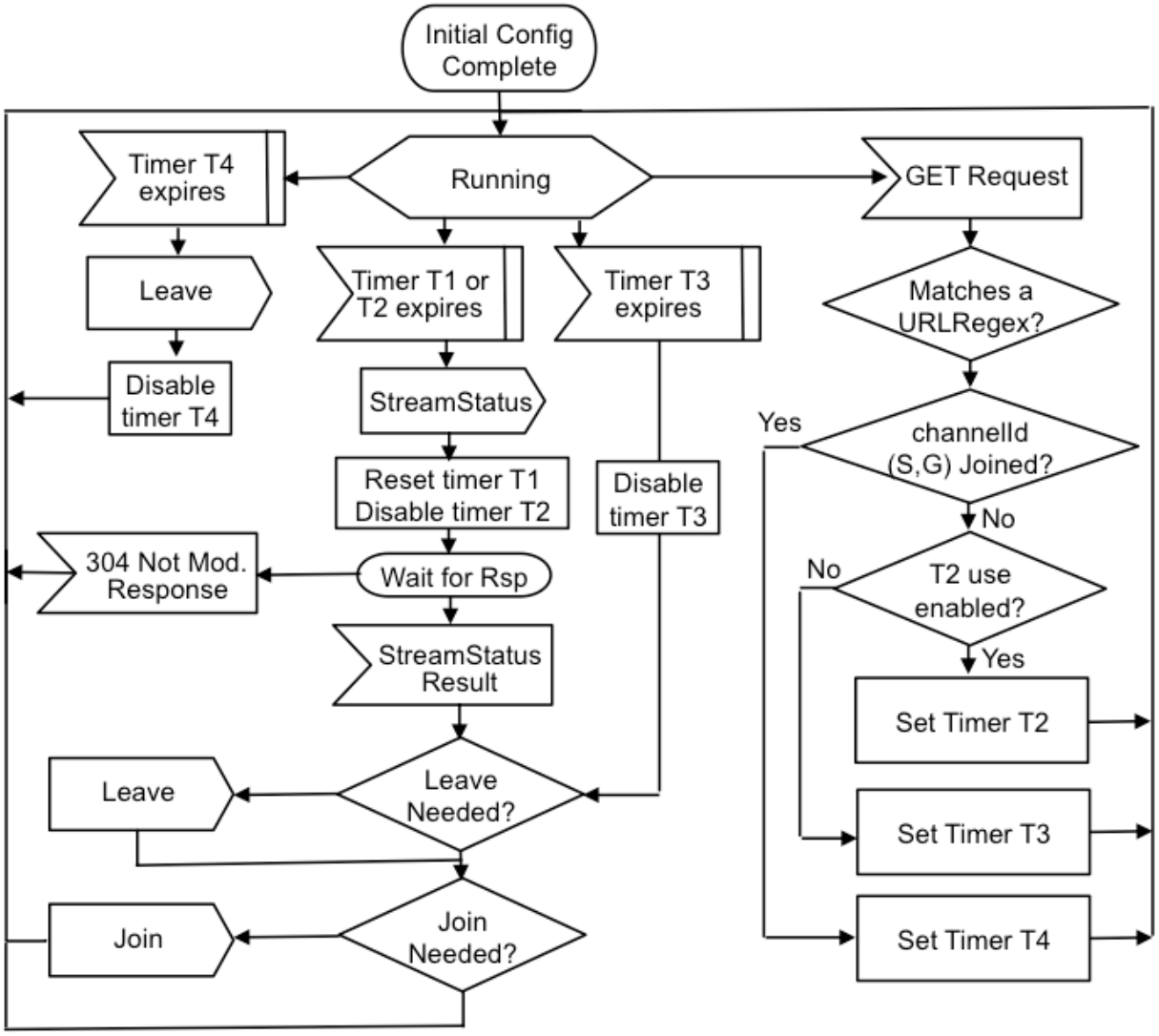

Figure 6 - StreamStatus & Group Membership State Machine

7.4.1 Join & Leave Triggers

The Multicast Controller can either explicitly or implicitly control the multicast group membership of the Gateway. With implicit control, some membership decisions could be made by the Gateway itself.

7.4.1.1 Explicit Triggers

If the Multicast Controller receives a StreamStatus message from a Gateway indicating that a Player is requesting segments for a Multicast-Active Stream, the Multicast Controller MAY explicitly signal AssignedStreamList in its StreamStatusResult. If the AssignedStreamList no longer contains the (S,G) for a group which the Gateway is currently a member, then the Gateway MUST perform an IGMPv3 or MLDv2 Leave for that (S,G). If the

AssignedStreamList contains the (S,G) for a group which the Gateway is not currently a member, then the Gateway MUST perform an IGMPv3 or MLDv2 Join for that (S,G).

7.4.1.2 Implicit Triggers

If a Player has been requesting segments from a Multicast-Active Stream for multicast.start.delay seconds (this is timer T3 in Figure 6), the last StreamStatusResult did not contain an AssignedStreamList and the Gateway does not have sufficient resources to receive that stream via multicast then the Gateway MAY leave one of its current multicast groups. The Gateway SHOULD factor the timeSinceLastRetrieval for each of its multicast streams when deciding which group to leave. For example, the multicast stream with the largest timeSinceLastRetrieval value is the least recently used stream and its associated multicast group could be left.

If a Player has been requesting segments from a Multicast-Active Stream for multicast.start.delay seconds, the last StreamStatusResult did not contain an AssignedStreamList and the Gateway has sufficient resources to receive that stream, then the Gateway MUST perform an IGMPv3 or MLDv2 Join for the associated (S,G).

If a Player has not requested a segment for a Multicast-Active Stream for multicast.stop.delay seconds (this is timer T4 in Figure 6), then the Gateway MUST leave the multicast group associated with that stream.

7.5 Multicast Content Delivery

The Multicast Controller controls the set of streams transmitted by the Multicast Server. After a Gateway successfully joins a multicast group it starts receiving ABR video segment files via multicast and caching them for subsequent retrieval by the Player.

A variety of events could occur which prevent the Gateway from receiving multicast video segment files as anticipated or which result in Players not requesting video segment files which are successfully being received via multicast. The StreamStatus message contains two attributes which can help the Multicast Controller or other management system recognize these situations – the multicastRxBytes and cacheHitBytes. The Gateway MUST include these counters when reporting the StreamStatus of streams with a deliveryState of ‘multicast’.

Video segment files are delivered to the Gateway’s EMC as described in [MS-EMC].

8 MC-EMC INTERFACE DEFINITION

8.1 Gateway Configuration

Gateway configuration occurs at initial boot up, but can also occur periodically such that the configuration of a Gateway can be refreshed after initialization.

8.1.1 Configuration Request (ConfigReq)

The ConfigReq message is used to identify the Gateway to the Multicast Configuration System (MCS) and to communicate any capabilities from the Gateway to the MCS. Capabilities will be sent using URL parameters.

Request Direction: EMC to MCS

Method: HTTP GET

Message URL: `http://<device-url>/mcs/ConfigReq/<deviceId>`

Children: Capability (0..N)

Table 1 - ConfigReq Parameter Definitions

Parameter	Use	Data Type	Description
deviceId	Required	xs:string	The MAC address of the transmitting Gateway in xx-xx-xx-xx-xx-xx form.

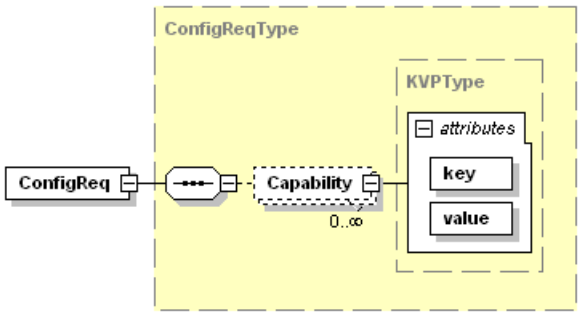

Figure 7 - ConfigReq

8.1.1.1 Capability

A Capability element contains a key and a value attribute as defined in the common KVPType.

Table 2 - Standard Capabilities

Key	Use	Description
maxMcastBitrate	Optional	The maximum total multicast delivery bit rate this Gateway can receive.
maxStreamRate	Optional	The maximum encoded bit rate the Gateway can support for an individual stream.
maxMcastSessions	Optional	The maximum number of M-ABR sessions this Gateway can receive.
mfestManipSupport	Optional	Whether or not the Gateway supports manifest manipulation. Valid values are “true” and “false”.
vendorId	Optional	The Organization Unique Identifier of the Gateway MAC address.
model	Optional	This string's syntax is that used by the Gateway vendor to identify hardware models.
softwareVersion	Optional	This string's syntax is that used by the Gateway software vendor to identify software versions.
defaultRoute	Optional	The IPv4 or IPv6 address of the Gateway’s default route.

All standard Gateway capabilities are in the form "[key]=[value]". For example: http://<device-url>/mcs/ConfigReq/<deviceId> ?maxMcastBitrate="16000000"&maxMcastSessions="5". The MCS MUST ignore capabilities it does not recognize. The MCS MUST NOT respond with an error when it receives a capability which it does not recognize.

8.1.2 Configuration Result (ConfigResult)

On a successful processing of a ConfigReq, the Multicast Configuration System (MCS) MUST return 200 OK with an XML response carrying the ConfigResult element, detailing the Gateway's configuration information. The ConfigResult element is defined in Section 8.1.2.1.

If the MCS does not have a record for the requested deviceId, the MCS MUST return a 404 Not Found error.

On standard HTTP protocol errors, the MCS MUST return the appropriate HTTP status code and status text.

If the request fails due to reasons other than unknown deviceId or standard HTTP protocol errors, the MCS MUST return a status code of 500 Internal Server Error and provide a ConfigResult element with details about the failure in its Response element.

Table 3 - ConfigReq Response Codes

Status Code	Meaning
200	OK
400	Bad Request
404	Not Found
500	Internal Server Error

8.1.2.1 ConfigResult Message

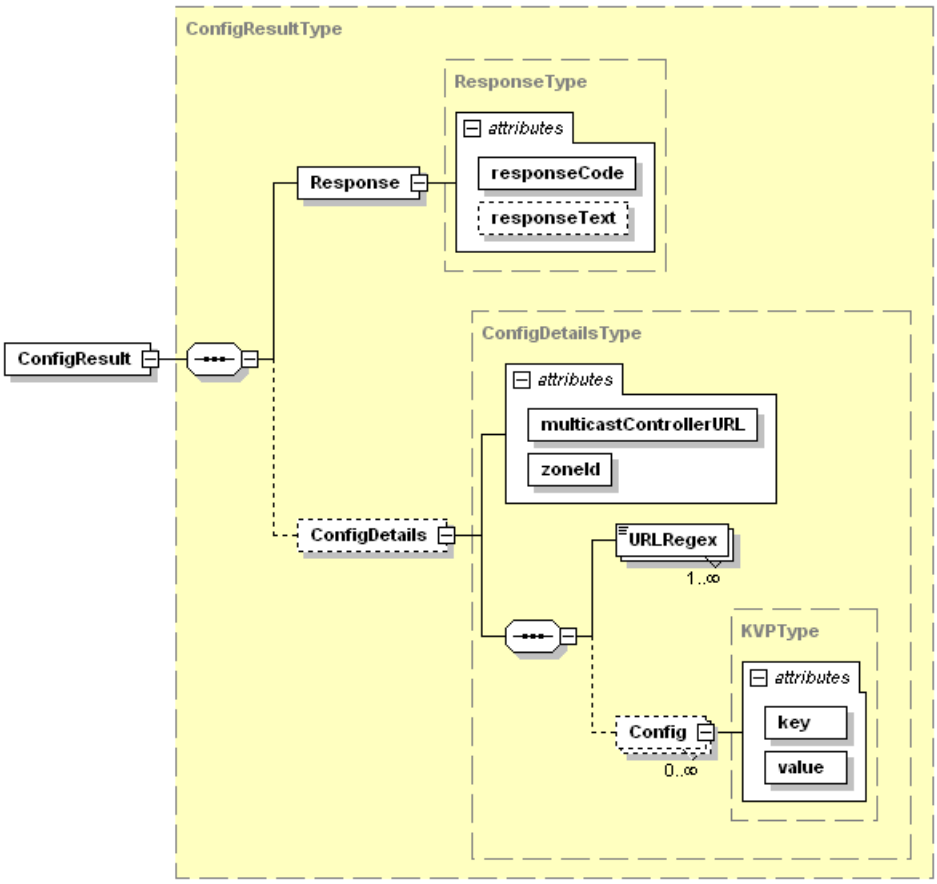

Figure 8 - ConfigResult

Children: Result (1)
ConfigDetails (0..1)

8.1.2.1.1 Response

The Response element is defined in Section 10.2.6. The Multicast Controller MUST provide a response code value of 200 on a successful completion of the Config request. Standard Config failure response codes are detailed in the Section 11.1.

8.1.2.1.2 ConfigDetails

Table 4 - ConfigResult Attribute Definitions

Attribute	Use	Data Type	Description
multicastControllerURL	Required	xs:integer	The URL of the multicast controller assigned to this Gateway.
zoneId	Required	xs:string	The multicast zone assigned to this Gateway.

Children: URLRegex (1..n)
Config (0..n)

8.1.2.1.3 URLRegex

The URLRegex element contains a regular expression the Gateway will use to match content requests.

Children: None

Table 5 - URL Regex Element Definition

Data Type	Description
xs:string	Regular expression for URL matching.

8.1.2.1.4 Config

The Config element contains a Gateway configuration item.

Children: None

Common Type: KVPType

Table 6 - Standard Configurations

Key	Use	Description	Default
streamStatus.interval	Optional	Number of seconds the Gateway waits between sending StreamStatus messages to the Multicast Controller. A value of -1 indicates that the Gateway MUST NOT send time-based StreamStatus messages.	300secs
multicast.start.delay	Optional	Number of seconds the Gateway waits after the first segment is delivered on a given stream before joining the multicast stream.	30secs
multicast.stop.delay	Optional	Number of seconds the Gateway waits between the last GET for a segment on the stream and when it leaves the multicast stream.	7200secs
multicast.channelMap.sourceAddress	Optional	The S part of the (S,G) for the multicast address where the Gateway listens for ChannelMapMsgs. This is either an IPv4 address, an IPv6 address or a FQDN.	none
multicast.channelMap.groupAddress	Optional	The G part of the (S,G) for the multicast address where the Gateway listens for ChannelMapMsgs. This is either an IPv4 address, an IPv6 address or a FQDN.	none
multicast.channelMap.port	Optional	The port for the (S,G) for the multicast address where the Gateway listens for ChannelMapMsgs.	none
multicast.norm.unicastNackServer	Optional	If configured, the Gateway sends unicast NORM NACKs to this address.	none
refresh.server	Optional	Address of a server for refreshing configurations during run time.	Same as MCS
refresh.interval	Optional	Number of seconds the Gateway waits between configuration refreshes.	86400
mfest.segmentDrops	Optional	Number of segments the Gateway is to drop from the manifest if it supports manifest manipulation.	0
streamStatus.active.threshold	Optional	Number of seconds since the most recent GET request for a Multicast-Ready Stream segment before the Gateway considers the stream inactive in its StreamStatus reporting.	30secs
streamStatus.history.depth	Optional	The number of channel change history events to include in StreamStatus messages.	0
streamStatus.eventing.eventDelay	Optional	The number of seconds to wait after a channel change to a Multicast-Ready Stream before sending a StreamStatus. A value of -1 indicates that the Gateway MUST NOT send event-based StreamStatus messages.	-1

Key	Use	Description	Default
proxy.transparency.disable	Optional	A boolean indicating whether or not the Gateway should disable transparent proxying. The default is 'false' indicating that transparent proxying enabled. A value of 'true' indicates that the Player has been manipulated to request segments directly from the Gateway and, thus, the transparent proxy function in the Gateway is not needed.	false
ip.tables.list	Optional	A list of IP addresses and ranges that the Gateway will perform HTTP GET processing on to determine REGEX matches.	none

All standard Gateway configurations are in the form "gateway.config.[key]=[value]". The Gateway MUST ignore configuration keys it does not recognize. The Gateway MUST log an error when it receives a configuration key which it does not recognize and continue operation.

8.1.3 Configuration Request/Result Examples

8.1.3.1 Configuration Request

```
GET /mcs/ConfigReq/01-de-ca-fb-ad-01maxMcastBitrate="16000000"
maxMcastSessions="2"
maxStreamRate="8000000"
mfestManipSupport="true"
vendorId="0aa00a"
model="HAL-9000"
softwareVersion="1.2.3"
defaultRoute="10.0.0.1"
```

8.1.3.2 Configuration Result Example: Success

```
HTTP/1.1 200 OK
Server: mcs 1.1
Cache-Control: no-cache
<ConfigResult
  xmlns="urn:com:cablelabs:ipmulticast:2015:02:13">
  <Response responseCode=”200”/>
  <ConfigDetails
    zoneId="zone1"
    multicastControllerURL="http://mc101.mabr.tvcdn.net/mc/StreamStatus">
    <URLRegex>http://linear\.private\.cableco\.net/hls/(?P<channelId>.*)/format/.*/bandwidth/
(?P<bitrate>.*)/frag/.*\.ts</URLRegex>
    <Config
      key="gateway.config.streamStatus.interval"
      value="180"/>
    <Config
      key="gateway.config. multicast.start.delay"
      value="60"/>
  </ConfigDetails>
</ConfigResult>
```

8.1.3.3 Configuration Result Example: Internal Server Error

```
HTTP/1.1 500 Internal Server Error
Server: mcs 1.1
Cache-Control: no-cache
<ConfigResult
  xmlns="urn:com:cablelabs:ipmulticast:2015:02:13">
  <Response responseCode=”701”
    responseText=”MC-EMCI Error: missing required capability.”/>
</ConfigResult>
```

8.2 Gateway Streaming Status

8.2.1 Stream Status Inform (StreamStatus)

As described in Section 7.3.2, the Gateway StreamStatus message to the Multicast Controller either periodically or in response to content requests (depending on its configuration). The purpose of sending the StreamStatus is to indicate that the Gateway is still operational and to communicate the relevant video streams that Players behind this Gateway are viewing.

Request Direction: Gateway to Multicast Controller

Method: HTTP POST

Message URL: `http://<device-url>/mc/StreamStatus`

Children: StreamState (0..n)
StreamHistory (0..n)

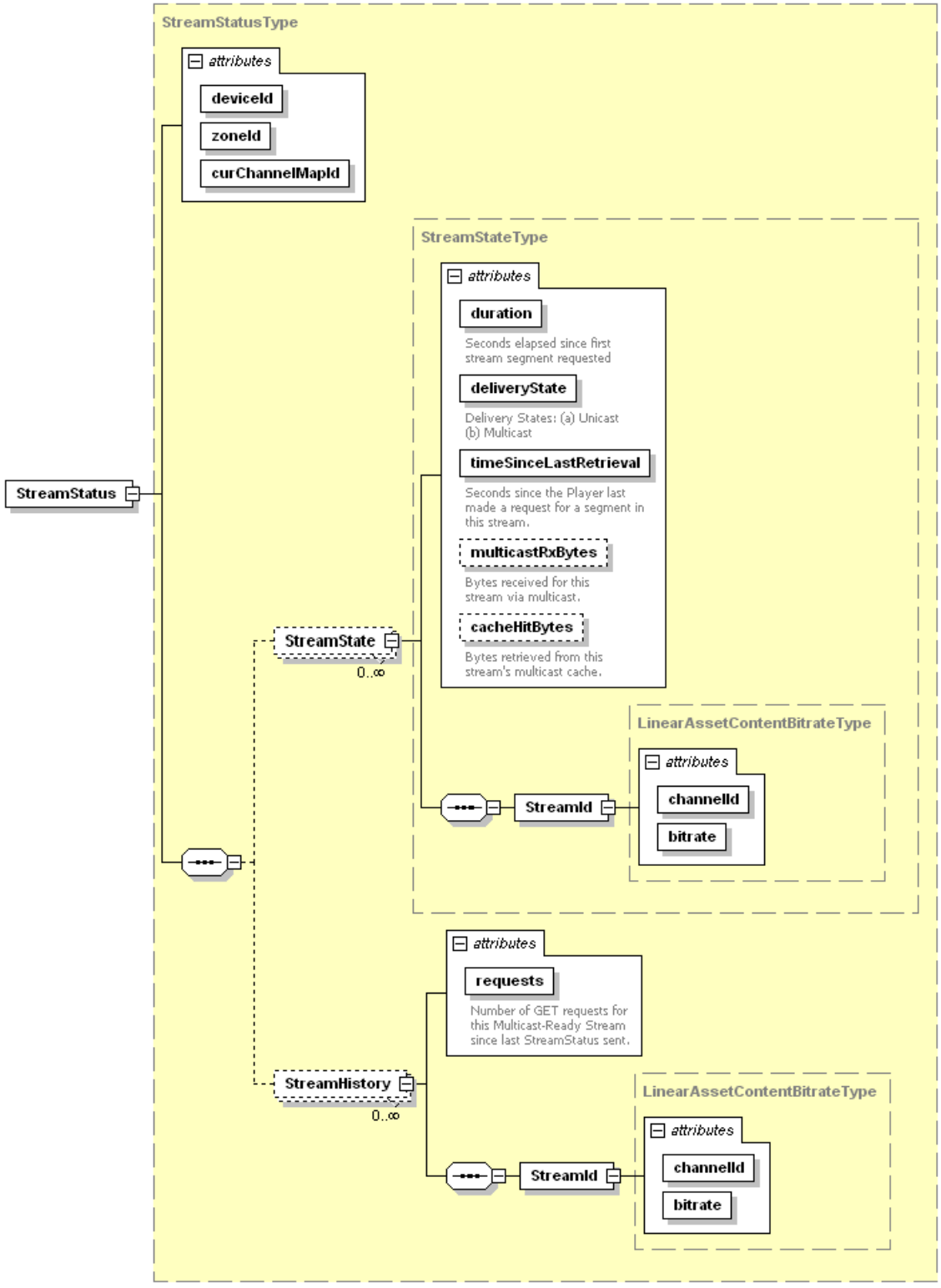

Figure 9 - StreamStatus

Table 7 - StreamStatus Attribute Definitions

Attribute	Use	Data Type	Description
deviceId	Required	xs:string	The MAC address of the transmitting Gateway in xx-xx-xx-xx-xx-xx form.
zoneId	Required	xs:string	A unique identifier for a multicast zone.
curChannelMapId	Required	xs:string	The unique identifier for a channel map from the most recently received ChannelMap (whether received in a StreamStatusResult or a ChannelMapMsg).

8.2.1.1 StreamState

This element represents the state of a given Multicast-Ready Stream (i.e., stream who's URL matches against a configured URLRegex) being streamed by this Gateway.

Children: StreamId (1) refer to Section 10.2.4

Table 8 - StreamState Attribute Definitions

Attribute	Use	Data Type	Description
Duration	Optional	xs:string	Seconds elapsed since the first segment in this stream was requested.
deliveryState	Required	StreamDelivery Type enum	The delivery state of the associated Multicast-Ready Stream. If the Gateway is currently joined to the corresponding multicast group for the stream then the Gateway MUST indicate the deliveryState as "multicast". Otherwise, the Gateway MUST indicate the deliveryState for the stream as "unicast".
timeSinceLast Retrieval	Required	xs:unsignedInt	Seconds since the Player last made a request for a segment in this stream.
multicastRxBytes	Optional	xs:unsignedInt	The number of bytes successfully received for this stream via multicast. (i.e., unerrored FEC bytes).
cacheHitBytes	Optional	xs:unsignedInt	The number of bytes retrieved from this stream's multicast cache.

8.2.2 Stream Status Response

On successful processing of a StreamStatus request, the Multicast Controller MUST return a 200 OK with an XML response carrying the StreamStatusResult element or a 304 Not Modified response. The StreamStatusResult element is defined in Section 8.2.2.1.

If the Gateway specifies an unknown deviceId, the Multicast Controller must return a status code of 404 Not Found. Upon receipt of a 404 Not Found, the Gateway SHOULD reinitialize and re-perform initial configuration.

On standard HTTP protocol errors, the Multicast Controller MUST return the appropriate HTTP status code and status text.

If the request fails due to reasons other than unknown deviceId or standard HTTP protocol errors, the Multicast Controller MUST return a status code of 500 Internal Server Error and provide the StreamStatusResult element in the XML body with details about the failure in a Response element. Other failure responses include the HTTP status code and status text that specifies the error.

Table 9 - StreamStatus Response Codes

Status Code	Meaning
200	OK
304	Not Modified
400	Bad Request
404	Not Found
500	Internal Server Error

8.2.2.1 StreamStatusResult Message

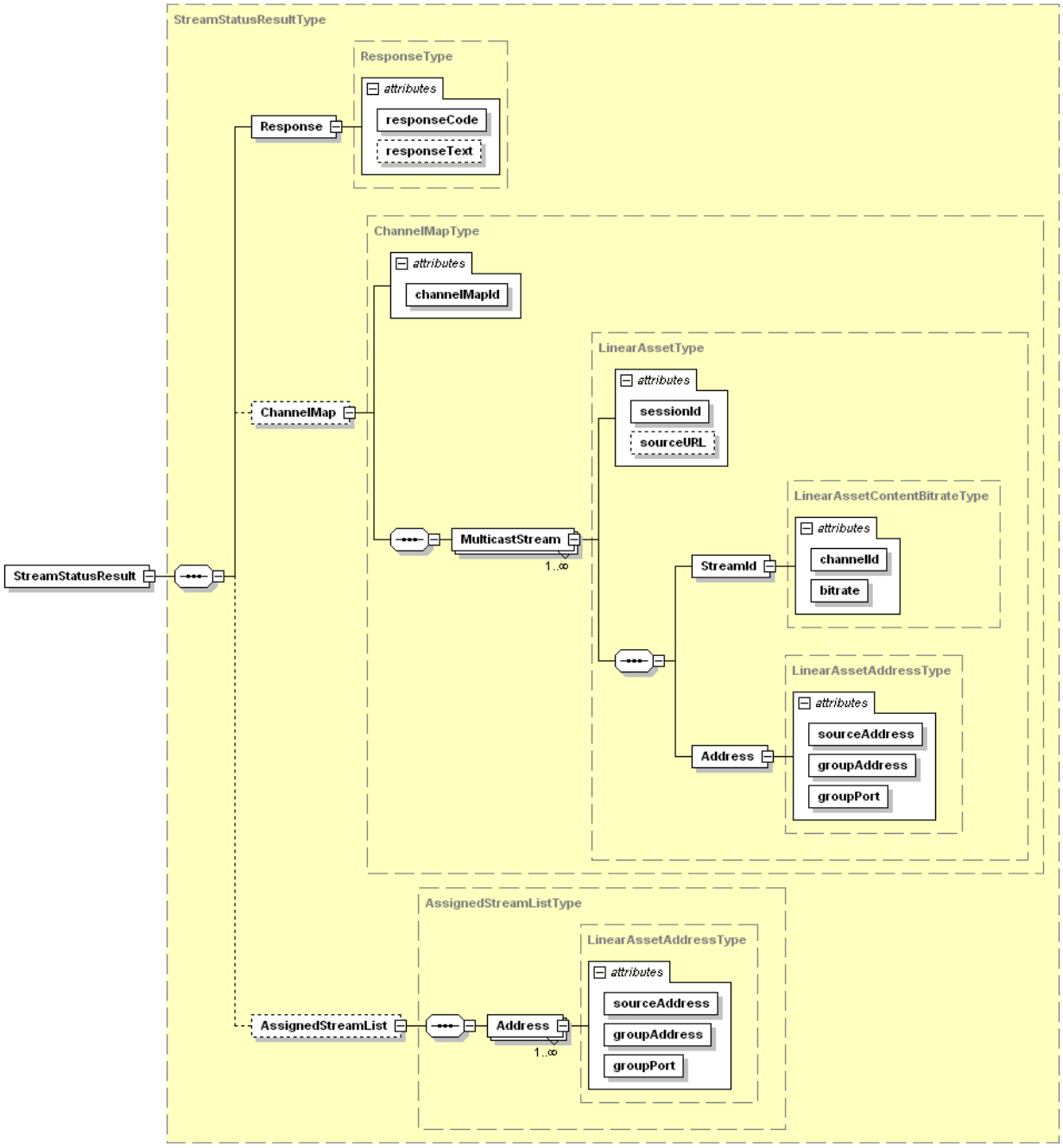

Figure 10 - StreamStatusResult

Children: Response (1)
ChannelMap (0..1)
AssignedStreamList (0..1)

8.2.2.1.1 Response

The Response element is defined in Section 10.2.6. The Multicast Controller MUST provide a response code value of 200 on a successful completion of the StreamStatus request. Standard StreamStatus failure response codes are detailed in the Section 11.1.

8.2.2.1.2 ChannelMap

The ChannelMap element contains a series of MulticastStreams which provide a mapping between a tuple of channelId and bit rate to (S,G). It is defined in Section 10.2.5.

8.2.2.1.3 AssignedStreamList

The AssignedStreamList element contains a set of (S,G) addresses that the Multicast Controller is commanding the Gateway to be a member of. It is defined in Section 10.2.3.

8.2.3 StreamStatus Message Examples

8.2.3.1 StreamStatus Request Example

```
POST /mc/StreamStatus HTTP/1.1
Host: mc
User-Agent: emc
Content-Type: application/xml
Content-Length: ...
<StreamStatus
  xmlns="urn:com:cablelabs:ipmulticast:2015:02:13"
  deviceId="01-de-ca-fb-ad-01"
  zoneId="zone1">
  <StreamState
    duration="876"
    timeSinceLastRetrieval="5"
    deliveryState="multicast">
    <StreamId
      channelId="ESPN2HD"
      bitrate="3800000"/>
  </StreamState>
</StreamStatus>
```

8.2.3.2 StreamStatusResult Example

```
HTTP/1.1 200 OK
Server: MC 1.0
Cache-Control: no-cache
<StreamStatusResult
  xmlns="urn:com:cablelabs:ipmulticast:2015:02:13">
  <Response responseCode=”200”/>
  <ChannelMap>
    <MulticastStream
      sourceURL="https://www.tvcdn.net/espn2.m3u8?bitrate=3800&channelId=ESPNHD"
      sessionId="6c68ebc0-6ab0-11e4-b116-123b93f75cba">
      <StreamId
        channelId="ESPNHD"
        bitrate="3800000"/>
      <Address
        groupAddress="224.1.1.1"
        groupPort="12345"
        sourceAddress="10.10.10.10"/>
    </MulticastStream>
    <MulticastStream
      sourceUrl=" https://www.tvcdn.net/espn2.m3u8?bitrate=3800&channelId=ESPN2HD"
      sessionId="502faa5a-6a41-4384-8254-d4ff13f48f60">
      <StreamId
        channelId="ESPN2HD" bitrate="3800000"/>
      <Address
```

```
        groupAddress="224.1.1.2"
        groupPort="12345"
        sourceAddress="10.10.10.10"/>
    </MulticastStream>
  </ChannelMap>
  <AssignedStreamList>
    <Address
      groupAddress="224.1.1.1"
      groupPort="12345"
      sourceAddress="10.10.10.10"/>
  </AssignedStreamList>
</StreamStatusResult>
```

8.2.3.3 StreamStatus Result Example: Internal Server Error

```
HTTP/1.1 500 Internal Server Error
Server: MC 1.0
Cache-Control: no-cache
<StreamStatusResult
  xmlns="urn:com:cablelabs:ipmulticast:2015:02:13">
  <Response responseCode=”703”
    responseText=”MC-EMCI Error: incorrect zoneId.”/>
</StreamStatusResult>
```

9 HTTP PROTOCOL

9.1 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.

For example, on the MC-EMC interface, the Multicast Controller is the server and the Embedded Multicast Client is the client. However, on the MC-MS interface, the Multicast Controller is the client and the Multicast Server is the server. Since the device which is the client varies depending on which interface is being considered, this section uses the generic language of "client" and "server" as it applicable to both interfaces.

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.

9.1.1 Connection Security

For communications between client and server components in the IP Multicast 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.

9.2 Request Messages

9.2.1 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.

9.2.2 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 instead of component or application names.

Table 10 - Application Abbreviations

Component or Application Name	app_abbreviation
Multicast Controller	mc
Multicast Server	ms

Note: the Embedded Multicast Client is not listed as the protocol does not require server support on this device.

The *operation* indicates the task to perform. Operations are defined with each of the individual messages.

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 Multicast Server state data (multicast). The Multicast Controller can issue an HTTP GET to retrieve the state details from a Multicast Server it manages, or the Multicast Controller can issue an HTTP POST to start a multicast stream. In either case, the operation is simply the name "multicast". Notice that it does not include "Get" or "Set".

The full HTTP headers for the “multicast” method are:

```
POST http://<device>/ms/multicast HTTP/1.1
GET http://<device>/ms/multicast 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.

9.2.3 HTTP Version

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

9.2.4 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 the If-Modified-Since header in requests when the request has been made previously. If the client includes the If-Modified-Since header, it MUST include the time of the last response (i.e., the time of its current state) in the request. If the server returns a response code of 304 Not Modified, the client MUST continue to use its existing state with respect to that request.

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 /mc/StreamStatus HTTP/1.1
Host: mc
User-Agent: emc
Content-Type: application/xml
Content-Length: ...
```

9.2.5 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.

Unlike [RMI HTTP] where each interface has its own namespace, for IP Multicast interfaces the functions across the interfaces are different enough that only a single namespace was used for all of the interfaces. The namespace used in the XML is shown below:

```
xmlns="urn:com:cablelabs:ipmulticast:2015:02:13"
```

There is no message body for an HTTP GET message.

9.2.6 Message Body – JSON

TBD

9.2.7 GET and POST Request Message Examples

GET Request Message Example:

```
GET /mcs/ConfigReq/01-de-ca-fb-ad-01 HTTP/1.1
Host: mcs
User-Agent: emc
Content-Type: application/xml
Content-Length: ...
<ConfigReq
  xmlns="urn:com:cablelabs:ipmulticast:2015:02:13">
  <Capability
    key="gateway.capability.maxMcastBitrate"
    value="7600000"/>
  <Capability
    key="gateway.capability.maxMcastSessions"
    value="2"/>
</ConfigReq>
```

POST Request Message Example

```
POST /mc/StreamStatus HTTP/1.1
Host: mc
User-Agent: emc
Content-Type: application/xml
Content-Length: ...
<StreamStatus
  xmlns="urn:com:cablelabs:ipmulticast:2015:02:13"
  deviceId="01-de-ca-fb-ad-01"
  zoneId="zone1">
  <StreamState
    duration="876"
    timeSinceLastRetrieval="5"
    deliveryState="multicast">
    <StreamId
      channelId="ESPN2HD"
      bitrate="3800000"/>
  </StreamState>
</StreamStatus>
```

9.3 Response Messages

9.3.1 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

9.3.2 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).

9.3.3 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: MC 1.0
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: MC 1.0
Cache-Control: no-cache
```

9.3.4 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 operation name specified in the URI of the HTTP POST request, is used with "Result" appended to the end. For example, if the URI of the POST is "/mc/StreamStatus", then the root element of the XML in the response message body will be "StreamStatusResult". 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 9.2.5.

9.3.5 Message Body – JSON

TBD

9.4 Message Flow

9.4.1 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 11 shows the message flow sequence between the client and server when the connection is closed after a single request-response pair.

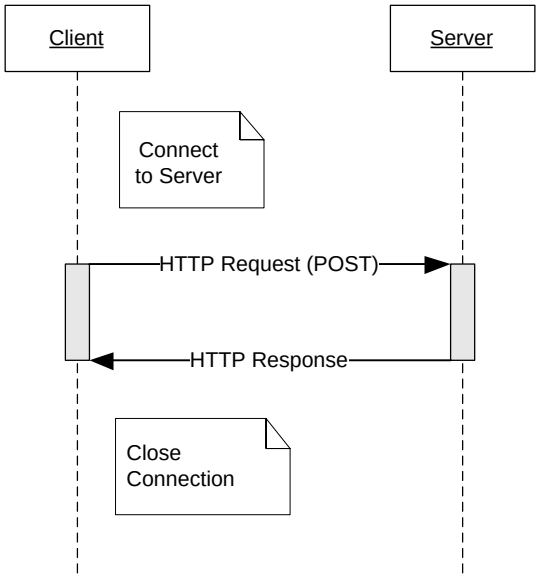

Figure 11 - Message Flow: Semi-Persistent Connection

Figure 12 shows the message flow sequence between the client and the server when the connection remains open between request-response pairs.

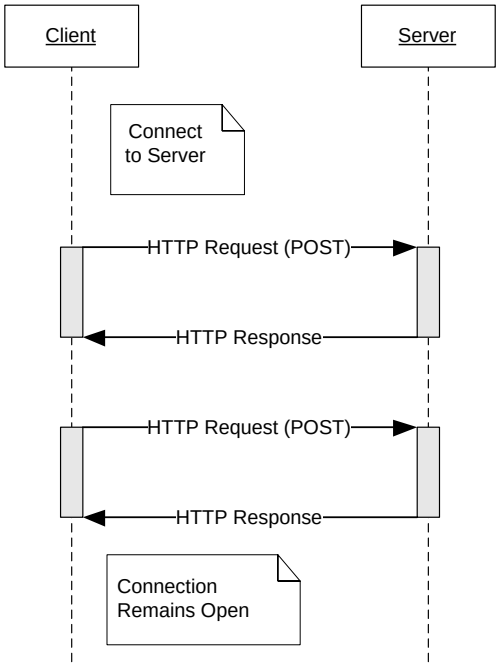

Figure 12 - Message Flow: Persistent Connection

9.4.2 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.

9.4.3 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.

10 COMMON XML ELEMENTS

10.1 Common Simple Data Types

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

Table 11 - Common Data Types

Data Type Name	Base Type	Type Constraints	Description
IPOrDomainType	xs:string	Refer to DVB SDNS v1.4	A dotted decimal IPv4 address or a domain name.
StreamDeliveryType	xs:string	"unicast" "multicast"	Indicates whether a Multicast-Ready Stream is currently being received via unicast or multicast.

10.2 Complex Data Elements

This section contains complex data types that have been created to support the IPM HTTP protocol.

10.2.1 KVPType

The KVPType allows for the encoding of key-value pairs in the protocol. It is loosely typed as both the key and the value portion are encoded as strings, but the encoded value can be of a different data type.

Table 12 - KVPType Attribute Definitions

Attribute	Use	Data Type	Description
key	Required	xs:string	The key of the key-value pair.
value	Required	xs:string	The value of the key-value pair.

10.2.2 LinearAsset

The LinearAsset object provides stream and address information for a given linear asset multicast session.

Children: LinearAssetContentBitrate (1)

LinearAssetAddress (1)

Table 13 - LinearAsset Attribute Definitions

Attribute	Use	Data Type	Description
sessionId	Required	xs:string	Globally unique identifier for this session
sourceURL	Optional	xs:string	URL of the top level manifest.

10.2.2.1 Example LinearAsset XML

```
<MulticastStream
  sessionId="6c68ebc0-6ab0-11e4-b116-123b93f75cba"
  sourceURL="https://www.tvcdn.net/espn2.m3u8?bitrate=3800&channelId=ESPNHD">
  <StreamId
    channelId="ESPN"
    bitrate="3800000"/>
  <Address
    groupAddress="224.1.1.1"
    groupPort="65535"
    sourceAddress="10.10.10.10"/>
</MulticastStream>
```

10.2.3 LinearAssetAddress

Table 14 - LinearAssetAddress Attribute Definitions

Attribute	Use	Data Type	Description
groupAddress	Required	dvb:IPOrDomainType	
groupPort	Required	xs:unsignedShort	Port number
sourceAddress	Required	dvb:IPOrDomainType	URL of the master manifest.

10.2.3.1 Example LinearAssetAddress XML

```
<Address
  groupAddress="224.1.1.1"
  groupPort="12345"
  sourceAddress="10.10.10.10"/>
```

10.2.4 LinearAssetContentRepresentation

Table 15 - LinearAssetContentRepresentation Attribute Definitions

Attribute	Use	Data Type	Description
channelId	Required	xs:string	Globally unique identifier for this linear asset
mediaFormat	Required	xs:string	ABR format for this linear asset (HLS, DASH)
mediaType	Optional	xs:string	Media Type (VIDEO, AUDIO
repid	Required	xs:unsignedLong	Representation Identifier for this linear asset

10.2.4.1 Example LinearAssetContentRepresentation XML

```
<StreamId
  channelId="ESPN"
  repid="3800000"/>
```

10.2.5 ChannelMap

The ChannelMap object can be embedded in a number of different messages and can be delivered both unicast and multicast.

Children: MulticastStream (1..n) (of LinearAssetType described above)

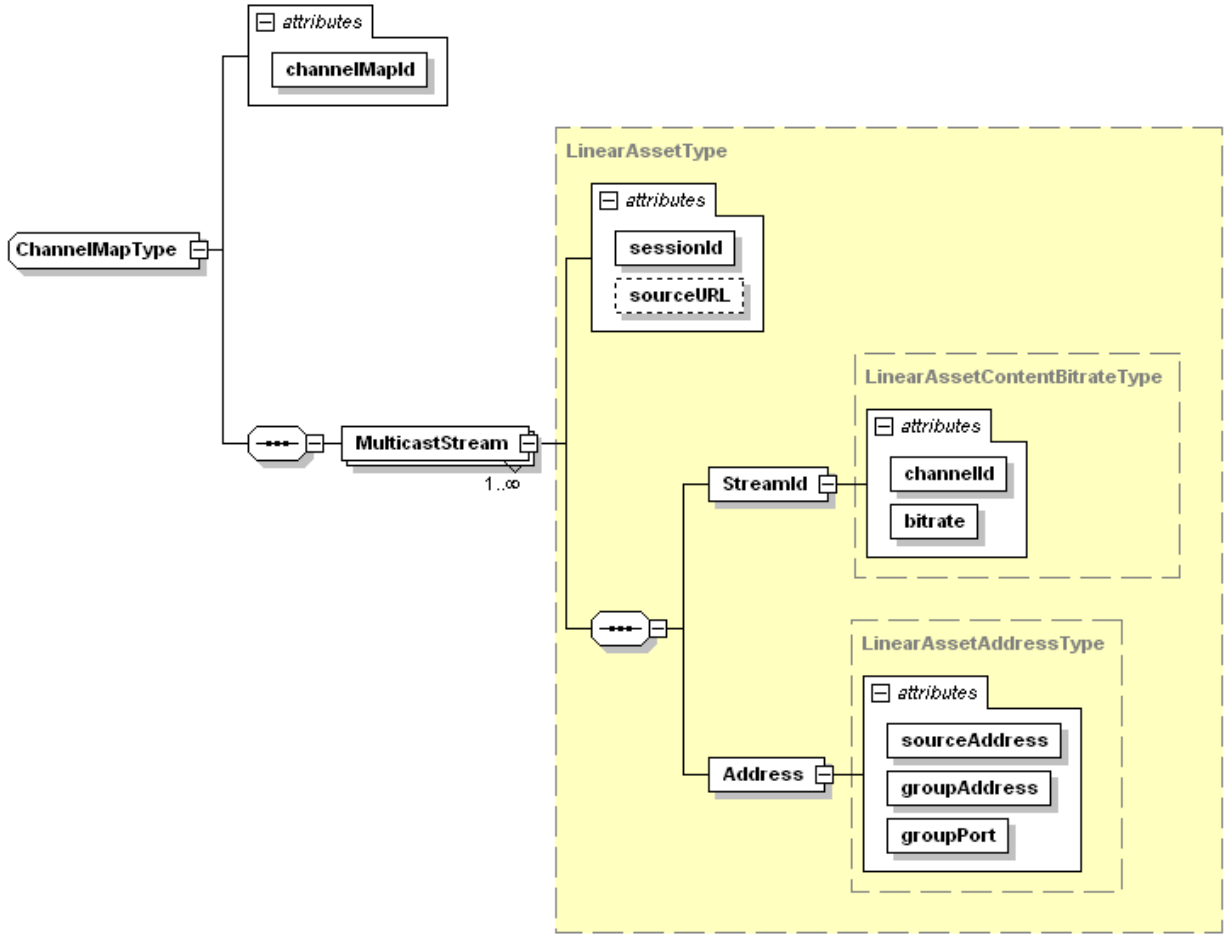

Figure 13 - ChannelMap Element

Table 16 - ChannelMap Attribute Definitions

Attribute	Use	Data Type	Description
channelMapId	Required	xs:string	Globally unique identifier for this channel map.

10.2.5.1 Example ChannelMap XML

```
<ChannelMap
  channelMapId=”0589d210-ad55-11e4-89d3-123b93f75cba”>
  <MulticastStream
    sourceURL="https://www.tvcdn.net/espn2.m3u8?bitrate=3800&channelId=ESPNHD"
    sessionId="6c68ebc0-6ab0-11e4-b116-123b93f75cba">
    <StreamId
      channelId="ESPNHD"
      bitrate="3800000"/>
    <Address
      groupAddress="224.1.1.1"
      groupPort="12345"
      sourceAddress="10.10.10.10"/>
  </MulticastStream>
  <MulticastStream
    sourceUrl=" https://www.tvcdn.net/espn2.m3u8?bitrate=3800&channelId=ESPN2HD"
    sessionId="502faa5a-6a41-4384-8254-d4ff13f48f60">
    <StreamId
```

```
      channelId="ESPN2HD"
      bitrate="3800000"/>
    <Address
      groupAddress="224.1.1.2"
      groupPort="12345"
      sourceAddress="10.10.10.10"/>
  </MulticastStream>
</ChannelMap>
```

10.2.6 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 17 - 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.

10.2.6.1 Response Element XML Example

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

11 COMMON CODES

11.1 Response Codes

Table lists common response codes for errors in the IP Multicast (IPM) system. These codes are used in the responseCode attribute of the Response element.

Table 18 - Response Codes

Code	Description
200	No Error
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
	600-range MC-MS Interface Errors
600	MC-MSI Error: Other
601	MC-MSI Error: Invalid sourceAddress specified.
602	MC-MSI Error: Invalid groupAddress specified.
602	MC-MSI Error: Error retrieving manifestUrl.
603	MC-MSI Error: Commanded bit rate not included in manifest.
604	MC-MSI Error: Error retrieving stream variant manifest.
605	MC-MSI Error: Error retrieving first stream segment.
606	MC-MSI Error: Error sending first segment to directed multicast group
607	MC-MSI Error: Unable to support requested FEC parameters.
608	MC-MSI Error: Encoded bit rate is less than specified multicast rate.
610	MC-MSI Error: Error contacting accessServer.
611	MC-MSI Error: CDN token-related error.
	700-range MC-EMC Interface Errors
701	MC-EMCI Error: missing required capability.
702	MC-EMCI Error: capabilities incompatible with multicast service offering.
703	MC-EMCI Error: incorrect zoneId.
	800-range Vendor-Specific Errors
800-899	Reserved for vendor-specific response codes.

Annex A Considerations for Hybrid STBs (Normative)

Hybrid set-tops are set-tops that support both QAM-based/CAS-based video delivery and IP-based/DRM-based video delivery. The primary issue with this type of device is: how does it know when to get a given linear content feed/channel via QAM-based delivery versus IP-based delivery?

The system described in the IP Multicast specifications essentially overlays multicast on a unicast IP ABR delivery architecture as an optimization. Thus, it fails back to unicast delivery whenever there is a multicast-related delivery issue. This mechanism is fairly fundamental to the system and makes it difficult for a hybrid set-top to use QAM-based video delivery as a fallback for IP-multicast video delivery as unicast IP video delivery already plays that role in the system.

However, using QAM-based video as the primary video source and then falling back to IP-based video fits well within the specified IP multicast. The details of how this would be achieved are beyond the scope of this specification, but it is recommended that operators/vendors considering incorporating IP Multicast into a hybrid STB and using these interface specifications consider that approach.

Another specification suite, the Converged-Policy, Session and Resource Management specification suite, is being developed to address related issues. It is recommended that parties interested in hybrid STBs look at these specifications as well when considering their architecture.

Annex B Schema (Normative)

```
<xs:schema xmlns="urn:com:cablelabs:ipmulticast:2015:02:13"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="urn:com:cablelabs:ipmulticast:2015:02:13"
xmlns:dvb="urn:dvb:metadata:iptv:sdns:2008-1"
targetNamespace="urn:com:cablelabs:ipmulticast:2015:02:13" elementFormDefault="qualified"
attributeFormDefault="unqualified">
  <xs:import namespace="urn:dvb:metadata:iptv:sdns:2008-1" schemaLocation="./sdns_v1.4r13.xsd"/>
  <!-- Simple Types & Attributes -->
  <xs:simpleType name="StreamDeliveryType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="unicast"/>
      <xs:enumeration value="multicast"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="MulticastStreamStatusType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="not-started"/>
      <xs:enumeration value="running"/>
      <xs:enumeration value="stopped"/>
      <xs:enumeration value="error"/>
      <xs:enumeration value="failed"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="LinearAssetAddressType">
    <xs:attributeGroup ref="SSMAddressType"/>
  </xs:complexType>
  <xs:complexType name="LinearAssetContentBitrateType">
    <xs:attributeGroup ref="ContentStreamBitrateType"/>
  </xs:complexType>
  <xs:complexType name="KVPType">
    <xs:attributeGroup ref="KVPAttType"/>
  </xs:complexType>
  <xs:complexType name="CapabilityType">
    <xs:sequence/>
  </xs:complexType>
  <xs:complexType name="ConfigType">
    <xs:sequence/>
  </xs:complexType>
  <xs:attributeGroup name="SSMAddressType">
    <xs:attribute name="sourceAddress" type="dvb:IPOrDomainType" use="required"/>
    <xs:attributeGroup ref="McastAddressType"/>
  </xs:attributeGroup>
  <xs:attributeGroup name="McastAddressSrcOptType">
    <xs:attribute name="sourceAddress" type="dvb:IPOrDomainType" use="optional"/>
    <xs:attributeGroup ref="McastAddressType"/>
  </xs:attributeGroup>
  <xs:attributeGroup name="McastAddressType">
    <xs:attribute name="groupAddress" type="dvb:IPOrDomainType" use="required"/>
    <xs:attribute name="groupPort" type="xs:unsignedShort" use="required"/>
  </xs:attributeGroup>
  <xs:attributeGroup name="SourceManifestRateType">
    <xs:attribute name="manifestUrl" type="xs:string" use="required"/>
    <xs:attribute name="bitrate" type="xs:string" use="optional"/>
  </xs:attributeGroup>
  <xs:attributeGroup name="NormOptionsType">
    <xs:attribute name="fecEnable" type="xs:boolean" use="optional"/>
    <xs:attribute name="fecBlockSize" type="xs:unsignedByte" use="optional"/>
    <xs:attribute name="fecRepairCount" type="xs:unsignedByte" use="optional"/>
  </xs:attributeGroup>
  <xs:attributeGroup name="MulticastTxOptionsType">
    <xs:attribute name="multicastDscp" type="xs:unsignedByte" use="optional"/>
    <xs:attribute name="multicastRate" type="xs:unsignedInt" use="optional"/>
  </xs:attributeGroup>
  <xs:attributeGroup name="ContentStreamBitrateType">
    <xs:attribute name="channelId" type="xs:string" use="required"/>
    <xs:attribute name="bitrate" type="xs:integer" use="required"/>
```

```
</xs:attributeGroup>
<xs:attributeGroup name="KVPAttType">
  <xs:attribute name="key" type="xs:string" use="required"/>
  <xs:attribute name="value" type="xs:string" use="required"/>
</xs:attributeGroup>
<!-- Complex Types -->
<xs:complexType name="ChannelMapType">
  <xs:sequence>
    <xs:element name="MulticastStream" type="LinearAssetType" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="channelMapId" use="required"/>
</xs:complexType>
<xs:complexType name="AssignedStreamListType">
  <xs:sequence>
    <xs:element name="Address" type="LinearAssetAddressType" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="StreamStatusResultType">
  <xs:sequence>
    <xs:element name="Response" type="ResponseType"/>
    <xs:element name="ChannelMap" type="ChannelMapType" minOccurs="0"/>
    <xs:element name="AssignedStreamList" type="AssignedStreamListType" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="ConfigReqType">
  <xs:sequence>
    <xs:element name="Capability" type="KVPType" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <!-- xs:attribute name="gatewayId" type="xs:string" use="required"/ -->
</xs:complexType>
<xs:complexType name="ResponseType">
  <xs:attribute name="responseCode" type="xs:integer" use="required"/>
  <xs:attribute name="responseText" type="xs:string" use="optional"/>
</xs:complexType>
<xs:complexType name="ConfigDetailsType">
  <xs:sequence>
    <xs:element name="URLRegex" type="xs:string" maxOccurs="unbounded"/>
    <xs:element name="Config" type="KVPType" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="multicastControllerURL" type="xs:string" use="required"/>
  <xs:attribute name="zoneId" type="xs:string" use="required"/>
</xs:complexType>
<xs:complexType name="ConfigResultType">
  <xs:sequence>
    <xs:element name="Response" type="ResponseType"/>
    <xs:element name="ConfigDetails" type="ConfigDetailsType" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="LinearAssetType">
  <xs:sequence>
    <xs:element name="StreamId" type="LinearAssetContentBitrateType"/>
    <xs:element name="Address" type="LinearAssetAddressType"/>
  </xs:sequence>
  <xs:attribute name="sessionId" type="xs:string" use="required"/>
  <xs:attribute name="sourceURL" type="xs:string"/>
</xs:complexType>
<xs:complexType name="MulticastStatusType">
  <xs:attribute name="status" type="MulticastStreamStatusType" use="required"/>
  <xs:attribute name="sessionId" type="xs:string" use="required"/>
  <xs:attribute name="sourceAddress" type="dvb:IPOrDomainType" use="required"/>
  <xs:attribute name="errorMsg" type="xs:string" use="optional"/>
  <xs:attribute name="errorTime" type="xs:dateTime" use="optional"/>
  <xs:attribute name="bytesSent" type="xs:unsignedLong" use="optional"/>
  <xs:attribute name="lastSegmentFileSent" type="xs:string" use="optional"/>
</xs:complexType>
<xs:complexType name="MulticastStatusResultType">
  <xs:sequence>
```

```
      <xs:element name="Setup" type="StartMulticastReqType"/>
      <xs:element name="Status" type="MulticastStatusType"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="MulticastStatusListType">
    <xs:sequence>
      <xs:element name="Status" type="MulticastStatusResultType" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="StreamStatusType">
    <xs:sequence>
      <xs:element name="StreamState" type="StreamStateType" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="StreamHistory" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="StreamId" type="LinearAssetContentBitrateType"/>
          </xs:sequence>
          <xs:attribute name="requests" use="required">
            <xs:annotation>
              <xs:documentation>Number of GET requests for this Multicast-Ready Stream since last
StreamStatus sent.</xs:documentation>
            </xs:annotation>
          </xs:attribute>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="deviceId" type="xs:string" use="required"/>
    <xs:attribute name="zoneId" type="xs:string" use="required"/>
    <xs:attribute name="curChannelMapId" type="xs:string" use="required"/>
  </xs:complexType>
  <xs:complexType name="StreamStateType">
    <xs:sequence>
      <xs:element name="StreamId" type="LinearAssetContentBitrateType"/>
    </xs:sequence>
    <xs:attribute name="duration" type="xs:integer" use="required">
      <xs:annotation>
        <xs:documentation>Seconds elapsed since first stream segment requested</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="deliveryState" type="StreamDeliveryType" use="required">
      <xs:annotation>
        <xs:documentation>Delivery States: (a) Unicast (b) Multicast</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="timeSinceLastRetrieval" type="xs:integer" use="required">
      <xs:annotation>
        <xs:documentation>Seconds since the Player last made a request for a segment in this
stream.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="multicastRxBytes" type="xs:unsignedInt">
      <xs:annotation>
        <xs:documentation>Bytes received for this stream via multicast.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="cacheHitBytes" type="xs:unsignedInt">
      <xs:annotation>
        <xs:documentation>Bytes retrieved from this stream's multicast cache.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
  </xs:complexType>
  <xs:complexType name="StartMulticastDetailsType">
    <xs:attribute name="sessionId" type="xs:string" use="required"/>
    <xs:attribute name="statusMsg" type="xs:string" use="optional"/>
    <xs:attributeGroup ref="SSMAddressType"/>
  </xs:complexType>
  <xs:complexType name="SendChannelMapDetailsType">
```

```
    <xs:attribute name="statusMsg" type="xs:string" use="optional"/>
    <xs:attributeGroup ref="SSMAddressType"/>
  </xs:complexType>
  <xs:complexType name="SendChannelMapReqType">
    <xs:sequence>
      <xs:element name="ChannelMap" type="ChannelMapType"/>
    </xs:sequence>
    <xs:attributeGroup ref="McastAddressSrcOptType"/>
  </xs:complexType>
  <xs:complexType name="ChannelMapMsgType">
    <xs:sequence>
      <xs:element name="ChannelMap" type="ChannelMapType"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="SendChannelMapResultType">
    <xs:sequence>
      <xs:element name="Response" type="ResponseType"/>
      <xs:element name="SendChannelMapDetails" type="SendChannelMapDetailsType" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="StartMulticastReqType">
    <xs:attributeGroup ref="McastAddressSrcOptType"/>
    <xs:attributeGroup ref="SourceManifestRateType"/>
    <xs:attributeGroup ref="NormOptionsType"/>
    <xs:attributeGroup ref="MulticastTxOptionsType"/>
    <xs:attribute name="multicastManifestEnable" type="xs:boolean" use="optional"/>
    <xs:attribute name="accessServer" type="dvb:IPOrDomainType" use="optional"/>
  </xs:complexType>
  <xs:complexType name="StartMulticastResultType">
    <xs:sequence>
      <xs:element name="Response" type="ResponseType"/>
      <xs:element name="StartMulticastDetails" type="StartMulticastDetailsType" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="MulticastStatusListResultType">
    <xs:sequence>
      <xs:element name="MulticastStatus" type="MulticastStatusResultType" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <!-- Message Elements -->
  <xs:element name="ConfigReq" type="ConfigReqType"/>
  <xs:element name="ConfigResult" type="ConfigResultType"/>
  <xs:element name="StreamStatus" type="StreamStatusType"/>
  <xs:element name="SendChannelMapReq" type="SendChannelMapReqType"/>
  <xs:element name="SendChannelMapResult" type="SendChannelMapResultType"/>
  <xs:element name="ChannelMapMsg" type="ChannelMapMsgType"/>
  <xs:element name="StreamStatusResult" type="StreamStatusResultType"/>
  <xs:element name="StartMulticastReq" type="StartMulticastReqType"/>
  <xs:element name="StartMulticastResult" type="StartMulticastResultType"/>
  <xs:element name="MulticastStatusResult" type="MulticastStatusResultType"/>
  <xs:element name="MulticastStatusListResult" type="MulticastStatusListResultType"/>
</xs:schema>
```

Appendix I Acknowledgements (Informative)

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

Contributor	Company Affiliation
Bill Blum	ARRIS
Dan Torbet	ARRIS
Sangeeta Ramakrishnan	Cisco
Coby Young	Comcast
Bart De Vleeschauwer	Telenet

Matt White – CableLabs

Andrew Sundelin – Consultant to CableLabs

Appendix II Revision History

The following ECN was incorporated into OC-SP-MC-EMCI-I02-160923:

ECN Identifier	Accepted Date	Title of EC	Author
MC-EMCI-N-16.1884-1	9/19/2016	Updates to MC-EMC specification	Torbet