HIKVISION PSIA扩展协议
DVR&NVR扩展部分分
Version 0.5
Revision 1
22 December 2010

Revision History	Description	Date	By
Version 0.5 Revision 1	Initial version	2010-12-22	孟宏

Disclaimer

Content

1 Introduction

This document specifies an interface that enables physical security and video management systems to communicate with a Recording and Content Management(RaCM) device in a standardized way.

2 Conformance

The RaCM Device will host PSIA compliant services and adhere to the PSIA Service Model.

3 Glossary and Relationship

3.1 Glossary of Terms

- ***Channel*** = handle/tag for an input source (port or stream); see IPMD spec for usage. For a DVR, a "channel" is the identifier, or handle, used to identify a local input stream (which may also be accessible to remote entities via a local /PSIA/Streaming/channel/<id> Resource). The "stream" may contain Video, Audio, and/or Metadata. For an NVR, a "channel" is used to identify a remote media stream which, if from an IPMD, should come from the remote device's /PSIA/Streaming/channels/<id> Resource. **Within the RaCM Device, such input 'channels' are not explicitly configured.** The term 'channel' is used as a guiding principle and logical construct. There is no single, explicit RaCM Resource that exists for the purposes to allow an external entity to manage (create, update, delete) 'channels'. However a 'channel' is implicitly created when a Track is created to record from a 'source' (local or remote). In effect, when a Track is created, the <SourceDescriptor> creates a logical input 'channel'. If a 2nd Track is created that records from the same source (i.e. same Device and stream URL), then it is simply referring to the same logical 'channel' that the 1st Track referred to. **These logical 'channels' are visible (Read-Only fashion) via the "/PSIA/ContentMgmt/status/channels" Resource.** It is also possible to search based on these 'channels' via "/PSIA/ContentMgmt/search" Resource.
- ***Track*** = Virtual storage container for a specific type of content (e.g. MPEG4, G.726, etc.). *[Channels and tracks are kept separate to allow the*

ability to mix and match channels totracks]. **RESTRICTION:** At this time, a Track's configuration contains only 1 <SourceDescriptor> which is intended to describe the source for the recorded media-stream. This media-stream will be construed as the equivalent of the input "channel" for the track, which means that each Track can only record one input stream. The media-stream however can be Multi-Media, if the source delivers such a stream (to be found at the <SrcUrl>), as is the case with IPMD. Even with legacy 3rd party Cameras (e.g. Axis cameras that support audio), a single RTSP URL can deliver a Multi-Media streaming session (via 2 RTP streams for the single RTSP Session).

- ***Source*** = Any input media device is a 'source', whether the input is a hardware oriented 'port' (e.g. NTSC/PAL, audio input jack, etc.), or an IP-based media device. All 'sources' are identified by ISO/IEC 9834-8:2005 128-bit UUIDs/GUIDs to guarantee uniqueness within any given system. Using the RaCM nomenclature specified herein, Sources send their data to RaCM devices on via 'streams' which are mapped to 'channels.' Channels are handles used to identify the specific input streams for a RaCM device. Channels that are recorded are then mapped onto 'tracks' for recording. See the following diagram for more details. **Within the RaCM Device, such 'sources' are not explicitly configured. These logical 'sources' are visible (Read-Only fashion) via the "/PSIA/ContentMgmt/status/sources" Resource.** It is also possible to search based on these 'sources' via "/PSIA/ContentMgmt/search" Resource.
- ***Metadata*** = Content based on PSIA metadata/event specification. This includes events, alarms, etc.
- ***Service*** = an intelligent REST resource.
- ***Resource*** = A directly addressable REST object.
- ***URI*** = a virtual path that specifically identifies a REST resource; this path must follow the service/resource hierarchy (see PSIA Service Model specification).
- ***Segment*** = A general term addressing a single, contiguous portion of a media track, or (in some cases) a stream. Basically, a 'media clip' that is less than the whole of a respective media track or stream, yet is individually accessible via one of the mechanisms specified in this document.

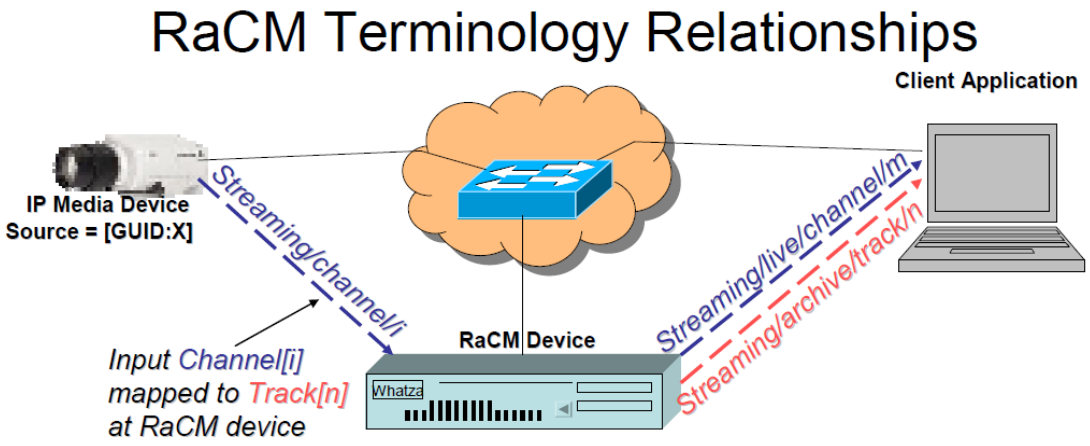
3.2 Relationship of Entities and Terminology

Below are the basic relationships between the terms defined above.

- **Source** = the device that is the origin point for input to a RaCM device. This is usually a camera, encoder (video and/or audio) or metadata generator. All devices have their own ISO/IEC 9834-8/ITU X.667, 128-bit UUID/GUID as their base identity.

- **Channel** = incoming media stream (input identifier). This is a handle/identifier for a specific input stream. If a device generated an audio and a video stream, which a RaCM device is recording, a 'channel' is used to identify each specific stream type (see 'Stream' below).
- **Stream** = in general, a network-based output media connection (output identifier). Basically, any PSIA device that outputs a media data stream onto a network is outputting a 'stream' for each media type. For example, an IP camera that supports two video resolutions, and audio, has the ability to generate 3 'streams'; one for each of the video resolutions/rates and one for the audio. This corresponds to the RTSP/SDP model. One session managing multiple streams. DVRs, which contain their own encoders, generate an internal data stream that is mapped to a channel identifier.
- **Track** = a recorded channel. Since input characteristics may change over time, 'tracks' are the virtual containers for recorded content associated with a 'channel'. Channels may be extant or extinct with respect to a track. That is why tracks are separate from channels. Channels are mapped to tracks for recording.

The following diagram depicts the relationships between streams, channels and tracks.



IP Media Device advertises its Streams by channels. Based on configuration the RaCM device selects the IPMD's stream channel ID (i) which is indigenous to the IP Media Device. This input stream becomes Channel 'm' on the RaCM device (since it has multiple inputs) for this particular live stream. Via configuration it is Mapped to track 'n' for recording. The track ID 'n' may, or may not, be equal to 'm'. When a client requests a live stream from the RaCM device, it uses the advertised Channel ID 'm'. When it requests a stream from the archived track, it uses the Advertised track ID 'n'. Searches based on the Source ID of 'GUID:X' will result in A list of all the tracks that match the IPMD's Source ID (GUID:X). In this case, it is Track 'n' whose track configuration references the source channel 'm'.

3.3 Arbitrary Stream and Track Associations

The RaCM specification does not currently support the maintenance of logical collections of streams from different sources or even multiple streams from the same source. Such abstractions are best performed by a higher-level, external entity (e.g. VMS); though track "grouping" may be supported in future release of RaCM. Bundling an Audio and Video stream together is already supported by the basic definition of a stream from the same source; however associating a Video stream from 1 camera to an Audio stream from another camera or to a POS (Point of Sale) Metadata stream is not supported. In general, the burden such abstractions should be avoided by RaCM, unless they fall out naturally from the existing data definitions. The external VMS entity can easily create such associations within its database.

3.4 XML Reserved Characters

Within an XML document, some characters are reserved for language use. If these characters appear in data values, they should be replaced with their Entity Reference equivalents (akin to ANSI Escapes) to avoid parsing errors.

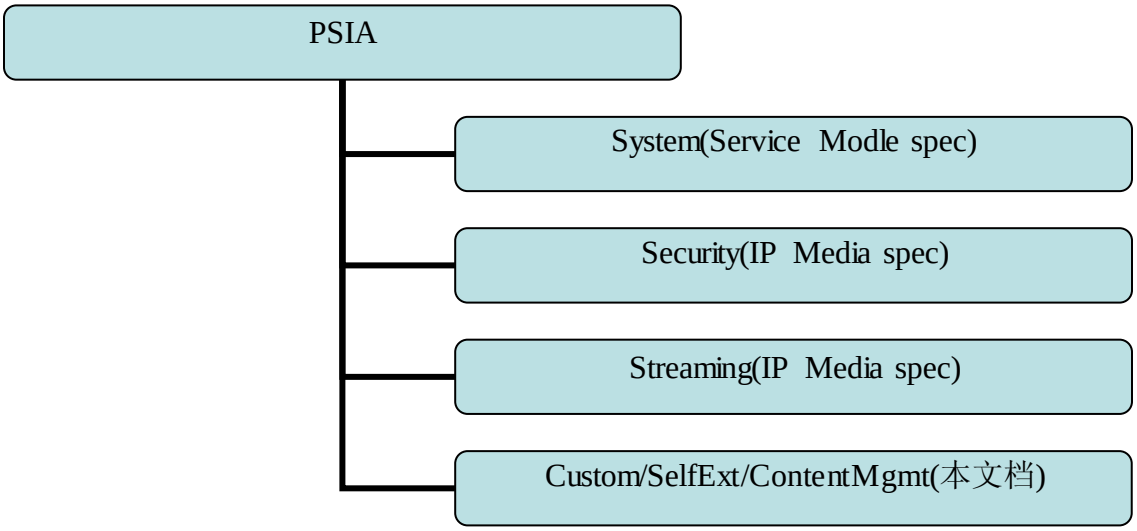
Character	Description	Entity Reference	Comments
<	Less than	&lt;	May never appear in data.
&	Ampersand	&amp;	May never appear in data.
>	Greater than	&gt;	Replace as best practice.
"	Double quote	&quot;	Replace as best practice.
'	Single quote	&apos;	Replace as best practice.
%	Percent	&#37;	Replace as best practice.
Note that &#0; (null) is not permitted.			

For Example, the URL "rtsp://144.70.13.92:554/PSIA/Streaming/tracks/27?offset=a07724**&**endtime=2009-05-18T10:31.25" would appear as follows in XML:

```
<playbackURI>rtsp://144.70.13.92:554/PSIA/Streaming/tracks/27?offset=a07724&amp;endtime=2009-05-18T10:31.25</playbackURI>
```

4 Resource Structure

本文档制定的相关接口均放在/PSIA/Custom/SelfExt/ContenMgmt分支下。



The above diagram depicts the basic REST resources supported by a spec-compliant PSIA RaCM device as outlined in this document. The colored boxes in the diagram indicate resources that are 'Services'. Services are resources that carry attributes defined by "description" and "capabilities" schemas (see below and PSIA Service Model specification). Additionally, the resource hierarchy determines the REST URI structures used to interact with each resource.

Resource Name	**Description**	**Mandatory/Optional**
Description	will respond to an HTTP GET with a <ResourceDescription> datablock	Mandatory
Capabilities	will respond to an HTTP GET with a resource-specific datablock	Generally Optional; Required for some RaCM Services
Index	will respond to an HTTP GET with a <ResourceList> datablock	Mandatory
Indexr	will respond to an HTTP GET with <ResourceList> datablock	Optional

5 General Rules, Guidelines

The following guidelines and requirements apply to those parties implementing this specification:

- All RaCM devices shall comply with the guidelines, formats, syntax, and base protocol definitions contained in the PSIA Service Model specification.
- All RaCM devices shall implement the following PSIA REST Resource hierarchies, as outlined on the PSIA IP Media Device (IPMD) Specification Version 1.0, Revision 0.7:
    - For DVRs, the "/PSIA/System/Video/…" REST resources and services, as is pertinent for the hardware capabilities for a given RaCM device.
    - For DVRs that support PTZ commands, the "/PSIA/PTZ/…" REST resources and services are to be implemented, as outlined in ***IPMD***, where the PTZ capabilities in the IPMD specification are functionally compatible for the DVR device.
    - For DVRs and NVRs, the "/PSIA/System/…" REST resource hierarchies must be implemented as defined in ***IPMD Sections***, for the configurable system based devices and I/O capabilities (network, serial, I/O, …) supported by a given RaCM device.
    - For DVRs, the "/PSIA/Streaming/Channels…" REST resource hierarchy for each hardware channel it supports. The "picture", and "requestKeyFrame" resources are not required.
    - For NVRs and hybrid DVRs, the "/PSIA/Streaming/Channels/status" REST resources in ***IPMD***.
    - NVRs and hybrid DVRs must comply with the management guidelines outlined in the following ***Section*** regarding the management of external IP media devices.
    - For DVRs and NVRs, the "/PSIA/Security…" REST resources, in IPMD Sections, as outlined by Section 13 of this document ("/PSIA/Security") are to be implemented.
    - All RaCM devices shall support the "/PSIA/System/time/…" REST resources for setting/reporting local time on their devices unless they use a network domain controller to access network time.
    - All DVRs and NVRs must provide system level status for their devices via the "/PSIA/System/status" REST resource defined in ***IPMD*** specification.
    - DVRs and NVRs that support internal video motion detection (VMD) functionality, must implement the "/PSIA/Custom/MotionDetection…" REST hierarchy described in IPMD specification where the VMD function they provide is compatible with the modes listed there.

Though the resource hierarchy described herein does cite specific resources in the IPMD and Service Model specifications, it does not preclude RaCM device implementers from incorporating other PSIA resources that are relevant. For example, in addition to the RaCM required services, a DVR/NVR may choose to implement the IPMD "/PSIA/Diagnostics" service resources. The objective of this specification is to provide the design details and intent for the base protocol implementation.

(Updated v1.1) As a rule, handling REST/management commands should not affect the streaming and recording behaviors. However, any REST/management command that modifies a streaming, codec, or network parameter, may be disruptive/destructive to recording and/or streaming. When a RaCM device performs a command that is disruptive, or destructive, to a streaming and/or recording operation it MUST return the HTTP status code "206 Reset Content" in the response message. This status code indicates a successful update, but indicates that there has been some significant affect to the streaming/recording operations.

A number of clients should be able to send API commands to a Content Management device concurrently. The maximum number of pending operations, and any associated timeouts, for each API command are defined in later in this specification.

(New v1.1) DVR manufacturers should carefully read all of this section, Section 10.2 and Section 13.3 to understand the configuration and correlation of 'channels' to each other, and to tracks, on a system that has local video codec hardware. Additionally, Section 13.3 describes how all input channels are enumerated with respect to type and status.

5.1 DVR & NVR Design Considerations

Digital Video Recorder (DVR) devices have design issues unique to their product class. Unlike NVRs, DVRs have onboard video and (in many cases) audio codec hardware as their input sources. Since this hardware is intrinsic to the device, i.e. it is not dynamically assignable to the unit like NVR input sources, DVRs must comply with the following design operational items, in addition to the general requirements listed in the above section of this document:

- All video codec hardware must be listed as pre-configured (i.e. with default settings) input 'channels' in the following PSIA resource hierarchies:
  - For DVR video input hardware that provides image setting parameters, such as brightness, contrast, sharpness, etc., these settings must be accessible in the "/PSIA/System/Video/inputs/channels…" resource hierarchy. This means that the 'VideoInputChannelList' XML schema document returned by a GET to the "/PSIA/System/Video/inputs/channels" resource must list all of the existing input channels as 'VideoInputChannel' elements. The IDs for each element must be set for each channel by the DVR and cannot be allowed to be changed by external entities .
  - DVR video codec hardware must be listed as pre-configured (i.e. using default values) input 'channels' in the "/PSIA/Streaming/channels…" resource. The returned 'StreamingChannelList" XML schema document must be pre-populated with all of the codec channels that are available for configuration and use. The channel IDs in the schema must be preset by the DVR.
- All audio codec hardware present on a DVR must be listed as existing, pre-configured (i.e. using default values) hardware input 'channels' in the "/PSIA/System/Audio/channels…" resource hierarchy. The channel IDs must be pre-populated in the 'AudioChannelList' XML schema document returned by GETs to the "/PSIA/System/Audio/channels…" resource. The channels IDs assigned by a DVR are immutable by external entities.
- It is strongly recommended that all channel ID values be unary-based ASCII/XML unsigned integer values to support consistency across vendors.
- 多个"/PSIA/Streaming/channels/<id>" 可能具有相同的 <videoInputChannelID>。多个"/PSIA/Custom/SelfExt/ContentMgmt/DynStreaming/channels/<id>" 可能具有相同的 <dynVideoInputChannelID>。考虑到stream与track的关系，强烈的建议 video input id, dynVideo input id, stream id, dynStream id采用如下编码方式：

  (1) 为了不同设备之间的兼容性，/PSIA/System/Video/inputs/channels/<ID>与/PSIA/Custom/SelfExt/ContentMgmt/DynVideo/inputs/channels/<ID>中的ID统一编号。例如：如果Video/inputs/channels/<ID>为1到9，则DynVideo/inputs/channels/<ID>从1，10开始。ID 0给DVR或混合DVR零通道保留，通过获取/PSIA/System/Video/inputs/channels通道列表中是否含有0，可知DVR或混合DVR是否支持0通道。

  (2) 流或动态流<ID>的编码采用8位十进制整形编码，高六位表示视频输入或动态视频输入通道号，低两位表示属于属于该通道的流编号，并且流通道号从1开始编。例如对于混合DVR固有通道1的第9条流，则其编号为109；对于混合DVR的动态通

道100的第99条流，则其编号为10099

- Tracks will record from their respective <SourceDescriptor>'s.
  - Thus, each active Track implicitly creates an input 'channel'.
  - The <id> for these 'channels' are determined by the RaCM Device automatically.
  - It is strongly recommended that these channel <id>'s match the <Track> ID's and are also propagated to the /PSIA/Streaming/channels/<id>'s. This would facilitate easy streaming of the "live" content that is feeding the Recorded <Track>, if exposing this Stream to an external Client is desired by the RaCM implementor. For the NVR, this "live" streaming feature is not critical, since the networked Camera's stream should also be viewable by any Client on the network without aid from the RaCM Device. For the DVR, this "live" stream serves a more critical function, since the Camera is unlikely to also be network attached to allow for independent "live" viewing.
- Redundant 'channel' handling: The actual way of handling this problem is implementation specific. When multiple tracks record from the same source (same Device/GUID and same stream URL), they essentially create identical 'channels'. After the first 'channel' is created by the RaCM Device, the subsequent identical 'channels' may simply refer to the first 'channel' or new 'channels' may be created which are identical to the first (essentially cloning the first 'channel'). However, in the second approach, the new 'channels' would have their own <id>'s, making the search interface less effective (since channel-based searches would require a list of all identical 'channel' <id>'s to truly search all the desired 'channels'). In contrast, the first approach serves the search and status interfaces better but is much more complex to implement, since the first and original 'channel' could not be removed based on any particular track being deleted since multiple tracks may be referencing it.

The above design items are described here to aid in providing commonality in the implementation of this specification, and provide clarity for those developing to the interfaces defined herein. Hybrid DVR/NVR products must still follow the above guidelines for the onboard codec hardware that is present on their respective devices. Please note that these design guidelines pertain to the definition and configuration of the video/audio input hardware. The information related to track configuration, status, etc., is addressed later in this document.

5.2 Input Source Management (Remote Camera Configuration)

NVRs, and hybrid DVRs, support external IP network devices as their input sources. These devices, usually IP cameras, may, or may not, be PSIA protocol compliant. This specification does not require the external input sources to be PSIA compliant in order to be supported by a RaCM device. However, within the industry there are different methods for managing the support for, configuration and status of, input IP media devices. PSIA RaCM devices must fall into one of two possible categories regarding the management of external IP media sources. The management categories are listed below:

Management M ode	Mode Description
Simple	This mode describes as RaCM device that manages track configuration, but does not manage the settings related to video/audio codecs and streaming at the external source camera/encoder. This means that management entities such as VMS and PSIM applications are required to manage the settings on the RaCM devices, and the external source devices, separately. I.e. the RaCM device does not change the settings on a source device when a codec or streaming related parameter is changed on a track. A Simple RaCM device will attempt to open a new session to the respective source with the new codec/streaming settings, but it will NOT modify the codec/streaming configuration settings on an external camera/encoder. In some cases, this is sufficient since codec settings on some cameras can be modified on-the-fly via the session setup parameters; in other cases the VMS/PSIM management application will have to modify the codec/streaming settings at the source prior to making track setting changes. In all cases in Simple management mode, the management application (VMS, PSIM, etc.) is responsible for the settings at all the devices and for the synchronization of those settings between the sources and consumers. The recommendation in this mode is that management applications SHOULD always change the settings at the source device (camera, encoder) and then modify the appropriate track settings on the dependent RaCM device(s). Please note that: A) this can produce race conditions between the source the and recoding device in some cases, and B) for some settings there will not be a disruption of service but the RaCM device should always attempt (and log, if possible) a reconnection of the streaming input when a track's codec setting is modified.
Proxy	RaCM devices that have the ability to remotely manage other source device's codec/streaming parameters are called 'proxy managers'. Modifications to codec/streaming settings on a proxy managing RaCM device will also be performed at the external source device by the RaCM device for those devices the RaCM unit knows how to configure. Basically, in proxy management mode the RaCM device receives the settings that affect streams and/or tracks, and performs, based on the sources behavior, all of the necessary configuration adjustments on behalf of the management application.RaCMv1.1不支持此种模式，我们通过引入动态视频输入及动态流服务来支持Proxy模式。

Since the PSIA system Security model is not completed yet within the PSIA, all version 1.1 RaCM devices that support external IP media devices as input sources must comply with the Simple management mode listed above. In order to aid interoperability, and to support the future use of Proxy management mode, a new RaCM v1.1 REST resource "/PSIA/ContentMgmt/sourceSupport" has been created. This resource identifies which mode of management and level of interoperability a RaCM device provides for each advertised camera/encode device it claims support for (see */PSIA/ContentMgmt/sourceSupport (new v1.1)*). Fundamentally, this RaCM resource provides a list of IP cameras and encoders, by manufacture and model, that are supported (i.e. compatible) along with the management mode for each (for v1.1 RaCM devices this is 'Simple' mode). This resource also addresses an area of compatibility and interoperability that the RaCM version 1.0 specification did not directly define. These areas are covered in more detail in the following sections of this document.

6 ContentMgmt Base Service

The ‘ContentMgmt’ base service is the ‘root’ for all Recording and Content Management (RaCM) device function related to the recording and management of multimedia data. This service is the base node in the REST resource hierarchy for all active functions provided by a RaCM device.

As mentioned in the previous section, the RaCM device must also comply with the PSIA Service Model specification and also implement the required IPMD Services.

6.1 /PSIA/ContentMgmt/profile

Due to the complexity of the functions entailed in a recording and content management (RaCM) device, all RaCM devices MUST provide a ‘profile’ resource (schema instance) such that entities accessing them may determine their functional level and basic attributes. Details are outlined below.

URI	/PSIA/Custom/SelfExt/ContentMgmt/profile		**Type**	Resource
Requirement Level	- All -			
Function	RaCM Mandatory REST resource/object that publishes the functional profile/level of a RaCM device and its operable service/resource structure.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET	None		<CMProfile>	
PUT	N/A	N/A	<ResponseStatus w/error code>	
POST	N/A	N/A	<ResponseStatus w/error code>	
DELETE	N/A	N/A	<ResponseStatus w/error code>	
Notes	The ‘GET’ request issued to retrieve an instance of the ‘CMCapabilities’ XML schema.			

Example(s)	<?xml version="1.0" encoding="UTF-8"?> <CMProfile version="1.0" xmlns="urn:psialliance-org"> <profileLevel>basic+</profileLevel> <supportedResourceList> <supportedResource>/PSIA/ContentMgmt/search</supportedResource> <supportedResource>/PSIA/ContentMgmt/status</supportedResource> <supportedResource>/PSIA/ContentMgmt/status/volumes</supportedResource> <supportedResource>/PSIA/ContentMgmt/status/tracks</supportedResource> <supportedResource>/PSIA/ContentMgmt/status/channels</supportedResource> <supportedResource>/PSIA/ContentMgmt/status/sources</supportedResource> <supportedResource>/PSIA/ContentMgmt/record</supportedResource> <supportedResource>/PSIA/ContentMgmt/record/profile</supportedResource> <supportedResource>/PSIA/ContentMgmt/record/tracks</supportedResource> <supportedResource>/PSIA/ContentMgmt/record/control</supportedResource> <supportedResource>/PSIA/ContentMgmt/schedules</supportedResource> </supportedResourceList> <mfgrInformation>WarpedAndJaded Inc., RecorderWare Version 2.01.4, Model 1600</mfgrInformation> <CMProfile> The above example represents the Content Management profile XML instance for a 'Basic' profile/level device that supports some optional features (hence the "basic+" tag). The advertised resource list outlines all the supported, non-overhead REST resources (such as 'index' and 'description'). Since this is a basic device only the minimum resources are listed.

6.1.1 /PSIA/ContentMgmt/profile Schema Definition

The profile schema for the Content Management base service describes 3 major areas:

- The functional profile/level of the RaCM device, which is either 'Basic' or 'Full'; devices that support optional functions must add a plus sign ('+') suffix to the profile level tag (e.g. "full+").
- A list of all the supported resources. This enables service users to determine what functions (i.e. REST resources) are active on a particular device instance.
- Optional, but recommended, Manufacturer information relevant to the make/model, type and version of the device.

XSD: To get the latest version of all RaCM XSDs, including this one (cmProfile.xsd) please download from the PSIA documents website: http://www.psialliance.org/documents_download.html

6.2 /PSIA/ContentMgmt/sourceSupport

For version 1.1-compliant RaCM devices that support external IP cameras and encoders as input devices, the 'sourceSupport' REST resource MUST be supported in order to advertise the types and models of devices that a unit is compatible with. ***Section Input Source Management*** of this specification describes the management modes a RaCM device may support for managing external IP input sources. In version 1.1, RaCM devices are only required to support Simple management mode.如果希望支持proxy模式，可以使用海康扩展接口来实现。 GETs from the "/PSIA/ContentMgmt/sourceSupport" resource return a schema that describes the manufacturers, makes and models of IP media devices that a RaCM device is compatible with. The following table provides the base details for the 'sessionSupport' resource. Please note that for version 1.1. RaCM devices this is a ***read-only*** resource.

URI	/PSIA/ContentMgmt/sourceSupport		**Type** Resource
Requirement Level	Basic (v1.1)		
Function	Description of the IP media devices, in mfgr, make and model, that a RaCM devices supports as input sources		
Methods	**Query String(s)**	**Inbound Data**	**Return Result**
GET	Optional: "send=head" "send=top", "send=middle" "send=bottom" OR… "mfgr=<mfgrName>"	None	<CMSourceSupport>
PUT	None	None	<ResponseStatus w/error code>
POST	None	None	<ResponseStatus w/error code>
DELETE	None	None	<ResponseStatus w/error code>
Notes	The schema and element definitions for this resource follow.		

Example XML

```
<?xml version="1.0" encoding="UTF-8"?>
<CMSourceSupport version="1.0" xmlns="urn:psialliance-org">
   < mediaDeviceListSize>17</ mediaDeviceListSize>
   < supportedMediaDeviceSourceList>
      <MediaDeviceSource>
         <MediaDeviceMfgr>GoodVision</MediaDeviceMfgr>
         <MediaDeviceMgmtMode>Simple</MediaDeviceMgmtMode>
         <MediaDeviceModel>GVX-1000</MediaDeviceModel>
         <MediaDeviceModel>GVX-1000PTZ</MediaDeviceModel>
         <MediaDeviceModel>GVX-1070D</MediaDeviceModel>
         <MediaDeviceModel>GVX-1150PTZ</MediaDeviceModel>
         <MediaDeviceModel>GVX-1500D</MediaDeviceModel>
         <MediaDeviceModel>GVX-2000D</MediaDeviceModel>
      </MediaDeviceSource>
      <MediaDeviceSource>
         <MediaDeviceMfgr>ClearView</MediaDeviceMfgr>
         <MediaDeviceMake>MaxView Series</MediaDeviceMake>
         <MediaDeviceMgmtMode>Simple</MediaDeviceMgmtMode>
         <MediaDeviceModel>MCV-100F</MediaDeviceModel>
         <MediaDeviceModel>MCV-250PTZ</MediaDeviceModel>
         <MediaDeviceModel>MCV-500</MediaDeviceModel>
         <MediaDeviceModel>MCV-650PTZ</MediaDeviceModel>
      </MediaDeviceSource>
      <MediaDeviceSource>
         <MediaDeviceMfgr>Gotcha</MediaDeviceMfgr>
         <MediaDeviceMake>Professional Series</MediaDeviceMake>
         <MediaDeviceMgmtMode>Simple</MediaDeviceMgmtMode>
         <MediaDeviceModel>GP2100F</MediaDeviceModel>
         <MediaDeviceModel>GP2500D</MediaDeviceModel>
         <MediaDeviceModel>GP3000PTZ</MediaDeviceModel>
         <MediaDeviceModel>GP3200FA</MediaDeviceModel>
         <MediaDeviceModel>GP3500PTZA</MediaDeviceModel>
      </MediaDeviceSource>
      <MediaDeviceSource>
         <MediaDeviceMfgr>PSIA</MediaDeviceMfgr>
         <MediaDeviceMake>Any PSIA compliant IP Media device</MediaDeviceMake>
         <MediaDeviceMgmtMode>Standard</MediaDeviceMgmtMode>
         <MediaDeviceMgmtMode>PTZ</MediaDeviceMgmtMode>
      </MediaDeviceSource>
   </ supportedMediaDeviceSourceList>
</CMSourceSupport>
```

The above example represents a RaCM device that supports 17 different camera models from 4 different manufacturers. Please note that the 'make' element is not used on every manufacturer. Also, the 'PSIA' IP Media Device support is listed as a generic manufacturer with Standard (i.e. non-PTZ) and a PTZ generic model support. This example is only referential.

Accesses to the 'sessionSupport' resource returns a 'CMSourceSupport' schema document instance. This schema document is primarily a list of the types and models of IP media devices that a RaCM device supports as compatible input sources. Each element in a list element is comprised of the following parameters:

Element name	Requirement	Description
"MediaDeviceMfgr"	Mandatory	XML string field that lists the manufacturer of the supported IP Media Device. This field is treated as ***case insensitive*** to aid in searching and matching (e.g. Luminati = luminati, effectively).
"MediaDeviceMake"	Optional	Optional XML string field that lists the make of an IP Media device. Some manufacturers come out with 'lines' (i.e. product lines) of cameras that share protocol attributes. This field is to aid in identifying products that belong to a certain product line (where/when that information is pertinent and relevant).
"MediaDeviceModel"	Mandatory	XML string that identifies the model of a certain IP Media device.
"MediaDeviceMgmtMode"	Mandatory	XML type restricted to "Simple" or "Proxy". "Simple" is the only valid value for version 1.1 RaCM devices.

Please note that all of the values contained in the above fields are treated in a case insensitive manner to aid in better interoperability between management applications, clients and the RaCM devices. The purpose of the "CMSourceSupport" schema is to provide consumers with a well ordered list of the media devices that it can record and stream. It is also acceptable for 'generic' media device support to be listed where the RaCM device has an industry (or mfgr) standards-based driver that supports compliant cameras generically (i.e. the "MediaDeviceMfgr" does not have to contain a literal manufacturer; it could contain a value like "PSIA" or "PSIA PTZ", etc.). The full schema definition follows.

6.2.1 Source Support XML Schema Definition

XSD: To get the latest version of all RaCM XSDs, including this one (cmSourceSupport.xsd) please download from the PSIA documents website: http://www.psialliance.org/documents_download.html

This schema definition is basically a list of elements that define the types of IP media devices that are supported, or are compatible with, a RaCM device. The first element is a count of the number of IP media device ***models*** (i.e. "MediaDeviceModel" entries) that are listed in a "CMSourceSupport" document instance. This is important since the size of the schema can be huge, in some cases, and due to the fact that consumers can ask for only the 'count' of the supported media devices (see following section). Please note that each list element, which has one manufacturer value, can support multiple model numbers/strings per that manufacturer value. The manufacturer name can also identify standards organizations, not specific manufacturers.

6.2.2 Access and Operation of Source Support

The REST URI structure of the "/PSIA/ContentMgmt/sourceSupport" resource allows consumers to

ask for portions, or all, of the schema document information (see table in above ***Section 8.4***). A consumer accessing source support by a simple GET to the "/PSIA/ContentMgmt/sourceSupport" resource will get the entire data list of all supported IP media devices in the "CMSourceSupport" schema instance. However, due to the fact that the size of this XML list could be prohibitively large, RaCM devices MUST support the following options for getting portions of the source support information without requiring the entire data set. There are two query parameters that can be added to the REST URI for getting portions of the overall information set. They are:

- "**send=**…". The send designator identifies that the consumer only wants a portion of the overall source support information. A value is supplied with the "send=" string as an NVP that designates the portion of the information to be sent. This is discussed in more detail below.
- "**mfgr=**…". The data designator specifies a 'filter' for the information to be returned. Basically, a consumer specifies what type of information it is looking for. This is described in detail below.

Please note that the 'send' and 'mfgr' query parameters are mutually exclusive; they cannot be used in the same HTTP/REST GET message since they each counter-actively affect what, and how much, of the source support information is returned by a RaCM device. Consumers of source information are to use these query string parameters to govern the amount of source information that is transferred since this information base can be very large. Each of the query string parameters is described below, in detail

6.2.2.1"Send=…" Query String Parameter

The "send=…" query string parameter (QSP) enables consumers to specify, or control, (in a coarse manner) the amount of source support information that they desire to receive. If the "send=…" QSP is not present when a GET to the "/PSIA/ContentMgmt/sourceSupport" resource is issued, ALL the source support information will be returned in the CMSourceSupport schema instance. In order to prevent data overrun, the "send=…" QSP is provided such that the consumer can ask for the source support list information in 'chunks'. The amount, and type, of information is specified by the value tag supplied with the "send=…" QSP. The value tags that can be supplied are listed below with descriptions.

Tag Value	URI Example	Description
"head"	GET /PSIA/ContentMgmt/sourceSupport?send=head	This tag value indicates that the consumer only wants the returned schema instance to contain the count of the number of source support *models* in its list; No element data should be returned. This allows consumers to gauge the approximate size of the source support information base, in total.
"top"	GET /PSIA/ContentMgmt/sourceSupport?send=top	This tag value indicates that the consumer wants the top 1/3rd (roughly) of the source support list. RaCM devices receiving this QSP should provide, approximately, the first one-third of the "CMSourceSupport" list.

"middle"	GET /PSIA/ContentMgmt/sourceSupport?send=middle	This tag value indicates that the consumer wants the middle 1/3rd (roughly) of the source support list. RaCM devices receiving this QSP should provide, approximately, the middle one-third of the "CMSourceSupport" list data.
"bottom"	GET /PSIA/ContentMgmt/sourceSupport?send=middle	Similar to the two above tags, this tag value indicates that the consumer wants the last 1/3rd (roughly) of the source support list. RaCM devices receiving this QSP should provide, approximately, the final one-third of the "CMSourceSupport" list data. Please note that it is up to the RaCM device on how to apportion the source support info into 'chunks'.

When a RaCM device receives a "send=head" request, the returned schema should only contain the "MediaDeviceModelCount" element and its value. For all of the other QSP strings, the RaCM device is to provide portions of the schema document that align on list element boundaries (i.e. manufacturer/make boundaries). This apportioning scheme prohibits the use of the "mfgr=…" QSP, which is described in the following section. The following examples are provided as additional descriptive information. These examples are based on the example XML schema instance listed in the table at the top of ***Section /PSIA/ContentMgmt/sourceSupport***.

Send 'head' Example:

```
(request)
GET /PSIA/ContentMgmt/sourceSupport?send=head HTTP/1.1
… (response)
HTTP/1.1 200 OK
Content-type:application/xml
Content-length :165
<?xml version="1.0" encoding="UTF-8"?>
<CMSourceSupport version="1.0" xmlns="urn:psialliance-org">
      <MediaDeviceModelCount>17</MediaDeviceModelCount>
</CMSourceSupport>
```

Send 'top' Example :

```
(request)
GET /PSIA/ContentMgmt/sourceSupport?send=top HTTP/1.1
… (response)
HTTP/1.1 200 OK
```

```
Content-type:application/xml
Content-length :<nnn>
<?xml version="1.0" encoding="UTF-8"?>
<CMSourceSupport version=”1.0” xmlns=”urn:psialliance-org”>
      <MediaDeviceModelCount>6</MediaDeviceModelCount>
      <SupportedMediaDeviceSourceList>
            <MediaDeviceSource>
                  <MediaDeviceMfgr>GoodVision</MediaDeviceMfgr>
                  <MediaDeviceMgmtMode>Simple</MediaDeviceMgmtMode>
                  <MediaDeviceModel>GVX-1000</MediaDeviceModel>
                  <MediaDeviceModel>GVX-1000PTZ</MediaDeviceModel>
                  <MediaDeviceModel>GVX-1070D</MediaDeviceModel>
                  <MediaDeviceModel>GVX-1150PTZ</MediaDeviceModel>
                  <MediaDeviceModel>GVX-1500D</MediaDeviceModel>
                  <MediaDeviceModel>GVX-2000D</MediaDeviceModel>
            </MediaDeviceSource>
      </SupportedMediaDeviceList>
</CMSourceSupport>
```

Send ‘middle’ Example :

```
(request)
GET /PSIA/ContentMgmt/sourceSupport?send=middle HTTP/1.1
… (response)
HTTP/1.1 200 OK
Content-type:application/xml
Content-length :<nnn>
<?xml version="1.0" encoding="UTF-8"?>
<CMSourceSupport version=”1.0” xmlns=”urn:psialliance-org”>
      <MediaDeviceModelCount>4</MediaDeviceModelCount>
      <SupportedMediaDeviceSourceList>
            <MediaDeviceSource>
                  <MediaDeviceMfgr>ClearView</MediaDeviceMfgr>
                  <MediaDeviceMake>MaxView Series</MediaDeviceMake>
                  <MediaDeviceMgmtMode>Simple</MediaDeviceMgmtMode>
                  <MediaDeviceModel>MCV-100F</MediaDeviceModel>
                  <MediaDeviceModel>MCV-250PTZ</MediaDeviceModel>
                  <MediaDeviceModel>MCV-500</MediaDeviceModel>
                  <MediaDeviceModel>MCV-650PTZ</MediaDeviceModel>
            </MediaDeviceSource>
      </SupportedMediaDeviceList>
</CMSourceSupport>
```

Send ‘bottom’ Example :

```
(request)
GET /PSIA/ContentMgmt/sourceSupport?send=bottom HTTP/1.1
… (response)
HTTP/1.1 200 OK
Content-type:application/xml
Content-length :<nnn>
<?xml version="1.0" encoding="UTF-8"?>
<CMSourceSupport version=”1.0” xmlns=”urn:psialliance-org”>
```

```
      <MediaDeviceModelCount>7</MediaDeviceModelCount>
      <SupportedMediaDeviceSourceList>
            <MediaDeviceSource>
                  <MediaDeviceMfgr>Gotcha</MediaDeviceMfgr>
                  <MediaDeviceMake>Professional Series</MediaDeviceMake>
                  <MediaDeviceMgmtMode>Simple</MediaDeviceMgmtMode>
                  <MediaDeviceModel>GP2100F</MediaDeviceModel>
                  <MediaDeviceModel>GP2500D</MediaDeviceModel>
                  <MediaDeviceModel>GP3000PTZ</MediaDeviceModel>
                  <MediaDeviceModel>GP3200FA</MediaDeviceModel>
                  <MediaDeviceModel>GP3500PTZA</MediaDeviceModel>
            </MediaDeviceSource>
            <MediaDeviceSource>
                  <MediaDeviceMfgr>PSIA</MediaDeviceMfgr>
                  <MediaDeviceMake>Any PSIA compliant IP Media device
                        </MediaDeviceMake>
                  <MediaDeviceMgmtMode>Standard</MediaDeviceMgmtMode>
                  <MediaDeviceMgmtMode>PTZ</MediaDeviceMgmtMode>
            </MediaDeviceSource>
         </SupportedMediaDeviceList>
</CMSourceSupport>
```

The above examples are for reference. The data set they contain is hypothetical and relatively small for the sake of simplicity. Please note that in these examples the RaCM device made its own arbitrary decision as to where to subdivide the CMSourceSupport information of 'top' versus 'middle' versus 'bottom'. Also note that the "MediaDeviceModelCount" size varies since the number of IP media device models varied per response. The next section describes the ability for consumers to ask for specific forms of information.

6.2.2.2 "mfgr=" Query String Parameter

The "mfgr=…" QSP enables consumers to ask for supported IP media device information related to a specific manufacturer or standards organization. Consumers using this capability can ask for supported model information related to a specific manufacturer or standard. An example request would look like:

```
GET /PSIA/ContentMgmt/souceSupport?mfgr=GoodVision
```

This type of request instructs the RaCM to only return a "CMSourceSupport" schema instance that lists the supported models for the manufacturer "GoodVision". The following is an example of what is expected to be returned:

```
…
 (request)
GET /PSIA/ContentMgmt/sourceSupport?mfgr=GoodVision HTTP/1.1
… (response)
HTTP/1.1 200 OK
Content-type:application/xml
Content-length :<nnn>
<?xml version="1.0" encoding="UTF-8"?>
<CMSourceSupport version="1.0" xmlns="urn:psialliance-org">
      <MediaDeviceListSize>6</MediaDeviceListSize>
```

```
      <SupportedMediaDeviceSourceList>
            <MediaDeviceSource>
                  <MediaDeviceMfgr>GoodVision</MediaDeviceMfgr>
                  <MediaDeviceMgmtMode>Simple</MediaDeviceMgmtMode>
                  <MediaDeviceModel>GVX-1000</MediaDeviceModel>
                  <MediaDeviceModel>GVX-1000PTZ</MediaDeviceModel>
                  <MediaDeviceModel>GVX-1070D</MediaDeviceModel>
                  <MediaDeviceModel>GVX-1150PTZ</MediaDeviceModel>
                  <MediaDeviceModel>GVX-1500D</MediaDeviceModel>
                  <MediaDeviceModel>GVX-2000D</MediaDeviceModel>
            </MediaDeviceSource>
        </SupportedMediaDeviceList>
</CMSourceSupport>
```

Please note that the manufacturer name string is treated in a case insensitive manner. Also, manufacturer strings that contain spaces must use the W3C special character replacement scheme. For example, a fictitious vendor named "Volcano Vision" would have a QSP that looks like "...mfgr=Volcano%20Vision…". RaCM devices must parse out, and convert, the special character symbols just like the processing associated with a standard URI. Additionally, a consumer may place multiple "mfgr=…" QSPs in a request. For example:

`GET /PSIA/ContentMgmt/souceSupport?mfgr=GoodVision&mfgr=Gotcha&mfgr=AccuMax` is a valid REST URI for the sourceSupport resource. The recommendation is that a consumer should ***not*** send more than 3 manufacturer IDs per request. If a RaCM device cannot process the number of 'mfgr' QSPs sent in a single request, it should return an HTTP status code of '503 Bad Request'.

7 /PSIA/ContentMgmt/record

This section describes the REST Interfaces to configure the logical storage used for media archival, along with the actual recording session configurations. A recording session is also known as a "track" in the RaCM Device. A track archives media (Audio, Video, Metadata) from a Source. The Source can be local (e.g. DVR recording from a local analog port) or remote (e.g. IP Network Media Device).

7.1 /PSIA/ContentMgmt/record/storageMounts

This REST Interface is used to configure the total storage available to the RaCM Device to be used to archive media. This storage is described as a List of Mount points, along with root directories and sizes. There is no affinity specified, at this time, with regards to assigning tracks or range of tracks to a particular Disk or Mount.

URI	/PSIA/ContentMgmt/record/storageMounts		**Type**	Resource
Requirement Level	Basic			
Function	Description of the REST method parameters and formats available to functionally manipulate the 'record/storage' resource.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET	None	None	<MountList>	
PUT	None	<MountList>	<ResponseStatus>	
POST	None	<Mount>	<ResponseStatus>	
DELETE	None	None	<ResponseStatus>	
Notes	This resource is used to manage the total storage allocation and logical mounts of the Recorder. It is allowable to DELETE the entire list, though any implementation is free to return an error for that operation if that capability is undesirable. Low level drive configuration should be done through "/PSIA/System/storage". Low level diagnostics, such as drive temperatures, should be available in future (TBD) extensions to IPMD, under "/PSIA/System…" or "/PSIA/Diagnostics…"			

This resource manages the <MountList> XML object and follows the same scheme used for <TrackList> manipulation and other similar examples from IPMD that manage a list-based XML resource (see "PSIA-REST List-Entry <id> Creation method", below.). As such, GET and PUT methods are used to access the entire <MountList>. POST is used (with a "dummy" <id> of 0) to create an individual entry within the list; where the newly created <id> is returned in the <ResponseStatus> given for the POST request.

Once an <Mount> entry is created, it can be accessed by its <id>, using this resource:

URI	/PSIA/ContentMgmt/record/storageMounts/<id>		Type	Resource
Requirement Level	Basic			
Function	Description of the REST method parameters and formats available to access a single <Mount> entry.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET	None	None	<Mount>	
PUT	None	<Mount>	<ResponseStatus>	
POST	N/A	N/A	<ResponseStatus w/ error code>	
DELETE	None	None	<ResponseStatus>	
Notes	POST (i.e. Create) is not allowed for individual <Mount> entry, with given explicit <id>.			

Example XML <MountList>:

```
<?xml version="1.0" encoding="UTF-8"?>
<MountList version="1.0" xmlns="urn:psialliance-org">
  <Mount>
    <id>1</id>
    <path>/dev/hda</path>
    <dir>/racm1/record tracks</dir>
    <size>200000000000</size>
    <descr>master ide</descr>
  </Mount>
  <Mount>
    <id>2</id>
    <path>/dev/sda</path>
    <dir>/racm1/record_tracks</dir>
    <size>500000000000</size>
    <descr>first scsi</descr>
  </Mount>
  <Mount>
    <id>3</id>
    <path>d:</path>
    <dir>/racm1/record_tracks</dir>
    <size>100000000000</size>
    <descr>win-dos drive</descr>
  </Mount>
</MountList>
```

XSD: To get the latest version of all RaCM XSDs, including this one (cmRacmMount.xsd) please download from the PSIA documents website: http://www.psialliance.org/documents_download.html

7.2 /PSIA/ContentMgmt/record/profile

Version 1.1 compatible RaCM devices have the ability to support more than one type of 'track'. Since a track is nothing more than a named handle to a virtual content container, PSIA does not specify how content is actually recorded, indexed, etc. However, the attributes and properties that define how a track is configured, how a track's content is understood, and how its contents may be played, are critical to interoperability. The definitions and descriptions of track attributes and track types are defined in following ***Sections***. The REST resource that advertises these recording attributes (i.e. track types) is defined here.

URI	/PSIA/ContentMgmt/record/profile		**Type**	Resource
Requirement Level	-All-			
Function	Description of the REST resource that advertises the track types supported by a RaCM device.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET	None	None	<CMRecordProfile>	
PUT	None	None	<ResponseStatus w/Error Code>	
POST	None	None	<ResponseStatus w/Error Code>	
DELETE	None	None	<ResponseStatus w/Error Code>	
Notes	This resource is ***read-only***.			
Example	<?xml version="1.0" encoding="UTF-8"?> <CMRecordProfile version="1.0" xmlns="urn:psialliance-org"> <trackType>standard</trackType> <trackType>polymorphic</trackType> <trackType>polytemporal</trackType> </CMRecordProfile>			
	The above example XML response instance indicates that the RaCM device supports all of track types defined by the version 1.1 RaCM specification. See "/PSIA/ContentMgmt/record (Recorder Configuration & Control)" for details about the attributes and behaviors of track types.			

The "/PSIA/ContentMgmt/record/profile" REST resource is a read-only advertisement to management applications (VMSs, PSIMs, etc.) of the supported track types which, in turn, governs their configuration options. All RaCM devices are required to support at least one of the sanctioned track types described in this specification. The "CMRecordProfile" schema definition also enables vendors to provide their own extensions to this information using a structured extension mechanism. The schema is described below.

7.2.1 /PSIA/ContentMgmt/record/profile Schema Definition

XSD: To get the latest version of all RaCM XSDs, including this one (cmRecordProfile.xsd) please download from the PSIA documents website:
http://www.psialliance.org/documents_download.html

RaCM devices list the track types they support which affects the configuration characteristics of the tracks (i.e. polymorphic tracks can have video and audio sources mixed (for example)

whereas standard tracks contain only one form of media data). For vendors that desire to add their own track type extensions, this is provided for in the schema with the following conditions:

- All RaCM devices MUST support at least one of the sanctioned version 1.1 track types (standard, polymorphic, polytemporal). Any custom track type extensions must be in addition to this requirement. This is to ensure that all RaCM have a base level of interoperability.
- All custom extensions listed in a "CMRecordProfile" MUST have the following contents:
    - The listed track type of "other" for the custom/extended track type.
    - In the "profileExtension" element:
        - A URI link to where the formal definition of the extension information can be accessed.
        - A brief description, embedded in the document instance, of what the extension information means.
        - The extension information itself.

The above requirements are levied to ensure that any vendor, or organization, specific extensions are only embedded in the PSIA RaCM framework if there is proper definition for interoperability amongst multiple parties.

Finally, the track type "group" is listed in the schema definition, but is not currently defined for RaCM version 1.1. This track type is reserved for future use.

7.3 /PSIA/ContentMgmt/record/tracks

This REST interface is used to configure a recording session or "track". A track is treated virtually with regards to how the implementation may actually store the archived media on disk.

The track is generally accessed or referred to by its <id> or <TrackGUID>. The <Stream> number is a logical (outbound streaming channel) number for the track and is not tightly coupled to the <id>, which is treated as an index into the <TrackList>.

Note on GUIDs

All UUIDs/GUIDs MUST be universally unique. They can be assigned by a central VMS Server or GUID broker or auto-generated locally. However, all UUIDs/GUIDs MUST be compliant with the ISO/IEC 9834-8 / ITU X.667 formats and definitions.

Note on Recording Modes

Currently, only 2 recording modes are defined:

- CMR – "Continuous Mode Recording" which implies recording media as it is available.
- EDR – "Event-Driven Recording" which implies recording media when events/alarms are detected.
- ALARM–报警录像，该模式记录报警录像.
- MOTION–移动侦测录像，该模式只记录移动侦测录像
- ALARMANDMOTION–报警与移动侦测录像，只有报警和移动侦测同时发生时才会录像.
- COMMAND–报警与移动侦测录像，只有报警和移动侦测同时发生时才会录像.

In either mode, the <LoopEnable> Boolean MUST be respected.

Track Size

The Track's size is determined by the <Size> value. However, optionally, some flexibility is allowed for best-effort by the implementation to honor the <Duration> value.

7.3.1 Custom Configuration Data (Extensions)

Custom configuration information can be added using <CustomExtensionList>, but for interoperability, the name is assigned to each extension object which MUST be registered with PSIA, and a schema (XSD) MUST be provided for the "xs:any" object(s).

7.3.2 PSIA-REST List-Entry <id> Creation method

Track <id>'s, along with most other list-based <id>'s, are managed (SET) by the target RaCM Device. The method for track creation (and return of <id> value is in accordance with other list- based XML object examples in IPMD). See the "/PSIA/ContentMgmt/record/tracks" Resource Description, **below.**

7.3.3 Streaming URL implied in <Track> configuration

There is an implied relationship between the REST URL's and RTSP URL's.

Within the <Track> configuration, the <Channel> value is used for "live" viewing of the media stream being recorded to that track, using the URL described in section Live Streams. The Track's REST <id> value is used for recorded (i.e. "archive") media streaming (see below).

7.3.4 Recording Source Description

The source for a recording track is logically described by the <SourceDescriptor>, which is part of the track configuration. The <SourceDescriptor> contains two important tags which help uniquely identify the media source: <SrcGUID> and <SrcUrl>. The <SrcGUID> is a GUID/UUID for a stream source, which may also provide multiple channels of output. The <SrcChannel> value allows for more specific description of the input media-stream at a logical level.

For a local DVR analog port, the <SrcUrl> should contain a symbolic reference to a local stream (encoded from a local /System/Video/inputs/channels) as follows:

rtsp://localhost/PSIA/Streaming/channels/<id>

Also, for both local and remote sources, the <SrcDriver> provides an optional, vendor specific, hint with regards to the name of an executable/driver to use for stream acquisition.

7.3.4.1 Source Description Correlation For Status Queries

The status of Tracks, Channels, and Sources can be queried from the Status resource under the ContentMgmt service.

The status of a <SrcChannel> can be queried via:
GET /PSIA/ContentMgmt/status/channels/<SrcChannel>

The status of a <Track> can be queried via:

GET /PSIA/ContentMgmt/status/tracks/<id>

The status of the <SrcGUID> can be queried via:
GET /PSIA/ContentMgmt/status/sources/<SrcGUID>

7.3.5 Recording Schedule overview

The <TrackSchedule> defines the recording schedule for the Track. It generally contains either external-references to schedules in the schedule database (/PSIA/ContentMgmt/schedules) or an embedded sequence of <ScheduleBlock>'s (typically just one). A <ScheduleBlock> is a single logical schedule, identified by GUID. The default (embedded) <ScheduleBlock>, in the example, contains the required identifiers <ScheduleBlockGUID> and <ScheduleBlockType>, along with a sequence of <ScheduleAction>'s, which are used to build a day-of-week schedule.

This default, day-of-week schedule contains <Actions> to perform during the defined period. A period is defined by a start-time and end-time, with each time expressed as Day-of-Week and Time-of-Day. The Day-of-Week value is identified by a restricted name string. The Time-of-Day is expressed in local time, in order to allow for a more human-intuitive definition of time from the local administrators perspective and also account for Day-Light-Savings Time. Thus, for a time period expressed as "midnight to 8am" local time, the intended elapsed time is 8 hours **normally**. However, if Day-Light-Savings Time is enabled, on the morning of transition (in the Spring and assuming the switch occurs at 2:00 am), the actual elapsed time would represent just 7 hours of time, during this morning of transition. In the Fall, the reverse would be true, and the actual elapsed time, for just that morning, would be 9 hours.

Example XML fragment ("lunch-time" period):

```
<ScheduleActionStartTime>
  <DayOfWeek>Monday</DayOfWeek>
  <!-- inclusive -->
  <TimeOfDay>12:00:00</TimeOfDay>
</ScheduleActionStartTime>
<ScheduleActionEndTime>
  <DayOfWeek>Monday</DayOfWeek>
  <!-- exclusive -->
  <TimeOfDay>1:00:00</TimeOfDay>
</ScheduleActionEndTime>
<ScheduleDSTEnable>true</ScheduleDSTEnable>
```

The start-time is inclusive of the time specified. The end-time is exclusive to allow aggregation of time period definitions to create a continuum without overlap/conflict.

<ExternalScheduleBlockReferences> allow for reduction in size and simplification of the <Track> configuration object, by enabling references to shared schedules in a database, as opposed to embedding them within each <Track> configuration.

7.3.6 Track Description NVP

One of the key elements in each track's parameter base is the "<Description>" element. This field is a Comma Separated Variable (CSV) string which contains a list of Name-Value-Pairs (NVP of form "Name=Value". With this scheme, the comma (',') and equal ('=') are treated as reserved characters; however, if a Name or Value string must contain and these characters, the XML encoding standard can be used to embed them if necessary (i.e. replace the '=' with &61, and replace

the ‘,’ with &44).

Parameter Name	Parameter Values	Comments
trackType	¾ “standard” = normal, single content base track ¾ “polymorphic” = multi-content type track (v1.1) ¾ “polytemporal” = multi-time segmented track (v1.1) ¾ “group”= reference to a group of tracks (v1.1)	Required Field
sourceTag	Manufacturer specific device-type string (e.g. make/model)	Optional
contentType	¾ “video” ¾ “audio” ¾ “metadata” ¾ “text NOTE: For polymorphic tracks this indicates the primary, or predominant, content type.	Required Field
codecType	See Appendix A.	Required Field
resolution	Required for audio/video. Field indicating resolution of the data elements in a datastream. For video, this is the horizontal by vertical resolution in a ‘Horizontal x Vertical’ format where ASCII ‘x’ separates the horizontal and vertical integer numbers. The assumed video format is progressive (i.e. frame based). For video streams that are interlaced (i.e. field based) and ASCII lower-case ‘i’ needs to be For audio, it is the bit-width of the samples. If a text protocol is enabled for double-byte characters, this field should be used to indicate “2B” character sets.	Required for Audio/Video
framerate	Frame rate of encoder output as a floating point number.	Required for Video
bitrate	Bit rate of datastream (bps or kbps integer).	Optional

Examples

```
<Description>trackType=standard,sourceTag=AXIS210a,contentType=video,codecType=MPEG4-
SP,resolution=640x480,framerate=25.0,bitrate=3200 kbps</Description>
```

Polymorphic and poly-temporal tracks are not defined in Version 1.0 of the RaCM specification. Additional definitions are committed for Version 1.1.

7.3.7 <MetadataEvtCfgList> (used in EDR Mode)

The Metadata Configuration is required to implement Event-Driven Recording. The <MetadataEvtCfgList> contains a list of <MetadataEvtCfg> items which describe the Domain/Class/Type of events/alarms that drive the Recording track. The fields here should comply with the PSIA Common Event Model Specification. The <evSrcGUID> and <evSrcUrl> identify the source entity from which the Metadata events/alarms are subscribed. The method of subscription is TBD, but should comply with PSIA Common Event Model. For legacy ip-cameras the <evSrcUrl> may specify a "hanging GET" url to retrieve a vendor-specific, motion detection metadata stream.

7.3.8 /PSIA/ContentMgmt/record/tracks

URI	/PSIA/ContentMgmt/record/tracks		**Type**	Resource
Requirement Level	Basic			
Function	Description of the REST method parameters and formats available to functionally manipulate the 'record/storage/tracks' resource.			
Methods	**Query Strin (s)**	**Inbound Data**		**Return Result**
GET	None	None		<TrackList>
PUT	N/A	<TrackList>		<ResponseStatus>
POST	N/A	<Track>		<ResponseStatus>
DELETE	N/A	N/A		<ResponseStatus w/error code>
Notes	**Track Creation**: POST (Create) will expect, as HTTP Payload, an individual <Track> object instead of the <TrackList>. For the Create operation, the <id> tag, within the <Track> XML, must contain a "dummy" value of 0 (zero). To lessen possibility for ambiguity, it is not permissible for the Client (xMS) to set the <id> during track creation, though it is possible for the Client to update (PUT) the entire <TrackList> (**with each entry containing valid <id>'s already set by the target**). This is in alignment with other such list-based REST Resources in IPMD.			

Reference Example <ResponseStatus> from Service Model Specification:

```
<?xml version="1.0" encoding="UTF-8"?>
<ResponseStatus version="1.0" xmlns="urn:psialliance-org">
  <requestURL>/Streaming/Channels</requestURL>
  <statusCode>1</statusCode>
  <!-- 0=1-OK, 2-Device Busy, 3-Device Error, 4-Invalid Operation, 5-Invalid XML Format, 6-Invalid XML
  Content; 7-Reboot Required -->
  <statusString>OK</statusString>
  <ID>1</ID>
</ResponseStatus>
```

7.3.9 /PSIA/ContentMgmt/record/tracks/<id>

Once created, individual Tracks are managed via:

URI	/PSIA/ContentMgmt/record/tracks/<id>		Type	Resource
Requirement Level	Basic			
Function	Resource to address (Read/Update) single <Track> by <id>.			
Methods	**Query Strin (s)**	**Inbound Data**		**Return Result**
GET	None	None		<Track>
PUT	None	<Track>		<ResponseStatus>
POST	N/A	N/A		<ResponseStatus w/error code>
DELETE	None	None		<ResponseStatus>

7.3.10 Example Track Creation Message Exchange

A. Client attempts track creation with "POST /PSIA/ContentMgmt/record/tracks" containing:

```
<?xml version="1.0" encoding="UTF-8"?>
<Track version="1.0" xmlns="urn:psialliance-org">
    <!-- new dummy value: -->
    <id>0</id>
    <Channel>12345</Channel>
    <Enable>true</Enable>
    <Description>trackType=standard,sourceTag=AXIS210a,contentType=video,codecType=MPEG4-
SP,resolution=640x480,framerate=20.0,bitrate=6000 kbps</Description>
    <TrackGUID>{A01AAAAA-BBBB-CCCC-DDDD-033595353625}</TrackGUID>
    <Size>4000000000</Size>
    <Duration>P10DT15H</Duration>
    <DefaultRecordingMode>CMR</DefaultRecordingMode>
    <LoopEnable>true</LoopEnable>

    <!-- … REST OF OBJ NOT INCLUDED… -->

  </Track>
```

B. If creation is successful, RaCM Device responds with:

```
<?xml version="1.0" encoding="UTF-8"?>
<ResponseStatus version="1.0" xmlns="urn:psialliance-org">
  <requestURL>/PSIA/ContentMgmt/record/tracks</requestURL>
  <statusCode>1</statusCode>
  <statusString>OK</statusString>
  <ID>777</ID>
</ResponseSta
tus>
```

Note that the <ID> tag is uppercase. This is to match the example in Service Model Specification (Section 10.1.4). The returned Track <id> value is 777, which will be used in the following Track Deletion example.

Example Track Deletion Message Exchange

A. Client attempts track deletion with "DELETE /PSIA/ContentMgmt/record/tracks/777" (no payload), using <id> previously given by the creation example response, above.

B. If deletion is successful, RaCM Device responds with:

```
<?xml version="1.0" encoding="UTF-8"?>
<ResponseStatus version="1.0" xmlns="urn:psialliance-org">
  <requestURL>/PSIA/ContentMgmt/record/tracks</requestURL>
  <statusCode>1</statusCode>
  <statusString>OK</statusString>
  <ID>777</ID>
</ResponseStatus>
```

More detailed Single Track XML example (<Schedule> incomplete)

```
<?xml version="1.0" encoding="UTF-8"?>
<TrackList version="1.0" xmlns="urn:psialliance-org">
  <Track>
    <id>1</id>
    <Channel>12345</Channel>
    <Enable>true</Enable>
    <Description>trackType=standard,sourceTag=AXIS210a,contentType=video,codecType=MPEG4-SP,
resolution=640x480,frameRate=20 fps,bitrate=6000 kbps</Description>
    <TrackGUID>{A01AAAAA-BBBB-CCCC-DDDD-033595353625}</TrackGUID>
    <Size>4000000000</Size>
    <Duration>P10DT15H</Duration>
    <DefaultRecordingMode>CMR</DefaultRecordingMode>
    <LoopEnable>true</LoopEnable>
    <SourceDescriptor>
      <SrcGUID>{E800A543-9D53-4520-8BB8-9509062C692D}</SrcGUID>
      <SrcChannel>1</SrcChannel>
      <StreamHint>video, mp4, 640x480, 20 fps, 6000 kbps</StreamHint>
      <SrcDriver>RTP/RTSP</SrcDriver>
      <SrcType>mp4 video</SrcType>
      <SrcUrl>rtsp://10.3.2.26/mpeg4/media.amp</SrcUrl>
      <SrcUrlMethods>DESCRIBE, SETUP, PLAY, TEARDOWN</SrcUrlMethods>
      <SrcLogin>admin:admin</SrcLogin>
    </SourceDescriptor>
    <TrackSchedule>
      <ExternalScheduleBlockReferences>
        <ScheduleBlockReference>
          <ScheduleBlockGUID>{F018AD02-BC04-4520-8BB8-123409AC5678}</ScheduleBlockGUID>
        </ScheduleBlockReference>
        <ScheduleReference>
          <ScheduleBlockGUID>{C2F37123-DD19-4520-8BB8-444307DB5565}</ScheduleBlockGUID>
        </ScheduleReference>
      </ExternalScheduleBlockReferences>
      <ScheduleBlock>
        <ScheduleBlockGUID>{ABC12345-CDEF-4520-8BB8-7135789C8790}</ScheduleBlockGUID>
        <ScheduleBlockType>/psia/recording/schedule/default</ScheduleBlockType>
        <ScheduleAction>
          <id>1</id>
          <ScheduleActionStartTime>
            <DayOfWeek>Monday</DayOfWeek>
            <!-- inclusive -->
            <TimeOfDay>00:00:00</TimeOfDay>
          </ScheduleActionStartTime>
          <ScheduleActionEndTime>
            <DayOfWeek>Monday</DayOfWeek>
```

```
            <!-- exclusive -->
            <TimeOfDay>08:00:00</TimeOfDay>
          </ScheduleActionEndTime>
          <ScheduleDSTEnable>true</ScheduleDSTEnable>
          <Description>PreMorning (Midnight to 8am, local time)</Description>
          <Actions>
            <Record>true</Record>
            <Log>true</Log>
            <SaveImg>true</SaveImg>
            <ActionRecordingMode>EDR</ActionRecordingMode>
            <PreRecordTimeSeconds>30</PreRecordTimeSeconds>
            <PostRecordTimeSeconds>30</PostRecordTimeSeconds>
            <Fps>30.0</Fps>
            <RateKbps>3000</RateKbps>
          </Actions>
        </ScheduleAction>
        <ScheduleAction>
          <id>2</id>
          <ScheduleActionStartTime>
            <DayOfWeek>Monday</DayOfWeek>
            <!-- inclusive -->
            <TimeOfDay>08:00:00</TimeOfDay>
          </ScheduleActionStartTime>
          <ScheduleActionEndTime>
            <DayOfWeek>Monday</DayOfWeek>
            <!-- exclusive -->
            <TimeOfDay>12:00:00</TimeOfDay>
          </ScheduleActionEndTime>
          <ScheduleDSTEnable>true</ScheduleDSTEnable>
          <Description>Morning (8am to noon, local time) </Description>
          <Actions>
            <Record>true</Record>
            <Log>true</Log>
            <SaveImg>true</SaveImg>
            <ActionRecordingMode>CMR</ActionRecordingMode>
            <PreRecordTimeSeconds>30</PreRecordTimeSeconds>
            <PostRecordTimeSeconds>30</PostRecordTimeSeconds>
            <Fps>30.0</Fps>
            <RateKbps>3000</RateKbps>
          </Actions>
        </ScheduleAction>
        <ScheduleAction>
          <id>3</id>
          <ScheduleActionStartTime>
            <DayOfWeek>Monday</DayOfWeek>
            <!-- inclusive -->
            <TimeOfDay>12:00:00</TimeOfDay>
          </ScheduleActionStartTime>
          <ScheduleActionEndTime>
            <DayOfWeek>Monday</DayOfWeek>
            <!-- exclusive -->
            <TimeOfDay>1:00:00</TimeOfDay>
          </ScheduleActionEndTime>
          <ScheduleDSTEnable>true</ScheduleDSTEnable>
          <Description>Lunch (noon to 1pm, local time)</Description>
          <Actions>
            <Record>true</Record>
            <Log>true</Log>
            <SaveImg>true</SaveImg>
            <ActionRecordingMode>EDR</ActionRecordingMode>
```

```
            <PreRecordTimeSeconds>30</PreRecordTimeSeconds>
            <PostRecordTimeSeconds>30</PostRecordTimeSeconds>
            <Fps>30.0</Fps>
            <RateKbps>3000</RateKbps>
          </Actions>
        </ScheduleAction>
        <ScheduleAction>
          <id>4</id>
          <ScheduleActionStartTime>
            <DayOfWeek>Monday</DayOfWeek>
            <!-- inclusive -->
            <TimeOfDay>13:00:00</TimeOfDay>
          </ScheduleActionStartTime>
          <ScheduleActionEndTime>
            <DayOfWeek>Monday</DayOfWeek>
            <!-- exclusive -->
            <TimeOfDay>17:00:00</TimeOfDay>
          </ScheduleActionEndTime>
          <ScheduleDSTEnable>true</ScheduleDSTEnable>
          <Description>Afternoon (1pm to 5pm, local time)</Description>
          <Actions>
            <Record>true</Record>
            <Log>true</Log>
            <SaveImg>true</SaveImg>
            <ActionRecordingMode>CMR</ActionRecordingMode>
            <PreRecordTimeSeconds>30</PreRecordTimeSeconds>
            <PostRecordTimeSeconds>30</PostRecordTimeSeconds>
            <Fps>30.0</Fps>
            <RateKbps>3000</RateKbps>
          </Actions>
        </ScheduleAction>
        <ScheduleAction>
          <id>5</id>
          <ScheduleActionStartTime>
            <DayOfWeek>Monday</DayOfWeek>
            <!-- inclusive -->
            <TimeOfDay>17:00:00</TimeOfDay>
          </ScheduleActionStartTime>
          <ScheduleActionEndTime>
            <DayOfWeek>Tuesday</DayOfWeek>
            <!-- exclusive -->
            <TimeOfDay>00:00:00</TimeOfDay>
          </ScheduleActionEndTime>
          <ScheduleDSTEnable>true</ScheduleDSTEnable>
          <Description>Night (5pm to midnight, local time)</Description>
          <Actions>
            <Record>true</Record>
            <Log>true</Log>
            <SaveImg>true</SaveImg>
            <ActionRecordingMode>EDR</ActionRecordingMode>
            <PreRecordTimeSeconds>30</PreRecordTimeSeconds>
            <PostRecordTimeSeconds>30</PostRecordTimeSeconds>
            <Fps>30.0</Fps>
            <RateKbps>3000</RateKbps>
          </Actions>
        </ScheduleAction>
        <!-- …etc, rest of week, definined similar to Monday schedule items -->
      </ScheduleBlock>
    </TrackSchedule>
    <MetadataEvtCfgList>
```

```
      <MetadataEvtCfg>
        <id>1</id>
        <!-- Motion in Zone2 of src cam: Operate in (or switch to) Event-Driven Mode -->
        <evDomain>metadata.psia.org</evDomain>
        <evClass>vmd</evClass>
        <evType>motionAny</evType>
        <evAttribute>zone2</evAttribute>
        <evSrcGUID>{E800A543-9D53-4520-8BB8-9509062C692D}</evSrcGUID>
        <evSrcUrl>TBD-CEM XPORT</evSrcUrl>
        <evPriority>4</evPriority>
        <Actions>
          <Record>true</Record>
          <Log>true</Log>
          <SaveImg>false</SaveImg>
          <ActionRecordingMode>EDR</ActionRecordingMode>
          <PreRecordTimeSeconds>30</PreRecordTimeSeconds>
          <PostRecordTimeSeconds>30</PostRecordTimeSeconds>
          <Fps>30.0</Fps>
          <RateKbps>3000</RateKbps>
        </Actions>
      </MetadataEvtCfg>
      <MetadataEvtCfg>
        <id>2</id>
        <!-- Motion in OTHER cam: Switch to Continuous Recording Mode -->
        <evDomain>metadata.psia.org</evDomain>
        <evClass>vmd</evClass>
        <evType>motionAny</evType>
        <evAttribute>zone2</evAttribute>
        <evSrcGUID>{7611CAB6-16A9-441a-BC02-D5017F5F79EF</evSrcGUID>
        <evSrcUrl>TBD-CEM XPORT</evSrcUrl>
        <evPriority>4</evPriority>
        <Actions>
          <Record>true</Record>
          <Log>true</Log>
          <SaveImg>false</SaveImg>
          <ActionRecordingMode>CMR</ActionRecordingMode>
          <PreRecordTimeSeconds>30</PreRecordTimeSeconds>
          <PostRecordTimeSeconds>30</PostRecordTimeSeconds>
          <Fps>30.0</Fps>
          <RateKbps>3000</RateKbps>
        </Actions>
      </MetadataEvtCfg>
      <MetadataEvtCfg>
        <!-- Any Emergency: switch to Continuous Recording Mode -->
        <id>3</id>
        <evDomain/>
        <evClass/>
        <evType/>
        <evAttribute/>
        <evSrcGUID>{E800A543-9D53-4520-8BB8-9509062C692D}</evSrcGUID>
        <evSrcUrl>TBD-CEM XPORT</evSrcUrl>
        <evPriority>0</evPriority>
        <Actions>
          <Record>true</Record>
          <Log>true</Log>
          <SaveImg>false</SaveImg>
          <ActionRecordingMode>CMR</ActionRecordingMode>
          <PreRecordTimeSeconds>30</PreRecordTimeSeconds>
          <PostRecordTimeSeconds>30</PostRecordTimeSeconds>
          <Fps>30.0</Fps>
```

```
          <RateKbps>3000</RateKbps>
        </Actions>
      </MetadataEvtCfg>
    </MetadataEvtCfgList>
  </Track>
</TrackList>
```

7.3.11 Track List Schema

XSD: To get the latest version of all RaCM XSDs, including this one (cmTrackList.xsd) please download from the PSIA documents website: http://www.psialliance.org/documents_download.html

7.4 /PSIA/ContentMgmt/record/control

This resource is used to send explicit control stimuli to the "record" service.

7.4.1 /PSIA/ContentMgmt/record/control/manual/start

URI	/PSIA/ContentMgmt/record/control/manual/start/tracks/<id>	**Type**	Resource
Requirement Level	Basic		
Function	Description of the REST method parameters and formats available to functionally manipulate the 'record/control/manual/start' resource.		
Methods	**Query String(s)**	**Inbound Data**	**Return Result**
GET	N/A	N/A	<ResourceDescription>
PUT	N/A	None	<ResponseStatus>
POST	N/A	N/A	<ResponseStatus w/error code>
DELETE	N/A	N/A	<ResponseStatus w/error code>
Notes	This resource is used to manually Start the recording track, regardless of recording mode. TBD: optional Metadata (Manual-Event xml) can be sent with the PUT Inbound Data for the Recorder to save. To Enable or Disable (i.e. permanent Stop) the track, the configuration interface should be used to update the track configuration object to set the enable/disable value accordingly.		

7.4.2 /PSIA/ContentMgmt/record/control/manual/stop

URI	/PSIA/ContentMgmt/record/control/manual/stop/tracks/<id>		Type	Resource
Requirement Level	Basic			
Function	Description of the REST method parameters and formats available to functionally manipulate the 'record/control/manual/stop' resource.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET	N/A	N/A	<ResourceDescription>	
PUT	N/A	None	<ResponseStatus>	
POST	N/A	N/A	<ResponseStatus w/error code>	
DELETE	N/A	N/A	<ResponseStatus w/error code>	
Notes	This resource is used to manually Stop the recording track, regardless of recording mode. To Enable or Disable (i.e. permanent Stop) the track, the configuration interface should be used to update the track configuration object to set the enable/disable value accordingly.			

7.4.3 /PSIA/ContentMgmt/record/control/lock

URI	/PSIA/ContentMgmt/record/control/locks		Type	Resource
Requirement Level	Basic			
Function	Description of the REST method parameters and formats available to functionally manipulate the '/PSIA/ContentMgmt/record/control/locks' resource.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET	None	None	<RecordingLockList>	
PUT	None	<RecordingLockList>	<ResponseStatus>	
POST	None	<RecordingLock>	<ResponseStatus>	
DELETE	N/A	N/A	<ResponseStatus>	
Notes	Used to manage the list of recording locks. (see "PSIA-REST List-Entry <id> Creation method",above.)			

URI	/PSIA/ContentMgmt/record/control/locks/<id>		Type	Resource
Requirement Level	Basic			
Function	Resource used to manage a single <RecordingLock> entry.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET	None	None	<RecordingLock>	
PUT	None	<RecordingLock>	<ResponseStatus>	
POST	N/A	N/A	<ResponseStatus w/error code>	
DELETE	None	None	<ResponseStatus>	
Notes				

Example XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<RecordingLockList version="1.0" xmlns="urn:psialliance-org">
  <RecordingLock>
    <!-- lock time range on particular Channel -->
    <id>1</id>
    <TrackId>12345</TrackId>
    <StartDateTime>2009-01-1T09:00:00-06:00</StartDateTime>
    <EndDateTime>2009-01-7T17:00:00-06:00</EndDateTime>
    <SrcGUID/>
  </RecordingLock>
  <RecordingLock>
    <!-- lock time range for all recorded channels -->
    <id>2</id>
    <TrackId>all</TrackId>
    <StartDateTime>2009-07-4T09:00:00-06:00</StartDateTime>
    <EndDateTime>2009-07-4T17:00:00-06:00</EndDateTime>
    <SrcGUID/>
  </RecordingLock>
  <RecordingLock>
    <!-- lock time range for particular source (GUID) -->
    <id>3</id>
    <TrackId>any</TrackId>
    <StartDateTime>2009-07-7T09:00:00-06:00</StartDateTime>
    <EndDateTime>2009-07-7T17:00:00-06:00</EndDateTime>
    <SrcGUID>{E800A543-9D53-4520-8BB8-9509062C692D}</SrcGUID>
  </RecordingLock>
</RecordingLockList>
```

XSD: To get the latest version of all RaCM XSDs, including this one (cmTrackLockList.xsd) please download from the PSIA documents website: http://www.psialliance.org/documents_download.html

7.5 /PSIA/ContentMgmt/record/metadata

This is a PLACEHOLDER for future development. Implementers can choose to implement this resourse as an experimental/custom feature, but no compliance testing will be done against this resource as of the v1.0-v1.1 timeframe.

This REST Interface is used push metadata into the Recorder's archive. Normally a track can be configured to record media (A/V) or metadata by having the recorder "subscribe" to a metadata source (i.e. PULL model). This interface allows for an alternate means of getting metadata saved to the archive. An example use of this function is the media tagging (annotation) scenario.

The saved metadata is retrieved using PSIA metadata transports, or, if recorded within a track, may simply be streamed using the normal streaming transports (RSTP/RTP), though the data encapsulation will comply with the forthcoming PSIA "Common Metadata/Event Management" specification.

URI	/PSIA/ContentMgmt/record/metadata		**Type**	Resource
Requirement Level	Full			
Function	Description of the REST method parameters and formats available to functionally manipulate the '/PSIA/ContentMgmt/record/metadata' resource.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET	N/A	N/A	<ResponseStatus w/error code>	
PUT	N/A	N/A	<ResponseStatus w/error code>	
POST	N/A	Metadata(*)	<ResponseStatus>	
DELETE	N/A	N/A	<ResponseStatus w/error code>	
Notes	This resource is used to push metadata into the Recorder for saving to the metadata archive. The actual storage format and methodologies are implementation			

8 /PSIA/ContentMgmt/schedules

This resource is used to manage the **recording schedule database** (i.e. schedules that are not embedded with the <Track> configuration objects).

URI	/PSIA/ContentMgmt/schedules		Type	Resource
Requirement Level	Full			
Function	Description of the REST method parameters and formats available to functionally manipulate the schedules resource.			
Methods	**Query Strin (s)**	**Inbound Data**	**Return Result**	
GET	None	None	<ScheduleBlockList>	
PUT	N/A	<ScheduleBlockList>	<ResponseStatus>	
POST	N/A	<ScheduleBlock>	<ResponseStatus>	
DELETE	N/A	N/A	<ResponseStatus w/error code>	
Notes	**ScheduleBlock Creation**: Unlike Target assigned <id>'s (e.g. <Track> id's), the ScheduleBlockGUID's are assigned by the external Client. However, it is expected that, for an individual <ScheduleBlock> POSTed during Creation, a successful operation will cause the returned <ResponseStatus> to contain a matching copy of the POSTed <ScheduleBlockGUID> value within the <ID> tag, for example:			

```
<?xml version="1.0" encoding="UTF-8"?>
<ResponseStatus version="1.0" xmlns="urn:psialliance-org">
  <requestURL>/PSIA/ContentMgmt/schedules</requestURL>
  <statusCode>1</statusCode>
  <statusString>OK</statusString>
  <ID>{F018AD02-BC04-4520-8BB8-123409AC5678}</ID>
</ResponseStatus>
```

XSD: To get the latest version of all RaCM XSDs, including this one (ScheduleBlockList is defined in cmTrackList.xsd) please download from the PSIA documents website: http://www.psialliance.org/documents_download.html

8.1 /PSIA/ContentMgmt/schedules/<ScheduleBlockGUID>

Once created, individual <ScheduleBlock>'s are managed via:

URI	/PSIA/ContentMgmt/schedules/<ScheduleBlockGUID>	Type	Resource
Requirement Level	Basic		

Function	Resource to address (Read/Update) single <ScheduleBlock> by <ScheduleBlockGUID>.		
Methods	**Query Strin (s)**	**Inbound Data**	**Return Result**
GET	None	None	<ScheduleBlock>
PUT	None	<ScheduleBlock>	<ResponseStatus>
POST	N/A	N/A	<ResponseStatus w/error code>
DELETE	None	None	<ResponseStatus>

9 /PSIA/ContentMgmt/search

This section of the specification defines the operation and parameters associated with the 'search' service within the PSIA Content Management hierarchy. Tables, examples and schemas are provided for defining and explaining the search functions.

9.1 /PSIA/ContentMgmt/search/profile

Due to the complexity of the functions entailed in a recording and content management (RaCM) device, all RaCM devices MUST provide a 'profile' resource (schema instance) such that entities accessing them may determine their functional level and basic attributes. Details are outlined below.

URI	/PSIA/ContentMgmt/search/profile		**Type**	Resource
Requirement Level	- All Profiles -			
Function	RaCM Mandatory REST resource/object that publishes the functional profile/level of a RaCM device and its operable service/resource structure.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET	None		<CMSearchProfile>	
PUT	N/A	N/A	<ResponseStatus w/error code>	
POST	N/A	N/A	<ResponseStatus w/error code>	
DELETE	N/A	N/A	<ResponseStatus w/error code>	
Notes	The 'GET' request issued to retrieve an instance of the 'CMCapabilities' XML schema.			

Example(s)	<?xml version="1.0" encoding="UTF-8"?> <CMSearchProfile version="1.0" xmlns="psiallianxce.org:resourcedescription"> <searchProfile>full</searchProfile> <textSearch>true</textSearch> <maxSearchTimespans>2</maxSearchTimespans> <maxSearchTracks>40</maxSearchtracks> <maxSearchSources>40</maxSearchSources> <maxSearchMetadatas>16</maxSearchMetadatas> <maxSearchMatchResults>100</maxSearchMatchResults> <maxSearchTimeout>120</maxSearchTimeout> <maxConcurrentSearches>8</maxConcurrentSearches> </CMSearchProfile> The above example represents a hypothetical RaCM device that supports a 'Full' profile for search its functionality (see below for more details). The succeeding "textSearch" parameter indicates that this device performs raw text string searches on recorded text, and text-based metadata such as Point-of-Sale, or Automated Teller Machine output. The next set of optional parameters (though recommended for Full devices) indicates the maximum number of specific search parameters that can be part of a single instance of a Search criteria. The following parameter, "maxSearchResults" indicates the maximum number of results the RaCM will pass back per search instance. The optional "maxSearchTimeout" parameter indicates that the RaCM device will timeout searches that exceed two minutes (120 seconds) to execute. The final parameter, which is optional, indicates the maximum number of concurrent search operations the RaCM device can support.

9.1.1 PSIA/ContentMgmt/search/profile Schema Definition

The ContentMgmt/search/profile schema is used to define the types of searches a RaCM device supports. A device with a search profile of 'Basic' only performs searches with one timespan per search request (see "*/PSIA/ContentMgmt/search*"). Devices that support the 'Full' search profile must outline their parameter limits, as described in the following schema.

XSD: To get the latest version of all RaCM XSDs, including this one (cmSearchProfile.xsd) please download from the PSIA documents website:
http://www.psialliance.org/documents_download.html

9.2 /PSIA/ContentMgmt/search

The Content Management 'Search' service is the primary component for conducting searches of content bases managed by a PSIA recording device. Fundamentally, searches are initiated using a parameter-based criteria set which is conveyed by the initiator to the device via the 'CMSearchDescription' XML schema. *Since not all programming languages allow content bodies with HTTP GET methods, both 'GET' and 'POST' are supported as message types for initiating searches*. The responding device passes back the results in a 'CMSearchResult' XML schema instance for search requests that had valid syntax. If a search request is syntactically invalid (i.e. no payload, malformed schema instance, etc.), an HTTP response with status code 400 (Bad Request) and a corresponding 'Response Status' are returned to the requester. Please note that a syntactically correct search, that has no matching criteria, returns a 'CMSearchResult' schema instance with a 'NO MATCHES' status string.

Essentially, most searches are conducted based on time and/or track and/or source related

search parameters. Full profile Content Management devices also support the potential for metadata search related parameters. More details, and examples, follow.

<table>
<tr><td>URI</td><td colspan="2">/PSIA/ContentMgmt/search/</td><td>Type</td><td>Service</td></tr>
<tr><td>Requirement Level</td><td colspan="4">- All Profiles -</td></tr>
<tr><td>Function</td><td colspan="4">Mandatory description of the REST method parameters and formats available to functionally manipulate the ‘search’ resource/object.</td></tr>
<tr><td>Methods</td><td>Query String(s)</td><td>Inbound Data</td><td colspan="2">Return Result</td></tr>
<tr><td>GET</td><td>None</td><td><CMSearchDescription></td><td colspan="2"><CMSearchResult or…></td></tr>
<tr><td>PUT</td><td>N/A</td><td>N/A</td><td colspan="2"><ResponseStatus w/error code></td></tr>
<tr><td>POST</td><td>None</td><td><CMSearchDescription></td><td colspan="2"><CMSearchResult or…ResponseStatus w/error code></td></tr>
<tr><td>DELETE</td><td>N/A</td><td>N/A</td><td colspan="2"><ResponseStatus w/error code></td></tr>
<tr><td>Notes</td><td colspan="4">The ‘GET’ or ‘POST’ messages require a “CMSearchDescription” XML document to engage a search. The schema definition is described in Section 10.4.1. An example XML document instance follows.</td></tr>
<tr><td>Example(s)</td><td colspan="4">

```
<?xml version="1.0" encoding="UTF-8"?>
<CMSearchDescription version="1.0" xmlns="urn:psialliance-org">
   <searchID>{812F04E0-4089-11A3-9A0C-0305E82C2906}</searchID>
   <trackIDList>
      <trackID>9</trackID>
      <trackID>22</trackID>
      <trackID>43</trackID>
   </trackIDList>
   <timeSpanList>
      <timeSpan>
         <startTime>2009-06-10T12:00:00Z</startTime>
         <endTIme>2009-06-10T13:30:00Z</endTime>
      <timeSpan>
   </timeSpanList>
   <contentTypeList>
      <contentType>video</contentType>
   </contentTypeList>
   <maxResults>40</maxResults>
   <metadataList>
   <metadataDescriptor>//metadata.psia.org/VideoMotion</metadataDescriptor>
   <metadataList>
</CMSearchDescription>
```

The above example is for a search of tracks 9, 22, and 43, between twelve noon and 1:30PM on June 10th, 2009 for Video Motion events. The requester does not want more than 40 results passed back in the search response, and, that matching results should only be video segments (based on the above “contentTypeList”).
</td></tr>
</table>

```
<?xml version="1.0" encoding="UTF-8"?>
<CMSearchDescription version="1.0" xmlns="urn:psialliance-org">
   <searchID>{77E105E8-4C21-AA05-37B4-189A070A5B22}</searchID>
   <sourceID> {3F2504E0-4F89-11D3-9A0C-0305E82C3301} </sourceID>
   <timeSpanList>
      <timeSpan>
         <startTime>2009-07-12T09:00:00Z</startTime>
         <endTIme>2009-07-12T17:30:00Z</endTime>
      <timeSpan>
   </timeSpanList>
   <contentTypeList>
      <contentType>video</contentType>
      <contentType>audio</contentType>
   </contentTypeList>
   <maxResults>40</maxResults>
   <metadataList>
   <metadataDescriptor>/metadata.psia.org/VideoMotion</metadataDescriptor>
   <metadataList>
</CMSearchDescription>
```

The above example is a search for Video Motion, between the hours of 9AM and 5:30PM, on July the 12th, with respect to a specific source whose ID (GUID) is: 3F2504E0-4F89-11D3-9A0C-0305E82C3301. This example GUID corresponds to an input source which can be resolved to an IP address. Please note that in this example, the search requester only wants video and audio segments returned for matches (contenTypeList).

```
<?xml version="1.0" encoding="UTF-8"?>
<CMSearchDescription version="1.0" xmlns="urn:psialliance-org">
   <searchID>{F44EC031-4F89-3D90-9A14-0305E828D902}</searchID>
   <timeStateList>
      <trackState>inactive</trackState>
      <trackState>locked</trackState>
      <trackState>errored</trackState>
   </timeStateList>
   <searchTimeout>150</searchTimeout>
</CMSearchDescription>
```

The above example is a simple search for all tracks that are in an inactive, or locked, or errored state. These states are described in detail in Section 11.4 of this document. Using track status as a search parameter only makes sense with track-related parameters, or as a filter to remove unwanted tracks from a search. Also noted in this example is the inclusion of the optional 'timeout' parameter ("searchTimeout"). This parameter is only valid if the RaCM device indicates in its search "capabilities" that is supports timeouts. In this example, the requester is indicating that is desires the search to take no longer than 2.5 minutes (150 seconds) to complete.

```
<?xml version="1.0" encoding="UTF-8"?>
<CMSearchDescription version="1.0" xmlns="urn:psialliance-org"
   <searchID>{812F04E0-4089-11A3-9A0C-0305E82C2906}</searchID>
   <timeSpanList>
      <timeSpan>
         <startTime>2009-06-10T12:00:00Z</startTime>
         <endTIme>2009-06-10T13:30:00Z</endTime>
      <timeSpan>
   </timeSpanList>
   <contentTypeList>
      <contentType>text</contentType>
      <contentType>metadata</contentType>
   </contentTypeList>
   <metadataList>
        <metadataDescriptor>//metadata.psia.org/PointOfSale</metadataDescriptor>
        <metadataDescriptor>//metadata.psia.org/User</metadataDescriptor>
   <metadataList>
   <searchText>"VOID"</searchText>
</CMSearchDescription>
```

The above example is for a search of all text and metadata associated with the Source 812F04E0-4089-11A3-9A0C-0305E82C2906 between twelve noon and 1:30PM on June 10th, 2009. The requester wants recorded text and metadata (based on the "contentTypeList") searched for the word "VOID". The metadata categories that the requester wants searched are restricted to Point-Of-Sale and User based metadata. Please note that ALL text searches require a "contentTypeList" to specify the content types to be searhed; valid types are "text" and/or "metadata".

9.2.1 Search Query Parameter Schema Definition

The following XML schema definition defines the search parameters that are provided to the ContentMgmt/search service. Basically, an entity can search based on time and/or source and/or tracks and/or metadata. Basic profile devices are not required to support metadata in their search criteria. Each search must be given a unique ID ("searchID") by the initiator. This ID is an ISO/IEC 9834-8/ITU X.667 compliant UUID/GUID. The responder echoes this value back in the search response (see next section) to correlate the results to the corresponding query. Basic profile devices are only required to support one query at-a-time. Full profile devices MUST advertise the number of concurrent search queries they can support (via the "/PSIA/ContentMgmt/search/profile" Resource).

The 'cmSearchDescription' schema allows searches based on: track lists, track state lists, source ID lists, channel ID lists, timespan lists, and metadata lists. Only one of these criteria is required for a valid search.

Optional information includes the ability for an initiator to specify that it only wants to receive a maximum number of responses to a search criteria (i.e. limit the potential number of responses). This is specified by the "maxResults" element. When this limit is specified in a search request, the responding device must not send more than the requested number of responses (matches). If the querying entity specifies a "maxResult" limit that is less than the number of total results that matched the original search criteria, it must re-issue the search query with the "search ResultsPosition" parameter indicating the number of prior results the inquirer has already received. For example, a client application searches for Video Motion events from the prior night and indicates it only wants a maximum of ten results passed back. The ContentMgmt device, while performing the search, discovers 23 matches. However, the device is only allowed to pass back the first ten results based on the inquirer's "maxResults" value. The device does so, but indicates in the search response schema instance a status string of "MORE". After this, if the inquirer desires to retrieve the next set of matching results, it must do so with the same search criteria and a "searchResultsPosition" value that indicates the cumulative number of results received by the inquirer. In other words, the "searchResultsPosition" is a walking index used to indicate where an inquirer desires the searching entity to resume the search.

Another optional search parameter that governs, or conditions, the search result contents is the "contentTypeList". This element allows one or more "contentType" designators. These designators instruct the searching Content Manager to only provide matching result segments that correspond to the designated content types. The supported content types are: "video", "audio", "metadata", "text", "mixed" (polymorphic – v1.1) and "other". If a search specifies one, or more content types, the searching Content Manager will ignore matches to the search criteria if the content segment/track does not match the specified content type(s).

If a RaCM device supports raw text searches, searches are allowed to pass in "searchText" as a search criteria, but only for "text" and/or "metadata" content types. The following conditions and restrictions apply regarding text searching:

- The 'content type' MUST be specified when conducting searches that involve text. The applicable content types are "text" and "metadata".
- There is an inherent limitation of one text string per search instance.
- The search string may not exceed 128 characters.
- ALL text searches are performed in a ***case insensitive*** manner such that any combination of upper and lower casing can be matched ('open matching').
- Any search string that contains whitespace, or special characters, MUST be started and ended with double-quotes (").

Since raw text searches are compute intensive, requesters, and RaCM devices, that honor search timeouts should either: A) ignore timeouts for text searches, or B) timeouts should be lengthened significantly for text searches.

Finally, RaCM devices that support search timeout limits must allow requesters to dynamically/optionally specify 'desired' timeout limits on a per-search basis. Please note that this is a hint to the RaCM device to restrict, in a best effort manner, the search operation duration to the time limit specified in the XML parameter set. If a RaCM device cannot honor the timeout it will process the search as best it can. Additionally, a RaCM device that does not support search timeouts should ignore a timeout if it errantly receives one on search request instance.

XSD: To get the latest version of all RaCM XSDs, including this one (cmSearchDescription.xsd) please download from the PSIA documents website:
http://www.psialliance.org/documents_download.html

9.2.2 Search Query Results Schema

The response to a ContentMgmt/search operation, using the "CMSearchDescription" schema, is defined by the "CMSearchResult" schema definition below. Fundamentally, the response echoes the initiator's search ID and an overall status value. The status consists of 2 parts: A) a boolean, called "responseStatus", indicates overall success or failure, which is followed by B) the "responseStatusString" which indicates a readable status string that further qualifies the overall status (e.g. TRUE + "OK", or TRUE + "MORE" [indicating more matches were found than the initiator desired to receive], or FALSE + "NO MATCHES [search failed to find matches to specified criteria], or FALSE + "INVALID TRACK ID" [indicating an invalid track ID had been provided]). The combination of an overall Boolean status indicator, plus a qualifying string, allows a requester to quickly determine if a search was successful and then have information detailing the operation.

After the response status, the responder provides the number of matches found, and a list of "matchElements" that correspond the each matching multimedia segment. Each "matchElement" contains the following information:

- The source ID of the input source that corresponds to the matching segment;
- The track ID of the multimedia track that corresponds to the matching segment;
- The timespan of the matching segment (since it is probably a subset of the overall search's timespan);
- A media segment descriptor that the search initiator can directly use to playback the corresponding match segment. It contains the following:
    - The content type of the segment (video, audio, metadata, text, other);
    - The codecType of the content in the segment (e.g. "MPEG4-SP" or G.726);
    - The rate attribute of the segment, if applicable;
    - The playback URI for the match segment (i.e. a URI of the responder's definition that contains all the information necessary to 'play' the segment, via RTSP, without the responder having to recommit another search).

Optionally, the responder can include the following information in a "matchElement" where applicable and/or supported:

- The channel ID of the source corresponding the match segment, if the input channel is still active (i.e. 'live');
- For metadata searches, the fully qualified PSIA Domain/Class/Type REST URI(s) of the corresponding events that correlate to the match segment.

9.2.2.1 Search Response Examples

The following example XML schema instances describe some of the potential responses to search queries. Please note that these examples are provided for reference but there are still many potential scenarios not directly addressed.

```
<?xml version="1.0" encoding="UTF-8"?>
<CMSearchResult version="1.0" xmlns="urn:psialliance-org">
  <searchID>{812F04E0-4089-11A3-9A0C-0305E82C2906}</searchID>
  <responseStatus>true</responseStatus>
  <responseStatusStrg>OK</responseStatusStrg>
  <numOfMatches>4</numOfMatches>
  <matchList>
    <matchElement>
      <sourceID>{b049902e72-0049-1158-c0d2-7e330680d93c}</sourceID>
      <trackID>27</trackID>
      <timeSpan>
        <startTime>2009-05-18T10:31.26</startTime>
        <endTime>2009-05-18T10:32:54</endTime>
      </timeSpan>
```

```
      <mediaSegmentDescriptor>
        <contenType>video</contenType>
        <codecType>MPEG4-SP</codecType>
        <rateType>”3 Mbps, 30 fps”</rateType>

<playbackURI>rtsp://144.70.13.92:554/PSIA/Streaming/tracks/27?offset=a07724&amp;endtime=2009-05-18T1
0:31.25</playbackURI>
      </mediaSegmentDescriptor>
      <metadataList>
        <metadataDescriptor>//metadata.psia.org/VideoMotion/motion</metadataDescriptor>
      </metadataList>
    </matchElement>
    <matchElement>
      <sourceID>{b049902e72-0049-1158-c0d2-7e330680755e}</sourceID>
      <trackID>30</trackID>
      <timeSpan>
        <startTime>2009-05-18T10:40.03</startTime>
        <endTime>2009-05-18T10:41:04</endTime>
      </timeSpan>
      <mediaSegmentDescriptor>
        <contenType>video</contenType>
        <codecType>H.264-BP</codecType>
        <rateType>”1 Mbps, 15 fps”</rateType>

<playbackURI>rtsp://144.70.13.92:554/PSIA/Streaming/tracks/30?offset=a08c94&amp;endtime=2009-05-18T1
0:41.04</playbackURI>
      </mediaSegmentDescriptor>
      <metadataList>
        <metadataDescriptor>//metadata.psia.org/VideoMotion/motionLeft</metadataDescriptor>
      </metadataList>
    </matchElement>
    <matchElement>
      <sourceID>{b049902e72-0049-1158-c0d2-7e330680755e}</sourceID>
      <trackID>31</trackID>
      <timeSpan>
        <startTime>2009-05-18T10:40.03</startTime>
        <endTime>2009-05-18T10:41:04</endTime>
      </timeSpan>
      <mediaSegmentDescriptor>
        <contenType>audio</contenType>
        <codecType>G.726</codecType>
        <rateType>”24 Kbps”</rateType>

<playbackURI>rtsp://144.70.13.92:554/PSIA/Streaming/tracks/31?offset=1446a90e&amp;endtime=2009-05-18
T10:41.04</playbackURI>
      </mediaSegmentDescriptor>
      <metadataList>
        <metadataDescriptor>//metadata.psia.org/VideoMotion/motionLeft</metadataDescriptor>
      </metadataList>
    </matchElement>
    <matchElement>
      <sourceID>{b049902e72-0049-1158-c0d2-7e330680755e}</sourceID>
      <trackID>32</trackID>
      <timeSpan>
        <startTime>2009-05-18T10:40.03</startTime>
        <endTime>2009-05-18T10:41:04</endTime>
      </timeSpan>
      <mediaSegmentDescriptor>
        <contenType>metadata</contenType>
        <codecType>PSIA-GMCH</codecType>
        <rateType>none</rateType>
        <playbackURI>rtsp://144.70.13.92:554/PSIA/Streaming/tracks/32?offset=93e6a90e&amp;endtime=2 00
9-05-18T10:41.04</playbackURI>
      </mediaSegmentDescriptor>
      <metadataList>
        <metadataDescriptor>//metadata.psia.org/VideoMotion/motionLeft</metadataDescriptor>
      </metadataList>
    </matchElement>
  </matchList>
</CMSearchResult>
```

The above example is in response to a search query for video motion events, with respect to a

set of specific sources, i.e. b049902e72-0049-1158-c0d2-7e330680d93c and b049902e72-0049-1158-c0d2-7e330680755e, for video motion events between 10:30 and 10:45AM on May 18th, 2009. Four matches ("matchElements") were found that matched the search criteria. Three of the match elements are video/audio/metadata segments corresponding to the same source (see "sourceID"). The status combination of "true" and "OK" indicate a complete response. If there had been more matches than the requester had allowed for reply, the status would have been "true" and "MORE" in the "responseStatus" and "responseStatusStrg" fields respectively. The "mediaURIDescriptor"s provided are exemplary only.

The first match, for source b049902e72-0049-1158-c0d2-7e330680d93c, has a segment, on track #27, that is an MPEG-4 video segment. The next two matches are tied to the same source, b049902e72-0049-1158-c0d2-7e330680755e, and are track segments (30 and 31, respectively) that include both H.264 video and G.726 audio. The playback URIs within the match elements are fictional examples that are syntactically correct. Match element playback URIs are opaque to the requester; they are only required to contain whatever information the responder needs to playback the corresponding media segment via RTSP.

```
<?xml version="1.0" encoding="UTF-8"?>
<CMSearchResult version="1.0" xmlns="urn:psialliance-org">
  <searchID>{812F04E0-4089-11A3-9A0C-0305E82C2906}</searchID>
  <responseStatus>false</responseStatus>
  <responseStatusStrg>NO MATCHES</responseStatusStrg>
  <numOfMatches>0</numOfMatches>
</CMSearchResult>
```

In this example, a valid search query had no matches for the original criteria provided.

```
<?xml version="1.0" encoding="UTF-8"?>
<CMSearchResult version="1.0" xmlns="urn:psialliance-org">
  <searchID>{812F04E0-4089-11A3-9A0C-0305E82C2906}</searchID>
  <responseStatus>true</responseStatus>
  <responseStatusStrg>INVALID TRACK ID</responseStatusStrg>
  <numOfMatches>1</numOfMatches>
  <matchList>
    <matchElement>
      <sourceID>{2040002e72-0049-1158-c0d2-7e3306803320}</sourceID>
      <trackID>11</trackID>
      <timeSpan>
        <startTime>2009-05-18T09:48.00</startTime>
        <endTime>2009-05-18T09:52:33</endTime>
      </timeSpan>
      <mediaSegmentDescriptor>
        <contentType>video</contentType>
        <codecType>MPEG4-ASP</codecType>
        <playbackURI>rtsp://144.70.13.92:554/PSIA/Streaming/tracks/11?offset=a07724</playbackURI>
      </mediaSegmentDescriptor>
      <metadataList>
        <metadataDescriptor>/metadata.psia.org/VideoMotion/motion</metadataDescriptor>
      </metadataList>
    </matchElement>
  </matchList>
</CMSearchResult>
```

In the above example, a search query was initiated that contained both a valid, and an invalid, track ID. The responder, in this case, did not fail the search, but found a single match for the valid track ID that was provided; a 4 ½ minute segment within track 11. It indicated that the overall search was successful, but that there was a partial failure since one of the track IDs was invalid. This example does not require the Content Managers to support partial operations. It only outlines the potential for using the "responseStatus" and "responseStatusStrg" fields to indicate various levels of completion.

9.2.2.2 Search Result Schema Definition

XSD: To get the latest version of all RaCM XSDs, including this one (cmSearchResult.xsd) please download from the PSIA documents website: http://www.psialliance.org/documents_download.html

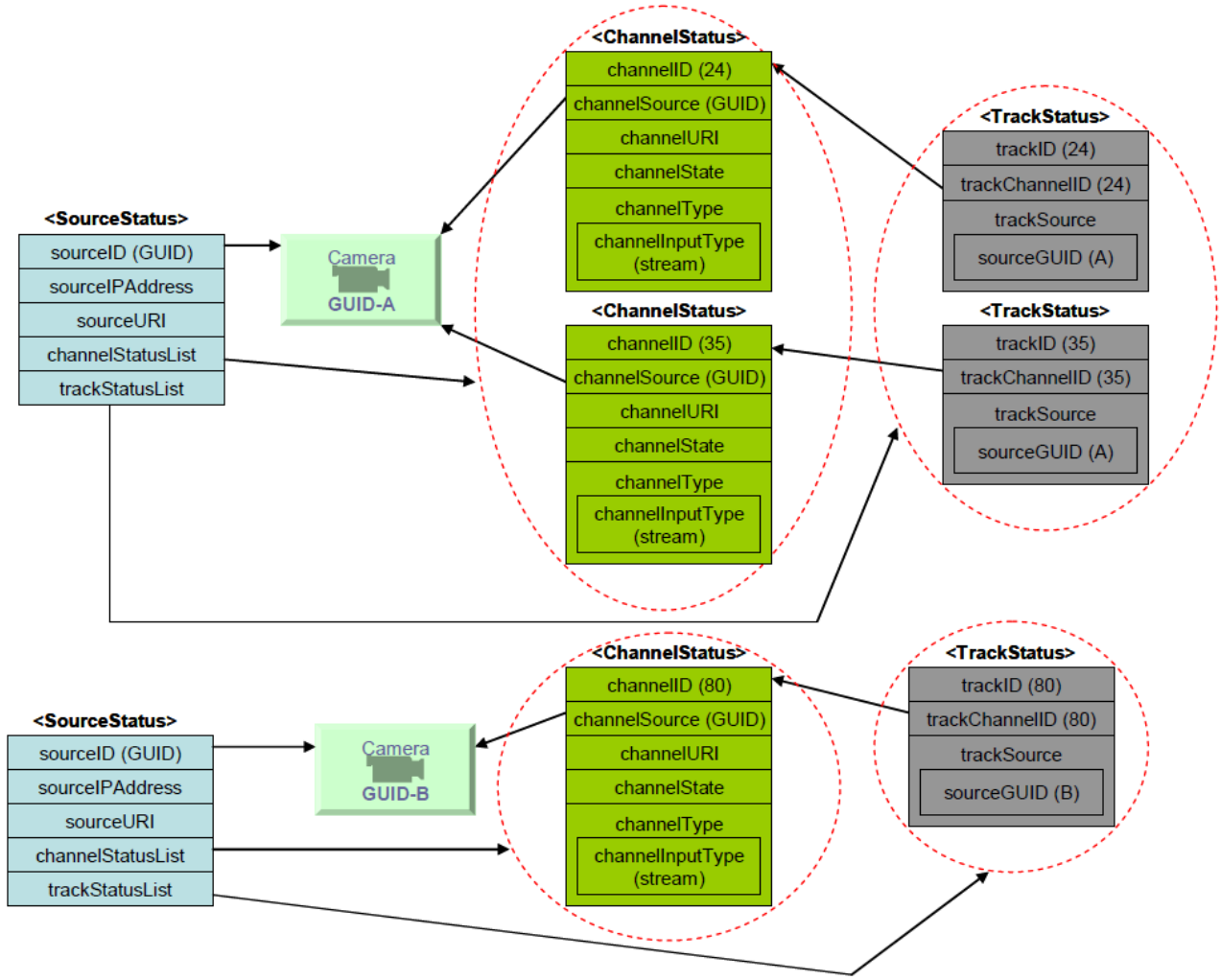
10 /PSIA/ContentMgmt/status

The Content Management resource hierarchy provides operational status information for the following resources:

- The entire content base and the mount 'volumes' that comprise the content base, as a whole;
- Each track within the content base;
- Each input 'channel', which are the equivalent to input streams;
- And, each input 'source' (device or port), which can provide more than one input channel.

This status data is provided in the 4 basic categories listed above. The status information indicates wellness, resource utilization, and basic attributes regarding the configuration of each resource. The description of each of these categories follows

Figure: Example relationship between 'source', 'channel', and 'track' Status values in NVR



In the NVR case, if Multi-Media recording is desired, then each 'channel' must refer to a Multi-Media stream, with the 'channelURI' capable of acquiring a Multi-Media stream.

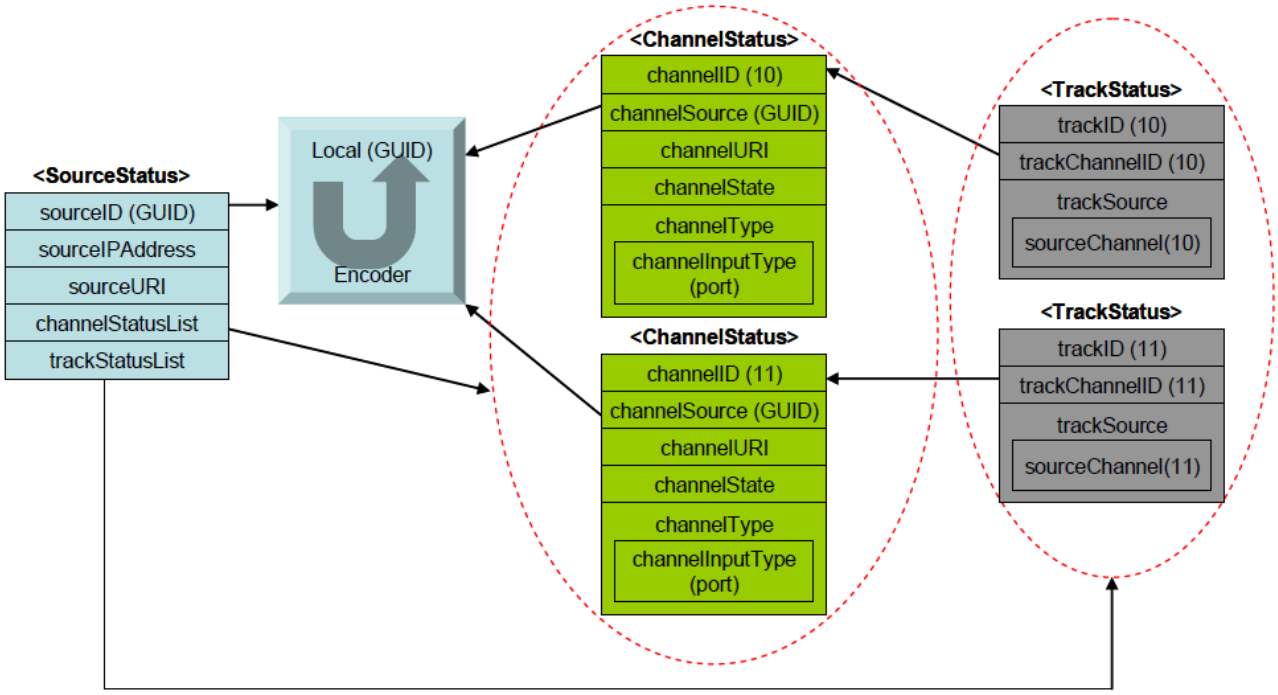


Figure: Example relationship between 'source', 'channel', and 'track' Status values in DVR

In the DVR case, the local stream being recorded may be Multi-Media. For example, the <TrackStatus>/<trackSource>/<sourceChannel> value of 10 is referring to /PSIA/Streaming/channels/10 which **strongly recommended** to be bound to local "/PSIA/System/Video/inputs/channels/10" and "/PSIA/System/Audio/channels/10" (if available).

10.1 /PSIA/ContentMgmt/status/volumes

For each RaCM device, there is at least one mounted 'volume' that provides the logical storage wherein the multimedia is managed (recorded, searched, played). The term "content base" addresses all of the 'mount volumes' that comprise the logical storage supporting the multimedia content. A mount volume is referenced via configuration as a 'Mount' (see "*/PSIA/ContentMgmt/record/storageMounts (Storage Allocation)*") within the storage configuration's "MountList." Each one of these configured "Mounts", and the associated parameters, comprises a "mountVolume". The only difference is that in the configuration of each "Mount" a "path", *and* a directory ("dir"), are provided to setup recording of multimedia content. When status is reported, the path and directory are concatenated a one complete path.

When reporting volume status, the overall state of the content base, as a whole, is always reported, also. Querying entities may retrieve the status of the content base and all volumes via the URI "/PSIA/ContentMgmt/status/volumes", or for a specific volume via the URI "/PSIA/ContentMgmt/status/volumes/<n>" where '<id>' is the Mount ID number of a specific mount volume.

URI	`/PSIA/ContentMgmt/status/volume` *or* `/PSIA/ContentMgmt/status/volumes/<n>`		**Type**	Resource
Requirement Level	- All -			
Function	Mandatory description of the REST method parameters and formats available to functionally manipulate the 'search' resource/object.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	

GET	None	None	<CMStatusVolume>
PUT	N/A	N/A	<ResponseStatus w/error code>
POST	N/A	N/A	<ResponseStatus w/error code>
DELETE	N/A	N/A	<ResponseStatus w/error code>
Notes	Volume status is relatively simple. A querying entity either issues a URI to get all volume status (no Mount index is specified), or a URI is provided for a specific volumes using the configured mount index. An XML document is returned with the content base and volume status. Examples follow.		

Example(s)

```
<?xml version="1.0" encoding="UTF-8"?>
<CMStatusVolume version="1.0" xmlns="urn:psialliance-org">
  <contentBaseStatus>
  <contentBaseSize>970.6</contentBaseSize>
  <contentBaseSizeUnit>GBs</contentBaseSizeUnit>
  <contentBaseID>{409504E0-4F89-11D3-9A0C-0305E82C3301}</contentBaseID>
  <contentBaseStatus>
  <mountVolumeStatusList>
    <mountVolumeStatus>
      <mountVolumePath>d:/archive1</mountVolumePath>
      <mountVolumeSize>970.6</mountVolumeSize>
      <mountVolumeSizeUnit>GBs</mountVolumeSizeUnit>
      <mountVolumeMountID>1</mountVolumeMountID>
      <mountVolumeState>active</mountVolumeState>
      <mountVolumeDate>2009-05-18T10:31.26</mountVolumeDate>
    </mountVolumeStatus>
  </mountVolumeStatusList>
</CMStatusVolume>
```

In the above example, a RaCM device has a content base that consists of a single mount volume that is 970.6 GBs. The content base has a GUID to uniquely identify it. The mount volume ID, which is 1, matches the storage configuration (see "*/PSIA/ContentMgmt/record/storageMounts (Storage Allocation)*"). The volume is "active" which indicates that it is successfully recording, etc. The 'mountVolumeDate' indicates the setup/creation date of for this mount volume.

```
<?xml version="1.0" encoding="UTF-8"?>
<CMStatusVolume version="1.0" xmlns="urn:psialliance-org">
  <contentBaseStatus>
  <contentBaseSize>2.4</contentBaseSize>
  <contentBaseSizeUnit>TBs</contentBaseSizeUnit>
  <contentBaseID>{e6229504E0-4F89-11D3-9A0C-0305E82C950c}</contentBaseID>
  <contentBaseStatus>
  <mountVolumeStatusList>
    <mountVolumeStatus>
      <mountVolumePath>/dev/hd0/archive1</mountVolumePath>
      <mountVolumeSize>1.2</mountVolumeSize>
      <mountVolumeSizeUnit>TBs</mountVolumeSizeUnit>
      <mountVolumeMountID>1</mountVolumeMountID>
      <mountVolumeState>locked</mountVolumeState>
      <mountVolumeDate>2007-10-22T13:40.17</mountVolumeDate>
    </mountVolumeStatus>
    <mountVolumeStatus>
      <mountVolumePath>/dev/hd1/archive1</mountVolumePath>
      <mountVolumeSize>1.2</mountVolumeSize>
      <mountVolumeSizeUnit>TBs</mountVolumeSizeUnit>
      <mountVolumeMountID>2</mountVolumeMountID>
      <mountVolumeState>active</mountVolumeState>
      <mountVolumeDate>2009-04-12T20:56.09</mountVolumeDate>
    </mountVolumeStatus>
  </mountVolumeStatusList>
</CMStatusVolume>
```

In the above example, a RaCM device has a 2.4TB content base comprised of two 1.2TB mount volumes with the IDs "1" and "2", respectively. The first volume has been 'locked'; content can be actively searched, etc., but it is in a read- only mode. The second mount volume is active.

```
<?xml version="1.0" encoding="UTF-8"?>
<CMStatusVolume version="1.0" xmlns="urn:psialliance-org">
  <contentBaseStatus>
  <contentBaseSize>8</contentBaseSize>
  <contentBaseSizeUnit>TBs</contentBaseSizeUnit>
  <contentBaseID>{e6229504E0-4F89-11D3-9A0C-0305E82C950c}</contentBaseID>
  <contentBaseStatus>
  <mountVolumeStatusList>
    <mountVolumeStatus>
      <mountVolumePath>/dev/hd0/content1</mountVolumePath>
      <mountVolumeSize>2</mountVolumeSize>
      <mountVolumeSizeUnit>TBs</mountVolumeSizeUnit>
      <mountVolumeMountID>1</mountVolumeMountID>
      <mountVolumeState>active</mountVolumeState>
      <mountVolumeDate>2009-02-12T03:57.01</mountVolumeDate>
    </mountVolumeStatus>
    <mountVolumeStatus>
      <mountVolumePath>//server10/archive1</mountVolumePath>
      <mountVolumeSize>6</mountVolumeSize>
      <mountVolumeSizeUnit>TBs</mountVolumeSizeUnit>
      <mountVolumeMountID>2</mountVolumeMountID>
      <mountVolumeState>errored</mountVolumeState>
      <mountVolumeDate>2009-02-12T03:57.53</mountVolumeDate>
    </mountVolumeStatus>
  </mountVolumeStatusList>
</CMStatusVolume>
```

In the above example, a RaCM device has an 8TB content base comprised of two mount volumes with the IDs "1" and "2", respectively. The first volume is mounted on local hardware and is 2TBs in size. It is active. The second volume, which is 6TBs in size, utilizes a remote mount (i.e. to a network-based machine) which is currently in an 'errored' state. This indicates that the 'mount' is broken and no content is being recorded, nor is any content on this volume currently accessible.

10.1.1 /PSIA/ContentMgmt/status/volume Attribute Definitions

The following tables and schemas define the attributes associated with reporting status for each RaCM device's content base and the mount volumes that comprise that content base. The schema name for this information is "CMVolumeStatus." This schema is comprised of two major sections: A) the content base status, and , B) the mount volume status list. Each content base has the following status properties:

10.1.1.1 Content Base Status Attributes Table

Attribute	Element Name	Requirement	Description
Content base size	"contentBaseSize" "contentBaseSizeUnit"	Mandatory	Two elements indicate the size of a content base: • "contentBaseSize" which is the size indicated in …. • "contentBaseSizeUnit" which indicates the unit of measurement for the size. This can be megabytes, mebibytes, gigabytes, gibibytes, terabytes, petabytes, or exabytes.
Content base ID	"contentBaseID"	Mandatory	The 128-bit UUID/GUID that uniquely identifies the content base. In simple devices, this may be the same UUID/GUID that identifies the device.
Content base name	"contentBaseName"	Optional	A human readable name for the content base (if one is configured).
Content base creation time	"contentBaseCreationTime"	Optional	The 'dateTime' of the creation/setup of the content base.

10.1.1.2 Mount Volume Status Information Table

Following the content base status information, in the schema, is the mount volume status list. This list contains one, or more, sets of information regarding the status of each mount volume in the content base. For each mount volume, the following required status information is provided.

Attribute	Element Name	Requirement	Description
Mount path for the volume	"mountVolumePath"	Mandatory	The complete mount path for the volume being reported. This includes the "path" and "dir" settings setup in the storage configuration (see "/PSIA/ContentMgmt/record/storageMounts (Storage Allocation)").
The mount volume's size	"mountVolumeSize" "mountVolumeSizeUnit"	Mandatory	The total size of the mount volume. Just like the content base size (see above) the size consists of two elements: A) the decimal number, and B) the unit used to report the size (e.g. MBs,

			GBs, etc.).
The 'mount ID' of the volume	"mountVolumeMountID"	Mandatory	This is the configured mount index of the mount volume.
The operational state of the mount volume	"mountVolumeState"	Mandatory	Each mount volume must be in one of the following states: • "active": everything is OK. Content is being actively managed on the volume. • "inactive": The mount volume is setup/configured but is currently inactive (not due to error). Content may be resident on the mount volume. • "locked". The referenced volume is in 'read-only' mode. • "errored". A non-recoverable error has taken the mount volume off-line and into an inactive state. • Mount volume active date: the date/time the corresponding volume was 'activated' (i.e. put in service).
Mount Volume Date	"mountVolumeDate"	Optional	Creation/mount configuration data of the referenced volume.
Mount volume ID	"mountVolumeID"	Optional	An optional 128-bit UUID/GUID that uniquely identifies the associated mount volume.
Mount volume status string	"mountVolumeStateString"	Optional, should be present; see description.	For the 'errored', 'inactive', and 'locked' states (see above), an optional human readable status string may be provided to help better describe the stimulus for the state. This string ***should*** be provided for all errors (i.e. "errored' states).

10.1.2 /PSIA/ContentMgmt/status/volume XSD

XSD: To get the latest version of all RaCM XSDs, including this one (cmVolumeStatus.xsd) please download from the PSIA documents website:
http://www.psialliance.org/documents_download.html

10.2 /PSIA/ContentMgmt/status/sources

Each channel and track within a RaCM device's content base is correlated to a 'source.' A source is the input device that originated the multimedia content managed by the RaCM device. For each source, the RaCM content manager maintains a set of status attributes. These status attributes relate to the source itself, the channels that source is inputting, and the track(s) that correspond to that source. All sources are addressed, or identified, by a 128-bit UUID/GUID (ISO/IEC 9834-8) assigned to that particular device. Status requesters can either gather status for all active sources using the "/PSIA/ContentMgmt/status/source" URI, or for a particular source using the "/PSIA/ContentMgmt/status/sources/<GUID>" URI where "<GUID" is the UUID/GUID string for

the specific source being queried. The following attributes comprise the status information contained in the XML schema for sources.

URI	/PSIA/ContentMgmt/status/sources *or* /PSIA/ContentMgmt/status/sources/<GUID>		Type	Resource
Requirement Level	- All -			
Function	Mandatory description of the REST method parameters and formats available to functionally manipulate the 'search' resource/object.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET	None	None	<CMSourceStatus>	
PUT	N/A	N/A	<ResponseStatus w/error code>	
POST	N/A	N/A	<ResponseStatus w/error code>	
DELETE	N/A	N/A	<ResponseStatus w/error code>	
Notes				

Example(s)

```
<?xml version="1.0" encoding="UTF-8"?>
<CMSourceStatus version="1.0" xmlns="urn:psialliance-org">
  <sourceStatusList>
    <sourceStatus>
      <sourceID>{9034c22e-01aa-3887-4151-b9002ce36d5d}</sourceID>
      <sourceIPAddress>144.70.13.92</sourceIPAddress>
      <channelStatusList>
        <channelStatus>
          <channelID>1</channelID>
          <channelSource>{9034c22e-01aa-3887-4151- b9002ce36d5d}</channelSource>
          <channelURI>rtsp://144.70.13.92:554/PSIA/Streaming/channels/22</channelURI>
          <channelState>active</channelState>
          <channelType>
            <channelInputType>stream</channelInputType>
            <contentType>video</contentType>
            <codecType>MPEG4-SP</codecType>
          </channelType>
        </channelStatus>
        <channelStatus>
          <channelID>2</channelID>
          <channelSource>{9034c22e-01aa-3887-4151- b9002ce36d5d}</channelSource>
          <channelURI>rtsp://144.70.13.92:554/PSIA/Streaming/channels/30</channelURI>
          <channelState>active</channelState>
          <channelType>
            <channelInputType>stream</channelInputType>
            <contentType>audio</contentType>
            <codecType>G.726</codecType>
          </channelType>
        </channelStatus>
      </channelStatusList>
      <trackStatusList>
        <trackStatus>
          <trackID>1</trackID>
          <trackSize>620</trackSize>
          <trackSizeUnit>MBs</trackSizeUnit>
          <trackState>active</trackState>
          <trackMode>CMR</trackMode>
          <trackType>
            <contentType>video</contentType>
            <codecType>MPEG4-SP</codecType>
          </trackType>
          <trackSource>
            <trackSourceGUID>{9034c22e-01aa-3887-4151- b9002ce36d5d} </trackSourceGUID>
          </trackSource>
          <trackChannelID>0</trackChannelID>
        </trackStatus>
        <trackStatus>
          <trackID>2</trackID>
          <trackSize>38</trackSize>
          <trackSizeUnit>MBs</trackSizeUnit>
          <trackState>active</trackState>
          <trackMode>CMR</trackMode>
              <trackType>
            <contentType>audio</contentType>
            <codecType>G.711</codecType>
          </trackType>
          <trackSource>
            <trackSourceGUID>{9034c22e-01aa-3887-4151-b9002ce36d5d} </trackSourceGUID>
          </trackSource>
          <trackChannelID>1</trackChannelID>
        </trackStatus>
      </trackStatusList>
    </sourceStatus>
    <sourceStatus>
      <sourceID>{e70144c03-01aa-3887-4151-3e550991d377}</sourceID>
      <sourceIPAddress>144.70.13.88</sourceIPAddress>
      <channelStatusList>
        <channelStatus>
          <channelID>3</channelID>
          <channelSource>{e70144c03-01aa-3887-4151-3550991d3778}</channelSource>

<channelURI>rtsp://144.70.13.88:554/PSIA/Streaming/channels/10</channelURI>
          <channelState>active</channelState>
          <channelType>
            <channelInputType>stream</channelInputType>
            <contentType>video</contentType>
            <codecType>H.264-BP</codecType>
          </channelType>
           </channelStatus>
        <channelStatus>
```

```
          <channelSource>{e70144c03-01aa-3887-4151-3550991d3778}</channelSource>
        <channelURI>rtsp://144.70.13.88:554/PSIA/Streaming/channels/11</channelURI>
        <channelState>active</channelState>
        <channelType>
          <channelInputType>stream</channelInputType>
          <contentType>audio</contentType>
          <codecType>G.726</codecType>
        </channelType>
      </channelStatus>
    </channelStatusList>
    <trackStatusList>
    <trackStatus>
      <trackID>3</trackID>
      <trackSize>510</trackSize>
      <trackSizeUnit>MBs</trackSizeUnit>
      <trackState>active</trackState>
      <trackMode>CMR</trackMode>
      <trackType>
        <contentType>video</contentType>
        <codecType>H.264-BP</codecType>
      </trackType>
      <trackSource>
        <trackSourceGUID>{e70144c03-01aa-3887-4151-3550991d3778}</trackSourceGUID>
      </trackSource>
      <trackChannelID>2</trackChannelID>
    </trackStatus>
    <trackStatus>
      <trackID>4</trackID>
      <trackSize>38</trackSize>
      <trackSizeUnit>MBs</trackSizeUnit>
      <trackState>active</trackState>
      <trackType>
        <contentType>audio</contentType>
        <codecType>G.726</codecType>
      </trackType>
      <trackSource>
        <trackSourceGUID>{e70144c03-01aa-3887-4151-3550991d3778}</trackSourceGUID>
      </trackSource>
      <trackChannelID>3</trackChannelID>
    </trackStatus>
   </sourceStatus>
  </sourceStatusList>
</CMSourceStatus>
```

In the above example, a very simple scenario is depicted. A RaCM device has 2 input sources. Each input source has 2 input channels (4 total), one each of video and audio. Each source also has both of the video/audio channels being recorded as tracks. The first source, "{9034c22e-01aa-3887-4151-b9002ce36d5d}', is an MPEG-4 camera/encoder with G.711 audio. Its input 'channels' are 1 & 2 as assigned by the RaCM device. These channels are 'active' and are being recorded on tracks 1 & 2, respectively. Please note that the channel and track numbers are indigenous to the RaCM device; i.e. they are not the channel numbers of the IP media device. Additionally, second source, "{e70144c03-01aa-3887-4151-3e550991d377}", which is an H.264 camera/encoder, is inputting video and audio (G.726) to the RaCM device on 'channels' 3 and 4. These 'channels' currently 'active' and are being recorded on tracks 3 and 4. This example depicts the minimum status information required. This example is also for a small, simple device that is supporting only 2 inputs. Obviously, more sources, channels, and tracks are possible, and probable. In this case, the URI for requresting status was "/PSIA/ContentMgmt/status/sources" which rendered all of the sources for the RaCM device.

```
<?xml version="1.0" encoding="UTF-8"?>
<CMSourceStatus version="1.0" xmlns="urn:psialliance-org">
  <sourceStatusList>
  <sourceStatus>
    <sourceID>{9034c22e-01aa-3887-4151-b9002ce36d5d}</sourceID>
    <sourceIPAddress>144.70.13.92</sourceIPAddress>
    <channelStatusList>
      <channelStatus>
        <channelID>8</channelID>
        <channelSource>{9034c22e-01aa-3887-4151-b9002ce36d5d}</channelSource>
        <channelURI>rtsp://144.70.13.92/streaming/channels/55</channelURI>
        <channelState>errored</channelState>
        <channelType>
          <channelInputType>stream</channelInputType>
          <contentType>video</contentType>
          <codecType>H.264-BP</codecType>
        </channelType>
        <channelStatusString>'Network Connection Error'</channelStatusString>
      </channelStatus>
    </channelStatusList>
    <trackStatusList>
      <trackStatus>
        <trackID>10</trackID>
        <trackSize>620</trackSize>
        <trackSizeUnit>MBs</trackSizeUnit>
        <trackState>idle</trackState>
        <trackMode>CMR</trackMode>
        <trackType>
          <contentType>video</contentType>
          <codecType>H.264-BP</codecType>
        </trackType>
        <trackSource>
          <trackSourceGUID>{9034c22e-01aa-3887-4151- b9002ce36d5d}</trackSourceGUID>
        </trackSource>
        <trackChannelID>8</trackChannelID>
      </trackStatus>
    </trackStatusList>
  </sourceStatus>
  <sourceStatusList>
</CMSourceStatus>

The above example depicts a source specific query. The source, at IP address 144.70.13.3,
has an error on its video input channel (channel 8 in this reference). Due to the channel
error, track '10' is currently in an idle state since it is receiving no data from the input
channel. The track status of 'idle' indicates that the error is not with the recording media
or software (i.e. the track is active but receiving not data). The REST URI used to retrieve
this status info was:
"/PSIA/ContentMgmt/status/sources/9034c22e-01aa-3887-4151-b9002ce36d5d"
```

10.2.1 /PSIA/ContentMgmt/status/sources Status Attributes

The following status information is returned for each source. The following descriptions cover the information contained in the XML Schema definition which follows in the subsequent section of this document. The name of the schema is "CMSourceStatus". It is comprised of a list of one, or more, sources with the each source having the following status information associated with it.

10.2.1.1 Source Status Attribute Definitions

Attribute	Element Name	Requirement	Description
Source ID	"sourceID"	Mandatory	128-bit UUID/GUID of the corresponding source device
Source IP Address	"sourceIPAddress"	Mandatory	IPv4/IPv6 address of the source device. For DVRs this is the IP address of the RaCM device.

Source URI	"sourceURI"	Optional	URI of the source stream. This is a convenience attribute. It cannot properly describe the URI used for actual media acquisition, if the Source produces multiple streams. In that case, each produced stream must be captured by different 'channels' in RaCM. The channelURI in the ChannelStatus is the actual URI for media acquisition
Source Description	"sourceDescription"	Optional	Informative description of the source since it may be local or remote, on a DVR or NVR.
(Source's) Channel status list	"channelStatusList"	Mandatory	Status list for input channels associated with the source device. The format is covered in Section 11.3
(Source's) Track Status List	"trackStatusList"	Mandatory	Status list for the tracks associated with the source device. The format is coverd in ***Section 11.4.***

As noted above, the specific formats of the channel and track status information are described in the following sections of this document. The IP address and URI information is provided such that a source can be uniquely identified on a network, and the information used to access the source device can be validated.

10.2.2 /PSIA/ContentMgmt/status/sources XSD

As mentioned above, the status information for a source device consists of three primary categories: A) source device information, B) channel status information for input channels associated with the source device, and B) track status information for the tracks (recorded multimedia content) associated with the referenced device.

XSD: To get the latest version of all RaCM XSDs, including this one (cmSourceStatus.xsd) please download from the PSIA documents website: http://www.psialliance.org/documents_download.html

10.3 /PSIA/ContentMgmt/status/channels

As defined in the opening section of this document, the term 'channels' refers to input data streams for a RaCM device. These 'channels' may be incoming IP-based multimedia streams (NVR scenario) or physical input ports (DVR scenario). Either way, a channel corresponds to a specific instance of a data stream. In other words, each video, audio, serial, and metadata input is a channel unto itself. Sources (see prior Section) may originate multiple channels within one session, but each data stream is its own 'channel'.

URI	`/PSIA/ContentMgmt/status/channels` *or* `/PSIA/ContentMgmt/status/channels/<id>`		**Type**	Resource
Requirement Level	- All -			
Function	Mandatory description of the REST method parameters and formats available to functionally manipulate the 'search' resource/object.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET	None	None	<CMChannelStatus>	
PUT	N/A	N/A	<ResponseStatus w/error code>	
POST	N/A	N/A	<ResponseStatus w/error code>	

DELETE	N/A	N/A	<ResponseStatus w/error code>
Notes			

Example(s)

```
<?xml version="1.0" encoding="UTF-8"?>
<CMChannelStatus version="1.0" xmlns="urn:psialliance-org">
  <channelStatusList>
    <channelStatus>
      <channelID>1</channelID>
      <channelSource>{9034c22e-01aa-3887-4151- b9002ce36d5d}</channelSource>
      <channelURI>rtsp://120.71.23.52/streaming/channels/10</channelURI>
      <channelState>active</channelState>
      <channelType>
        <channelInputType>stream</channelInputType>
        <contentType>video</contentType>
        <codecType>MPEG4-SP</codecType>
      </channelType>
    </channelStatus>
    <channelStatus>
      <channeleID>2</channeleID>
      <channelSource>{9034c22e-01aa-3887-4151- b9002ce36d5d}</channelSource>
      <channelURI>rtsp://120.71.23.52/streaming/channels/11</channelURI>
      <channelState>active</channelState>
      <channelType>
        <channelInputType>stream</channelInputType>
        <contentType>audio</contentType>
        <codecType>G.726</codecType>
      </channelType>
    </channelStatus>
    <channelStatus>
      <channelID>3</channelID>
      <channelSource>{e70144c03-01aa-3887-4151-3e550991d377}</channelSource>
      <channelURI>rtsp://150.81.23.12/streaming/channels/1</channelURI>
      <channelState>active</channelState>
      <channelType>
        <channelInputType>stream</channelInputType>
        <contentType>video</contentType>
        <codecType>H.264-BP</codecType>
      </channelType>
    </channelStatus>
    <channelStatus>
      <channelID>4</channelID>
      <channelSource>{e70144c03-01aa-3887-4151-3e550991d377}</channelSource>
      <channelURI>rtsp://150.81.23.12/streaming/channels/2</channelURI>
      <channelState>active</channelState>
      <channelType>
        <channelInputType>stream</channelInputType>
        <contentType>audio</contentType>
        <codecType>G.726</codecType>
      </channelType>
    </channelStatus>
    <channelStatus>
      <channelID>5</channelID>
      <channelSource>{ff0144c03-01aa-3887-4151-33950991d38c}</channelSource>
      <channelURI>rtsp://110.30.15.77/streaming/channels/22</channelURI>
      <channelState>active</channelState>
      <channelType>
        <channelInputType>stream</channelInputType>
        <contentType>video</contentType>
        <codecType>MPEG4-ASP</codecType>
      </channelType>
    </channelStatus>
    <channelStatus>
      <channelID>6</channelID>
      <channelSource>{108144c03-01aa-3887-4151-33950991f81f}</channelSource>
      <channelURI>rtsp://110.30.15.77/streaming/channels/23</channelURI>
      <channelState>active</channelState>
      <channelType>
        <channelInputType>stream</channelInputType>
        <contentType>video</contentType>
        <codecType>H.264-MP</codecType>
      </channelType>
    </channelStatus>
  </channelStatusList>
</CMChannelStatus>

The above example is for a hypothetical NVR that has 6 input SINGLE-MEDIA streams (3 video +
3 audio) coming in from 3 different sources. All tracks are active.
```

```
<?xml version="1.0" encoding="UTF-8"?>
<CMChannelStatus version="1.0" xmlns="urn:psialliance-org">
  <channelStatusList>
    <channelStatus>
      <channelID>1</channelID>
      <channelSource>{a534c22e-01aa-3887-4151- b9002ce36d72}</channelSource>
      <channelURI>rtsp://localhost/PSIA/Streaming/channels/1</channelURI>
      <channelState>active</channelState>
      <channelType>
        <channelInputType>port</channelInputType>
        <contentType>video</contentType>
        <codecType>H.264-BP</codecType>
      </channelType>
    </channelStatus>
    <channelStatus>
      <channelID>2</channelID>
      <channelSource>{a534c22e-01aa-3887-4151- b9002ce36d72}</channelSource>
      <channelURI>rtsp://localhost/PSIA/Streaming/channels/2</channelURI>
      <channelState>errored</channelState>
      <channelType>
        <channelInputType>port</channelInputType>
        <contentType>video</contentType>
        <codecType>H.264-BP</codecType>
      </channelType>
      <channelStatusString>"No Video Signal"</channelStatusString>
    </channelStatus>
    <channelStatus>
      <channelID>3</channelID>
      <channelSource>{a534c22e-01aa-3887-4151-b9002ce36d72}</channelSource>
      <channelURI>rtsp://localhost/PSIA/Sstreaming/channels/3</channelURI>
      <channelState>active</channelState>
      <channelType>
        <channelInputType>port</channelInputType>
        <contentType>video</contentType>
        <codecType>H.264-BP</codecType>
      </channelType>
    </channelStatus>
    <channelStatus>
      <channelID>4</channelID>
      <channelSource>{a534c22e-01aa-3887-4151- b9002ce36d72}</channelSource>
      <channelURI>rtsp://localhost/PSIA/Streaming/channels/4</channelURI>
      <channelState>inactive</channelState>
      <channelType>
        <channelInputType>port</channelInputType>
        <contentType>video</contentType>
        <codecType>H.264-BP</codecType>
      </channelType>
      <channelStatusString>"Disabled"</channelStatusString>
    </channelStatus>
  </channelStatusList>
</CMChannelStatus>

The above scenario is for a 4-port DVR. All of the channels are hardware based, not IP- based.
The source ID for all of the channels/ports is the UUID/GUID of the RaCM DVR device (i.e. a
self reference). Channel #1 (2nd channel) is in an "errored" state because it is not receiving
a video signal. Channel #3 is inactive because an administrative action disabled the port (i.e.
forced it to an "inactive" state).
```

The "CMChannelStatus" schema defines all the related channel status information. Channels report status via a channel status list. Each enty in the list utilizes the following status attribute definitions.

10.3.1 /PSIA/ContentMgmt/status/channels Status Attributes

Attribute	Element Name	Requirement	Description

Channel ID	"channelID"	Mandatory	Integer ID (index) of the input channel
Channel Source	"channelSource"	Mandatory	128-bit UUID/GUID of the source device. For NVRs this is the UUID/GUID of the remote source device. For DVRs, this is the UUID/GUID of the DVR device itself.
Channel URI	"channelURI"	Mandatory	URI used to access this 'channel's media.
Channel State	"channelState"	Mandatory	The state of the input channel: State values are: •**"active":** normal operative state. •**"idle":** channel is OK and active, but not data is being received in the present state. •**"inactive":** the channel is inoperative due to recording mode or administrative intervention (i..e. a non-error condition). **"errored":** the channel has encountered and error and is not receiving data.
Channel Type	"channelType"	Mandatory	All input channels are one of 2 possible types: •**"port":** a physical input port is the source of the stream (e.g. composite video). •**"stream":** the input stream is logical; i.e. it is an IP-based data stream (RTP, HTTP, etc.)
Channel Status String	"channelStatusString"	Mandatory / Optional; *Mandatory for Channel State "errored"*	Optional human-readable text string describing the current channel state. This string is MANDATORY for the channel state "errored" (see above).

The above status information is reported for the channels on a RaCM device in a list ("channelStatusList"). When a status requester 'GETs' channel status, it can either A) get all of the channels on a RaCM device (URI:"/PSIA/ContentMgmt/status/channels"), or B) it can get status for a particular channel. In the latter case, the requester must specify the target channel by referencing the channel ID in the URI (e.g. "/PSIA/ContentMgmt/status/channels/<id>") where "<id>" is the channel ID (see above) for the target channel. Information regarding the specifics of the status information is contained in the following schema definition section.

10.3.2 /PSIA/ContentMgmt/status/channels XML Schema Definition

XSD: To get the latest version of all RaCM XSDs, including this one (cmChannelStatus.xsd) please download from the PSIA documents website:
http://www.psialliance.org/documents_download.html

10.4 /PSIA/ContentMgmt/status/tracks

All recorded information for a given RaCM device is recorded onto one, or more, 'tracks'. As described earlier in this specification, tracks are virtual containers. They may, or may not, correspond to a file, or set of files, etc. Basically, a track is a handle to specific type of multimedia information (e.g. video or audio) from a specific source. The PSIA places no other constraints on the implementation of tracks. For status information, tracks, just like channels, render specific status attributes for each track, in a list. For generic status queries (i.e. all tracks), the list contains information about *all* tracks on a RaCM device. The URI for gathering status on all tracks is "/PSIA/ContentMgmt/status/tracks." A specific track can be queried for its status information using the URI "/PSIA/ContentMgmt/status/tracks/<id>" where "<id>" is the ID number of the target track. In order to keep status gathering simple, no schemas are employed to query status; only the REST URIs are employed. If an entity needs to gather information on subsets of all the tracks, then individual queries need to be made per track, track information needs to be gathered by source (see above Section 11.2).

URI	/PSIA/ContentMgmt/status/tracks *or* /PSIA/ContentMgmt/status/tracks/<id>		**Type**	Resource
Requirement Level	- All -			
Function	Mandatory description of the REST method parameters and formats available to functionally manipulate the 'search' resource/object.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET	None	None	<CMChannelStatus>	
PUT	N/A	N/A	<ResponseStatus w/error code>	
POST	N/A	N/A	<ResponseStatus w/error code>	
DELETE	N/A	N/A	<ResponseStatus w/error code>	
Notes				

Example(s)

```
<?xml version="1.0" encoding="UTF-8"?>
<CMTrackStatus version="1.0" xmlns="urn:psialliance-org">
  <trackStatusList>
    <trackStatus>
      <trackID>19</trackID>
      <trackSize>320.6</trackSize>
      <trackSizeUnits>MBs</trackSizeUnits>
      <trackState>active</trackState>
      <trackType>
        <contentType>video</contentType>
        <codecType>MPEG4-SP</codecType>
      </trackType>
      <trackMode>CMR</trackMode>
      <trackSource>
        <trackSourceID>{9034c22e-01aa-3887-4151- b9002ce36d5d}</trackSourceID>
      </trackSource>
      <trackChannelID>11</trackChannelID>
    </trackStatus>
  </trackStatusList>
</CMTrackStatus>

The above example is for a status query for a specific track; in this case track #19. This
track contains MPEG-4 video data. This example contains the minimum information required
for a track.
```

```
<?xml version="1.0" encoding="UTF-8"?>
<CMTrackStatus version=”1.0” xmlns="urn:psialliance-org">
  <trackStatusList>
    <trackStatus>
      <trackID>12</trackID>
      <trackSize>416</trackSize>
      <trackSizeUnits>MBs</trackSizeUnits>
      <trackState>active</trackState>
      <trackType>
        <contentType>video</contentType>
        <codecType>H.264-BP</codecType>
      </trackType>
      <trackMode>CMR</trackMode>
      <trackSource>
        <trackSourceGUID>{9034c22e-01aa-3887-4151- b9002ce36d5d}</trackSourceGUID>
      </trackSource>
      <trackChannelID>8</trackChannelID>
      <trackTimeRange>
        <trackOldestTime>2009-05-12T09:17:06</trackOldestTime>
        <trackNewestTime>2009-06-14T13:22:43</trackNewestTime>
      </trackTimeRange>
      <trackObjects>2608012</trackObjects>
      <trackVersion>1.1</trackVersion>
      <trackCreationDate>2009-05-12T09:17:06</trackCreationDate>
      <trackMountVolumeID>1</trackMountVolumeID>
      <trackUtilization>79.6</trackUtilization>
    </trackStatus>
  </trackStatusList>
</CMTrackStatus>

The above example is for a status query for a specific track; in this case track #12. This track
contains H.264 video data. This example contains the maximum information that can be reported
for a track.
```

10.4.1 /PSIA/ContentMgmt/status/tracks Status Attributes

Status information for each track consists of both mandatory status attributes, and optional items. The optional items provide greater insight into the status of the content of a track and its basic storage attributes. The track status attributes are listed in the following table

Attribute	Element	Requirement	Description
Track ID	“trackID”	Mandatory	Track ID number of the specific track being reported
Track Size	“trackSize” “trackSizeUnit”	Mandatory	The size of the track. This is reported in 2 parts: •A number in “trackSize” indicating the current size of the referenced track in…. •The unit used to report the size (“trackSizeUnit”). Units are MBs. GBs, TBs, PBs, MiBs, GiBs, and XBs.
Track State	“trackState”	Mandatory	The operating state of the track. State choices are: •“**active**”: track is OK and actively recording.: •“**idle**”: track is OK but not receiving data.: •“**inactive**”: track is dormant. •“**locked**”: track has been put into “read- only” mode. •“**errored**”: track has encountered an error and is not recording.

Track Type	"trackType"	Mandatory	The type of the reported track: •"contentType" indicates of the track is video, audio, metadata, text or mixed (polymorphic). •"codecType" indicates the primary codec format / protocol of the track. (see Appendix A)
Track Mode	"trackMode"	Mandatory	The recording mode of the track: •"**CMR**": Continuous Mode Recording(time-lapsed, or loop-based recording). •"**EDR**": Event Driver Recording. •"**CMR-Scheduled**": track is in CMR mode based on a schedule (track is multi-modal). •"**EDR-Scheduled**": track is in EDR mode based on a schedule (track is multi-modal). •"**Manual Recording**": track is recording based on user driven activity.
Track Source	"trackSource"	Mandatory	The track source varies based on whether the input is an IP-based device (NVR scenario), or a hardware based input port (DVR scenario). The following are used to identify track sources: •IP-based input = 128-bit UUID/GUID of the source device. •Hardware based input = Channel ID of the local port-based stream RaCM device.
Track's Channel ID	"trackChannelID"	Mandatory / Optional	For all scenarios where the channels are still operative/configured on a RaCM device, the channel ID must be provided. Otherwise, this data is not required.
Track Status String	"trackStatusString"	Mandatory / Optional	For the "errored" track state (see above), this is a required string that provides a human readable status description of the error encountered. For all other states, this string is optional.
Track Time Range	"trackTimeRange"	Optional	Oldest/newest timestamps of the data contained within a track.
Track's Number of Objects	"trackObjects"	Optional	Number of objects (i.e. frames, GOPs/GOVs, events, etc.) contained in the track.
Track Version	"trackVersion"	Optional	Version of the track's format/implementation.
Track's Creation Date	"trackCreation"	Optional	Creation 'dateTime' of the track.
Track's Mount	"trackMountVolumeID"	Optional	One, or more, Mount Volume ID numbers referencing the mount

Volume ID			volumes that the track occupies.
Track Utilization	"trackUtilization"	Optional	Float indicating the percentage of utilization of the allocated track space (i.e. "24.7" = %24.7).

10.4.2 /PSIA/ContentMgmt/status/tracks XSD

XSD: To get the latest version of all RaCM XSDs, including this one (cmTrackStatus.xsd) please download from the PSIA documents website: http://www.psialliance.org/documents_download.html

11 Streaming and Playback

RaCM devicea must offer PSIA compliant streaming services for the playback of recorded media information. For RaCM devices that also offer the ability to serve live streams to clients, and other multimedia consumers, as a proxy server, these streams must also be provided in a PSIA compliant manner. The requirements for multimedia streaming are specified in the "PSIA IP Media Device API Specification.", Sections 5.1 ("Media Streaming, Streaming with RTSP and SDP") and 7.12 ("/Streaming") and its subsections. These sections of the IP Media Device specification cover just live streaming. Additionally, a RaCM device has the ability to stream recorded data, and the ability to configure incoming streams (i.e. "channels") indirectly via track configuration using the '/ContentMgmt/record' service. Due to these unique functional differences, the following exceptions, qualifications, and additions to the IP Media Device specification

11.1 Streaming URIs

11.1.1 Live Streams

The IP Media Device specification, *Section 5.1.2*, specifies the following URI structure for a client/consumer to initiate and RTSP Streaming session:

```
rtsp://<address>:<port>/Streaming/Channels/<id>(?parm1=value1&parm2-=value2…)
```

This is compatible with channel definitions for RaCM devices as well. All input streams, port or network-based, are mapped to RaCM device 'channels' identified by channel IDs. Each track, in its configuration, and status, information also contains the corresponding channel ID for its input stream. So, channel IDs can be obtained by reading either the track configuration information (see "*/PSIA/ContentMgmt/record/tracks (Recording Session Configuration)*") or the track status information (see "*/PSIA/ContentMgmt/status/tracks*").

Additional support for the description of multiple channels, not necessarily from the same source is provided by adding channel IDs to an RTSP URI. In these cases, the parameter tag "channel=" is used for each additional channel. The following example depicts this case.

rtsp://10.1.2.55/Streaming/channels/7?channel=8&channel=11&channel=20

In the above example, a client desires to retrieve an RTSP description of channels 7, 8, 11, and 20. Please note that this RTSP URI construct will only work for the RTSP DESCRIBE message/method.

11.1.2 Archive Streams

RaCM devices record multimedia information onto 'tracks' which are accessible for RTSP

Streaming via 2 identification methods: A) track IDs, and B) source device UUID/GUIDs. The first requirement is obvious; a consumer desires a playback stream from a specific track. The URI structure for Streaming this media information is:

```
rtsp://<address>:<port>/Streaming/tracks/<id>(?parm1=value1&parm2-=value2…)
```

The above 2 URI constructs are direct derivatives of the PSIA REST resource hierarchy for media information and match the RaCM notations described herein. Additionally, a client/consumer that establishes an RTSP session to a RACM device and issues a "DESCRIBE" (see RFC 2326), for a channel or track, only receives an SDP description of the media information for that specific, channel or track, not for an entire "presentation" (see RFC 2326). For the use of time as a parameter in the management of streaming sessions, please see "Time-related Streaming" below

Also, as is noted for live streaming, above, requesters may gather RTSP descriptions for multiple tracks by appending additional track IDs to the end of an RTSP DESCRIBE URI as query parameters. The parameter value tag for each additional track ID is "track=". The URI below is an example of a multi-track RTSP DESCRIBE URI.

rtsp://10.1.2.55/Streaming/tracks/19?track=20&track=27&track=28

In the above URI example, a requester is attempting to retrieve an RTSP/SDP description of multiple tracks via a single message. In this case, the requester wants an SDP description for tracks 19, 20, 27, and 28. This prevents a requester/client from having to issue multiple 'DESCRIBEs' to get multi-track media attributes. *There is one restriction: when imposing a time range on a DESCRIBE, all of the tracks, in a multi-track URI, must share the same time range.* This is described in more detail in ***Section 12.1.4*** below.

11.1.3 Source-based Streaming

For clients/consumers that desire to receive RTSP-based streamed media information for all channels, or tracks, from a specific source device, the following URI structure

```
rtsp://<address>:<port>/Streaming/sources/<GUID>/<type>
```

The field "<GUID>" in the above URI template references the 128-bit UUID/GUID of the target source device. The "<type>" field is required to differentiate between 'live' and 'archived' media content via the following tags:

- "channels": indicates 'live' media data, and
- "tracks": indicates archived content.

The following source-based URI examples are provided:

rtsp://10.1.2.55/Streaming/sources/{91c0822d4-033f-5a01-00a7-498330c09f5b}/channels

The above example references a client's desire to receive an RTSP DESCRIBE that lists *all* of the live media streams that a source device provides via a RaCM device. This is useful for source devices that are originating video and audio, and potentially, metadata/events. The DESCRIBE's SDP response would list ***all*** of the available channels for that specific input device. This enables client/consumers to SETUP multiple channels in a single RTSP session (see RFC 2326; setup via multiple RTSP sessions is also supported, though less efficient).

rtsp://10.1.2.39:554/Streaming/sources/{91c0822d4-033f-5a01-00a7-498330c09f5b}/tracks

The above example references a URI that causes an RTSP DESCRIBE operation to return an SDP

description of all the tracks on a RaCM device that may be streamed to a client/consumer associated with a specific source device.

All source-based Streams MUST utilize the above URI structure complete with the "type" tag for RTSP session initiation and setup. A source-based URI that does not list the "type" tag is an error and will result in an RTSP failure. For client/consumers that desire to see all of the channels *and* tracks associated with a specific source device, the following section

Streaming Operations.

11.1.4 Time-related Streaming

For archive streams, whether track or source-based in notation, there usually is a time component that indicates the desired time range with respect to a track or source. In the cases where a consumer needs to specify the specific time range associated with a streaming session, URI request line parameters are employed for defining the specific time range. The following parameter tags are used:

- "starttime": This parameter tag indicates that it will be followed by and ISO 8601 timestamp indicating the start time of the media stream the consumer is targeting for description, setup, or playing.
- "endtime": This parameter tag indicates that it will be followed by an ISO 8601 time stamp indicating the ending time the consumer desires to be the termination point for a media stream. This parameter field is optional. If it is not present, the stream is to proceed from the start time until the session is manually terminated, or paused, by the consumer.

The format of the time stamps is ISO 8601 as specified in Section 3.7 of RFC 2326, "Absolute Time.' This format is almost identical to XML 'dateTime' except there are no dashes to separate the fields. Basically, the format is: YYYYMMDD"T"HHmmSS.fraction"Z" where Y=year, M=month, D=day, "T" is the time separator, H=hour, m=minutes, S=seconds, and "Z" is the optional field indicating Zulu (GMT) time. A time stamp example is: "20090526T13:42.58Z" which represents May 26th, 2009 at 1:42.58 PM GMT.

Given the above formatting information, a client using RTSP session management would append the time stamps to the end of its URI (either track or source-based) as a way of designating the target timeframe associated with a streaming session. A track-based example follows:

rtsp://10.1.2.39:554/Streaming/tracks/18?starttime=20090731T092241.06Z&endtime=20090731T093000Z

In the above example, a requester desires to describe/setup/play a media stream that spans the time range on July 31st, 2009 from 9:22.06AM GMT to 9:30AM GMT.

As noted in Section 12.1.2, when multiple tracks are identified in a single RTSP URI, they can only share one time range. This restriction is instated to prevent run-on URIs that exceed standard buffer length allocations.

The appending of timestamps as parameters to a URI request line is pertinent for all URIs, even those returned by the 'SearchResponse' since a requester may not desire an entire time segment. Please reference RFC 2326 for more details on URI and parameter formats.

11.2 Streaming Configuration and Status

All RaCM devices must support the ability to play live input video/audio streams to clients as a live

stream server as outlined in the IP Media Device specification. However, the ability to modify codec settings, that affect the corresponding streams, varies with respect to DVR versus NVRs. These areas are covered in Sections 7 and ?? of this document. Streaming parameters that are independent of the source codec setting are described in this section of the document.

11.2.1 Streaming Status

The IP Media Device specification in ***IPMD Sections 7.12.1and 7.12.4*** specifies REST interfaces for getting status information on live channels and on sessions. Since RaCM devices already provide the equivalent information in the ContentMgmt/status service functions, these sections are *optional* and *not required* for implementation by RaCM devices. It is recommended that RaCM devices implement these interfaces for interoperability reasons.

11.2.2 QoS Parameter for playback (new v1.1)

See the IP Media Device specification in Sections 7.4.22 – 7.4.25 for static QoS settings per network interface and per media class/type (i.e. /PSIA/System/Network/interfaces/<ID>/qos/cos and /PSIA/System/Network/interfaces/<ID>/qos/dscp). The values set here for <TrafficType>'s of "video" and "audio" will apply.

In addition to IPMD, RaCM allows the following overrides, on a per-request basis, via the Query String parameters shown here:

```
rtsp://<uri_base>?dscp=<code>
rtsp://<uri_base>?cos=<priority>
rtsp://<uri_base>?dscp=<code>&cos=<priority>
```

Where <uri_base> refers to the URI's described in Section 14.1.

(Note also that, in a XML document, the ampersand must be encoded as "&" -- See "*XML Reserved Characters*")

Differentiated Services

The "**dscp="** Query String parameter is used to request use of a DiffServ codepoint value (6 bit value, given in decimal range of 0-63) for the data stream.

RFC2597 (Assured Forwarding) defines the following codepoints:

	Class 1	Class 2	Class 3	Class 4
Low Precedence	Binary 001010 "AF11" / dscp=**10**	Binary 010010 "AF21" / dscp=**18**	Binary 011010 "AF31" / dscp=**26**	Binary 100010 "AF41" / dscp=**34**
Medium Precedence	Binary 001100 "AF12" / dscp=**12**	Binary 010100 "AF22" / dscp=**20**	Binary 011100 "AF32" / dscp=**28**	Binary 100100 "AF42" / dscp=**36**
High Precedence	Binary 001110 "AF13" / dscp=**14**	Binary 010110 "AF23" / dscp=**22**	Binary 011110 "AF33" /dscp=**30**	Binary 100110 "AF43" / dscp=**38**

Note that for translation to simple priority levels, the priority number generally increases to the right/bottom of the table.

RFC2598 (Expedited Forwarding) defines codepoint: Binary 101110, "EF" / dscp=**46**.

RFC2474 (Differentiated Services) also defines "Class Selector" code points, Binary xxx000, using the upper precedence bits in same was as legacy TOS Precedence values.

IEEE 802.1p / 802.1q – Class of Service

The "**cos="** Query String parameter is used to request an IEEE 802.1p priority level (3 bit value, if supported by the local network) for the data stream.

Priority	Traffic Type
0	Best Effort
1	Background
2	Spare
3	Excellent Effort
4	Controlled Load
5	Video
6	Voice
7	Network Control

Multiple QoS Parameters

It is possible to specify both QoS override parameters (e.g. both Class of Service and a DiffServ codepoint), which would be interpreted as enabling both, wherever applicable; however, the actual behavior will be dependent on the network capability and administrative guidelines.

11.3 Streaming Operations

This section provides an operational overview of some of the key design aspects regarding how client, management applications, and consumers can 'find' and access the live media streams they desire to receive from a compliant PSIA RaCM device that provide streaming proxy/server functionality. Streams are usually accessed via parameters associated with one of the following methods:

- Media type
- Source
- Track association
- Search

A description of each of these methods, and how they can be employed using the standard RaCM interfaces, follows.

Media type: The Media type method identifies operations where an entity wants to see all, or some, channels of a particular media/content type (e.g. 'video', 'audio', 'metadata', etc.). An entity can detect the appropriate channel types via 2 RaCM interfaces: A) "*/PSIA/ContentMgmt/record/tracks (Recording Session Configuration)*", or B) "*/PSIA/ContentMgmt/status/channels*". The first interface supplies track configuration information which contains the media/content type parameters and the associated channel ID for each track (also the Source's UUID/GUID). The second interface provides the content/media type information for each channel along with the codec type and the Source UUID/GUID of the source device. Using this information an entity can the access the desired channel using the supplied channel IDs.

Source: The Source method refers to entities that desire to get the attributes of all the streams associated with a particular source device/endpoint. These types of operations are usually related to a particular scene or field-of-view (FoV). Information regarding the streams assocated with one, or more, sources can be obtained 2 primary ways: A) via Source status information ("*/PSIA/ContentMgmt/status/sources*"), or B) via track configuration information ("*/PSIA/ContentMgmt/record/tracks (Recording Session Configuration)*"). Using the source status interface, an entity can get the status information on all sources. For each source, the status information containts the UUID/GUID of the associated source plus a complete channel list and track list. Each channel and track list has their IDs, content/media types and codec types. Using track configuration information is a less direct way to get the source IDs, but the information is present. Once an entity has a source's UUID/GUID, it can issue an RTSP streaming session using the URI structure outlined in "*Streaming URIs*", above, and receive an SDP description of all the source's media streams.

Track Association: All track information in both the track configuration attributes ("*/PSIA/ContentMgmt/record/tracks (Recording Session Configuration)*") and in the status information provided for tracks, and sources, contains the corresponding channel IDs for the input streams linked to the respective track.

Search: Presently, the Content Management Search services are only for recorded content. However, the Search responses always contain the track IDs for each of the 'matches' to a search

(see Section "/PSIA/ContentMgmt/search (Search Service and Resources)"). Using these track IDs, an entity deciding to access the corresponding live media can get the channel ID for a particular track using either: A), the track's configuration information ("*/PSIA/ContentMgmt/record/tracks (Recording Session Configuration)*"), or B) track status information ("*/PSIA/ContentMgmt/status/tracks*"). Once the channel ID is obtained, the appropriate RTSP Streaming URI can be constructed (see "*Streaming URIs*").

11.4 Playback

11.4.1 Playback Requirements

This section describes the requirements for the replay protocol. **Note: These requirements apply to the protocol itself, not any implementation of it.**

1. **The replay mechanism must support the following client operations:**
   - Start playback from a specific source, from a specific time.
   - Play forwards at any speed (both faster and slower than real time)
   - Play backwards at any speed (both faster and slower than real time)
   - Play I-frames only, or a subset of the I-frames, to reduce bandwidth and client processing requirements and allow very fast playback in either direction.
   - Pause/resume playback
   - Step forwards by a single frame (when in paused mode)
   - Step backwards by a single frame (when in paused mode)

   Note that this does not imply that the replay protocol must include methods corresponding to each of these operations, only that the protocol must facilitate the implementation of these operations by the client application.
2. **The replay mechanism shall allow lossless playback.** It must be possible to play back stored footage with no loss of information, regardless of network conditions, such that clients are always able to display every frame that was successfully recorded. (This does not preclude a "lossy" mode in addition to this).
3. **Every frame shall have an stable, absolute timestamp.** Each frame must carry a timestamp indicating the absolute time at which it was captured by the NVT. These timestamps must be communicated to the client during replay (either directly or indirectly). The replay mechanism shall support timestamps with at least millisecond accuracy.
4. **Synchronization of audio/video/metadata.** The replay mechanism must make it possible for a client to accurately synchronize audio and video streams, and to associate timestamped metadata items with individual frames.
5. **Synchronization of multiple streams.** The replay mechanism must make it possible for a client to accurately synchronize playback of media streams from different sources, and to allow users the same playback control operations with synchronized streams (as detailed in requirement 1 as are available with single streams.

6. **The replay mechanism shall support standard players.** It shall be possible for standard players such as QuickTime and VLC to replay stored footage, although not necessarily with the same functionality that might be available to an ONVIF-aware client.

11.4.2 Usage patterns

The usage patterns for video replay within a CCTV context are significantly different from those of most internet video applications, where users typically play a clip or presentation from start to finish, perhaps pausing occasionally. When reviewing CCTV footage on the other hand, users typically play forwards for only short periods of time, but frequently pause, single step, jump to different points in time, fast forward and rewind.

11.4.3 Use of RTSP

The replay protocol is based on RTSP [RFC 2326]. However because RTSP does not directly support all of the requirements, there are defined several extensions to the protocol; these are detailed below.

In addition, we make the following stipulations on the usage of RTSP:

1. Interleaved mode ([RFC 2326] section 10.12) MUST be supported by the RACM DEVICE.
2. Clients SHOULD use either interleaved mode or RTP/TCP (if supported) in order to satisfy requirement 2 above.
3. The RACM DEVICE MAY elect not to send RTCP packets during replay. In typical usage, at least with an ONVIF-aware client, RTCP packets are not required, because usually a reliable transport will be used, and because absolute time information is sent within the stream, making the timing information in RTCP sender reports redundant.

11.4.4 RTP header extension

• In order to satisfy requirement 3, we need to be able to associate an absolute timestamp with each packet, or each group of packets with the same RTP timestamp (e.g. a video frame). This is achieved using an RTP header extension containing an NTP timestamp and some additional information also useful for replay.

• The replay mechanism uses the extension ID 0xABAC (tbd) for the replay extension.

• Below shows the general form of an RTP packet containing this extension:

<table>
<tr><td colspan="2">V=2</td><td>P</td><td>X=1</td><td colspan="2">CC</td><td>M</td><td colspan="2">PT</td><td colspan="2">sequence number</td></tr>
<tr><td colspan="12">Timestamp</td></tr>
<tr><td colspan="12">synchronization source (SSRC) identifier</td></tr>
<tr><td colspan="9">0xABAC</td><td colspan="3">length=3</td></tr>
<tr><td colspan="12">NTP timestamp…</td></tr>
<tr><td colspan="12">…NTP timestamp</td></tr>
<tr><td>C</td><td>E</td><td>D</td><td colspan="3">mbz</td><td colspan="7">Padding/Frame Type Indicator (FTI, optional)</td></tr>
<tr><td colspan="12">Payload…</td></tr>
</table>

- The fields of this extension are as follows:

- NTP timestamp. An NTP [RFC 1305] timestamp indicating the absolute UTC time associated with the access unit.
- C: 1 bit. Indicates that this access unit is a “clean point”, e.g. the start of an intra-coded frame in the case of video streams.

- E: 1 bit. Indicates the end of a contiguous section of recording. The last access unit in each track before a recording gap, or at the end of available footage, SHALL have this bit set.
- D: 1 bit. Indicates that this access unit follows a discontinuity in transmission. It is primarily used during reverse replay; the first packet of each GOP has the D bit set since it does not chronologically follow the previous packet in the data stream.
- Frame Type Indicator (FTI): This *optional* field provides additional information about the type of video unit (frame, slice) that is present in the MTU payload. Since this field is optional, if an FTI is present, the first 2 octets of this field, which is treated as a 3-octet string, are set to 0x46, 0x49 ("FI" ASCII) to indicate the presence of the frame type. The last octet, the on adjacent to the Payload, has the following format:
    - The high order bit is the Frame bit. If this bit is ON (mask |= 0x80), then the video unit is a frame. If this bit is OFF (mask =0x7F), the video unit is a sub- frame indicating for H.264 media streams that it is a slice. Where a slice equals a frame, the frame bi applies.
    - The low order 7 bits indicate the type of frame or slice. The values are:
        - **0x00:** Intra-frame (I-frame, I-VOP).
        - **0x01**: Predictive Inter-frame (P-frame, P-VOP);
        - **0x02:** Bidirectional Predictive Inter-frame (B-frame, B-VOP);
        - All other values are reserved.

  If the FTI is not supported by an implementation, then the first 2 octets of Padding MUST not be equal to 0x46, 0x49. Zero-based padding is recommended.

- The replay header extension MUST be present in the first packet of every access unit (e.g. video frame). It MAY NOT be present in subsequent packets of an access unit.

11.4.4.1 Compatibility with the JPEG header extension

The replay header extension may co-exist with the header extension used by the JPEG RTP profile; this is necessary to allow replay of JPEG streams that use this extension. The JPEG extension is simply concatentated with the replay extension; its presence is indicated by an extension length field with a value greater than 3, and by the extension start codes of 0xFFD8 or 0xFFFF at the start of the fourth word of the extension content.

The following illustrates a JPEG packet that uses both extensions:

<table>
<tr><td colspan="2">V=2</td><td>P</td><td>X=1</td><td colspan="2">CC</td><td>M</td><td colspan="2">PT</td><td colspan="3">sequence number</td></tr>
<tr><td colspan="11">Timestamp</td></tr>
<tr><td colspan="11">synchronization source (SSRC) identifier</td></tr>
<tr><td colspan="8">0xABAC</td><td colspan="3">length=N+3</td></tr>
<tr><td colspan="11">NTP timestamp…</td></tr>
<tr><td colspan="11">…NTP timestamp</td></tr>
<tr><td>C</td><td>E</td><td>D</td><td colspan="3">mbz</td><td colspan="7">Padding</td></tr>
<tr><td colspan="8">0xFFD8</td><td colspan="3">length=N</td></tr>
<tr><td colspan="11">extension payload: sequence of additional JPEG marker segments padded with 0xFF to the total extension length</td></tr>
<tr><td colspan="11">Payload…</td></tr>
</table>

11.4.4.2 NTP Timestamps

The NTP timestamps in the RTP extension header MUST increase monotonically over successive packets within a single RTP stream. They SHOULD correspond to wallclock time as measured at the RaCM device, adjusted if necessary to preserve monotonicity.

11.4.5 Initiating Playback

Playback is initiated by means of the RTSP PLAY method. For example:

```
PLAY rtsp://192.168.0.1/path/to/recording RTSP/1.0
CSeq: 123
Session: 12345678
Range: clock=20090615T114900.440Z-
Rate-Control: no
```

Reverse playback is indicated using the Scale header field with a negative value. For example to play in reverse without no data loss a value of –1.0 would be used.

```
PLAY rtsp://192.168.0.1/path/to/recording RTSP/1.0
CSeq: 123
Session: 12345678
Range: clock=20090615T114900.440Z-
Rate-Control: no
Scale: -1.0
```

11.4.5.1 Range header field

The Range field MUST be expressed using absolute times only; the other formats defined by [RFC 2326] SHALL NOT be used.

Either open or closed ranges may be used. In the case of a closed range, the range is increasing (end time later than start time) for forward playback and decreasing for reverse playback. The direction of the range MUST correspond to the value of the Scale header.

In all cases, the first point of the range indicates the starting point for replay.

Examples:

```
PLAY rtsp://192.168.0.1/path/to/recording RTSP/1.0
CSeq: 123
Session: 12345678
Range: clock=20090615T114900.440Z-20090615T115000
Rate-Control: no

PLAY rtsp://192.168.0.1/path/to/recording RTSP/1.0
CSeq: 123
Session: 12345678
Range: clock=20090615T115000.440Z-20090615T114900
Rate-Control: no
Scale: -1.0
```

11.4.5.2 Rate-Control header field

The Rate-Control field is an ONVIF extension and may be either “yes” or “no”. If this field is not present, “yes” is assumed, and the stream is delivered in real time using standard RTP timing mechanisms. If this field is “no”, the stream is delivered as fast as possible, using only the flow

control provided by the transport to limit the delivery rate.

11.4.5.3 Frames header field

The Frames header field may be used to request that only key frames be played, and optionally a minimum interval between successive key frames in the stream. The latter can be used to limit the number of frames received even in the presence of "I-frame storms" caused by many receivers requesting frequent I-frames.

The format of this header is

Frames: frametype[/interval]

where frametype may be "key" or "all", and interval is a time interval expressed in milliseconds. The interval argument SHALL NOT be present if the frametype is "all".

Example:

```
Frames: key/4000
```

The RACM DEVICE MUST support the Frames header field. This does not preclude the use of the Scale header field as an alternative means of limiting the data rate. The implementation of the Scale header field may vary between different RACM DEVICE implementations, as stated by [RFC 2326].

11.4.5.4 Clean points

The transmitted video stream MUST begin at a clean point. The rules for choosing the starting frame are as follows:

- If the requested start time is within a section of recorded footage, the stream starts with the first clean point at or before the requested start time. This is the case regardless of playback direction.
- If the requested start time is within a gap in recorded footage and playback is being initiated in the forwards direction, the stream starts with the first clean point in the section following the requested start time.
- If the requested start time is within a gap in recorded footage and playback is being initiated in the reverse direction, the stream starts with the last clean point in the section preceding the requested start time.

11.4.5.5 Time-gapped Playback

When playback occurs for tracks, or media segments, that contain time gaps within them (i.e. for tracks that employ 'Event Driven' recording), the following rules apply:

- RaCM devices shall play back the audio/video/metadata with the NTP timestamps reflecting the relative time (i.e. time between media packets in the RTP clock field), *and* indicating the absolute time in the RTP extension header (see above). As a note, clients can obtain the RaCM device's local time via the 'System/time/localTime' and '/System/time/timeZone' resources. The RaCM device shall playback the stream(s) as a contiguous, back-to-back media stream.
- Consumers of a media playback stream that notices a time jump/gap based on the RTP clock, and NTP timestamp fields, may RTSP 'PAUSE' the stream if they desire to manage the time gap sequences. For each PAUSE the consumer MUST either issue a PLAY, to resume media streaming,

or issue a TEARDOWN to destroy the media streaming session.

- As noted above, all time gaps MUST start with a clean point, i.e. a key/intra frame.

11.4.6 Reverse replay

Reverse replay is initiated using the Range header field as described above.

11.4.6.1 Packet transmission order

The order in which video packets are transmitted during reverse replay is based on GOPs, where a GOP consists of a clean point followed by a sequence of non-cleanpoint packets.

During reverse playback, GOPs are sent in reverse order, but packets within a GOP are sent in forward order. The first packet of each GOP MUST have the "discontinuity" bit set in its RTP extension header. The last packet of a GOP immediately following a gap (or the beginning of available footage) MUST have the E bit set in its RTP extension header.

When transmitting only key frames, or when the codec is not motion-based (e.g. JPEG), a GOP is considered to consist of a single frame, but may still be composed of multiple packets. In this case the packets within each frame are again sent in forward order, while the frames themselves are sent in reverse order.

Audio and metadata streams MAY be transmitted in an order mirroring that of the video stream, or alternatively in simple reverse order. However where a metadata document is split over multiple packets, those packets MUST be transmitted in forward order.

11.4.6.2 RTP sequence numbers

The RTP sequence numbers of packets transmitted during reverse playback SHALL increment monotonically *in the order of delivery,* not in the intended order of playback.

11.4.6.3 RTP timestamps

The RTP timestamps of packets transmitted during reverse playback SHALL be the same as they would be if those same packets were being transmitted in the forwards direction, following the rules laid out in [RFC 3550] and supporting documents. Unlike the sequence numbers, the RTP timestamps correspond to the intended playback order, not the delivery order. The RACM DEVICE MAY use the same RTP timestamps that were originally received when the stream was recorded.

11.4.7 Currently recording footage

If the client commences playback from the current real world time or shortly before it, it can end up playing footage in real time as it is being recorded. In this event the server simply continues to send stream data to the client as it receives it.

Note that the E bit is not set on access units currently being recorded even though each access unit

sent to the replay client will typically be the last one known to the RACM DEVICE. If recording stops however, the E bit is set on the last access unit of the recording.

11.4.8 End of footage

If playback reaches a point after which there is no further data in one or more of the streams being sent, it stops transmitting data but does not enter the "paused" state. If the RACM DEVICE resumes recording after this has happened, delivery will resume with the new data as it is received.

11.4.9 Go To Time

As stated in [RFC 2326] section 10.5, a PLAY command received when replay is already in progress will not take effect until the existing play operation has completed. This specification adds a new RTSP header, “Immediate”, which overrides this behaviour:

```
PLAY rtsp://192.168.0.1/path/to/recording RTSP/1.0
CSeq: 123
Session: 12345678
Range: clock=20090615T114900.440Z-
Rate-Control: no
Immediate: yes
```

If the RACM DEVICE receives a PLAY command with the Immediate header set to “yes”, it will immediately start playing from the new location, cancelling any existing PLAY command. The first packet sent from the new location SHALL have the D (discontinuity) bit set in its RTP extension header.

12 Polymorphic/Poly-temporal Track Support (new v1.1)

This section of the document specifies the PSIA RaCM design for a standards compliant design that accommodates and supports both polymorphic 'tracks' and poly-temporal (multi-time segmented) 'tracks'.

A 'track', as described in the prior section of this specification (in RaCM nomenclature), is a virtual container. It does not bear any direct association with how content (video, audio, metadata, text, etc.) is stored, managed, searched, played, and configured. Basically, this renders a track as essentially a 'handle' to a set of parameters and attributes that define the characteristics and content of a media repository. The RaCM specification does not infer, nor prescribe, any aspects of the implementation of content management except for the external interface definitions which are accessible via the network.

Presently, the RaCM specification prescribes a 1-to-1 match between a specific form of media content, i.e. MPEG-4 video, G.726 audio, text, etc., and a track. This 1-to-1 correlation of a track ID and attribute set to a specific form of content is spawned from the RTSP/SDP requirements to describe all 'tracks' (now using the SDP RFC 2327/4556 nomenclature) with unique attribute sets such that media decoders (primarily video and audio) can be setup to receive their particular datastreams on specific RTP connections. Additionally, each RTP connection, in itself, has its own indigenous set of parameters associated with each datastream (e.g. time, payload ID, socket connection, SSRC ID, etc.). It is noteworthy to cite that certain codec related attributes can change within a stream/track without 'breaking' that codec instance. For video codecs, as long as the codec profile and resolution do not change, the frame rate, bit rate, and quality levels (i.,e. quantization) can vary within a given stream/track. Some items that cannot change within a specific stream/codec instance are: entropy encoding method, codec profile, pixel/sample width, resolution, color format (e.g. from YUV 4:2:0 to YUV 4:2;2).

The above track/stream mapping to one specific set of track parameters across the recording and streaming functions does not handle 2 functional cases, well. These cases are:

- **Polymorphic content tracks/streams:** Tracks, or streams, that bear content that has more than one set of codec-related content attributes.
- **Poly-temporal temporal media sessions:** This term addresses RTSP sessions that need to stream several tracks, or datastreams, that do not share the same 'start' times. They may be chained, or cascaded, from a time perspective, potentially with overlap, but the streams do not start, and in many cases, stop, at the same time. Since RTSP's design assumes a common start time for all the advertised ('DESCRIBE'd) streams, it does not encompass the poly-temporal concept.

These 2 scenarios require additional definition in the RaCM specification such that a standards compliant, or nearly standards compliant, method for configuring, advertising and playing polymorphic and poly-temporal media can be accomplished. For the rest of this specification, unless the terms polymorphic or poly-temporal are specifically cited, the term 'poly-attribute' covers both cases.

12.1 Poly-attribute Tracks and Stream Management

This section of the document covers the following areas that are related to poly-attributed content management and poly-attributed stream management.

- Track attributes and parameter definitions;
- Accessing descriptions of poly-attribute characteristics via searching and session management mechanisms and data definitions;
- Correlating poly-attributed datastreams to network/RTP stream parameters, per session,

Each of the above areas is covered in the order described above.

12.1.1 Poly-attribute Tracks and Streams

For recording devices, tracks not only have inherent attributes associated with the recorded content, but also with respect to the live input that feeds, or fed, that content base. Therefore, the data characteristics associated with a track, "contentType", "codecType", "Description", etc., also advertise the attributes of any active live stream (i.e. "channel") that is feeding that track. Given this relationship, there are 3 ways for a potential data consumer to get the attribute information necessary to setup a data session:

- Read the track attributes for a set of one, or more, tracks
- Perform a search and utilize the 'SearchResponse' information to get the specific attributes for a data session;
- Directly issue an RTSP DESCRIBE for a specific source (via its UUID/GUID) or track.

Usually the last two methods are performed, as needed, after track attribute information has been read (though not always). All data sessions require codec and format attributes as part of the setup parameters. This is due to the fact that audio/video decoders must know certain codec attributes prior to decoding an incoming stream. The most detailed information comes in the SDP response to an RTSP DESCRIBE request. However, general attributes related to content and codec types are needed prior to RTSP session initiation in order that basic attributes can be verified or validated prior to committing the creation of an RTSP session. Track parameters assist in this process.

12.1.2 Track attributes

Each track, as described in Section 9, has its own schema definition for defining the parameters associated with its content base, and the input channel from which the content originated. One of the key elements in each track's parameter base is the "Description" element. This field is a string comprised of up to 7 AVP ("Attribute-Value Pair") tokens. An AVP token has the following format:

```
<attribute>"="<value>
```

where "<attribute>" and "<value>" represent UTF-8 strings. The definition of each field token follows:

Field Position	Field Name	Field Description
1	Track type ("trackType")	Required: Indicates type of track. Valid tags are: ¾ "standard" = norma.l, single content base track ¾ "polymorphic" = multi-content type track ¾ "polytemporal" = multi-time segmented track
2	Source Tag ("sourceTag")	Optional: Tag to indicate device type, if known. Usually a mfgr/make/model tag.
3	Content Type ("contentType")	Required: Indicates base (for polymorphic track the initial content type) content type for the track. Values are: ¾ "video" ¾ "audio" ¾ "metadata" ¾ "text" ¾ "other" (private definition)
4	Codec Type ("codecType")	Required for all non-text content types. See Appendix A for details
5	Resolution ("resolution")	Required for audio/video. Field indicating resolution of the data elements in a datastream. For video, this is the horizontal by vertical resolution in an 'Horizontal xVertical' format where ASCII 'x' separates the horizontal and vertical integer numbers. The assumed video format is progressive (i.e. frame based). For video streams that are interlaced (i.e. field based) and ASCII lower-case 'i' needs to be For audio, it is the bit-width of the samples. If a text protocol is enabled for double-byte characters, this field should be used to indicate "2B" character sets.
6	Frame rate ("frameRate")	Required for video. Frame rate of encoder output.
7	Bit rate ("bitRate")	Optional, but recommended. Bit rate of audio/video data stream.

Any field tokens not populated in the "Description" string are left absent. However, a RaCM device SHOULD provide as much useful information as possible for describing a poly-attributed track. The

following example "Description" element indicates a standard MPEG-4, VGA video track, from an AXIS IP camera. The video stream is rated at 3.2Mbps and 25 frames/second.

```
<Description>trackType=standard,sourceTag=AXIS210a,contentType=video,codecType=MPEG4-SP,
       resolution=640x480,frameRate=25fps,bitRate=3200kbps</Description>
```

The following example has no source tag for an H.264 VGA IP camera.

```
<Description>trackType=standard,contentType=video,codecType=H.264-BP
,resokution=640x480,frameRate=30fps,bitRate=3600kbps</Description>
```

The next example is for a polymorphic video track. The initial video is from an H.264 video encode source doing NTSC-based 2CIF (720x240) @ 15 fps during normal recording, but moving to 4CIF @ 30 fps during event/alarm recording.

```
<Description>trackType=polymorphic,sourceTag=ABCodec,contentType=video,codecType=H.264-
BP/G.711a,resolution=640x240/640x480,frameRate-15/30fps</Description>
```

Please note that in the above example, there is no specific bit rate cited, so it is therefore absent. Additionally, multiple attribute fields, such as 'codecType', the varying resolutions, and frame rates are listed as sets of choices where a 'slash' is utilized to separate the choices. So, all multiple attribute fields are to use the ASCII/UTF-8 'slash' character ("/") to separate the attributes in a list.

The following example is for a poly-temporal video track. The parameters of the "Description" string indicate that the video segments change frame rate with respect to the time segments (though the codec type never changes).

```
<Description>trackType=polytemporal,sourceTag=ABCodec,contentType=video,codecType=H.264-
BP,resolution=640x480,frameRate=15/30fps,bitRate=360/700kbps</Description>
```

The above track description indicates that the codec type does not change, but the frame rate does vary per each temporally related video segment within the track. The bit rate listed is an optional reference which helps indicated the approximate codec output rate at the corresponding frame rates. Since poly-temporal tracks can vary in structure, the guidelines for the track description field variables are only that the initial 'contentType' and 'codecType' characteristics are listed. However, as much useful information should be provided to help describe the contents and behavior of a track. No further requirements are levied on track description fields due to the fact that the SDP response to an RTSP DESCRIBE, or the 'SearchResponse' parameters, indicate the necessary internal details regarding the content, and temporal, structure of archived and live datastreams. Additionally, tracks and streams that are polymorphic and poly-temporal should list a compound track type of "polymorphic/polytemporal" such that interrogators can understand the nature of a track. An example follows.

```
<Description>trackType=polytemporal/polytemporal,sourceTag=IPCam9000,contentType=video,
codecType=H.264-BP/G.711,resolution=320x240/640x480,frameRate=15/30fps/Description>
```

This track description example is the type would typically be coupled to a track that records in Event-Driven mode. The description indicates this is a polymorphic track that is comprised of multiple temporal segments. In this case, this track is comprised of video and audio. The video

varies, per segment, from QVGA to VGA, and from 15fps to 30fps. In this example, no overall bit rates are supplied as information in this description.

12.2 Poly-attribute Stream Description

This section of the document deals with the rendering the attributes of datastreams (both live and archived) for streaming via RTSP and SDP. The constructs discussed in this section are taken from RFC 2326 (RTSP), RFCs 2327/4566 (SDP) and RFC 3388 (Grouping Media Lines in SDP). The internals of these specifications are not covered here in detail and it is expected that the reader will familiarize themselves with these specifications. Some basic concepts are covered here for description's sake.

RTSP is the control session protocol for managing streaming sessions. This protocol uses SDP as the format and syntactical standard for advertising stream attributes for media-based sessions. Whenever a potential consumer issues an RTSP DESCRIBE message/request, the server entity responds with all of the necessary attributes related to all of the media forms it may provide for streaming in SDP format. As time has passed, additional definitions have been added to SDP grammar for managing more complex media session types. Some of these additions are outlined in RFCs 3388 and 5583. These primarily relate to identifying multiple media types possible within a single session. This capability parallels the need to identify the multiple media types for poly-attribute datastreams. This section describes the process for enabling both polymorphic and poly-temporal datastreams. The four areas covered are:

- Session type identification;
- Media type descriptions and identification;
- Media level time management;
- Session and connection management for poly-attribute sessions.

12.2.1 SDP Session Section

Each SDP descriptor instance has two primary sections: 1) the session section (which comes first), and 2) the media sections. All PSIA stream sessions that have poly-attribute capabilities MUST include the session information attribute ("i=… ") line in the SDP session section. The construction of this session information string only requires that the first word indicate the session type. The 4 possible tags that are to be used for the first field are:

- "polymorphic", or…
- "polytemporal", or…
- "polymorphic/polytemporal", or…
- "standard";

Some example session information strings are:

…
"i=polymorphic media session description"

…
"i=polytemporal media session"

The first example string indicates to a client or consumer that the session parameters that will follow in an SDP instance describes the polymorphic media session attributes. The second string indicates that an SDP instance contains attributes for a poly-temporal session. For standard sessions, no session information line is required, yet if one is present it MUST start with the "i=standard…" tag.

12.2.2 SDP Media Section

Each media section within a poly-attributed SDP session descriptor MUST identify its codec specific attributes that require a specific codec instance. Each such media instance MUST identify the following identity information:

- Track ID ("trackID"): This line correlates a media instance with the track it is associated with. This is the PSIA RaCM track ID outlined for each track as described in Section 9. For poly-attributed tracks, this is the over-arching track that the media instance (datastream) is associated with. Please note that the term "track ID" has 2 meanings: A) it is the term used by SDP definitions (RFCs 2327/4566) to identify a handle to particular multimedia stream, and B) RaCM further correlates the prior SDP term to the specific 'track' (i.e. the identifier to a specific container of recorded media) on a RaCM device. Therefore, a(n) SDP 'track ID" = a RaCM 'track ID' within a(n) SDP instance such that the 2 terms are unified in meaning.
- Media ID ("mid"): As specified in RFC 3388, the media ID is uniquely associated with each media instance parameter set. This field, in PSIA usage, is an ASCII unsigned integer unique to the server for the described media stream/instance. Please note that, per RFC 3388, media IDs are members of a set of one, or more IDs, that comprise a 'Flow ID' ("FID"). The term 'Flow ID' is used to define the set of media IDs that comprise a "group" (e.g. "a=group:FID 6 7 8 9" is a Flow ID group comprised of 4 media IDs, six , seven, eight and nine, respectively).

The above IDs are used in a hierarchical manner. The track ID identifies the root poly-attributed track correlated to each media instance. The media id, however, uniquely identifies the specific media instance, and its characteristics, as a parameter set. This enables correlation to the original RaCM track instance while enabling the ability to setup each media instance as its own stream (i.e.RTP port number and payload ID). In summary, for recorded content, the SDP "track ID" is the RaCM track ID, and the SDP 'media ID' identifies a specific set of stream attributes. The following example depicts 4 unique media instances:

```
…
a=group:FID 7001 7002 7003 7004
m=video 5002 RTP/AVP 97
a=mid:7001
a=trackID:7
a=control:rtsp://PSIA/Streaming/tracks/7/mid=7001
a=rtpmap:97 H264/90000
a=fmtp:97 packetization-mode=1;profile-level-id=4D400C;sprop-
parameter-sets=J01ADKkYUI/LgDUGAQa2wrXvfAQ=,KN4JF6A=a=
b=AS:110
```

```
a=framerate:15
a=framesize:97 320-240
a=cliprect:0,0,240,320
m=video 5004 RTP/AVP 98
a=mid:7002
a=trackID:7
a=control:rtsp://PSIA/Streaming/tracks/7/mid=7002
a=rtpmap:98 H264/90000
a=fmtp:98 packetization-mode=1;profile-level-id=4D400C;sprop-
parameter-sets=J01ADKkYUI/LgDUGAQa2wrXvfAQ=,KN4JF6A=a=
b=AS:1200
a=framerate:30
a=framesize:98 640-480
a=cliprect:0,0,480,640
m=video 5005 RTP/AVP 98
a=mid:7003
a=trackID:7
a=control:rtsp://PSIA/Streaming/tracks/7/mid=7003
a=rtpmap:99 jpeg/90000
a=framerate:5
a=framesize:99 960-720
a=cliprect:0,0,720,960
m=audio 5003 RTP/AVP 8
b=AS:64
a=mid:7004 a=trackID:7
a=control:rtsp://PSIA/Streaming/tracks/7/mid=7004
...
```

In the above example there are 4 media instances within a 'group' ("a=group:FID") all of which are associated with track #7: three video instances and one audio instance that share the same 'track'. Each stream is uniquely identified by its own media ID {"a=mid:…") which is associated with the specified group. The group lists its flow ID ("FID") set which indicates which streams are interrelated media streams. This Flow ID set lists the media IDs that comprise the 'group'. In this case the flow ID set is comprised of media IDs 7001, 7002, 7003, and 7004. Media instance #7001 represents a QVGA, 110Kbps, 15fps H.264 video datastream mapped to socket port number 5002, and RTP payload ID 97. Media instance #7002 represents a VGA, 1.2Mbps, 30fps H.264 video datastream mapped to socket port number 5004 using RTP payload ID 98. Finally, media ID #7003 is a motion JPEG video stream with a frame rate of 5fps and a resolution of 960x720 (no bandwidth estimation is provided; i.e. "b=AS:…"). This MJPEG media stream is mapped to socket port number 5005 and has an RTP payload ID of 99. Additionally, a G.711 A-law (RTP payload ID = '8') audio stream, associated with track ID #7 is mapped to socket port number 5003 using RTP payload ID 8. All of these payload IDs are in the dynamic RTP payload ID allocation range. Also note that the RaCM device uses all of the well known SDP mechanisms for advertising frame size/resolution (i.e. "framesize" and the older "cliprect") since both mechanisms have been used in the industry.

In the next example, the above data streams are archived and have a polymorphic nature with respect to time (i.e. each of the media attributes correlates to a specific time segment).

```
…
a=group:FID 7001 7002
m=video 5002 RTP/AVP 97
a=mid:7001
a=trackID:7
a=control:rtsp://Streaming/tracks/7/mid=700
1 a=rtpmap:97 H264/90000
a=fmtp:97 packetization-mode=1;profile-level-id=4D400C;sprop-
parameter-sets=J01ADKkYUI/LgDUGAQa2wrXvfAQ=,KN4JF6A=a=
b=AS:100
a=framerate:15
a=framesize:97 320-240
a=cliprect:0,0,240,32
0
a=range:clock=20090722T103002.54Z-20090722104431.68Z
m=video 5004 RTP/AVP 98
a=mid:7002
a=trackID:7
a=control:rtsp://PSIA/Streaming/tracks/7/mid=700
2 a=rtpmap:98 H264/90000
a=fmtp:97 packetization-mode=1;profile-level-id=4D400C;sprop-
parameter-sets=J01ADKkYUI/LgDUGAQa2wrXvfAQ=,KN4JF6A=a=
b=AS:1200
a=framerate:30
a=framesize:98 640-480 a=cliprect:0,0.480.640
a=range:clock=20090722T104431.71Z-20090722111723.09
Z m=audio 5003 RTP/AVP 96
b=AS:32
a=control:trackID=8;uri=rtsp://PSIA/Streaming/tracks/
8 a=rtpmap:96 G726-32/8000
a=range:clock=20090722T103002.54Z-20090722111723.09Z
...
```

In the above example, we have a polymorphic and poly-temporal set of media streams. The ‘range’ attribute (“a=range…), used within a media section, denotes the time span for that media instance; typically the range attribute is used at the session layer, but poly-temporal media The PSIA uses UTC time for its ‘range’ attribute which is noted by the “clock=” tag. The time notation is in ISO 8602 format similar, but not identical, to XML ‘dateTime’ (see RFC 2326 for more details).

The first video instance, which is QVGA H.264 video @ 15fps, ranges from July 22, 2009 10:30.02.54AM to 10:44.31.68AM. This is followed, chronologically, by a VGA H.264 video segment, @ 30fps, that ranges from July 22, 10:44.31.71AM to 11:17.23.09AM the same day.

Please note that the accompanying audio track (track ID #8) spans both of the video media segments. This audio track is not part of the Flow/media ID group because it is a member of a separate track (#8). The implementation of poly-attributed tracks is up to the manufacturer. However, media *content that is a member of a different, or separate RaCM track, should not be included in flow group unless it was explicitly configured, or allocated, in a track with other media content*.

The next example is a pure poly-temporal example. It could possibly represent an event driven recording session. The video instances have the same characteristics, but they are 'gapped' with respect to time.

```
…
a=group:FID 701 702
m=video 5002 RTP/AVP 97
a=mid:701
a=trackID:7
a=control:rtsp://PSIA/Streaming/tracks/7/mid=701
a=rtpmap:97 H264/90000
a=fmtp:97 packetization-mode=1;profile-level-id=4D400C;sprop-
parameter-sets=J01ADKkYUI/LgDUGAQa2wrXvfAQ=,KN4JF6A=a=
b=AS:640
a=framerate:15
a=framesize:97 640-480
a=cliprect:0,0,480,640
a=range:clock=20090824T113002.54Z-20090824114431.68Z
m=video 5002 RTP/AVP 97
a=mid:702
a=trackID:7
a=control:rtsp://PSIA/Streaming/tracks/7/mid=702
a=rtpmap:98 H264/90000
a=fmtp:98 packetization-mode=1;profile-level-id=4D400C;sprop-
parameter-sets=J01ADKkYUI/LgDUGAQa2wrXvfAQ=,KN4JF6A=a=
b=AS:640
a=framerate:15
a=framesize:98 640-480 a=cliprect:0,0,480,640
a=range:clock=20090824T125131.71Z-20090924131723.09Z
m=audio 5003 RTP/AVP 0
b=AS:64
a=trackID:8 a=control:rtsp://PSIA/Streaming/tracks/8
a=range:clock=20090924T103002.54Z-20090824131723.09Z
...
```

In the above example, the server has a VGA H.264 video track,. @ 15fps, that has 2 media instances/segments. Both occur on August 24th, 2009. The first media segment is from 11:30.02AM to 11:44.31.68AM. The next video segment is from 12:51.31.71PM to 1:17.23.09PM. Please note that due to the fact that both of the video media segments share the same codec attributes, they also share the same network socket connection (port #5002) and RTP payload ID (97). The media segments (i.e. separate media IDs) are only used to notify the consumer that there are time gaps in the video media. Also in the above example, the accompanying audio track (since it falls within the

corresponding time frame) is a G.711 Mu-law (PCMU; RTP Payload ID of zero) that does NOT have a media ID, in this case, since it has its own separate track ID.

Please note that in the above SDP segments, each media instance should MUST have an attribute line indicating the format parameters ("a=fmtp:…") for that specific codec instance. For MPEG-4 codecs this includes the VOS/VOL codec parameters; for H.264 this includes sequence properties ("sprop…") parameter sets..

Also of note, though it is allowable according to RFC 3388 to include audio in a flow group, this practice is not supported by the PSIA. Audio is always considered to be either related track to some set of video tracks and requires lip-synch, or independently play-able as a standalone stream.

12.2.3 Stream Session Management

The final major aspect of poly-attributed tracks, and channels, is in the stream setup, teardown and management mechanics. As defined above, each media instance/stream is mapped to a set of media attributes that are correlated to an RTP session (i.e. socket) and payload ID. This gives the consumer the ability to know what the codec parameters, time associations (if any) and mappings to network connections are: Fundamentally, consumers of poly-attributed streams MUST issue RTSP SETUP messages for each media instance ("mid") that they plan to receive, and, they MUST use the media segment's specified URI which is defined by the "a=control:…" attribute (e.g. "a=control:rtsp://PSIA/Streaming/tracks/7/mid=4"). All of the setup is done irrespective of when a particular stream may occur with respect to time (i.e. poly-temporal tracks or channels).

The following example RTPS SETUP messages indicate the construction of the SETUP URI when activating each media instance an entity plans on receiving. Note that in the example which for archived video, i..e. 'tracks', this field could be replaced for 'channels' when consuming live poly-attributed data.

```
RTSP SETUP rtsp://Streaming/tracks/7/mid=7001…
…
RTSP SETUP rtsp://Streaming/tracks/7/mid=7002…
…
RTSP SETUP rtsp://Streaming/tracks/7/mid=7003…
…
```

In the above example, a consumer desires to receive 3 media streams within a flow group. Each identified with its own media ID ("mid"). The above examples correlate to the exemplary SDP segments used in the prior section of this document. All of the media streams are correlated to track #7 though each is specifically controlled since the consumer can reassign socket numbers per stream if needed. The stream once set, can flow in any order since they may, or may not be, 'wall clock' independent of each other. The consumer must be ready to receive any stream at any time designated by its SDP description, if any is present. Consumers must correlate the payload IDs to the codec-specific instances.

13 /PSIA/Security

RaCM devices are required to provide session-level security for the management operations, such as configuration, status, etc., on a RaCM unit. The requirements fall into 2 basic categories:

- **HTTP security/authentication**: Per the PSIA ***Service Model Specification, Section 4.3***, PSIA devices shall support HTTP and HTTPS session modes for all REST sessions. Additionally, PSIA devices, systems and applications are required to support both HTTP basic and digest-based authentication (RFC 2617).
- **User-based security**: RaCM devices are required to support the setup of user logins on the device itself per the ***IP Media Device API Specification, Sections 4.2.4, 7.10, and 7.11.*** Per the IP Media specification, only the "/Security/AAA/users" and the "/Security/AAA/users/<id>" resources are required to be supported. Support for administrative access, "/Security/adminAccesses" and "/Security/adminAccesses/<id>" are highly recommended.

RaCM devices implementing the above meet the minimum PSIA requirements and are functionally equivalent, from a session level security perspective, PSIA IP Media devices.

14 /PSIA/Custom/SelfExt/ContentMgmt/DynVideo

URI	/PSIA/Custom/SelfExt/ContentMgmt/DynVideo		Type	Service
Methods	Query String(s)	Inbound Data	Return Result	
Notes	Dynamical Video service.			

RaCM v1.1目前不支持对添加的source进行管理。为了支持对source管理并且兼容PSIA IPMD协议，本文档通过扩展DynVideo和DynStreaming服务接口。动态视频通道（IP Camera）与IPMD协议中的视频输入通道具有很大相似性，从应用层面来看，完全可以等同视频输入通道。

动态视频通道产生动态流。当动态视频输入通道被删除时，与该动态视频输入通道相关联的所有动态流通均被删除。

14.1 /PSIA/Custom/SelfExt/ContentMgmt/DynVideo/inputs

URI	/PSIA/Custom/SelfExt/ContentMgmt/DynVideo/inputs		Type	Resource
Function	Access the dynamical video inputs.			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<DynVideoInput>	
Notes	可以把IP Camera添加到NVR或混合DVR，作为NVR或混合DVR的一个动态视频输入通道管理。NVR或混合DVR根据动态视频通道的参数设置IP Camera的对应参数。			

DynVideoInput XML Block

```
<DynVideoInput version="1.0" xmlns="urn:selfextension:psiaext-ver10-xsd">
  <DynVideoInputChannelList/> <!-- opt -->
</DynVideoInput>
```

14.2 /PSIA/Custom/SelfExt/ContentMgmt/DynVideo/inputs/search

URI	/PSIA/Custom/SelfExt/ContentMgmt/DynVideo/inputs/search		Type	Resource
Function	搜索动态通道			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<DynVideoInputChannel>	

Notes	<adminProtocol> IP Camera的管理协议 <adminPort>管理端口

DynVideoSourceList XML Block

```
<DynVideoSourceList version="1.0" xmlns="urn:selfextension:psiaext-ver10-xsd">
  <dynVideoSourceDescriptor> <!-- opt -->
    <id> <!-- req, xs:string;id --></id>
    <adminProtocol> <!--req, xs:string "HIKVISION, SONY, PSIA, ONVIF …"> </adminProtocol>
    <addressingFormatType>
      <!-- req, xs:string, "ipaddress,hostname"-->
    </addressingFormatType>
    <hostName>    <!-- dep, xs:string -->   </hostName>
    <ipAddress>   <!-- dep, xs:string -->   </ipAddress>
    <subnetMask>  <!-- opt, xs:string, subnet mask for IPv4 address -->    </subnetMask>
    <ipv6Address> <!-- dep, xs:string -->   </ipv6Address>
    <bitMask>     <!-- opt, xs:integer, bitmask IPv6 address -->  </bitMask>
    <serialNumber>    <!--opt, xs:string -->    </serialNumber>
    <macAddress>  <!--opt, xs:string;   --> </macAddress>
    <firmwareVersion> <!-- opt, req, xs:string -->   </firmwareVersion>
    <adminPortNo> <!-- opt, xs:integer -->  </adminPortNo>
    <userName>     <!-- opt, xs:string -->    </userName>
    <password>     <!-- opt, xs:string -->    </password>
    <srcInputPortNums> <!-- req, xs:string; id --> </srcInputPortNums>
  </dynVideoSourceDescriptor>
<DynVideoSourceList>
```

14.3 /PSIA/Custom/SelfExt/ContentMgmt/DynVideo/inputs/channels

URI	/PSIA/Custom/SelfExt/ContentMgmt/DynVideo/inputs/channels		Type	Resource
Function	Access dynamical video input channels.			
Methods	Query String(s)	Inbound Data		Return Result
GET		None		<DynVideoInputChannelList>
PUT		<DynVideoInputChannelList>		<ResponseStatus>
POST		<DynVideoInputChannel>		<ResponseStatus>
Notes	Dynamical video inputport can be created or deleted.			

DynVideoInputChannelList XML Block

```
<DynVideoInputChannelList version="1.0" xmlns="urn:selfextension:psiaext-ver10-xsd">
  <DynVideoInputChannel/> <!-- opt -->
</DynVideoInputChannelList>
```

14.4 /PSIA/Custom/SelfExt/ContentMgmt/DynVideo/inputs/channels/status

URI	/PSIA/Custom/SelfExt/ContentMgmt/DynVideo/inputs/channels/status		Type	Resource
Function	Access dynamical video input channels status.			
Methods	Query String(s)	Inbound Data	Return Result	
GET		None	<DynVideoInputChannelStatusList>	
Notes				

DynVideoInputChannelStatusList XML Block

```
<DynVideoInputChannelStatusList version="1.0" xmlns="urn:selfextension:psiaext-ver10-xsd">
  <DynVideoInputChannelStauts/>    <!-- opt -->
</DynVideoInputChannelStatusList>
```

14.5 /PSIA/Custom/SelfExt/ContentMgmt/DynVideo/inputs/channels/<ID>

URI	/PSIA/Custom/SelfExt/ContentMgmt/DynVideo/inputs/channels/*ID*		Type	Resource
Function	Access dynamical video input channel properties.			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<DynVideoInputChannel>	
PUT		<DynVideoInputChannel>	<ResponseStatus>	
DELETE			<ResponseStatus>	
Notes	<sourceInputPortDescriptor>设备使用该标签的内容添加动态通道。 <adminProtocol> IP Camera的管理协议 <adminPort>管理端口 <srcInputPort> 待添加的设备可能有多个videoInputPort，该标签用于指定添加哪个videoInputPort。 删除动态通后，与该动态相关联的动态流将被删除。 <srcLogin> 待添加通道的用户名和密码例如：“admin:12345”			

DynVideoInputChannel XML Block

```
<DynVideoInputChannel version="1.0" xmlns="urn:selfextension:psiaext-ver10-xsd">
  <id>   <!-- req, xs:string;id --></id>
  <name>     <!-- opt, xs:string>  </name>
  <sourceInputPortDescriptor> <!-- req -->
    <adminProtocol> <!--req, xs:string "HIKVISION, SONY, PSIA, ONVIF …"> </adminProtocol>
    <addressingFormatType>
      <!-- req, xs:string, "ipaddress,hostname"-->
    </addressingFormatType>
    <hostName>    <!-- dep, xs:string -->   </hostName>
    <ipAddress>   <!-- dep, xs:string -->   </ipAddress>
    <ipv6Address> <!-- dep, xs:string -->   </ipv6Address>
```

```
    <adminPortNo> <!-- req, xs:integer --> </adminPortNo>
    <srcInputPort> <!-- req, xs:string; id --> </srcInputPort>
    <userName>     <!-- req, xs:string -->   </userName>
    <password>     <!-- req, xs:string -->   </password>
  </sourceInputPortDescriptor>
  <powerLineFrequencyMode> <!-- opt, xs:string "50hz, 60hz" --> </powerLineFrequencyMode>
  <whiteBalanceMode>
    <!-- opt, xs:string,
      "manual,auto,indoor/incandescent,fluorescent/white, fluorescent/yellow,outdoor,
      black&white" -->
  </whiteBalanceMode>
  <whiteBalanceLevel><!-- dep, xs:integer, 0..100 --></whiteBalanceLevel>
  <exposureMode><!-- opt, xs:string, "manual, auto" --></exposureMode>
  <Exposure><!-- opt -->
    <exposureTarget><!-- req, xs:integer, microseconds --></exposureTarget>
    <exposureAutoMin><!-- req, xs:integer, microseconds --></exposureAutoMin>
    <exposureAutoMax><!-- req, xs:integer, microseconds --></exposureAutoMax>
  </Exposure>
  <GainWindow><!-- opt -->
    <RegionCoordinatesList>    <!-- opt -->
      <RegionCoordinates><!-- opt -->
        <positionX><!-- req, xs:integer;coordinate --></positionX>
        <positionY><!-- req, xs:integer;coordinate --></positionY>
      </RegionCoordinates>
    </RegionCoordinatesList>
  </GainWindow>
  <gainLevel><!-- dep, xs:integer, 0..100 -->   </gainLevel>
  <brightnessLevel>   <!-- opt, xs:integer, 0..100 -->   </brightnessLevel>
  <contrastLevel> <!-- opt, xs:integer, 0..100 -->   </contrastLevel>
  <sharpnessLevel>    <!-- opt, xs:integer, 0..100 -->   </sharpnessLevel>
  <saturationLevel>   <!-- opt, xs:integer, 0..100 -->   </saturationLevel>
  <hueLevel> <!-- opt, xs:integer, 0..100 -->   </hueLevel>
  <gammaCorrectionEnabled> <!-- opt, xs:boolean --> </gammaCorrectionEnabled>
  <gammaCorrectionLevel>  <!-- opt, xs:integer, 0..100 -->  </gammaCorrectionLevel>
  <WDREnabled>    <!-- opt, xs:boolean -->  </WDREnabled>
  <WDRLevel> <!-- opt, xs:integer, 0..100 -->   </WDRLevel>
  <LensList> <!-- opt -->
    <Lens>    <!-- opt -->
      <lensModuleName> <!-- opt, xs:string -->   </lensModuleName>
      <irisMode>
        <!-- opt, xs:string, "manual,auto,override" -->
      </irisMode>
      <focusMode>
        <!-- opt, xs:string, "manual,auto,autobackfocus,override" -->
      </focusMode>
    </Lens>
  </LensList>
  <DayNightFilter>    <!-- opt -->
    <dayNightFilterType>
      <!-- req, xs:string, "day,night,auto" -->
    </dayNightFilterType>
    <switchScheduleEnabled><!-- opt, xs:boolean -->  </switchScheduleEnabled>
    <beginTime>   <!-- dep, xs:time --> </beginTime>
    <endTime> <!-- dep, xs:time --> </endTime>
  </DayNightFilter>
<DynVideoInputChannel>
```

14.6 /PSIA/Custom/SelfExt/ContentMgmt/DynVideo/inputs/channels/<ID>/password

URI	/PSIA/Custom/SelfExt/ContentMgmt/DynVideo/inputs/channels/<ID>/password	Type	Resource

Function	设置提供该通道源的访问密码		
Methods	**Query String(s)**	**Inbound Data**	**Return Result**
PUT		None	<DynVideoInputPassword>
Notes	<oldPassword> 动态通道源的旧的访问密码。旧密码验证成功后，才能修改密码。 <newPassword> 要修改的密码		

DynVideoInputPassword XML Block

```
<DynVideoInputPassword version="1.0" xmlns=urn:selfextension:psiaext-ver10-xsd>
  <oldPassword> <!-- req, xs:string --> </oldPassword>
  <newPassword> <!--req, xs:string --> </newPassword>
</DynVideoInputPassword>
```

14.7 /PSIA/Custom/SelfExt/ContentMgmt/DynVideo/inputs/channels/<ID>/netParam

URI	/PSIA/Custom/SelfExt/ContentMgmt/DynVideo/inputs/channels/<ID>/netParam	Type	Resource
Function	设置提供该通道的源的网络相关参数		
Methods	**Query String(s)**	**Inbound Data**	**Return Result**
PUT		None	<DynVideoInputNetParam>
Notes	<ipAddress> 待设定的IPv4地址。是否含有此元素，取决于提供该通道的源是否支持IPv4。 <ipv6Address> 待设定的IPv6地址。是否含有此元素，取决于提供该通道的源是否支持IPv6 <adminPortNo>设定提供该通道的源的管理端口		

DynVideoInputNetParam XML Block

```
<DynVideoInputNetParam version="1.0" xmlns=urn:selfextension:psiaext-ver10-xsd>
  <ipAddress>    <!-- opt, xs:string -->    </ipAddress>
  <ipv6Address>   <!-- opt, xs:string -->   </ipv6Address>
  <adminPortNo>   <!-- opt, xs:integer -->  </adminPortNo>
</DynVideoInputNetParam>
```

14.8 /PSIA/Custom/SelfExt/ContentMgmt/DynVideo/inputs/channels/<ID>/status

URI	/PSIA/Custom/SelfExt/ContentMgmt/DynVideo/inputs/channels/status	Type	Resource
Function	Access dynamical video input channels status.		
Methods	**Query String(s)**	**Inbound Data**	**Return Result**
GET		None	<DynVideoInputChannelStatusList>

Notes	<online> 该动态通道是否在线 <dynStreamingChannelIdList> 该动态通道含有的动态流 <relatedDynIO>与该通道相关联的动态IO

DynVideoInputChannelStatus XML Block

```
<DynVideoInputChannelStatus version="1.0" xmlns=urn:selfextension:psiaext-ver10-xsd>
  <id> <!-- req, xs:string; id --> </id>
  <sourceInputPortDescriptor> <!-- req -->
    <adminProtocol> <!--req, xs:string "HIKVISION, SONY, PSIA, ONVIF …"> </adminProtocol>
    <addressingFormatType>
      <!-- req, xs:string, "ipaddress,hostname"-->
    </addressingFormatType>
    <hostName>    <!-- dep, xs:string -->   </hostName>
    <ipAddress>   <!-- dep, xs:string -->   </ipAddress>
    <ipv6Address> <!-- dep, xs:string -->   </ipv6Address>
    <adminPortNo> <!-- opt, xs:integer -->  </adminPortNo>
    <srcInputPort> <!-- req, xs:string; id --> </srcInputPort>
  </sourceInputPortDescriptor>
  <online> <!-- req, xs:boolean --> </online>
  <dynStreamingChannelIdList> <!-- req -->
    <dynStreamingChannelId> <!-- req, xs:string; id -->  </dynStreamingChannelId>
  </dynStreamingChannelIdList>
  <relatedDynIO> <!-- opt -->
    <dynInputPortIdList>
      <dynInputPortId/> <!-- opt -->
    </dynInputPortIdList>
    <dynOutputPortIdList>
      <dynOutputPortId/> <!-- opt -->
    </dynOutputPortIdList>
  </relatedDynIO>
</DynVideoInputChannelStatus>
```

14.9 /PSIA/Custom/SelfExt/ContentMgmt/DynVideo/inputs/channels/<ID>/focus

URI	/PSIA/Custom/SelfExt/ContentMgmt/DynVideo/inputs/channels/*ID*/focus		Type	Resource
Function	Manually focus a video input channel.			
Methods	Query String(s)	Inbound Data	Return Result	
PUT	focus	<FocusData>	<ResponseStatus>	
Notes	<focus>: focus vector data. Negative numbers focus near, positive numbers focus far. Numerical value is a percentage of the maximum focus speed of the lens module.			

FocusData XML Block

```
<FocusData version="1.0" xmlns="urn:psialliance-org">
  <focus><!-- req, xs:intger, -100..100 --> </focus>
</FocusData>
```

14.10 /PSIA/Custom/SelfExt/ContentMgmt/DynVideo/inputs/channels/<ID>/iris

URI	/PSIA/Custom/SelfExt/ContentMgmt/DynVideo/inputs/channels/*ID*/iris		Type	Resource
Function	Manually adjust iris for a video input channel.			
Methods	Query String(s)	Inbound Data	Return Result	
PUT	iris	<IrisData>	<ResponseStatus>	
Notes	<iris> negative numbers close iris, positive numbers open iris. Numerical value is a percentage of the maximum iris speed of the lens module.			

IrisData XML Block

```
<IrisData version="1.0" xmlns="urn:psialliance-org">
  <iris> <!-- req, xs:integer, -100..100 -->    </iris>
</IrisData>
```

14.11 /PSIA/Custom/SelfExt/ContentMgmt/DynVideo/inputs/channels/<ID>/lens

URI	/PSIA/Custom/SelfExt/ContentMgmt/DynVideo/inputs/channels/*ID*/lens		Type	Resource
Function	Query lens information.			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<LensStatus>	
Notes	<absoluteFocus> indicates the current absolute focus position. 0 is focus near, 100 is focus far. <absoluteIris> indicates the current absolute iris position. 0 is completely closed, 100 is completely open.			

LensStatus XML Block

```
<LensStatus version="1.0" xmlns="urn:psialliance-org">
  <Absolute>
    <absoluteFocus>         <!-- req, xs:integer, 0..100 --> </absoluteFocus>
    <absoluteIris>          <!-- req, xs:integer, 0..100 --> </absoluteIris>
  </Absolute>
</LensStatus>
```

14.12 /PSIA/Custom/SelfExt/ContentMgmt/DynVideo/inputs/channels/<ID>/overlays

URI	/PSIA/Custom/SelfExt/ContentMgmt/DynVideo/inputs/channels/*ID*/overlays	Type	Resource

Function	Configure and access text and image overlays.		
Methods	**Query String(s)**	**Inbound Data**	**Return Result**
GET			<VideoOverlay>
PUT		<VideoOverlay>	<ResponseStatus>
DELETE			<ResponseStatus>
Notes	IP media devices can overlay additional information on the encoded video stream. These overlays can be either text information or a set of images. Overlays are composited together in ID-order when displayed in the video. Overlay images are managed with `/PSIA/System/Video/overlayImages`.		

VideoOverlay XML Block

```
<VideoOverlay version="1.0" xmlns="urn:psialliance-org">
  <TextOverlayList/>        <!-- opt -->
  <ImageOverlayList/>       <!-- opt -->
</VideoOverlay>
```

14.13 /PSIA/Custom/SelfExt/ContentMgmt/DynVideo/inputs/channels/<ID>/overlays/text

URI	/PSIA/Custom/SelfExt/ContentMgmt/DynVideo/inputs/channels/*ID*/overlays/text	Type	Resource
Function	Access and configure text overlays for a particular video channel.		
Methods	**Query String(s)**	**Inbound Data**	**Return Result**
GET			<TextOverlayList>
PUT		<TextOverlayList>	<ResponseStatus>
POST		<TextOverlay>	<ResponseStatus>
DELETE			<ResponseStatus>
Notes	A set of text overlays is managed. They are composited over the video signal in increasing ID-order.		

TextOverlayList XML Block

```
<TextOverlayList version="1.0" xmlns="urn:psialliance-org">
  <TextOverlay/>  <!-- opt -->
</TextOverlayList>
```

14.14 /PSIA/Custom/SelfExt/ContentMgmt/DynVideo/inputs/channels/<ID>/overlays/text/<ID>

URI	/PSIA/Custom/SelfExt/ContentMgmt/DynVideo/inputs/channels/*ID*/overlays/text/*ID*	Type	Resource
Function	Access and configure a particular text overlay for a video channel.		
Methods	**Query String(s)**	**Inbound Data**	**Return Result**

GET			TextOverlay>
PUT		<TextOverlay>	<ResponseStatus>
DELETE			<ResponseStatus>
Notes	A text overlay can contain time information and static text with color and transparency information.		

TextOverlay XML Block

```
<TextOverlay version="1.0" xmlns="urn:psialliance-org">
  <id>                    <!-- req, xs:string;id -->       </id>
  <enabled>               <!-- req, xs:boolean -->         </enabled>
  <timeStampEnabled>      <!-- opt, xs:boolean -->         </timeStampEnabled>
  <dateTimeFormat>        <!-- dep, xs:string -->          </dateTimeFormat>
  <backgroundColor>       <!-- opt, xs:hexBinary;color --> </backgroundColor>
  <fontColor>             <!-- opt, xs:hexBinary;color --> </fontColor>
  <fontSize>              <!-- opt, xs:integer, pixels --> </fontSize>
  <displayText>           <!-- req, xs:string -->          </displayText>
  <horizontalAlignType> <!-- opt, xs:string, "left,right,center" --></horizontalAlignType>
  <verticalAlignType>     <!-- opt, xs:string, "top,bottom" -->       </verticalAlignType>
</TextOverlay>
```

14.15 /PSIA/Custom/SelfExt/ContentMgmt/DynVideo/inputs/channels/<ID>/overlays/image

URI	/PSIA/Custom/SelfExt/ContentMgmt/DynVideo/inputs/channels/*ID*/overlays/image	**Type**	Resource
Function	Access and configure image overlays for a particular video channel.		
Methods	**Query String(s)**	**Inbound Data**	**Return Result**
GET			<ImageOverlayList>
PUT		<ImageOverlayList>	<ResponseStatus>
POST		<ImageOverlay>	<ResponseStatus>
DELETE			<ResponseStatus>
Notes	A set of image overlays is managed. They are composited over the video signal in increasing ID-order.		

ImageOverlayList XML Block

```
<ImageOverlayList version="1.0" xmlns="urn:psialliance-org">
  <ImageOverlay/> <!-- opt -->
</ImageOverlayList>
```

14.16 /PSIA/Custom/SelfExt/ContentMgmt/DynVideo/inputs/channels/<ID>/overlays/image/<ID>

URI	/PSIA/Custom/SelfExt/ContentMgmt/DynVideo/inputs/channels/*ID*/overlays/image/*ID*	**Type**	Resource

Function	Access and configure a particular image overlay for a video channel.		
Methods	**Query String(s)**	**Inbound Data**	**Return Result**
GET			<ImageOverlay>
PUT		<ImageOverlay>	<ResponseStatus>
DELETE			<ResponseStatus>
Notes	An image overlay can contain time information and static text with color and transparency information. In order to enable image overlay, an image must have been previously uploaded to the device using the `/PSIA/System/Video/overlayImages` command.		

ImageOverlay XML Block

```
<ImageOverlay version="1.0" xmlns="urn:psialliance-org">
  <id>                          <!-- req, xs:string;id -->           </id>
  <enabled>                     <!-- req, xs:boolean -->             </enabled>
  <imageName>                   <!-- req, xs:string -->              </imageName>
  <positionX>                   <!-- opt, xs:integer;coordinate -->  </positionX>
  <positionY>                   <!-- opt, xs:integer;coordinate -->  </positionY>
  <transparentColorEnabled> <!-- opt, xs:boolean --> </transparentColorEnabled>
  <transparentColor>            <!-- dep, xs:hexBinary;color -->     </transparentColor>
</ImageOverlay>
```

14.17 /PSIA/Custom/SelfExt/ContentMgmt/DynVideo/inputs/channels/<ID>/privacyMask

URI	/PSIA/Custom/SelfExt/ContentMgmt/DynVideo/inputs/channels/*ID*/privacyMask	Type	Resource
Function	Access and configure privacy masking.		
Methods	**Query String(s)**	**Inbound Data**	**Return Result**
GET			<PrivacyMask>
PUT		<PrivacyMask>	<ResponseStatus>
Notes	Privacy masking can be enabled and the region list configured per channel.		

PrivacyMask XML Block

```
<PrivacyMask version="1.0" xmlns="urn:psialliance-org">
  <enabled>  <!-- req, xs:boolean -->  </enabled>
  <PrivacyMaskRegionList/>  <!-- opt -->
</PrivacyMask>
```

14.18 /PSIA/Custom/SelfExt/ContentMgmt/DynVideo/inputs/channels/<ID>/privacyMask/regions

URI	/PSIA/Custom/SelfExt/ContentMgmt/DynVideo/inputs/channels/*ID*/privacyMask/regions	Type	Resource

Function	Access and configure privacy mask regions.		
Methods	**Query String(s)**	**Inbound Data**	**Return Result**
GET			<PrivacyMaskRegionList>
PUT		<PrivacyMaskRegionList>	<ResponseStatus>
POST		<PrivacyMaskRegion>	<ResponseStatus>
DELETE			<ResponseStatus>
Notes	Privacy masking consists of a set of regions that are combined to grey or black out areas of a video input.		

PrivacyMaskRegionList XML Block

```
<PrivacyMaskRegionList version="1.0" xmlns="urn:psialliance-org">
  <PrivacyMaskRegion/><!-- opt -->
</PrivacyMaskRegionList>
```

14.19 /PSIA/Custom/SelfExt/ContentMgmt/DynVideo/inputs/channels/<ID>/privacyMask/regions/<ID>

URI	/PSIA/Custom/SelfExt/ContentMgmt/DynVideo/inputs/channels/ID/privacyMask/regions/ID	Type	Resource

Function	Access and configure a particular privacy mask region.		
Methods	**Query String(s)**	**Inbound Data**	**Return Result**
GET			<PrivacyMaskRegion>
PUT		<PrivacyMaskRegion>	<ResponseStatus>
DELETE			<ResponseStatus>
Notes	Region coordinates are dependent on video resolution. Regions will be "drawn" from the coordinates provided in a top-down fashion. At least three <RegionCoordinates> blocks must be provided for a single <PrivacyMaskRegion> block. Ordering of <PrivacyMaskRegion> blocks is insignificant.		

PrivacyMaskRegion XML Block

```
<PrivacyMaskRegion version="1.0" xmlns="urn:psialliance-org">
  <id>                        <!-- req, xs:string;id -->              </id>
  <enabled>                   <!-- req, xs:boolean -->                </enabled>
  <RegionCoordinatesList>     <!-- req -->
    <RegionCoordinates>       <!-- req, at least one if list is defined -->
      <positionX>             <!-- req, xs:integer;coordinate -->     </positionX>
      <positionY>             <!-- req, xs:integer;coordinate  -->    </positionY>
    </RegionCoordinates>
  </RegionCoordinatesList>
</PrivacyMaskRegion>
```

15 /PSIA/Custom/SelfExt/ContentMgmt/ZeroVideo

URI	/PSIA/Custom/SelfExt/ContentMgmt/ZeroVideo		Type	Service
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
Notes	Zero Video service.			

15.1 /PSIA/Custom/SelfExt/ContentMgmt/ZeroVideo/channels

URI	/PSIA/Custom/SelfExt/ContentMgmt/ZeroVideo/channels		Type	Resource
Function	Access zero video channels.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET		None	<ZeroVideoChannelList>	
Notes	Since zero video input channels are resources that are defined by the hardware configuration of the device, they cannot be created or deleted.			

ZeroVideoChannelList XML Block

```
<ZeroVideoChannelList version="1.0" xmlns="urn:selfextension:psiaext-ver10-xsd">
  <ZeroVideoChannel/> <!-- opt -->
</ZeroVideoChannelList>
```

15.2 /PSIA/Custom/SelfExt/ContentMgmt/ZeroVideo/channels /<ID>

URI	/PSIA/Custom/SelfExt/ContentMgmt/ZeroVideo/channels/*ID*	Type	Resource
Function	Access zero video input channel properties.		

Methods	Query String(s)	Inbound Data	Return Result
GET			<ZeroVideoChannel>
PUT		<ZeroVideoChannel>	<ResponseStatus>
Notes	none		

ZeroVideoChannel XML Block

```
<ZeroVideoChannel version="1.0" xmlns="urn:selfextension:psiaext-ver10-xsd">
  <id>   <!-- req, xs:string;id --></id>
  <enabled> <!--req, xs:Boolean --> </eanbled>
  <inputPort> <!-- req, xs:string -->  </inputPort>
  <brightnessLevel>   <!-- opt, xs:integer, 0..100 -->   </brightnessLevel>
  <contrastLevel> <!-- opt, xs:integer, 0..100 -->   </contrastLevel>
  <sharpnessLevel>    <!-- opt, xs:integer, 0..100 -->   </sharpnessLevel>
  <saturationLevel>   <!-- opt, xs:integer, 0..100 -->   </saturationLevel>
  <hueLevel> <!-- opt, xs:integer, 0..100 -->   </hueLevel>
<ZeroVideoChannel>
```

15.3 /PSIA/Custom/SelfExt/ContentMgmt/ZeroVideo/channels /<ID>/enlarge

URI	/PSIA/Custom/SelfExt/ContentMgmt/ZeroVideo/channels/ID/enlarge	Type	Resource
Function	Get or set zero chan video input enlarge configuration		

Methods	Query String(s)	Inbound Data	Return Result
GET			<ZeroVideoEnlarge>
PUT		<ZeroVideoEnlarge>	<ResponseStatus>
Notes	mousePosition: device use this element to decide which sub screen should be enlarged.		

ZeroVideoEnlarge XML Block

```
<ZeroVideoEnlarge version="1.0" xmlns="urn:selfextension:psiaext-ver10-xsd">
  <stat> <!--req, xs:string, "normal, enlarge" --> </stat>
  <mousePosition> <!--wr,dep -->
    <x> <!--req, xs:integer --> </x>
    <y> <!--req, xs:integer --> </y>
  </mousePosition>
<ZeroVideoEnlarge>
```

15.4 /PSIA/Custom/SelfExt/ContentMgmt/ZeroVideo/channels/<ID>/switchScreen

URI	/PSIA/Custom/SelfExt/ContentMgmt/ZeroVideo/channels/ID/switchScreen		Type	Resource
Function	Switch screen			
Methods	Query String(s)	Inbound Data	Return Result	
PUT		<ZeroVideoSwitch>	<ResponseStatus>	
Notes				

ZeroVideoSwitch XML Block

```
<ZeroVideoSwitch version="1.0" xmlns="urn:selfextension:psiaext-ver10-xsd">
  <mode> <!--req, xs:string, "back, next" --> </mode>
<ZeroVideoSwitch>
```

15.5 /PSIA/Custom/SelfExt/ContentMgmt/ZeroVideo/channels/<ID>/previewCfg

URI	/PSIA/Custom/SelfExt/ContentMgmt/ZeroVideo/channels/ID/previewCfg		Type	Resource
Function	Get or set zero chan video input preview configuration			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<ZeroVideoPreview>	
PUT		<ZeroVideoPreview>	<ResponseStatus>	
Notes	screenMode: sub screen nums per screen subScreenOrder: sub screen order. Attribute 'order' represent sub screen order, for example: order="1,3,5,2,4".			

ZeroVideoPreview XML Block

```
<ZeroVideoPreview version="1.0" xmlns="urn:selfextension:psiaext-ver10-xsd">
  <screenMode> <!--req, xs:integer --> </screenMode>
  <enAudio> <!-- req, xs:boolean --> </enAudio>
  <switchInterval> <!--req, xs:integer, in sec--> <switchInterval>
  <subScreenOrderList> <!-- req -->
    <subScreenOrder order="">
      <id> <!--xs:string; id --> </id>
      <screenMode> <!--req, xs:integer --> </screeMode>
    </subScreenOrder>
  <subScreenOrderList>
<ZeroVideoPreview>
```

16 /PSIA/Custom/SelfExt/ContentMgmt/ZeroStreaming

URI	/PSIA/Custom/SelfExt/ContentMgmt/ZeroStreaming		Type	Service
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
Notes	Zero Streaming service			

16.1 /PSIA/Custom/SelfExt/ContentMgmt/ZeroStreaming/status

URI	/PSIA/Custom/SelfExt/ContentMgmt/ZeroStreaming/status		Type	Resource
Function	Query the device zero streaming status.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<ZeroStreamingStatus>	
Notes	This command accesses the status of all device zero streaming sessions.			

ZeroStreamingStatus XML Block

```
<ZeroStreamingStatus version="1.0" xmlns="urn:selfextension:psiaext-ver10-xsd">
  <totalStreamingSessions>  <!-- req, xs:integer -->  </totalStreamingSessions>
  <StreamingSessionStatusList/>  <!-- dep, only if there are sessions -->
</ZeroStreamingStatus>
```

16.2 /PSIA/Custom/SelfExt/ContentMgmt/ZerStreaming/channels

URI	/PSIA/Custom/SelfExt/ContentMgmt/ZeroStreaming/channels		Type	Resource
Function	Zero Streaming channels.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<ZeroStreamingChannelList>	
PUT		<ZeroStreamingChannelList>	<ResponseStatus>	
POST		<ZeroStreamingChannel>	<ResponseStatus>	
DELETE			<ResponseStatus>	

Notes	Zero Streaming channels may be hardwired, or it may be possible to create multiple streaming channels per input if the device supports it. To determine whether it is possible to dynamically create streaming channels, check the defined HTTP methods in `/PSIA/Custom/SelfExt/ContentMgmt/Streaming/channels/description`.

ZeroStreamingChannelList XML Block

```
<ZeroStreamingChannelList version="1.0" xmlns="urn:selfextension:psiaext-ver10-xsd">
  <ZeroStreamingChannel/> <!-- opt -->
</ZeroStreamingChannelList>
```

16.3 /PSIA/Custom/SelfExt/ContentMgmt/ZeroStreaming/channels/<ID>

URI	`/PSIA/Custom/SelfExt/ContentMgmt/ZeroStreaming/channels/`*ID*		**Type**	Resource
Function	Access zero streaming channels.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<ZeroStreamingChannel>	
PUT		<ZeroStreamingChannel>	<ResponseStatus>	
DELETE			<ResponseStatus>	
Notes	<videoInputChannelID> refers to `/PSIA/Custom/SelfExt/ContentMgmt/ZeroVideo/channels/`*ID*.			

ZeroStreamingChannel XML Block

```
<ZeroStreamingChannel version="1.0" xmlns="urn:selfextension:psiaext-ver10-xsd">
  <id>   <!-- req, xs:string;id --></id>
  <channelName> <!-- req, xs:string --> </channelName>
  <enabled>  <!-- req, xs:boolean -->  </enabled>
  <Video>
    <!-- opt -->
    <enabled><!-- req, xs:boolean --></enabled>
    <videoInputChannelID> <!-- req, xs:string;id --> </videoInputChannelID>
    <videoCodecType>
      <!-- req, xs:string, "MPEG4,MJPEG,3GP,H.264,MPNG" -->
    </videoCodecType>
    <videoResolutionWidth> <!-- req, xs:integer -->  </videoResolutionWidth>
    <videoResolutionHeight> <!-- req, xs:integer --> </videoResolutionHeight>
    <videoQualityControlType>
      <!-- opt, xs:string, "cbr,vbr" -->
     </videoQualityControlType>
    <constantBitRate> <!-- dep, xs:integer, in kbps -->  </constantBitRate>
    <vbrUpperCap> <!-- dep, xs:integer, in kbps -->  </vbrUpperCap>
    <vbrLowerCap> <!-- dep, xs:integer, in kbps -->  </vbrLowerCap>
    <maxFrameRate><!-- req, xs:integer, maximum frame rate x100 -->   </maxFrameRate>
  </Video>
</ZeroStreamingChannel>
```

16.4 /PSIA/Custom/SelfExt/ContentMgmt/ZeroStreaming/channels/<ID>/status

URI	/PSIA/Custom/SelfExt/ContentMgmt/ZeroStreaming/channels/*ID*/status		Type	Resource
Function	Get the list of zero streaming sessions associated with a particular channel.			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<ZeroStreamingSessionStatusList>	
Notes	Use of IPv4 or IPv6 addresses depends on the value of the <ipVersion> field in /PSIA/System/Network/interfaces/*ID*/ipAddress.			

StreamingSessionStatus XML Block

```
<ZeroStreamingSessionStatusList version="1.0" xmlns="urn:selfextension:psiaext-ver10-xsd">
  <StreamingSessionStatus>
    <clientAddress>          <!-- req -->
      <ipAddress>  <!-- dep, xs:string -->  </ipAddress>
      <ipv6Address>  <!-- dep, xs:string -->  </ipv6Address>
    </clientAddress>
    <clientUserName>         <!-- opt, xs:string -->              </clientUserName>
    <startDateTime>          <!-- opt, xs:datetime -->            </startDateTime>
    <elapsedTime>            <!-- opt, xs:integer, seconds -->    </elapsedTime>
    <bandwidth>              <!-- opt, xs:integer, in kbps -->    </bandwidth>
  </StreamingSessionStatu
s>
</ZeroStreamingSessionStatusList>
```

17 /PSIA/Custom/SelfExt/ContentMgmt/DynStreaming

URI	/PSIA/Custom/SelfExt/ContentMgmt/DynStreaming		Type	Service
Methods	Query String(s)	Inbound Data	Return Result	
Notes	dynamical Streaming service			

17.1 /PSIA/Custom/SelfExt/ContentMgmt/DynStreaming/status

URI	/PSIA/Custom/SelfExt/ContentMgmt/DynStreaming/status		Type	Resource
Function	Query the device streaming status.			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<DynStreamingStatus>	
Notes	This command accesses the status of all device streaming sessions.			

DynStreamingStatus XML Block

```
<StreamingStatus version="1.0" xmlns="urn:selfextension:psiaext-ver10-xsd">
  <totalStreamingSessions>  <!-- req, xs:integer -->  </totalStreamingSessions>
  <StreamingSessionStatusList/>  <!-- dep, only if there are sessions -->
</StreamingStatus>
```

17.2 /PSIA/Custom/SelfExt/ContentMgmt/DynStreaming/channels

URI	/PSIA/Custom/SelfExt/ContentMgmt/DynStreaming/channels		Type	Resource
Function	Streaming channels.			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<DynStreamingChannelList>	
PUT		<DynStreamingChannelList>	<ResponseStatus>	
POST		<DynStreamingChannel>	<ResponseStatus>	
DELETE			<ResponseStatus>	

Notes	动态流为动态视频通道所固有。如果动态通道支持创建或删除动态流的话，可以在此动态通道内创建或创建动态流。通过查看/PSIA/Custom/SelfExt/ContentMgmt/DynVideo/inputs/channels/<ID>/status 中<supportMutiStream>的值，可以获取该动态通道是否支持创建动态流。

DynStreamingChannelList XML Block

```
<DynStreamingChannelList version="1.0" xmlns="urn:selfextension:psiaext-ver10-xsd">
  <dynStreamingChannel/>  <!-- opt -->
</DynStreamingChannelList>
```

17.3 /PSIA/Custom/SelfExt/ContentMgmt/DynStreaming/channels/<ID>

URI	/PSIA/Custom/SelfExt/ContentMgmt/DynStreaming/channels/*ID*	Type	Resource
Function	Access streaming channels.		

Methods	Query String(s)	Inbound Data	Return Result
GET			<StreamingChannel>
PUT		<StreamingChannel>	<ResponseStatus>
DELETE			<ResponseStatus>

Notes

<ControlProtocolList> identifies the control protocols that are valid for this type of streaming.

<Unicast> is for direct unicast streaming.

<Multicast> is for direct multicast streaming.

<videoSourcePortNo> and <audioSourcePortNo> are the source port numbers for the outbound video or audio streams.

<videoInputChannelID> refers to /PSIA/System/Video/inputs/channel/*ID*.

<audioInputChannelID> refers to /PSIA/System/Audio/channels/*ID*. It must be configured as an input channel.

Use of IPv4 or IPv6 addresses depends on the value of the <ipVersion> field in /PSIA/System/Network/interfaces/*ID*/ipAddress.

<Security> determines whether SRTP is used for stream encryption.

<audioResolution> is the resolution for the outbound audio stream in bits.

DynStreamingChannel XML Block

```
<dynStreamingChannel version="1.0" xmlns="urn:selfextension:psiaext-ver10-xsd">
  <id>   <!-- req, xs:string;id --></id>
  <channelName> <!-- req, xs:string --> </channelName>
  <enabled>  <!-- req, xs:boolean -->  </enabled>
  <Transport>
    <!-- req -->
    <rtspPortNo>  <!-- opt, xs:integer -->  </rtspPortNo>
    <maxPacketSize>    <!-- opt, xs:integer -->  </maxPacketSize>
    <audioPacketLength>    <!-- opt, xs:integer -->  </audioPacketLength>
    <audioInboundPacketLength><!-- opt, xs:integer -->   </audioInboundPacketLength>
    <audioInboundPortNo>   <!-- opt, xs:integer -->  </audioInboundPortNo>
    <videoSourcePortNo>    <!-- opt, xs:integer -->  </videoSourcePortNo>
    <audioSourcePortNo>    <!-- opt, xs:integer -->  </audioSourcePortNo>
    <ControlProtocolList>
      <!-- req -->
      <ControlProtocol>
        <!-- req -->
        <streamingTransport>
           <!-- req, xs:string, "HTTP,RTSP" -->
         </streamingTransport>
      </ControlProtocol>
    </ControlProtocolList>
    <Unicast>
      <!-- opt -->
      <enabled>   <!-- req, xs:boolean -->  </enabled>
      <interfaceID>    <!-- opt, xs:string -->   </interfaceID>
      <rtpTransportType>
         <!-- opt, xs:string, "RTP/UDP,RTP/TCP" -->
       </rtpTransportType>
```

```
    </Unicast>
    <Multicast>
      <!-- opt -->
      <enabled>   <!-- req, xs:boolean -->  </enabled>
      <userTriggerThreshold> <!-- opt, xs:integer --></userTriggerThreshold>
      <destIPAddress>  <!-- dep, xs:string -->   </destIPAddress>
      <videoDestPortNo><!-- opt, xs:integer --></videoDestPortNo>
      <audioDestPortNo><!-- opt, xs:integer --></audioDestPortNo>
      <destIPv6Address><!-- dep, xs:string --></destIPv6Address>
      <ttl><!-- opt, xs:integer --></ttl>
    </Multicast>
    <Security>
      <!-- opt -->
      <enabled><!-- req, xs:boolean --></enabled>
    </Security>
  </Transport>
  <Video>
    <!-- opt -->
    <enabled><!-- req, xs:boolean --></enabled>
    <dynVideoInputChannelID><!-- req, xs:string;id --></dynVideoInputChannelID>
    <videoCodecType>
      <!-- req, xs:string, "MPEG4,MJPEG,3GP,H.264,MPNG" -->
     </videoCodecType>
    <videoScanType>
      <!-- opt, xs:string, "progressive,interlaced" -->
     </videoScanType>
    <videoResolutionWidth> <!-- req, xs:integer -->  </videoResolutionWidth>
    <videoResolutionHeight>    <!-- req, xs:integer -->  </videoResolutionHeight>
    <videoPositionX>  <!-- opt, xs:integer -->  </videoPositionX>
    <videoPositionY>  <!-- opt, xs:integer -->  </videoPositionY>
    <videoQualityControlType>
      <!-- opt, xs:string, "cbr,vbr" -->
     </videoQualityControlType>
    <constantBitRate> <!-- dep, xs:integer, in kbps -->  </constantBitRate>
    <fixedQuality><!-- opt, xs:integer, percentage, 0..100 -->    </fixedQuality>
    <vbrUpperCap> <!-- dep, xs:integer, in kbps -->  </vbrUpperCap>
    <vbrLowerCap> <!-- dep, xs:integer, in kbps -->  </vbrLowerCap>
    <maxFrameRate><!-- req, xs:integer, maximum frame rate x100 -->   </maxFrameRate>
    <keyFrameInterval> <!-- opt, xs:integer, milliseconds --> </keyFrameInterval>
    <rotationDegree>  <!-- opt, xs:integer, degrees, 0..360 --></rotationDegree>
    <mirrorEnabled>   <!-- opt, xs:boolean -->  </mirrorEnabled>
    <snapShotImageType><!-- opt, xs:string, "JPEG,GIF,PNG" --> </snapShotImageType>
  </Video>
  <Audio>
    <!-- opt -->
    <enabled> <!-- req, xs:boolean -->  </enabled>
    <audioInputChannelID>  <!-- req, xs:string;id --></audioInputChannelID>
    <audioCompressionType>
      <!-- req, xs:string,
        "G.711alaw,G.711ulaw,G.726,G.729,G.729a,G.729b,PCM,MP3,AC3,AAC,ADPCM"
      -->
     </audioCompressionType>
    <audioInboundCompressionType>
      <!-- opt, xs:string,
        "G.711alaw,G.711ulaw,G.726,G.729,G.729a,G.729b,PCM,MP3,AC3,AAC,ADPCM"
      -->
     </audioInboundCompressionType>
    <audioBitRate><!-- opt, xs:integer, in kbps -->  </audioBitRate>
    <audioSamplingRate>    <!-- opt, xs:float, in kHz --></audioSamplingRate>
    <audioResolution>  <!-- opt, xs:integer, in bits -->  </audioResolution>
  </Audio>
</dynStreamingChannel>
```

17.4 /PSIA/Custom/SelfExt/ContentMgmt/DynStreaming/channels/<ID>/status

URI	/PSIA/Custom/SelfExt/ContentMgmt/DynStreaming/channels/*ID*/status		Type	Resource
Function	Get the list of streaming sessions associated with a particular channel.			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<DynStreamingSessionStatusList>	
Notes	Use of IPv4 or IPv6 addresses depends on the value of the <ipVersion> field in /PSIA/System/Network/interfaces/*ID*/ipAddress.			

DynStreamingSessionStatus XML Block

```
<dynStreamingSessionStatusList version="1.0" xmlns="urn:selfextension:psiaext-ver10-xsd">
  <StreamingSessionStatus>
    <clientAddress>          <!-- req -->
      <ipAddress>  <!-- dep, xs:string -->  </ipAddress>
      <ipv6Address>  <!-- dep, xs:string -->  </ipv6Address>
    </clientAddress>
    <clientUserName>         <!-- opt, xs:string -->              </clientUserName>
    <startDateTime>          <!-- opt, xs:datetime -->            </startDateTime>
    <elapsedTime>            <!-- opt, xs:integer, seconds -->    </elapsedTime>
    <bandwidth>              <!-- opt, xs:integer, in kbps -->    </bandwidth>
  </StreamingSessionStatu
s>
</dynStreamingSessionStatusList>
```

17.5 /PSIA/Custom/SelfExt/ContentMgmt/DynStreaming/channels/<ID>/http

URI	/PSIA/Custom/SelfExt/ContentMgmt/DynStreaming/channels/*ID*/http		Type	Resource
Function	Access a live stream via http.			
Methods	Query String(s)	Inbound Data	Return Result	
GET	videoCodecType videoScanType videoResolutionWidth videoResolutionHeight videoPositionX videoPositionY videoQualityControlType constantBitRate		Stream over HTTP	

POST	fixedQuality vbrUpperCap vbrLowerCap maxFrameRate keyFrameInterval rotationDegree mirrorEnabled snapShotImageType	<Video>	
Notes	This function is used to request a stream from the device using HTTP or HTTPS. This API uses HTTP server-push with the MIME type multipart/x-mixed-replace. HTTP streaming must be enabled on the channel. To determine the format of the video returned, either the parameters in <Video> or the query string values are used, depending on the capabilities of the encoder.		

	For supported values, query `/PSIA/Streaming/channels/ID/http/capabilities`.

Example

```
GET /PSIA/Streaming/channels/777/http?videoCodecType=MJPEG HTTP/1.1
…
HTTP/1.1 200 OK
Content-Type: multipart/x-mixed-replace; boundary=<boundary>
--<boundary>
Content-Type: image/jpeg
Content-Length: xxx

Image data for a single frame
--<boundary>
…
```

17.6 /PSIA/Custom/SelfExt/ContentMgmt/DynStreaming/channels/<ID>/picture

URI	/PSIA/Custom/SelfExt/ContentMgmt/DynStreaming/channels/*ID*/picture		Type	Resource
Function	Get a snapshot of the current image.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET	videoResolutionWidth videoResolutionHeight videoPositionX videoPositionY rotationDegree mirrorEnabled snapShotImageType		Picture over HTTP	
POST		<Video>		
Notes	All devices must support <snapShotImageType> of "JPEG". To determine the format of the picture returned, either the parameters in <Video> or the query string values are used, or, if the `Accept:` header field is present in the request and the server supports it, the picture is returned in that format. For supported values, query `/PSIA/Streaming/channels/ID/picture/capabilities`. Examples: GET /PSIA/Streaming/channels/123456/picture?snapShotImageType=JPEG POST /PSIA/Streaming/channels/123456/picture … <?xml version="1.0" encoding="UTF-8"?> <Video>…</Video> GET /PSIA/Streaming/channels/123456/picture Accept: image/jpeg			

17.7 /PSIA/Custom/SelfExt/ContentMgmt/DynStreaming/channels/<ID>/requestKeyFrame

URI	/PSIA/Custom/SelfExt/ContentMgmt/DynStreaming/channels/*ID*/requestKeyFrame			Type	Resource
Function	Request that the device issue a key frame on a particular channel.				
Methods	Query String(s)	Inbound Data	Return Result		
PUT					
Notes	The key frame that is issued should include everything necessary to initialize a video decoder, i.e. parameter sets for H.264 or VOS for MPEG-4.				

18 /PSIA/Custom/SelfExt/ContentMgmt/Storage

URI	/PSIA/SelfExt/Custom/ContentMgmt/storage		Type	Service
Function	配置存储设备			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<storage>	
Notes				

storage XML Block

```
<storage version="1.0" xmlns="urn:selfextension:psiaext-ver10-xsd">
  <hddList>  <!-- opt --> </hddList >
  <nasList>  <!-- opt --> </nasList >
  <workMode>  <!-- opt, xs:string, "group, quota, extract" --> <workMode>
</storage>
```

18.1 /PSIA/Custom/SelfExt/ContentMgmt/Storage/hdd

URI	/PSIA/SelfExt/Custom/ContentMgmt/Storage/hdd		Type	Resource
Function	设备硬盘管理			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<hddList>	
Notes				

hddList XML Block

```
<hddList version="1.0" xmlns="urn:selfextension:psiaext-ver10-xsd">
  <hdd>  <!-- opt --> </hdd>
</hddList>
```

18.2 /PSIA/Custom/SelfExt/ContentMgmt/Storage/hdd/<ID>

URI	/PSIA/SelfExt/Custom/ContentMgmt/Storage/hdd/<ID>		Type	Resource
Function	配置某个具体硬盘			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<hdd>	

PUT		<hdd>	<ResponseStatus>
Notes	<property>硬盘属性RW读写，RO只读，Redund 冗余 <group>该硬盘归属于哪个盘组，仅当硬盘工作在盘组模式时有效		

hdd XML Block

```
<hdd version="1.0" xmlns="urn:selfextension:psiaext-ver10-xsd">
  <id>   <!-- ro, req, xs:string;id --></id>
  <hddName>  <!-- ro, req, xs:string -->    </hddName>
  <hddPath>  <!-- ro, opt, xs:string -->    </hddPath>
  <hddType>
    <!-- ro, req, xs:string, "IDE,SATA,eSATA,RAID5", etc -->
  </hddType>
  <status> <!--ro, req, xs:string "ok, unformatted, error, idle, mismatch" --> </status>
  <capacity> <!-- ro, req, xs:float, in MB -->  </capacity>
  <freeSpace><!-- ro, req, xs:float, in MB -->  </freeSpace>
  <property> <!--req, xs:string "RW, RO, Redund"--> </property>
  <group> <!-- opt, xs:string; id --> </group>
</hdd>
```

18.3 /PSIA/Custom/SelfExt/ContentMgmt/Storage/hdd/<ID>/format

URI	/PSIA/SelfExt/Custom/ContentMgmt/Storage/hdd/<ID>/format	Type	Resource
Function	格式化某个硬盘		
Methods	Query String(s)	Inbound Data	Return Result
PUT			<hdd>
Notes	格式化期间，可以通过下面的接口获取格式化进度		

18.4 /PSIA/Custom/SelfExt/ContentMgmt/Storage/hdd/<ID>/formatStatus

URI	/PSIA/SelfExt/Custom/ContentMgmt/Storage/hdd/<ID>/formatStatus	Type	Resource
Function	格式化某个硬盘		
Methods	Query String(s)	Inbound Data	Return Result
GET			<formatStatus>
Notes			

formatStatus XML Block

```
<formatStatus version="1.0" xmlns="urn:selfextension:psiaext-ver10-xsd">
  <formating><!-- ro, req, xs:boolean -->  </formating>
  <percent>  <!-- ro, req, xs:integer "0-100" -->  </percent>
</formatStatus>
```

18.5 /PSIA/Custom/SelfExt/ContentMgmt/Storage/nas

URI	/PSIA/SelfExt/Custom/ContentMgmt/Storage/nas		Type	Resource
Function	网络附属存储设备管理			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<nasList>	
PUT		<nasList>	<ResponseStatus>	
POST		<nas>	<ResponseStatus>	
Notes				

nasList XML Block

```
<nasList version="1.0" xmlns="urn:selfextension:psiaext-ver10-xsd">
  <nas>  <!-- opt --> </nas>
</nasList>
```

18.6 /PSIA/Custom/SelfExt/ContentMgmt/Storage/nas/<ID>

URI	/PSIA/Custom/SelfExt/ContentMgmt/Storage/nas/<*ID*>		Type	Resource
Function	配置指定的网络附属存储设备			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<nas>	
PUT		<nas>	<ResponseStatus>	
DELETE			<ResponseStatus>	
Notes	<nasType>网络盘类型，目前支持NFS，iSCSI。 <property>网络盘属性RW读写，RO只读，RDD冗余 <group>该硬盘归属于哪个盘组，仅当硬盘工作在盘组模式时有效			

NAS XML Block

```
<nas version="1.0" xmlns="urn:selfextension:psiaext-ver10-xsd">
  <id>   <!-- req, xs:string; id -->   </id>
  <addressingFormatType>
    <!-- req, xs:string, "ipaddress,hostname"-->
  </addressingFormatType>
  <hostName> <!-- dep, xs:string -->   </hostName>
  <ipAddress><!-- dep, xs:string -->   </ipAddress>
  <ipv6Address>   <!-- dep, xs:string -->   </ipv6Address>
```

```
  <portNo>   <!-- req, xs:integer -->  </portNo>
  <nasType> <!--req, xs:string, "NFS, iSCSI …" -->  </nasType>
  <path> <!--req, xs:string -->  </path>
  <status> <!--ro, req, xs:string "online, offline" --> </status>
  <capacity> <!-- ro, req, xs:float, in MB -->  </capacity>
  <freeSpace><!-- ro, req, xs:float, in MB -->  </freeSpace>
  <property> <!--req, xs:string "RW, RO, RDD"--> </property>
  <group> <!-- opt, xs:string; id --> </group>
</nas>
```

18.7 /PSIA/Custom/SelfExt/ContentMgmt/Storage/nas/<ID>/format

URI	/PSIA/SelfExt/Custom/ContentMgmt/nas/<ID>/format		Type	Resource
Function	格式化某个网盘			
Methods	Query String(s)	Inbound Data	Return Result	
PUT			<hdd>	
Notes	格式化期间，可以通过下面的接口获取格式化进度			

18.8 /PSIA/Custom/SelfExt/ContentMgmt/Storage/nas/<ID>/formatStatus

URI	/PSIA/SelfExt/Custom/ContentMgmt/Storage/nas/<ID>/formatStatus		Type	Resource
Function	格式化某个硬盘			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<formatStatus>	
Notes				

formatStatus XML Block

```
<formatStatus version="1.0" xmlns="urn:selfextension:psiaext-ver10-xsd">
  <formating><!-- ro, req, xs:boolean -->  </formating>
  <percent>  <!-- ro, req, xs:integer "0-100" -->   </percent>
</formatStatus>
```

18.9 /PSIA/Custom/SelfExt/ContentMgmt/Storage/group

URI	/PSIA/SelfExt/Custom/ContentMgmt/Storage/diskGroup	Type	Resource

Function	盘组管理		
Methods	**Query String(s)**	**Inbound Data**	**Return Result**
GET			<diskGroupList>
Notes			

groupList XML Block

```
<diskGroupList version="1.0" xmlns="urn:selfextension:psiaext-ver10-xsd">
  <diskGroup>     <!-- req --> </diskGroup>
</diskGroupList>
```

18.10 /PSIA/Custom/SelfExt/ContentMgmt/Storage/diskGroup/<ID>

URI	/PSIA/SelfExt/Custom/ContentMgmt/Storage/diskGroup/<ID>	Type	Resource
Function	管理特定的盘组		
Methods	**Query String(s)**	**Inbound Data**	**Return Result**
GET			<diskGroup>
PUT		<diskGroup>	<ResponseStatus>
Notes	trackId see /PSIA/ContentMgmt/record/tracks/<id>		

diskGroup XML Block

```
<diskGroup version="1.0" xmlns="urn:selfextension:psiaext-ver10-xsd">
  <id>   <!-- req, xs:string; id -->    </id>
  <trackList>     <!-- req, xs:boolean -->
    <trackID> <!-- opt, xs:string; id --> <trackID>
  </trackList>
</diskGroup>
```

19 /PSIA/Custom/SelfExt/ContentMgmt/download

URI	/PSIA/Custom/SelfExt/ContentMgmt/download		Type	Resource
Function	Down load a special record segment			
Methods	Query String(s)	Inbound Data	Return Result	
GET		<downloadRequest >	Record data	
PUT		< downloadRequest>	<ResponseStatus>	
Notes	playbackURI is returned by the search service. In the url, there may be some information about the name or size of the segment. For example, rtsp://<host>/Streaming/tracks/<id>?name=track1segment1&size=1024B			

downloadRequest XML Block

```
<downloadRequest version="1.0" xmlns="http://urn:selfextension:psiaext-ver10-xsd">
  <playbackURI> <!—req, xs:string --> </playbackURI>
</downloadRequest>
```

20 Metadata Identity String(MIDS; "metaID")

In order to have a large variety of metadata types, that can be commonly processed, and yet allow flexibility in designing and developing metadata product components, a hierarchical namespace, forming a metadata taxonomy, is employed. This notation is based on a URI structure. The format is:

/<domain>/<class>/<type>[/attribute/LID][/TransID][/…]

Definitions for the above URI fields are:

20.1 MIDS Field Definitions

Field/Name	Requirement Level	Comments
Domain	Mandatory	The 'virtual domain' name of the ordaining body for the ***format and definitions*** that are used for the associated metadata/event information. The domain determines the format, and thus the processing and interpretation, of metadata/event instance data.
Class	Mandatory	Domain-specific 'Class' of the metadata/event information. Some examples are: "VideoMotion", "AccessCtl", "PtOfSale", "Intrusion", "VideoAnalytics", etc.
Type	Mandatory	Class-dependent type of metadata/event information. For example, within a class called "VideoMotion" there would be types such as: "motion", "motionStart", "motionStop", "zoneActive", "zoneInactive", etc.
Attribute/LID ('Local ID')	Dependent / Optional	Free-form field that is available for use as additional descriptive information using the following rules: > The convention is that this field MUST be used as the 'Local ID' field for all metadata/event occurrences that are related to, or associated with, a channel/port/stream ID (i.e. the 'source local ID'; see Section 7.1). > For metadata/event occurrences that have no correlation to a channel or port (etc.), this field is optional.
TransID (Transaction ID)	Optional	A string field that uniquely identifies this occurrence instance to the source. If a source entity requires a transactional level acknowledgement, then this field MAY be used as an identifier for expressly acknowledging a specific metadata/event instance. Please note that the source UUID/GUID and timestamp of a metadata/event instance are the standard fields used for uniqueness. Additional fields are optional.

In this hierarchical namespace scheme, the Domain, Class and Type fields are REQUIRED. The Attribute/LID and TransID fields are optional. To provide consistent parsing and decoding, the above described fields are 'positonal' within an MIDS URI. Empty slots after the Domain/Class/Type need not be present. Intervening slots that are empty (e.g. an ID field is present but there is no attribute/LID field) are note by adjacent 'slashes' ("//"). The following example depicts an 'empty' URI Attribute/LID field:

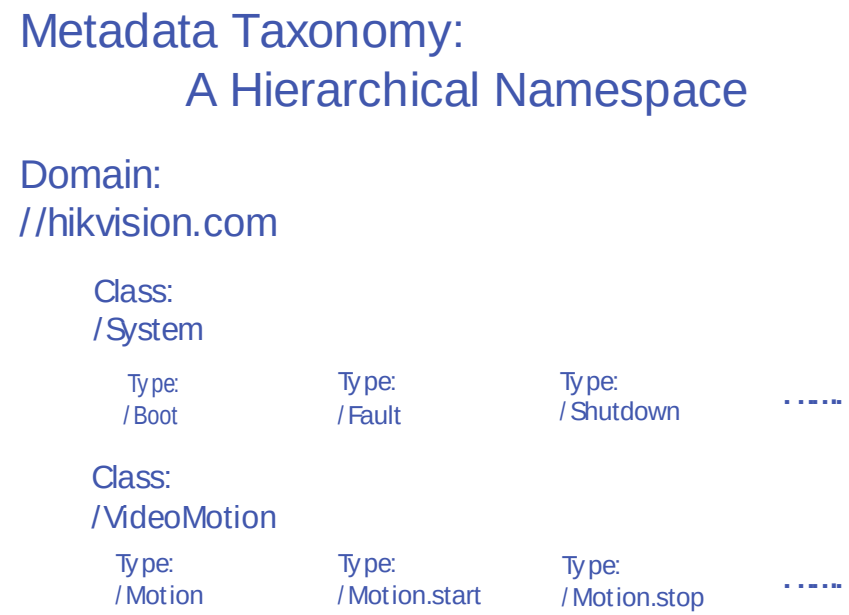
```
/hikvision.com/VideoMotion/motion//C1EB2D39
```

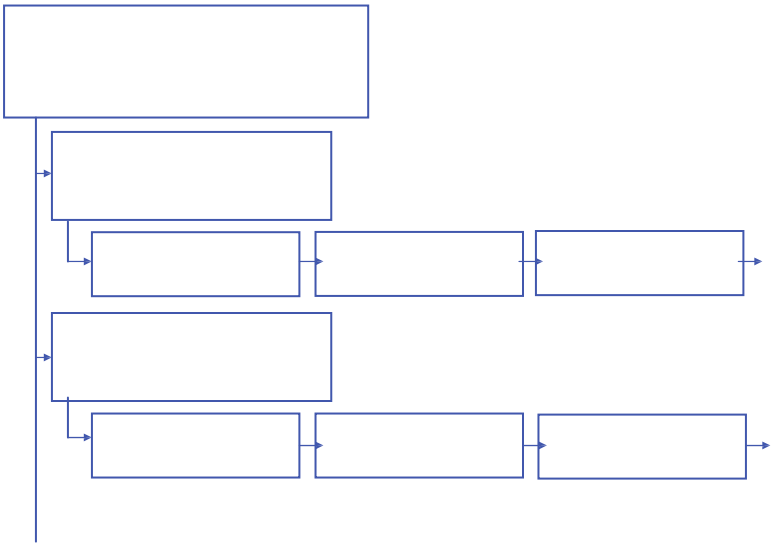
In the above example, a hypothetical intrusion alarm carries a TransID field ("C1EB2D39") in its MIDS, but no attribute/LID field. As such, the empty attribute/LID field is noted by the adjacent slashes ("//") after the type field of "motion."

Other information may be appended to the end of an MIDS, as needed (though it is not encouraged). Any appended, after the ID field, is ignored by the common processing code and considered instance or manufacturer specific.

The figure below depicts the relationship between domains, classes, and types.

Figure 18.2.1: Domain/Class/Type Hierarchy





The aforementioned taxonomy enables a vast amount of flexibility in the definition of numerous classes, types, and versions, of metadata information while avoiding 'collisions' among metadata publishers. It also advantageously lends itself to subscribing, filtering, and forwarding logic since it is hierarchical in nature with ordinally positioned fields. Additionally, the MIDS design follows well known URI definitions in a REST-like manner, while providing a level of user friendliness via its self-declaring structure. A final benefit is that this structure can be optimized for extremely fast 'look-ups'. The term metadata 'category' covers a specific 'domain/class' pair.

20.1.1 Domain:event.hikvision.com

Domain:event.hikvison.com 定义如下表：

Class	Type	Attribute/LID ('Local ID')	TransID (Transaction ID)
VideoMotion	motion	video input port/dynamical video input port	
	motionStart		
	motionStop		
Intrusion	alarmIn	alarm In port	

20.1.2 Domain:log.hikvision.com

Domain:log.hikvison.com 定义如下表：

Class	Type	Attribute/LID ('Local ID')	TransID (Transaction ID)
Alarm	alarmIn	alarm in port	
	alarmOut	alarm out port	
	motionStart	video input port	
	motionStop	video input port	
	hideStart	video input port	
	hideStop	video input port	
	vcaStart	video input port	
	vcaStop	video input port	
Exception	videoLost	video input port	
	videoException	video input port	
	videoFormatMismatch	video input port	
	illlegealAccess		
	hdError		
	hdFull		
	netBroken		
	recordError	video input port	
	ipcDisconnect	video input port	
	ipcIpConfilict	video input port	
	ipConflict		
Operation	devicePowerOn		
	devicePowerOff		

	deviceRecycle		
	stopAbnormal		
	localLogin		
	localLogOut		
	localCfgPara		
	localUpdate		
	localStartRec		
	localStopRec		
	localCtrlPtz		
	localLockFile		
	localUnlockFile		
	localManulAlarm		
	localFormatDisk		
	localAddIpc		
	localDelIpc		
	localSetIpc		
	localPlayByFile		
	localPlayByTime		
	localDownloadCfgFile		
	localUploadCfgFile		
	localDownloadRecFile		
	localDownloadPicFile		
	localAddNas		
	localDelNas		
	localSetNas		
	localConfRebRaid		
	localConfSpareRaid		
	localAddRaid		
	locaDelRaid		
	localMigRaid		
	localRebRaid		
	localQuickConfRaid		
	localAddVd		
	localDelVd		
	localStartPicRec		
	localStopPicRec		
	localSetSnmp		
	localResetPasswd		
	localTagOperation		
	remotePowerOff		
	remotePowerRecycle		
	remoteLogin		
	remoteLogout		
	remoteCfgPara		
	remoteUpgrade		
	remoteStartRec		

	remoteStopRec		
	remoteCtrlPtz		
	remoteLockFile		
	remoteUnlockFile		
	remoteManulAlarm		
	remoteFormatHd		
	remoteAddIpc		
	remoteDelIpc		
	remoteSetIpc		
	remotePlayByFile		
	remotePlayByTime		
	remoteDownloadCfgFile		
	remoteUploadCfgFile		
	remoteDownloadRecFile		
	remoteGetPara		
	remoteGetStatus		
	remoteStartTransChan		
	remoteStopTransChan		
	startVoiceTalk		
	stopVoiceTalk		
	remoteArm		
	remoteDisArm		
	remoteAddNas		
	remoteDelNas		
	remoteSetNas		
	remoteConfRebRaid		
	remoteConfSpareRaid		
	remoteAddRaid		
	remoteDelRaid		
	remoteMigRaid		
	remoteRebRaid		
	remoteQuickConfRaid		
	remoteAddVd		
	remoteDelVd		
	remoteRpVd		
	remoteUpgradeRaid		
	remoteStartPicRec		
	remoteStopPicRec		
	remotePicBackUp		
	remoteSetSnmp		
	remoteTagOperation		
Infomation	hddInfo		
	smartInfo		
	startRec		
	stopRec		
	delExpiredRec		

	nasInfo		
	raidInfo		

21 /PSIA/Custom/SelfExt/ContentMgmt/logSearch

本服务定义了海康威视扩展的日志搜索协议。本服务借鉴了RaCM协议serach的实现方式，除了请求及相应的xml内容不同外，整个日志搜索机制与RaCM中的search服务相同。

URI	/PSIA/Custom/SelfExt/ContentMgmt/logSearch/		**Type**	Service
Requirement Level	- All Profiles -			
Function	Mandatory description of the REST method parameters and formats available to functionally manipulate the 'logSearch' resource/object.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET	None	<CMSearchDescription>	<CMSearchResult or…>	
PUT	N/A	N/A	<ResponseStatus w/error code>	
POST	None	<CMSearchDescription>	<CMSearchResult or…ResponseStatus w/error code>	
DELETE	N/A	N/A	<ResponseStatus w/error code>	
Notes	The 'GET' or 'POST' messages require a "CMSearchDescription" XML document to engage a search. An example XML document instance follows.			
Example(s)	<?xml version="1.0" encoding="UTF-8"?> <CMSearchDescription version="1.0" xmlns="urn:psialliance-org"> <searchID>{812F04E0-4089-11A3-9A0C-0305E82C2906}</searchID> <timeSpanList> <timeSpan> <startTime>2009-06-10T12:00:00Z</startTime> <endTIme>2009-06-10T13:30:00Z</endTime> <timeSpan> </timeSpanList> <metadataList> <metadataDescriptor> <metaID>/event.hikvision.com/VideoMotion</metaID> </metadataDescriptor> <metadataList> <searchResultPosition> 20 </searchResultPosition> <maxResults> 40 </maxResults> </CMSearchDescription>			

```
<?xml version="1.0" encoding="UTF-8"?>
<CMSearchDescription version="1.0" xmlns="urn:psialliance-org">
   <searchID>{77E105E8-4C21-AA05-37B4-189A070A5B22}</searchID>
   <sourceID> {3F2504E0-4F89-11D3-9A0C-0305E82C3301} </sourceID>
   <timeSpanList>
      <timeSpan>
         <startTime>2009-07-12T09:00:00Z</startTime>
         <endTIme>2009-07-12T17:30:00Z</endTime>
      <timeSpan>
   </timeSpanList>
   <contentTypeList>
      <contentType>video</contentType>
      <contentType>audio</contentType>
   </contentTypeList>
   <maxResults>40</maxResults>
   <metadataList>
      <metadataDescriptor>
         <metaID>/event.hikvision.com/VideoMotion</metaID>
      </metadataDescriptor>
   <metadataList>
</CMSearchDescription>
```

The above example is a search for Video Motion, betw een the hours of 9AM and 5:30PM, on July the 12th, w ith respect to a specific source whose ID (GUID) is: 3F2504E0-4F89-11D3-9A0C-0305E82C3301. This example GUID corresponds to an input source w hich can be resolved to an IP address. Please note that in this example, the search requester only wants video and audio segments returned for matches (contenTypeList).

```
<?xml version="1.0" encoding="UTF-8"?>
<CMSearchDescription version="1.0" xmlns="urn:psialliance-org">
   <searchID>{F44EC031-4F89-3D90-9A14-0305E828D902}</searchID>
   <metadataList>
      <metadataDescriptor>
         <metaID>/event.hikvision.com/VideoMotion</metaID>
      </metadataDescriptor>
   <metadataList>
</CMSearchDescription>
```

The above example is a simple search for all log.

```
<?xml version="1.0" encoding="UTF-8"?>
<CMSearchDescription version="1.0" xmlns="urn:psialliance-org"
   <searchID>{812F04E0-4089-11A3-9A0C-0305E82C2906}</searchID>
   <timeSpanList>
      <timeSpan>
         <startTime>2009-06-10T12:00:00Z</startTime>
         <endTIme>2009-06-10T13:30:00Z</endTime>
      <timeSpan>
   </timeSpanList>
   <metadataList>
      <metadataDescriptor>
        <metaID>/event.hikvision.com/PointOfSale</metaID>
      </metadataDescriptor>
   <metadataList>
   <searchText>”VOID”</searchText>
</CMSearchDescription>
```

22 /PSIA/Custom/SelfExt/Bond

URI	/PSIA/Custom/SelfExt/Bond		Type	Service
Function	Get or set the configuration information of Bond net interfaces.			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<BondList>	
Notes	Bond网卡配置			

BondList XML Block

```
<BondList version="1.0" xmlns="urn:selfextension:psiaext-ver10-xsd">
  <Bond>
</BondList>
```

22.1 /PSIA/Custom/SelfExt/Bond/<ID>

URI	/PSIA/Custom/SelfExt/Bond/*ID*		Type	Resource
Function	Get or set the configuration information of Bond net interface			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<Bond>	
PUT		<Bond>	<ResponseStatus>	
Notes				

Bond XML Block

```
<Bond version="1.0" xmlns="urn:selfextension:psiaext-ver10-xsd">
  <id>        <!-- req, xs:string --> </id>
  <enabled>   <!-- req, xs:boolean --> </enabled>
  <workMode> <!-- req, xs:string;"balance-rr, active-backup" --> </workMode>
  <primaryIf> <!-- req, xs:string;id --></primaryIf>
  <slaveIfList> <!-- req -->
    <ethernetIfId>   <!-- req, xs:string; id -->    </ethernetIfId>
  </slaveIfList>
  <IPAddress>
    <ipVersion>          <!-- req, xs:string, "v4,v6,dual" --></ipVersion>
    <addressingType>     <!-- req, xs:string, "static,dynamic,apipa" --> </addressingType>
    <ipAddress>          <!-- dep, xs:string -->        </ipAddress>
    <subnetMask>   <!-- dep, xs:string, subnet mask for IPv4 address -->    </subnetMask>
    <ipv6Address>        <!-- dep, xs:string -->       </ipv6Address>
    <bitMask>            <!-- dep, xs:integer, bitmask IPv6 address --> </bitMask>
    <DefaultGateway>     <!-- dep -->
      <ipAddress>        <!-- dep, xs:string -->       </ipAddress>
```

```
      <ipv6Address>    <!-- dep, xs:string -->          </ipv6Address>
    </DefaultGateway>
    <PrimaryDNS>          <!-- dep -->
      <ipAddress>         <!-- dep, xs:string -->       </ipAddress>
      <ipv6Address>    <!-- dep, xs:string -->          </ipv6Address>
    </PrimaryDNS>
    <SecondaryDNS>        <!-- dep -->
      <ipAddress>         <!-- dep, xs:string -->       </ipAddress>
      <ipv6Address>     <!-- dep, xs:string -->         </ipv6Address>
    </SecondaryDNS>
  </IPAddress>
  <Link xmlns="urn:selfextension:psiaext-ver10-xsd">      <!-- opt -->
    <MACAddress> <!-- req, xs:string> </MACAddress>
    <autoNegotiation> <!-- req, xs:boolean> </autoNegotiation>
    <speed> <!-- req, xs:integer, "10, 100, 1000" --><speed>
    <duplex> <!-- req, xs:string, "half, full"> </duplex>
    <MTU> <!-- req, xs:integer --> </MTU>
  </Link>
</Bond>
```

23 /PSIA/Custom/SelfExt/Holiday

URI	/PSIA/Custom/SelfExt/Holiday		Type	Resource
Function	Access the list of PTZ configuration.			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<holidayList >	
PUT		<holidayList>	<ResponseStatus>	
Notes				

holidayList XML Block

```
<HolidayList version="1.0" xmlns="http://urn:selfextension:psiaext-ver10-xsd">
  <holiday/> <!-- opt -->
</HolidayList>
```

23.1 /PSIA/Custom/SelfExt/Holiday/ID

URI	/PSIA/Custom/SelfExt/Holiday/ID/		Type	Resource
Function	Access or control a ptz channel.			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<holiday >	
PUT		<holiday>	<ResponseStatus>	
Notes	是否含有复合元素<holidayDate>、<holidayWeek>及<holidayMonth>由<holidayMode>决定 <holidayMode>:date: 假期从某年某月某日到某年某月某日 <holidayMode>:week: 假期从某月的第几个周几到某月的第几个周几 <holidayMode>:month: 假期从某月某日到某月某日			

holiday XML Block

```
<holiday version="1.0" xmlns="http://urn:selfextension:psiaext-ver10-xsd">
  <id>                  <!-- req, xs:string;id -->                  </id>
  <enabled>             <!-- req, xs:boolean -->                    </enabled>
  <holidayMode>       <!-- req, xs:string, "date, weeek, month" -->  </holidayMode>
  <holidayName>       <!-- req, xs:string -->  </holidayName>
  <holidayDate>  <!-- dep -->
   <startDate> <!-- dep, xs:date --> </startDate>
   <endDate>   <!-- req, xs:date --> </endDate>
  </holidayDate>
```

```
  <holidayWeek>  <!-- dep -->
   <startWeek>   <!-- req -->
     <monthOfYear> <!-- req --> </monthOfYear>
     <sequence> <!-- req, xs:integer, 1…5 --> </sequence>
     <dayOfWeek>
       <!-- req, ISO8601 weekday number, 1=Monday" -->
     </dayOfWeek>
   </startWeek>
   <endWeek>   <!-- req -->
     <monthOfYear> <!-- req --> </monthOfYear>
     <sequence> <!-- req, xs:integer, 1…5 --> </sequence>
     <dayOfWeek>
         <!-- req, ISO8601 weekday number, 1=Monday" -->
     </dayOfWeek>
   </endWeek>
  </holidayWeek>
  <holidayMonth>  <!-- dep -->
   <startMonth>   <!-- req -->
    <monthOfYear> <!-- req, xs:integer, "1…12" --> </monthOfYear>
    <dayOfMonth> <!-- req, xs:integer, "1…31" --> </dayOfMonth>
   </startMonth>
   <endMonth>   <!-- req -->
    <monthOfYear> <!-- req, xs:integer, "1…12" --> </monthOfYear>
    <dayOfMonth> <!-- req, xs:integer, "1…31" -->  <dayOfMonth>
   </endMonth>
  </holidayMonth>
</holiday>
```

24 Appendix A: Codec Type Dictionary

The ‘Codec Tag’s below represent the literal ASCII strings employed to identify the specific codec standards listed below.

Codec Tag (Literal)	Codec Tag Description
Audio Codecs	
G.711	ITU G.711 (PCM) audio codec format (a-/u-law determined by SDP or XML
G.711a	ITU G.711(PCM) audio codec format; a-law encoding
G.711u	ITU G.711 (PCM) audio codec format; u-law encoding
G.726	ITU G.726 (ADPCM)audio codec format (bitrate advertised by SDP or XML)
G.723.1	ITU G.723.1 audio codec format
G.728	ITU G.728 audio codec format
G.722, G.722.1, G.722.2	ITU G.722/.1/.2 audio codec formats (SB-ADPCM)

G.728 ITU G.728 (LD-CELP) audio codec format G.729, G.729.1 ITU G.729/.1 (CS-ACELP) audio codec format MP3 MPEG-1/Layer 3 audio codec format
AAC MPEG-2/4 Advanced Audio Codec format

Video Codecs

MPEG4-SP ISO/IEC 14496-2 MPEG-4 Simple Profile
MPEG4-ASP ISO/IEC 14496-2 MPEG-4 Advanced Simple Profile
MPEG4-MP ISO/IEC 14496-2 MPEG-4 Main Profile
H.264-BP ISO/IEC 14496-10/ITU H.264 Baseline Profile H.264-MP ISO/IEC 14496-10/ITU H.264 Main Profile H.264-HP ISO/IEC 14496-10/ITU H.264 High Profile
H.264SVC-BP ISO/IEC 14496-10/ITU H.264 Scalable Video Codec (SVC), Baseline Profile encoding (Must read s-props/p-props and SDP for embedded stream info)
H.264SVC-MP ISO/IEC 14496-10/ITU H.264 Scalable Video Codec (SVC), Main Profile encoding (Must read s-props/p-props and SDP for embedded stream info)
MPEG2-MP ISO/IEC 13818 MPEG-2 Main Profile
MJPEG Motion version (multi-frame) of ISO/IEC JPEG video encoding (see below) JPEG ISO/IEC 10918 JPEG video encoding JPEG2000 ISO/IEC 1544