Cisco Virtual Media Packager Release 2.9 API Guide

First Published: June 2014
Last Updated: April 2017

Cisco Systems, Inc.
www.cisco.com

Cisco has more than 200 offices worldwide.
Addresses, phone numbers, and fax numbers
are listed on the Cisco website at
www.cisco.com/go/offices.

Table of Contents

Table of Contents

Table of Figures

Table of Tables

1 Purpose

1.1 Scope

This document provides a basic overview of the Media Origination System and describes the API used to administer and control the system.

2 Functional Overview

2.1 System Overview

The Cisco Virtual Media Packager provides cloud based Media Ingest, packaging, storage and origination functions for Multi-Screen Video Delivery. The system takes input from the media Encoding systems and produces output that can be consumed and distributed by the CDN to various client devices.

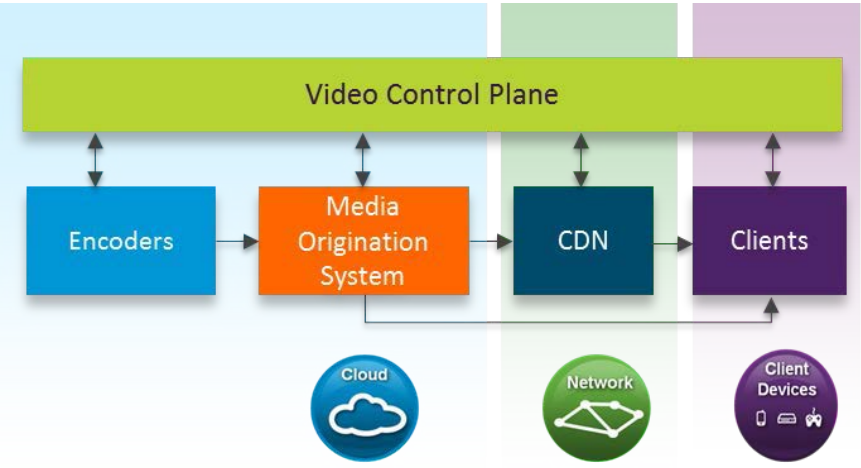
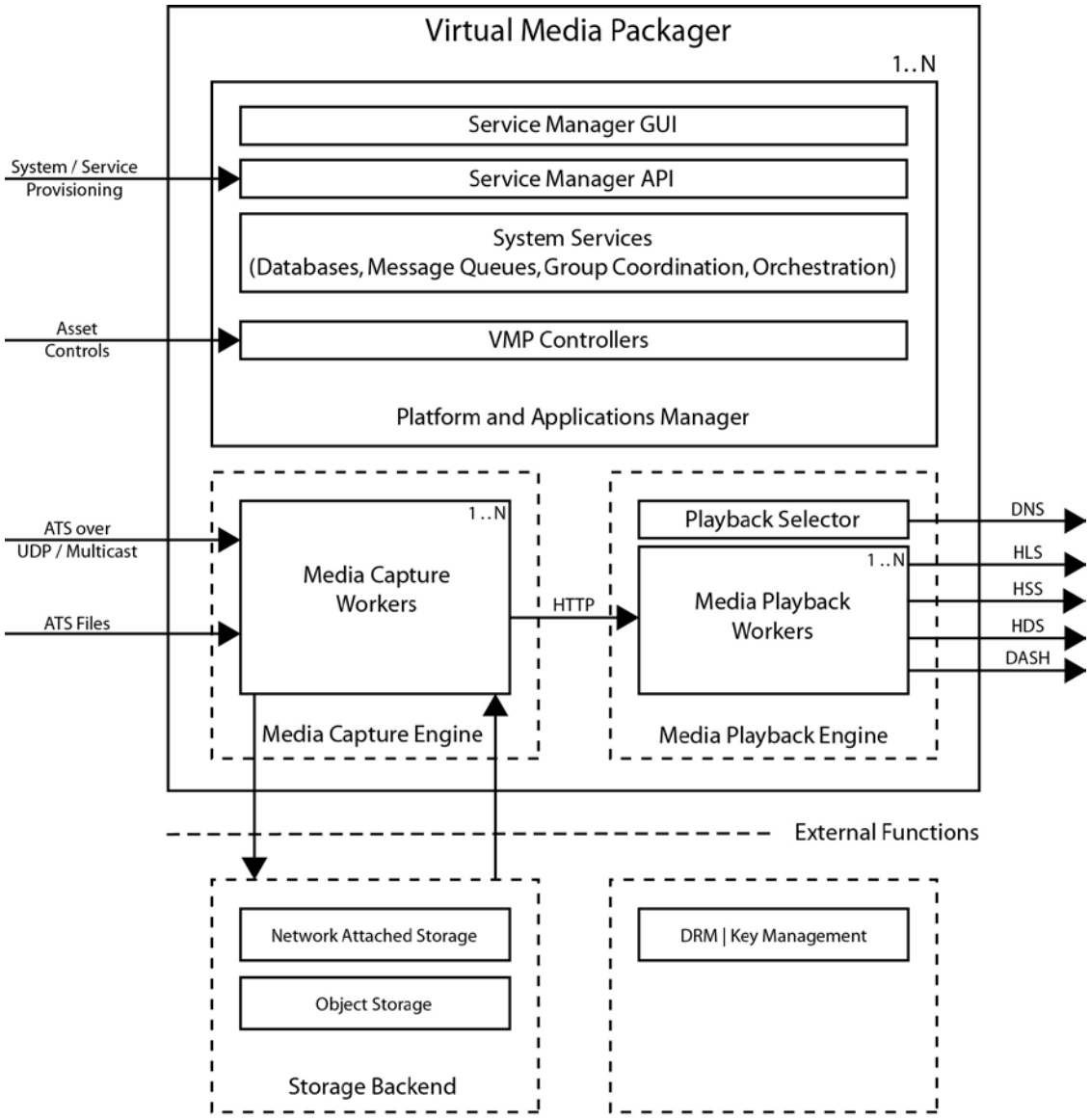


Figure 1 Media Origination and end to end IP video distribution

The Media Origination System can be instantiated in data-center environments. The System is built as a collection of well-defined applications that work together to deliver services that can address various use cases such as Linear TV, VOD and Cloud DVR. Each VMP application is implemented as a modern distributed application with horizontal scaling and automated provisioning capabilities. Based on the service definition, the composition of the applications within the service can be altered, providing flexibility for the operators to pick and choose the required capabilities. Figure 2 provides the block diagram of the media origination system. The Media capture and the playback engines can be instantiated multiple times within the system based on the number of services instantiated.

VMP supports Live, VOD, and Cloud DVR Service Workflows.



Virtual Media Packager

Figure 2

S.No	Component	Description	Implementation
1	VMP Platform and Applications Manager (PAM)	The Platform and Applications Manager provides bootstrapping functions for the VMP system. It abstracts the data-center virtualization platforms from the application and creates the launch pad to launch the full VMP System. It provides a container to host many of the common system level components such as data bases, group coordination SW, Message Queues, orchestration tools, Dynamic DNS. Hosts the major management and control functions such as the Service Manager, the Task Controllers and Content Managers.	Implemented as a collection of virtual machines based on the HA Requirements. Odd number of virtual machines is required for fault tolerance.
2	VMP Service Manager (SM)	The VMP Service Manager provides the API and the GUI to administer the VMP system and services. The VMP Service Manager executes in the same Virtual Machine as that of the Platform and Applications Manager.	Hosted by the PAM Virtual Machines
3	Media Capture Engine (MCE)	The Media Capture Engine acquires media from various sources and packages the media in to common format and stores the package in to the configured storage backend. The Common format package includes the original media, meta-data, indexing. The media file format should be of Adaptive Transport stream type and the system does not alter the source media files.	The Media Capture Engine is implemented as a set of virtual machines (also called Capture Workers) The Capture Sessions within the MCE are controlled by the controllers hosted in the PAM.

4	Media Playback Engine (MPE)	The Media Playback Engine is the VMP just-in-time packager (JITP), used for Live and VOD video playout. The MPE processes incoming requests for manifests, segments, fragments, and other related resources, and plays recorded content to multi-screen IP-connected ABR endpoints.	Implemented as a set of virtual machines (also called Playback Workers)
5	Media Playback Engine (MPE)	The Media Playback engine provides Just-in- Time packaging capabilities to re-package the common format content in to various target formats for end-client consumption across devices. The Media Playback engine also applies the required content protection to the target format contents and integrates with various DRM ecosystems.	Implemented as a set of virtual machines (also called Playback Workers)
6	Centralized Logging System (CLS)	The CLS provides a central location for the aggregation, analysis, and display of VMP error logs and events.	Deploy the CLS from vCenter.
7	App Engine	The app engine nodes serve as the IPVS node for MCE/NGMPE to implement load balance.	A minimum of 6 app engines (2 Redis, 2 HAProxy, and 2 IPVS) are required to enable Media Service.

Table 1 VMP Components

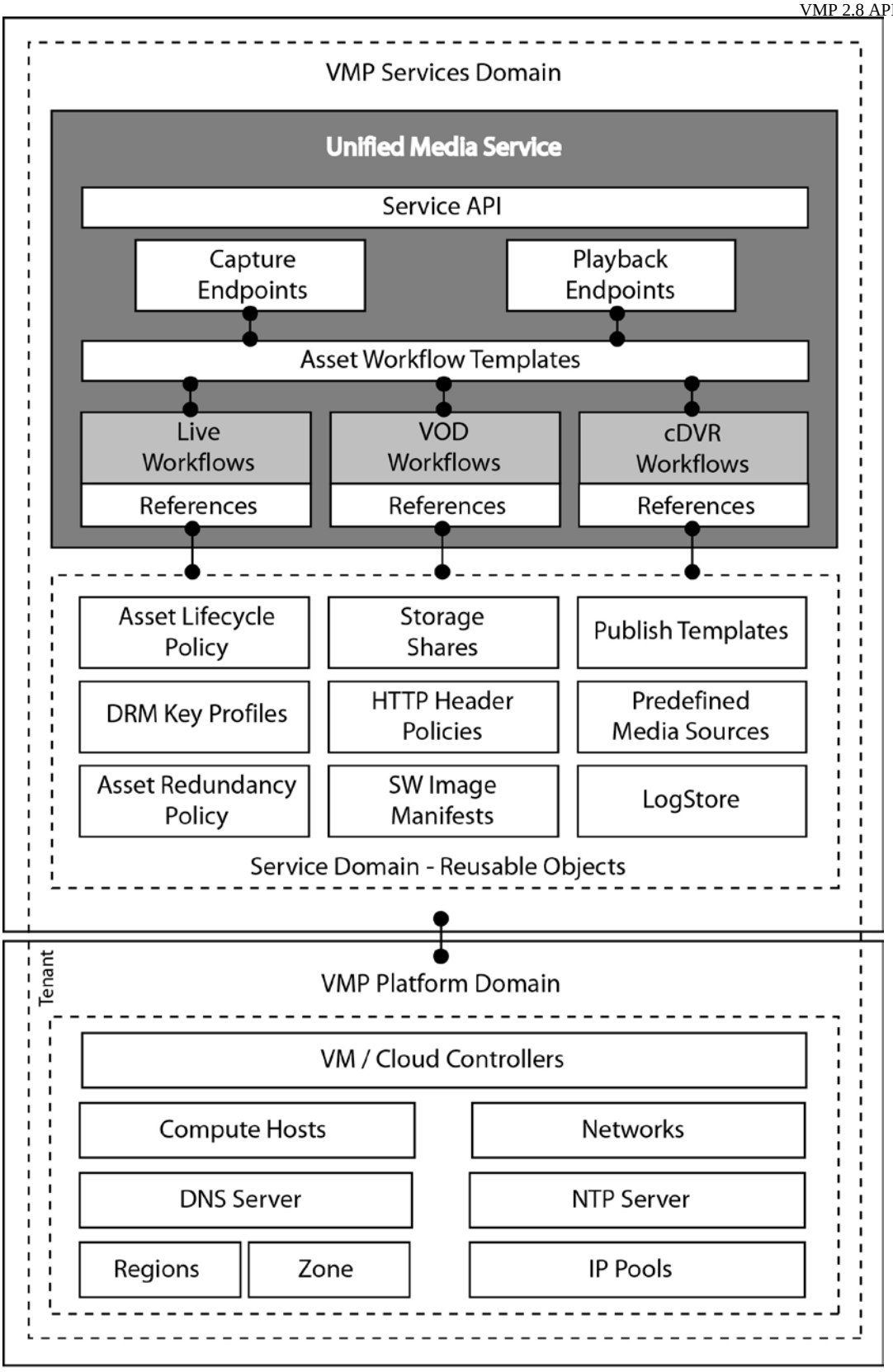
2.2 Administrative Domains & Configuration Objects

The VMP contains two administrative Domains. Each of these domains can support multiple administrative roles and constraints within the roles.

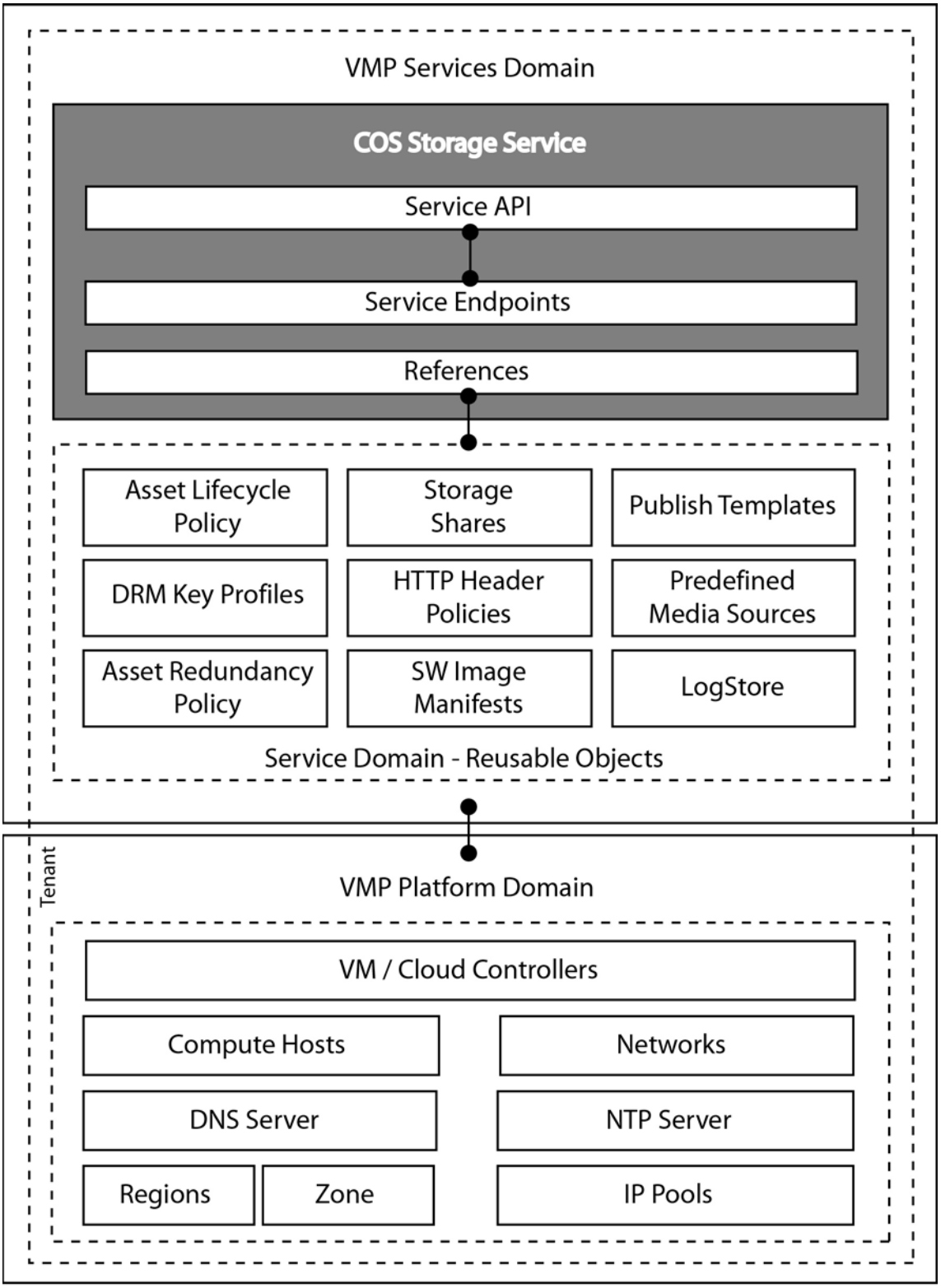
Figure 3 provides an overview of the administrative domains and the managed objects that belong to different domains.

Platform Domain – The Platform domain provides the run-time environment for various services to function. There is utVMPt one Platform domain. The infrastructure resources such as compute, storage, and other system wide policies are configured in the Platform domain. The Platform operator is in-charge of administration for this domain.
The Platform Domain objects provide interfaces for which the platform operator can expose the physical infrastructure provisioned for the VMP software components to function.

Services Domain – The Services Domain provides a logical container to host media services and Object Storage Services. The Service Operator is in-charge of the administration of this domain. Service Domain Objects makes use of the VMP platform and the infrastructure to install software components that are required for the functioning of the media origination services.


VMP Services Domain
Unified Media Service
Service API
Capture Endpoints
Playback Endpoints
Asset Workflow Templates
Live Workflows
VOD Workflows
cDVR Workflows
References
References
References
Asset Lifecycle Policy
Storage Shares
Publish Templates
DRM Key Profiles
HTTP Header Policies
Predefined Media Sources
Asset Redundancy Policy
SW Image Manifests
LogStore
Service Domain - Reusable Objects
Tenant
VMP Platform Domain
VM / Cloud Controllers
Compute Hosts
Networks
DNS Server
NTP Server
Regions
Zone
IP Pools

VMP Domains



VMP Administrative Domains

Figure 3

2.3 VMP API Terminology

VMP Service	A VMP Service is a collection of resources, policies and templates used to implement media ingest, control, storage and playback for multi-screen use cases such as cDVR, VOD, TSTV, Live, or Cisco Object Store (COS).
Tenant	A tenant is a Domain, Service Provider or Business Entity, which manages, monitors, and offers VMP Services. In a VMP model, the Tenant may be a different business entity than the VMP Product owner and Platform provider. Operational Data within VMP are never shared across Tenants.
VMP Service Instance	An instantiation of a VMP Service. Multiple Service instances may co-exist within tenants. Each Service Instance is independently managed, monitored and scaled.
VMP Service Template	A Service Template is a predefined set of definitions for creating variances of a VMP Service. A Template specifies the boundaries of the features and capabilities for a Service Instance. A Service Instance is created by instantiating and customizing a Service Template. Service Templates are not field-customizable.
Region	A Region comprises of one or more Zones. It is an abstraction of the underlying cloud platform and may be associated with a geographical region, datacenter(s), or a service area.
Zone	A Zone is a set of Cloud platform components (compute, network, storage, security) that are fate-shared. This can be mapped to the underlying cloud platform provider, such as a datacenter in *vCenter*, an Availability Zone in *AWS*, or any other combination of cloud resource topologies that are to be treated as fate-shared. Each Zone is associated with one Cloud Controller.
Cloud Controller	A Cloud Controller is a processing entity that manages the cloud platform components - compute hosts, networking components, storage and security. Example: *vCenter*, *OpenStack* controller, *AWS*. APIs specific to the technology it supports, in order to manage the resources it can provide, may be used to access the Cloud Controller.
Application	An Application is a set of code objects that provides a well-defined function/capability. Examples: Media Capture Media Playback, Media Storage, and Workflow Manager.

Application Instance Controller	An Application Instance Controller is a set of processes instantiated in one or more VMs used to manage Application Instance and provide interfaces for interworking with other Application Instances, Configuration, Monitoring. It also manages the life cycle of its Application Instance components by instantiating as many VMs that are allowed and required by the policies for a Service Instance.
Application Task Controller	Application Task Controller manages the life cycle of the tasks for an application instance, and may interwork with a peer Application Task Controller to deliver High Availability of the function in a Service Instance scope.
Application Worker	An Application Worker is a run time machine instantiation (VM or appliance) that provides one or more sub functions that are required to complete the function(s) delivered by the Application Instance. Application Workers can be clustered for scale out.
Application Instance	An Application Instance is a run time instantiation of an Application, in a Region. The Application Instance may support delivery of a function for one or more Service Instances belonging to one Tenant.

Application Image manifest	An Application image manifest is a set of OVA software images, scripts and RPMs that are used to instantiate a particular version of the Application. The images in a set are tested as a test unit. The released software image manifests are named and used to specify the software used for a Service Instance.
Node	A VMP Node is a VM or an appliance that is managed and monitored within VMP. An Application Controller, Worker, SM and PAM are VMP Nodes. COS nodes (CDE based appliances) are VMP nodes as well. External hardware/machines are not VMP Nodes when they are only used by VMP components (example: LDAP servers, Storage Servers, external load balancers etc.)
Service Flow	A Service Flow is a specific binding of end-to-end application processes, sockets, ports and network interfaces within a Service Instance. It produces one output workflow. It can be monitored and measured. A Service Instance has zero or more Service Flows at any time. Example, a recording in progress is a workflow and has a Service Flow binding of the Media Capture Worker process/ports, Storage Worker interface and some Workflow Manager endpoints. Similarly, a Live playback session using memory storage has a Service Flow involving particular Media Capture worker process/port, Media playback worker process/port and optionally a Software Load balancer path.
Service Path	A Service Path is a specific combination of the run time machines and processes used to instantiate one or more Service Flows for a Service Instance. More than one Application Worker in a cluster may belong to a Service Path. In effect, it is the end- to-end plumbing of VMP Nodes used to deliver a set of Service Flows.
Service Management APIs	HTTP RESTful APIs to manage Services and VMP product. VMP Service Manager hosts these APIs. These are used to create/manage Service (for example, to create a CDVR service instance), discover available Service Instances, discovery Service Use APIs for a Service Instance, manage policies and resources, monitor Service Instances, and administer
Service Use APIs	Service Use APIs are a set of APIs that can be used to manage and control a Service Flow and discover the assets in VMP. Examples are Recording Management Interface for a CDVR Service Instance, VOD asset discovery for ingested assets for a VOD Service Instance. The Service Use APIs are discoverable via Service Management Interface and are specific to the Service Instances.

Platform Operator	The Platform Operator is the Cloud Platform, Resource and Product owner and operator. The role of the Platform Operator is to manage topology, platform resources, and create/manage Tenants.
Resource	A Resource is an item of use by the VMP Service Instances. It could be a defined Region, Zone, Media Source definition, Channel Lineup, External Storage, and External Services. The Resources are named. They are not managed by VMP. Resources may be owned by Platform Operator or by Tenant (only some valid combinations). Platform Operator may allow use
Policy	A Policy is a rule set or constraint that is used by Application instances to control the delivery of a SLA for a Service Instance. Example, there can be Content Redundancy Policies (unique/Common copy), Content Protection Policy (what DRM to apply), HA Policy for Geo redundancy. Tenant owns a Policy. A Policy affects how the Applications are instantiated, Resources used and Assets' lifecycle.

2.4 VMP API Classification

VMP offers different sets of APIs to control the platform services and the asset ingest process. The VMP APIs can be classified into the following categories.

API Classification	Description	Configuration Objects
System Provisioning API	These are APIs provided to provision some of the basic information required to setup the VMP Platform. This information is essential to provision services and manage the asset ingest process in the system. The System Provisioning API is also used to create logical containers (Service domains) that can have independent administrative control over the service instances.	Region Zone DNSServ er DNSForward er DNSZoneMa p NTPServer Node[1] NASStore Tenant (internal use only in this release)

[1] Node object is supported in VMP version 2.2 as a means to define the manually created VMs and inserting them for use in the VMP operations.

Service Management API	Service Management APIs are applied with the Service Domain. These APIs are used to create Service Instances based on Pre-defined Service Templates. Service Instance construct is used to bind applications and their configuration to implement well defined services workflows such as Linear, VOD and Cloud DVR. Based on the constraints imposed by the service templates, the configuration of the applications created for the service will be modified.	*Reusable Objects:* • *ImageManifest* • *KeyProfile* • *HTTPHeaderPolicy* • *ABRAssetPublishTemplate* • *ESAMProfile* • *NASStore* • *AssetLifeCyclePolicy* • *AssetRedundancyPolicy* • *ChannelSource* • *NASMediaSource* • *DynamicSource* • *ChannelLineup* • *DynamicLineup* • *StreamProfile* • *AuthProfile* • *SwiftStore* • *LogStore*[4] *Service Instance specific objects:* • *ServiceInstance* • *PlaybackEndpoint (with resource and capacity*[5] *SLA)* • *CaptureEndpoint (with resource and capacity SLA)* • *AssetWorkflowTemplate* • *ServiceAPI* *Monitoring objects:* • *NodeStatus* • *AppStatus* • *Event* • *Diagnostics* • *Statistics*[6]

[4] Not supported in current release.
[5] Capacity SLA is not supported in this release for Playback and Capture Endpoint.

Service Control API	The Service Control APIs are invoked at a service instance level to orchestrate ingest and packaging of media for the use-cases defined by the service template.	*MediaAsset API*

Table 2 VMP - API Classification

2.5 VMP API Methodology

VMP APIs use the REST methodology to manipulate and operate the state of the system and the services. For each API classification, a set of configuration objects is defined. The configuration objects have a well-defined structure based on JSON Schema. Configuration Objects can be further classified in to re-usable objects such as Policies, Resources, and Templates and service instance specific objects such as capture end points, playback end points etc.

Common Objects such as Policies, Resources and templates can be created once and de-referenced across various other containers to help reduce the administrative overhead in repeating the configuration. The User of the VMP API should create these re-usable objects before referencing in other configuration objects.

The API provides options to create, retrieve, update, and delete (CRUD) using standard HTTP methods.

HTTP Method	Operation
POST	Create Object
GET	Retrieve Object
PUT	Update Object
DELETE	Delete Object

Table 3 HTTP Methods to Operations Mapping

All PUT and POST methods with a body in the Request must set the 'Content-Type' header to 'application/json' unless noted specifically in the object APIs in this document.

2.5.1 HTTP Response Codes

VMP Service Manager typically returns the following status in response to the REST HTTP requests:

a. 200 OK. This indicates that the REST request was successfully handled. For DELETE methods, multiple DELETE operation of the same REST resource also returns 200 OK. A response body of the Content Type 'application/json' may be present depending on the operation. For certain resources like Service Instances, a PUT results in 200 OK, but there may be further operational errors that are subsequently reported as Events and Logs.

b. 404 Not Found. This is a failure to find the REST resource for GET and PUT objects. No response body is returned.

c. 400 Bad Request. This response indicates that the request was syntactically incorrect. The response body of Content Type 'application/json' is returned with 'error' description. This could be because of invalid URL structure composition, incorrect JSON structure in request body.

d. 500 Internal Server Error. This indicates that the request could not be processed due to a failure in VMP.

2.5.2 Service Provisioning and Management APIs Transport Security.

The Service Manager hosts the Service Provisioning and Management APIs. The default installation supports access to the APIs using HTTPS protocol over port 8043.

A default self-signed certificate is installed by default in the PAM node for SM APIs. This certificate (and key) can be updated in the field according to user requirements.

2.5.3 Authorization

All access by Platform Operator and Tenant' administer and technician roles MUST be authenticated and authorized. Typically, the technician role cannot modify configurations. The authorization constrains the access to the objects allowed to be managed by Platform Operator and Tenant. So a Platform Operator authorization shall not allow access to tenant objects. One role 'administrator' role is supported and the login allows full access as Platform Operator and Service Administrator.

The Authorization scheme is based on the client presenting an access Token in the HTTP Authorization Header in each request.

The default installation comes with a preset Access Token. Therefore, the Service Manager allows access to clients with the Access Token.

The token is installed by default. The token is based on the JWT (JSON Web Token) format for future compatibility and auto created in the installation procedure. Subsequently, any API call to the Service Manager must present that token in the Authorization HTTP header.

2.5.4 System Security

When the VMP system is installed, the default CLI access username and password can be changed from factory default. Once the first Service Manger instance is up, the changed default username and password must be synchronized to the rest of the system by invoking a Service Manger API. This API must be invoked before any further service provisioning. The invocation of this API is not needed if the factory default CLI username and password are not changed. In current release, the username can only be admin.

Update system username and password

PUT /v2/mgmtcontrol/defaultuser

Message Body:
{

```
   " properties"  : {
      " username"  : " admin" ,
      " password"  : “new-password”
   }
}
```

Returns:

HTTP/1.1 200 OK

2.5.5 Managed Objects

The objects defined in this document are managed objects and they are identifiable and addressable directly. Each object follows the same structure as follows:

Name	A name of an object is a unique label to identify that object within the scope of the object type and scope (like tenant or system). For example, a ChannelSource can have a name ABC and an AssetWorkflowTemplate can be named ABC as well. But two ChannelSources within the same tenant cannot be named ABC.
Id	A unique ID allocated by VMP Management to the management object. It is a read only attribute and MUST be treated as an opaque reference. This is used as object reference to connect the objects. For example, the zoneRef key in a ComputeHost links the ComputeHost object to a Zone object via the Id of the Zone object in the zoneRef value. When objects are referenced by other objects, they cannot be deleted, unless the references are removed first. The Id is assigned by VMP and returned in GET response.
externalId	The URL for the object This externalId is returned in the GET Response.
Type	The type or class of objects to which the object belongs. The type is returned in a GET
Properties	The Properties contain type specific key-values for an object.

2.5.6 Object Schema

Available separately.

2.6 System Provisioning API

The System Provisioning APIs are used to create and manipulate system level objects in the Platform Domain. The System Provisioning APIs can be invoked by the Platform Operator.

Figure 4 provides the relationship of the various objects within the Platform Domain. The shaded objects will be supported in a future release.

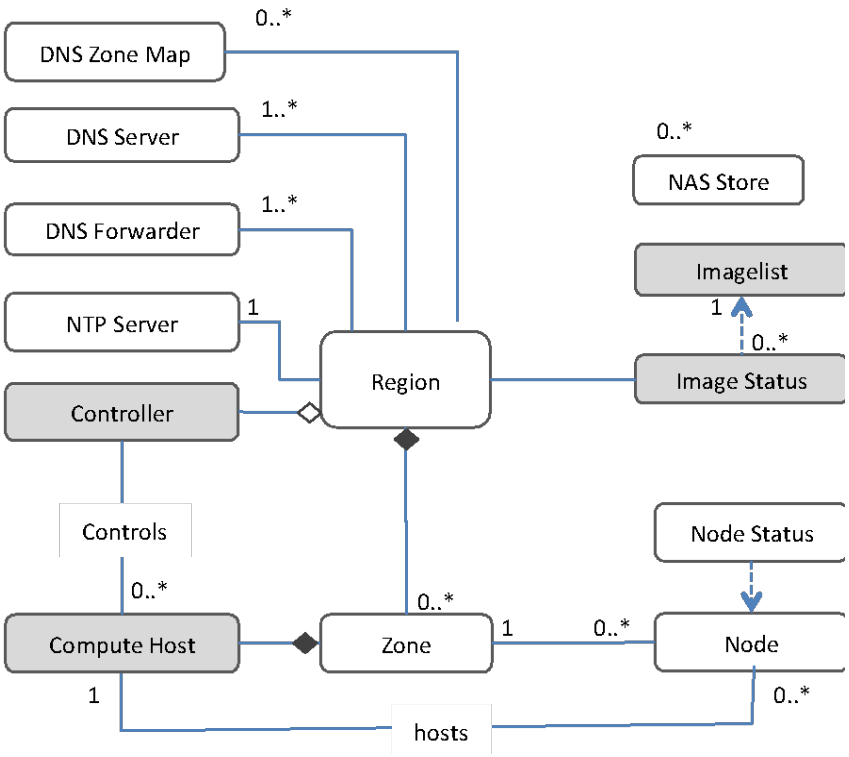


Figure 4 Platform Domain - Object Relationship

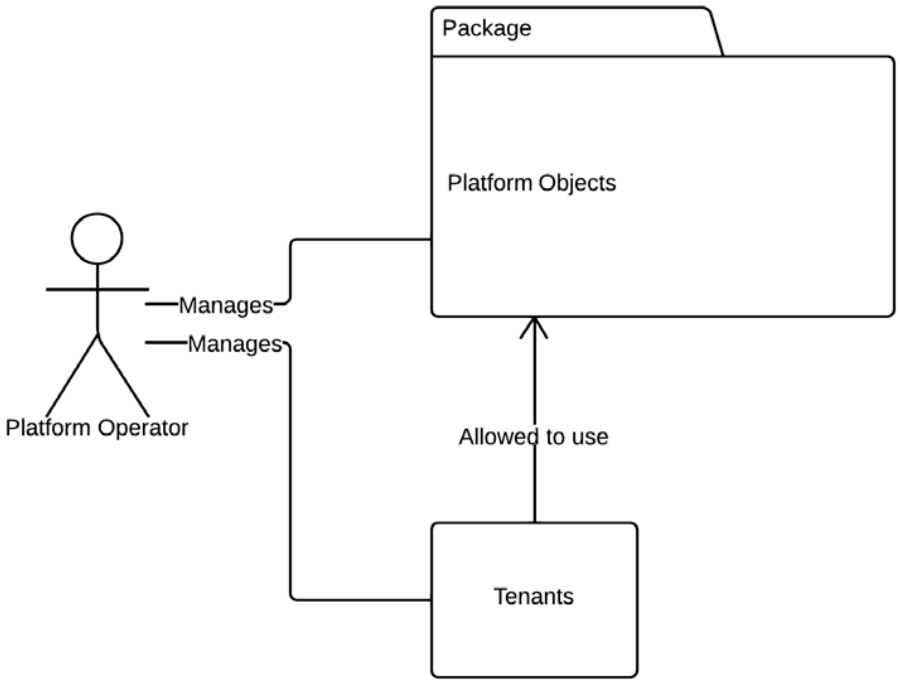


Figure 5 Platform operator functions

2.6.1 Tenant

The Tenant object is the root object for the Platform domain and the Service domain. The object supports the domain name and authorization for the domain.

For Platform domain, the tenant object is ‘system’ and is read only. The domain name is the domain name for the VMP that was supplied at the installation.

For Service domain, a default Read-Only object by the name ‘0’ exists for future use.

Get all tenants

GET /v2/tenants

Returns a collection of tenant objects.

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

[{
   "name" : "system",
   "id" : "smtenants.smtenant.0", "type" :
   "tenants",
   "externalId" : "/v2/tenants/system", "properties" : {
      "domain" : "VMP.domain.com",
      "description" : "the system"
   }
}]
```

Get one tenant

GET /v2/tenants/{tenantName}

Create a Tenant

Not allowed in this release. Only default tenants ‘0’ and ‘system’ exist.

Delete a Tenant

Not Allowed in this release

Update a Tenant

Not Allowed in this release

2.6.2 Region

In this release, only one Region is supported. The object is named ‘region-0’ and can only be read.

Get all regions

GET /v2/regions

Returns a collection of region objects.

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

[{
   "name" : "region-0",
   "id" : "smtenant_system.smregion.region-0",
   "type" : "regions",
   "externalId" : "/v2/regions/region-0",
   "properties" : {
      "description" : "default region", "controllersRef"
      : []
   }
}]
```

Get one region

GET /v2/regions/{regionName}

2.6.3 Zone

The zone is in the scope of a region. The regionRefparameter is the ‘id’ of the region object in which the zone exists.

Get all zones

GET /v2/zones

Returns a collection of zone objects

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

[{
   "name" : "zone1",
   "id" : "smtenant_system.smzone.zone1", "type" :
   "zones",
   "externalId" : "/v2/zones/zone1", "properties" : {
```

```
      "description" : "default zone",
      "regionRef" : "smtenant_system.smregion.region-0"
   }
}]
```


Get one zone
GET /v2/zones/{zoneName}

Create a zone
POST /v2/zones/{zoneName}

Delete a zone
DELETE /v2/zones/{zoneName}

Update a zone
PUT /v2/zones/{zoneName}

2.6.4 DNSServer

The DNSServer object defines the Name server for VMP. This is used to insert the VMP A Records and for load balancing components within VMP.
Depending on the installation time choice made, the DNSServer can be supported on the VMP PAM or provided by an external server.
When PAM acts as the DNSServer, all PAM nodes (high availability) act as redundant DNS Servers. This object is setup at the time of installation and cannot be changed.

Get all dns servers
GET /v2/regions/{regionName}/dnsservers

Returns a collection of DNS server objects


```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

[{
   "name" : "dns-alpha",
   "id" : " smregion_region-0.smdnsserver.dns-alpha", "type" :
   "dnsservers",
   "externalId" : "/v2/regions/region-0/dnsservers/dns-alpha", "properties" : {
      "description" : "a dnsserver", "ipAddr" :
      "10.0.0.2",
      "domain" : "VMP.domain.com",
      "authType" : "tsig",
      "tsigAlgorithm" : "hmac-md5.sig-alg.reg.int", "tsigKey"
      : "somekey"
```

```
    }
}]
```


Get one DNS server

GET /v2/regions/{regionName}/dnsservers/{dnsserverName}

2.6.5 DNSForwarder

The DNSForwarder object defines the Query server for VMP. This is used to resolve the DNS names by VMP components.
This object is setup at the time of PAM installation and can be updated afterwards.

Get all dns forwarders

GET /v2/regions/{regionName}/dnsforwarders

Returns a collection of DNSForwarder objects

HTTP/1.1 200 OK

Content-Type: application/json; charset=utf-8


```
[{
    "name" : "dnsq-1",
    "id" : " smregion_region-0.smdnsforwarder.dnsq-1", "type" :
    "dnsforwarders",
    "externalId" : "/v2/region/region-0/dnsforwarders/dnsq-1", "properties" : {
        "description" : "a dns query server/forwarder", "ipAddr" :
        "10.0.0.3",
        "domain" : "VMP.domain.com
    }
}]
```


Get one DNS forwarder

GET /v2/regions/{regionName}/dnsforwarders/{dnsforwarderName}

2.6.6 NTPServer

There is one NTP Server Object per region. Each object may have one or more NTP Server instances defined.

Get all NTP Servers

GET /v2/regions{regionName}/ntpservers

Returns a collection of ntpserver objects
HTTP/1.1 200 OK

```
Content-Type: application/json; charset=utf-8

[{
   "name" : "ntpservers-1",
   "id" : "smregion_region-0.smntpserver.ntpservers-1", "type" :
   "ntpservers",
   "externalId" : "/v2/region/region-0/ntpservers/ntpservers-1",  "properties" : {
       "description" : "a bunch of NTP servers", "servers" : [
            "10.2.3.4",
            "10.2.3.5"
       ]
    }
}]
```

Get one NTP server

GET /v2/regions/{regionName}/ntpservers/{ntpserverName}

Update an NTP server

PUT /v2/regions/{regionName}/ntpservers/{ntpserverName}

2.6.7 DNSZoneMap

The DNSZoneMap object defines the DNS zones and binds subnets to domain names. There is one DNSZoneMap object per region. By default, there are no domains and networks defined in the object. Such a configuration allows all DNS updates to be processed when there is no matching domain name in the map. Once a domain name is added to the DNSZoneMap, only the IP addresses that match the indicated subnets can be added as A records for that domain.

This object is setup at the time of PAM installation and can be updated afterwards.

Get all dns zone maps

GET /v2/regions/{regionName}/dnszonemaps

Returns a collection of DNSZoneMap objects

GET /v2/regions/region-0/dnszonemaps

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

[{
   "name" : "zonemap",
   "id" : "smregion_region-0.smdnsforwarder.zonemap",
   "type" : "dnszonemaps",
   "externalId" : "/v2/region/region-0/dnszonemaps/zonemap",
```

```
   " properties" : {
      " description" : " a dns zone map collection" , " map" : [ {
" domain" : " VMP.domain.com" ,
          " subnets" : [ ’10.10.1’, ’10.10.2’ ]
} ]
   }
}]
```


Get one dns zone map

GET /v2/regions/{regionName}/dnszonemaps/{dnszonemapName}

Update one dns zone map

PUT /v2/regions/{regionName}/dnszonemaps/{dnszonemapName}

2.6.8 Node

The Node is an external machine (usually a VM) that can be used by VMP. The Node object defines the properties of such a machine.

The Node objects are associated with a zone.

Get all Nodes

GET /v2/regions/{regionName}/nodes

Returns a collection of Node objects

GET /v2/regions/region-0/nodes

HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8


```
[{
   " name" : " node-1" ,
   " id" : " smregion_region-0.smnode.node-1" , " type" :
   " nodes" ,
   " externalId" : " /v2/region/region-0/nodes/node-1" , " properties" :
   {
      " description" : " a mpe worker node" , " adminState" :
      " inservice" ,
      " zoneRef" : "  smtenant_system.smzone.zone1" ,
      “aic”:null,
       " image" : {
            " imgTag" : " mpe" , " personality" :
            " worker" , " version" : " 1.0"
      },
```

```
    "interfaces" : [
        {
            "type" : "mgmt",
            "inet" : "10.10.1.1"
        },
        {
            "type" : "data-in",
            "inet" : "10.10.1.1"
        },
        {
            "type" : "data-out",
            "inet" : "10.10.2.1"
        }
    ]
  }
}]
```


Get one Node
GET /v2/regions/{regionName}/nodes/{nodeName}

Update one Node
PUT /v2/regions/{regionName}/nodes/{nodeName}

Delete one Node
DELETE /v2/regions/{regionName}/nodes/{nodeName}

Create one Node
POST /v2/regions/{regionName}/nodes/{nodeName}

2.6.9 NAS Store

NASStore objects defines a read/write NAS device that can be used to store assets that VMP creates.
For Live services, the assets are the segments for the specified DVR Window duration for a channel; for VOD, the assets are the ingested and indexed content.

Get all NAS stores
GET /v2/nasstores

Returns a collection of objects

HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8


```
        "name" : "nas-1",
        "id" : "smtenant_system.smnasstore.nas-1",
        "type" : "nasstores",
```

```
"externalId" : "/v2/nasstores/nas-1",
"properties" : {
    "description" : "nas storage", "share" :
    "/sharelocation", "version" : "3.0",
    "servers" : [
        {
            "rangeStart" : "10.1.1.1",
            "rangeEnd" : "10.1.1.20"
        }
    ]
}
```


Get one NAS store

GET /v2/nasstores/{nasstoreName}

Create a NAS store

POST /v2/nasstores/{nasstoreName}

Delete a NAS store

DELETE /v2/nasstores/{nasstoreName}

Update a NAS store

PUT /v2/nasstores/{nasstoreName}

2.6.10 Default User

DefaultUser object stores the default CLI access username and password to access any VM in the VMP system. There is only one default username and password. The password is encrypted when persisted in storage. The default username and password are only used by internal applications to communicate across VMs since each VM is separately secured. This object must be updated to match the chosen username and password at installation. See Installation Guide for detail. Since all APIs are communicated through SSL only, this object is not visible on the wire.

Update the default username and password

PUT /v2/mgmtcontrol/defaultuser

Message Body


```
{
  "properties" : { "username" :
    "admin", "password" :
    "default"
  }
```


Update default username and password

PUT /v2/mgmtcontrol/defaultuser

2.7 Service Management API

Service Management API allows the tenant to create policies, templates, and resources that can be referenced from multiple service instances. The Service Management API also provides interfaces to manage service instances and configure one or more asset workflows within the service instance. The Asset Workflows are used to bind policies, resources, and templates to implement Asset ingest and packaging operations on one or more media source.

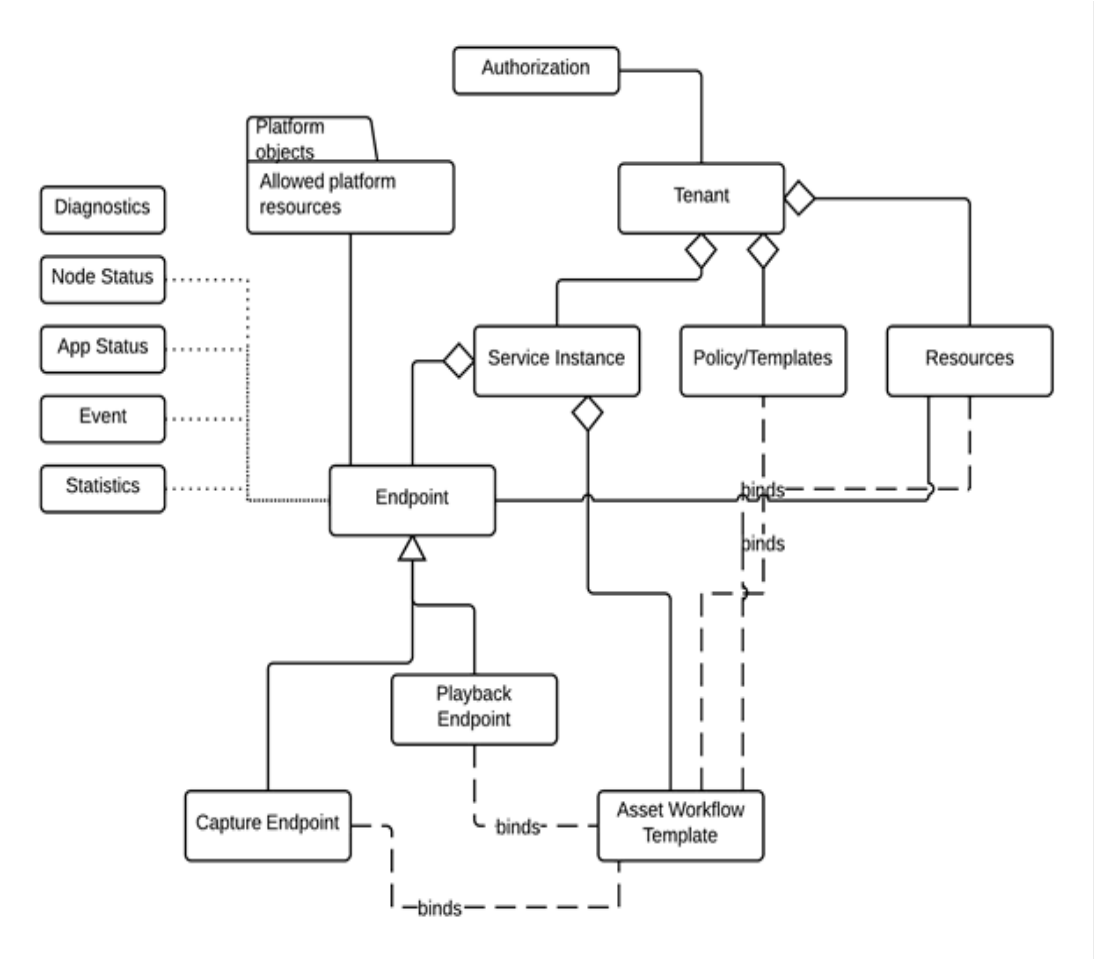


Figure 6 Service Domain Objects and Inter-relationship

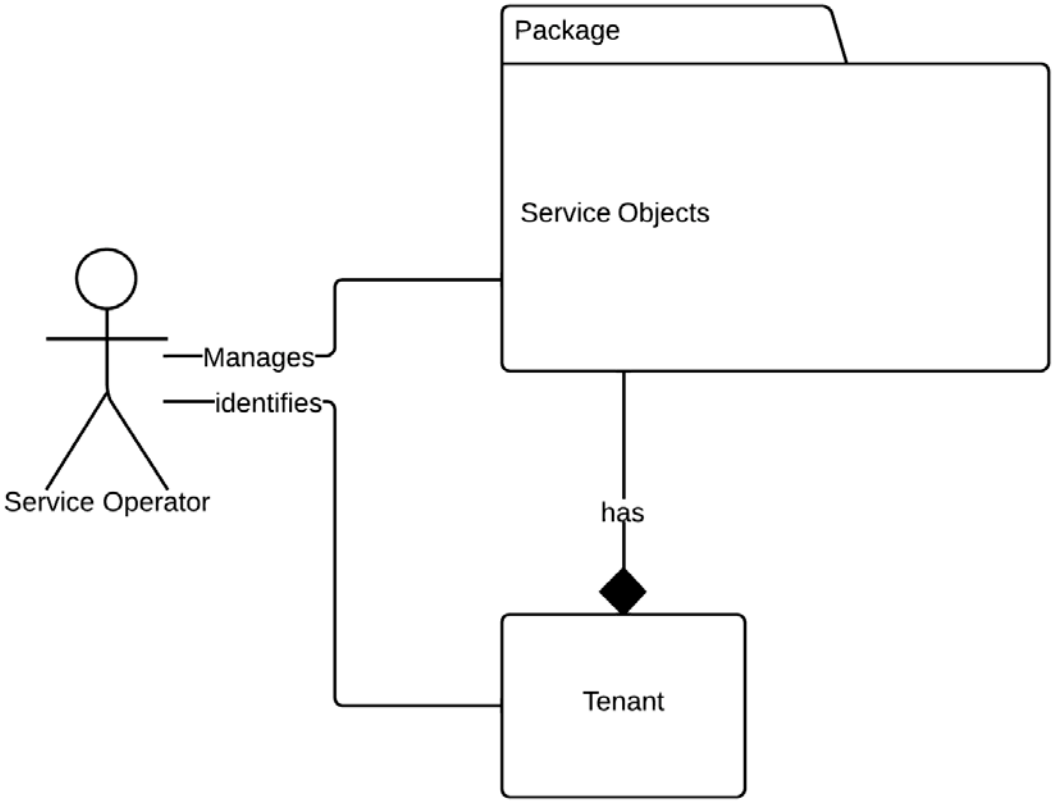


Figure 7 Service Operator functions

The URLs for Service Management use the following patterns:

- /v2/{collectionName}[?ref=objectID[&ref=objectID]+] o /v2/{collectionName}/{objectName}
- /v2/{scopeCollection}/{scopeObjectName}/{collectionName}[?ref=objectID[&ref=objectID]+]
- /v2/{scopeCollection}/{scopeObjectName}/{collectionName}/{objectName} o /v2/id/{objectID}

The collection and scopeCollection are object collections. The scopeObjectName and objectName are managed objects (see below). The scopeCollectionName and scopeObjectName allow hierarchical access to lower level collectionName objects.

In addition, the query parameters for collection support object access (GET) by reference ('ref') to one or more object IDs.

Query parameters also support filters to allow certain match conditions (like find all ComputeHosts that are associated with a Zone with a given zone object ID.

Note that the Service 'Use' APIs are not necessarily HTTP or REST or JSON. These interfaces are to support industry standard, or cisco or vendor specific control APIs. They are documented later in this document. However, the discovery of the Service APIs is supported via the HTTP REST Management interface.

2.7.1 Service Template

Service Templates are the documents that define the Service Instance behavior, its validation, configuration constraints, and schema versions.

Service Templates are built into the VMP release and are read-only. Only services that have the templates released are supported in VMP and the VMP GUI.

Get all service templates

GET /v2/servicetemplates

Returns a collection of objects

HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8


```
[
   {
      " id" : " smbase.smservicetemplate.ums_0.v1" , " name" :
      " ums_0.v1" ,
      " type" : " servicetemplates" ,
      " externalId" : " /v2/servicetemplates/ums_0.v1" ,
      " properties" : {
         " serviceType" : " ums" , " description" :
         " UMS Service" , " state" : " active" ,
         " templateType" : " ums_0"
      }
   },
   {
      " id" : " smbase.smservicetemplate.cos_0.v1" , " name" :
      " cos_0.v1" ,
      " type" : " servicetemplates" ,
      " externalId" : " /v2/servicetemplates/cos_0.v1" ,
      " properties" : {
         " serviceType" : " cos" ,
         " description" : " Cisco Object Store (COS) Service" , " state" :
         " active" ,
         " templateType" : " cos_0"
      }
```

```
    }
]
```

Get one service template

GET /v2/servicetemplates/{servicetemplateName}

2.7.2 Image Manifest

An ImageManifest defines a valid combination of images that can be used to instantiate a Service Instance. The user creates Image Manifests and for each Service Instance, a manifest is associated. The Nodes corresponding to the associated Image Manifest are used in that Service Instance.

The imgTagand versioncorrespond to the OVA image needed. There are only two imgTagvalues required –
mpeand mce.

Get all image manifests

GET /v2/imagemanifests

Returns a collection of objects

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

[
    {
        "name" : "imageset.1",
        "id" : "smtenant_0.smimagemanifest.imageset.1", "type" :
        "imagemanifests",
         "externalId" : "/v2/imagemanifests/imageset.1", "properties"
        : {
            "description" : "the images to use for a service instance", "images" : [
                {
                    "imgTag" : "mpe",
                    "version" : "1.0.1"
                },
                {
                    "imgTag" : "mce",
                    "version" : "2.0"
                }
            ]
        }
    }
]
```

Get one image manifest

GET /v2/imagemanifests/{imageManifestName}

Create an image manifest
POST /v2/imagemanifests/{imageManifestName}

Delete an image manifest
DELETE /v2/imagemanifests/{imageManifestName}

Update an image manifest
PUT /v2/imagemanifests/{imageManifestName}

2.7.3 Key Profile

KeyProfile object defines the details about the key type and key provider server details. The Key Profile object is associated in an ABRAssetPublishTemplate. When an asset is played out within a scope of an ABRAssetPublishTemplate, the corresponding Key Profile object is used to obtain and apply the Keys from the server.

The following key profile types (properties.type) are supported:

- aes keygenerator An internal HLS-AES-256 key generator. For every Live Channel asset, a separate KeyRotation interval can be used when specified (See ChannelLineup object)
- **irdeto-pr Irdeto (Playready)**
- irdeto-hls Irdeto (HLS)
- keystore-hls Cisco Keystore (HLS)
- verimatrix-hls Verimatrix (HLS)
- verimatrix-pr Verimatrix (Playready)
- axs-hds Adobe Access License Server (along with one of the above key profiles)
- vgc-hls Vgc (HLS)
- insys insys
- CECC-DASH valid for KMS types EZDRM snd BUYDRM
- Nagra supports DRM types HLS-AES-128

Each of the above types has type specific properties. Please refer to the schema details (available separately).

Get all key profiles
GET /v2/keyprofiles

Returns a collection of objects

HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

```
[
    {
        "name" : "verimatrix-hls",
        "id" : "smtenant_0.smkeyprofile.verimatrix-hls", "type" :
        "keyprofiles",
         "externalId" : "/v2/keyprofiles/verimatrix-hls", "properties" : {
            "type" : "verimatrix",
            "description" : "verimatrix key profile", "keyAquisition" : {
                "uri" : "https://172.34.29.15:5000/", "caCert" :
                "<sample clipped>", "userName" : "test",
                "passPhrase" : "pass123", "account" :
                "vcasdemo"
            },
            "drmType" : "hls-aes-128"
        }
    }
]
```


Get one key profile
GET /v2/keyprofiles/{keyprofileName}

Create a key profile
POST /v2/keyprofiles/{keyprofileName}

Delete a key profile
DELETE /v2/keyprofiles/{keyprofileName}

Update a key profile
PUT /v2/keyprofiles/{keyprofileName}

2.7.4 HTTP Header Policy

The HTTPHeaderPolicy object defines the cache and other headers for responses sent for playback towards HTTP/1.1 and HTTP/1.0 clients. The headers can be defined for master manifest, stream manifest and chunk responses.
The HTTPHeaderPolicy is associated with an AssetPublishTemplate.

Get all HTTPHeaderPolicy
GET /v2/httpheaderpolicies

Returns a collection of objects

HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

```
[
    {
        "name" : "headers-live",
        "id" : "smtenant_0.smabrhttpheaderpolicy.headers-live", "type" :
        "httpheaderpolicies",
         "externalId" : "/v2/httpheaderpolicies/headers-live", "properties" : {
            "description" : "cache policy for live assets", "masterManifest" : {
                "http_1_0" : [
                    {
                        "header" : "Expires",
                        "value" : "3600"
                    }
                ],
                "http_1_1" : [
                    {
                        "header" : "Cache-control", "value" :
                        "max-age=3600"
                    }
                ]
            },
            "streamManifest" : { "http_1_0" : [
                    {
                        "header" : "Expires",
                        "value" : "3600"
                    }
                ],


                "http_1_1" : [
                    {
                        "header" : "Cache-control", "value" :
                        "max-age=3600"
                    }
                ]
            },
            "chunk" : {
                "http_1_0" : [
                    {
                        "header" : "Expires",
                        "value" : "3600"
                    }
                ],
                "http_1_1" : [
                    {
                        "header" : "Cache-control", "value" :
                        "max-age=3600"
```

```
                        }
                    ]
                }
            }
}
]
```


Get one ABR http header policy
GET /v2/httpheaderpolicies/{policyName}

Create an ABR http header policy
POST /v2/httpheaderpolicies/{policyName}

Delete an ABR http header policy
DELETE /v2/httpheaderpolicies/{policyName}

Update an ABR http header policy
PUT /v2/httpheaderpolicies/{policyName}

2.7.5 Asset Publish Template

AssetPublishTemplate object defines a particular output format with allowed variant playlists, segment duration, KeyProfile to be applied, HTTPHeaderPolicy to be applied and key rotation interval.

One or more AssetPublishTemplate objects are associated in an Asset Workflow Template.

Get all ABR publish templates
GET /v2/assetpublishtemplates

Returns a collection of objects

HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8


```
[{
 "name" : "publish-live-hls",
 "id" : "smtenant_0.smpublishtemplate.publish-live-hls",
 "type" : "assetpublishtemplates",
 "externalId" : "/v2/assetpublishtemplates/publish-live-hls",
 "properties" : {
  "description" : "a sample HLS publish template",
  "packageFormat" : "hls",
  "segmentDurationInSec" : 10,
  "keyProfileRef" : "smtenant_0.smkeyprofile.verimatrix-hls",
```

```
" keyRotationIntervalInSec" : 3600,
" httpHeaderPolicyRef" : " smtenant_0.smhttpheaderpolicy.headers-live" ,
" variants" : [
{
 " name"            : " appleTv" ,
 " version"         : " 4" ,
    " order"           : " bitrate" ,
 " selectivePublish" : " true" ,
" videoStreams" : [
 {
  " format" : " H264" ,
   " bitrate" : " 2000000"
  },
  {
   " format" : " H264" ,
   " bitrate" : " 4000000"
  }
 ],
 " audioStreams" : [
{
  " name" : " audio-en" ,
  " format" : " AAC" ,
  " language" : " en" ,
   " bitrate" : " 24000"
  },
  {
   " name" : " audio-fr" ,
   " format" : " AAC" ,
   " language" : " fr" ,
   " bitrate" : " 24000"
  }
```


2.7.6 AssetLifeCyclePolicy

AssetLifeCyclePolicy object defines the lifecycle management policy for the assets. For Live assets, it means that the DVR window can be specified using this policy to expire the segments after a certain time. For VOD and CDVR assets, the whole assets can be automatically deleted or moved to another storage for archival (or a combination of both actions) after a specified time.

The example below is for a channel asset's DVR window of 1 hour. When a policy is not specified for a Live service, a default DVR window is applied (3 segments). For VOD assets, if a policy is not specified, the assets don't expire unless a control message is sent to delete the API (See Service Use APIs).

Get all AssetLifeCyclePolicy objects

GET /v2/assetlifecyclepolicies

Returns a collection of objects

HTTP/1.1 200 OK

Content-Type: application/json; charset=utf-8


```
[
   {
      " id" :" smtenant_0.smassetlifecyclepolicy.asset" , " name" :
      " asset" ,
      " type" : " assetlifecyclepolicies" ,
      " externalId" :" /v2/assetlifecyclepolicies/asset" , " transactionId" :" 45d121a0-
      0f54-48a4-9e0a-e00d975bee0a" , " properties" : {
         " description" : " dest" , " type" :
         " dvrwindow" , " rules" : [
            {
               " enabled" : true,
               " matchTags" : " segment" ,
               " action" : " move" ,
               " moveParams" : {
                  " storageRef" :" smtenant_0.smcosstore.cos-1"
               },
               " timeSec" : 3600
            }
         ]
      }
   }
]
```


Get one asset life cycle policy

GET /v2/assetlifecyclepolicies/{policyName}

Create an asset life cycle policy

POST /v2/assetlifecyclepolicies/{policyName}

Delete an asset life cycle policy

DELETE /v2/assetlifecyclepolicies/{policyName}

Update an asset life cycle policy

PUT /v2/assetlifecyclepolicies/{policyName}


```
],
  " defaultAudioStream"  : " audio-en"
 },
 {
  " name"            : " iphone" ,
```

```
  "version"          : "3",
  "order"            : "rank",
  "selectivePublish" : "false",

  "videoStreams"  : [
   {
    "format"  : "H264",
    "rank"  : "3"
   },
   {
    "format"  : "H264",
    "rank"  : "1"
   }
  ],
  "audioStreams"  : [
   {
    "name"  : "audio-en",
    "format"  : "AAC",
    "language"  : "en",
    "bitrate"  : "9600"
   }
  ],
  "defaultAudioStream"  : "audio-en"
 },
 {
  "name"  : "mac",
  "version"  : "4"
 }
]
}]
```


Get one ABR publish template

GET /v2/assetpublishtemplates/{publishtemplateName}

Create an ABR publish template

POST /v2/assetpublishtemplates/{publishtemplateName}

Delete an ABR publish template

DELETE /v2/assetpublishtemplates/{publishtemplateName}

Update an ABR publish template

PUT /v2/assetpublishtemplates/{publishtemplateName}

2.7.7 Asset Redundancy Policy

The AssetRedundancyPolicy object defines how the Service Instance needs to create redundant assets.
The policy works in conjunction with control messages for CDVR where the matchTag is matched with the desired copyright required and channel lineup rightsTag. For the Live asset, the matchTag is used to match against the Channel Lineup. For VOD, it is matched against the control message. For example, for a Live service, a Channel identified by ContentId ABC in Channel Lineup and has the rightsTag as 'common', any associated AssetRedundancyPolicy which has a rule that has a matchTag 'common' will be applicable.

There are two types of asset redundancy policies that apply to CDVR. copyCount specifies the number of copies that will be recorded. keepCount specifies the number of copies of a recording that will be retained after the recording is completed, the rest will be deleted. keepCount is always less than or equal to the copyCount value configured.

Get all asset redundancy policies

GET /v2/assetredundancypolicies

Returns a collection of objects

HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8


```
[
  {
    " id" :" smtenant_0.smassetredundancypolicy.redundancy-pol-cos" , " name" :
    " redundancy-pol-cos" ,
    " type" :" assetredundancypolicies" ,
    " externalId" :" /v2/assetredundancypolicies/redundancy-pol-cos" , " transactionId" :
    " 1257449d-899a-43c8-b373-4b94f4fbe49c" , " properties" : {
      " description" : " A sample local and remote mirroring policy for COS objects" , " rules" : [
        {
          " matchTag" : " local" ,
          " trigger" : " start" ,
          " copyCount" : 2,
          " keepCount" : 2, " enabled" :
          true
        },
        {
          " matchTag" : " remote" ,
          " trigger" : " start" ,
          " copyCount" : 3,
          " keepCount" : 0, " enabled" :
          false
        }
            ]
      }
```

}]

Get one asset redundancy policy
GET /v2/assetredundancypolicies/{policyName}

Create an asset redundancy policy
POST /v2/assetredundancypolicies/{policyName}

Delete an asset redundancy policy
DELETE /v2/assetredundancypolicies/{policyName}

Update an asset redundancy policy
PUT /v2/assetredundancypolicies/{policyName}

2.7.8 Channel Source

A Channel Source is a type of media source object which defines the channel source information like stream type, multicast (or others in future) source URLs and Control addresses. It also includes the StreamProfile reference to reuse the StreamProfile objects. Stream type of MPEG ATS and TS are supported.

Get all channel sources
GET /v2/channelsources

Returns a collection of objects

HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8


```
[
      {
            "name" : "src-abc-1",
            "id" : "smtenant_0.smchannelsource.src-abc-1", "type" :
             "channelsources",
             "externalId" : "/v2/channelsources/src-abc-1", "properties"
             : {
               "channelId" : "abc.ca",
                "description" : "The channel source for ABC in CA, USA", "streamType" : "ATS",
                "streams" : [
                   {
                         "profileRef" : "smtenant_0.smstreamprofile.HD_4M", "sources" : [
                               {
                                     "sourceUrl" : "udp://237.7.7.7:7000",
                                     "sourceIpAddr" : "10.0.0.20"
                               }
                         ]
```

```
                },
                {
                        "profileRef" : "smtenant_0.smstreamprofile.SD_2M", "sources" : [
                                {
                                        "sourceUrl" : "udp://237.7.7.8:7000",
                                        "sourceIpAddr" : "10.0.0.10"
                                }
                        ]
                }
            ]
}
        }
]
```

Get one channel source
GET /v2/channelsources/{sourceName}

Create a channel source
POST /v2/channelsources/{sourceName}

Delete a channel source
DELETE /v2/channelsources/{sourceName}

Update a channel source
PUT /v2/channelsources/{sourceName}

2.7.9 NAS Media Source

NASMediaSource defines the NAS storage that can be read and contains ATS content that can be ingested and indexed into VMP. It contains a list of address ranges to allow multiple addresses to be provides for a NAS cluster that exports the share to be used.

NASMediaSource is used in the AssetWorkflowTemplate's assetLineupRef member for VOD service. It is not supported for Live Service.

Get all NAS media sources
GET /v2/nasmediasources

Returns a collection of objects

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

[
        {
```

```
            "name" : "my-nas",
            "id" : "smtenant_0.smnasmediasource.my-nas", "type" :
             "nasmediasources",
             "externalId" : "/v2/nasmediasources/my-nas", "properties" :
             {
                  "description" : "nas source for vod content", "share" :
                  "/media/vod",
                  "version" : "3.0", "servers" : [
                        {
                              "rangeStart" : "10.0.10.1",
                              "rangeEnd" : "10.0.10.20"
                        }
                  ]
            }
      }
]
```


Get one NAS media source

GET /v2/nasmediasources/{sourceName}

Create a NAS media source

POST /v2/nasmediasources/{sourceName}

Delete a NAS media source

DELETE /v2/nasmediasources/{sourceName}

Update a NAS media source

PUT /v2/nasmediasources/{sourceName}

2.7.10 Dynamic Source

DynamicSource defines the media source that can be discovered dynamically. It contains a media source type and URL that can be used to discover the actual media source.

The DynamicSource is used in the mediaSourceRef of an AssetWorkflowTemplate object.

Get all dynamic media sources

GET /v2/dynamicsources

Returns a collection of objects

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

{
   "id" : "smtenant_0.smdynamicsource.dyna-source-1", "name" :
   "dyna-source-1",
   "type" : "dynamicsources",
   "externalId" : "/v2/dynamicsources/dyna-source-1", "transactionId" : "2d45b6a6-
   7796-4b38-8970-e5e640235f0a", "properties" : {
      "url" : "http://rmis.VMP.com", "type" :
      "amr1-discovery",
      "description" : "A sample dynamic source"
   }
}
```

Where type values can be amr1-discovery.

Get one dynamic source

GET /v2/dynamicsources/{sourceName}

Create a NAS media source

POST /v2/dynamicsources/{sourceName}

Delete a NAS media source

DELETE /v2/dynamicsources/{sourceName}

Update a NAS media source

PUT /v2/dynamicsources/{sourceName}

2.7.11 Channel Lineup

A ChannelLineup object defines the live channel assets that will be produced using an

AssetWorkflowTemplate. It specifies a list of channel sources, associated ContentId, rightsTag and optional encryption Info for each channel asset. The rightsTag is used to select the appropriate asset redundancy policy to apply. The keyInfo is optional and used only when a KeyProfile of aes-keygenerator (VMP local key generator) is used in the Asset WorkflowTemplate. When ESAMProfile is used in the Asset Workflow Template for a given Channel Lineup, each Channel in the Channel Lineup can be associated with a poisAcquisitionPointId value that will be used as the Acquisition Point Id in the interface to the poisServer (in ESAMProfile).

The ChannelLineup is used in the assetLineupRef of an AssetWorkflowTemplate object for a Live service.

Get all channel lineups

GET /v2/channellineups

Returns a collection of objects

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

[
    {
        "name" : "national",
        "id" : "smtenant_0.smchannellineup.national", "type" :
         "channellineups",
         "externalId" : "/v2/channellineups/national", "properties" : {
            "description" : "national channels lineup", "sourcesRef" : [
                {
                    "sourceRef" : "smtenant_0.smchannelsource.src-abc-1", "contentId" :
                    "abc-ca",
                    "rightsTag" : "common", "keyInfo" : {
                        "keyRotationIntervalInSec" : 60
                    },
                    "customConfigs : [
                        {
                            "name"  : "poisAcquisitionPointId", "value"  :
                            "abc-ca-pois"
                    ]
                },
                {
                    "sourceRef" : "smtenant_0.smchannelsource.src-fox", "contentId" :
                    "fox",
                    "rightsTag" : "common"
                }
            ]
        }
    }
```

]

Get one channel lineup
GET /v2/channellineups/{lineupName}

Create a channel lineup
POST /v2/channellineups/{lineupName}
Delete a channel lineup
DELETE /v2/channellineups/{lineupName}

Update a channel lineup
PUT /v2/channellineups/{lineupName}

2.7.12 Dynamic Lineup

A DynamicLineup object defines media channels that can be discovered

dynamically The DynamicLineup is used in the assetLineupRef of an

AssetWorkflowTemplate object.

Get all channel lineups
GET /v2/dynamiclineups

Returns a collection of objects

HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8


```
{
   " id" : " smtenant_0.smdynamiclineup.dyna-lineup-1" , " name" : " dyna-
   lineup-1" ,
   " type" : " dynamiclineups" ,
   " externalId" : " /v2/dynamiclineups/dyna-lineup-1" , " transactionId" : " d0e3925e-
   cc9a-414e-a34c-9eb61910aa4c" , " properties" : {
      " interfaces" : [
         {
            " url" : " http://rm.VMP.com/" , " type" :
            " amr1-control"
         }
      ],
      " description" : " Sample dynamic lineup"
   }
}
```

The type value can be amr1-control.

Get one channel lineup
GET /v2/dynamiclineups/{lineupName}

Create a channel lineup
POST /v2/dynamiclineups/{lineupName}

Delete a channel lineup
DELETE /v2/dynamiclineups/{lineupName}

Update a channel lineup
PUT /v2/dynamiclineups/{lineupName}

2.7.13 ESAMProfile

ESAM Profile objects define the external POIS server access parameters. The following versions are supported for the POIS server interface:

- OC-SP-ESAM-API-I01
- OC-SP-ESAM-API-I03

Get all ESAMProfile profiles
GET /v2/esamprofiles

Returns a collection of objects

HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8


```
[
      {
            "name" : "esamserver",
            "id" : "smtenant_0.smesamprofile.esamserver", "type" :
             "esamprofiles",
             "externalId" : "/v2/esamprofiles/esamserver", "properties" : {
                  "poisServerUrl" :          "http://esam.sp.com:3030/"
                  "description" : "POIS Server",
                  "version" : "OC-SP-ESAM-API-I01"
              }
      }
]
```


Get one ESAMProfile

GET /v2/esamprofiles/{profileName}

Create an ESAMProfile
POST /v2/esamprofiles/{profileName}

Delete an ESAMProfile
DELETE /v2/esamprofiles/{profileName}

Update an ESAMProfile
PUT /v2/esamprofiles/{profileName}

2.7.14 ESAM Template

VMP supports the CableLabs ESAM Template for real-time dynamic ad insertion for VOD HLS. The ESAM Temlate related APIs are listed below.

Get all ESAMTemplate
GET /v2/esamtemplates

Returns a collection of objects

HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8


```
[
  {
    " id" : " smtenant_0.smesamtemplate.new_template" ,
    " name" : " new_template" ,
    " type" : " esamtemplates" ,
   " externalId" : " /v2/esamtemplates/new_template" ,
    " properties" : {
      " description" : "  description  for esam template" ,
       " template" : "
    <ManifestConfirmConditionNotification>
     <ManifestResponse>
<SegmentModify>
       <FirstSegment>
       <Tag value=" #EXT-X-DISCONTINUITY" />
   <Tag value=" #EXT-X-SIGNAL-START:SignalID
                  =$startSignal$" />
       </FirstSegment>
  <SpanSegment>
  <Tag value=" #EXT-X-SIGNAL-SPAN:
```

```
          SignalID=$startSignal$,
         TimeFromSignalStart=${timeFromSignal}" />
        </SpanSegment>
        <LastSegment>
 <Tag value=" #EXT-X-SIGNAL-END:SignalID
                =$endSignal$" />
        <Tag value=" #EXT-X-DISCONTINUITY" />
</LastSegment>
      </SegmentModify>
     </ManifestResponse>
    </ManifestConfirmConditionNotification>
              "
     }
  ]
```

Get one ESAMTemplate
GET /v2/esamtemplates/{templateName}

Create an ESAMTemplate
POST /v2/esamtemplates/{templateName}

Delete an ESAMTemplate
DELETE /v2/esamtemplates/{templateName}

Update an ESAMTemplate
PUT /v2/esamtemplates/{templateName}

2.7.15 Stream Profile

StreamProfiles objects define the source stream parameters like encoding and bitrate. These are reusable and can be associated to multiple ChannelSource objects.

Get all stream profiles
GET /v2/streamprofiles

Returns a collection of objects

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

[
      {
            " name" : " HD_7M" ,
            " id" : " smtenant_0.smstreamprofile.HD_7M" , " type" :
             " streamprofiles" ,
             " externalId" : " /v2/streamprofiles/HD_7M" ,
```

```
    "properties" : {
                    "encodingType" : "H.264",
                    "bitrate" : "7000000"
                }
        },
        {
                "name" : "SD_3M",
                "id" : "smtenant_0.smstreamprofile.SD_3M", "type" :
                 "streamprofiles",
                 "externalId" : "/v2/streamprofiles/SD_3M", "properties" : {
                    "encodingType" : "H.264",
                    "bitrate" : "3000000"
                }
        }
]
```


Get one stream profile
GET /v2/streamprofiles/{profileName}

Create a stream profile
POST /v2/streamprofiles/{profileName}

Delete a stream profile
DELETE /v2/streamprofiles/{profileName}

Update a stream profile
PUT /v2/streamprofiles/{profileName}

2.7.16 ServiceInstance

ServiceInstance object represents a Service Instance of a service Type (Live, VOD, cDVR). A Service instance is produced by using the Nodes for capture and playback endpoints. The endpoints are selected using the AssetWorkflowTemplate. There can be one or more AssetWorkflowTemplates in a Service Instance. The Service instance uses ImageManifest to specify which images (and hence Nodes) are to be selected for software instantiation and use.

The Service Instance can be stopped or started using the state parameter.
The templateRef parameter specified the Service Template that will control the Service Instance behavior. The title is a customizable name of the service instance. This name shows up on the GUI as the instance name.

The customConfig is used to specify additional Service instance level parameters to accommodate the changes to software installation for MCE and MPE without upgrading the GUI. The allowed name value pairs in the customConfigs are available.
In this release, the Service Instance can only be Read and Updated. A set of 5 unified media service

instances in inactive state come preinstalled. These can be customized and used (activated).

Get all Service Instances

GET /v2/serviceintances

Returns a collection of objects

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

[
      {
            "name" : "ums-0-1", "title" :
            "national-live",
            "id" : "smtenant_0.smserviceinstance.ums-0-1",  "type" :
             "serviceintances",
             "externalId" : "/v2/serviceinstances/ums-0-1",  "properties" : {
                  "state" : "active", "regionsRef" : [
                        "smtenant_system.smregion.0"
                  ],
                  "imageManifestRef" : "smtenant_0.smimagemanifest.0", "description" : "Live
                  service with nas used for media storage", "templateRef" :
                  "smbase.smservicetemplate.ums_0.v1", "customConfigs" : [ {
                        "name" : "test1",
                        "value" : "dummy",
                    }]
            }
      }
]
```

Get one Service Instance

GET /v2/serviceintances/{instanceName}

Update a Service Instance

PUT /v2/serviceintances/{instanceName}

2.7.17 Playback Endpoint

A PlaybackEndpoint is a logical configuration reference point to define associate resources and policy for the playback of assets from VMP. One Service Instance can have one or more Playback Endpoints.
An Origin Service Domain fqdn) maps to one Playback Endpoint.

Get all Playback Endpoints

GET /v2/serviceinstances/{serviceinstanceName}/playbackendpoints

Returns a collection of objects

Example:
GET /v2/serviceinstances/ums-0-1/playbackendpoints

HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8


```
{
   " id" :" smserviceinstance_ums_0_1.smplaybackep.pe1" , " name" :
   " pe1" ,
   " type" : " playbackendpoints" ,
   " externalId" :" /v2/serviceinstances/ums-0-1/playbackendpoints/pe1" , " transactionId" :
   " 04757d8b-5d20-4b7a-811e-f3f2a2b21780" , " properties" : {
      " slaType" : " resource" ,
      " resourceSLA" : {
         " minNode" : 1,
         " desiredNode" : 2,
         " maxNode" : 2
      },
      " address" : { " vip" : " " ,
         " fqdn" : " live1.VMP21.com"
      },
      " regionRef" :" smtenant_system.smregion.region-0" , " description" :
      " Playback endpoint"
   }
}
```


Get one Playback Endpoint

GET /v2/serviceinstances/{instanceName}/playbackendpoints/{peName}

Create a Playback Endpoint

POST /v2/serviceinstances/{instanceName}/playbackendpoints/{peName}

Delete a Playback Endpoint

DELETE /v2/serviceinstances/{instanceName}/playbackendpoints/{peName}

Update a Playback Endpoint

PUT /v2/serviceinstances/{instanceName}/playbackendpoints/{peName}

2.7.18 Capture Endpoint

A CaptureEndpoint is a logical configuration reference point to define associate resources and policy for the capture, ingest and storage of content in VMP. One Service Instance can have one or more CaptureEndpoint objects.

Get all Capture Endpoints

GET /v2/serviceinstances/{instanceName}/captureendpoints

Returns a collection of objects

GET /v2/serviceinstances/ums-0-1/captureendpoints HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8


```
[
      {
            "name" : "ce1",
             "type" : "captureendpoints",
             "externalId" : "/v2/ serviceinstances/ums-0-1/captureendpoints/ce1", "properties" : {
                  "description" : "capture endpoint",
                  "slaType" : "resource", "resourceSLA" : {
                        "minNode" : 1,
                        "desiredNode" : 3,
                        "maxNode" : 6,
                        "maxStorage" : 0
                  },
                    "address":{ "vip":"",
                         "fqdn":"cep.cdvr.com"
                    },
                  "defaultStorageRef" : "smtenant_system.smnasstore.my-nas", "regionRef" :
                  "smtenant_system.smregion.region-0", "storagesRef" : [
                        {
                              "storageRef" : "smtenant_system.smnasstore.my-nas"
                        }
                  ]
             }
      }
]
```


Get one Capture Endpoint

GET /v2/serviceinstances/{instanceName}/captureendpoints/{name}

Create a Capture Endpoint

POST /v2/serviceinstances/{instanceName}/captureendpoints/{name}

Delete a Capture Endpoint

DELETE /v2/serviceinstances/{instanceName}/captureendpoints/{name}

Update a Capture Endpoint

PUT /v2/serviceinstances/{instanceName}/captureendpoints/{name}

2.7.19 State Cache Endpoint

A StateCacheEndpoint provides data caching functions for all the capture endpoints within the same service instance. This is required for any service instance, which has capture endpoints. A single StateCacheEndpoint can be shared among multiple capture endpoints within the same service instance.

Get State cache Endpoints

GET /v2/serviceinstances/{instanceName}/statecacheendpoints

Returns a collection of object

HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

```
. [
      {
            " name" : " state-cache-live" , " type" :
             " statecacheendpoints" ,
             " externalId" : " /v2/ serviceinstances/ums-0-
1/statecacheendpoints/state-cache-live" ,
             " properties" : {
                  " description" : " state cache for live service" , " slaType" : " resource" ,
                  " resourceSLA" : { " minNode" : 2,
                        " desiredNode" : 2,
                        " maxNode" : 2
                  },
                    " address" :{
                         " vip" :" 172.19.21.220"
                    },
                  " regionRef" : " smtenant_system.smregion.region-0”
             }
      }
]
```

Get one State Cache Endpoint

GET /v2/serviceinstances/{instanceName}/statecacheendpoints/{name}

Create a State Cache Endpoint

POST /v2/serviceinstances/{instanceName}/statecacheendpoints/{name}

Delete a State Cache Endpoint

DELETE /v2/serviceinstances/{instanceName}/statecacheendpoints/{name}

Update a State Cache Endpoint

PUT /v2/serviceinstances/{instanceName}/statecacheendpoints/{name}

2.7.20 Asset Workflow Template

An AssetWorkflowTemplate is a driving object to create assets using the specified resources, endpoints, policies, and templates. A Service Instance can have zero or more AssetWorkflowTemplates.

The template has a state to enable or disable. When the state is enabled, the assets are ingested and published. When the state is disabled, there are no active assets being ingested or published. For Live, that means no capture, indexing or playout when the state is disabled. For VOD, there are no active ingests, indexing or playout as well.

Any Storage object referenced in AssetWorkflowTemplate overrides the capture endpoint definition.

For VOD service, when the VOD source asset is a CIF asset, it is possible to have the Playback Engine retrieve the asset from the Capture engine over HTTP interface, or directly from the Capture Endpoint Storage (NAS).

If the " assetDownload" flag is set to “enabled”, the associated recording asset is downloadable.

Get all Asset Workflow Templates

GET /v2/serviceinstances/{instanceName}/assetworkflowtemplates

Returns a collection of objects

```
GET /v2/serviceinstances/ums-0-1/assetworkflowtemplates HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
[
   {
      " id" : " smserviceinstance_ums_0_1.smassetworkflowtemplate.test-awf" ,   " name" : " test-
      awf" ,
      " type" : " assetworkflowtemplates" ,
      " externalId" : " /v2/serviceinstances/ums-0-1/assetworkflowtemplates/test-
awf" ,
      " properties" : {
         " type" : " livecaptureplayback" ,
         " captureEndpointRef" : " smserviceinstance_ums_0_1.smcaptureep.test- capture-
ep" ,
         " playbackEndpointRef" : " smserviceinstance_ums_0_1.smplaybackep.test- playback-
ep" ,
         " storageRef" : " smtenant_system.smnasstore.nas-1" ,
         " assetLineupRef" : " smtenant_0.smchannellineup.ca" ,
         " mediaSourceRef" : " " ,
```

```
        " esamProfileRef" : " " ,
        “esamTemplateRef”:smtenant_0.smes
        amtemplate.new_template”,
        " assetRedundancyPolicyRef" :
" smtenant_0.smassetredundancypolicy.redundancy-pol-cos" ,  " assetLifecyclePolicyRef" :
" smtenant_0.smassetlifecyclepolicy.dvrwindow-3600" ,  " state" :
        " enabled" ,
         " assetDownload" : “enabled”,
        " assetTemplates" : [
           {

              " abrPublishTemplateRef" : " smtenant_0.smpublishtemplate.hls- template-
1"
           }
        ]
     }
  }
]
```


The supported types are: livecaptureplayback, vodcaptureplayback, cdvrcaptureplayback, cdvrcapture, playbackciforigindynamic. Custom types are not listed.

Get one Asset Workflow Template

GET /v2/serviceinstances/{instanceName}/assetworkflowtemplates/{Name}

Create an Asset Workflow Template

POST /v2/serviceinstances/{instanceName}/assetworkflowtemplates/{Name}

Delete an Asset Workflow Template

DELETE /v2/serviceinstances/{instanceName}/assetworkflowtemplates/{Name}

Update an Asset Workflow Template

PUT /v2/serviceinstances/{instanceName}/assetworkflowtemplates/{Name}

2.7.21 ServiceAPI

The ServiceAPI object is a read only object that provides available Service Control API interface hosting server and URL details. Once a Service instance is created and the AssetWorkflowTemplate is activated, the capture and playback endpoints can receive control messages. For example, for the VOD management in VOD service, the capture endpoint listens for VOD asset control and management messages.
A user can query the Service API objects to find out where the VOD APIs are hosted, for example, for a VOD service.

The following example returns the ServiceAPI object for a VOD service which supports ingest and

indexing of an asset. Note that the 'interfaces' is an array of published Service Control APIs. For VOD, the Asset Management interface is hosted on the indicated URL.

Get ServiceAPI objects in a service instance

GET /v2/serviceinstances/{instanceName}/serviceapis

Returns a collection of objects for all endpoints in the service instance
GET /v2/serviceinstances/ums-0-1/serviceapis

HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8


```
[
    {
        " name" :"  smserviceinstance_ums_0_1.smcaptureep.ce1" ,
        " id" :" smserviceinstance_ums_0_1.smserviceapi. smserviceinstance_ums_0_1
.smcaptureep.ce1" ,

        " type" :" serviceapis" ,
        " externalId" :" /v2/serviceinstances/ums-0-
1/serviceapis/smserviceinstance_ums_0_1.smcaptureep.ce1" ,  " properties" :{
            " endpointRef" :" smserviceinstance_ums_0_1.smcaptureep.ce1" ,
            " interfaces" :[
                {
                    " type" :" assetMgmt" ,
                    " url" :" http://am-ce1-
vod21.VMP.domain.com:7001/v1/assetworkflows/vod_wf/assets/" ,
                    " version" :" 1.0" ,  " assetWorkflowTemplate" :" vod_wf" ,
                    " assetWorkflowTemplateType" :" vodcaptureplayback"
                }
            ]
        }
    }
]
```


The Service API object includes one or more interfaces that can be used to control the service. Different types of interfaces can be supported by the service API model. In VMP, asset management APIs are advertised through the service API object. Interfaces for asset management can be specified for every asset workflow template. The interface URL is mapped to the asset workflow name.

The interface URL points to the REST end point that can be used to create, delete, and retrieve media asset objects. The URL is typically of the form http://<interface_host>/<interface_base_uri>. The <interface_host> is typically of the form "FQDN:PORT"

2.8 System and Service Monitoring

2.8.1 ServiceStatus

ServiceStatus is a read-only object that provides the status of a service

instance. User can use this information to see the operational status of

a service instance.

Get all Node Statuses for nodes in a region
GET /v2/serviceinstances/{instanceName}/servicestatuses

Returns a collection of objects
GET /v2/serviceinstances/ums-0-1/servicestatuses

HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8


```
[
   {
"id" : "smserviceinstance_ums_0_1.smservicestatus.service-status-ums-0-1",  "name" :
       "service-status-ums-0-1",
       "type" : "smservicestatus",
```

```
      "externalId" : "/v2/serviceinstances/ums-0-1/servicestatuses/service- status-ums-0-
1",
                  "transactionId": "20de98cc-30f2-4bc1-b269-0e19365191a8", "properties" : {
            "opStatus" : "active"
   }
]
```


2.8.2 NodeStatus

NodeStatus is a read-only object that provides the status of a Node object. It provides the state, fault detail (if any), linkage to a service instance and endpoint, the application using it, and all interface details.

User can use this information to see the usage of the provisioned Node objects for a given VMP

deployment. Get all Node Statuses for nodes in a region
GET /v2/regions/{regionName}/nodestatuses

Returns a collection of objects
GET /v2/regions/region-0/nodestatuses

HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8


```
[
   {
      "name" : "node-1",
      "id" : "smregion_region-0.smnodestatus.node-1", "type" :
      "nodestatuses",
      "externalId" : "/v2/regions/region-0/nodestatuses/node-1",
   "transactionId": "20de98cc-30f2-4bc1-b269-0e19365191a8",
"properties" : {

            "hostname" : "mpec-live_0_1-03.VMP.domain.com", "serviceInstanceRef" :
            "smtenant_0.smserviceinstance.ums_0_1", "endpointRef" :
            "smserviceinstance_ums_0_1.smplaybackep.pe0",
               "state" : "created", "faultStatus" :
               "None", "faultDetail" : "",
               "lastModifiedTime" : "2014-03-20T14:32:00.023Z", "interfaces" : [{
                  "interface" : "eth0",
                  "type" : "mgmt", "inet" :
                  "10.0.1.11",
                  "status" : "active"
               }],
               "image" : {  "imgTag" :
```

```
                    "mpe",
                    "personality" : "controller", "version" :
                    "2.1"
                },
                "configRef" : "smregion_region-0.smnode.node-1",

            }
        }
    }
]
```


Get one NodeStatus object
GET /v2/regions/{regionName}/nodestatuses/{nodeName}

2.8.3 Event

Type – each event has a type class. This filter limits the returned events to the specified type. Valid type values are Node, Application, Ext-Resource, High-Availability and Config. (The type values should use all lower case, i.e. "node application".
Event is a read-only object that provides a view into a state change inside VMP. It provides the insight into progress of tasks, warnings and errors.

User can use this information to see the system reaction to any configuration updates, runtime checks for events by severity.

Events are reported for region (SM, PAM and other common functions in a region), and service instance for MPE, MCE applications.

Get Event objects for nodes in a region
GET /v2/regions/{regionName}/events

Get Event objects for nodes in a service instance
GET /v2/serviceinstances/{instanceName}/events

Returns a collection of objects
GET /v2/serviceinstances/ums-0-1/events


```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
[{
    "name" : "mce-32406-3454-23-1394748208",
    "id" : "smserviceinstance_ums_0_1.smevent.mce-32406-3454-23-1394748208", "type" :
    "events",
    "externalId" : "/v2/serviceinstances/ums-0-1/events/mce-32406-3454-23-
1394748208",
```

```
    "properties" : { "source" : {
                "imgTag" : "mce", "personality" :
                "worker", "node" : "10.1.10.1"
            },
            "type" : "Config",
            "subType" : "Config",
            "event" : "smserviceinstance_ums_0_1.smchannellineup.national", "detailText" : "Bad
            sourceRef 0",

            "eventTime" : "1394748208",
            "eventsDropped" : 0, "severity" :
            "major", "location" : {
                "ipAddr" : "10.10.1.12",
                "processName" : "awm"
            }
        }
}]
```


Get one Event object

GET /v2/serviceinstances/{instanceName}/events/{eventName}

Event Query Filters:

The Events can be far too large to return in one query. When all Events are requested, only the last 5 minutes of events are returned. In the request, Event collection can be filtered to return for specific time range and event limit in response.

The following parameters in the query can be added to limit to a time range (both must be present):

1. startTime– timestamp of events after this time
2. endTime– timestamp of events before this time
3. severity– a string to match and filter the events by severity. Valid values are – critical, major, warning, info
4. type– each event has a type class. This filter limits the returned events to the specified type. Valid type values are Node, Application, Ext-Resource, High-Availabilityand Config.

For example:

GET /v2/serviceinstances/{instanceName}/events?startTime=2014-03-20T14:00:00.000Z&endTime=2014-03-20T14:10:00.000Z

To limit the Event Objects to a sub-range, specify the time range and add Range header in the HTTP request

Range: items=1-100

where the range can be

n-(retrieve from n and above, n is zero based)

-m(retrieve the last m items)
n-m(retrieve from n to m items)

The range query is within the bound of the startTime/endTimeif specified.

If startTimeand endTimeare absent, the range is within the bound of the entire events

available. On a single request, limit of up to 500 events can be returned.

2.8.4 AppStatus

Each Service Instance configuration instantiated certain combination of applications for each endpoint in the service instance. Each Application reports its status (via an AppStatus object) in terms of meeting the endpoint SLA and status. In addition the AppStatus includes a summary of the Nodes it is using, the number of open Events of severity Critical, Major and Warning. AppStatus is a read-only object that provides a view into the application status for each endpoint in a service instance.

User can use this information to see the system reaction to any configuration updates, runtime checks for current event count by severity.

AppStatus objects are reported for service instance for MPE, MCE applications in VMP.

Get all AppStatus objects for nodes in a service instance

GET /v2/serviceinstances/{instanceName}/appstatuses

Returns a collection of objects
GET /v2/serviceinstances/ums-0-1/appstatuses

HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8


```
[{
      " name" : " ce1.mce.10.0.1.1" ,
      " id" : " smserviceinstance_ums_0_1.smappstatus. ce1.mce.10.0.1.1" , " type" :
      " appstatuses" ,
      " externalId" : " /v2/serviceinstances/ums-0-1/appstatuses/
  ce1.mce.10.0.1.1" ,
    " properties" : {
          " appName" : " mce" ,
          " endpointRef" : " smserviceinstance_ums_0_1.smcaptureep.ce0" , " slaType" :
          " resource"
          " slaStatus" : { " nodesInUse" : 4,
              " bandwidthStatus" : " unavailable" ,
```

```
                "sessionsStatus" : "unavailable",
                "storageStatus" : "normal", "nodeStatus" :
                "normal"
            },
            "eventsOpen" : { "critical" : 0,
                "major" : 0,
                "warning" : 2
            }
        }
}]
```


2.8.5 Diagnostics

Diagnostics object can be created and defined to control the logs from the VMP components. Each diagnostics object defines a diagnostics Id (logLabel), a set of matched conditions, a desired maximum Log Level, the start time of the logging definition to be applied, interval duration for which the diagnostics definition will be effective. A Diagnostics config object can be enabled or disabled.
The logLabelgets printed in every log record to allow correlation of various logs. The Log levels allowed are one of – critical, error, warn, info, debug, trace– in the order of detail provided.
It is recommended that the log levels debug and trace be limited to a suitable duration in order to reduce the performance impact these may have when the diagnostics is effective.
Diagnostics can be set per region for the Platform domain objects and Service Management, and they can be configured per Service Instance.

The conditionsproperty is an array of condition objects. Any condition match in the array is considered a hit. Upon a hit, the log is produced (provided the logLevelmatch and enableis true). Each condition object can have a spec for app, pathor flow(corresponds to the application, Service Path and Service Path). At least one of them must be specified.

The appspec contains a typeand func. Valid app type values are sm, pam, wfm, mce, mpe, and storage. For each app type, there can be a set of functions (func) whose values will be provided with the software release.

The flowspec contains a type and an id. Valid flow types are contentedwhere the flow matches the contentID for example.

The pathspec contains a typeand inetwhere inetis an array of IP addresses. The valid path type values are
ipAddr, srcIpAddr, destIpAddr.

All spec parameters in app, flow and path are matched without case sensitivity.

Get all Diagnostics for a Service Instance

GET /v2/serviceinstances/{instanceName}/diagnostics

Returns a collection of objects

GET /v2/serviceinstances/ums-0-1/diagnostics HTTP/1.1

200 OK
Content-Type: application/json; charset=utf-8


```
[
      {
            "name" : "diag-vod.john",
            "id" : "smserviceinstance_ums_0_1.smdiagnostics.diag-vod.john", "type" :
             "smdiagnostics",
             "externalId" : "/v2/serviceinstances/ums-0-1/diag-vod.john", "properties" : {
                  "logLabel" : "DIAG.VOD",
                      "conditions" : [ {
                         "app" :     {

                              "type" : "mce",
                              "func" : "capture"
                         },
                         "flow" :      {
                              "type" : "contentId",
                              "id" : "7ceb5d3c7b44eba38172cf8c218c9ee5"
                         }
                  } ],
                  "start" : {
                        "at" : 0
                  },
                    "end" : {
                        "after" :       3600
                  }
                  "logLevel" : "debug", "enable" :
                  false
            }
      }
]
```


Get all Diagnostics for a region

GET /v2/regions/{regionName}/diagnostics

Returns a collection of objects

GET /v2/regions/region-0/diagnostics HTTP/1.1

200 OK
Content-Type: application/json; charset=utf-8

```
[
      {
            "name" : "diag-servicemgr",
            "id" : "smregion_region-0.smdiagnostics.diag-servicemgr", "type" :
             "smdiagnostics",
             "externalId" : "/v2/region-0/diagnostics/diag-servicemgr", "properties" : {
                  "logLabel" : "DIAG.001.SM",
                        "conditions" : [ {
                           "app" :     {
                             "type" : "sm",
                             "func" : "config"
                        },
                        "path" : {
                             "type" : "srcIpAddr", "inet" :
                             ["10.1.1.1"]
                        }
                  } ],
                  "start" : {
                        "at" : 0
                  },

                    "end" : {
                        "after" : 3600
                  }
                  "logLevel" : "debug", "enable" :
                  false
            }
      }
]
```


Get one Diagnostics

GET /v2/serviceinstances/{instanceName}/diagnostics/{Name} GET /v2/regions/{regionName}/diagnostics/{Name}

Create a Diagnostics

POST /v2/serviceinstances/{instanceName}/diagnostics/{Name} POST /v2/regions/{regionName}/diagnostics/{Name}

Delete a Diagnostics

DELETE /v2/serviceinstances/{instanceName}/diagnostics/{Name} DELETE /v2/regions/{regionName}/diagnostics/{Name}

Update a Diagnostics

PUT /v2/serviceinstances/{instanceName}/diagnostics/{Name} PUT

/v2/regions/{regionName}/diagnostics/{Name}

A Sample LOG output has the following format (note the location of the logLabel):

2014-06-05T18:33:11.946Z 5289 ServiceMgr server.js:27 [DIAG.001.SM] INFO -> Service manager listening on port 8043.

2.8.6 Statistics

Statistics object is a read only object that conveys accounting information of properties being maintained in the system. The statistics buckets are grouped by service instance. Inside each service instance, the statistics can be indexed by AssetWorkflowTemplate name, *CaptureEndpoint*, PlaybackEndpoint name, or by category.

The statistics objects are organized into 15 minute interval buckets. Up to the last 14 days of statistics buckets are being kept in the system. The statistics buckets can be retrieved from the management API using different filters.

Get all Statistics objects for a Service Instance

GET /v2/serviceinstances/{instanceName}/statistics

Returns a collection of objects

GET /v2/serviceinstances/ums-0-1/statistics?category=capture HTTP/1.1 200

OK
Content-Type: application/json; charset=utf-8


```
[
   {

      " id" :  " smserviceinstance_ums_0_1_statistics.smstats.capture.1412208000000" ,
      " name" : " capture.1412208000000" ,  " type" :
      " statistics" ,
      " externalId" : " /v2/serviceinstances/ums-0-
1/statistics/capture.1412208000000" ,
      " properties" : {  " category" :
         " capture" ,
         " workflowName" : " live-workflow" ,
         " endpointName" : " ce-1" , " stats" :  [
            {
               " key" : " key0" ,  " value" : 5,
               " type" : " counter"
            },
            {
               " key" : " key0_gauge" ,
               " value" : 2000,
               " type" : " gauge"
            }
         ],
         " bucketTime" : " 1412208000000"
      }
   }
```

]

Statistics Query Filters:

The Statistics objects may be too large to return in one query. When all Statistics objects are requested, only up to 500 items will be returned in a single request. In the request, Statistics collection can be filtered to return for specific time range and type in response.

The following parameters in the query can be added to specify the scope of search:

1. startTime– timestamp of statistics after this time
2. endTime– timestamp of statistics before this time
3. category– a string to match by category. Valid values are – capture, playback
4. workflowName – name of AssetWorkflowTemplate
5. endpointName – name of CaptureEndpoint or PlaybackEndpoint
6. channelId – channel identifier

For example:

GET /v2/serviceinstances/ums-0-1/statistics?category=capture&workflowName= live-workflow&startTime=2014-10-01T14:00:00.000Z&endTime=2014-10- 01T14:15:00.000Z

To limit the Statistics Objects to a sub-range, specify the time range and add Range header in the HTTP request

Range: items=1-100

where the range can be

n-(retrieve from n and above, n is zero based)
-m(retrieve the last m items)
n-m(retrieve from n to m items)

The range query is within the bound of the startTime/endTimeif specified.

If startTimeand endTimeare absent, the range is within the bound of the entire statisticsobjects

available. On a single request, limit of up to 500 statisticsobjects can be returned.

2.8.7 Alarm

Alarm provides a view into operating conditions inside VMP. Error condition will trigger alarm to be raised. The alarm will be cleared when the error condition has been recovered. An active alarm can be acknowledged, in which the user-id and time of acknowledgement will be recorded.

User can use this information to oversee the overall operating condition of the system.

Alarms are reported for region (SM, PAM and other common functions in a region), and service instance for MPE, MCE and controller applications

Get Alarm objects for nodes in a region

GET /v2/regions/{regionName}/alarms

Get Alarm objects for nodes in a service instance

GET /v2/serviceinstances/{instanceName}/alarms

Update one alarm object, only the ackUser and ackTime field will be updated

PUT /v2/serviceinstances/{instanceName}/alarms/{alarmName}

Returns a collection of objects

GET /v2/serviceinstances/ums-0-1/alarms

HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8


```
[
   {
      " id" : " smserviceinstance_ums_0_1_alarms.smalarm.awm.WorkflowFailed.awt-
live" ,
      " name" : " awm.WorkflowFailed.awt-live" ,  " type" :
      " alarms" ,
      " externalId" : " /v2/serviceinstances/ums-0-
1/alarms/awm.WorkflowFailed.awt-live" ,
      " properties" : {  " type" :
         " config" ,
         " subType" : " config" ,
         " detailText" : " workflow creation failed" ,
         " alarmName" : " awm.WorflowFailed.awt-live" ,
```

```
        " severity" : " critical" ,  " category" :
        " awm.AssetWorkflow" ,  " location" : {
            " ipAddr" : " 10.0.0.1" ,
            " processName" : " " ,
            " processId" : " "
        },
        " lastUpdateTime" : " 1412236569195" ,  " state" :
        " raise" ,
        " setTime" : " 1412236569195" ,
        " clearTime" : " 1412236559200" ,
        “ackUserId”: “admin" ,  “ackTime”:
        " 1412236669005" ,
        " source" : {
            " imgTag" : " mce-worker" ,
            " personality" : " control" ,  " node" :
            " 10.16.123.6"
        }
      }
   }
]
```


Get one alarm object

GET /v2/serviceinstances/{instanceName}/alarms/{alarmName}

Alarm Query Filters:

The Alarms may be too large to return in one query. When all Alarms are requested, up to 500 items are returned. In the request, Alarm collection can be filtered to return for specific time range and alarm limit in response.

The following parameters in the query can be added to limit to a time range (both must be present):

1. startTime– timestamp of alarms after this time
2. endTime– timestamp of alarms before this time
3. severity– a string to match and filter the alarms by severity. Valid values are – critical, major, warning, info
4. type– each alarm has a type class. This filter limits the returned alarms to the specified type. Valid type values are Node, Application, Ext-Resource, High-Availabilityand Config.
5. state – condition of the alarm. Valid state values are clear or raise.

For example:

GET /v2/serviceinstances/{instanceName}/alarms?startTime=2014-10-20T14:00:00.000Z&endTime=2014-10-20T14:10:00.000Z

To limit the Alarm Objects to a sub-range, specify the time range and add Range header in the

HTTP request

Range: items=1-100
where the range can be

n-(retrieve from n and above, n is zero based)
-m(retrieve the last m items)
n-m(retrieve from n to m items)

The range query is within the bound of the startTime/endTimeif specified.

If startTimeand endTimeare absent, the range is within the bound of the entire

alarms available. On a single request, limit of up to 500 alarms can be returned.

2.9 Media Asset Management API

Media Asset is the resource that represents the asset that is captured and packaged by the Media Origination System. Media asset is created within the context of an asset workflow template (AWT), also referred to as asset workflow in this section. Asset workflow specifies the capture and playback endpoint policies to create live, VOD or Cloud-DVR assets. The media asset, once captured, is dynamically packaged into the different ABR formats for streaming.

The media asset management API allows an external content management application (such as Scheduler or Content Management System) to control the lifecycle of media asset. The control management application is referred to as client application or client in this section.

2.9.1 API publish and discovery

Once an asset workflow is activated, the asset management API is published with the Service API object, as described in section 2.7.20. The Service API object provides the list of interfaces supported by the service instance. The interfaces have a specific type. The asset management API is published with interface type **assetMgmt**. Since the assets are created in context of an asset workflow, there is a unique asset management interface for each asset workflow. The workflow template name and type are published along with the API url.

The client application selects the asset management API by matching the asset workflow template name and interface type of **assetMgmt**.

For example, for service instance **ums-0-1**, asset workflow **cdvr-wf**, the client application discovers the asset management API as shown below.

GET /v2/serviceinstances/ums-0-1/serviceapis

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

[
    {
        " name" :"  smserviceinstance_ums_0_1.smcaptureep.ce1" ,
        " id" :" smserviceinstance_ums_0_1.smserviceapi. smserviceinstance_ums_0_1
.smcaptureep.ce1" ,
        " type" :" serviceapis" ,  " externalId" :" /v2/serviceinstances/ums1/serviceapis/
        smserviceinstance_
ums_0_1.smcaptureep.ce1" ,
        " properties" :{
            " endpointRef" :" smserviceinstance_ums_0_1.smcaptureep.ce1" ,  " interfaces" :[
                {
                    " type" :" assetMgmt" ,
                    " url" :" https://am-ce1-ums-0-
1.VMP.domain.com:7001/v1/assetworkflows/cdvr-wf/assets/" ,


                    " version" :" 1.0" ,
                    " assetWorkflowTemplate" :" cdvr-
                    wf" ,
                    " assetWorkflowTemplateType" :" cdvrcaptureplayback"
                }
            ]
        }
    }
]
```

2.9.2 Media Asset State

Each media asset has a state to reflect its current lifecycle status.

State	Description
PENDING	Asset capture is scheduled to start in future
CAPTURING	Asset capture (recording) is in progress
COMPLETE	Asset capture is complete
FAILED	Asset capture or publish failed
DELETE_COMPLETE	Asset delete completed
DELETE_FAILED	Asset delete Failure

The asset lifecycle is managed by an external content management application. The asset lifecycle state is notified to the statusCallback URL, specified by the client at the time of asset creation. In case of failure, if the client application chooses to re-create the asset, it needs to delete and create the asset again.

2.9.3 Live Asset Management API

The following operations are supported for Live asset management:

Operation	Description
Create Live Asset	HTTP POST to ingest Live asset from the given source.
Retrieve Live asset	HTTP GET to retrieve the status and other details of a Live asset.
Restart Live asset	HTTP PUT to stop and start the asset/channel.
Bulk retrieve Live asset status	HTTP GET to retrieve a collection of Live assets based on filters.
Delete Live asset	HTTP DELETE to delete Live asset from storage

2.9.3.1 Creation

This API initiates ingest of Live media into the VMP system.

Note: The Stream name should be unique.

Request:

```
POST /v1/assetworkflows/live0/assets/ HTTP/1.1
{
   " contentId" :" abc" ,
   " userData”:" {\" channelName\" :\" awm_channel\" ,\" channelId\" :\" live_channel1\" ,\”des
cription\" :\”test channel\" }”,
   " customConfigs" :[
         {
            " name" :" poisAcquisitionPointId" ,
            " value" :" abc"
         }
    ],
   " mediaSource" :{
      " ebpMode" :" true" ,
      " streams" :[
         {
            " name" :" 1980x1080p30_4500" ,
            " sourceUrl" :" udp://228.1.2.21:878/" ,
            " sourceIp" :" 10.23.1.101"
         },
         {
            " name" :" 1080x720p30_2500" ,
            " sourceUrl" :" udp://228.1.2.21:879/" ,
            " sourceIp" :" 10.23.1.101"
```

```
        }
      ]
   }
}
```

Response:

```
HTTP/1.1 201 Created
Date: Fri, 7 Oct 2005 17:17:11 GMT
Content-Type: application/json
Location:    https://am-cep2-ums-0-1.VMP.com:7001/v1/assetworkflows/live0/assets/abc
{
   " contentId" :" abc" ,
   " assetMgmtUrl" :" ,
   " customConfigs" :[
         {
            " name" :" poisAcquisitionPointId" ,
            " value" :" abc"
         }
    ],
   " mediaSource" :{
      " ebpMode" :" true" ,
      " streams" :[
         {
            " name" :" 1980x1080p30_4500" ,
            " sourceUrl" :" udp://228.1.2.21:878/" ,
            " sourceIp" :" 10.23.1.101"
         },
         {
            " name" :" 1080x720p30_2500" ,
            " sourceUrl" :" udp://228.1.2.21:879/" ,
            " sourceIp" :" 10.23.1.101"
         }

      ]
   },
   " status" :{
      " state" :" INIT"
   }
}
```

2.9.3.2 Deletion

Deleting the Live media asset from the VMP system is performed by sending a DELETE method on the asset management URL, which is returned in the location header and **assetMgmtUrl**attribute in create response.

DELETE https://am-cep1-ums-0- 1.VMP.com:7001/v1/assetworkflows/live0/assets/abc

< HTTP/1.1 202 Accepted
< Content-Type: application/json; charset=utf-8
< Content-Length: 45
< Date: Mon, 30 Jun 2014 19:55:44 GMT

Asset delete for live0/abc accepted

2.9.3.3 Retrieval

Asset management URL is used to retrieve details about a media asset.

GET /v1/assetworkflows/live0/assets/abc HTTP/1.1

HTTP/1.1 200 OK
Date: Fri, 7 Apr 2014 17:17:11 GMT
Content-Type: application/json


```
{
    " alternateContent" :{

    },
    " assetMgmtUrl" :" ,
    " contentId" :" awm_channel" ,
    " statusCallback" :{
        " url" :"
    },
    " userData" :" {\" channelName\" :\" awm_channel\" ,\" channelId\" :\" live_channel1\" ,\" de
scription\" :\" \" }" ,
    " customConfigs" :[
        {
            " name" :" poisAcquisitionPointId" , " value" :" abc"
        }
    ],
    " enableRelocation" :true,
    " mediaSource" :{
        " ebpMode" :true,



        " streams" :[
            {
                " bitrate" :" 4122075" ,
                " elementaryStreams" :[
                    {
                        " bitRate" :" 3869700" ,
```

```
                " ccType" :" 608" ,
                " format" :" avc1.674D40" ,
                " frameRate" :" 59" ,
                " height" :" 720" ,
                " pid" :" 481" ,
                " type" :" video" ,
                " width" :" 960"
            },
            {
                " bitRate" :" 127125" , " bitsPerSample" :" 16" ,
                " channelCount" :" 2" ,
                " format" :" mp4a.804C" ,
                " language" :" eng" ,
                " languageRole" :" m" ,
                " pid" :" 482" ,
                " sampleRate" :" 48000" ,
                " type" :" audio"
            },
            {
                " bitRate" :" 125250" , " bitsPerSample" :" 16" ,
                " channelCount" :" 2" ,
                " format" :" mp4a.804C" ,
                " language" :" spa" ,
                " languageRole" :" a" ,
                " pid" :" 483" ,
                " sampleRate" :" 48000" ,
                " type" :" audio"
            }
        ],
        " name" :" AWM_3M" ,
        " sourceIp" :" 12.0.0.41" ,
        " sourceUrl" :" udp://232.235.235.245:27100"
    },
    {
        " bitrate" :" 3069675" ,
        " elementaryStreams" :[
            {
                " bitRate" :" 2817300" ,
                " ccType" :" 608" ,
                " format" :" avc1.674D40" ,
                " frameRate" :" 29" ,
                " height" :" 480" ,
                " pid" :" 481" ,
                " type" :" video" ,
                " width" :" 848"
            },
            {
                " bitRate" :" 127125" ,
```

```
            " bitsPerSample" :" 16" , " channelCount" :" 2" ,
            " format" :" mp4a.804C" ,
            " language" :" eng" ,
            " languageRole" :" m" ,
            " pid" :" 482" ,
            " sampleRate" :" 48000" ,
            " type" :" audio"
        },
        {
            " bitRate" :" 125250" , " bitsPerSample" :" 16" ,
            " channelCount" :" 2" ,
            " format" :" mp4a.804C" ,
            " language" :" spa" ,
            " languageRole" :" a" ,
            " pid" :" 483" ,
            " sampleRate" :" 48000" ,
            " type" :" audio"
        }
    ],
    " name" :" AWM_2M" ,
    " sourceIp" :" 12.0.0.41" ,
    " sourceUrl" :" udp://232.235.235.245:27200"
},
{
    " bitrate" :" 2543475" ,
    " elementaryStreams" :[
        {
            " bitRate" :" 2291100" ,
            " ccType" :" 608" ,
            " format" :" avc1.674D40" ,
            " frameRate" :" 29" ,
            " height" :" 368" ,
            " pid" :" 481" ,
            " type" :" video" ,
            " width" :" 640"
        },
        {
            " bitRate" :" 127125" , " bitsPerSample" :" 16" ,
            " channelCount" :" 2" ,
            " format" :" mp4a.804C" ,
            " language" :" eng" ,
            " languageRole" :" m" ,
            " pid" :" 482" ,
            " sampleRate" :" 48000" ,
            " type" :" audio"
        },
        {
```

```
            " bitRate" :" 125250" , " bitsPerSample" :" 16" ,
            " channelCount" :" 2" ,
            " format" :" mp4a.804C" ,
            " language" :" spa" ,
            " languageRole" :" a" ,

            " pid" :" 483" ,
            " sampleRate" :" 48000" ,
            " type" :" audio"
        }
    ],
    " name" :" AWM_1500Kbps" ,
    " sourceIp" :" 12.0.0.41" ,
    " sourceUrl" :" udp://232.235.235.245:27300"
},
{
    " bitrate" :" 2017275" ,
    " elementaryStreams" :[
        {
            " bitRate" :" 1764900" ,
            " ccType" :" 608" ,
            " format" :" avc1.674D40" ,
            " frameRate" :" 29" ,
            " height" :" 272" ,
            " pid" :" 481" ,
            " type" :" video" ,
            " width" :" 480"
        },
        {
            " bitRate" :" 127125" , " bitsPerSample" :" 16" ,
            " channelCount" :" 2" ,
            " format" :" mp4a.804C" ,
            " language" :" eng" ,
            " languageRole" :" m" ,
            " pid" :" 482" ,
            " sampleRate" :" 48000" ,
            " type" :" audio"
        },
        {
            " bitRate" :" 125250" , " bitsPerSample" :" 16" ,
            " channelCount" :" 2" ,
            " format" :" mp4a.804C" ,
            " language" :" spa" ,
            " languageRole" :" a" ,
            " pid" :" 483" ,
            " sampleRate" :" 48000" ,
            " type" :" audio"
        }
    ],
```

```
            " name" :" AWM_1M" ,
            " sourceIp" :" 12.0.0.41" ,
            " sourceUrl" :" udp://232.235.235.245:27400"
        },
        {
            " bitrate" :" 5174475" ,
            " elementaryStreams" :[
                {
                    " bitRate" :" 4922100" ,
                    " ccType" :" 608" ,
                    " format" :" avc1.674D40" ,

                    " frameRate" :" 59" ,
                    " height" :" 720" ,
                    " pid" :" 481" ,
                    " type" :" video" ,
                    " width" :" 1280"
                },
                {
                    " bitRate" :" 127125" , " bitsPerSample" :" 16" ,
                    " channelCount" :" 2" ,
                    " format" :" mp4a.804C" ,
                    " language" :" eng" ,
                    " languageRole" :" m" ,
                    " pid" :" 482" ,
                    " sampleRate" :" 48000" ,
                    " type" :" audio"
                },
                {
                    " bitRate" :" 125250" , " bitsPerSample" :" 16" ,
                    " channelCount" :" 2" ,
                    " format" :" mp4a.804C" ,
                    " language" :" spa" ,
                    " languageRole" :" a" ,
                    " pid" :" 483" ,
                    " sampleRate" :" 48000" ,
                    " type" :" audio"
                }
            ],
            " name" :" AWM_4M" ,
            " sourceIp" :" 12.0.0.41" ,
            " sourceUrl" :" udp://232.235.235.245:27000"
        }
    ]
},
" output" :[
    {
        " name" :" livecaptureplayback" ,
        " packageFormat" :" hss" , " variants" :[
```

```
            {
                " audioStreams" :[

                ],
                " name" :" hd" ,
                " publishUrl" :" , " videoStreams" :[

                ]
            },
            {
                " audioStreams" :[

                ],
                " name" :" sd" ,
                " publishUrl" :" ,

                " videoStreams" :[

                ]
            },
            {
                " audioStreams" :[

                ],
                " name" :" low" ,
                " publishUrl" :" , " videoStreams" :[

                ]
            }
        ]
    },
    {
        " name" :" livecaptureplayback" ,
        " packageFormat" :" hls" ,
        " segmentDurationInSec" :2, " variants" :[
            {
                " audioStreams" :[

                ],
                " name" :" hd" ,
                " publishUrl" :" ,
                " version" :" 3" , " videoStreams" :[

                ]
            },
            {
                " audioStreams" :[

                ],
```

```
                " name" :" sd" ,
                " publishUrl" :" ,
                " version" :" 4" , " videoStreams" :[

                ]
            },
            {
                " audioStreams" :[

                ],
                " name" :" low" ,
                " publishUrl" :" ,
                " version" :" 2" , " videoStreams" :[

                ]
            }
        ]
    }

],
" status" :{ " captureStatus" :[
        {
            " captureEngine" :" 11.0.0.86" ,
            " instanceId" :" instance1" , " state" :" CAPTURING" ,
            " resourceMpd" :" http://11.0.0.86/live0/abc/MPD" , " streams" :[
                {
                    " name" :" AWM_3M" ,
                    " state" :" CAPTURING"
                },
                {
                    " name" :" AWM_1M" ,
                    " state" :" CAPTURING"
                },
                {
                    " name" :" AWM_2M" ,
                    " state" :" CAPTURING"
                },
                {
                    " name" :" AWM_1500Kbps" ,
                    " state" :" CAPTURING"
                },
                {
                    " name" :" AWM_4M" ,
                    " state" :" CAPTURING"
                }
            ]
        },
        {
            " captureEngine" :" 11.0.0.169" ,
```

```
                " instanceId" :" instance0" ,  " state" :" CAPTURING" ,
                " resourceMpd" :" http://11.0.0.169/live0/abc/MPD" ,
                " streams" :[
                    {
                        " name" :" AWM_3M" ,
                        " state" :" CAPTURING"
                    },
                    {
                        " name" :" AWM_2M" ,
                        " state" :" CAPTURING"
                    },
                    {
                        " name" :" AWM_1500Kbps" ,
                        " state" :" CAPTURING"
                    },
                    {
                        " name" :" AWM_1M" ,
                        " state" :" CAPTURING"
                    },
                    {
                        " name" :" AWM_4M" ,
                        " state" :" CAPTURING"

                    }
                ]
            }
        ],
        " startTime" :" 2014-09-19T17:44:38.389Z" ,  " state" :" CAPTURING"
    }
}

{
    " contentId" :" abc" ,
    " userData”:" {\" channelName\" :\" awm_channel\" ,\" channelId\" :\" live_channel1\" ,\”de
scription\" :\”test channel\" }”,
    " storageId" :" smtenant_system.smnasstore.0" ,

    ],
    " assetMgmtUrl" :" ,
    " mediaSource" :{
        " streams" :[
            {
                " name" :" 6mbps" ,  " sourceUrl" :" udp://228.1.2.21:878/" ,
                " sourceIp" :" 10.23.1.101" ,
                " elementaryStreams" :[
                    {
                        " pid" :" 201" ,
                        " type" :" video" ,
                        " format" :" avc1.676400" ,
```

```
            " bitRate" :" 6300000" ,
            " frameRate" :" 59" ,
            " height" :" 720" ,
            " width" :" 1280" ,
            " ccType" : " 608"
        },
        {
            " pid" :" 34" ,
            " type" :" audio" ,
            " format" :" mp4a.804C" ,
            " language" :" eng" ,
            " languageRole" :" m" ,
            " bitRate" :" 96000" ,
            " sampleRate" :" 48000" ,
            " bitsPerSample" :" 16" ,
            " channelCount" :" 2"
        },
        {
            " pid" :" 35" ,
            " type" :" audio" ,
            " format" :" mp4a.804C" ,
            " language" :" spa" ,
            " languageRole" :" a" ,
            " bitRate" :" 63750" ,
            " sampleRate" :" 48000" ,
            " bitsPerSample" :" 16" ,
            " channelCount" :" 2"
        },

        {
            " pid" :1001,
            " type" :" scte35"
        }
    ]
},
{
    " name" :" 3mbps" , " sourceUrl" :" udp://228.1.2.21:879/" ,
    " sourceIp" :" 10.23.1.101" ,
    " elementaryStreams" :[
        {
            " pid" :" 202" ,
            " type" :" video" ,
            " format" :" avc1.676400" ,
            " bitRate" :" 3000000" , " frameRate" :" 59" ,
            " height" :" 720" ,
            " width" :" 1280" ,
            " ccType" : " 608"
        },
        {
```

```
                    " pid" :" 34" ,
                    " type" :" audio" ,
                    " format" :" mp4a.804C" ,
                    " language" :" eng" ,
                    " languageRole" :" m" ,
                    " bitRate" :" 96000" , " sampleRate" :" 48000" ,
                    " bitsPerSample" :" 16" , " channelCount" :" 2"
                },
                {
                    " pid" :" 35" ,
                    " type" :" audio" ,
                    " format" :" mp4a.804C" ,
                    " language" :" spa" ,
                    " languageRole" :" a" ,
                    " bitRate" :" 63750" , " sampleRate" :" 48000" ,
                    " bitsPerSample" :" 16" , " channelCount" :" 2"
                },
                {
                    " pid" :1001,
                    " type" :" scte35"
                }
            ]
        }
    ]
},
" alternateContent" : {
    " poisUrl" :" ,
    " poisQueryWithBinaryData" :true,
    " poisVersion" :" OC-SP-ESAM-API-I01"
},

" status" :{ " state" :" CAPTURING" ,
    " startTime" :" 2014-01-22T12:00:00.000Z" ,
    " captureStatus" :[
        {
            " captureEngine" :" 10.1.1.1" , " state" :" CAPTURING" ,
            " streams" :[
                {
                    " name" :" 6mbps" ,
                    " sourceUrl" :" udp://228.1.2.21:878/" ,
                    " state" :" CAPTURING"
                },
                {
                    " name" :" 3mbps" ,
                    " sourceUrl" :" udp://228.1.2.21:879/" ,
                    " state" :" CAPTURING"
                }
            ]
        },
```

```
        {
            " captureEngine" :" 10.1.1.2" , " state" :" FAILED" ,
            " reason" :" ingest failure" ,
            " streams" :[
                {
                    " name" :" 6mbps" ,
                    " sourceUrl" :" udp://228.1.2.21:878/" ,
                    " state" :" CAPTURING"
                },
                {
                    " name" :" 3mbps" ,
                    " sourceUrl" :" udp://228.1.2.21:879/" ,
                    " state" :" FAILED" ,
                    " reason" :" ingest failure"
                }
            ]
        }
    ]
},
" output" : [
    {
        " name" :" enc" , " packageFormat" :" hls" ,
        " variants" :[
            {
                " name" :" default" ,
                " version" :" 4" , " publishUrl" :"
            },
            {
                " name" :" iphone" ,
                " version" :" 3" ,
                " publishUrl" :" , " videoStreams" :[
                    " 6300000" ,

                    " 3000000"
                ],
                " audioStreams" :[
                    " 9600"
                ]
            }
        ],
        " drmSystem" :{ " type" :" verimatrix" ,
            " name" :" vcasHls"
        },
        " httpHeaderPolicy" :{
            " manifest" :[
                {
                    " header" :" Cache-control" ,
                    " value" :" max-age=0, volatile-storage, no-store"
                }
```

```
                ],
                " chunk" :[
                    {
                        " header" :" Cache-control" ,
                        " value" :" max-age=1800"
                    }
                ]
            }
        },
        {
            " name" :" enc" , " packageFormat" :" hss" ,
            " variants" :[
                {
                    " name" :" default" , " publishUrl" :"
                },
                {
                    " name" :" xbox" ,
                    " publishUrl" :" , " videoStreams" :[
                        " 6300000" ,
                        " 3000000"
                    ],
                    " audioStreams" :[
                        " 9600"
                    ]
                }
            ],
            " drmSystem" :{
                " type" :" irdeto" ,
                " name" :" irdetoHss"
            },
            " httpHeaderPolicy" :{
                " manifest" :[
                    {
                        " header" :" Cache-control" ,
                        " value" :" max-age=0, volatile-storage, no-store"
                    }

                ],
                " chunk" :[
                    {
                        " header" :" Cache-control" ,
                        " value" :" max-age=1800"
                    }
                ]
            }
        }
    ]
}
```

2.9.3.4 Restart

This API initiates the restart of a Live media asset/channel into the VMP system
Restarting an asset/channel will stop the channel, and start the channel. The DVR window is lost on restart. PUT is updated with latest configuration from Channel Source and Channel Lineup, for live channel.

Request:

```
PUT /v1/assetworkflows/live0/assets/abc HTTP/1.1
{
   " contentId" :" abc" ,
   " userData":" {\" channelName\" :\" awm_channel\" ,\" channelId\" :\" live_channel1\" ,\"des
cription\" :\"test channel\" }",  " restart" :
   true, " customConfigs" :[
         {
              " name" :" poisAcquisitionPointId" ,
              " value" :" abc"
         }
    ],
   " mediaSource" :{
      " ebpMode" :" true" ,
      " streams" :[
         {
            " name" :" 1980x1080p30_4500" ,
            " sourceUrl" :" udp://228.1.2.21:878/" ,
            " sourceIp" :" 10.23.1.101"
         },
         {
            " name" :" 1080x720p30_2500" ,
            " sourceUrl" :" udp://228.1.2.21:879/" ,
            " sourceIp" :" 10.23.1.101"
         }
      ]
   }
}
```

Response:

```
HTTP/1.1 202 Accepted
Date: Fri, 7 Oct 2005 17:17:11 GMT
Content-Type: application/json
{
   " contentId" :" abc" ,
   " userData":" {\" channelName\" :\" awm_channel\" ,\" channelId\" :\" live_channel1\" ,\"des
cription\" :\"test channel\" }",
   " assetMgmtUrl" :" ,

    " customConfigs" :[
          {
               " name" :" poisAcquisitionPointId" ,
               " value" :" abc"
          }
     ],
```

```
    " mediaSource" :{
        " ebpMode" :" true" ,
        " streams" :[
            {
                " name" :" 1980x1080p30_4500" ,
                " sourceUrl" :" udp://228.1.2.21:878/" ,
                " sourceIp" :" 10.23.1.101"
            },
            {
                " name" :" 1080x720p30_2500" ,
                " sourceUrl" :" udp://228.1.2.21:879/" ,
                " sourceIp" :" 10.23.1.101"
            }
        ]
    },
    " status" :{
        " state" :" DELETING"
    }
}
```


2.9.3.5 Bulk Retrieval

The bulk retrieval API is common for Live, VOD and CDVR asset workflows. This is described in section 2.9.6.

2.9.4 VOD Asset Management API

The following operations are supported for VOD asset management:

Operation	Description
Create VOD asset	HTTP POST to ingest VOD asset from the given source.
Retrieve VOD asset details	HTTP GET to retrieve the status and other details of a VOD asset.
Bulk retrieve VOD asset status	HTTP GET to retrieve a collection of VOD assets based on filters.
Cancel VOD capture	HTTP DELETE to cancel an in-progress VOD capture, in PENDING or CAPTURING state
Delete VOD asset	HTTP DELETE to delete VOD asset from storage, in COMPLETE or FAILED state

2.9.4.1 Creation

This API initiates ingest of VOD media into the VMP system, which is indexed and prepared as a CIF asset. In the example, the 'interface_base_uri', which is discovered as described above.

Note: The stream name should be unique.

Request:

POST <interface_base_uri>
HTTP/1.1 Host: <interface_host>
Content-Type: application/json

Content-Length: nnn

```
{
      "contentId" : "7ceb5d3c7b44eba38172cf8c218c9ee5",
      "userData" : "XMen",
      "statusCallback" : {
         "url" : "http://cms1.cisco.com/cms/notification"
      },
      "mediaSource" : {
         "ebpMode" : "false",
         "streams" : [
            {
               "name" : "1980x1080p30_4500",
               "sourceUrl" :
               "file:///7ceb5d3c7b44eba38172cf8c218c9ee5/Xmen~7ceb5d3c7b44eba38172
               cf8c218c9ee5_1980x1080p30_4500.ts"
            },
            {
```

```
            "name" : "1080x720p30_2500",
            "sourceUrl" :
            "file:///7ceb5d3c7b44eba38172cf8c218c9ee5/Xmen~7ceb5d3c7b44eba38172
            cf8c218c9ee5_1080x720p30_2500.ts"
          }
        ]
    }
}
```


Response:


```
HTTP/1.1 201 Created
Date: Fri, 7 Oct 20 17:17:11 GMT
Content-Type: application/json
Location: http://am-ce1-
vod21.VMP.com:7001/v1/assetworkflows/vod_wf/assets/7ceb5d3c7b44eba38172cf8c21
8c9ee5
  {
    "contentId" : "7ceb5d3c7b44eba38172cf8c218c9ee5",
    "userData" : "XMen",
    "assetMgmtUrl" : "https://am-ce1-ums-0-
1.VMP.com:7001/v1/assetworkflows/vod_wf/assets/7ceb5d3c7b44eba38172cf8c218c9e
e5",
    "statusCallback" : {
      "url" : "http://cms1.cisco.com/cms/notification"
    },
    "mediaSource" : {
      "ebpMode" : "false",
      "streams" : [
        {
          "name" : "1980x1080p30_4500",

          "sourceUrl" :
" file:///7ceb5d3c7b44eba38172cf8c218c9ee5/Xmen~7ceb5d3c7b44eba38172cf8c218c9e
e5_1980x1080p30_4500.ts"
        },
        {
          "name" : "1080x720p30_2500",
          "sourceUrl" :
" file:///7ceb5d3c7b44eba38172cf8c218c9ee5/Xmen~7ceb5d3c7b44eba38172cf8c218c9e
e5_1080x720p30_2500.ts"
        }
      ]
    },
status" : {
    "state" : "INIT",
```

```
        "captureStatus" : [
            { "state" : "INIT", "reason" :
                "Initialized"
            }]
    }
  }
```


VOD request for content trimming. This has to work with a live asset, in this example ums-live/abc-10p1 In the request, resourceMpd indicates the source of the content to be processed by MCE.

Request:


```
POST <interface_base_uri>
HTTP/1.1 Host: <interface_host>
Content-Type: application/json
Content-Length: nnn

{
    "contentId":"vod4",
    "userData":"content trimming from liveAsset",
    "resourceMpd":"http://capture.xbao.com/live/4KCHAN/MPD",
    "captureStart":"2014-09-12T06:40:10.000Z",
    "captureEnd":"2014-09-12T07:10:10.000Z",
    "mediaSource":{
        "ebpMode":true,
        "streams":[
            {
                "name":"4K1",
                "sourceUrl":""
            }
        ]
    }
}
```


Response:


```
HTTP/1.1 201 Created
Date: Fri, 7 Oct 20 17:17:11 GMT
Content-Type: application/json
Location: http://am-ce1-vod21.VMP.com:7001/v1/assetworkflows/vod/assets/vod4
  {
    "contentId" : "vod4",
    "userData" : "content trimming from liveAsset",
    "resourceMpd" :
```

```
      " http://capture.xbao.com/live/4KCHAN/MPD" , " captureStart"
      : " 2014-09-12T06:40:10.000Z" , " captureEnd" : " 2014-09-
      12T07:10:10.000Z" ,
      " assetMgmtUrl" : " https://am-ce1-ums-0-
1.VMP.com:7001/v1/assetworkflows/vod/assets/vod4" , " statusCallback" : {
         " url" : " http://cms1.cisco.com/cms/notification"
      },
      " mediaSource" : {
         " ebpMode" : " false" ,
         " streams" : [
            {
               " name" : " 1980x1080p30_4500" ,
               " sourceUrl" : " "
            },
            {
               " name" : " 1080x720p30_2500" ,
               " sourceUrl" : " "
            }
         ]
      },
status" : {
      " state" : " INIT" ,
           " captureStatus" : [
              { " state" : " INIT" , " reason" :
                 " Initialized"
              }]
    }
  }
```


The detail recorded asset information can be retrieved by HTTP GET request to the assetMgmtUrl.

Request
GET
https://am-ce1-ums-0-1.VMP.com:7001/v1/assetworkflows/vod/assets/vod4

Response:


```
{
   array :{
     contentId : vod4 ,
     userData : content trimming from liveAsset ,


     codec : H264 ,
     customConfigs :[
```

```
],
 storageId : smtenant_system.smnasstore.FREELIVE ,
 enableRelocation :true,
 assetMgmtUrl : https://am-ce1-ums-0-
 1.xbao.com:7001/v1/assetworkflows/vod/assets/vod4 , alternateContent :{

},
 mediaSource :{
   ebpMode :true,
   streams :[
    {
       name : 4K1 ,
       sourceUrl :  ,
       bitrate : 910125 ,
       elementaryStreams :[
        {
           pid : 205 ,
           type : video
           ,
           format : avc1.6742E015966281004B60284040405000003E90000EA60940 ,
           bitRate : 750000 ,
           frameRate : 3003
           , height : 288 ,
           width : 512
        },
        {
           pid : 34 ,
           type : audio
           ,
           format : mp4a.804c ,
           bitRate : 96000 ,
           language : eng ,
           languageRole : m ,
           sampleRate : 48000 ,
           bitsPerSample : 16 ,
           channelCount : 2
        },
        {
           pid : 35 ,
           type : audio
           ,
           format : mp4a.804c ,
           bitRate : 64125 ,
           language : spa ,
           languageRole : a ,
           sampleRate : 48000 ,
```

```
            bitsPerSample : 16 ,
            channelCount : 2
          },
          {
            pid : 1001 ,

            type : scte35
          }
        ]
      }
    ]
},
 output :[
  {
     name : vod ,
     publishTemplateName : publish_template_hls_clear ,
     packageFormat : hls ,
     variants :[
      {
         name : mac ,
         publishUrl :
         http://vod.play.xbao.com/vod/vod4/mac.m3u8 , version :
         4 ,
         videoStreams :[

        ],
         audioStreams :[

        ]
      },
      {
         name : default ,
         publishUrl :
         http://vod.play.xbao.com/vod/vod4/vod4.m3u8 , version :
         4
      }
    ],
     segmentDurationInSec :10,
     httpHeaderPolicy :{
       manifest :[
        {
           header : Cache-control
           , value : max-age=0
        }
      ],
       chunk :[
        {
           header : Cache-control
```

```
            , value : max-age=0
          }
        ]
      }
    }
  ],
      " captureStart" :" 2014-09-12T06:40:10.000Z" ,
   captureEnd : 2014-09-12T07:10:10.000Z ,
   resourceMpd :
   http://capture.xbao.com/live/4KCHAN/MPD , status :{
     state : COMPLETE ,

     reason :  ,
     startTime : 2014-09-12T06:40:09.675Z ,
     endTime : 2014-09-12T07:10:10.804Z ,
     captureStatus :[
      {
        instanceId : instance0
        , state : COMPLETE ,
        storageSizeBytes
        :187985846, streams :[
         {
           name : 4K1 ,
           state : COMPLETE ,
           fileSize : 187944336
           ,
           downloadUrl :
           http://capture.xbao.com/FILEDOWNLOAD/vod/vod4/4K1_0x.ts , bitrate :
           910125
         }
       ],
        captureEngine : 100.7.41.162
     }
   ]
  }
 }
}
```

2.9.4.2 Deletion

Deleting the VOD media asset from the VMP system is performed by sending a DELETE method on the asset management URL, which is returned in the location header and **assetMgmtUrl**attribute in create response.

DELETE /<interface_base_uri>/7ceb5d3c7b44eba38172cf8c218c9ee5 HTTP/1.1 Host: <interface_host>

HTTP/1.1 200 OK
Date: Fri, 7 Oct 2005 17:17:11 GMT

2.9.4.3 Retrieval

Asset management URL is used to retrieve details about a media asset. Deleting the VOD media asset from the VMP system is performed by sending a DELETE method on the asset management URL, which is returned in the location header and **assetMgmtUrl**attribute in create response.

GET <interface_base_uri>/{contentId} Host: <interface_host>

HTTP/1.1 200 OK
Date: Fri, 7 Oct 2005 17:17:11 GMT
Content-Type: application/json
Content-Length: 100


```
  {
     "contentId" : "7ceb5d3c7b44eba38172cf8c218c9ee5",
     "userData" : "XMen",
     "assetMgmtUrl" : "https://am-ce1-ums-0-
1.VMP.com:7001/v1/assetworkflows/vod_wf/assets/7ceb5d3c7b44eba38172cf8c218c9e
e5",
     "statusCallback" : {
        "url" : "http://cms1.cisco.com/cms/notification"
     },
     "mediaSource" : {
        "ebpMode" :
        "false", "streams" :
        [
           {
              "name" : "1980x1080p30_4500",
              "sourceUrl" :
"file:///7ceb5d3c7b44eba38172cf8c218c9ee5/Xmen~7ceb5d3c7b44eba38172cf8c218c9e
e5_1980x1080p30_4500.ts"
           },
```

```
            {
                "name" : "1080x720p30_2500",
                "sourceUrl" :
"file:///7ceb5d3c7b44eba38172cf8c218c9ee5/Xmen~7ceb5d3c7b44eba38172cf8c218c9e
e5_1080x720p30_2500.ts"
            }
          ]
       },
       status" : {
          "state" : "COMPLETE",
          "endTime" : "2005-10-06T19:49:46.808Z",
          "startTime" : "2005-10-06T19:49:41.548Z",
          "captureStatus" : [
              { "captureEngine" : "11.0.0.5", "state" :
                    "COMPLETE"
           }]
       },
       "output" : [
            {
                    "type" : "hls",
                    "url" :
"http://origin.sp.com/vod_wf/7ceb5d3c7b44eba38172cf8c218c9ee5/hd.m3u8"
            },
            {
                    "type" : "hls",
                    "url" : "
http://origin.sp.com/vod_wf/7ceb5d3c7b44eba38172cf8c218c9ee5/sd.m3u8", "version" :
                    "3"
            },

            {
                    "type" : "hss",
                    "url" : "
http://origin.sp.com/vod_wf/7ceb5d3c7b44eba38172cf8c218c9ee5/7ceb5d3c7b44eba3
8172cf8c218c9ee5.ism/Manifest"
            }
       ]
   }
```


2.9.4.4 Asset notification callback

If asset notification callback is specified in the asset create input json("statusCallback") then AWM will POST the notification to the callback url whenever the asset state is moved from one state to another.

2.9.4.4.1 Asset moved to pending state


```
{
    " workflowId" :" vodcaptureplayback" ,
    " contentId" :" CiscoTest1" ,
    " userData" :" Test description for VOD asset" ,
    " assetMgmtUrl" :" ,
    " status" :{ " captureStatus" :[

        ],
        " state" :" PENDING
        "
    }
}
```


2.9.4.4.2 Asset moved to capturing state


```
{
    " workflowId" :" vodcaptureplayback" ,
    " contentId" :" CiscoTest1" ,
    " userData" :" Test description for VOD asset" ,
    " assetMgmtUrl" :" ,
    " status" :{ " captureStatus" :[
            {
                " instanceId" :" instance0" ,
                " state" :" CAPTURING" ,
                " streams" :[
                    {
                        " name" :" 1mbps" ,
                        " state" :" CAPTURING"
                    },
                    {
                        " name" :" 2mbps" ,
                        " state" :" CAPTURING"
                    },
                    {
                        " name" :" 3mbps" ,
                        " state" :" CAPTURING"
                    },
                    {
                        " name" :" 4mbps" ,

                        " state" :" CAPTURING"
                    }
                ],
                " captureEngine" :" 11.0.0.81"
            }
        ],
        " state" :" CAPTURING" ,
        " startTime" :" 2015-01-07T20:18:10.657Z"
```

```
    }
}
```

2.9.4.4.3 Asset moved to complete state

```
{
    " workflowId" :" vodcaptureplayback" , " contentId" :" CiscoTest1" ,
    " userData" :" Test description for VOD asset" , " assetMgmtUrl" :" ,
    " status" :{ " captureStatus" :[
            {
                " instanceId" :" instance0" , " state" :" COMPLETE" ,
                " storageSizeBytes" :1197514595,
                " streams" :[
                    {
                        " name" :" 1mbps" ,
                        " state" :" COMPLETE" ,
                        " fileSize" :" 127765897"
                    },
                    {
                        " name" :" 2mbps" ,
                        " state" :" COMPLETE" ,
                        " fileSize" :" 225757381"
                    },
                    {
                        " name" :" 3mbps" ,
                        " state" :" COMPLETE" ,
                        " fileSize" :" 356526124"
                    },
                    {
                        " name" :" 4mbps" ,
                        " state" :" COMPLETE" ,
                        " fileSize" :" 487431109"
                    }
                ],
                " captureEngine" :" 11.0.0.81"
            }
        ],
        " endTime" :" 2015-01-07T20:18:21.317Z" ,
        " startTime" :" 2015-01-07T20:18:10.657Z" ,
        " state" :" COMPLETE"
    }
}
```

2.9.4.4.4 Asset moved to failed state

If capture engine node is not able to read the input file from the storage


```
{
    " workflowId" :" vodcaptureplayback" , " contentId" :" vod_asset3" ,
    " userData" :" Test description for VOD asset" , " assetMgmtUrl" :" ,
    " status" :{ " captureStatus" :[
            {
                " instanceId" :" instance0" , " state" :" FAILED" ,
                " reason" :" Unknown:IOM_UV_ERROR" , " streams" :[
                    {
                        " name" :" 664kbps" ,
                        " state" :" FAILED" ,
                        " reason" :" Unknown:IOM_UV_ERROR"
                    },
                    {
                        " name" :" 1mbps" ,
                        " state" :" FAILED" ,
                        " reason" :" Capture Error"
                    },
                    {
                        " name" :" 2mbps" ,
                        " state" :" FAILED" ,
                        " reason" :" Unknown:IOM_UV_ERROR"
                    },
                    {
                        " name" :" 3mbps" ,
                        " state" :" FAILED" ,
                        " reason" :" Unknown:IOM_UV_ERROR"
                    }
                ],
                " captureEngine" :" 11.0.0.80"
            }
        ],
        " state" :" FAILED"
    }
}
```


If capture engine node is not reachable internally to create the asset


```
{
    " workflowId" :" vodcaptureplayback" , " contentId" :" vod_asset10" ,
    " userData" :" Test description for VOD asset" , " assetMgmtUrl" :" ,
    " status" :{ " captureStatus" :[

        ], " state" :" FAILED" ,
        " reason" :" vod_asset10: asset create failed, CaptureEndPoint cep1 task create failed: POST:
http://11.0.0.85:5001/api/mce/tasktask create failed after retry, rspTime 5036 ms, connect ECONNREFUSED"
```

```
    }
}
```

2.9.4.4.5 Asset delete completed

```
{
    " workflowId" :" vodcaptureplayback" , " contentId" :" CiscoTest1" ,
    " userData" :" Test description for VOD asset" ,
    " assetMgmtUrl" :" ,
    " status" :{ " captureStatus" :[
        ],
        " endTime" :" 2015-01-08T20:34:41.274Z" ,
        " startTime" :" 2015-01-08T20:34:31.383Z" ,
        " state" :" DELETE_COMPLETE"
    }
}
```

2.9.4.5 Bulk Retrieval

The bulk retrieval API is common for Live, VOD and CDVR asset workflows. This is described in section 2.9.6.

2.9.5 CDVR Asset Management API

This release of VMP provides two interfaces for CDVR Asset Management.

1) AMR2 – These are the APIs that are an extension of the VMP CDVR APIs
2) AMR1 – These are the APIs that are used when interfacing with the Cisco Recorder Manager.

2.9.5.1 AMR2 APIs

The following Cloud DVR (CDVR) asset management operations are supported:

Operation	Description
Create (schedule)	HTTP POST to schedule a recording from the given source, at the specified startTime and endTime.
Modify recording asset	HTTP PUT to modify a recording. See details below regarding the allowed modifications.
Retrieve recording status	HTTP GET to retrieve the status and other details of a recording.
Bulk retrieve recording status	HTTP GET to retrieve a collection of recording assets based on filters.
Cancel recording asset	HTTP DELETE to cancel in-progress recording, in PENDING or CAPTURING state
Delete CDVR asset	HTTP DELETE to remove CDVR asset, in COMPLETED or FAILED state

2.9.5.1.1 Create (Schedule) Recording asset

This API is used to schedule Live-To-VOD or CDVR recordings on the VMP system. The asset is created based on the resources and policies specified in the asset workflow template.

Note: The stream name should be unique.

POST request contains the payload of the recording request.

POST response body has recording asset status, when asset is created successfully. The following HTTP response codes could be returned:

201 CREATED: The recording asset is created. The handler to asset created is passed in the location headers. Its also returned as assetMgmtUrl in the response body.
400 Bad Request: Schema error, mandatory parameters missing.
500 Internal Server Error: System encountered error processing this request.

Here is an example of create recording asset from linear multicast source.

Request:

POST https://am-ce1-ums-0-1.VMP.domain.com:7001/v1/assetworkflows/cdvr_wf/assets
HTTP/1.1 Host: <interface_host>
Content-Type: application/json
Content-Length: nnn


```
{
    " contentId" :" 7ceb5d3c7b44eba38172cf8c218c9ee5" ,
    " statusCallback" :{
        " url" :"  http://cms1.cisco.com/cms/notification"
    },
    " captureStart" :" 2014-07-10T19:00:00Z" ,
    " captureEnd" :" 2014-07-10T19:30:00Z" ,
    " mediaSource" :{
        " ebpMode" :" true" ,
        " streams" :[
            {
                " name" :" 1980x1080p30_4500" ,
                " sourceUrl" :" udp://228.1.2.21:878/" ,
                " sourceIp" :" 10.23.1.101"
            },
            {
                " name" :" 1080x720p30_2500" ,
                " sourceUrl" :" udp://228.1.2.21:879/" ,
                " sourceIp" :" 10.23.1.101"
            }
```

```
        ]
    }
}
```

Response:

```
HTTP/1.1 201 Created
Date: Fri, 7 Oct 2014 17:17:11 GMT
Content-Type: application/json
Location: http://am-ce1-
ums1.VMP.domain.com:7001/v1/assetworkflows/cdvr_wf/assets/7ceb5d3c7b44eba3817
2cf8c218c9ee5

{
    " contentId" :" 7ceb5d3c7b44eba38172cf8c218c9ee5" ,
    " assetMgmtUrl" : http://am-ce1-
ums1.VMP.domain.com:7001/v1/assetworkflows/cdvr_wf/assets/7ceb5d3c7b44eba3817
2cf8c218c9ee5,

    " statusCallback" :{
        " url" :"  http://cms1.cisco.com/cms/notification"
    },
    " captureStart" :" 2014-07-10T19:00:00Z" ,
    " captureEnd" :" 2014-07-10T19:30:00Z" ,
    " mediaSource" :{
        " ebpMode" :" true" ,
        " streams" :[
            {
                " name" :" 1980x1080p30_4500" ,
                " sourceUrl" :" udp://228.1.2.21:878/" ,
                " sourceIp" :" 10.23.1.101"
            },
            {
                " name" :" 1080x720p30_2500" ,
                " sourceUrl" :" udp://228.1.2.21:879/" ,
                " sourceIp" :" 10.23.1.101"
            }
        ]
    },
    status"  : {
        " state"  : " PENDING"
    }
}
```

2.9.5.1.2 Modify Recording

A CDVR Asset can be updated by performing a PUT instead of a POST to the same asset management interface API end point that was used to create the asset. The asset lifecycle state (CAPTURING, COMPLETE, FAILED, DELETE_COMPLETE or DELETE_FAILED) is notified to the statusCallback URL, specified by the client application at the time of asset create.

Only the following updates are allowed.

1) captureStart – Start time of a scheduled recording can be modified, only if the capture has not started.
2) End Time –. End time if a recording can be modified, if the capture is not complete.

In case of failure, if the client application chooses to re-create the asset, it needs to delete and create the asset again.

HTTP PUT request contains the updated asset with the modified captureStart or captureEnd time. The following HTTP response codes could be returned:

202 ACCEPTED: The recording asset modification is accepted. 400 Bad Request: Schema error, mandatory parameters missing. 404 Not Found: The requested recording asset is not found

500 Internal Server Error: System encountered error processing this request.

Here is an example of modify capture end time of scheduled recording asset.

Request:

PUT http://am-ce1-ums1.VMP.domain.com:7001/v1/assetworkflows/cdvr_wf/assets/7ceb5d3c7b44eba38172cf8c218c9ee5 HTTP/1.1
Host: <interface_host> Content-Type: application/json Content-Length: nnn


```
{
    " contentId" :" 7ceb5d3c7b44eba38172cf8c218c9ee5" ,
    " statusCallback" :{
        " url" :" http://cms1.cisco.com/cms/notification"
    },
    " captureStart" :" 2014-07-10T19:00:00Z" ,
    " captureEnd" :" 2014-07-10T19:30:00Z" ,
    " mediaSource" :{
        " ebpMode" :" true" ,
        " streams" :[
            {
                " name" :" 1980x1080p30_4500" ,
                " sourceUrl" :" udp://228.1.2.21:878/" ,
                " sourceIp" :" 10.23.1.101"
```

```
            },
            {
                "name":"1080x720p30_2500",
                "sourceUrl":"udp://228.1.2.21:879/",
                "sourceIp":"10.23.1.101"
            }
        ]
    }
}
```

Response:

HTTP/1.1 202 Accepted
Date: Fri, 7 Oct 2014 17:17:11 GMT
Content-Type: application/json

2.9.5.1.3 Cancel and Delete Recording

Cancel or deletion of recording asset from the VMP system is performed by sending a HTTP DELETE method on the asset management URL, which is returned in the location header and **assetMgmtUrl**attribute in create response.

Depending upon the state of recording asset, the following action is taken by the VMP system.

Delete asset in PENDING state.	If the recording asset is in PENDING state, the capture has not started. Delete the recording event from VMP system.
Delete asset in CAPTURING state.	If the recording asset is in CAPTURING state, delete will stop the recording. The asset remains on the storage, and is moved to
Delete asset in COMPLETE state.	If the recording asset is in COMPLETE state, delete will remove the asset from storage.
Delete asset in FAILED state.	If the recording asset is in FAILED state, delete will remove the asset from storage.

DELETE /v1/assetworkflows/awt-wf/assets/7ceb5d3c7b44eba38172cf8c218c9ee5 HTTP/1.1
Host: <interface_host>

HTTP/1.1 200 OK
Date: Fri, 7 Oct 2005 17:17:11 GMT

2.9.5.1.4 Retrieval

Asset management URL is used to retrieve status and other details about a recording asset. The "downloadUrl" attribute shows where the recording asset can be downloaded. The "downloadUrl" attribute is applicable only if the "assetDownload" flag is set to "enabled" in the asset workflow template (see Asset Workflow Template section).

GET / v1/assetworkflows/cdvrcaptureplayback/assets/f9af957a1544f7 Host:
<interface_host>

```
HTTP/1.1 200 OK
Date: Fri, 7 Oct 2005 17:17:11 GMT
Content-Type: application/json
Content-Length: nnn
{
    " contentId" :" f9af957a1544f7" ,
    " customConfigs" :[

    ],
    " storageId" :" smtenant_0.smcosstore.cosStore" ,
    " enableRelocation" :true,  " assetMgmtUrl" :" https://am-
    cep1-ums-0-
1.awmha.com:7001/v1/assetworkflows/cdvrcaptureplayback/assets/f9af957a1544f7"
,
    " alternateContent" :{

    },
    " mediaSource" :{  " streams" :[
            {
                " name" :" Profile1" ,
                " sourceUrl" :" udp://232.235.235.245:27100" ,
                " sourceIp" :" " ,
                " bitrate" :" 4117628" ,
                " elementaryStreams" :[
                    {
                        " pid" :" 481" ,
                        " type" :" video" ,

" format" :" avc1.674D401F96560780B761C81000003E90000EA60E220005B8D0000DBB87F18E
0ED0A149C0" ,
                        " bitRate" :" 3869753" ,
                        " frameRate" :" 1501" ,
                        " height" :" 720" ,
                        " width" :" 960"
                    },
                    {
                        " pid" :" 482" ,
                        " type" :" audio" ,
                        " format" :" mp4a.804c" ,
                        " bitRate" :" 134625" ,
                        " language" :" eng" ,
                        " languageRole" :" m" ,
                        " sampleRate" :" 48000" ,
                        " bitsPerSample" :" 16" ,
```

```
                    " channelCount" :" 2"
                },
                {
                    " pid" :" 483" ,
                    " type" :" audio" ,
                    " format" :" mp4a.804c" ,
                    " bitRate" :" 113250" ,
                    " language" :" spa" ,
                    " languageRole" :" a" ,
                    " sampleRate" :" 48000" ,
                    " bitsPerSample" :" 16" ,
                    " channelCount" :" 2"
                },
                {
                    " pid" :" 0" ,
                    " type" :" scte35"
                }
            ]
        },
        {
            " name" :" Profile2" ,
            " sourceUrl" :" udp://232.235.235.245:27200" ,
            " sourceIp" :" " ,
            " bitrate" :" 3061106" ,
            " elementaryStreams" :[
                {

                    " pid" :" 481" ,
                    " type" :" video" ,

" format" :" avc1.674D401F965606A1ED80A0400000FA40003A983888003D090000F4247F18E0
ED0A149C" ,
                    " bitRate" :" 2817356" ,
                    " frameRate" :" 3003" ,
                    " height" :" 480" ,
                    " width" :" 848"
                },
                {
                    " pid" :" 482" ,
                    " type" :" audio" ,
                    " format" :" mp4a.804c" ,
                    " bitRate" :" 117000" ,
                    " language" :" eng" ,
                    " languageRole" :" m" ,
                    " sampleRate" :" 48000" ,
                    " bitsPerSample" :" 16" ,
```

```
                    " channelCount" :" 2"
                },
                {
                    " pid" :" 483" ,
                    " type" :" audio" ,
                    " format" :" mp4a.804c" ,
                    " bitRate" :" 126750" ,
                    " language" :" spa" ,
                    " languageRole" :" a" ,
                    " sampleRate" :" 48000" ,
                    " bitsPerSample" :" 16" ,
                    " channelCount" :" 2"
                },
                {
                    " pid" :" 0" ,
                    " type" :" scte35"
                }
            ]
        },
        {
            " name" :" Profile3" ,
            " sourceUrl" :" udp://232.235.235.245:27300" ,
            " sourceIp" :" " ,
            " bitrate" :" 2521414" ,
            " elementaryStreams" :[
                {
                    " pid" :" 481" ,
                    " type" :" video" ,

" format" :" avc1.674D401F965605017FCB80A0400000FA40003A983888002DC60000B7187F18
E0ED09129C" ,
                    " bitRate" :" 2291164" ,
                    " frameRate" :" 3003" ,

                    " height" :" 368" ,
                    " width" :" 640"
                },
                {
                    " pid" :" 482" ,
                    " type" :" audio" ,
                    " format" :" mp4a.804c" ,
                    " bitRate" :" 117000" ,
                    " language" :" eng" ,
                    " languageRole" :" m" ,
                    " sampleRate" :" 48000" ,
                    " bitsPerSample" :" 16" ,
```

```
                    " channelCount" :" 2"
                },
                {
                    " pid" :" 483" ,
                    " type" :" audio" ,
                    " format" :" mp4a.804c" ,
                    " bitRate" :" 113250" ,
                    " language" :" spa" ,
                    " languageRole" :" a" ,
                    " sampleRate" :" 48000" ,
                    " bitsPerSample" :" 16" ,
                    " channelCount" :" 2"
                },
                {
                    " pid" :" 0" ,
                    " type" :" scte35"
                }
            ]
        }
    ]
},
" output" :[
    {
        " name" :" cdvrcaptureplayback" ,
        " publishTemplateName" :" no_variant_template" ,  " packageFormat" :" hls" ,
        " variants" :[
            {
                " name" :" default" ,

" publishUrl" :" http://cdvrcaptureplayback.ums.awmha.com/cdvrcaptureplayback/f9
af957a1544f7/f9af957a1544f7.m3u8" ,
                " version" :" 4"
            }
        ],
        " segmentDurationInSec" :2
    }
],
" captureStart" :" 2015-02-10T21:32:00.000Z" ,
" captureEnd" :" 2015-02-10T21:38:00.000Z" ,

" status" :{
    " state" :" COMPLETE" ,
    " reason" :" " ,
    " startTime" :" 2015-02-10T21:32:00.000Z" ,
    " endTime" :" 2015-02-11T05:15:00.256Z" ,
    " captureStatus" :[
```

```
            {
                " instanceId" :" instance0" ,
                " state" :" COMPLETE" ,
                " storageSizeBytes" :424210889,
                " streams" :[
                    {
                        " name" :" Profile1" ,
                        " state" :" COMPLETE" ,
                        " fileSize" :" 181607572" ,
                        " recordingDuration" :" 368" ,
" downloadUrl" :" http://cep.awmha.com/FILEDOWNLOAD/cdvrcaptureplayback/f9af957a
1544f7/instance0/Profile1_0x.ts" , // will be included only if asset download  is enabled on the asset
workflow template.
                        " bitrate" :" 4117628"
                    },
                    {
                        " name" :" Profile2" ,
                        " state" :" COMPLETE" ,
                        " fileSize" :" 133327007" ,
                        " recordingDuration" :" 368" ,

" downloadUrl" :" http://cep.awmha.com/FILEDOWNLOAD/cdvrcaptureplayback/f9af957a
1544f7/instance0/Profile2_0x.ts" ,
                        " bitrate" :" 3061106"
                    },
                    {
                        " name" :" Profile3" ,
                        " state" :" COMPLETE" ,
                        " fileSize" :" 109230786" ,
                        " recordingDuration" :" 368" ,
" downloadUrl" :" http://cep.awmha.com/FILEDOWNLOAD/cdvrcaptureplayback/f9af957a
1544f7/instance0/Profile3_0x.ts" , // downloadUrl will be added only if asset  download is enabled
on asset workflow template page
                        " bitrate" :" 2521414"
                    }
                ],
                " captureEngine" :" 11.0.0.80"
            }
        ]
    },
    " stationId" :" 1"
}
```

2.9.5.1.5 Bulk Retrieval

The bulk retrieval API is common for Live, VOD, and CDVR asset workflows. This is described in section 2.9.6.

2.9.5.2 AMR1 APIs

2.9.5.2.1 CDVR AssetWorkflow creation for AMR1

The CDVR AssetWorkflow creation for AMR-1 is identical to CDVR Workflow creation for AMR2 with the addition of the control interface in the assetworkflow template.

```
" controlInterfaces" : [
   {
      " type" : " amr1-control" ,
      " url" :
      " http://localhost:8250"
   }
],
```

2.9.5.2.2 AMR1 API Discovery

The AMR1 controlUrl is the API end point used to create Recording Events can be discovered through the Recorder Info object sent out by the Recorder. This will be sent periodically once an Assetworkflow is created. This controlUrl should prepended to the HTTP REST endpoints specified below for each AMR1 operation.

The AMR1 deliveryUrl specifies the template used to construct the delivery URL.

```
<?xml version="1.0" encoding="UTF-8"?>
<RecorderInfo
    controlApiVersion="1.0"
    controlUrl="http://am-0-ums-0-1.VMP.com:8251/amr1/assetworkflows/amCdvr0"
    deliveryUrl="http://am-0-ums-0-1.VMP.com:8251/amCdvr0/{CONTENT_ID}/MPD"
    regionId="region-0"
    totalRecordingBandwidth="6291456"
    usedRecordingBandwidth="0"
    recordingStatus="OPERATIONAL"
    totalDeliveryBandwidth="6291456"
    usedDeliveryBandwidth="0"
    numActiveDeliveryStreams="0"
    deliveryStatus="OPERATIONAL"
    maxStorageCapacity="1048576"
    usedStorageCapacity="0"
    storageStatus="OPERATIONAL"
    numActiveRecordings="0" />
```

2.9.5.2.3 Create Recording Event

Recording Event can be created by POSTing the RecordingEvent XML to the REST endpoint

<control_url>/recorder/capture/RecordingEvent

```
POST <control_url>/recorder/capture/RecordingEvent HTTP/1.1

<?xml version="1.0" encoding="UTF-8"?>
<RecordingEvent CID="abcde12345" codec="H264" copyType="common"
             startTime="2012-11-20T09:00:00Z" endTime="2012-11-20T10:00:00Z"
             stationId="nbc.west" recordingContext="context">
   <rmNotifyUrl recorderNotify="/RecorderManager/recorder"
                recordingNotify="/RecorderManager/capture" />
   <Profile name="6mbps" bitrate="6300000">
      <Source sourceURL="udp://128.1.23.111:800/" sourceIP="10.22.2.101" />
   </Profile>
   <Profile name="3mbps" bitrate="3100000">
       <Source sourceURL="udp://128.1.25.111:845/" sourceIP="10.22.2.103" />
   </Profile>
</RecordingEvent>
```

POST response does not have a body. The following HTTP response codes could be returned:

202 ACCEPTED: The RecordingEvent request is accepted

400 Bad Request: Syntax failed, or unknown capture source identifiers.

500 Internal Server Error: System encountered error processing this request.

2.9.5.2.4 Modify Recording Event

A RecordingEvent, which is already scheduled, can be modified. Only the following attributes of the RecordingEvent can be modified:

- endTime

If the endTime is smaller than or equal to the current time, the recording will be stopped immediately.

The same URI used to create a RecordingEvent is used to modify it.

<control_url>/recorder/capture/RecordingEvent

The following HTTP response codes could be returned:

202 ACCEPTED: The RecordingEvent modify request is accepted. 400 Bad Request: Syntax failed, or unknown capture source identifiers. 404 Not Found: The RecordingEvent is not found.

2.9.5.2.5 Get Recording Event Status

The status of a Recording Event can be retrieved by performing a GET on the URL –

<control_url>/recorder/capture/RecordingEvent/{CID}

The following HTTP response codes could be returned:

200 OK: The Recording Event is found and returned successfully
404 Not Found: The Recording Event is not found.
An example of XML status returned is below –

```
<?xml version="1.0" encoding="UTF-8"?>
<RecordingEvent>
   <CID>abcde12345</CID>
   <ProfileStatus profileName="6mbps" state="CAPTURING" actualBitRate="6459750" />
   <ProfileStatus profileName="3mbps" state="CAPTURING" actualBitRate="3259750" />
</RecordingEvent>
```

2.9.5.2.6 Cancel Recording Event

A Recording Event can be cancelled by issuing a HTTP DELETE on the RecordingEvent URL.

<control_url>/recorder/capture/RecordingEvent/{CID}

The following HTTP response codes could be returned:

200 OK: The Recording Event is canceled successfully 404
Not Found: The Recording Event is not found.
Get Recording Event List

The list of all recordings scheduled on a recorder can be obtained by a GET on the url –

<control_url>/recorder/capture/RecordingEventList

This list can be filtered on –

1) state of the recording using query parameter “state”
2) start time using query parameter “timeRangeStart”
3) end time using query parameter

“timeRangeEnd” The following HTTP response

codes could be returned:

200 OK: The RecordingEventList is returned successfully
400 Bad Request: Parameters are missing or attribute name
are wrong. 500 Internal Server Error

```
GET <control_url>/recorder/capture/RecordingEventList?state=PENDING HTTP/1.1
```

```
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: text/html; charset=utf-8
Content-Length: 3238
ETag: "1221151775”
Date: Fri, 12 Sep 2014 22:14:09 GMT
Connection: keep-alive

<?xml version="1.0" encoding="UTF-8"?>
<RecordingEventList>
   <RecordingEventItem CID="recordingList_0" state="PENDING" />
   <RecordingEventItem CID="recordingList_3" state="PENDING" />
   <RecordingEventItem CID="recordingList_1" state="PENDING" />
   <RecordingEventItem CID="recordingList_2" state="PENDING" />
   <RecordingEventItem CID="recordingList_4" state="PENDING" />
   <RecordingEventItem CID="recordingList_6" state="PENDING" />
   <RecordingEventItem CID="recordingList_8" state="PENDING" />
   <RecordingEventItem CID="recordingList_5" state="PENDING" />
   <RecordingEventItem CID="recordingList_7" state="PENDING" />
   <RecordingEventItem CID="recordingList_9" state="PENDING" />
   <RecordingEventItem CID="recordingList_10" state="PENDING" />
</RecordingEventList>
```

2.9.5.2.7 Delete Recording

A recording can be deleted by a DELETE on the following URL – **<control_url>/recorder/content/{CID}**

The following HTTP response codes could be returned:

200 OK: The capture content is deleted successfully

400 Bad Request

404 Not found The recorded asset is not found.

2.9.5.2.8 Get Recorder Status

The Recorder Status can be retrieved by a GET to **<control_url>/**recorder/info/RecorderInfo

The following HTTP response codes could be returned:

200 OK: The Recorder Info is found and returned successfully 404 Not Found:The Recorder Info is not found.

Below is an example of the Recorder status returned in the HTTP body.

```
<?xml version="1.0" encoding="UTF-8"?>
<RecorderInfo
    controlApiVersion="1.0"
    controlUrl="http://am-0-ums-0-1.VMP.com:8251/amr1/assetworkflows/amCdvr0"
    deliveryUrl="http://am-0-ums-0-1.VMP.com:8251/amCdvr0/{CONTENT_ID}/MPD"
    regionId="region-0"
    totalRecordingBandwidth="6291456"
    usedRecordingBandwidth="0"
    recordingStatus="OPERATIONAL"
    totalDeliveryBandwidth="6291456"
    usedDeliveryBandwidth="0"
    numActiveDeliveryStreams="0"
    deliveryStatus="OPERATIONAL"
    maxStorageCapacity="1048576"
    usedStorageCapacity="0"
    storageStatus="OPERATIONAL"
    numActiveRecordings="0" />
```

2.9.5.2.9 AMR1 Discovery API

This API is used to discover a content by the scheduleId. HTTP REST endpoint is **/recordingInfo/{scheduleId}?mode=redirect**

Following methods are supported

GET

Request parameters:

Parameters	Status	Type	Description
scheduleId	Required	xs:string	Unique label for the schedule.
mode	Required	xs:string	**redirect** – The response is an HTTP redirect to the recorder deliveryUrl template with the content. Will

Request Example:

GET <base-url>/recordingInfo/12345?contentType=Mpeg2&mode=redirect HTTP/1.1

Redirect Response Example:

HTTP/1.1 302 Moved

Location: http:// 192.169.203.220/cdvr-c/ab13bab4c12d/MPD

HTTP Status Codes:

The http status codes are as follows:

- 200 OK
- 302 Moved
- 400 Bad Request
- 404 Recording not found (may be also returned for an invalid URL)
- 500 Internal Server Error

2.9.6 Bulk Retrieval

The asset management base URL is used to bulk retrieve the status of a collection of assets. One or more filters are specified to scope the asset collection.

The following filters are supported.

- Filter by state of the asset.
- Filter by start time or end time of asset capture

The response of bulk retrieval is paginated. Pagination matadata is specified as an HTTP extension header

x-assetmgmt-metadata, as shown below:

x-assetmgmt-metadata: { " tot_num_assets" :10, " num_assets_per_page" :1,
" tot_num_pages" :10, " page_num" :2,
" prev_page" :" /v1/assetWorkflows/amldLive1/assets?limit=1&page=1" ,
" next_page" :" /v1/assetWorkflows/amldLive1/assets?limit=1&page=3" }

The following query strings are supported on asset management <interface_base_uri> to filter and paginate the results.

state	Return assets of the specified state only. State could be PENDING, CAPTURING, COMPLETE, FAILED or DELETE_FAILED.
timeRangeStart	Return assets that started capture after timeRangeStart. This is UTC time specified as ISO string.
timeRangeEnd	Return assets that started capture before timeRangeEnd. This is UTC time specified as ISO string.
contentId	Returns all assets of whose contentIds, the specified query parameter is a substring. The substring match is case-insensitive
limit	Pagination parameter to limit the number of assets in each response. Range is [1..500]. Default is 500.
page	Pagination parameter to specify the page number to retrieve. Range [1..tot_num_pages]. Default is 1 (first page).

2.9.6.1.1 Retrieve all assets (no filters)

If no filters are specified, all assets are retrieved, limited by the pagination parameter of number of assets per page of 500.

GET /v1/assetworkflows/live_wf/assets HTTP/1.1

HTTP/1.1 200 OK
Date: Fri, 7 Apr 2014 17:17:11 GMT

Content-Type: application/json x-
assetmgmt-metadata:
{" tot_num_assets" :2," num_assets_per_page" :500," tot_num_pages" :1," page_num" :1,
" prev_page" :null," next_page" :null}
[
{
" contentId" :" abc" ,
" userData":" {\" channelName\" :\" abc\" ,\" channelId\" :\" live_channel1\" ,\"
description\" :\"test channel\" }", " assetMgmtUrl" :
" http://am-cep1-
live.VMP.com:7001/v1/assetworkflows/live0/assets/abc" ,

```
        " status" :{
            " state" :" CAPTURING" ,
            " startTime" :" 2014-01-22T12:00:00.000Z" ,
            " captureStatus" :[
                {
                    " captureEngine" :" 10.1.1.1" ,
                    " state" :" CAPTURING" ,
                    " streams" :[
                        {
                            " name" :" 6mbps" ,
                            " sourceUrl" :" udp://228.1.2.21:878/" ,
                            " state" :" CAPTURING"
                        },
                        {
                            " name" :" 3mbps" ,
                            " sourceUrl" :" udp://228.1.2.21:879/" ,
                            " state" :" CAPTURING"
                        }
                    ]
                },
                {
                    " captureEngine" :" 10.1.1.2" ,
                    " state" :" FAILED" ,
                    " reason" :" ingest failure" ,
                    " streams" :[
                        {
                            " name" :" 6mbps" ,
                            " sourceUrl" :" udp://228.1.2.21:878/" ,
                            " state" :" CAPTURING"
                        },
                        {
                            " name" :" 3mbps" ,
                            " sourceUrl" :" udp://228.1.2.21:879/" ,
                            " state" :" FAILED" ,
                            " reason" :" ingest failure"
                        }
                    ]
                }
            ]
        }
    },

    {
        " contentId" :" espn" ,
        " userData”:" {\" channelName\" :\" espn\" ,\" channelId\" :\" live_channel1\" ,\
”description\" :\”test channel\" }”,
```

```
        " assetMgmtUrl" : " http://am-cep1-
live.VMP.com:7001/v1/assetworkflows/live0/assets/espn" , " status" :{
            " state" :" CAPTURING" ,
            " startTime" :" 2014-01-22T12:00:00.000Z" ,
            " captureStatus" :[
                {
                    " captureEngine" :" 10.1.1.1" ,
                    " state" :" CAPTURING" ,
                    " streams" :[
                        {
                            " name" :" 6mbps" ,
                            " sourceUrl" :" udp://228.1.2.21:878/" ,
                            " state" :" CAPTURING"
                        },
                        {
                            " name" :" 3mbps" ,
                            " sourceUrl" :" udp://228.1.2.21:879/" ,
                            " state" :" CAPTURING"
                        }
                    ]
                },
                {
                    " captureEngine" :" 10.1.1.2" ,
                    " state" :" FAILED" ,
                    " reason" :" ingest failure" ,
                    " streams" :[
                        {
                            " name" :" 6mbps" ,
                            " sourceUrl" :" udp://228.1.2.21:878/" ,
                            " state" :" CAPTURING"
                        },
                        {
                            " name" :" 3mbps" ,
                            " sourceUrl" :" udp://228.1.2.21:879/" ,
                            " state" :" CAPTURING"
                        }
                    ]
                }
            ]
        }
    }
]
```


2.9.6.1.2 Retrieve assets by pagination limit and page

Here is an example of requesting page 2 with limit 1.

```
GET /v1/assetWorkflows/live0/assets?limit=1&page=2 HTTP/1.1

HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
x-assetmgmt-metadata:
{" tot_num_assets" :10," num_assets_per_page" :1," tot_num_pages" :10," page_num" :2,
" prev_page" :" /v1/assetWorkflows/live/assets?limit=1&page=1" ," next_page" :" /v1/
assetWorkflows/live0/assets?limit=1&page=3" }
[
    {
        " contentId" :" espn" ,
        " userData”:" {\" channelName\" :\" awm_channel\" ,\" channelId\" :\" live_chann
el1\" ,\”description\" :\”test channel\" }”,
        " assetMgmtUrl" :" ,
        " status" :{
            " state" :" CAPTURING" ,
            " enableRelocation" :true,  " startTime" :" 2014-
            01-22T12:00:00.000Z" ,
            " captureStatus" :[
                {
                    " captureEngine" :" 10.1.1.1" ,
                    " state" :" CAPTURING" ,
                    " streams" :[
                        {
                            " name" :" 6mbps" ,
                            " sourceUrl" :" udp://228.1.2.21:878/" ,
                            " state" :" CAPTURING"
                        },
                        {
                            " name" :" 3mbps" ,
                            " sourceUrl" :" udp://228.1.2.21:879/" ,
                            " state" :" CAPTURING"
                        }
                    ]
                },
                {
                    " captureEngine" :" 10.1.1.2" ,
                    " state" :" FAILED" ,
                    " reason" :" ingest failure" ,
                    " streams" :[
                        {
                            " name" :" 6mbps" ,
                            " sourceUrl" :" udp://228.1.2.21:878/" ,
                            " state" :" CAPTURING"
                        },
                        {
```

```
                        " name" :" 3mbps" ,
                        " sourceUrl" :" udp://228.1.2.21:879/" ,
                        " state" :" CAPTURING"
                    }
                ]

            }
        ]
    }
}
]
```


2.9.6.1.3 Retrieve assets by state

GET https://am-cep1-ums-0-1.VMP.com:7001/v1/assetworkflows/cdvr_wf/assets?state=COMPLETE

HTTP/1.1 200 OK
Date: Thu, 3 July 2014 17:17:11 GMT
Content-Type: application/json x-assetmgmt-metadata:
{" tot_num_assets" :1," num_assets_per_page" :500," tot_num_pages" :1," page_num" :1, " prev_page" :null," next_page" :null}


```
[
  {
    " contentId" :" b6faab0e474d52d064d601a0f3c7acdc" ,
    " assetMgmtUrl" :" https://am-cep1-ums-0-1.cdvr.com:7001/v1/assetworkflows/cdvr_wf/assets/b6faab0e474d52d064d601a0f3c7acdc" ,
    " status" :{ " captureStatus" :[
        {
          " instanceId" :" instance0" ,
          " state" :" COMPLETE" ,
          " streams" :[
            {
              " fileSize" :" 39915681" ,
              " md5" :" 930c8b1e6f1522dc3cdb182e45d51182
              " , " name" :" Profile1" , " recordingDuration" :" 92" ,
              " state" :" COMPLETE"
            },
            {
              " fileSize" :" 29491438" ,
              " md5" :" 4d7e3d5ab4813a9f4ee65f6945e5584f" ,
              " name" :" Profile2" , " recordingDuration" :" 92" ,
              " state" :" COMPLETE"
            },
```

```
                {
                    " fileSize" :" 24104542" ,  " md5" :" 61974bfb132b4e6494c4ff812c467a23" ,
                    " name" :" Profile3" ,
                    " recordingDuration" :" 92" ,
                    " state" :" COMPLETE"
                }
            ],

            " captureEngine" :" 11.0.0.181"
        }
    ],
    " state" :" COMPLETE" ,
    " startTime" :" 2014-09-20T00:08:27.000Z" ,
    " endTime" :" 2014-09-20T00:10:01.492Z"
  }
 }
]
```


2.9.6.1.4 Retrieve assets created between a time range

The bulk retrieval API can be used to retrieve assets created between a time range. The time range is specified as UTC ISO string. The following queries are supported.

- **timeRangeStart**: Retrieve assets that started capture after a certain time.
- **timeRangeEnd**: Retrieve assets that started capture before a certain time.
- **timeRangeStart** and **timeRangeEnd**: Retrieve assets that started capture between a time range.

Here is an example of retrieving completed assets that are created between a time range.

GET https://am-cep1-ums-0-1.VMP.com:7001/v1/assetworkflows/cdvr-wf/assets?state=COMPLETE&timeRangeStart=2014-09-20T01:00:00Z&timeRangeEnd=2014-09-20T13:30:00Z

HTTP/1.1 200 OK

Date: Thu, 3 July 2014 17:17:11 GMT

Content-Type: application/json x-

assetmgmt-metadata:
{" tot_num_assets" :1," num_assets_per_page" :500," tot_num_pages" :1," page_num" :1," prev_page" :null," next_page" :null}


```
[
  {
    " contentId" :" b6faab0e474d52d064d601a0f3c7acdc" ,
    " assetMgmtUrl" :" https://am-cep1-ums-0-1.cdvr.com:7001/v1/assetworkflows/cdvr_wf/assets/b6faab0e474d52d064d601a0f3c7acdc" ,
```

```
" status" :{ " captureStatus" :[
        {
          " instanceId" :" instance0" ,
          " state" :" COMPLETE" ,
          " captureEngine" :" 11.0.0.181" ,
          " streams" :[
            {

              " fileSize" :" 39915681" ,
              " md5" :" 930c8b1e6f1522dc3cdb182e45d51182
              " , " name" :" Profile1" , " recordingDuration" :" 60" ,
              " state" :" COMPLETE"
            },
            {
              " fileSize" :" 29491438" ,
              " md5" :" 4d7e3d5ab4813a9f4ee65f6945e5584f" ,
              " name" :" Profile2" , " recordingDuration" :" 60" ,
              " state" :" COMPLETE"
            },
            {
              " fileSize" :" 24104542" , " md5" :" 61974bfb132b4e6494c4ff812c467a23" ,
              " name" :" Profile3" ,
              " recordingDuration" :" 60" ,
              " state" :" COMPLETE"
            }
          ]
        }
      ],
      " state" :" COMPLETE" ,
      " startTime" :" 2014-09-20T01:00:00.000Z" ,
      " endTime" :" 2014-09-20T02:00:00.000Z"
    }
  },
  {
    " contentId" :" a32bab0e474d52d064d601a0f3c7ac89" ,
    " assetMgmtUrl" :" https://am-cep1-ums0-
1.cdvr.com:7001/v1/assetworkflows/cdvr_wf/assets/a32bab0e474d52d064d601a0f3c7
```

```
ac89" ,
      " status" :{  " captureStatus" :[
            {
               " instanceId" :" instance0" ,
               " state" :" COMPLETE" ,
               " captureEngine" :" 11.0.0.182" ,
               " streams" :[
                  {
                     " fileSize" :" 39915681" ,
                     " md5" :" 930c8b1e6f1522dc3cdb182e45d51182" ,
                     " name" :" Profile1" ,  " recordingDuration" :" 60" ,
                     " state" :" COMPLETE"
                  },
                  {
                     " fileSize" :" 29491438" ,
                     " md5" :" 4d7e3d5ab4813a9f4ee65f6945e5584f" ,
                     " name" :" Profile2" ,  " recordingDuration" :" 60" ,
                     " state" :" COMPLETE"
                  },
                  {
                     " fileSize" :" 24104542" ,
                     " md5" :" 61974bfb132b4e6494c4ff812c467a23" ,
                     " name" :" Profile3" ,  " recordingDuration" :" 60" ,
                     " state" :" COMPLETE"
                  }
               ]
            }
         ],
         " state" :" COMPLETE" ,
         " startTime" :" 2014-09-20T13:00:00.000Z" ,
         " endTime" :" 2014-09-20T14:00:00.000Z"
      }
   }

]
```

2.9.7 Asset management data-model

2.9.7.1 Asset create data-model

```
{
    " properties" :{
        " restart" :{              // Used only to restart a live channel
            " type" :" boolean" ,
            " required" :false
        },
        " contentId" :{
            " type" :" string" ,
            " pattern" :" ^[A-Za-z0-9~._-]+$" ,  " required" :true
        },
        " userData" :{

            " type" :" string" ,
            " required" :false
        },
        " mediaSource" :{
            " type" :" object" ,
            " required" :true,
            " properties" :{
                " ebpMode" :{
                    " type" :" boolean" ,
                    " required" :true
                },
                " streams" :{
                    " type" :" array" ,
                    " required" :true,
                    " items" :{
                        " type" :" object" ,
                        " required" :true,
                        " properties" :{
                            " name" :{
                                " type" :" string" , " required" :true
                            },
                            " sourceUrl" :{
                                " type" :" string" ,
                                " required" :true
                            },
                            " sourceIp" :{
                                " type" :" string" ,
                                " required" :false
                            }
                        }
                    }
                }
            }
```

```
        },
        " statusCallback" :{
            " type" :" object" ,
            " required" :false,
            " properties" :{
                " url" :{
                    " type" :" string" , " required" :true
                }
            }
        },
        " captureStart" :{              //Used only for cdvr asset
            " type" :" string" ,
            " required" :false,
            " format" :" date-time"
        },
        " captureEnd" :{            //Used only for cdvr asset
            " type" :" string" ,
            " required" :false,
            " format" :" date-time"
        }
    }

}
```

2.9.7.2 Asset GET data-model

```
{
    " properties" :{ " contentId" :{
            " type" :" string" ,
            " pattern" :" ^[A-Za-z0-9~._-]+$" , " required" :true
        },
        " userData" :{ " type" :" string" ,
            " required" :false
        },
        " storageId" :{
            " type" :" string" ,
            " required" :false
        },
        " enableRelocation" :{
            " type" :" boolean" ,
            " required" :false
        },
        " customConfigs" :{
            " type" :" array" ,
            " required" :false,
            " items" :{
                " type" :" object" , " required" :false,
                " properties" :{
                    " name" :{
```

```
                    " name" :" string" ,
                    " required" :true
                },
                " value" :{
                    " name" :" string" ,
                    " required" :true
                }
            }
        }
},
" mediaSource" :{
    " type" :" object" ,
    " required" :false,
    " properties" :{
        " ebpMode" :{
            " type" :" boolean" ,
            " required" :true
        },
        " streams" :{
            " type" :" array" ,
            " required" :true,
            " items" :{
                " type" :" object" ,
                " required" :true,
                " properties" :{
                    " name" :{

                        " type" :" string" , " required" :true
                    },
                    " sourceUrl" :{
                        " type" :" string" ,
                        " required" :true
                    },
                    " sourceIp" :{
                        " type" :" string" ,
                        " required" :false
                    },
                    " bitrate" :{
                        " type" :" string" ,
                        " required" :false
                    },
                    " elementaryStreams" :{
                        " type" :" array" ,
                        " required" :false, " items" :{
                            " type" :" object" ,
                            " required" :" false" , " properties" :{
                                " pid" :{
                                    " type" :" string" ,
                                    " required" :true
```

```
                    },
                     " type" :{
                         " type" :" string" ,
                         " required" :true
                    },
                    " format" :{ " type" :" string" ,
                         " required" :false
                    },
                    " bitRate" :{ " type" :" string" ,
                         " required" :false
                    },
                    " frameRate" :{ " type" :" string" ,
                         " required" :false
                    },
                    " height" :{ " type" :" string" ,
                         " required" :false
                    },
                     " width" :{
                              " type" :" string" ,
                              " required" :false
                    },
                    " ccType" :{ " type" :" string" ,
                         " required" :false
                    },
                    " sampleRate" :{ " type" :" string" ,

                              " required" :false
                    },
                    " bitsPerSample" :{
                         " type" :" string" ,
                         " required" :false
                    },
                    " channelCount" :{
                         " type" :" string" ,
                         " required" :false
                    },
                    " language" :{ " type" :" string" ,
                         " required" :false
                    },
                    " languageRole" :{
                         " type" :" string" ,
                         " required" :false
                    }
               }
          }
     }
}
```

```
        }
    },
    " statusCallback" :{
        " type" :" object" ,
        " required" :false,
        " properties" :{
            " url" :{
                " type" :" string" , " required" :true
            }
        }
    },
    " assetMgmtUrl" :{
        " type" :" string" ,
        " required" :false
    },
    " endTime" :{ " type" :" string" ,
        " required" :false
    },
    " startTime" :{
        " type" :" string" ,
        " required" :false
    },
    " alternateContent" :{
        " type" :" object" ,
        " required" :true,
        " properties" :{
            " poisUrl" :{
                " type" :" string" ,
                " required" :false
            },

            " poisQueryWithBinaryData" :{
                " type" :" boolean" , " required" :false
            },
            " poisVersion" :{
                " type" :" string" ,
                " required" :false
            }
        }
    },
    " status" :{ " type" :" object" ,
        " required" :false,
        " properties" :{
            " state" :{
                " type" :" string" ,
                " enum" :[
                    " INIT" ,
                    " PENDING" ,
                    " CAPTURING" ,
```

```
            " COMPLETE" ,
            " CANCELED" ,
            " DELETING" ,
            " DELETE_COMPLETE" ,
            " DELETE_FAILED" ,
            " FAILED"
        ],
        " required" :true
},
" reason" :{
        " type" :" string" ,
        " required" :false
},
" startTime" :{
        " type" :" string" ,
        " required" :false
},
" captureStatus" :{
        " type" :" array" ,
        " required" :false,
        " items" :{
            " type" :" object" ,
            " required" :false,
            " properties" :{
                " instanceId" :{
                    " type" :" string" ,
                    " required" :true
                },
                " captureEngine" :{ " type" :" string" ,
                    " required" :true
                },
                " state" :{
                    " type" :" string" ,
                    " enum" :[
                        " CAPTURING" ,

                        " COMPLETE" ,
                        " CANCELED" ,
                        " FAILED"
                    ],
                    " required" :true
                },
                " reason" :{
                    " type" :" string" ,
                    " required" :false
                },
                " storageSizeBytes" :{
                    " type" :" integer" ,
                    " required" :false
```

```
                    },
                    " streams" :{
                        " type" :" array" ,
                        " required" :false,
                        " items" :{
                            " type" :" object" ,
                            " required" :false, " properties" :{
                                " name" :{
                                    " type" :" string" ,
                                    " required" :true
                                },
                                " sourceUrl" :{ " type" :" string" ,
                                    " required" :true
                                },
                                 " state" :{
                                    " type" :" string" , " enum" :[
                                        " CAPTURING" ,
                                        " COMPLETE" ,
                                        " DELETING" ,
                                        " FAILED"
                                     ],
                                      " required" :true
                                },
                                " fileSize" :{ " type" :" string" ,
                                    " required" :false
                                },
                                " recordingDuration" :{ " type" :" string" ,
                                    " required" :false
                                },
                                " downloadUrl" :{           //based on asset download flag
                                    " type" :" string" ,
                                      " required" :false
                                },
                                " bitrate" :{ " type" :" string" ,
                                    " required" :false
                                }
                            }

                        }
                    }
                }
            }
        }
    },
    " output" :{ " type" :" array" ,
        " required" :false,
        " items" :{
            " type" :" object" , " required" :false,
```

```
" properties" :{
    " name" :{
        " type" :" string" ,
        " required" :true
    },
    " packageFormat" :{
        " type" :" string" ,
        " required" :true, " enum" :[
            " hls" ,
            " hss" ,
            " hds"
        ]
    },
    " publishTemplateName" :{
        " type" :" string" ,
        " required" :false
    },
    " segmentDurationInSec" :{
        " type" :" integer" ,
        " required" :false,
        " minimum" :2,
        " maximum" :10
    },
    " variants" :{ " type" :" array" ,
        " required" :false, " items" :{
            " type" :" object" ,
            " required" :true,
            " properties" :{
                " name" :{
                    " type" :" string" , " required" :true
                },
                " version" :{ " type" :" string" ,
                    " require" :false,
                    " enum" :[
                        " 2" ,
                        " 3" ,
                        " 4" ,
                        " 5"

                    ]
                },
                " publishUrl" :{
                    " type" :" string" ,
                    " required" :true
                },
                " videoStreams" :{
                    " type" :" array" ,
                    " required" :true,
                    " items" :{
```

```
                    " type" :" string" , " required" :true
                }
            },
            " audioStreams" :{
                " type" :" array" ,
                " required" :true,
                " items" :{
                    " type" :" string" , " required" :true
                }
            }
        }
    },
    " drmSystem" :{ " type" :" object" ,
        " required" :true,
        " properties" :{
            " type" :{
                " type" :" string" , " required" :true
            },
            " name" :{
                " type" :" string" , " required" :true
            }
        }
    },
    " httpHeaderPolicy" :{
        " type" :" object" ,
        " required" :true,
        " properties" :{
            " manifest" :{
                " type" :" array" ,
                " required" :true,
                " items" :{
                    " type" :" object" ,
                    " required" :true,
                    " properties" :{
                        " header" :{ " type" :" string" ,
                            " required" :true
                        },
                        " value" :{
                            " type" :" string" ,

                            " required" :true
                        }
                    }
                }
            },
            " chunk" :{
                " type" :" array" ,
                " required" :true,
```

```
                                    " items" :{
                                        " type" :" object" ,
                                        " required" :true,
                                        " properties" :{
                                            " header" :{ " type" :" string" ,
                                                " required" :true
                                             },
                                              " value" :{
                                                " type" :" string" ,
                                                " required" :true
                                            }
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}
```

3 Building and Operating VMP Services

Various functions of the VMP system are implemented in context of “services”. The Services offer a logical container to bind different policies, workflows and templates to ingest, package, store and publish Multi-Screen Video Content.

3.1 Pre-Requisites

System is fully installed and the Service Management API is available to configure the system.

3.2 Services Provisioning

The VMP system comes pre-loaded with a set of pre-defined generic Unified Media Service templates. Each service template can be configured for a combination of Live, VOD and CDVR media services. The different types of media services are further defined in Asset Workflow templates. The administrators can access the service templates using the APIs specified in section 3.6.1. The service templates are referenced during the creation of service instances. Each Service instance conforms to utVMPt one service template from the pre-defined list.
For each Service domain (or tenant) in the system, provisioning process involves the creation of policies, templates and resources. These are objects that are re-usable within a service domain.

3.2.1 Creating Platform Domain objects

Before any service is created, verify the following objects and update them as necessary: Zone, DNSServer, DNSForwarder, Node, NTPServer and DNSZoneMap.

3.2.2 Creating Policies, Templates and Resources

Step	Operation	Type	Scope	Description
1	Create Media Sources	Resource	Service Domain	Media Sources can be channel sources, NAS media source or HTTP based sources. Sources define the location from where input media streams can be retrieved. For Live Services: StreamProfiles and Channel Sources are used to define the multicast feeds.

2	Create Storage Resource	Resource	Service Domain	Storage to be used by the VMP system is defined using the “Store Resources”. The system supports one or more store types including (NAS storage and SWIFT based object storage). The Assets that are packaged by the VMP system are stored in the storage backend described by the Store configuration. One or more storage resources can be specified in the system and can later be bound to a service instance as part of the
3	Create Key Profile	Resource	Service Domain	Key Profile describes the DRM system to be used during the Asset Publish Process. One or more DRM system configuration can be supported at any time. The Publish templates refer to the specific key profile to be used for a given packaging format.
4	Create output HTTP Header policies	Policy	Service Domain	This defines the types of HTTP headers to be applied for various resources generated as part of the asset publish process. These headers can be interpreted by the CDNs to implement caching workflows.
4	Create Publish Templates	Template	Service Domain	Publish Template describes the output that is created by the VMP system that is consumable by CDN or end client. The Publish templates define the output media package format along with the publishing rules to define the variants of the published asset. The templates also include the references to the DRM Key Profiles to be used for output assets.
5	Create Asset Life-Cycle Policy	Policy	Service Domain	This defines the life cycle management policies to be used for the ingested media assets.
6	Create Asset Redundancy	Policy	Service Domain	Defines if redundant copies of media assets are required to be stored in the system.

Table 4 Service Domain Object Creation

3.2.3 Creating a Service Instance

Service instances are created based on the service template definitions. The service instance container can reference the objects created in the previous step.
Each Service Instance can perform on one or more ***asset workflows.***
Each Service Instance can be associated with one or more **Capture Endpoints and Playback Endpoints.** These endpoints contain SLA guidelines which are used to create Virtual Machines to implement the SLA. The process of launching the Virtual machines for the given SLA is automatic and does not require admin intervention. By defining capture and playback endpoints, the service operator is assigning the compute resources used to implement the capture and publish operations in the system. A Capture Endpoint requires a StateCache Endpoint to store data caching information. A single StateCache Endpoint is to be shared by multiple Capture Endpoints in the same service instance.

3.2.3.1 Creating Asset Workflows

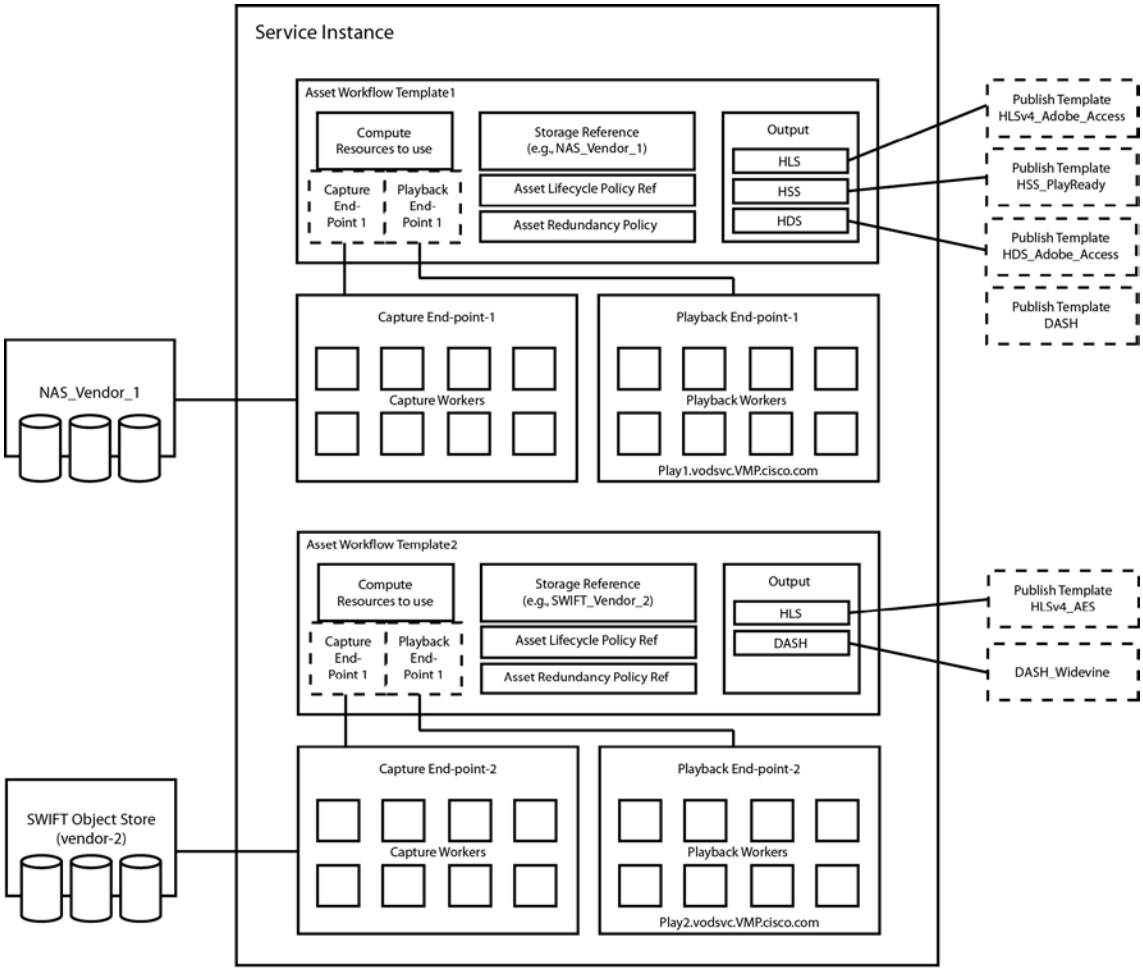
Asset workflow template is used to define the asset packaging and publishing operations. Each Asset workflow contains the following
Pre-defined Input Media Sources (optional) available to the workflow. This is specifically useful when the sources are known during the service provisioning time (for example, live channels). For VOD use case, the sources are discovered as and when the contents are pitched and transcoded – so the pre-defined model is not preferable.
Pre-defined Output Media Asset Types to be generated – this defines the output that is generated by the asset workflow. One or more asset types can be published. Each of the output media asset type contains a reference to the publish template that describes the output packaging format. For example, one could have HLS, HSS, HDS, and DASH assets published by the playback end point and each of these conform to a specific publish template as defined in the service domain.
The Capture and Playback Endpoints that are used to implement the workflows. Note that the endpoints are defined at the Service Instance Scope.
The Reference to the Storage system (such as the NAS store) to be used for the Workflow template
The Reference to the Asset Life cycle policy to be applied for all the stored assets produced by this template. The Asset life cycle policy is where the Live Time window policies and other asset expiration policies can be set.

Once workflows are created and associated with the Service Instance, the Service Control APIs can be used to trigger the Asset Creation and publishing activities.
If the assets are pre-declared in the system (as in the Channel Lineup scenario), the system automatically creates the media assets that map to the live channels.

3.2.3.2 Executing Asset Creation Workflows

In cases where the Media Sources and the output assets are not pre-declared, the Media Asset API can be used to invoke the Asset workflow on a specific Media Source. The Asset workflow is executed by the VMP system and produces the Assets as defined in the Asset workflow publish templates.

Figure 8



Example Service Instance