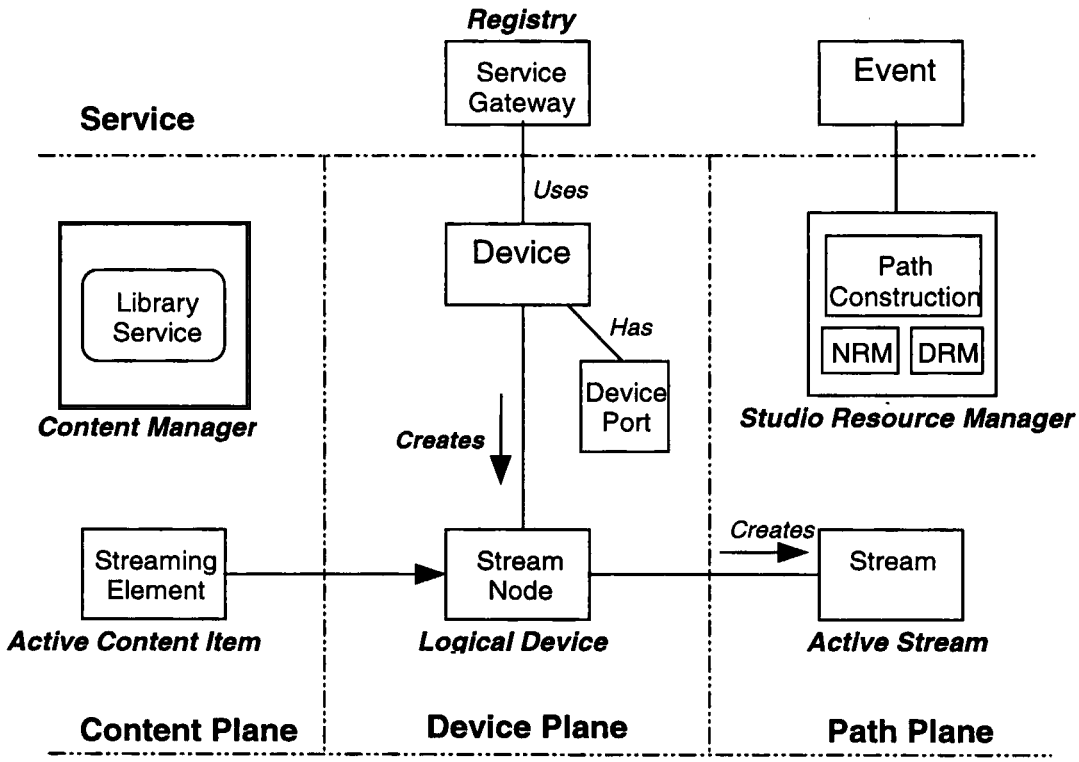
5 Studio Object Model

5.1 Object Diagram



5.2 Reference to Functional Architecture

The *studio object model* is illustrated in section 5.1. It has been partitioned to illustrate the division of objects between the 4 principal planes: Content, Service, Path and Device. Each object is described in detail below.

5.2.1 Service Plane

5.2.1.1 Event

Events represent sequences of studio activities, each of which may rely on multiple studio resources. An event can be triggered by any number of conditions, for example:

- when a specific button is pushed
- when specific resources become available
- when a specific error occurs
- when another event is triggered
- at a specific time of day

Event sequencing can be arbitrarily complex allowing the construction of advanced workflows. Note that the event mechanism is available in the service plane is a toolkit for the creation of the required event logic within the studio.

5.2.1.2 *Service Gateway*

The Service Gateway (SG) object is principally responsible for providing a device registration and identification function. When a device is installed in the studio, it must register with the SG and identify itself, providing information such as its capabilities, number and type of ports, status (on-line, off-line, etc), timing information (e.g. latencies) and other pertinent data. The SG will maintain a repository of all the live devices/objects in the studio. This repository can be accessed by users to dynamically discover these devices/objects and by the Studio Resource Management as the basis for resource tracking, allocation, and reservation. The Service Gateway shall also be responsible for implementing security measures and access control.

5.2.2 Path Plane

The responsibility of the Path Plane is to establish a path through the studio's devices in order to accomplish a particular Event or task in a timely fashion and with a guaranteed quality of service. If desired, the path plane will select the devices needed to perform the task and will define and establish the connections from one device to another, or through chained devices, over a network of routers. It includes the Path Construction object which identifies devices needed to perform a particular task, the Network Resource Management (NRM) system that manages the routers and the Device Resource Manager (DRM) that manages devices. Each of these entities are optional and a simple studio configuration can be operated directly from a client/application if desired.

5.2.2.1 *Path Construction*

The responsibility of the Path Construction Service (PCS) is to map an Event or task (received from a client or application) into a detailed resource request for devices and/or network resources. For instance, an Event may require a transmission of a video signal from cameraA to the transmitterB. If the signal from cameraA must be transcoded before sending it to the transmitter, the Path Construction Service would be responsible for identifying what type of transcoder is needed, e.g. transcoderC. In actuality, there may be several options available for achieving a particular task, e.g. use a transcoderC or use a transcoderB followed by a transcoderD. Once the options are determined, the Path Construction Service will interrogate the Service Gateway to determine if the components exist to utilize one or more of these options.

If the components do not exist for any of the options, the request is denied; otherwise, the PCS sends a detailed request, for devices A to C to B, to the Device Resource Manager to determine if the components are available at the appropriate time and to reserve the resources if necessary. In the Integrated Management (IM) Mode, the PCS will compute the route and will instruct the *Network Resource Manager* to execute the connection. In this case, the PCS is the only entity which directly accesses the Network Resource Manager. This enables the coordinated allocation of both studio devices and network resources, which can ultimately lead to higher utilization of studio resources and a higher success rate for satisfying user requests for studio resources. In the Non-Integrated Management (NIM) Mode, the PCS will send a detailed resource request to the NRM, which will compute the route. A positive response from both managers will allow the PCS to acknowledge the Event or task request.

5.2.2.2 *DRM*

The Device Resource Manager is responsible for tracking studio devices and for allocating and reserving these resources. The Reservation System is necessary to insure the availability of necessary resources to the mission critical tasks.

5.2.2.3 *NRM*

The Network Resource Manager is responsible for determining the topology of the network, monitoring the status of network links and switches/routers and for establishing connections across the network in response to requests from the Path Construction object, the System

Resource Manager and/or studio clients/applications. In the Integrated Mode, the NRM object may be exclusively accessed by the PCS (in cases where efficient utilization of studio resources is desired).

5.2.2.4 Controllable Stream

Streams represent the control points for the actual flow of data. While a StreamNode is capable of setting up and creating streams between itself and another StreamNode, it is handy to collect all of the actual control functionality into a different class designed for that purpose. This is the role that the Stream class plays. All control functions such as Start, Stop, Pause, etc. are contained within this class and clients can use these methods to frame accurately control the flow of data through single or multi-segment stream paths. Simply put, a StreamNode creates a Stream object to control the data flow and returns it to the client. Until the stream is finished and the resources need to be deallocated, this is the only important view that the client requires. It makes the task of performing complex stream control operations relatively easy from a client's perspective.

5.2.3 Device Plane

5.2.3.1 Device

The Device class (i.e. the software object that controlls access to the physical device) contains all of the device's hardware level attributes and control functions. This includes attributes like power on/off, device status, number of physical ports, maximum number of streams, etc. Those devices that have hardware functions beyond this limited set can subclass device and add the appropriate attributes and methods. In the case of a tape deck, this will include the user controlled front panel functions like Play, Stop, Fast Forward, etc. The Device class and its subclasses mirror the physical traits of the device and look most like the actual object in a traditional, non-streaming sense. For the most part, this class is not used directly in the Streaming model and is reserved for use by applications that do not require network controlled streaming, or are not utilizing the resource management features of the system.

5.2.3.2 Device Port

The *Device Port* class represents a physical port for a device. Each device will have one or more physical ports and will produce an equal number of Device Port objects to model them. The Device Port objects are used primarily during resource allocation to allow the SRM to bind and pre-allocate the port availability for a future streaming session. As with the Device class, these objects are not generally used by clients during the Controlled Streaming process and as such are somewhat less interesting than the other device classes.

5.2.3.3 Stream Node

StreamNodes represent the heart of the streaming process setup and stream creation methods. Simply put, a StreamNode represents the data that is to be streamed. In this object model, the data is generally considered to be intelligent enough to know how to stream itself and, as such, the reservation, setup and stream creation method reside in this class. This can be more easily understood if we take the example of a file on a video server. The file is represented by StreamNode subclass (called StreamingElement) that has all the information necessary to provide the client with information such as its pre-roll latency, length, format, transmission time, etc., as well as methods to perform the operations of pre-rolling the video, creating a stream and tearing down the connection when streaming is finished. The idea behind the name StreamNode is that this data represents the source point for data streaming into and out of a device.

5.2.4 Content Plane

5.2.4.1 Content Manager

The Content Manager (CM) is responsible for registering, tracking and allocating content assets within the studio. It communicates directly with clients/applications in the studio in addition to coordinating tasks with the other planes. In addition to the tasks listed above, the content manager should facilitate the searching and browsing of the studio's content and should provide security and access control to these items.

5.2.4.2 Streaming Element

A content item that has been loaded onto one of the media servers (or tape servers) and may be recovered at a later point. An object corresponding to the streaming element is created using the class StreamingElement by the appropriate media server when the media server receives a request from a client/application. A reference to the streaming element object is returned to the requester for subsequent use via the media server interface. Software objects never have access to the underlying structure of the streaming element, which can be represented on the media server in an arbitrary way. It is a characteristic of streaming element objects that they may be streamed, that is the StreamingElement class is derived from the class StreamNode.

5.2.4.3 Stream Operation

A path through *N* components will generate *N-1* streams. This section describes the semantics associated with the control of these streams.

Each Stream within a path must be controlled by the client independently of other streams. The client is responsible for ensuring that delay computations associated with the delivery of content from one component to another are taken into account. Delay information is obtained from the Service Gateway and may be used in these computations. A component that offers services such as a filter (i.e. contains at least one source StreamNode and one sink StreamNode) is expected to handle the full Stream command syntax. The operation of a filter component that receives stream content prior to the requested start time is undefined, e.g., the component may pass the content through without loss of data or the component may drop data prior to the start time.

A device that advertises a *TransferDelay* must be capable of buffering at least a transfer delay's worth of data. When a filter receives a start time it is required that the data delivered include up to a TransferDelay's worth of data already buffered within the component. For example, a component that advertises a Transfer Delay of 20msec must be able to buffer at least 20msec of content internally. This requirement ensures that components that are within a Stream path will permit continuous streaming of content between temporally concatenated streams.