SCTE · ISBE
STANDARDS

Digital Video Subcommittee

AMERICAN NATIONAL STANDARD

ANSI/SCTE 224 2018r1

Event Scheduling and Notification Interface (ESNI)

NOTICE

The Society of Cable Telecommunications Engineers (SCTE) / International Society of Broadband Experts (ISBE) Standards and Operational Practices (hereafter called "documents") are intended to serve the public interest by providing specifications, test methods and procedures that promote uniformity of product, interchangeability, best practices and ultimately the long-term reliability of broadband communications facilities. These documents shall not in any way preclude any member or non-member of SCTE•ISBE from manufacturing or selling products not conforming to such documents, nor shall the existence of such standards preclude their voluntary use by those other than SCTE•ISBE members.

SCTE•ISBE assumes no obligations or liability whatsoever to any party who may adopt the documents. Such adopting party assumes all risks associated with adoption of these documents, and accepts full responsibility for any damage and/or claims arising from the adoption of such documents.

Attention is called to the possibility that implementation of this document may require the use of subject matter covered by patent rights. By publication of this document, no position is taken with respect to the existence or validity of any patent rights in connection therewith. SCTE•ISBE shall not be responsible for identifying patents for which a license may be required or for conducting inquiries into the legal validity or scope of those patents that are brought to its attention.

Patent holders who believe that they hold patents which are essential to the implementation of this document have been requested to provide information about those patents and any related licensing terms and conditions. Any such declarations made before or after publication of this document are available on the SCTE•ISBE web site at http://www.scte.org.

All Rights Reserved
© Society of Cable Telecommunications Engineers, Inc. 2018
140 Philips Road
Exton, PA 19341

Table of Contents

List of Figures

List of Tables

Introduction

1.1. Executive Summary

Video distribution is transitioning from Quadrature Amplitude Modulation (QAM) to delivery over Internet Protocol (IP) networks. These IP networks are delivering video content to subscribers via many distribution paths to many IP-connected devices, including mobile phones, tablets and game consoles for video content of all forms, including Video on Demand (VOD) and linear, live content.
Many QAM systems were developed to enable programmers to inform and affect the content delivery to subscribers. For example, during a regional sports blackout, a video provider may use an Integrated Receiver and Decoder (IRD) to provide alternate content to a unique geographic area serving a set of subscribers.
As distributors migrate to IP-delivered content, systems must be created to replicate the traditional functional systems in order to create a contiguous service capability between QAM and IP video delivery. Additionally, providers are also delivering single mezzanine quality feeds to the distributor. This requires the distributor to also replicate the functionality on the traditional delivery system.

1.2. Scope

This document defines the Event Scheduling and Notification Interface (ESNI), which is a web interface facilitating the transmission of event and policy information. ESNI provides a functional method for providers to communicate upcoming schedule or signal-based events and corresponding policy to distributors. This interface allows existing content distribution controls traditionally performed via manual control in IRD’s by providers to be replaced with a programmatic interface (this standard). ESNI policy enables control of content distributed to audiences based on attributes of that audience including (but not limited to) geographic location and device type.

1.3. Benefits

ESNI can be used to communicate details regarding regional blackout/alternate content selection, market protection, or other content restrictions as they may relate to a defined audience. This method can also inform the distributor of other events such as advertising breaks and availability for digital ad insertion, network PVR record times and restrictions, or program information (i.e. improve accuracy of electronic program guide). Additionally, ESNI supports an audit method that allows the provider to query the status of policy execution and verify the execution result.

1.4. Intended Audience

Content Providers, Multi-Channel Video Program Distributors, TV Everywhere Providers/Distributors.

1.5. Areas for Further Investigation or to be Added in Future Versions

There are two areas for further investigation and potential future versions. One area would be providing the ability to verify the audiences accessible on the distributor execution platform. Another would be further definition, enhancement, and standardization of the metadata fields in this standard.

2. Normative References

The following documents contain provisions, which, through reference in this text, constitute provisions of this document. At the time of Subcommittee approval, the editions indicated were valid. All documents are subject to revision; and while parties to any agreement based on this document are encouraged to investigate the possibility of applying the most recent editions of the documents listed below, they are reminded that newer editions of those documents might not be compatible with the referenced version.

2.1. SCTE References

- ANSI/SCTE 35 2017, Digital Program Insertion Cueing Message for Cable.
- ANSI/SCTE 236 2017, Content Metadata.

2.2. Standards from Other Organizations

- ISO 8601:2004, Data elements and interchange formats -- Information interchange -- Representation of dates and times (Coordinated Universal Time).
- IETF RFC 2014, Internet Research Task Force Research Group Guidelines and Procedures, A Weinrib, J. Postel October 1996.
- IETF RFC 2119, Key words for use in RFCs to Indicate Requirement Levels. S. Bradner. March 1997.
- IETF RFC 3986, Uniform Resource Identifier (URI): Generic Syntax. T. Berners-Lee, R. Fielding, L. Masinter. January 2005.
- IETF RFC 4684, Constrained Route Distribution for Border Gateway Protocol/MultiProtocol Label Switching (BGP/MPLS) Internet Protocol (IP) Virtual Private Networks (VPNs). P. Marques, R. Bonica, L. Fang, L. Martini, R. Raszuk, K. Patel, J. Guichard. November 2006.
- IETF RFC 6749, The OAuth 2.0 Authorization Framework. D. Hardt. October 2012.
- IETF RFC 7230, Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing. R. Fielding, J. Reschke. June 2014.
- IETF RFC 7231, Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content. R. Fielding, J. Reschke. June 2014.
- W3C XML Base (Second Edition). W3C Recommendation 28 January 2009. http://www.w3.org/TR/xmlbase/.
- W3C XML Schema Part 2: Datatypes Second Edition. W3C Recommendation 28 October 2004. http://www.w3.org/TR/xmlschema-2/.
- W3C XML Path Language (XPath) 2.0 (Second Edition). W3C Recommendation 14 December 2010. http://www.w3.org/TR/xpath20/.
- W3C XML Linking Language (XLink) Version 1.1. W3C Recommendation 06 May 2010. http://www.w3.org/TR/xlink11/.

2.3. Published Materials

- No normative references are applicable.

3. Informative References

The following documents might provide valuable information to the reader but are not required when complying with this document.

3.1. SCTE References

- No informative references are applicable.

3.2. Standards from Other Organizations

- Real-time Event Signaling and Management API, OC-SP-ESAM-API-I03-131025, October 25, 2013, Cable Television Laboratories, Inc.

3.3. Published Materials

- No informative references are applicable.

4. Compliance Notation

shall	This word or the adjective "***required***" means that the item is an absolute requirement of this document.
shall not	This phrase means that the item is an absolute prohibition of this document.
forbidden	This word means the value specified shall never be used.
should	This word or the adjective "*recommended*" means that there may exist valid reasons in particular circumstances to ignore this item, but the full implications should be understood and the case carefully weighted before choosing a different course.
should not	This phrase means that there may exist valid reasons in particular circumstances when the listed behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.
may	This word or the adjective "*optional*" means that this item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because it enhances the product, for example; another vendor may omit the same item.
deprecated	Use is permissible for legacy purposes only. Deprecated features may be removed from future versions of this document. Implementations should avoid use of deprecated features.

5. Abbreviations and Definitions

5.1. Abbreviations

Ad-ID	Advertisement Identifier Registry
EIDR	Entertainment Identifier Registry
ESAM	Event Signaling and Management
ESNI	Event Scheduling and Notification Interface
HMAC-SHA256	Hashed Message Authentication Code with Secure Hash Algorithm, 256 bits
HREF	Hypertext REFerence

HTTP	Hypertext Transfer (or Transport) Protocol
IRD	integrated receiver/decoder
MVPD	multichannel video programming distributor
NPT	normal play time
QAM	quadrature amplitude modulation
TZD	time zone designator
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
URN	Uniform Resource Number
UTC	Coordinated Universal Time (or Universal Time Coordinated)
VIRD	virtual IRD
VOD	video on demand
XML	eXtensible Markup Language

5.2. Definitions

ESNI Event	Point of interest relative to the timeline of a media presentation. In this standard, it is represented by an in-band signal, wall clock time, or offset from the start of the media.
Event Scheduling and Notification Service	Receives ESNI event and policy information and provides feedback on the application of the events and policies.
Scheduler	Also referred to as the Alternate Content Scheduling System, it maintains the metadata associated with alternate content events, such as program information, duration, and affected geo-location.
Virtual IRD	A logical grouping of subscribers loosely associated with their geographic location and conceptually similar to an IRD coverage area.

5.3. Style Conventions

The following conventions are used within this document.

- XML Elements names are capitalized and are represented in the document in bold, fixed-width font as in **`Element`**. When expressing an element hierarchy, a forward slash is used as in **`Element1`**/**`Element2`**. Element names comprised of multiple words are expressed using camel-case as in **`ThisElement`**. Element cardinality is expressed in this document as <minOccurs>…<maxOccurs>, where the values are the minimum and maximum occurrences of the element respectively.
- XML Attribute names are preceded by an '@' character, begin with a lower-case letter and represented in the document as a fixed-width font as in `@attribute`. When expressing an attribute as a child of a specific element, a forward slash is used as in **`Element`**/`@attribute`. Attribute names comprised of multiple words are expressed using camel-case as in `@thisAttribute`. Attributes *may* be characterized as mandatory (M), optional (O), optional with default value (OD) or conditionally mandatory (CM).
- XML Type names are enclosed in "<" and ">" characters and are represented in the document in bold, fixed-width font as in **`<Type>`**.

6. Overview (Informative)

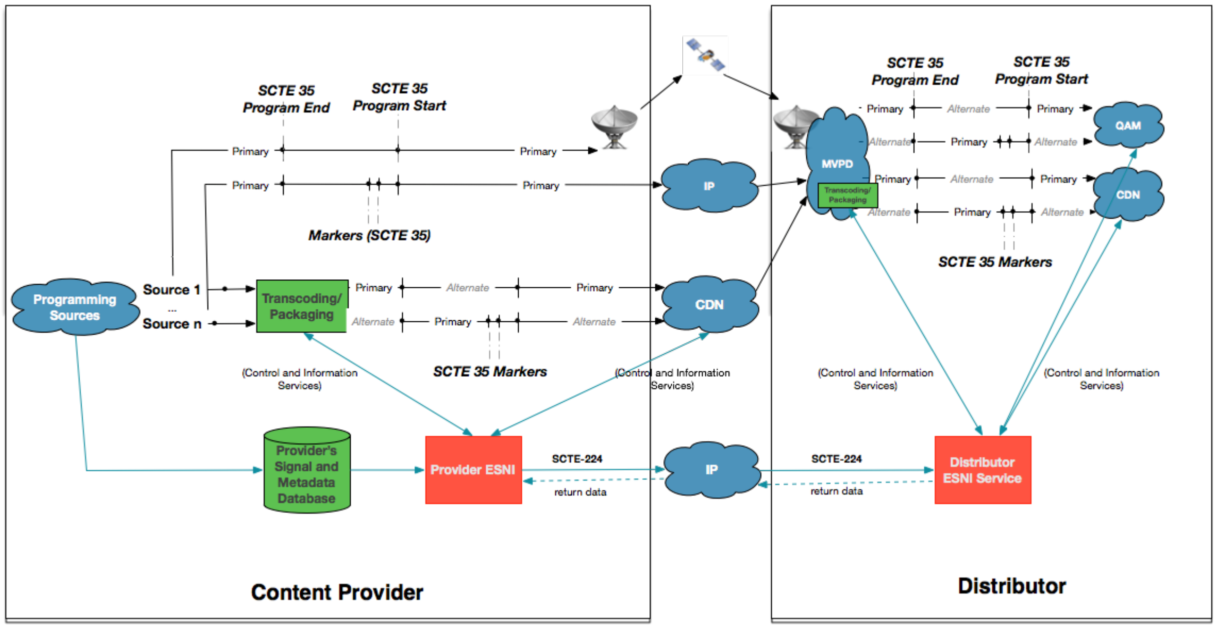


Figure 1 - Conceptual Cable Alternate Content Ecosystem (Non-Normative)

Table 1 - Alternate Content Element Descriptions

Alternate Content Elements	Description
Provider ESNI	Provides ESNI events and policy information to distributor systems. *May* optionally provide program information. *May* also provide direct to consumer event directives.
Distributor ESNI Service	Receives ESNI event and policy information from a provider and makes feedback available on the application of the events and policies back to the provider. Interacts with delivery systems to detect ESNI events and apply policies.
Programming Sources	Video/audio sources that *may* contain SCTE 35 signals.
Transcoding/Packaging	Sends and receives signal data and *may* receive directives to affect the video/audio stream, if necessary, for downstream processing and/or event decision-making.
Provider Signal and Metadata Database	Houses signal and/or programming metadata that *may* be used by the Provider ESNI service for the purposes of policy distribution and/or event directive enforcement.

7. XML Practices

The ESNI data model is specified in the XML Schema, and the reader is assumed to be reasonably informed on practices associated with XML processing. Note, for the SCTE 224 schema, versions of the schema are identified by the “version” attribute of the schema in the form of “yyyymmdd”. With such general practice and understanding, throughout the document there *may* be expressly identified XML areas of practice to which attention is called to eliminate possible ambiguity or lack of clarity.

7.1. Document Order

One such topic as it relates to this standard is the preservation of strict document order throughout the processing of a XML documents. Strict document order ***shall*** be both preserved during transfer as well as during processing.

7.2. Representation of time

Unless otherwise specified, all duration and date and time values in this data model ***shall*** use the formats specified in [XMLSCHEMA2], which are closely related to [ISO8601].
The XML Schema dateTime data type allows for values to include a Time Zone Designator (TZD). If a TZD is not present, the time is assumed to be local relative to the end-user, as represented by the user-agent or device. While it *may* be safe to assume local time when communicating within the same time zone, it is ambiguous when used in communicating across different time zones. Time values in this data model *should* include a time zone designator.
When a TZD is present, the time is specified as a local time followed by a numeric offset from UTC. If the time zone is UTC (i.e. an offset of zero), the special designator "Z" *may* be used.
The following values represent the same instant in time

- 18:30:00Z
- 18:30:00+00:00
- 22:30:00+04:00
- 11:30:00−07:00

The XML Schema duration data type is the appropriate format for carrying a time period.

7.3. Use of identifiers and references

All elements defined in this specification ***shall*** be identified by a Uniform Resource Identifier (URI) using the `@id` attribute. The `@id` ***shall*** be expressed in a relative form and ***shall not*** be combined with an `@xml:base` and ***shall*** be relative to the defined service. The `@id` values are not meant to imply any mandatory convention outside of the basic URI format and *may* be defined by the provider.
Reusable elements defined in this standard ***shall*** be locatable by their @id attribute and therefore their `@id` attributes ***shall*** result in a URL relative to the service base.

7.4. Use of namespaces

All documents based on this standard ***shall*** declare the following namespaces as part of the document element or use fully qualified namespaces.

Table 2 - Namespaces

Recommended Prefix	Use	Namespace
<default>	M	http://www.scte.org/schemas/224
xlink	CM	http://www.w3.org/1999/xlink
action	O	urn:scte:224:action
audience	O	urn:scte:224:audience
Conventions used in the tables: For attributes: M=Mandatory, O=Optional, OD=Optional with Default Value, CM=Conditionally Mandatory. For elements: <minOccurs>..<maxOccurs>, where N=unbounded		

Example **`Media`** element with namespace declarations:

```
<Media xmlns="http://www.scte.org/schemas/224"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  id="/media/1" description="TRU" lastUpdated="2018-02-03T19:31:32.3280038Z">
  <MediaPoint id="/mp1" description="Start of Program 1">
    <Apply>
      <Policy xlink:href="/p/1"></Policy>
    </Apply>
  </MediaPoint>
</Media>
```

7.5. Use of XLink

Elements of type **`<ReusableType>`** or **`<AuditType>`**, and their derivatives, allow the use XLink to reference existing resources. These XLink references shall be type "simple" as defined in the XLink Recommendation.

8. Data Model (Normative)

A provider manages the ESNI data using the interface defined in Section 9 on a distributor's Event Scheduling and Notification Interface set of services. The provider manages data model entities for **`Media`**, **`Policy`**, **`ViewingPolicy`**, and **`Audience`** and associated dependent elements of which an overview is shown in Figure 2.
The data model also supports an Audit element, which can be accessed by the provider on the distributor's Event Scheduling and Notification Interface service to audit the application of ESNI data.

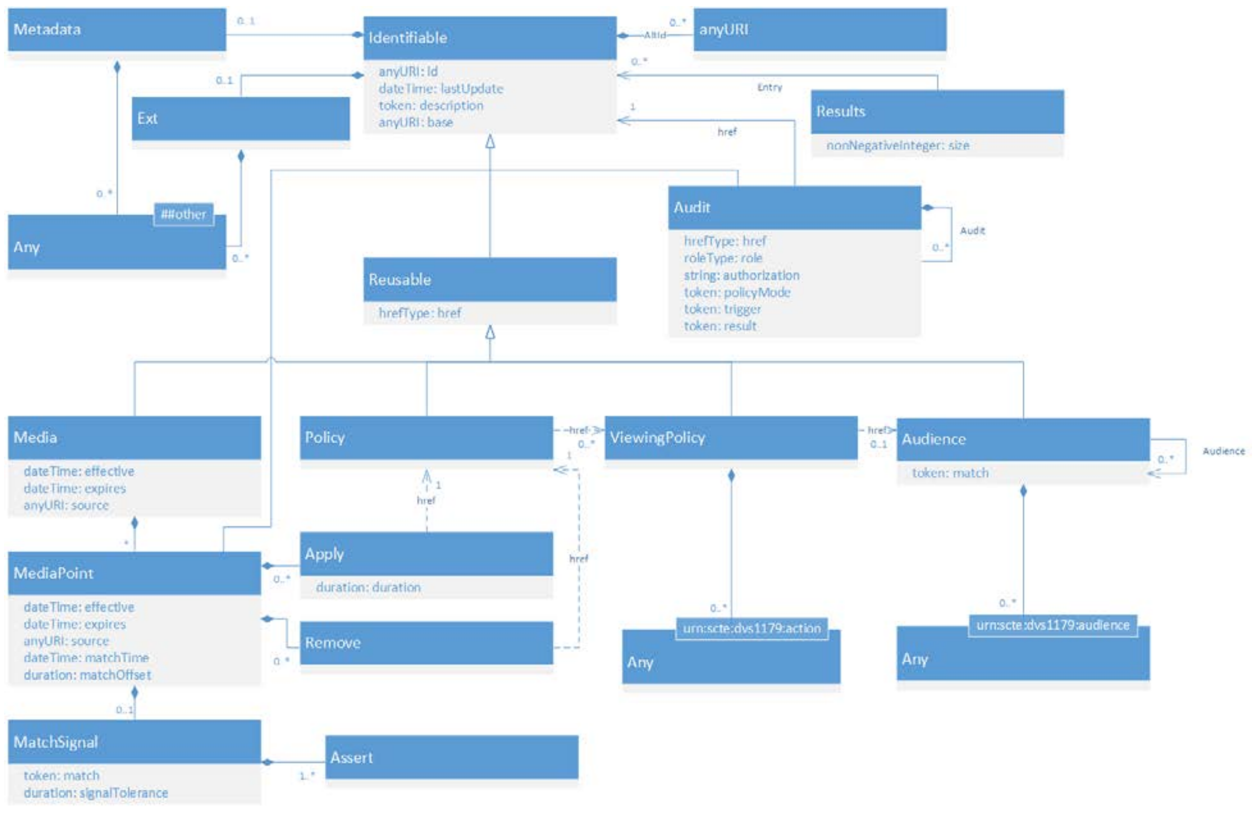


Figure 2 - Data Model UML Class Diagram

8.1. IdentifiableType

IdentifiableType is an abstract base type, which includes basic attributes for identification, description, and versioning, as well as a collection of alternative ID values to support correlation with other identification schemes and an Ext element for general extensibility.

Table 3 - Semantics of the IdentifiableType Complex Type

Element or Attribute Name	Use	Description
`@id`	CM	A Uniform Resource Identifier (URI). This value ***shall*** directly identify the resource that it represents and *may* also locate the identified resource, however, it ***shall not*** be presumed to do so unless provided as a URL per 7.3. This attribute ***shall*** be provided unless xlink:href is present.
`@description`	O	A common description used for the resource. See the derived types for guidance on how to use this attribute. This value is not required to be unique and ***shall not*** be used to definitively identify the content. The language of this attribute content ***shall*** be consistent throughout the document. Additional language descriptions *may* be provided through the **`Metadata`** element.
`@lastUpdated`	CM	The date and time when the resource was last modified. Primary versioning is only mutable by the provider. This value is mandatory if the `@id` attribute is also provided. If the element contains an external resource that has been updated, the provider ***shall*** update this value.
`@xml:base`	O	Explicitly provides the base URI for the purpose of resolving the relative URIs contained in `@xlink:href` attributes. See 7.3 for more details.
`AltID`	0..N	*May* be used to express additional identifiers (e.g. EIDR for entertainment or Ad-ID for advertisements) that identify the associated content. Alternative identification *may* be used for house identifiers or identification that does not possess a global namespace or uniqueness. The values used in this element are contextual to its parent element.
`Metadata`	0..1	Allows for descriptive metadata associated with this resource. Important to note that it allows for any metadata from any namespace and does not prescribe a specific metadata format nor validate the structure of that metadata.
`Ext`	0..1	This element is provided for extensibility. It *may* be used to contain any attribute or element from any other namespace.
Conventions used in the tables: For attributes: M=Mandatory, O=Optional, OD=Optional with Default Value, CM=Conditionally Mandatory. For elements: <minOccurs>..<maxOccurs>, where N=unbounded		

8.1.1. AltID

The Alternate Identifier *may* be used to express additional identifiers (e.g. EIDR, Ad-ID, etc.) that identify the associated content. Alternative identification *may* be used for house identifiers or identification that does not possess a global namespace or uniqueness. The values used in this element are contextual to its parent element. Details of a provider's implementation of the **`AltID`** element should be described in the provider's User's Guide (see Appendix C).

Table 4 - Semantics of the AltID element

Element or Attribute Name	Use	Description
`@description`	O	A freetext description or label for the alternate identifier being provided.

Example AltID element of a Media, with "description" attribute.

```
<MediaPoint id="media/tbs" description="Some Program" lastUpdated="2017-08-09T14:58:21.7767975Z">
        <AltID description="Ad-ID">CNPA0484000H</AltID>
</MediaPoint>
```

8.1.2. Metadata

Metadata allows for descriptive metadata associated with a resource. Metadata allows for any metadata in any format ranging from the standard urn:scte:224:metadata, or SCTE 236, or a proprietary namespace. The values used in this element are contextual to its parent element. Details of a provider's implementation of the **`Metadata`** element should be described in the provider's User's Guide (see Appendix C).

Table 5 - Semantics of the Metadata element

Element or Attribute Name	Use	Description
`Any`	0..N	An element provided using either the standard urn:scte:224:metadata namespace or a proprietary namespace and expressing a property of the containing resource. Common metadata properties are defined by SCTE external to this specification.

8.2. ReusableType

ReusableType is an abstract base type extended from IdentifiableType. ReusableType adds an optional attribute `@xlink:href` to be used when the element is provided as a reference to a resource of the same type.
ReusableType elements can be expressed in two modes: Globally Reusable or Reference. Globally Reusable elements have a unique identifier, provided in the `@id` attribute, and are intended to be referenced and reused from other elements, but managed separately. Reference elements occur within another identifiable element and include an `@xlink:href` attribute, which identifies a globally reusable element.
This approach allows a flexible application of reuse modes for **`Media`**, **`Policy`**, **`ViewingPolicy`** and **`Audience`** elements to support efficient and concise data model maintenance.

Table 6 - Reusable Modes

	Child of IdentifiableType	***shall* have @id**	***shall* have @xlink:href**	***may* have child elements**
Globally Reusable	No	Yes	No	Yes
Reference	Yes	No	Yes	No

Table 7 - Semantics of the ReusableType Complex Type

Element or Attribute Name	**Use**	**Description**
`@id`	CM	Inherited from *IdentifiableType* (See 8.1)
`@description`	O	Inherited from *IdentifiableType* (See 8.1)
`@lastUpdated`	CM	Inherited from *IdentifiableType* (See 8.1)
`@xml:base`	O	Inherited from *IdentifiableType* (See 8.1)
`@xlink:href`	CM	Presence of this attribute indicates that this element is a reference to an existing resource. The resource is both identifiable and locatable using the URL provided in this attribute. The presence of this attribute explicitly indicates a reference to a globally defined element. Therefore, it ***shall not*** be present along with the `@id` attribute and further, any other attribute that is expressed ***shall*** be ignored as it *may* conflict with the identified referenced element.
`AltID`	0..N	Inherited from *IdentifiableType* (See 8.1.1)
`Metadata`	0..1	Inherited from *IdentifiableType* (See 8.1.2)
`Ext`	0..1	Inherited from *IdentifiableType* (See 8.1)
Conventions used in the tables: For attributes: M=Mandatory, O=Optional, OD=Optional with Default Value, CM=Conditionally Mandatory. For elements: <minOccurs>..<maxOccurs>, where N=unbounded		

8.3. Media

The term, Media, is purposely broad and is applicable to all forms of on-demand, time-shifted and linear content, as well as other forms of media not explicitly mentioned here. Media is instantiated as a global **`Media`** element, which contains a collection of **`MediaPoint`** elements.

A **`Media`** element *may* be associated with one or more sources of content that comprise a branded presentation experience. The source identifier is important for matching signals (time or in-band), which *may* trigger the application (or removal) of a given Policy. In the case where a subscriber presentation is drawn from multiple sources, the definition of the set of sources comprising a given subscriber experience is out of scope of this standard.

The primary identification mechanism of the **`Media`** element is the `@id` attribute. The **`Media`** element *may* carry or provide alternative identifiers by means of the contained child **`AltID`** element. The `@description` attribute provides expanded descriptive text. The **`Metadata`** child element *may* be used for carrying more complex descriptive metadata data structures, potentially from other namespaces.

The main purpose of the **`Media`** element is to provide contextual encapsulation for **`MediaPoint`** elements, which are defined in section 8.4.

Table 8 - Semantics of the Media element

Element or Attribute Name	Use	Description
`@id`	CM	Inherited from *IdentifiableType* (See 8.1) This value *may* directly identify the content feed that it represents, but it ***shall not*** be presumed to do so. If this value does not directly identify the associated content feed then at least one **`AltID`** ***shall*** be provided that does identify the content, network or asset.
`@description`	O	Inherited from *IdentifiableType* (See 8.1) A common description used for the Media, which *may* be a call-sign commonly used for a linear network such as “ESPN” or an asset title such as “Titanic”.
`@lastUpdated`	CM	Inherited from *IdentifiableType* (See 8.1)
`@xml:base`	O	Inherited from *IdentifiableType* (See 8.1)
`@xlink:href`	CM	Inherited from *ReusableType* (See 8.2)
`@effective`	O	The date and time when any contained child **`MediaPoint`** element first becomes eligible for consideration. This value ***shall not*** be greater than the start of any explicit **`MediaPoint`**`/@effective` attribute value.
`@expires`	O	The date and time when any contained child **`MediaPoint`** element is no longer eligible for consideration. This value ***shall not*** be less than any explicit **`MediaPoint`**`/@expires` attribute value.
`@source`	O	This value *may* be used to explicitly identify the default URI of the content associated with this **`Media`** element. Providing a default `@source` attribute would be helpful if this **`Media`** element is used to reference multiple content sources.
`AltID`	0..N	Inherited from *IdentifiableType* (See 8.1.1)

Element or Attribute Name	Use	Description
`Metadata`	0..1	Inherited from *IdentifiableType* (See 8.1.2) *May* be used here to include metadata describing the linear feed or asset. It is intended to be a contextual placeholder for use should a sender of the data wish to associate metadata with a given identifiable object. Any identifiers carried in the metadata, or what the metadata data model used are outside the scope of this standard.
`Ext`	0..1	Inherited from *IdentifiableType* (See 8.1)
`MediaPoint`	0..N	A sequence of **`MediaPoint`** elements. See 8.4 for semantics. An empty set of MediaPoints indicates that there are no applicable MediaPoints for the time range defined by `@effective` and `@expires`. The **`MediaPoint`** elements are expressed in the order in which they *should* be considered within the **`Media`** element. Therefore, the order of **`MediaPoint`** elements ***shall*** be preserved.
Conventions used in the tables: For attributes: M=Mandatory, O=Optional, OD=Optional with Default Value, CM=Conditionally Mandatory. For elements: <minOccurs>..<maxOccurs>, where N=unbounded		

Example **`Media`** (namespace declarations omitted for readability):

```
<Media id="/media/tbs" description="TBS" lastUpdated="2018-02-10T12:00:00+00:00">
  <!-- Network / Super -->
  <MediaPoint id="/network/1" description="TBS">
    <AltID>http://dx.doi.org/10.5239/C370-DCA5</AltID>
    <Apply>
      <Policy xlink:href="/policy/6"/>
    </Apply>
  </MediaPoint>
  <!-- Program Begins -->
  <MediaPoint id="program/20997C44" matchTime="2014-11-05T12:00:00Z" >
    <AltID>http://dx.doi.org/10.5240/9EF1-2DA2-5C1F-98B4-F784-E</AltID>
    <Apply duration="PT2H">
      <Policy xlink:href="/policy/5"/>
    </Apply>
    <MatchSignal match="ALL">
      <Assert>/SpliceNull</Assert>
      <Assert>//SegmentationUpid[@segmentationTypeId='8']="20997C44"</Assert>
    </MatchSignal>
  </MediaPoint>
</Media>
```

8.4. MediaPoint

A MediaPoint is a temporal point of interest within the context of a Media. A MediaPoint is used to represent some change in state or to signal an expected event, or unexpected events such as program over/under-runs. MediaPoints are associated with specific locations in the content using either timestamps or embedded signals.

The evaluation of a MediaPoint is bounded by the MediaPoint's `@effective` and `@expires` window. By default, a MediaPoint is only evaluated once, meaning that once a MediaPoint `@matchTime` or **`MatchSignal`** criteria is met, the MediaPoint is no longer evaluated. If a MediaPoint *shall* be evaluated multiple times, the `@reusable` attribute should be set to "true".

If `@reusable` is true, a MediaPoint *shall* be evaluated each time it is triggered throughout its `@effective` and `@expires` window. A MediaPoint is also only evaluated once if the `@reusable` attribute is present and set to "false".

MediaPoints are primarily used to trigger changes in Policy (See 8.5), but *may* be used to provide additional metadata such as alternative identifiers (**`AltID`**), descriptive text (`@description`) or metadata structures from other namespaces, using the **`Metadata`** element.

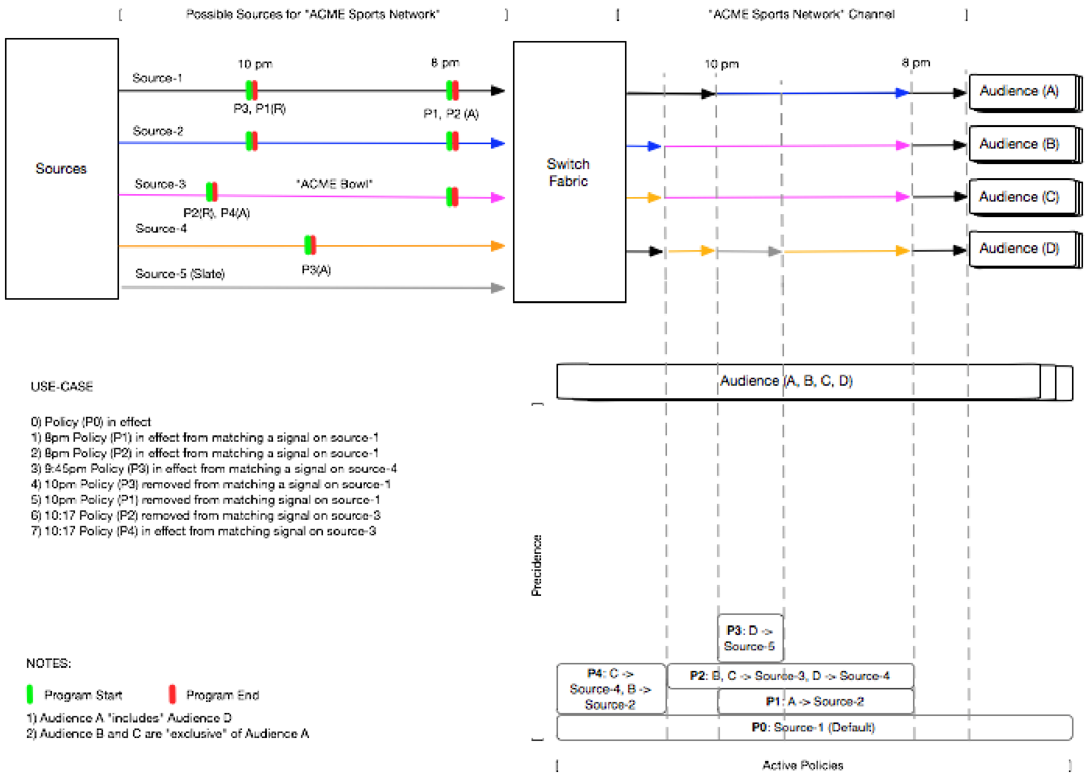
Policy applied by a MediaPoint without a `@matchTime`, `@matchOffset` or **`MatchSignal`** is herein referred to as "resident" (i.e. the Policy within the bounds of the Media is always to be considered/evaluated). Resident Policy is subject to the constraints of @effective and @expires attributes.

If a **`MediaPoint`** element includes both a `@matchTime` attribute and **`MatchSignal`** element, the first criteria to be met ***shall*** be applied. If the time in the `@matchTime` attribute, potentially adjusted by `@signalTolerance` attribute, is reached in the absence of a signal matching the **`MatchSignal`** element, the Policy ***shall*** be applied. Once the time is reached and the Policy is applied, the MediaPoint ***shall*** no longer be evaluated. If a signal does arrive after the time is reached, it *may* apply to another MediaPoint (i.e., arrival is within another MediaPoint's @effective and @expires window.

If a signal matching the **`MatchSignal`** element is received prior to reaching the time in the `@matchTime` attribute, potentially adjusted by `@signalTolerance`, the Policy ***shall*** be applied and the MediaPoint ***shall*** no longer be evaluated**.**
A MediaPoint provides the means to match a temporal location either via a signal or wall clock time. If by a signal, the source of content identified by the `@source` (or by the **`Media`** element) ***shall*** be used to perform the match.
A MediaPoint ***shall*** be associated to a single content source identified by the `@source` attribute. If the `@source` attribute has been omitted, the source is assumed to be identified by the **`Media`**`/@source` attribute or implied by the **`Media`**`/@id`. The value provided *may* be the identifier of a source from a pool of possible sources which *may* be drawn upon for use in presentation to an Audience. In either case, the actual experience delivered to a given Audience is dictated by the Policies referenced in a MediaPoint. A clarifying example is provided below. Two important observations:

1. The "Media" encapsulates Policy potentially for a branded network/channel as an Audience would see it.
2. The Policy activation as a result of a signal match or time match could be targeted to a particular source that *may* be part of an Audience presentation. More specifically, a MediaPoint *may* be tied to a single content source identified by the `@source` attribute for the purposes of Policy application and removal. The `@source` attribute on the MediaPoint takes precedence over the `@source` attribute on the parent **`Media`**.

The "base" or "default" source can either be an implementation specific configuration or could be a resident Policy specifying which source *should* be presented in the absence of any other applied Policy. Resident Policy is still considered per the precedence rules as outlined in section 9.



USE-CASE

0) Policy (P0) in effect
1) 8pm Policy (P1) in effect from matching a signal on source-1
2) 8pm Policy (P2) in effect from matching a signal on source-1
3) 9:45pm Policy (P3) in effect from matching a signal on source-4
4) 10pm Policy (P3) removed from matching a signal on source-1
5) 10pm Policy (P1) removed from matching signal on source-1
6) 10:17 Policy (P2) removed from matching signal on source-3
7) 10:17 Policy (P4) in effect from matching signal on source-3

1) Audience A "includes" Audience D
2) Audience B and C are "exclusive" of Audience A

Figure 3 - Policy Stacking Use-Case

Table 9 - Semantics of the MediaPoint element

Element or Attribute Name	Use	Description
`@id`	O	Inherited from *IdentifiableType* (See 8.1) Although the `@id` is optional, it ***shall*** be unique within the provider's identity, if provided.
`@description`	O	Inherited from *IdentifiableType* (See 8.1) A common description, which *may* be used to express the name of a program, chapter or advertisement.
`@lastUpdated`	CM	Inherited from *IdentifiableType* (See 8.1) This value, when provided, ***shall not*** be greater than the parent **`Media/@lastUpdated`** and ***shall*** correctly identify the time when the **`MediaPoint`** was last changed within the context of the parent **`Media`** element. This value *may* be used by the distributor to aid in updating existing **`Media`** resources by indicating which **`MediaPoint`**(s) have changed.

Element or Attribute Name	**Use**	**Description**
`@xml:base`	O	Inherited from *IdentifiableType* (See 8.1)
`@effective`	O	The time when this **`MediaPoint`** first becomes eligible for consideration. When omitted, the **`Media`**`/@effective` attribute ***shall*** be used.
`@expires`	O	The time when this **`MediaPoint`** is no longer eligible for consideration. When omitted, the **`Media`**`/@expires` ***shall*** be used.
`@matchTime`	O	The time at which referenced Policy(s) ***shall*** be applied or removed. `@matchTime` ***shall*** be within the effective window. The actual time is impacted by `@signalTolerance` as described in the text above. Note: matchTime *may* be used to specify an expected cue time and *shall* only take precedence if a matchSignal is not specified for the MediaPoint.
`@matchOffset`	O	When provided, represents an offset from the beginning of the VOD or time-shifted asset, expressed as duration. The definition of the beginning of an asset is outside the scope of this specification.
`@source`	O	Uniquely identifies which source **`MatchSignal`** *should* be matched against. If not present, the source ***shall*** be identified by the parent **`Media`**`/@source` attribute. If neither the **`MediaPoint`** nor **`Media`** have a `@source` attribute then this value is undefined and left to be interpreted by the implementation.
`@expectedDuration`	O	When provided, represents the duration of the activity described by the MediaPoint. For example, for a program start it *may* represent duration of program, for an ad avail, it *may* represent the avail duration.
`@order`	O	When provided, identifies the order of MediaPoints within a Media, starting with “0” as the first MediaPoint.
`@reusable`	O	When provided, designates a MediaPoint for multiple evaluations throughout its effectiveness window, and not just the first time it matches. TRUE or FALSE
`AltID`	0..N	Inherited from *IdentifiableType* (See 8.1.1)

Metadata	0..1	Inherited from *IdentifiableType* (See 8.1.2) *May* be used here to include metadata describing the program, chapter or advertisement that begins at this **MediaPoint**.
Ext	0..1	Inherited from *IdentifiableType* (See 8.1)
Remove	0..N	Upon a match (time or signal) the referenced Policy(s) ***shall*** be removed.
Apply	0..N	Upon a match (time or signal) the referenced Policy(s) ***shall*** be applied.
MatchSignal	0..1	If supplied and within the bounds of `@effective` and `@expires` *should* be evaluated for matching upon the detection of any signal in the referenced Media. See 8.4.1 for semantics.
Conventions used in the tables: For attributes: M=Mandatory, O=Optional, OD=Optional with Default Value, CM=Conditionally Mandatory. For elements: <minOccurs>..<maxOccurs>, where N=unbounded		

Example **MediaPoint** (namespace declarations omitted for readability):

```
<MediaPoint id="/program/20997C44" description="World Cup Curling"
  matchTime="2014-11-05T12:00:00+00:00" source="source-1">
  <AltID>http://dx.doi.org/10.5240/9EF1-2DA2-5C1F-98B4-F784-E</AltID>
  <Apply duration="PT2H">
    <Policy xlink:href="/policy/5"/>
  </Apply>
</MediaPoint>
```

8.4.1. MatchSignal

MatchSignal provides the context to evaluate and identify any component of a fully expressed XML representation of a qualified SCTE 35 signal as a set of XPath 2.0 assertions. When the assertions are evaluated and found to return an affirmative response, the Policy related to the MatchSignal would then also be evaluated. The **MatchSignal** element wraps a sequence of **Assert** XPath expressions and if the set of XPath 2.0 assertions yield a negative result, the Policy related to the MatchSignal ***shall not*** be removed or applied. The `@match` attribute needs to be consulted for the level of inclusivity/exclusivity.

The MatchSignal would be a match, which results in application (or removal) of the Policy if all of the **Assert** Xpath expressions evaluate to true when the `@match` is set to “ALL”.

The MatchSignal would be a match, which results in application (or removal) of the Policy if any of the **Assert** Xpath expression evaluates to true when the `@match` is set to “ANY”.

The MatchSignal would be a match, which results in application (or removal) of the Policy only if each **Assert** Xpath expression evaluates to false when the `@match` is set to “NONE”.

Table 10 - Semantics of the MatchSignal element

Element or Attribute Name	Use	Description
@match	OD default: ALL	One of the following values: ALL – All Assert statements ***shall*** evaluate to true ANY – At least one Assert statement ***shall*** evaluate to true NONE – None of the Assert statements ***shall*** evaluate to true

`@signalTolerance`	O	A duration, that when added to the **`MediaPoint`**`/@matchTime` is the time at which the parent MediaPoint ***shall*** be activated if the signal has not yet been matched.
`Assert`	1..N	This element carries an XPath 2.0 expression as if it were going to be evaluated against the SCTE 35 schema compliant XML document. An implementation *should* evaluate the expression against the SCTE 35 schema compliant document but *may* choose other methods outside the scope of this standard. An expression, when evaluated, results in a positive match for the **`Assert`**, returning true as if the fn:boolean() had been applied, false otherwise.
Conventions used in the tables: For attributes: M=Mandatory, O=Optional, OD=Optional with Default Value, CM=Conditionally Mandatory. For elements: <minOccurs>..<maxOccurs>, where N=unbounded		

Example SCTE 35 signal (namespace declarations omitted for readability):

```
<SpliceInfoSection>
  <TimeSignal>
    <SpliceTime ptsTime="491203647"/>
  </TimeSignal>
  <SegmentationDescriptor segmentationEventId="1342177265">
    <DeliveryRestrictions noRegionalBlackoutFlag="false" archiveAllowedFlag="false"
      webDeliveryAllowedFlag="false" deviceRestrictions="3"/>
    <SegmentationUpid segmentationUpidType="8">000000001EA25F7D</SegmentationUpid>
  </SegmentationDescriptor>
  <AvailDescriptor providerAvailId="555"/>
</SpliceInfoSection>
```

A **`MatchSignal`** below would evaluate to ‘true’

```
<MatchSignal match="ALL">
  <Assert>//SegmentationDescriptor/SegmentationUpid[@segmentationUpidType=8]</Assert>
  <Assert>//SegmentationDescriptor/SegmentationUpid[text()='000000001EA25F7D']</Assert>
</MatchSignal
```

A **`MatchSignal`** below would evaluate to ‘true’

```
<MatchSignal match="ALL">
  <Assert>//SegmentationUpid[@segmentationUpidType=8 and .='000000001EA25F7D']</Assert>
</MatchSignal
```

A **`MatchSignal`** below would evaluate to ‘false’ because the segmentation UPID does not match.

```
<MatchSignal match="ALL">
  <Assert>//SegmentationDescriptor/SegmentationUpid[@segmentationUpidType=8]</Assert>
  <Assert>//SegmentationDescriptor/SegmentationUpid[text()='000000001EA25F7E']</Assert>
</MatchSignal
```

A **`MatchSignal`** below would evaluate to ‘true’ because `@match` attribute specifies `ANY`.

```
<MatchSignal match="ANY">
  <Assert>//SegmentationDescriptor/SegmentationUpid[@segmentationUpidType=8]</Assert>
  <Assert>//SegmentationDescriptor/SegmentationUpid[text()='000000001EA25F7E']</Assert>
</MatchSignal
```

A **`MatchSignal`** below would evaluate to ‘true’ because the segmentation UPID does not match.

```
<MatchSignal match="NONE">
  <Assert>//SegmentationDescriptor/SegmentationUpid[@segmentationUpidType=8]</Assert>
  <Assert>//SegmentationDescriptor/SegmentationUpid[text()='000000001EA25F7E']</Assert>
</MatchSignal
```

8.4.2. Remove

Removes the referenced Policy so it is no longer in effect. Note, Remove, removes all instances of a Policy, even if the Policy is in the state from multiple Apply actions. Also note, that all Removes are handled prior to any Apply in a MediaPoint, if they are both present.

Table 11 - Semantics of the Remove element

Element or Attribute Name	Use	Description
`Policy`	1..1	A reference to the Policy to be removed.
Conventions used in the tables: For attributes: M=Mandatory, O=Optional, OD=Optional with Default Value, CM=Conditionally Mandatory. For elements: <minOccurs>..<maxOccurs>, where N=unbounded		

Example **`Remove`** (namespace declarations omitted for readability).

```
<MediaPoint id="/program/1343222" description="Accidental Happiness"
  matchTime="2014-11-05T12:00:00Z">
  <Remove>
    <Policy xlink:href="/policy/5"/>
  </Remove>
</MediaPoint>
```

8.4.3. Apply

Apply indicates the action to be taken on a Policy. Per this specific action, the `@priority` and then precedence order ***shall*** be enforced should more than one Policy be in effect. The `@priority` attribute is provided as a means for the provider to convey relative priority between resources. Details of a provider's implementation of the `@priority` attribute should be described in the provider's User's Guide (see Appendix C).

Table 12 - Semantics of the Apply element

Element or Attribute Name	Use	Description
`@duration`	O	The length of time that the associated Policy will be applied. When this value is present, the Policy ***shall*** be removed once the duration has passed from the time the Policy was actually applied. Note that the application time of the Policy is dictated by the parent **`MediaPoint`** element.
`@priority`	O	An integer value representing the priority at which this policy should be applied, with "0" being the highest.
`Policy`	1..1	A reference to the Policy to be applied.
Conventions used in the tables: For attributes: M=Mandatory, O=Optional, OD=Optional with Default Value, CM=Conditionally Mandatory. For elements: <minOccurs>..<maxOccurs>, where N=unbounded		

Example **`Apply`** (namespace declarations omitted for readability). Note in this example that the Policy will be implicitly removed after 2 hours based on the value of the `@duration` attribute.

```
<MediaPoint id="/program/20997C44" description="World Cup Curling"
  matchTime="2014-11-05T12:00:00+00:00" source="source-1">
  <Apply duration="PT2H" priority="10">
    <Policy xlink:href="/policy/5"/>
  </Apply>
</MediaPoint>
```

8.5. Policy

Policy defines one or more Actions against a defined set of Audience(s). A Policy *may* allow or restrict behaviors, presentation, or other end-user experiences related to the rights, protections, or other rules of the referenced Media. Note that application of Policy *may* be subject to managed infrastructure outside the control of the provider.
A Policy is a collection of ViewingPolicy(s), which is the literal encapsulation of Audience and one or more actions.
Alternate content policy is comprised primarily of a definition of an Audience-Action pair where the action defined is applied (or removed) to the audience of which a user of the Media *may* be a member of at any given point in time.
A Policy is applied or removed by association with a MediaPoint.

Table 13 - Semantics of the Policy element

Element or Attribute Name	Use	Description
`@id`	CM	Inherited from *IdentifiableType* (See 8.1)
`@description`	O	Inherited from *IdentifiableType* (See 8.1)
`@lastUpdated`	CM	Inherited from *IdentifiableType* (See 8.1)
`@xml:base`	O	Inherited from *IdentifiableType* (See 8.1)
`@xlink:href`	CM	Inherited from *ReusableType* (See 8.2)
`AltID`	0..N	Inherited from *IdentifiableType* (See 8.1.1)
`Metadata`	0..1	Inherited from *IdentifiableType* (See 8.1.2)
`Ext`	0..1	Inherited from *IdentifiableType* (See 8.1)
`ViewingPolicy`	0..N	At least one ***shall*** be provided if **`Policy`** element is not a reference (i.e. `@xlink:href` is not present).
Conventions used in the tables: For attributes: M=Mandatory, O=Optional, OD=Optional with Default Value, CM=Conditionally Mandatory. For elements: <minOccurs>..<maxOccurs>, where N=unbounded		

Example **`Policy`** (namespace declarations omitted for readability).

```
<Policy id="/policy/1">
  <ViewingPolicy xlink:href="/viewingPolicy/1"/>
  <ViewingPolicy xlink:href="/viewingPolicy/2"/>
  <ViewingPolicy xlink:href="/viewingPolicy/3"/>
</Policy>
```

8.6. ViewingPolicy

A ViewingPolicy is an association of one or more actions to a defined Audience. Policy is said to be applied (or be removed) when the attributes of a user or device match any of the Audience(s) in a ViewingPolicy. Upon a single match, the system ***shall not*** continue evaluation of **`Audience`** elements within a Policy context (i.e. within a Policy, the first ViewingPolicy of which an Audience is a member applies). However, multiple Policy(s) *may* be evaluated at the same time resulting in overlapping Audiences with potentially conflicting Actions. In such a scenario, the enforcement ***shall*** follow the Policy precedence rules outlined in section 9.5.

Table 14 - Semantics of the ViewingPolicy element

Element or Attribute Name	Use	Description
`@id`	CM	Inherited from *IdentifiableType* (See 8.1)
`@description`	O	Inherited from *IdentifiableType* (See 8.1)
`@lastUpdated`	CM	Inherited from *IdentifiableType* (See 8.1)
`@xml:base`	O	Inherited from *IdentifiableType* (See 8.1)
`@xlink:href`	CM	Inherited from *ReusableType* (See 8.2)
`AltID`	0..N	Inherited from *IdentifiableType* (See 8.1.1)
`Metadata`	0..1	Inherited from *IdentifiableType* (See 8.1.2)
`Ext`	0..1	Inherited from *IdentifiableType* (See 8.1)
`Audience`	0..1	See 8.7 for semantics. This element ***shall*** be provided if the parent element is not a reference (i.e. `@xlink:href` is not present). Only one **`Audience`** element is allowed here, however an **`Audience`** *may* be composed of multiple **`Audience`** elements.
`Any`	0..N	An element provided using either the standard urn:scte:224:action namespace or a proprietary namespace and expressing an action that corresponds to this ViewingPolicy. Common actions are defined by SCTE external to this specification. At least one element ***shall*** be provided if the parent element is not a reference (i.e. `@xlink:href` is not present).
Conventions used in the tables: For attributes: M=Mandatory, O=Optional, OD=Optional with Default Value, CM=Conditionally Mandatory. For elements: <minOccurs>..<maxOccurs>, where N=unbounded		

Example **`ViewingPolicy`** (namespace declarations omitted for readability):

```
<ViewingPolicy id="/simple">
  <Audience xlink:href="audience/1"/>
  <action:MaxResolution>480</action:MaxResolution>
</ViewingPolicy>
```

8.7. Audience

An Audience is a set of characteristics, which together define a subset of users. An Audience *may* be characterized in many ways. For example, bound to geography, device or device capability.

Audience and Action definitions are extensible, however values of the **`Any`** element types ***shall*** have unambiguous definitions so that behavior of the execution is deterministic regardless of implementation.

An **`Audience`** element *may* be comprised of other **`Audience`** elements. The `@match` ALL|ANY|NONE value applies to both contained **`Audience`** elements and Audience properties provided using the **`Any`** element. For instance, if `@match` is ALL then all Audience properties and all contained **`Audience`** elements ***shall*** match the context for the containing **`Audience`** to match. If referenced, the **`Audience`** ***shall*** have been previously defined and provided.

Table 15 - Semantics for the AudienceType Complex Type

Element or Attribute Name	Use	Description
`@id`	CM	Inherited from *IdentifiableType* (See 8.1)
`@description`	O	Inherited from *IdentifiableType* (See 8.1)
`@lastUpdated`	CM	Inherited from *IdentifiableType* (See 8.1)
`@xml:base`	O	Inherited from *IdentifiableType* (See 8.1)
`@xlink:href`	CM	Inherited from *ReusableType* (See 8.2)
`@match`	OD default: `ALL`	One of the following values: `ALL` – All nested **`Audience`** elements and/or Audience properties ***shall*** evaluate to true `ANY` – At least one nested **`Audience`** element or Audience property ***shall*** evaluate to true `NONE` – None of the nested **`Audience`** elements and/or Audience properties ***shall*** evaluate to true
`AltID`	0..N	Inherited from *IdentifiableType* (See 8.1.1)
`Metadata`	0..1	Inherited from *IdentifiableType* (See 8.1.2)
`Ext`	0..1	Inherited from *IdentifiableType* (See 8.1)
`Audience`	0..N	Potentially nested **`Audience`**
`Any`	0..N	An element provided using either the standard urn:scte:224:audience namespace or a proprietary namespace and expressing a property of the containing **`Audience`**. Common audience properties are defined by SCTE external to this specification.
Conventions used in the tables: For attributes: M=Mandatory, O=Optional, OD=Optional with Default Value, CM=Conditionally Mandatory. For elements: <minOccurs>..<maxOccurs>, where N=unbounded		

Example **`Audience`** (namespace declarations omitted for readability):

```
<Audience id="/audience/1" match="ANY">
  <audience:Zip>30062</audience:Zip>
</Audience>
```

8.8. Results

A **`Results`** element is used to return a collection of **`<IdentifiableType>`** elements. This wrapper element ***shall*** be used in GET query requests to the ESNI interface, notably the request for **`Audit`** elements, to return a collection of elements which meet the specified query parameter criteria. The **`Results`** wrapper element allows for pagination given the potential for large result sets as described in 9.4.

Table 16 - Semantics for the Results element

Element or Attribute Name	Use	Description
`@size`	O	Indicates the total number of entries which meet the query criteria associated with the Results. The value *may* change between query requests. This may exceed the number IdentifiableType entries contained as a result of pagination.
<**`IdentifiableType`**>	0..N	An element which extends IdentifiableType and meets the query criteria.
Conventions used in the tables: For attributes: M=Mandatory, O=Optional, OD=Optional with Default Value, CM=Conditionally Mandatory. For elements: <minOccurs>..<maxOccurs>, where N=unbounded		

Example **`Results`** (namespace declarations omitted for readability):

```
<Results size="2">
  <Audit id="/auditguid1" xlink:href="/audience/1"
    authorization="admin" lastUpdated="2018-04-01T02:00:00+00:00"
    trigger="PUT" xlink:role="Audience" result="SUCCESS" />
  <Audit id="/auditguid2" xlink:href="/audience/2"
    authorization="admin" lastUpdated="2018-04-01T02:00:00+00:00"
    trigger="PUT" xlink:role="Audience" result="SUCCESS" />
</Results>
```

8.9. Audit

An Audit represents a historical event. Audit elements capture actions and activities that directly result in either the state of the implementing system to be altered (such as the application or removal of a Policy) or any evaluation of MediaPoints, Audience(s), Policy, or any other system representation of elements described in this document (such as result of the Assert statements contained in a **`MatchSignal`** element). An Audit *may* also represent a result of a status query request or a decision query request.

The Audit activity ***shall*** capture the detail as described of at least, but not limited to, the following actions or activities:

1. Any service call:
   - **`Media`**
   - **`Audience`**
   - **`Policy`**
   - **`ViewingPolicy`**
2. Any state change (including system errors or exception states). For example, the PUT'ing of a **`Media`** element *may* result in a change of state of Policy(s).
3. Any evaluation of the following: **`Assert`**, **`ViewingPolicy`** match (Audience member inclusion)
4. Status query request: Status of **`Policy`** and **`ViewingPolicy`** for an **`Audience`** or **`Media`**.
5. Decision query request: Messages that will be used and actions that would be returned for a particular scenario described by the **source**, **audiencetype**, **audiencevalue**, **segmentationupidtype**, and **signaled**, for future events (see 9.5).

The `@description` of the Audit *may* contain notes relevant to the Audit entry.

Table 17 - Semantics for the AuditType Complex Type

Element or Attribute Name	Use	Description
`@id`	CM	Inherited from *IdentifiableType* (See 8.1)

@description	O	Inherited from *IdentifiableType* (See 8.1)
@lastUpdated	CM	Inherited from *IdentifiableType* (See 8.1) This value ***shall*** be the absolute time at which the action described in the @trigger attribute took place. This value ***shall*** always be present unless the value of @trigger is NONE.
@xml:base	O	Inherited from *IdentifiableType* (See 8.1)
@xlink:href	CM	A reference to the resource being reported in this Audit. This ***shall*** match the @id value originally used. Note that this use of @xlink:href is not inherited from ReusableType and thus *may* be used in combination with the @id attribute.
@xlink:role	CM	When @xlink:href is provided, this attribute ***shall*** be present and provide the name of the resource being referenced. It *may* be the element name of any IdentifiableType (e.g. **Media**, **Policy**, etc.)
@authorization	CM	Authorization token used (if API) or if an action was taken (Policy applied), the authorization of the system that PUT (set) the **Policy** element. When the trigger is associated with an Authorization event then this value ***shall*** be provided.
@policyMode	CM	Indicates either Policy application or removal captured by this Audit. One of: APPLY – A Policy was applied. The @trigger will explain the cause. REMOVE – A Policy was removed. The @trigger will explain the cause.
@trigger	M	Indicates the event that caused this Audit to take place. One of: NONE – No definable trigger TIME – Caused by arriving at either a defined @matchTime or @matchOffset, subject to @signalTolerance SIGNAL – Caused by matching an in-band signal DURATION – Caused by reaching the end of a defined duration. Note that this can only be used to indicate the removal of a Policy. GET – A client attempted to retrieve a resource from the server using a GET request. PUT – A client attempted to add or update a resource on the server using a PUT request. DELETE – A client attempted to delete a resource from the server using a DELETE request. STATUS – Provided as a result of a status request. MANUAL – The Audit was triggered manually.
@result	OD default:	One of the following values:

	`SUCCESS`	`SUCCESS` – The action defined in `@trigger` was successful or if the `@trigger` is STATUS then the associated Policy is active. The `@description` *may* be used as an informative note. `FAIL` – The action defined in `@trigger` was (fully or partially) unsuccessful or if the `@trigger` is STATUS then the associated Policy is inactive. The `@description` ***shall*** supply informative descriptive text of what caused the error.
`AltID`	0..N	Inherited from *IdentifiableType* (See 8.1.1)
`Metadata`	0..1	Inherited from *IdentifiableType* (See 8.1.2)
`Ext`	0..1	Inherited from *IdentifiableType* (See 8.1)
`Audit`	0..N	Recursive element used to provide more detail. The recursion is used to provide appropriate context. For example, a ViewingPolicy Audit provided as a child of a Policy Audit.
Conventions used in the tables: For attributes: M=Mandatory, O=Optional, OD=Optional with Default Value, CM=Conditionally Mandatory. For elements: <minOccurs>..<maxOccurs>, where N=unbounded		

Example **`Audit`** (namespace declarations omitted for readability):

```
<Results size="2">
  <Audit id="/auditguid1" xlink:href="/audience/1"
    authorization="admin" lastUpdated="2018-04-01T02:00:00+00:00"
    trigger="PUT" xlink:role="Audience" result="SUCCESS" />
  <Audit id="/auditguid2" xlink:href="/audience/2"
    authorization="admin" lastUpdated="2018-04-01T02:00:00+00:00"
    trigger="PUT" xlink:role="Audience" result="SUCCESS" />
</Results>
```

9. Service interface definition

The service definitions are simplified one-way interfaces (from provider to distributor) using REST style HTTP request/response with the following basic convention:
`https://<distributor determined base endpoint>/<provider determined resource URI>`
where together form a proper, fully qualified URL. For example:
Base: `https://esni.somecompany.com`
URI: `/media/1`
Fully qualified, locatable resource: `https://esni.somecompany.com/media/1`
`Media`, **`Policy`**, **`ViewingPolicy`**, **`Audience`**, **`Audit`** are the only defined managed resources.
Relative URI's appearing in a document are always resolved relative to either an element, a document entity, or an external entity however, @id is to always be relative to the document entity.

Assume, for example, that the distributor established a service endpoint at `http://host`. The provider submits a **`ViewingPolicy`** using a PUT to `http://host/vp/vp1`. The `@id` attribute of the resource ***shall*** always be relative to the service endpoint as shown:

```
<ViewingPolicy id="/vp/vp1">
  <Audience xlink:href="/a/44"/>
  <action:Content>SomeContent</action:Content>
</ViewingPolicy>
```

The following submission to the same URL ***shall*** fail:

```
<ViewingPolicy id="/vp1">
  <Audience xlink:href="/a/44"/>
  <action:Content>SomeContent</action:Content>
</ViewingPolicy>
```

Each of the following four examples are valid and equivalent ways to reference this **`ViewingPolicy`** within a **`Policy`** that is submitted using `PUT http://host/policy/p1`. (namespace declarations omitted for readability)

Relative to document element:

```
<Policy id="/policy/p1">
  <ViewingPolicy xlink:href="/vp/vp1"/>
</Policy>
```

Fully qualified:

```
<Policy id="/policy/p1">
  <ViewingPolicy xlink:href="http://host/vp/vp1"/>
</Policy>
```

Relative to explicit @xml:base:

```
<Policy id="/policy/p1" xml:base="http://host">
  <ViewingPolicy xlink:href="/vp/vp1"/>
</Policy>
```

Relative to explicit, local @xml:base:

```
<Policy id="/policy/p1">
  <ViewingPolicy xml:base="http://host/vp" xlink:href="/vp1"/>
</Policy>
```

Additionally, links *may* be provided to external entities:

```
<Policy id="/policy/p1">
  <ViewingPolicy xlink:href="http://someotherhost/vp/vp1"/>
</Policy>
```

This reference would fail given that the resulting URL (`http://host/vp1`) would not be valid:

```
<Policy id="/policy/p1">
  <ViewingPolicy xlink:href="/vp1"/>
</Policy>
```

When providing an unmanaged, **`<IdentifiableType>`** element as a child of a managed resource, the `@id` attribute represents the URL relative to the parent `@id` attribute. These unmanaged resources *should not* be submitted directly to the server on their own, but only as children of a managed resource. However, they *may* be retrieved (using GET) or referenced by the parent-qualified URL.

For example, the following **`MediaPoint`** element *may* be subsequently referenced as `/media/1/mp/1`:

```
<Media id="/media/1">
  <MediaPoint id="/mp/1"/>
</Media>
```

9.1. Broken References

Globally reusable **`Policy`**, **`ViewingPolicy`**, and **`Audience`** definitions can be managed independently of the application or removal of the Policy through `@xlink:href` on **`Media`**/**`MediaPoint`**(s).

Missing references to **`Policy`**, **`ViewingPolicy`**, and **`Audience`** or any change in the **`Media`** or **`MediaPoint`**(s) which would result in a broken reference ***shall not*** be allowed and implementations *should* make every effort to prevent broken references from occurring.

Any action that results in a reference of a managed resource being broken (referenced but not accessible from a GET to that resource) is considered invalid and ***shall*** result in an error preventing the action from occurring. Further, the implementation ***shall*** log an error, which will be available via the audit interface. A broken reference could result in the failure of the processing chain, therefore it is expected that implementers prevent the state from occurring. A GET request of a resource that does not or no longer exists is not a broken reference and ***shall*** result in a 404 status code.

9.2. Authentication and Authorization

It is strongly suggested that authentication and authorization use request signing as outlined in Appendix B. If request signing is used, clients ***shall*** also use TLS/SSL along with the authorization, as outlined here, for all service calls to the defined interfaces. The signing process ***shall*** follow the process specified in Appendix B – Request Signing.
A standard Date HTTP header ***shall*** also be used. The value supplied for the Date header ***shall*** use the same value used to compute the *Authorization* header. The value ***shall*** conform to the "preferred" Date/Time formats as specified in RFC 7231 and ***shall*** include a time zone designator. For example:
Sun, 06 Nov 1994 08:49:37 GMT
The Date value specified ***shall*** be within 5 minutes of receipt of the request, otherwise a failed authorization error ***shall*** result.
Distribution of the client identifier and secret used for the signing are outside of the scope of this standard.
Refer to the *Appendix B – URL Signing Specification,* RFC 2014 and RFC 7231 RFC for additional detail on signing.

9.3. Common Interface

All resources in this specification (**`Media`**, **`Audience`**, **`Policy`**, **`ViewingPolicy`**, **`Audit`**) ***shall*** support a common RESTful HTTP interface in addition to a general-purpose query interface, which *should* be some portion of the URL of the resource. For example, if the resource PUT is identified by /media/1 off of a base, the resource ***shall*** be available to GET /media/1, the general-purpose query interface could be off the base. Top-level resources are defined in the XML Schema as global elements with the exception of the Results element, which is a generic container.
The use of HTTP ***shall*** be required within the general RFC 7231 guidelines. Methods used outside the ones described explicitly by an implementation *should* respond with the appropriate HTTP status code.

9.3.1. GET

A successful request ***shall*** result in a response with HTTP status code 200 and the response body ***shall*** contain the resource formatted per this specification.
The service *may* use any valid HTTP status code (e.g. 3xx, 4xx, 5xx) to notify the client of an exception. The service ***shall not*** use a 2xx status code when there is an exception of any kind.
The service *may* provide either Last-Modified or ETag headers in a GET response and then process conditional requests for subsequent responses.

9.3.2. PUT

A successful request to PUT a new resource ***shall*** result in a HTTP status code 201. A request to PUT an existing resource (update) ***shall*** result in a HTTP status code of 204 with no content body.

The service *may* use any valid HTTP status code (e.g. 3xx, 4xx, 5xx) to notify the client of an exception. The service ***shall not*** use a 2xx status code when there is an exception of any kind. Some resources (e.g. **`Audit, MediaPoint`**) are read-only in which case any request to use this method ***shall*** result in HTTP status code 405.
Example Request:

```
PUT /media/1 HTTP/1.1
Host: esni.somecompany.com
Content-Type: application/xml
Date: Sun, 06 Nov 2018 08:49:37 GMT

Authorization: HMAC-SHA256 Credential=client_id/esni, SignedHeaders=content-type;date;host,
Signature=29dc7257e71b19699d57fb2313b2b36cf5422fb60c163a69882c877b1a6d3965

<?xml version="1.0" encoding="UTF-8"?>
<Media xmlns="http://www.scte.org/schemas/224"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  id="/media/1" description="TRU" lastUpdated="2018-02-03T19:31:32.3280038Z">
  <MediaPoint id="/mp1" description="Start of Program 1">
    <Apply>
      <Policy xlink:href="/p/1"></Policy>
    </Apply>
    <MatchSignal match="ALL">
      <Assert>//SegmentationUpid[@segmentationUpidType = 8]] </Assert>
      <Assert>//DeliveryRestrictions/@noRegionalBlackoutFlag[. = false()]]</Assert>
    </MatchSignal>
  </MediaPoint>
</Media>
```

Example Response:

```
HTTP/1.1 201 Created
```

9.3.3. DELETE

A successful request to DELETE an existing resource ***shall*** result in HTTP status code of 204. Some resources (e.g. **`Audit, MediaPoint`**) are read-only in which case any request to use this method ***shall*** result in HTTP status code 405.

9.3.4. POST

A POST request is not defined by this specification and *should* result in HTTP status code 405 (Method Not Allowed).

9.4. GET Queries

A GET against base service endpoint ***shall*** return a Results element including a list of **`<IdentifiableType>`** elements matching the specified query criteria. If the query-string is not present, then information for all **`<IdentifiableType>`** elements meeting the defaults is returned. An implementation *may* constrain the maximum returned dataset size if the requesting query is either unconstrained or exceeds an implementation's maximum. The query string parameters are used to scope the list requests, and follow the HTTP path portion of the request using the following standard form:
`?parameter=value&parameter=value...`

Table 18 - Service Query Parameters

Parameter	Use	Description
altID	O	Limit the results to entries with a specified **`AltID`**.
role	O	Limit the results to entries with a specified type (e.g. **`Audit`**, **`Audience`**, **`Media`**, **`MediaPoint`**, **`Policy`**, **`ViewingPolicy`**). If no role is specified then all types will be included. Multiple role values *may* be present.
limit	O	Indicates the maximum number of entries to be returned.
updatedAfter	O	Limit the results to entries which have `@lastUpdated` after the specified dateTime. This parameter *should* be URL-encoded. This parameter value ***shall*** chronologically follow updatedBefore.
updatedBefore	O	Limit the results to entries which have `@lastUpdated` before the specified dateTime. This parameter *should* be URL-encoded. This parameter value ***shall*** chronologically precede updatedAfter.
status	O	Use of this parameter explicitly calls for a status audit at a particular point in time defined by the value. The value ***shall*** be formatted as an explicit dateTime and *should* be URL-encoded.
offset	OD default: `0`	Specifies the starting entry in the results (zero-based list). If no offset is specified, the beginning of the list is the first entry in Results.
Conventions used in the tables: M=Mandatory, O=Optional, OD=Optional with Default Value		

9.5. GET Decision Request Queries

A GET against decision request service endpoint ***shall*** return a Results element including a list of elements used in providing a decision for the decision query request parameters: `?parameter=value&parameter=value...` The source, eventtime, audiencetype, and audiencevalue, are required to get a decision response, if the signal information is also available, then it is used as the trigger as well. The response to the query request is a Results message with the **Media**, **MediaPoint**(s), **Policy**(s), **ViewingPolicy**(s), **Audience**(s), and actions used to make the decision.

Table 19 – Decision Request Query Parameters

Parameter	Use	Description
source	M	The source of the media the user is trying to watch.
eventtime	M	The time of the event being requested
audiencetype	M	An enumeration that's used to describe the audience characterization type from either the standard urn:scte:224:audience namespace or a proprietary namespace and expressing a property of the containing **Audience**.
audiencevalue	M	simple value providing the audience characterization based on the type (i.e., for Zip audience it could be "80202".
segmentationupidtype	O	segmentationupidtype usually representing "16" a program start or "17" a program end. If the segmentationupidtype is not present, the request will be considered a time-based request.
signalid	O	Text value used to match in the MediaPoint's assert element along with the segmentationupidtype. If the signalid is not present, the request will be considered a time-based request
Conventions used in the tables: M=Mandatory, O=Optional, OD=Optional with Default Value		

9.6. Managing Foreign Reference Updates

It is expected that the distributor will resolve and retrieve foreign references (`@xlink:href` points to an external system) when they are encountered as the result of a PUT of a resource. However, as the distributor will not be aware of entity updates in the foreign reference, it is incumbent on the provider who originally sent the reference to notify the distributor of any updates. The provider ***shall*** signal the update to the distributor by setting the `@lastUpdated` attribute of the **`<IdentifiableType>`** containing the referenced element. The distributor ***shall*** re-evaluate the applicable references at this point and take resulting actions. An example follows.

Assume provider has PUT the following **`Policy`** to the distributor with a reference to **`ViewingPolicy`** defined in an external system (http://someotherhost):

```
<Policy id="/policy/p1" lastUpdated="2018-04-01T02:00:00+00:00">
  <ViewingPolicy xlink:href="http://someotherhost/vp/vp1"></ViewingPolicy>
</Policy>
```

If the definition of the ViewingPolicy is modified on "someotherhost" on the following day then the provider should notify the distributor by changing the `@lastUpdated` attribute of that **`Policy`** element:

```
<Policy id="/policy/p1" lastUpdated="2018-04-02T11:53:00+00:00">
  <ViewingPolicy xlink:href="http://someotherhost/vp/vp1"></ViewingPolicy>
</Policy>
```

10. Policy Lifecycle and precedence

It is important to understand the lifecycle of a Policy and how overlapping policies can be effectively managed.

10.1. Audience Membership

Audience *may* be comprised of other `Audience` elements. Audience describes characteristics of a population based on identified and extensible set of attributes. One is considered a member of that Audience if, depending on the matching rules of the Audience, any of the attributes are characteristic of that member. Additionally, one ***shall*** be considered a member of the first Audience attribute set that matches for a given Policy.

10.2. Policy Lifecycle

A Policy ***shall*** only be applied or removed through the use of MediaPoints. The impact of additions, subtractions, and updates to MediaPoints to a Media context are explained as follows:

1. There are two modes associated with the application of a Policy to a MediaPoint, "apply" and "remove" through the `Apply` and `Remove` elements respectively. A Policy is applied or removed when:
    a. the `MediaPoint` element has a `@matchTime` or `@matchOffset` value that matches the referenced relative time point and an `Apply` or `Remove` element is provided. If an `Apply/@duration` is specified, there is an implicit `Remove` to occur at the time that the Policy was applied plus the duration specified in the `@duration` attribute.
    b. the `MediaPoint` element contains a `MatchSignal` element that matches a corresponding SCTE 35 (or other) signal relative to the referenced `Media` and an `Apply` or `Remove` is specified, there is an implicit `Remove` indicated to occur at the time of receipt of the actual match of the signal plus the duration specified in the `@duration` attribute.
    c. both a `@matchTime` or a `@matchOffset` and `MatchSignal` element are in use and either the `MatchSignal` matches or the `@matchTime/@matchOffset` matches per the same semantics as noted in (a) and (b).
    d. neither the `@matchTime`, `@matchOffset` nor `MatchSignal` are present and the referenced `Policy` is considered to be "always" applied as long as it is present in the `Media`/`MediaPoint`.
2. When an updated (version) of the `Media` is received with differences that result in `Policy` being applied or removed. Such as:
    a. A new `Media` element containing a `MediaPoint` element in a previous version that is updated to no longer contain that `MediaPoint` element. The `Policy` associated with the `MediaPoint` ***shall*** be removed and any `Policy` that *should* be in effect as a result of evaluation at that point in time ***shall*** be in effect.
    b. A referenced `Policy` is added, deleted or updated
    c. An `Audience` or `ViewingPolicy` of a referenced `Policy` are added, deleted or updated
    d. Any `MediaPoint`, `Policy`, `ViewingPolicy`, or `Audience` that references global elements that are broken (reference a value that are not present) ***shall*** be considered an invalid `Policy` in whole and ***shall not*** be evaluated.
3. When a `Policy` has been actively applied with a `@duration` set and the duration has passed the referenced `Policy` ***shall*** be implicitly removed.

10.3. Precedence and Stacking

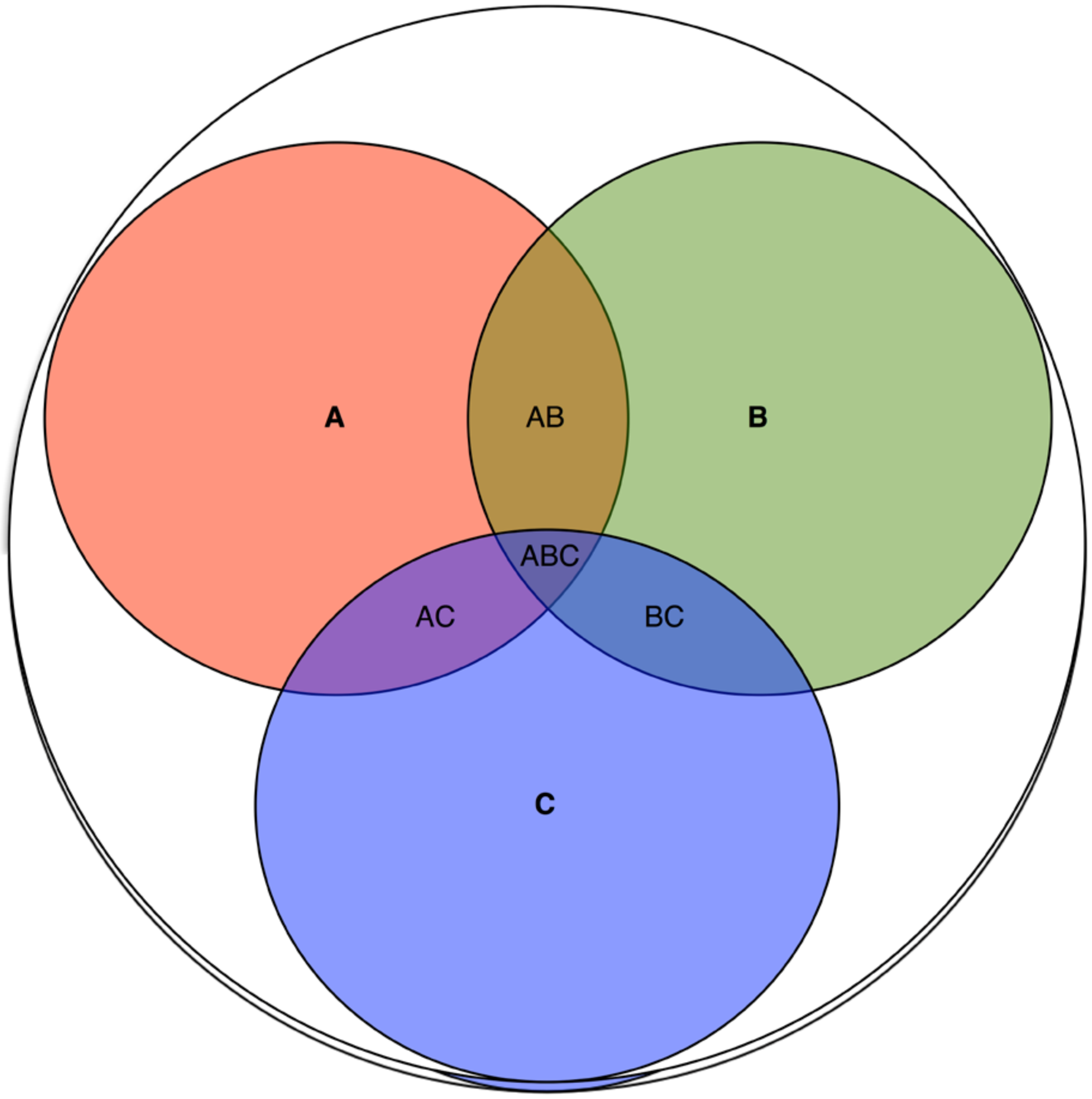


Figure 4 - Policy Precedence and Stacking

The order of **`MediaPoint`** elements is important as it sets the order of precedence for the application and removal of **`Policy`**. If provided, the @order attribute provides an explicit order of MediaPoints within the Media. Given that several policies *may* be active at any given point in time relative to the **`Media`** it is noted that the **`Policy`** application is both "additive" and "ordered" in a first-in-last-out order for **Apply** actions without `@priority`. When `@priority` is provided in an **Apply**, the order of precedence for the active policy set is based on `@priority` (with "0" being the highest") and then document order. When multiple active Policies conflict for a given Audience, the topmost Policy in the set *should* take precedence.

In Figure 4, the outer circle is the universe of our addressable Audience. The colored circles each are a specific addressable **`Audience`**. The labels (A, B, C) are the Actions applied to the **`Audience`** in the order of which they are applied.

For example, the intersecting "brown" Audience, with the AB (more properly A∩B) label would have both the A Action applied (or removed) AND the B Action applied (or removed) in that order thereby B having a higher precedence, in terms of the Action applied or removed, over A *should* the Actions conflict in some way.

The scope of a conflict is when two Actions cannot be applied where the effects of one Action are not completely exclusive or independent of another Action.

Example: A∩B

Precedence ***shall*** be determined by a first-in-last-out order for **Apply** actions without `@priority`. When `@priority` is provided in an **Apply**, the order of precedence for the active policy set is based on `@priority` (with "0" being the highest") and then the document order by which, Policy declared later in a document is said to have higher precedence. The **`Policy`** is applied or removed as indicated by the use of **`MediaPoint`** matching.

Example: A∩B∩C

Example: With priority. For a Media that contains the following MediaPoints in alphabetical order. MediaPoint V has **Apply** for Policy V with priority 5, MediaPoint W has **Apply** for Policy W with no priority. At the point where MediaPoints V and W have been triggered, Policy V is on top, with Policy W below it. Next, MediaPoint X has **Apply** for Policy X with priority 5. When MediaPoint X is triggered, Policy X is now on top, because it is tied, priority order wise, with Policy V, but it is later in the document order. Now, MediaPoint Y has **Apply** for **Policy** Y, with priority 5, and **Remove** for **Policy** V, so **Policy** Y is now the top policy.

Appendix A: Use cases and Examples

Use Case Examples (all examples, namespace declarations are omitted for readability):

1. An Audience for users in Boulder, CO. Note that the user might be in any of the listed zip codes:

```
<Audience id="/audience/co/boulder" match="ANY">
  <audience:Zip>80301</audience:Zip>
  <audience:Zip>80302</audience:Zip>
  <audience:Zip>80303</audience:Zip>
  <audience:Zip>80304</audience:Zip>
  <audience:Zip>80305</audience:Zip>
</Audience>
```

2. An Audience of authenticated users on a private network:

```
<Audience id="/audience/auth/private" match="ALL">
  <audience:Authenticate>true</audience:Authenticate>
  <audience:Network>PRIVATE</audience:Network>
</Audience>
```

3. An Audience of users with devices having digital output:

```
<Audience id="/audience/deviceGroup/23" match="ALL">
  <audience:Feature>DIGITAL_OUTPUT</audience:Feature>
</Audience>
```

4. An Audience of users in a Placement Opportunity:

```
<Audience id="/audience/PLACEMENT_OPPORTUNITY" match="ALL">
  <audience:Default>PLACEMENT_OPPORTUNITY</audience:Default>
</Audience>
```

5. An Audience of users with devices that are NOT running with Linux or Android:

```
<Audience id="/audience/deviceGroup/42" match="NONE">
  <audience:OS>LINUX</audience:OS>
  <audience:OS>ANDROID</audience:OS>
</Audience>
```

6. An Audience of users in the United States using devices with dedicated displays (such as a tablet) and experiencing technical difficulties:

```
<Audience id="/audience/tech_difficulty" match="ALL">
  <audience:ISO3166>US</audience:ISO3166>
  <audience:Default>TECHNICAL_DIFFICULTY</audience:Default>
  <audience:DeviceFeature>DEDICATED_DISPLAY</audience:DeviceFeature>
</Audience>
```

7. A ViewingPolicy to display a slate for an Audience experiencing technical difficulties:

```
<ViewingPolicy id="/viewingpolicy/1">
  <Audience xlink:href="/audience/tech_difficulty"/>
  <action:Content>slates/tdslate.mpg</action:Content>
</ViewingPolicy>
```

8. A ViewingPolicy to blackout an Audience Boulder:

```
<ViewingPolicy id="/viewingpolicy/2">
  <Audience xlink:href="/audience/co/boulder"/>
  <action:Content>urn:scte:224:action:blackout</action:Content>
</ViewingPolicy>
```

9. A ViewingPolicy to restrict an Audience to SD (resolution=480) only:

```
<ViewingPolicy id="/viewingpolicy/3">
  <Audience xlink:href="/audience/deviceGroup/23"/>
  <action:MaxResolution>480</action:MaxResolution>
</ViewingPolicy>
```

10. A ViewingPolicy to allow the distributor to perform dynamic ad insertion for an Audience:

```
<ViewingPolicy id="/viewingpolicy/4">
  <Audience xlink:href="/audience/co/boulder"/>
  <action:Dai>true</action:Dai>
</ViewingPolicy>
```

11. A Policy with multiple ViewingPolicy(s):

```
<Policy id="/policy/1">
  <ViewingPolicy xlink:href="/viewingpolicy/1"/>
  <ViewingPolicy xlink:href="/viewingpolicy/2"/>
  <ViewingPolicy xlink:href="/viewingpolicy/3"/>
</Policy>
```

12. Use Case: Show alternative content to users unless they are using a location-aware device and are within the boundaries of Denver county. Furthermore, re-check that the user is within the boundary at least once every 10 minutes.
    a. An Audience of users with location-aware devices in Denver county:

```
<Audience id="/audience/1" match="NONE">
  <Audience match="ALL">
    <audience:Feature>LOCATION_AWARE</audience:Feature>
    <audience:FIPS>08031</audience:FIPS>
  </Audience>
</Audience>
```

    b. A ViewingPolicy to show alternative content and revalidate every 10 minutes:

```
<ViewingPolicy id="/viewingpolicy/5">
  <Audience xlink:href="/audience/1"/>
  <action:Content>content/134322233</action:Content>
  <action:Revalidate>PT10M</action:Revalidate>
</ViewingPolicy>
```

13. Use Case: Show alternative football games instead of the national game of the week for local audiences. Note that only one of the ViewingPolicy(s) *may* be applied. Evaluate each ViewingPolicy in document order until the first match is found.
    c. An Audience of users in either Indiana or New Jersey:

```
<Audience id="/audience/njin" match="ANY">
  <Audience xlink:href="/audience/new_jersey"/>
  <Audience xlink:href="/audience/indianna"/>
</Audience>
```

    d. A ViewingPolicy to show the Indiana vs. Rutgers game to the Audience defined above:

```
<ViewingPolicy id="/vp/6">
  <Audience xlink:href="/audience/njin"/>
  <action:Content>Indiana_at_Rutgers</action:Content>
</ViewingPolicy>
```

    e. An Audience of users in either Michigan or Maryland:

```
<Audience id="/audience/mima" match="ANY">
  <Audience xlink:href="/audience/michigan"/>
  <Audience xlink:href="/audience/maryland"/>
</Audience>
```

    f. A ViewingPolicy to show the Michigan State vs. Maryland game to the Audience defined above:

```
<ViewingPolicy id="/vp/7">
  <Audience xlink:href="/audience/mima"/>
  <action:Content>MichiganState_at_Maryland</action:Content>
</ViewingPolicy>
```

    g. An Audience of users in either Nebraska or Wisconsin:

```
<Audience id="/audience/newi" match="ANY">
  <Audience xlink:href="/audience/nebraska"/>
  <Audience xlink:href="/audience/wisconsin"/>
</Audience>
```

    h. A ViewingPolicy to show the Nebraska vs. Wisconsin game to the Audience defined above:

```
<ViewingPolicy id="/vp/8">
  <Audience xlink:href="/audience/newa"/>
  <action:Content>Nebraska_at_Wisconsin</action:Content>
</ViewingPolicy>
```

    i. A Policy to evaluate each of the above defined ViewingPolicy(s) in document order:

```
<Policy id="/p/2">
  <ViewingPolicy xlink:href="/vp/6"/>
  <ViewingPolicy xlink:href="/vp/7"/>
  <ViewingPolicy xlink:href="/vp/8"/>
</Policy>
```

14. Use Case: Provide network default behavior for common, but unscheduled situations.
    a. An Audience of users experiencing technical difficulties:

```
<Audience id="/audience/tech_difficulty" match="ALL">
  <audience:Default>TECHNICAL_DIFFICULTY</audience:Default>
</Audience>
```

    b. A ViewingPolicy to show a slate stating something like, “we are experiencing technical difficulties” to the Audience defined above:

```
<ViewingPolicy id="/vp/tech_difficulty">
  <Audience xlink:href="/audience/tech_difficulty"/>
  <action:Content>TechDifficultiesSlate</action:Content>
</ViewingPolicy>
```

    c. An Audience of users tuned to a network, which is currently off air:

```
<Audience id="/audience/off_air" match="ALL">
  <audience:Default>OFF_AIR</audience:Default>
</Audience>
```

    d. A ViewingPolicy to show a slate stating something like, “this network is off air” to the Audience defined above:

```
<ViewingPolicy id="/vp/off_air">
  <Audience xlink:href="/audience/off_air"/>
  <action:Content>OffAirSlate</action:Content>
</ViewingPolicy>
```

    e. An Audience of users trying to watch a program, which is signaled as a web blackout in an SCTE 35 Segmentation Descriptor and no other content alternatives are provided:

```
<Audience id="/audience/web_blackout" match="ALL">
  <audience:Default>WEB_BLACKOUT</audience:Default>
</Audience>
```

    f. A ViewingPolicy to show a slate stating something like, “this program is blacked out” to the Audience defined above:

```
<ViewingPolicy id="/vp/web_blackout">
  <Audience xlink:href="/audience/web_blackout"/>
  <action:Content>WebBlackoutSlate</action:Content>
</ViewingPolicy>
```

    g. A Policy to evaluate each of the above defined ViewingPolicy(s) in document order:

```
<Policy id="/p/3">
  <ViewingPolicy xlink:href="/vp/tech_difficulty"/>
  <ViewingPolicy xlink:href="/vp/off_air"/>
  <ViewingPolicy xlink:href="/vp/web_blackout"/>
</Policy>
```

    h. A **`MediaPoint`** to indicate that a Policy *should* be applied as a network default. Notice that there is no `@mediaTime`, `@mediaOffset` or **`MatchSignal`**, indicating that this Policy *should* be applied any time the Audience conditions are matched for the Media:

```
<MediaPoint id="/mp/1" description="Network Defaults">
  <Apply><Policy xlink:href="/p/1"/></Apply>
</MediaPoint>
```

15. A MediaPoint to indicate the start of a program based on wall clock time:

```
<MediaPoint id="/mp/2" description="The Office" matchTime="2018-10-04T04:00:00Z">
  <AltID>http://dx.doi.org/10.5240/141D-AD7B-F7A3-80C7-ED4F-U</AltID>
</MediaPoint>
```

16. A MediaPoint to indicate the start of a chapter in a VOD asset:

```
<MediaPoint id="/mp/3" description="Chapter 3: Fallen" matchOffset="PT31M5S"/>
```

17. A MediaPoint based on an in-band signal. Since there is no @matchTime or @matchOffset, the Policy can only be applied if the signal is found in-band. Furthermore, the signal *may*

occur at any time within the effective window of the MediaPoint. The Policy ***shall*** only be applied if a matching signal is found – otherwise, this MediaPoint ***shall*** have no effect.

```
<MediaPoint id="/mp/4"
  effective="2018-10-01T14:00:00Z" expires="2018-10-01T15:00:00Z">
  <Apply><Policy xlink:href="p/1"/></Apply>
  <MatchSignal>
    <Assert>//SegmentationUpid[@segmentationUpidType='8']="20997C44"</Assert>
  </MatchSignal>
</MediaPoint>
```

18. A MediaPoint based on an in-band signal with a fallback to wall clock time if the signal is not found. The nominal time of this MediaPoint is indicated by the @matchTime. In this example, the @effective attribute allows the signal to arrive up to 5 minutes early. The @signalTolerance attribute specifies that if the signal does not arrive by 3 minutes after the @matchTime then the associated Policy ***shall*** be applied at that time:

```
<MediaPoint id="/mp/5" matchTime="2018-10-01T14:00:00Z"
  effective="2018-10-01T13:55:00Z" expires="2018-10-01T14:03:00Z">
  <Apply><Policy xlink:href="p/1"/></Apply>
  <MatchSignal signalTolerance="PT3M">
    <Assert>//SegmentationUpid[@segmentationUpidType='8']="20997C44"</Assert>
  </MatchSignal>
</MediaPoint>
```

19. A MediaPoint, which indicates a Policy ***shall*** be applied and then subsequently removed in exactly 1 hour:

```
<MediaPoint id="/mp/11" matchTime="2018-10-01T14:00:00Z"
  effective="2018-10-01T13:55:00Z" expires="2018-10-01T14:05:00Z">
  <Apply duration="PT1H"><Policy xlink:href="/p/1"/></Apply>
  <MatchSignal signalTolerance="PT5M">
    <Assert>//SegmentationUpid[@segmentationUpidType='8']="20997C44"</Assert>
  </MatchSignal>
</MediaPoint>
```

20. A MediaPoint, which indicates a Policy ***shall*** be removed based on a matching in-band signal. When this MediaPoint is combined with the previous example, it serves to allow for an optional, early termination of the Policy. The early termination will only occur if a matching in-band signal is found – otherwise, the duration in the previous example will cause the Policy to be removed.

```
<MediaPoint id="/mp/12"
  effective="2018-10-01T13:55:00Z" expires="2018-10-01T15:05:00Z">
  <Remove><Policy xlink:href="/p/1"/></Remove>
  <MatchSignal signalTolerance="PT5M">
    <Assert>//SegmentationDescriptor/@segmentationTypeId="18"</Assert>
  </MatchSignal>
</MediaPoint>
```

21. The following represents a simple network feed all together.

```
<Media id="/media/tbs" description="TBS" lastUpdated="2018-02-10T12:00:00Z">
  <!-- Network / Super -->
  <MediaPoint id="/network/1" description="TBS">
    <AltID>http://dx.doi.org/10.5239/C370-DCA5</AltID>
    <Apply><Policy xlink:href="/policy/6"/></Apply>
  </MediaPoint>
  <!-- Program Begins -->
  <MediaPoint id="/program/20997C44" description="World Cup Curling"
    matchTime="2014-11-05T12:00:00Z">
    <AltID>http://dx.doi.org/10.5240/9EF1-2DA2-5C1F-98B4-F784-E</AltID>
    <Apply duration="PT2H"><Policy xlink:href="/policy/5"/></Apply>
    <MatchSignal signalTolerance="PT12S" match="ALL">
      <Assert>//SegmentationUpid[@segmentationUpidType='8']="20997C44"</Assert>
    </MatchSignal>
  </MediaPoint>
  <!-- First Ad Break -->
  <MediaPoint id="/program/20997C44/pod/1" description="Ad Break #1"
    effective="2014-11-05T12:00:00Z" expires="2014-11-05T12:30:00Z">
    <Apply duration="PT90S"><Policy xlink:href="/policy/7"/></Apply>
    <MatchSignal match="ALL">
```

```
      <Assert>//SpliceInsert</Assert>
    </MatchSignal>
  </MediaPoint>
  <!-- Second Ad Break -->
  <MediaPoint id="/program/20997C44/pod/2" description="Ad Break #2"
    effective="2014-11-05T12:30:00Z" expires="2014-11-05T01:15:00Z">
    <Apply duration="PT120S"><Policy xlink:href="/policy/8"/></Apply>
    <MatchSignal match="ALL">
      <Assert>//SpliceInsert</Assert>
    </MatchSignal>
  </MediaPoint>
  <!-- Third Ad Break -->
  <MediaPoint id="/program/20997C44/pod/3" description="Ad Break #3"
    effective="2014-11-05T01:15:00Z" expires="2014-11-05T02:00:00Z">
    <Apply duration="PT60S"><Policy xlink:href="/policy/9"/></Apply>
    <MatchSignal match="ALL">
      <Assert>/SpliceInsert/Program</Assert>
    </MatchSignal>
  </MediaPoint>
  <!-- Potential Program Breakaway (may never happen) -->
  <MediaPoint id="/program/20997C44/breakaway" description="Breakaway"
    effective="2014-11-05T12:00:00Z" expires="2014-11-05T02:00:00Z">
    <Remove><Policy xlink:href="/policy/5"/></Remove>
    <MatchSignal match="ALL">
      <Assert>/SegmentationDescriptor[@segmentationTypeId='19']</Assert>
    </MatchSignal>
  </MediaPoint>
  <!-- Next Program Begins (has no associated policy) -->
  <MediaPoint id="/program/20997C45" description="News"
    matchTime="2014-11-05T02:00:00Z">
    <Remove><Policy xlink:href="/policy/5"/></Remove>
    <MatchSignal signalTolerance="PT12S" match="ALL">
      <Assert>//SegmentationUpid[@segmentationUpidType='8']="20997C45"</Assert>
    </MatchSignal>
  </MediaPoint>
</Media>
```

22. The Results of an Audit of the first MediaPoint from example 21:

```
<Results size="1">
  <Audit id="/audit/1"
    xlink:role="Media" xlink:href="/media/tbs" trigger="NONE">
    <Audit id="/audit/2" lastUpdated="2014-11-05T12:00:00Z"
      xlink:role="MediaPoint" xlink:href="/media/tbs/program/20997C44" trigger="SIGNAL">
      <Audit id="/audit/3" lastUpdated="2014-11-05T12:00:00Z" result="SUCCESS"
        xlink:role="Policy" xlink:href="/policy/5" trigger="SIGNAL" policyMode="APPLY">
        <Audit id="/audit/4" lastUpdated="2014-11-05T12:00:00Z" result="SUCCESS"
          xlink:role="ViewingPolicy" xlink:href="/vp/1" trigger="SIGNAL" policyMode="APPLY"/>
      </Audit>
    </Audit>
  </Audit>
</Results>
```

23. Based on **`Audience`** in example 1 and **`ViewingPolicy`** in example 8, the following is an example Result of an **`Audit`** query request on **`ViewingPolicy`**
    a. A status request for any ViewingPolicy being applied at 2018-04-01 2:00AM UTC, referencing an Audience with @id="audience/co/boulder":

```
GET https://esni.somecompany.com
/audit?role=ViewingPolicy&audience=%2Faudience%2Fco%2Fboulder&status=2018-04-01T02%3A00%3A00Z
```

    b. This returns the following results:

```
<Results size="1">
  <Audit id="/audit/1"
    xlink:role="Media" xlink:href="/media/tbs" trigger="STATUS">
    <Audit id="/audit/2" lastUpdated="2018-04-01T02:00:00+00:00"
      xlink:role="MediaPoint" xlink:href="/media/tbs/mp1" trigger="STATUS">
      <Audit id="/audit/3" lastUpdated="2018-04-01T02:00:00+00:00" result="SUCCESS"
        xlink:role="Policy" xlink:href="/policy/1" trigger="STATUS">
        <Audit id="/audit/4" lastUpdated="2018-04-01T02:00:00+00:00" result="SUCCESS"
          xlink:role="ViewingPolicy" xlink:href="/viewingpolicy/2" trigger="STATUS"/>
      </Audit>
    </Audit>
  </Audit>
</Results>
```

24. Based on **`Media`** in example 21, the following is an example Result of an **`Audit`** query request for **`Policy`**.
    a. A status request for any Policy being applied at 2014-11-06 1:00AM UTC on a Media with @id="/media/tbs"

```
GET https://esni.somecompany.com/audit?role=Policy&status=2014-11-
06T01%3A00%3A00Z&media=%2Fmedia%2Ftbs
```

    b. The following results indicates that one ViewingPolicy has failed:

```
<Results size="1">
  <Audit id="/audit/1"
    xlink:role="Media" xlink:href="/media/tbs" trigger="STATUS">
    <Audit id="/audit/2" lastUpdated="2018-04-01T02:00:00+00:00"
      xlink:role="MediaPoint" xlink:href="/media/tbs/mp1" trigger="STATUS">
      <Audit id="/audit/3" lastUpdated="2018-04-01T02:00:00+00:00" result="FAIL"
        xlink:role="Policy" xlink:href="/policy/7" trigger="STATUS"
        description="See ViewingPolicy status">
        <Audit id="/audit/4" lastUpdated="2018-04-01T02:00:00+00:00" result="SUCCESS"
          xlink:role="ViewingPolicy" xlink:href="/viewingpolicy/1" trigger="STATUS"/>
        <Audit id="/audit/5" lastUpdated="2018-04-01T02:00:00+00:00" result="FAIL"
          xlink:role="ViewingPolicy" xlink:href="/viewingpolicy/2" trigger="STATUS"
          description="This ViewingPolicy was not executed due to some reason"/>
      </Audit>
    </Audit>
  </Audit>
</Results>
```

25. Based on **`Media`** in example 16, the following is an example Result of an Decision query request.
    a. A request for a decision 2018-10-01 2:05PM UTC, with an audience of

```
GET
https://esni.somecompany.com/decide?source=xyz&audiencetype=zip&audiencevalue=80820&eventtime=
2018-10-01T014%3A05%3A00Z&segmentationupidtypeid=8&signalid=20997C44
```

    b. The following results indicates that one ViewingPolicy has failed:

```
<Results size="1">
  <Audit id="/audit/1"
    xlink:role="Media" xlink:href="/media/tbs" trigger="STATUS">
    <Audit id="/audit/2" lastUpdated="2018-04-01T02:00:00+00:00"
      xlink:role="MediaPoint" xlink:href="/media/tbs/mp1" trigger="STATUS">
      <Audit id="/audit/3" lastUpdated="2018-04-01T02:00:00+00:00" result="FAIL"
        xlink:role="Policy" xlink:href="/policy/7" trigger="STATUS"
        description="See ViewingPolicy status">
        <Audit id="/audit/4" lastUpdated="2018-04-01T02:00:00+00:00" result="SUCCESS"
          xlink:role="ViewingPolicy" xlink:href="/viewingpolicy/1" trigger="STATUS"
          description="We have a match so action would be action:Content http://slate"/>
        <Audit id="/audit/5" lastUpdated="2018-04-01T02:00:00+00:00" result="FAIL"
          xlink:role="Audience" xlink:href="/audience/2" trigger="STATUS"/>
      </Audit>
    </Audit>
  </Audit>
</Results>
```

26. A ViewingPolicy, which indicates no midroll DAI replacement based on the C3 window for a viewer in a Placement Opportunity. This could be used to protect C3 content replacement.

```
<ViewingPolicy id="/vp/25_C3" description="DAI for C3">
  <Audience xlink:href="/audience/placement_opportunity"/>
  <Capture>
    <StartWindow>
      <Offset>0</Offset>
    </StartWindow>
    <StopWindow>
      <Offset>PT3D3H</Offset>
    </StopWindow>
    <MidrollDAI>false</MidrollDAI>
  </Capture>
</ViewingPolicy>
```

27. A ViewingPolicy, which indicates no midroll DAI replacement based on the C7 window for a viewer in a Placement Opportunity. This could be used to protect C7 content replacement

```
<ViewingPolicy id="/vp/25_C7" description="DAI for C7">
  <Audience xlink:href="/audience/placement_opportunity"/>
  <Capture>
    <StartWindow>
      <Offset>0</Offset>
    </StartWindow>
    <StopWindow>
      <Offset>PT7D1H</Offset>
    </StopWindow>
    <MidrollDAI>false</MidrollDAI>
  </Capture>
</ViewingPolicy>
```

28. A ViewingPolicy, which indicates ad exclusion for a viewer in a Placement Opportunity.

```
<ViewingPolicy id="/vp/ad_exclusion" description="Ad exclusion">
  <Audience xlink:href="/audience/placement_opportunity"/>
    <AdExclusion>true</AdExclusion>
</ViewingPolicy>
```

29. A Policy, which indicates ad exclusion ViewingPolicy.

```
<Policy id="/vp/ad_exclusion" description="Ad exclusion">
  <ViewingPolicy xlink:href="/vp/ad_exclusion"/>
</Policy>
```

30. A MediaPoint, which indicates an approach to convey break metadata for exclusion, using the urn:scte:224:metadata namespace and the NetworkSpotProduct.

```
<MediaPoint id="providerXYZ.com/media/break/1233/start"
   description="Avail Start">
   effective="2016-07-05T14:59:50.0Z"
   expires="2016-07-05T15:10:00.0Z"
   matchTime="2016-07-05T15:00:00.0Z"
   expectedDuration="PT60S">
   <Metadata>
      <metadata:Slot ownerType="PROVIDER" ownerName="XYZ" DAIModel="SINGLE_ADVERTISER"
position=1 duration="PT30S">
      </metadata:Slot>
      <metadata:Slot ownerType="DISTRIBUTOR" ownerName="ABCD" DAIModel="SINGLE_ADVERTISER"
position=2 duration="PT30S">
         <metadata:NetworkSpotProduct productCategory="SUV" productCode="T111"/>
      </metadata:Slot>
   </Metadata>
   <Apply duration="PT60S">
     <Policy xlink:href="/policy/ad_exclusion"/>
   </Apply>
   <MatchSignal signalTolerance="PT5M">
      <Assert>
         /SpliceInfoSection/SegmentationDescriptor[@segmentationTypeId=35]/SegmentationUpid[
@segmenationUpidType=14 text()=`CNPA0484000H`]
      </Assert>
   </MatchSignal>
</MediaPoint>
```

31. A MediaPoint, which indicates no midroll DAI replacement based on the C3 window for a viewer in a Placement Opportunity. This MediaPoint also shows an approach to convey break metadata, using **AltID** element.

```
<MediaPoint id="providerXYZ.com/media/break/1234/start"
   description="Avail Start">
   effective="2016-07-05T14:59:50.0Z"
   expires="2016-07-05T15:10:00.0Z"
   matchTime="2016-07-05T15:00:00.0Z"
   expectedDuration="PT30S">
   <AltID description="Ad-ID">CNPA0484000H</AltID>
     <Metadata>
      <metadata:Slot ownerType="PROVIDER" ownerName="XYZ" DAIModel="SINGLE_ADVERTISER"
duration="PT30S" offset="PT0S">
      </metadata:Slot>
      <metadata:Slot ownerType="DISTRIBUTOR" ownerName="ABCD" DAIModel="SINGLE_ADVERTISER"
duration="PT30S" offset="PT30S">
      </metadata:Slot>
   </Metadata>
   <Apply duration=“PT30S”>
     <Policy xlink:href="/policy/NoDAI_C3"/>
   </Apply>
</MediaPoint>
```

Appendix B: Request signing

This appendix details the process to sign a request. The signature is used to authorize requests submitted to all interfaces.

The process documented herein specifies how to generate the signature value which ***shall*** be added to the request with the Authorization header. Adding the Authorization header to the HTTP request is referred to as signing the request.

There are four tasks to be completed to create a signed request. Each task is a multi-step process, with the result of each task being fed into the subsequent task.

Signing Tasks

1. Create a Canonical Request
    a. Arrange the request contents into a standard (canonical) format.
    b. Create a hash (digest) of the canonical request and add the hash to the canonical request.
    c. Create a digest of the updated canonical request.
2. Create a String to Sign
    a. The string to sign is composed of specific elements of the request including the algorithm (e.g. SHA256), date, credential and digest (hash) of the canonical request (from Task 1).
3. Create a Signature
    a. Derive a signing key by performing a succession of recursive keyed hash operations (HMAC) on the request date, service and signing value using a supplied secret as the key for the hashing operation.
    b. Calculate a signature using the derived signing key as the hash key
4. Add Signing Information to the Request
    a. Add the calculated signature to the request using the Authorization header

Task 1: Create a Canonical Request For Signature

To begin the signing process, you create a string that includes information from your request in a standardized (canonical) format. Formatting the request into an unambiguous, canonical format before signing it ensures that when implementation receives the request, it can calculate the same signature that you calculated. You must the follow the steps here to create a canonical version of the request; otherwise, your version and the version calculated by the implementer won't match, and the request denied.

The following example shows the pseudocode for how to create a canonical request.

Canonical request pseudocode

```
CanonicalRequest =
   HTTPRequestMethod + '\n' +
   CanonicalURI + '\n' +
   CanonicalQueryString + '\n' +
   CanonicalHeaders + '\n' +
   SignedHeaders + '\n' +
   HexEncode(Hash(RequestPayload))
```

In this pseudocode, Hash represents a function that produces a message digest, typically SHA-256. (Later in the process you specify which hashing algorithm you're using.) HexEncode represents a function that returns the base-16 encoding of the digest in lowercase characters. For example, HexEncode("m") returns the value 6d rather than 6D. Each input byte must be represented as exactly two hexadecimal characters.

To show you how to construct the canonical form of an HTTP request, we'll use the following sample request. This is what a request might look like as it is sent from the provider to a distributor, except this example does not include the signing information yet.

Sample request

```
PUT /policy/1 HTTP/1.1
Host: esni.somecompany.com
Content-Type: application/xml
Date: Mon, 16 Mar 2018 22:32:15 GMT

<?xml version="1.0" encoding="UTF-8"?>
<Policy id="/policy/1" xmlns="http://www.scte.org/schemas/224"
  xmlns:xlink="http://www.w3.org/1999/xlink">
  <ViewingPolicy xlink:href="/viewingpolicy/1"/>
  <ViewingPolicy xlink:href="/viewingpolicy/2"/>
</Policy>
```

The sample request is a PUT request (method) that sends an **`Policy`** message to the service. This action takes a single message payload – the reusable **`Policy`**.

To create a canonical request, concatenate the following components from each step into a single string:

1. Start with the HTTP request method (GET, PUT, etc.), followed by a newline character.

Sample request: request method

```
PUT
```

2. Add the canonical URI parameter, followed by a newline character. The canonical URI is the URI-encoded version of the absolute path component of the URI—everything in the URI from the HTTP host to the question mark character ("?") that begins the query string parameters (if any). Then add a newline character.

Normalize URI paths according to RFC 3986 by removing redundant and relative path components. Each path segment must be URI-encoded. If the absolute path is empty, use a forward slash (/). In the example request, “audience” follows the host in the URI, so the absolute path is shown in the following example.

Sample request: canonical URI

```
/policy/1
```

3. Add the canonical query string, followed by a newline character. If the request does not include a query string, set this value in the canonical query to an empty string (essentially, a blank line). The example query does not contain a query string.

Sample request: canonical query string (NOTE: intentionally blank given the PUT example)

```

```

4. Add the canonical headers, followed by a newline character. The canonical headers consist of a list of all the HTTP headers that you are intend to be validated as part of the request. At a minimum the client ***shall*** include host, content-type and date.

Sample request: canonical headers

```
content-type:application/xml
date:Mon, 16 Mar 2018 22:32:15 GMT
host:esni.somecompany.com
```

To create the canonical headers list, convert all header names to lowercase and trim excess white space characters out of the header values. When you trim, remove leading spaces and trailing spaces, and convert sequential spaces in the value to a single space. However, do not remove extra spaces from any values that are inside quotation marks.

The following pseudocode describes how to construct the canonical list of headers:

CanonicalHeadersEntry = Lowercase(HeaderName) + ':' + Trimall(HeaderValue)

CanonicalHeaders = CanonicalHeadersEntry0 + CanonicalHeadersEntry1 + ... + CanonicalHeadersEntryN

Lowercase represents a function that converts all characters to lowercase. The Trimall function removes excess white space before and after values and from inside non-quoted strings, per RFC 7230 Section 3.2.3.

Build the canonical headers list by iterating through the collection of header names and sorting the headers lowercase character code. Construct each header by using the following rules:

- Append the lowercase header name followed by a colon.
- Append a comma-separated list of values for that header. If there are duplicate headers, the values are comma-separated. Do not sort the values in headers that have multiple values.
- Append a new line ('\n').

Note

Each individual header is followed by a newline character, meaning that the complete list ends with a newline character.

5. Add the signed headers, followed by a newline. This value is a list of the names of the headers that you included in the canonical headers. By adding this list of headers, you tell recipient which headers in the request are part of the signing process. By including this value, you indicate that the implementers service can ignore any additional headers (for example, any headers added by a proxy) for purposes of validating the request.

The host, content-type, and date headers must be included as signed headers. They must be included in the list of signed headers.

Sample request: signed headers

```
content-type;date;host
```

To create the signed headers list, convert all header names to lowercase, sort them by character code, and use a semicolon to separate the header names. The following pseudocode describes how to construct a list of signed headers. LOWERCASE represents a function that converts all characters to lowercase.

SignedHeaders = Lowercase(HeaderName0) + ';' + Lowercase(HeaderName1) + ";" + ... + Lowercase(HeaderNameN)
Build the signed headers list by iterating through the collection of header names, sorted by lowercase character code. For each header name except the last, append a semicolon (';') to the header name to separate it from the following header name.

6. Using a SHA256 hash (digest) function, create a hashed value from the payload in the body of the HTTP request:

Structure of payload

```
HashedPayload = Lowercase(HexEncode(Hash(requestPayload)))
```

If the payload is empty, use the empty string as the input to the hash function. For example, suppose that your request includes the following payload:

Sample request: payload

```
PUT /policy/1 HTTP/1.1
Host: esni.somecompany.com
Content-Type: application/xml
Date: Mon, 16 Mar 2018 22:32:15 GMT

<?xml version="1.0" encoding="UTF-8"?>
<Policy id="/policy/1" xmlns="http://www.scte.org/schemas/224"
  xmlns:xlink="http://www.w3.org/1999/xlink">
  <ViewingPolicy xlink:href="/viewingpolicy/1"/>
  <ViewingPolicy xlink:href="/viewingpolicy/2"/>
</Policy>
```

The following example shows the result of using HMAC SHA-256 hash of the example payload. (When you create the string to sign (*Task 2: Create a String to Sign*), you will specify the signing algorithm that you used to hash the payload. This standard will be using HMAC and SHA256, therefore you will specify HMAC-SHA256 as the signing algorithm.) The hashed payload must be represented as a lowercase hexadecimal string.

Sample request: hashed payload

```
2a7857979b2ecf99c79f23a36015eebd590b9b9178ee137f773be7191e745887
```

7. To construct the finished canonical request, combine all the components from each step as a single string. As noted, each component ends with a newline character. If you follow the canonical request pseudocode explained earlier, the resulting canonical request is shown in the following example:

Sample request: Canonical form

```
PUT
/policy/1

content-type:application/xml
date:Mon, 16 Mar 2018 22:32:15 GMT
host:esni.somecompany.com
content-type;date;host
2a7857979b2ecf99c79f23a36015eebd590b9b9178ee137f773be7191e745887
```

8. Create a digest (hash) of the canonical request by using the same algorithm that you used to hash the payload. The hashed canonical request must be represented as a string of lowercase hexadecimal characters. The following sample example shows the result of using HMAC SHA-256 to hash the example canonical request.

Sample request: Hashed canonical request

```
c960e4c7c1c1d72d7191408834b690f520b0b5ffdf72a7950ec8a7de313abda0
```

You include the hashed canonical request as part of the string to sign in *Task 2: Create a String to Sign*.

Task 2: Create a String to Sign

The *string to sign* includes meta information about your request and about the canonical request that you created in *Task 1: Create Canonical Request*. You will use the string to sign and a derived key that you create later as inputs when you calculate the request signature (*Task 3: Create a Signature*).

To create the string to sign, concatenate the algorithm, date, credential scope, and the digest of the canonical request, as shown in the following pseudocode. NOTE: The HashedCanonicalRequest can be and *should* be an empty string for GET and DELETE requests.

Structure of string to sign

```
StringToSign =
  Algorithm + '\n' +
  RequestDate + '\n' +
  CredentialScope + '\n' +
  HashedCanonicalRequest))
```

As an example, let's construct the string to sign by using the same sample request from Task 1: Create A Canonical Request:

Sample HTTPS request

```
PUT /policy/1 HTTP/1.1
Host: esni.somecompany.com
Content-Type: application/xml
Date: Mon, 16 Mar 2018 22:32:15 GMT

<?xml version="1.0" encoding="UTF-8"?>
<Policy id="/policy/1" xmlns="http://www.scte.org/schemas/224"
  xmlns:xlink="http://www.w3.org/1999/xlink">
  <ViewingPolicy xlink:href="/viewingpolicy/1"/>
  <ViewingPolicy xlink:href="/viewingpolicy/2"/>
</Policy>
```

To create the string to sign

1. Start with the algorithm designation, followed by a newline character. This value is the hashing algorithm that you're using to calculate the digests in the canonical request. (For SHA256, HMAC-SHA256 is the algorithm.)

```
HMAC-SHA256
```

2. Append the request date value, followed by a newline character.
   The value must conform to the “preferred” Date/Time formats as specified in RFC 7231 and must include a time zone designator. This value must match the value used in any previous steps. For example:

```
Mon, 16 Mar 2018 22:32:15 GMT
```

3. Append the credential scope value, followed by a newline character.
   This value is simply the termination string ("esni") in lowercase characters.

```
esni
```

4. Append the hash of the canonical request that you created in task 1. This value is not followed by a newline character. The hashed canonical request must be lowercase base-16 encoded, as defined by Section 8 of RFC 4648.

```
c960e4c7c1c1d72d7191408834b690f520b0b5ffdf72a7950ec8a7de313abda0
```

The following string to sign is a request to PUT **`Policy`** element on March 16, 2018.

Sample string to sign

```
HMAC-SHA256
Mon, 16 Mar 2018 22:32:15 GMT
esni
c960e4c7c1c1d72d7191408834b690f520b0b5ffdf72a7950ec8a7de313abda0
```

Task 3: Calculate the Signature

Before you calculate a signature, you derive a signing key from your provided secret key. (Don't just use your secret key to sign the request.)

You then use the *derived signing key* and the *string to sign* that you created in ***Task 2: Create a String to Sign*** as the inputs to a keyed hash function. The hex-encoded result from the keyed hash function is the signature.

To calculate a signature

1. Derive your signing key. To do this, you use your secret access key to create a series of hash-based message authentication codes (HMACs) as shown by the following pseudocode, where HMAC(key, data) represents an HMAC-SHA256 function that returns output in binary format. The result of each hash function becomes input for the next one.

Pseudocode for deriving a signing key

```
kSecret = Your Secret Key
kService = HMAC(kSecret, "esni")
```

Make sure that you specify the HMAC parameters in the correct order for the programming language that you are using. This example shows the key as the first parameter and the data as the second parameter, but the function you are using might specify the key and data in a different order.

Use the digest for the key derivation. Most languages have functions to compute either a binary format hash, commonly called a digest, or a hex-encoded hash, called a hexdigest. The key derivation requires that you use a digest.

As an example, the follow two example show the inputs to deriving a signing key and the resulting output, where kSecret = superSecretKey!.

The following example uses the same parameters from the sample request in Task 1 and Task 2 (a request to PUT **`Policy`** element on March 16, 2018).

Sample inputs

```
HMAC("superSecretKey!","esni")
```

2. Calculate the signature. To do this, use the signing key that you derived and the string to sign as inputs to the keyed hash function. After you've calculated the signature as a digest, convert the binary value to a hexadecimal representation.

The following pseudocode shows how to calculate the signature.

```
signature = HexEncode(HMAC(derived-signing-key, string-to-sign))
```

The following example shows the resulting signature if you use the sample signing key and the sample string to sign from Task 2:

Sample signature

```
626d60b90ec50654aa8ced29e066febbcd70664648bedefba54b3f025e33c339
```

Task 4: Add the Signing Information to the Request

After you calculate the signature, you add it to the request. The signature information must be added to a request in the following way:

- In a header named Authorization. (Although the header is named Authorization, the signing information is actually used for authentication—establishing who the request came from.)

Adding Signing Information to the Authorization Header

Include signing information by adding it to header named Authorization. The contents of the header are created after you have calculated the signature as described in the preceding steps, so the Authorization header is not included in the list of signed headers.

The following pseudocode shows the construction of the Authorization header.

Authorization: algorithm Credential=access key ID/credential scope, SignedHeaders=SignedHeaders, Signature=signature

The following example shows a finished Authorization header (Note the example value is a single line, but is folded for readability):

```
Authorization: HMAC-SHA256 Credential=testAccessKeyId1/esni,SignedHeaders=content-
type;date;host,Signature=626d60b90ec50654aa8ced29e066febbcd70664648bedefba54b3f025e33c339
```

Take note of the following:

- There is a space between the algorithm and Credential, not a comma. On the other hand, the SignedHeaders and Signature values are separated from the earlier values using a comma.
- The Credential value starts with the client identifier, which is prefixed with a slash (/) onto the credential scope value that you calculated earlier. In contrast, the secret key is used to derive the signing key for the signature, but is not included in the signing information that's sent in the request.

Appendix C: User's Guide Recommendations

This ESNI Standard defines the details and format for implementing ESNI, but providers have found that for a distributor to fully utilize the providers implementation and execute the instructions properly, Informative User's Guides should be created. Each providers user's guide will be slightly different based on their implementation, but following sections identify the minimum items that should be detailed in a provider's User's Guide.

C.1 Audience Messages

C.1.1 Identifiers

This ESNI standard says that the @id element of each message can be any text based Uniform Resource Identifier (URI). Using that guidance, providers should further identify the structure of their identifiers, for audiences, such as the following: {provider}/audience/{callsign}/name as an example of an audience id.

C.4.1 Metadata

This ESNI standard allows for Metadata to use urn:scte:224:metadata or any proprietary namespace to convey information about an Audience. A provider needs to describe the namespace it will use if at all. Additionally, it will need to define the elements of the namespace used and what the consumer is to do with the metadata.

C.1.2 Audience Types

This ESNI audience message allows a large number of audience types based on the audience schema. As a provider, the User's Guide should detail the audiences used in your implementation and any unique requirements for using your audience definitions.

C.1.3 Audience Uses

This ESNI audience has multiple settings for the @match attribute for audiences. A provider should describe their audience concepts, for example, whether their audiences are whitelist, blacklist, or both.

C.2 ViewingPolicy Messages

C.2.1 Identifiers

This ESNI standard says that the @id element of each message can be any text based Uniform Resource Identifier (URI). Using that guidance, providers should further identify the structure of their identifiers, for ViewingPolicy, such as the following: {provider}/viewingpolicy/<Callsign}/name as an example of a ViewingPolicy id.

C.4.1 Metadata

This ESNI standard allows for Metadata to use urn:scte:224:metadata or any proprietary namespace to convey information about a ViewingPolicy. A provider needs to describe the namespace it will use if at all. Additionally, it will need to define the elements of the namespace used and what the consumer is to do with the metadata.

C.2.2 ViewingPolicy Action Types

This ESNI ViewingPolicy message allows a large number of actions based on the audience schema. As a provider, the User's Guide should detail the actions used in your implementation and any unique requirements for using your action definitions.

C.3. Policy Messages

C.3.1 Identifiers

This ESNI standard says that the @id element of each message can be any text based Uniform Resource Identifier (URI). Using that guidance, providers should further identify the structure of their identifiers, for Policy, such as the following: {provider}/policy/{callsign}/name as an example of a Policy id.

C.4.1 Metadata

This ESNI standard allows for Metadata to use urn:scte:224:metadata or any proprietary namespace to convey information about a Policy. A provider needs to describe the namespace it will use if at all. Additionally, it will need to define the elements of the namespace used and what the consumer is to do with the metadata.

C.3.1 Priority

This ESNI standard provides a `@priority` attribute (in the **Apply**) when a Policy is applied. If a provider is going to utilize the `@priority`, they need to detail how the priorities are implemented at a minimum.

C.4 Media and MediaPoint Messages

C.4.1 MediaPoint Types

This ESNI standard describes general MediaPoints and "resident" MediaPoints. A provider should detail the use of "resident" MediaPoints relative to their implementation.

C.4.2 Identifiers

This ESNI standard says that the @id element of each message can be any text based Uniform Resource Identifier (URI). Using that guidance, providers should further identify the structure of their identifiers, for Policy, such as the following: {provider}/policy/{callsign}/name as an example of a Policy id.

C.4.2 AltIDs

This ESNI standard says that the AltID element can be any text value. The AltID element does provide a @description attribute to clarify what the alternate id is, but since the descriptions are only freetext as well, the provider needs to describe what AltIDs an operator can expect and what the description of that identifier will be.

C.4.1 Metadata

This ESNI standard allows for Metadata to use urn:scte:224:metadata or any proprietary namespace to convey information about a Media or MediaPoint. A provider needs to describe the namespace it will use if at all. Additionally, it will need to define the elements of the namespace used and what the consumer is to do with the metadata.

C.4.3 MatchSignal

This ESNI standard leaves the <MatchSignal><Assert> up to the provider, so the provider should provide examples and explanation of the XPath logic used in their <assert> elements, so the distributor can properly handle the XPath matching required to match MediaPoints and SCTE 25 signals.

C.5 Message Delivery

The ESNI standard details the HTTP Restful implementation of ESNI, but a provider must detail their message timing, Media length, and signing requirements.

C.5.1 Message Timing

Each provider has a different timing for their protocol, that impacts storage, management, and execution of ESNI on the distributor side. The provider's User's Guide should address questions like, How often is the message set refreshed? How are updates sent? Are messages sent in order?, etc.

C.5.2 Media Size

Each provider has a different length for their protocol, that impacts storage, management, and execution of ESNI on the distributor side. The provider's User's Guide should address questions like, How many days are covered by a Media? Do you create a new Media for each day?

C.5.3 Signing Requirements

This ESNI standard details message signing in Appendix B, but if there are additional requirements like https or specific ways keys are exchanged for the ESNI signing, that can be detailed as well.