OpenCable™ Specifications
Alternate Content

Real-time Event Signaling and Management API

OC-SP-ESAM-API-I03-131025

ISSUED

DISCLAIMER

This document is published by Cable Television Laboratories, Inc. ("CableLabs®").

CableLabs reserves the right to revise this document for any reason including, but not limited to, changes in laws, regulations, or standards promulgated by various agencies; technological advances; or changes in equipment design, manufacturing techniques, or operating procedures described, or referred to, herein. CableLabs makes no representation or warranty, express or implied, with respect to the completeness, accuracy, or utility of the document or any information or opinion contained in the report. Any use or reliance on the information or opinion is at the risk of the user, and CableLabs shall not be liable for any damage or injury incurred by any person arising out of the completeness, accuracy, or utility of any information or opinion contained in the document.

This document is not to be construed to suggest that any affiliated company modify or change any of its products or procedures, nor does this document represent a commitment by CableLabs or any cable member to purchase any product whether or not it meets the described characteristics. Nothing contained herein shall be construed to confer any license or right to any intellectual property, whether or not the use of any information herein necessarily utilizes such intellectual property. This document is not to be construed as an endorsement of any product or company or as the adoption or promulgation of any guidelines, standards, or recommendations.

Document Status Sheet

Document Control Number:	OC-SP-ESAM-API-I03-131025			
Document Title:	Real-time Event Signaling and Management API			
Revision History:	I01 – Released 9/10/12 I02 – Released 10/1/13 I03 – Released 10/25/13			
Date:	October 25, 2013			
Status:	~~Work in Progress~~	~~Draft~~	**Issued**	~~Closed~~
Distribution Restrictions:	~~Author Only~~	~~CL/Member~~	~~CL/ Member/ Vendor~~	**Public**

Key to Document Status Codes

Work in Progress	An incomplete document, designed to guide discussion and generate feedback that may include several alternative requirements for consideration.
Draft	A document in specification format considered largely complete, but lacking review by Members and vendors. Drafts are susceptible to substantial change during the review process.
Issued	A stable document, which has undergone rigorous member and vendor review and is suitable for product design and development, cross-vendor interoperability, and for certification testing.
Closed	A static document, reviewed, tested, validated, and closed to further engineering change requests to the specification through CableLabs.

Trademarks

Contents

Figures

Tables

This page left blank intentionally.

1 SCOPE

1.1 Introduction and Purpose

This document details the APIs for processing real-time signals (e.g., SCTE 35 and others) and real-time manifest generation. It also supports asynchronous signal processing for live streams and URI based content (ex. file reference).

Documented herein are two interfaces that are used either for signal processing and stream conditioning or for manifest conditioning.

The first interface allows a signal acquisition system (e.g., transcoder, fragmenter, etc.) to submit signals to a signal processing system and receive stream conditioning data and directives (Section 8); while the second interface allows submission of a confirmed signal and receipt of manifest manipulation instructions (Section 9). A given device may be comprised of one or more signal acquisition systems.

For example, a real-time transcoder submits an SCTE 35 splice insert message to a Placement Opportunity Information System (POIS). The POIS confirms the validity of the signal and returns information allowing a transcoder to identify and update the start or end times of a signal region. The call also may return relevant information such as signal region duration(s) that could be used for conditioning the stream being encoded. The returned information may additionally include auxiliary data to be inserted into the video stream for downstream systems, which could consume the auxiliary data and process according to specific application requirements.

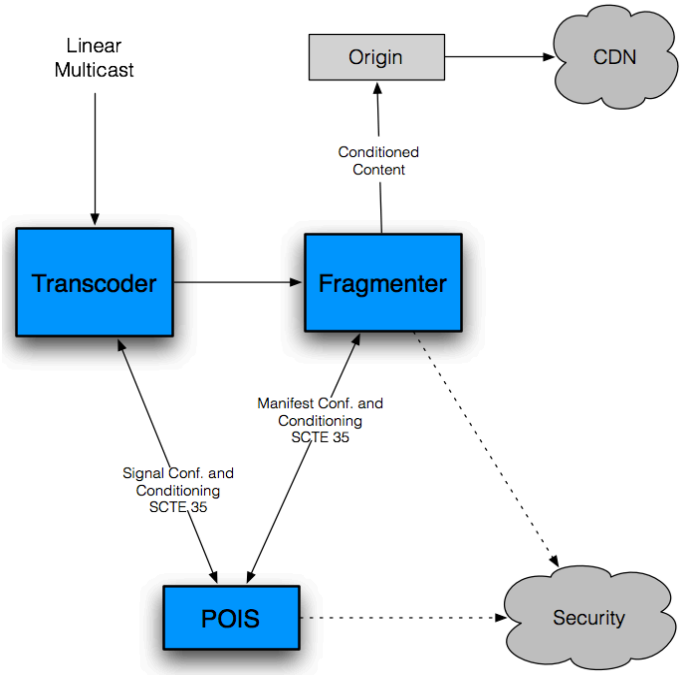

Figure 1 - Example of Real-time IP Video Application

Furthermore, in an Adaptive Bitrate (ABR) packaging application, the ABR fragmenter also submits the previously confirmed signal to the POIS. In return, manifest-specific conditioning instructions could be provided, which the fragmenter inserts on behalf of the end-to-end system.

For URI based content, a transcoder can operate in one of two operational modes. In the first mode, the transcoder calls the POIS to obtain the signal regions for content in a file. Figure 2 illustrates this model.

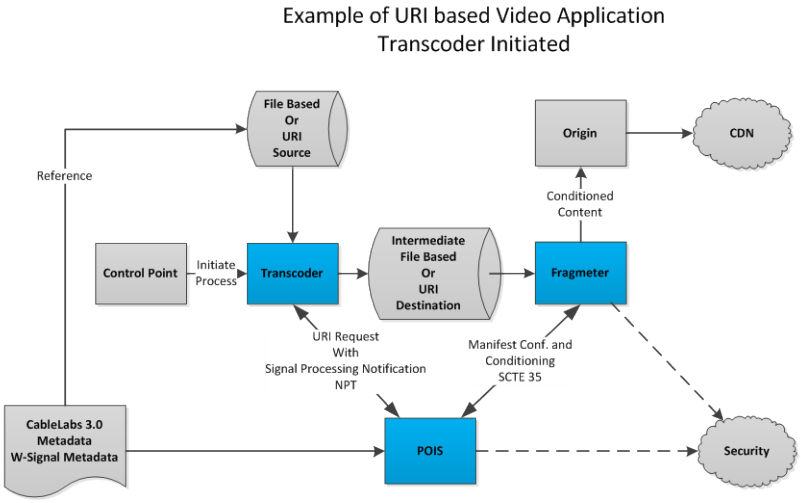

Figure 2 - Example of File-based IP Video Application (transcoder initiated)

The following illustrates the message exchanges between the Transcoder and the POIS to initiate and monitor a batch process.

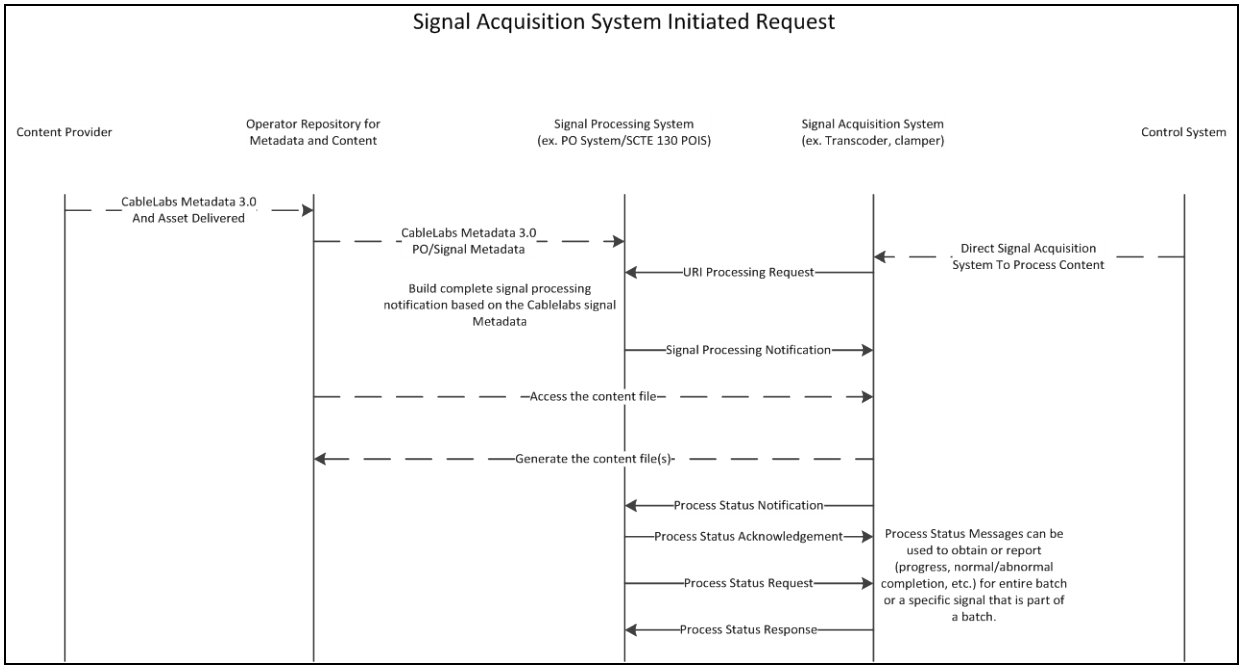

Figure 3 - Mode 1 Message Exchange

In the second mode, the transcoder is directed to process the file-based content based on directives in the request. Figure 4 illustrates this model.

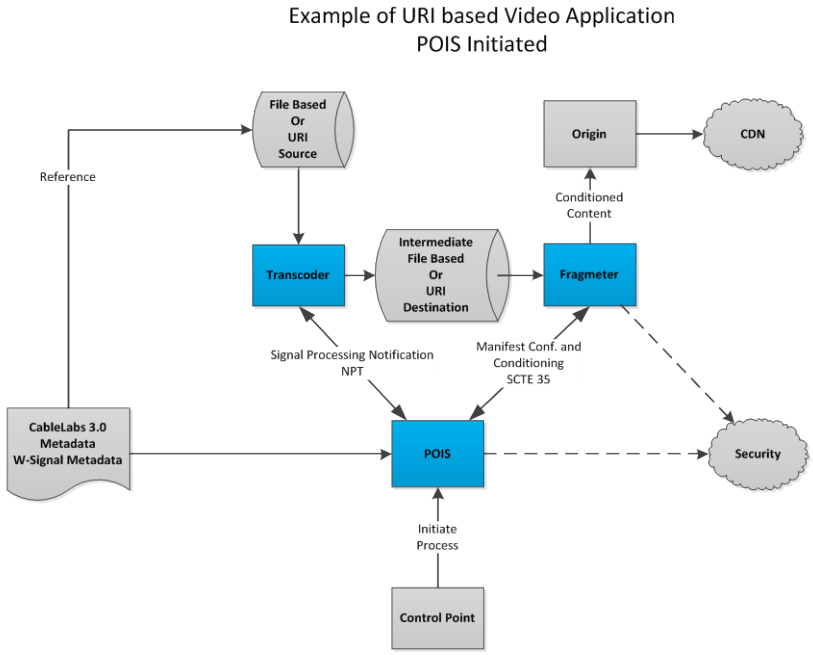

Figure 4 - Example of file-based IP video application (POIS Initiated)

The following illustrates the message exchanges between the POIS and the Transcoder to initiate and monitor a batch process.

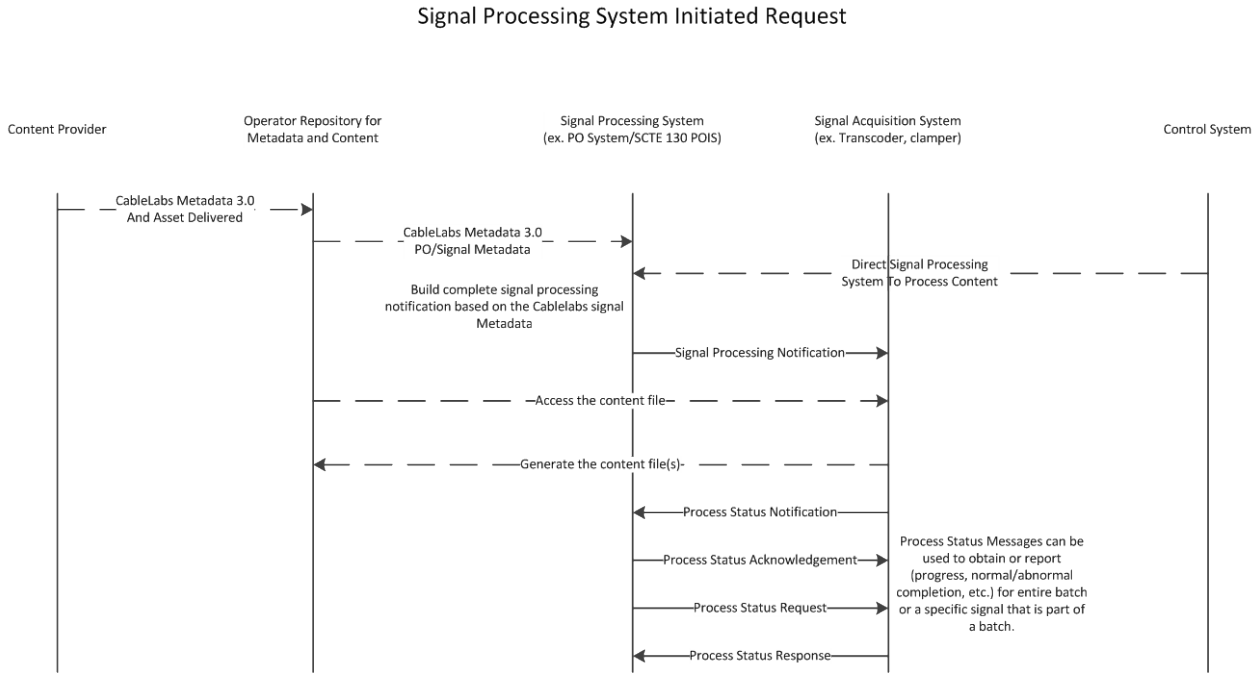

Figure 5 - Mode 2 Message Exchange

In both modes of operation, the Process Status Notification/Process Status Acknowledgement and Process Status Request/Process Response messages are used to obtain or report on the status of a batch request. For example, in both modes, an implementation may require a Process Status Notification be sent by the Signal Acquisition System or issue a Process Status Request from the Signal Processing System as soon as the Signal Processing Notification message is sent to the Signal Acquisition System.

To fully understand the ABR conditioning portions of this document, the reader is expected to be familiar with and understand the different ABR delivery formats and their individual terminology.

The Event Signaling and Management API supports both JSON and XML event and notification message payload formats with the caller controlling the payload format using standard HTTP semantics. This specification leverages the CableLabs Metadata 3.0 signaling schema.

1.2 Requirements

Throughout this document, the words that are used to define the significance of particular requirements are capitalized. These words are:

"SHALL"	This word means that the item is an absolute requirement of this specification.
"SHALL NOT"	This phrase means that the item is an absolute prohibition of this specification.
"SHOULD"	This word means that there may exist valid reasons in particular circumstances to ignore this item, but the full implications should be understood and the case carefully weighed before choosing a different course.
"SHOULD NOT"	This phrase means that there may exist valid reasons in particular circumstances when the listed behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.
"MAY"	This word means that this item is truly optional. 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.

2 REFERENCES

2.1 Normative References

In order to claim compliance with this specification, it is necessary to conform to the following standards and other works as indicated, in addition to the other requirements of this specification. Notwithstanding, intellectual property rights may be required to use or implement such normative references.

All references are subject to revision, and parties to agreement based on this specification are encouraged to investigate the possibility of applying the most recent editions of the documents listed below.

[CEP 3.0] Content Encoding Profiles 3.0 Specification, OC-SP-CEP3.0-I04-121210, December 10, 2012, Cable Television Laboratories, Inc.

[CONTENT 3.0] CableLabs Metadata 3.0 Specification, MD-SP-CONTENTv3.0-I02-121210, December 10, 2012, Cable Television Laboratories, Inc.

[ID-HLS] Internet Draft, HTTP Live Streaming, draft-pantos-http-live-streaming-08, expires September 24, 2012.

[ISO 8601] ISO 8601:2004, Data elements and interchange formats -- Information interchange -- Representation of dates and times (Coordinated Universal Time).

[JSON] JavaScript Object Notation (JSON), http://www.json.org.

[RFC 2616] IETF RFC 2616, Hypertext Transfer Protocol -- HTTP/1.1. R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee. June 1999.

[RFC 4648] IETF RFC 4648, The Base16, Base32, and Base64 Data Encodings. S. Josefsson. October 2006.

[SCTE 35] ANSI/SCTE 35 2012, Digital Program Insertion Cueing Message for Cable.

2.2 Informative References

This specification uses the following informative references.

[SCTE 67] ANSI/SCTE 67 2010, Recommended Practice for SCTE 35 Digital Program Insertion Cueing Message for Cable.

2.3 Reference Acquisition

- Cable Television Laboratories, Inc., 858 Coal Creek Circle, Louisville, CO 80027;
  Phone +1-303-661-9100; Fax +1-303-661-9199; http://www.cablelabs.com
- Internet Engineering Task Force (IETF), Internet: http://www.ietf.org/
  Note: Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time.
  The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.
  Internet-Drafts may also be accessed at http://tools.ietf.org/html/
- SCTE, Society of Cable Telecommunications Engineers, Inc., 140 Philips Road, Exton, PA 19341 Phone: +1-800-542-5040, Fax: +1-610-363-5898, Internet: http://www.scte.org

3 TERMS AND DEFINITIONS

This specification uses the following terms:

Base64 Binary	Binary contents coded in Base64 format.
Fragment	Small content chunk (typically two seconds of video).
Fragmenter	Also referred to as packager or encapsulator.
Media Segment	The media segment is the M3U8 extended recorded commencing with the #EXTINF tag through to its paired URI line inclusive of any line in between that starts with #EXT.
Segment	An MPEG-2 Transport Stream divided into a series of small media files typically of equal duration (for example ten seconds).
Signal Point	A particular point of interest within a video essence.
Signal Region	A region of interest within a video essence.

4 ABBREVIATIONS AND ACRONYMS

This specification uses the following abbreviations:

ABR	Adaptive Bitrate
API	Application Programming Interface
DASH	MPEG Dynamic Adaptive Streaming over HTTP
GUID	Globally Unique Identifier
HDS	HTTP Dynamic Streaming (Adobe Zeri)
HLS	HTTP Live Streaming (Apple)
HSS	HTTP Smooth Streaming (Microsoft Smooth)
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
LAN	Local Area Network
NTP	Network Time Protocol
POIS	Placement Opportunity Information System
URI	Uniform Resource Identifier
UTC	Coordinated Universal Time
VOD	Video on Demand
WAN	Wide Area Network
XML	Extensible Markup Language

5 MESSAGE TRANSPORT RECOMMENDED PRACTICE

The signal processing services are exposed as an HTTP RESTful endpoint. Each API has a unique URI and both the event payload and the notification payload SHALL support XML objects and MAY support JSON. HTTP POST is the required request method. The request message's HTTP Accept header is set to the value "application/json" or "application/xml" for the application's desired return data format.

The server SHALL return a valid HTTP [RFC 2616] status code indicating the result of the message exchange and application processing. For application processing errors, the HTTP payload SHALL contain additional information in the StatusCode element, as detailed in Section 6.1.

System to system communications are considered to be in a trusted environment; therefore, security-related concerns are currently outside the scope of this specification. Further, systems may experience communications failures, and it is left to the implementer to devise the best messaging resiliency and timeout tactics to meet specific customer needs. It is suggested the implementer provide configurations around potential retries in the event synchronous notifications are not received, as well as configurations around what behavior would be expected in such cases. For example, one might provide a default permissive configuration in the event a notification is not received: the signaled event is allowed to pass. Another may take the same behavior but also retry multiple times on some interval. Another may take a more restrictive default approach where it is not allowed to pass if a timeout or ambiguous error occurs. All of these are valid and will depend on the needs and characteristics of the environment. It is therefore strongly suggested that implementations allow for default behavior to be configurable.

5.1 Time Synchronization

It is expected that time synchronization with multiple high accuracy sources will be maintained by all components of the systems. Protocols such as NTP allow time synchronization in the sub-millisecond on LANs and up to a few milliseconds on WANs.

6 COMMON CONVENTIONS

Note 1: Unless noted as optional, all attributes, elements, and objects are required.

Note 2: In all cases, unrecognized attributes, elements, and objects are to be ignored.

As a convention used throughout this API, a JSON array is identified using a plural name like 'spots' or 'segments'. The XML equivalent utilizes an XML element sequence with a singular element tag that is capitalized., for example, 'Spot' or 'Segment'. In documentation situations herein where duplicating the element name is redundant or confusing, the XML value is utilized and the JSON equivalent is to be assumed by the reader.

6.1 StatusCode

The StatusCode object provides return status information to the caller and is returned for all application-level errors. The StatusCode object may optionally be included in a response payload to provide warning or informational details in addition to the typically expected payload.

The StatusCode element has the following XML schema:

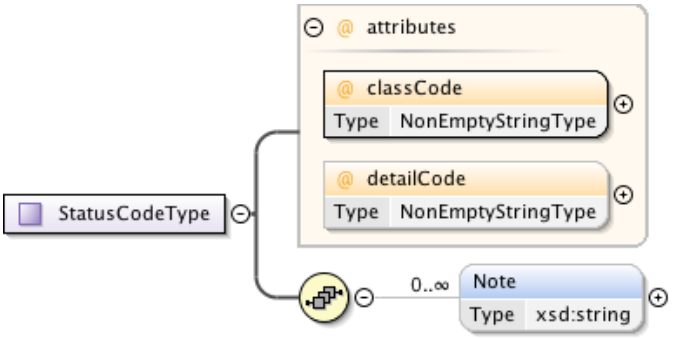

Figure 6 - StatusCode XML Schema

6.1.1 StatusCode Semantics

Table 1 - StatusCode Semantic Attributes

Name	Required	Description
classCode	Y	One of the values from Table 3 as a string.
detailCode	N	Optional value as a string with a specific code relative to the classCode. If the classCode is set to error, the detail code SHOULD be present. If the classCode is a success, the detailCode SHOULD NOT be present. Table 4 enumerates the defined values for detailCode.

Table 2 - StatusCode Semantics Elements

Name	Required	Description
Note	N	Optional sequence of descriptive strings. Typically, the notes/Note object provides descriptive text such as human readable error messages.

Table 3 - classCode Attribute Values

Value	Description
0	Success
1	Error
2	Warning
3	Information

Table 4 - detailCode Attribute Values

Value	Description
0	Reserved
1	General error
2	Resource not found
3	Missing mandatory input parameter

6.2 Date and Time Format (UTCZulu)

Object time values SHALL follow the [ISO 8601] extended time format of [hh]:[mm]:[ss] except the *StreamTime* object values which are in their native time formats. The following list describes the extended time formats:

[hh] — Zero-padded hour between 00 and 24 (where 24 is only used to notate midnight at the end of a calendar day).

[mm] — Minute between 00 and 59.

[ss] — Second between 00 and 60 (where 60 is only used to notate an added leap second).

Decimal fractions MAY be added to the seconds component using a decimal point as a separator between the time element and its fraction. The number of decimal places for the decimal fraction SHALL be limited to three places for ISO 8601 times.

Combined date and time objects SHALL use the UTC format, and all times SHALL be provided as zero UTC offset or Zulu times, referred to herein as *UTCZulu*. Thus, all combined date and time objects SHALL be formatted as:

2001-12-17T11:12:42.123Z

6.3 Duration Format (ISODuration)

Duration values, referred to as *ISODuration* herein, SHALL follow the [ISO 8601] format as well and are represented by the format P[n]Y[n]M[n]DT[n]H[n]M[n]S. In these representations, the [n] is replaced by the value for each of the date and time elements that follow the [n]. Leading zeroes MAY be included, if applicable. The capital letters *P*, *Y*, *M*, *W*, *D*, *T*, *H*, *M*, and *S* are designators for each of the date and time elements and are **not** replaced.

The following list describes the date and time elements:

P — Duration designator placed at the start of the duration representation.

Y — Year designator that follows the value for the number of years.

M — Month designator that follows the value for the number of months.

W — Week designator that follows the value for the number of weeks.

D — Day designator that follows the value for the number of days.

T — Time designator that precedes the time components of the representation.

H — Hour designator that follows the value for the number of hours.

M — Minute designator that follows the value for the number of minutes.

S — Second designator that follows the value for the number of seconds.

For example, "P3Y6M4DT12H30M5S" represents a duration of three years, six months, four days, twelve hours, thirty minutes, and five seconds. Date and time elements including their designator may be omitted if their value is zero, and lower order elements MAY also be omitted for reduced precision. For example, "P23DT23H" and "P4Y" are both acceptable duration representations.

To resolve ambiguity, "P1M" is a one-month duration and "PT1M" is a one-minute duration (note the time designator, T, that precedes the time value). Seconds MAY have a decimal fraction specified with a period as "PT30.5S", which is an example of 30 and one-half seconds.

The duration value SHALL be formatted using the canonical form where each field does not exceed the field's standard maximum value. The second and minute values SHALL be less than 60, and the hours field SHALL be less than 24. For example, 90-seconds is to be formatted as PT1M30S and **not** as PT90S.

6.4 Base64 Binary Format

Binary contents SHALL be coded in Base64 format per section 6.8 of [RFC 4648] with W3C recommendations.

6.5 StreamTimes

The *StreamTimes* object SHALL carry one or more native stream time values. There SHALL be at least one stream time entry when present.

The XML schema SHALL be a sequence of one or more of the following:

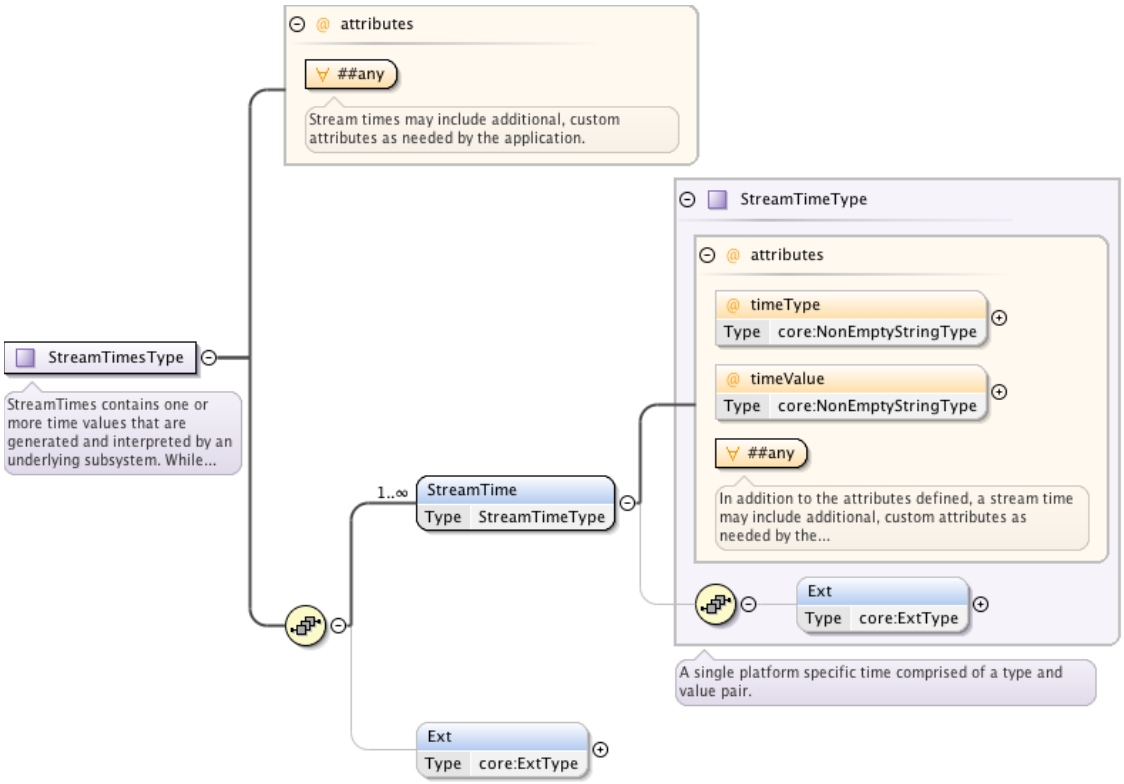

Figure 7 - StreamTime XML Schema

6.5.1 StreamTime type Semantics

Table 5 - StreamTime Type Semantics Attributes

Name	Required	Description
timeType	N	A value exactly as it appears from Table 6.
timeValue	N	The native format time value encoding of the signal point (e.g., the splice time or cue time) as specified in Table 7.

Table 6 - timeType Attribute Values

Value	Description
DASH	MPEG DASH
HLS	Apple HTTP Live Streaming
NPT	Normal Play Time
PTS	MPEG-2 Transport Stream
HSS	Microsoft HTTP Smooth Streaming (HSS)
HDS	Adobe HTTP Dynamic Streaming (HDS/Zeri)
SignalID	CableLabs Metadata 3.0 Signal ID
private:*	Add extensibility

Table 7 - timeValue Formats

Identifier	Description
DASH	TBD
HLS	Time in 90 KHz clock ticks. The MPEG PTS value.
NPT	TBD
PTS	Time in 90 KHz clock ticks. The MPEG PTS value.
HSS	Time in hundred nanosecond units.
HDS	Time in seconds dot ('.') decimal fractional second.
SignalID	Value passed as coded in the CableLabs metadata. The value may be a GUID, base64 encoded GUID or other value.
private:*	extensibility

7 COMMON API

The Common API is a set of messages used for both Signal Confirmation and Conditioning (see Section 8) and Manifest Confirmation and Conditioning (see Section 9) to support asynchronous processing of URI based content (ex. a VOD asset file), status information, and other common messages.

7.1 BatchInfo

For URI based processing, the *BatchInfo* element identifies the batch, and optionally identifies the content source and one or more content destinations. The following table calls out specific treatment of the attributes and elements that are part of the BatchInfo element.

The batchId attribute SHALL be specified. The batchId uniquely identifies the request and SHALL be passed on all subsequent ESAM messages specific to the batch.

BatchInfo references the CableLabs Metadata 3.0 MovieType for the source and destination elements (see [CONTENT 3.0]). The SourceURL SHALL be specified when using the MovieType when specifying source or destination content.

Table 8 - BatchInfo

Name	Required	Description
batchId	Y	SHALL be set to a GUID. The batchId uniquely identifies the request. This batchId is passed on all subsequent ESAM messages specific to the batch.
Source	N	Optional element specifying the source of the content.
Destination	N	Optional element specifying the destination of the processed content. One or more Destination elements MAY be specified.

Table 9 provides usage detail of the CableLabs 3.0 content:MovieType element. In addition to the attributes and elements listed in Table 9, an implementation MAY choose to use other attributes and elements. Specific usage of those attributes and elements is out of scope of this specification.

Table 9 - CableLabs content:MovieType Usage

Name	Required	Description
xsi:type	Y	SHALL be set to “content:MovieType”.
uriId	Y	SHALL be set to the identifier for asset referenced in content:SourceUrl.
Content:SourceUrl	Y	SHALL be set to the URL to the content source to process or destination to create.
Content:POGroupRef	OMIT	SHOULD NOT be present. If present, the value can be ignored.
Content:SignalGroupRef	OMIT	SHOULD NOT be present. If present, the value can be ignored.

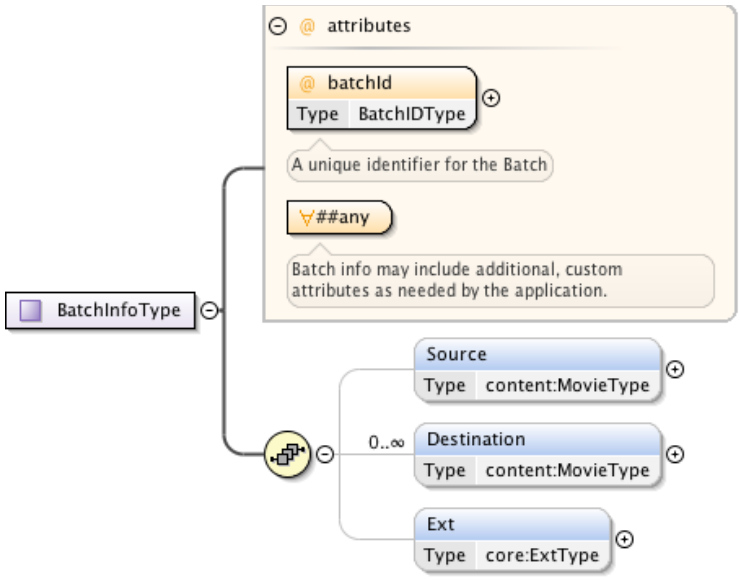

Figure 8 - BatchInfoType XML Schema

7.2 ProcessStatusNotification

The ProcessStatusNotification message is used to send status information about a process. This includes a UriProcessingRequest identified by a *batchId* or processing related to an *acquisitionSignalID*.

If the signal processing system does not recognize or support the specified attribute/element or its semantics, the signal processing system is to ignore the data.

The following XML payload is returned:

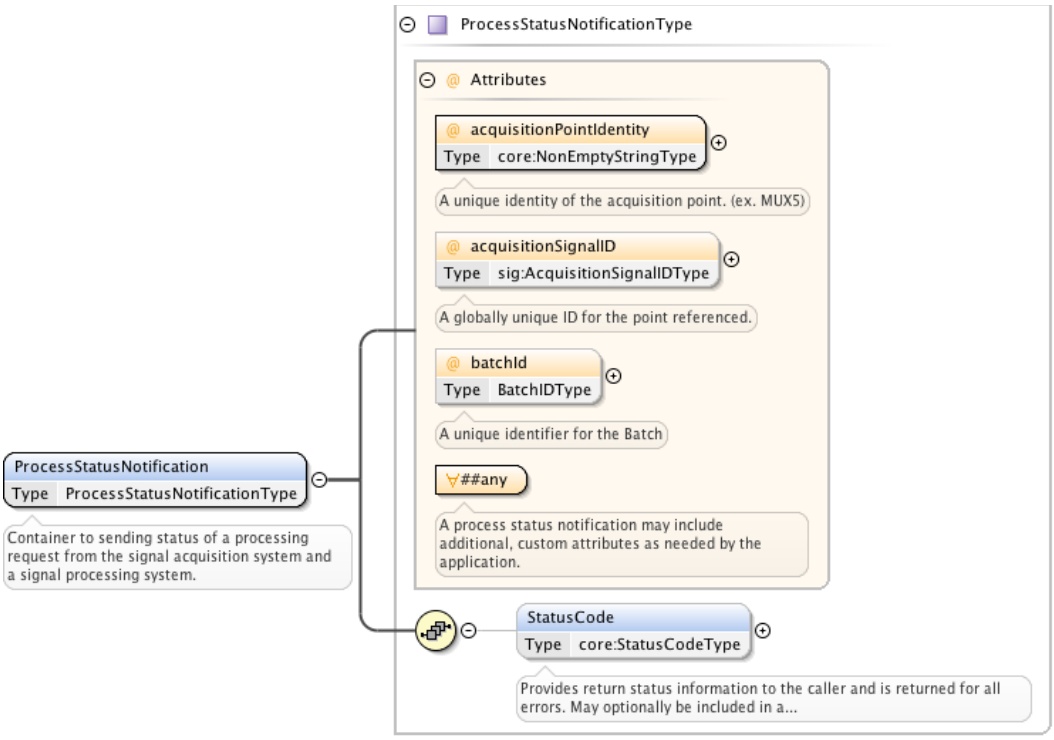

Figure 9 - ProcessStatusNotification XML Schema

7.2.1 ProcessStatusNotification Attributes

Table 10 - ProcessStatusNotification Attributes

Name	Required	Description
acquisitionPointIdentity	Y	The identification of the processing point reporting status SHALL be specified.
acquisitionSignalID	O	An optional acquisitionSignalID MAY be specified.
batchId	O	An optional batchId attribute MAY be specified.

An *acquisitionSignalID* and/or *batchId* MAY be supplied depending on the use case.

1. Only *batchId* is passed to report the status of a batch.
2. *acquisitionSignalID* and/or *batchId* are passed if reporting on the processing of a specific *acquisitionSignalID* in a batch.
3. Only *acquisitionSignalID* is passed to report status on a specific *acquisitionSignalId* during a non-batch process.

StatusCode information SHALL also be returned. See Section 6.1 for more information on *StatusCode.* The classCode attribute SHALL be compliant with Table 3 in Section 6.1.1. The values returned SHALL be compliant with the following:

- Return 0 (Success) upon successful completion of a batch operation.
- Return 1 (Error) upon unsuccessful complete of a batch operation. A *detailCode* value from Table 4 in Section 6.1.1 SHOULD also be returned.
- Return 3 (Information) if the batch operation has not yet completed.

One or more *Note* Elements SHOULD also be supplied to provide additional details.

7.2.2 ProcessStatusNotification Example

To improve readability, all XML schemas in this section are assumed to have the following namespace declaration:

```
xsi:schemaLocation="urn:cablelabs:iptvservices:esam:xsd:common:1 OC-SP-ESAM-
API-I0x-Common.xsd"
xmlns="urn:cablelabs:iptvservices:esam:xsd:common:1"
xmlns:core="urn:cablelabs:md:xsd:core:3.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

7.2.2.1 Successful Process Status Notification Message

The transcoder notifies the POIS that it has successfully completed a URI request.

```
<ProcessStatusNotification acquisitionPointIdentity="Batch_Processing_Point_1"
batchId="27b777c1ee9941479677b56203953300" acquisitionSignalID="4A6A94EE-62FA-11E1-B1CA-
882F4824019B">
        <StatusCode classCode="0">
                <core:Note>Processing completed</core:Note>
        </StatusCode>
</ProcessStatusNotification>
```

7.2.2.2 Partially Complete Process Status Notification Message

The transcoder notifies the POIS that it has completed 75% of a URI request.

```
<ProcessStatusNotification acquisitionPointIdentity="Batch_Processing_Point_1"
batchId="27b777c1ee9941479677b56203953300" acquisitionSignalID="4A6A94EE-62FA-11E1-B1CA-
882F4824019B">
        <StatusCode classCode="3">
                <core:Note>Processing 75% complete</core:Note>
        </StatusCode>
</ProcessStatusNotification>
```

7.2.2.3 Failed Batch Status Notification Message

The transcoder notifies the POIS that it has failed to complete a URI request due to an unfound resource.

```
<ProcessStatusNotification acquisitionPointIdentity="Batch_Processing_Point_1"
batchId="27b777c1ee9941479677b56203953300" acquisitionSignalID="4A6A94EE-62FA-11E1-B1CA-
882F4824019B">
        <StatusCode classCode="1" detailCode=”2”>
                <core:Note>Source not found.</core:Note>
        </StatusCode>
</ProcessStatusNotification>
```

7.3 ProcessStatusAcknowledgement

The *ProcessStatusAcknowledgment* message is used to acknowledge a *ProcessStatusNotification* (see Section 7.2) message.

If the signal processing system does not recognize or support the specified attribute/element or its semantics, the signal processing system is to ignore the data.

The following XML payload is returned:

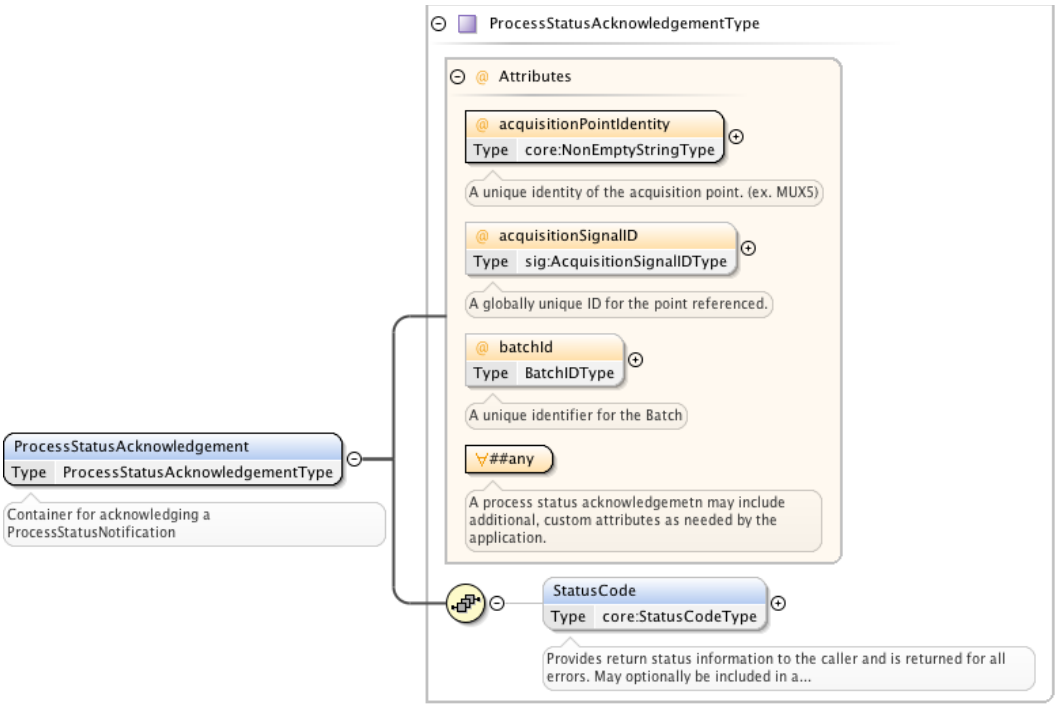

Figure 10 - ProcessStatusAcknowledgement XML Schema

7.3.1 ProcessStatusAcknowledgement Attributes

Table 11 - ProcessStatusAcknowledgement Attribute

Name	Required	Description
acquisitionPointIdentity	Y	The identification of the processing point reporting status SHALL be specified.
acquisitionSignalID	O	An optional acquisitionSignalID MAY be specified.
batchId	O	An optional batchId attribute MAY be specified.

The *acquisitionSignalID* and *batchId* values, if present, SHALL be returned as passed in the *ProcessStatusNotification* message. *StatusCode* information may also be returned. See Section 6.1 for more information on *StatusCode.*

7.3.2 ProcessStatusAcknowledgement Example

To improve readability, all XML schemas in this section are assumed to have the following namespace declaration:

```
xsi:schemaLocation="urn:cablelabs:iptvservices:esam:xsd:common:1 OC-SP-ESAM-
API-I0x-Common.xsd"
xmlns="urn:cablelabs:iptvservices:esam:xsd:common:1"
xmlns:core="urn:cablelabs:md:xsd:core:3.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

7.3.2.1 ProcessStatusAcknowledgement Message

The POIS acknowledges the transcoder status message.

<ProcessStatusAcknowlegement acquisitionPointIdentity="Batch_Processing_Point_1"
batchId="27b777c1ee9941479677b56203953300">

```
        <StatusCode classCode="0">
                <core:Note>Acknowledge</core:Note>
        </StatusCode>
</ProcessStatusAcknowledgement>
```

7.4 ProcessStatusResponse

The ProcessStatusResponse message is sent in response to a *ProcessStatusRequest* message (see Section 7.5). This includes an UriProcessingRequest identified by a *batchId* or processing related to an *acquisitionSignalID*.

If the signal processing system does not recognize or support the specified attribute/element or its semantics, the signal processing system is to ignore the data.

The following XML payload is returned:

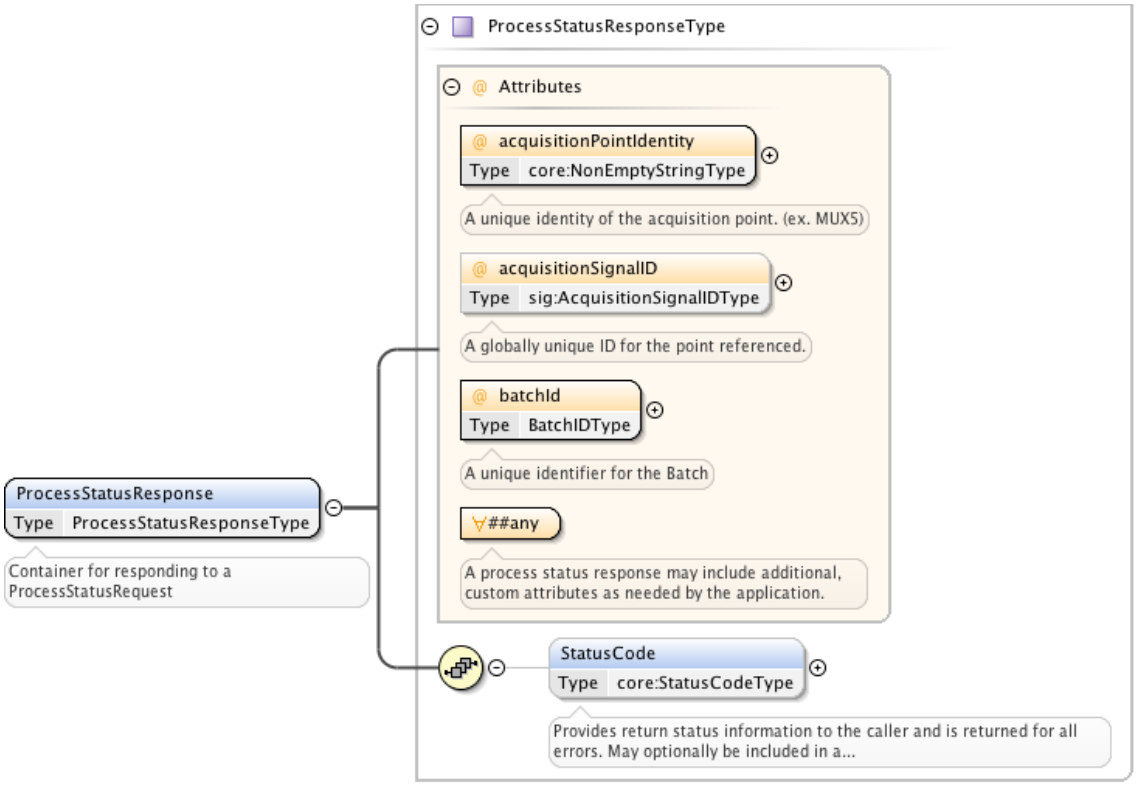

Figure 11 - ProcessStatusResponse XML Schema

7.4.1 ProcessStatusResponse Attributes

Table 12 - ProcessStatusResponse Attribute

Name	Required	Description
acquisitionPointIdentity	Y	The identification of the processing point reporting status SHALL be specified.
acquisitionSignalID	O	An optional acquisitionSignalID MAY be specified.
batchId	O	An optional batchId attribute MAY be specified. The batchId uniquely identifies the request.

The *acquisitionSignalID* and *batchId* values, if present, SHALL be returned as passed in the *ProcessStatusRequest* message.

StatusCode information SHALL also be returned. See Section 6.1 for more information on *StatusCode.* The classCode attribute SHALL be compliant with Table 3 in Section 6.1.1. The values returned SHALL be compliant with the following:

- Return 0 (Success) upon successful completion of a batch operation.
- Return 1 (Error) upon unsuccessful complete of a batch operation. A detailCode value from Table 4 in Section 6.1.1 SHOULD also be returned.
- Return 3 (Information) if the batch operation has not yet completed.

One or more *Note* Elements SHOULD also be supplied to provide additional details.

7.4.2 ProcessStatusResponse Example

To improve readability, all XML schemas in this section are assumed to have the following namespace declaration:

```
xsi:schemaLocation="urn:cablelabs:iptvservices:esam:xsd:common:1 OC-SP-ESAM-
API-I0x-Common.xsd"
xmlns="urn:cablelabs:iptvservices:esam:xsd:common:1"
xmlns:core="urn:cablelabs:md:xsd:core:3.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

See Section 7.3.2 for addition examples of the usage of the *StatusCode* Element.

7.4.2.1 Successful Process Status Response Message

The transcoder responds to the POIS that it has successfully completed processing a specific signal in a URI request.

```
<ProcessStatusResponse acquisitionPointIdentity="Batch_Processing_Point_1"
batchId="27b777c1ee9941479677b56203953300" acquisitionSignalID="4A6A94EE-62FA-11E1-B1CA-
882F4824019B">
        <StatusCode classCode="0">
                <core:Note>Processing completed</core:Note>
        </StatusCode>
</ProcessStatusResponse>
```

7.5 ProcessStatusRequest

The ProcessStatusRequest message is sent to obtain status on a process. A *ProcessStatusResponse (see Section 7.4)* is sent in response to a *ProcessStatusRequest* message. This includes a UriProcessingRequest identified by a *batchId* or processing related to an *acquisitionSignalID*.

If the signal processing system does not recognize or support the specified attribute/element or its semantics, the signal processing system is to ignore the data.

The following XML payload is returned:

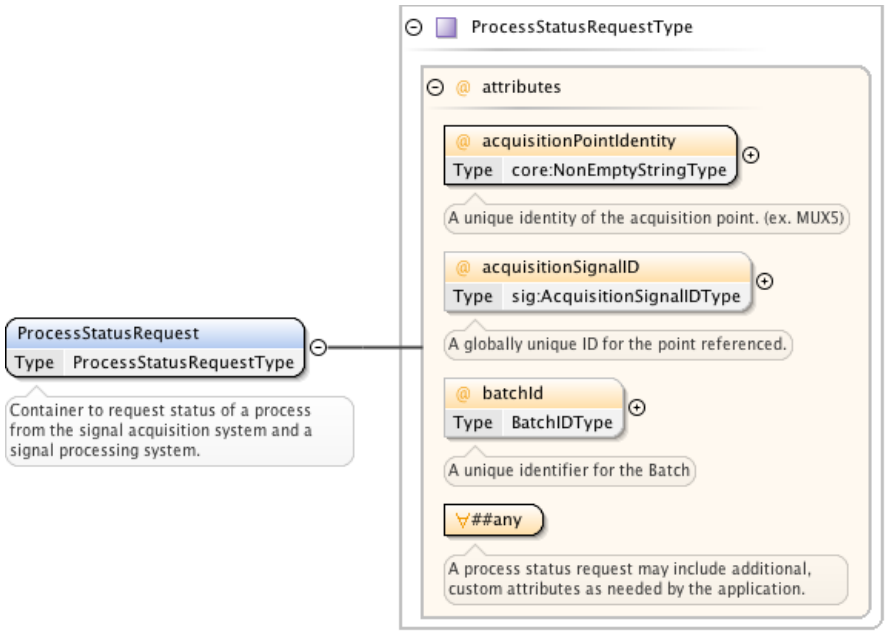

Figure 12 - ProcessStatusRequest XML Schema

7.5.1 ProcessStatusRequest Attributes

Table 13 - ProcessStatusRequest Attributes

Name	Required	Description
acquisitionPointIdentity	Y	The identification of the processing point reporting status SHALL be specified.
acquisitionSignalID	N	An optional acquisitionSignalID MAY be specified.
batchId	N	An optional batchId attribute MAY be specified. The batchId uniquely identifies the request. This batchId is passed on all subsequent ESAM messages specific to the batch.

An *acquisitionSignalID* and/or *batchId* MAY be supplied depending on the use case.

1. Only *batchId* is passed to request the status of a batch.
2. *acquisitionSignalID* and/or *batchId* are passed if requesting on the processing of a specific *acquisitionSignalID* in a batch.
3. Only *acquisitionSignalID* is passed to request status on a specific *acquisitionSignalId* during a non-batch process.

7.5.2 *ProcessStatus*Request Example

To improve readability, all XML schemas in this section are assumed to have the following namespace declaration:

```
xsi:schemaLocation="urn:cablelabs:iptvservices:esam:xsd:common:1 OC-SP-ESAM-
API-I0x-Common.xsd"
xmlns="urn:cablelabs:iptvservices:esam:xsd:common:1"
xmlns:core="urn:cablelabs:md:xsd:core:3.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

7.5.2.1 Process Status Request Message

The POIS queries the transcoder about processing of a specific signal id that is part of a URI request.

<ProcessStatusRequest acquisitionPointIdentity="Batch_Processing_Point_1" batchId="27b777c1ee9941479677b56203953300" acquisitionSignalID="4A6A94EE-62FA-11E1-B1CA-882F4824019B"/>

7.6 SignalStateRequest

The SignalStateRequest message is used to obtain conditioning information for a Linear Stream. The main use case of this would be to handle failover of a transcoder. When the new transcoder comes on line, it would reach out to the signal processing system to determine the state of the linear stream (ex. whether it is in a blackout or not). The signal processing system is expected to send the appropriate notification message in response.

The following XML payload is passed in the SignalStateRequest message:

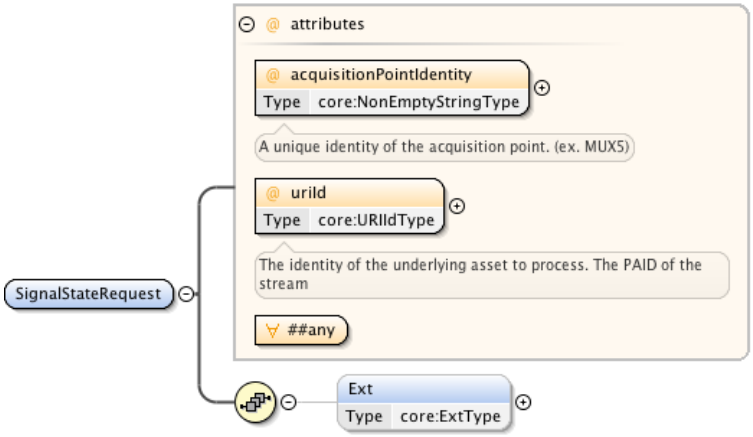

Figure 13 - SignalStateRequest XML Schema

7.6.1 SignalStateRequest Attributes

Table 14 - SignalStateRequest Attributes

Name	Required	Description
acquisitionPointIdentity	Y	The acquisitionPointIdentity attribute SHALL be specified. A unique identity of the acquisition point identifying the signal acquisition system (ex. transcoder) used to process the content available at the specified URI.
uriId	Y	SHALL be set to the identifier for asset being processed.

7.6.2 SignalStateRequest Example

To improve readability, all XML schemas in this section are assumed to have the following namespace declaration:

```
xsi:schemaLocation="urn:cablelabs:iptvservices:esam:xsd:common:1 OC-SP-ESAM-API-I0x-Common.xsd"
xmlns="urn:cablelabs:iptvservices:esam:xsd:common:1"
xmlns:core="urn:cablelabs:md:xsd:core:3.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

7.6.2.1 SignalStateRequest Message

<SignalStateRequest acquisitionPointIdentity="mso.com/ProcessingPt"
uriId="provider.com/Asset/UNVA2001081701004002">
</SignalStateRequest >

7.7 UriProcessingRequest

The UriProcessingRequest message is used to obtain conditioning information for URI based content. See Section 8.5 for a sample SignalProcessingNotification message returned in response to a UriProcessingRequest.

If the signal processing system does not recognize or support the specified attribute/element or its semantics, the signal processing system is to ignore the data.

The following XML payload is returned:

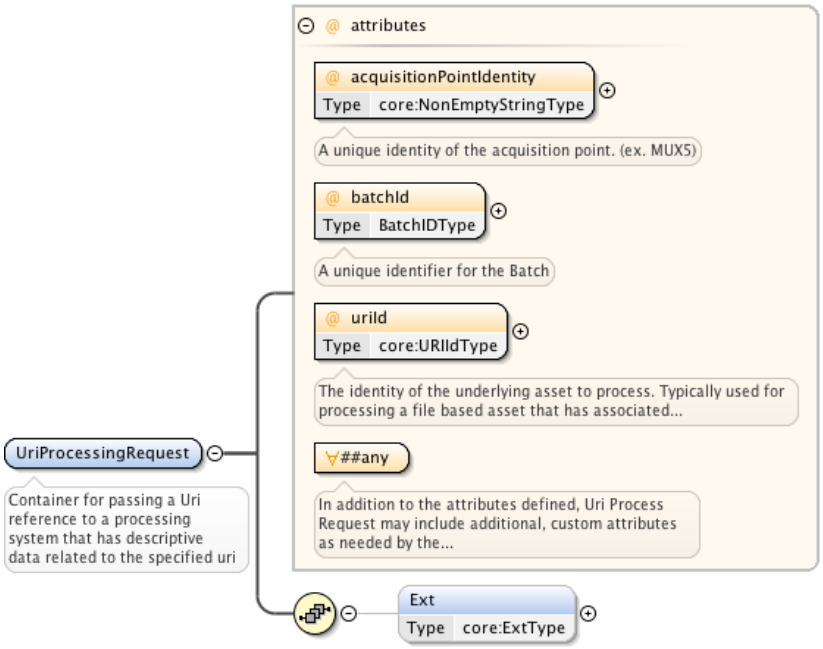

Figure 14 - UriProcessingRequest XML Schema

7.7.1 UriProcessingRequest Attributes

Table 15 - UriProcessingRequest Attributes

Name	Required	Description
acquisitionPointIdentity	Y	The acquisitionPointIdentity attribute SHALL be specified. A unique identity of the acquisition point identifying the signal acquisition system (ex. transcoder) used to process the content available at the specified URI.
batchId	Y	The batchId attribute SHALL be specified. The batchId uniquely identifies the request. This batchId is passed on all subsequent ESAM messages specific to the batch.
uriId	Y	SHALL be set to the identifier for asset being processed.

7.7.2 UriProcessingRequest Example

To improve readability, all XML schemas in this section are assumed to have the following namespace declaration:

```
xsi:schemaLocation="urn:cablelabs:iptvservices:esam:xsd:common:1 OC-SP-ESAM-
API-I0x-Common.xsd"
xmlns="urn:cablelabs:iptvservices:esam:xsd:common:1"
xmlns:core="urn:cablelabs:md:xsd:core:3.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

7.7.2.1 Batch Request Message

The transcoder notifies the POIS that it is about to process URI sourced content. See Section 8.5.2.6 for a sample response message.

<UriProcessingRequest acquisitionPointIdentity="mso.com/VODProcessingPt/1"
batchId="27b777c1ee9941479677b56203953300" uriId="provider.com/Asset/UNVA2001081701004002">
</UriProcessingRequest>

8 SIGNAL CONFIRMATION AND CONDITIONING API

The Signal Confirmation and Conditioning API is used between a signal acquisition system (e.g., transcoder, fragmenter, etc.), and a signal processing system (e.g., POIS). The originating system generates a signal event and submits it using the defined JSON or XML payload to the processing system. The event parameters provide as much information as possible about the signal point. Example signal points are a splice out/exit point indicated by an SCTE 35 splice_info_section()/splice_insert() command or an SCTE 35 segmentation_descriptor associated with an SCTE 35 time_signal() command.

Using the provided information, the processing system confirms the validity of the signal and finds the appropriate placement opportunity metadata. Then, the system derives information about the Signal Region start point and end point(s) with the corrected UTC, NPT, and/or SCTE 35 point data. Specific identifiers (e.g., CableLabs 3.0 Signal IDs) may also be obtained. Finally, the signal processing system generates a response notification detailing how the acquisition system might condition the video content and provide auxiliary data to be inserted for downstream usage. Additional information describing how video content may be conditioned can be found in the CableLabs Content Encoding Profiles 3.0 Specification ([CEP 3.0]).

8.1 CableLabs MD Signaling Schema

The XML schema for signal processing and conditioning SHALL be the CableLabs MD Signaling Schema version 3.0 ([CONTENT 3.0]). Originating systems SHALL use the *SignalProcessingEvent* element to send events to the processing system, while processing systems SHALL use the *SignalProcessingNotification* to represent the notification payload.

8.2 CableLabs MD Content Schema

The XML schema for content SHALL be the CableLabs MD Content Schema version 3.0 ([CONTENT 3.0]). When dealing with URI sourced material (see Section 7.7), the MovieType element SHALL be used to identify and describe the source and destination content.

8.3 Signal Confirmation Use Cases

The following use cases are meant as a guideline for defining the schemas for the event and notification elements and are not intended to be an exhaustive representation of the system usage. The schemas are designed with extensibility features that are intended to support yet-to-be-defined signaling models and formats.

8.3.1 SpliceInsert to Signal Ads

The content provider inserts an SCTE 35 splice insert marker into the stream to signal a placement opportunity for a distributor to replace a national ad with a local ad. The transcoder generates a *SignalProcessingEvent* element and sends it to the POIS endpoint. The POIS confirms whether or not the signal is valid and notifies the transcoder of the condition points within the stream with a *SignalProcessingNotification* message.

8.3.2 TimeSignal with SegmentationDescriptor to Signal an Ad

The content provider inserts an SCTE 35 segmentation descriptor into the stream to signal a placement opportunity for a distributor to replace a national ad with a local ad. The transcoder generates a *SignalProcessingEvent* element and sends it to the POIS endpoint. The POIS confirms whether or not the signal is valid and notifies the transcoder of the condition points within the stream with a *SignalProcessingNotification* message.

8.3.3 TimeSignal with SegmentationDescriptor to signal a Blackout

The content provider inserts an SCTE 35 segmentation descriptor into the stream to signal the beginning of a region that may be subject to content restrictions (i.e., blackouts). The transcoder generates a *SignalProcessingEvent* element and sends it to the POIS endpoint. The POIS confirms whether or not the signal is a valid blackout and

notifies the transcoder of the condition points within the stream and includes a directive to repeat the signal for a set amount of time using the *SignalProcessingNotification* message.

8.3.4 Normalize Signal Format

In order to provide a uniform signal to the downstream application, the POIS may be used to replace (or modify elements of) the existing signal sent in the request event.

8.3.5 Out-of-Band Notification of Blackout

The content provider notifies the distributor of a blackout region in a way other than an SCTE 35 in-band signal as intended with this specification. The POIS is instructed to send an out-of-band *SignalProcessingNotification* message to the transcoder detailing the insertion of an in-band signal to be processed by downstream systems.

8.3.6 Regions Defined Using Signal Metadata

The content provider notifies the distributor of regions of interest in a URI referenced content asset. The POIS will send *SignalProcessingNotification* messages in a synchronous or asynchronous model to the transcoder detailing the insertion of in-band signaling, with unique identifiers for the start and end of each region of interest, to be processed by downstream systems. Regions of interest may be empty; that is, the start and end point of a region of interest may point to the same position in an asset. A URI referenced content asset may not have any regions of interest. For example, there may be no advertising in an asset received by a distributor.

8.4 SignalProcessingEvent

This element is a wrapper for the *AcquiredSignal* element described below in the following sections. The wrapper is used to contain multiple signals within one event message.

The content of *AcquiredSignal*, generally an SCTE 35 descriptor, can be either fully parsed or sent as binary payload. This is expressed as a choice in the AcquiredSignal element so that intermediary systems that are not interested in the content of the event can forward it as a pass-through element using the *BinarySignalType* data type.

The following XML payload is submitted to the endpoint URI:

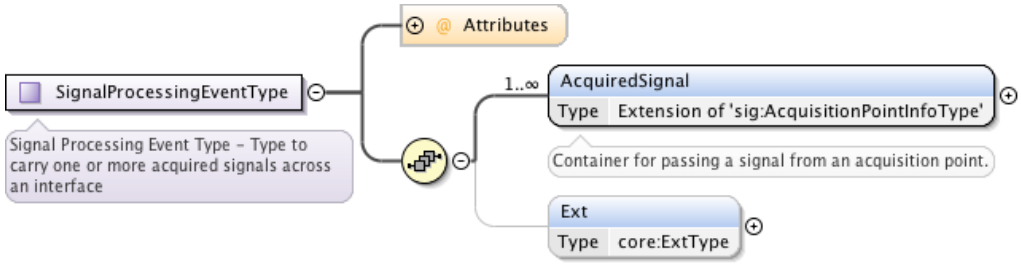

Figure 15 - SignalProcessingEvent XML Schema

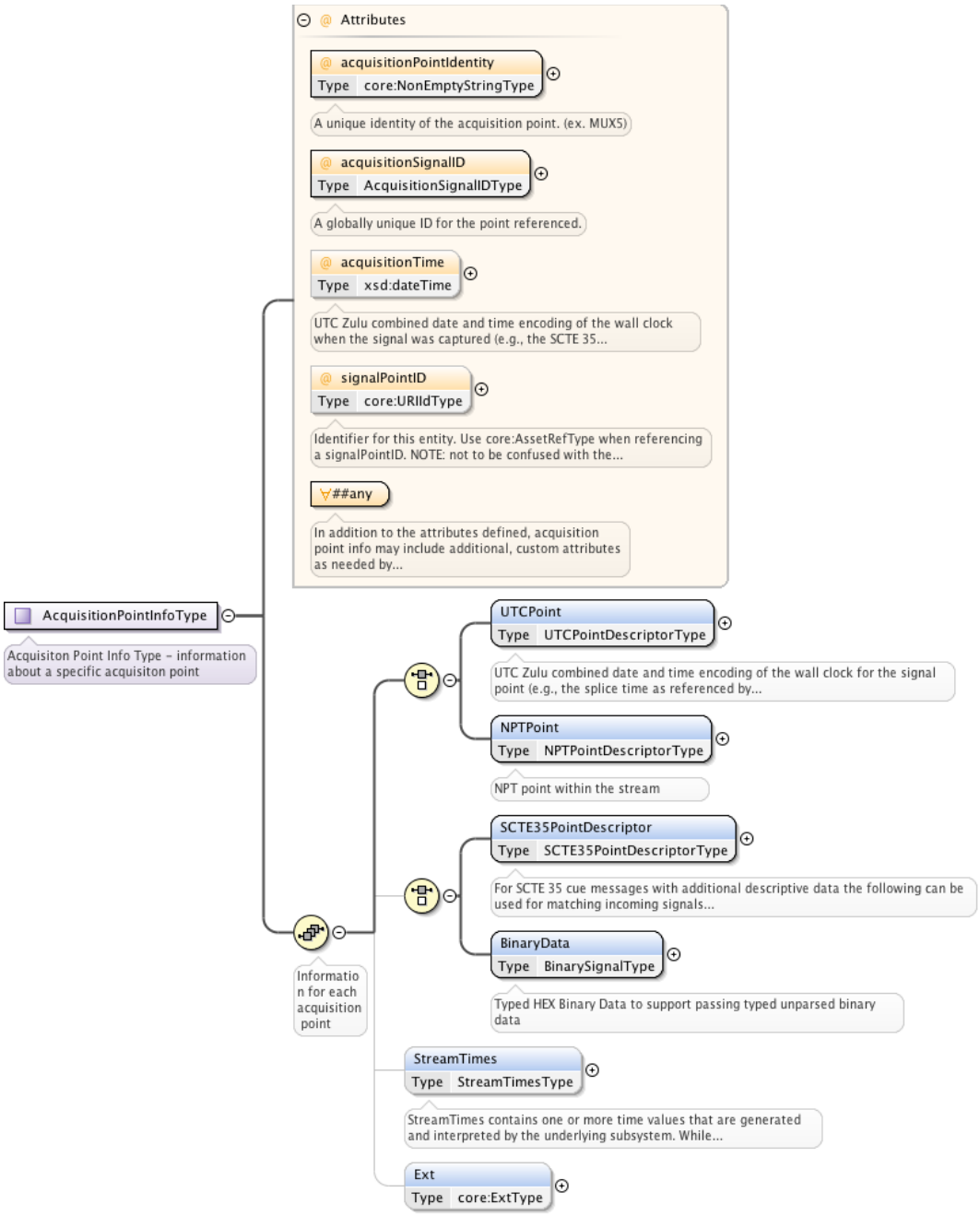

Figure 16 - AcquiredSignal XML Schema

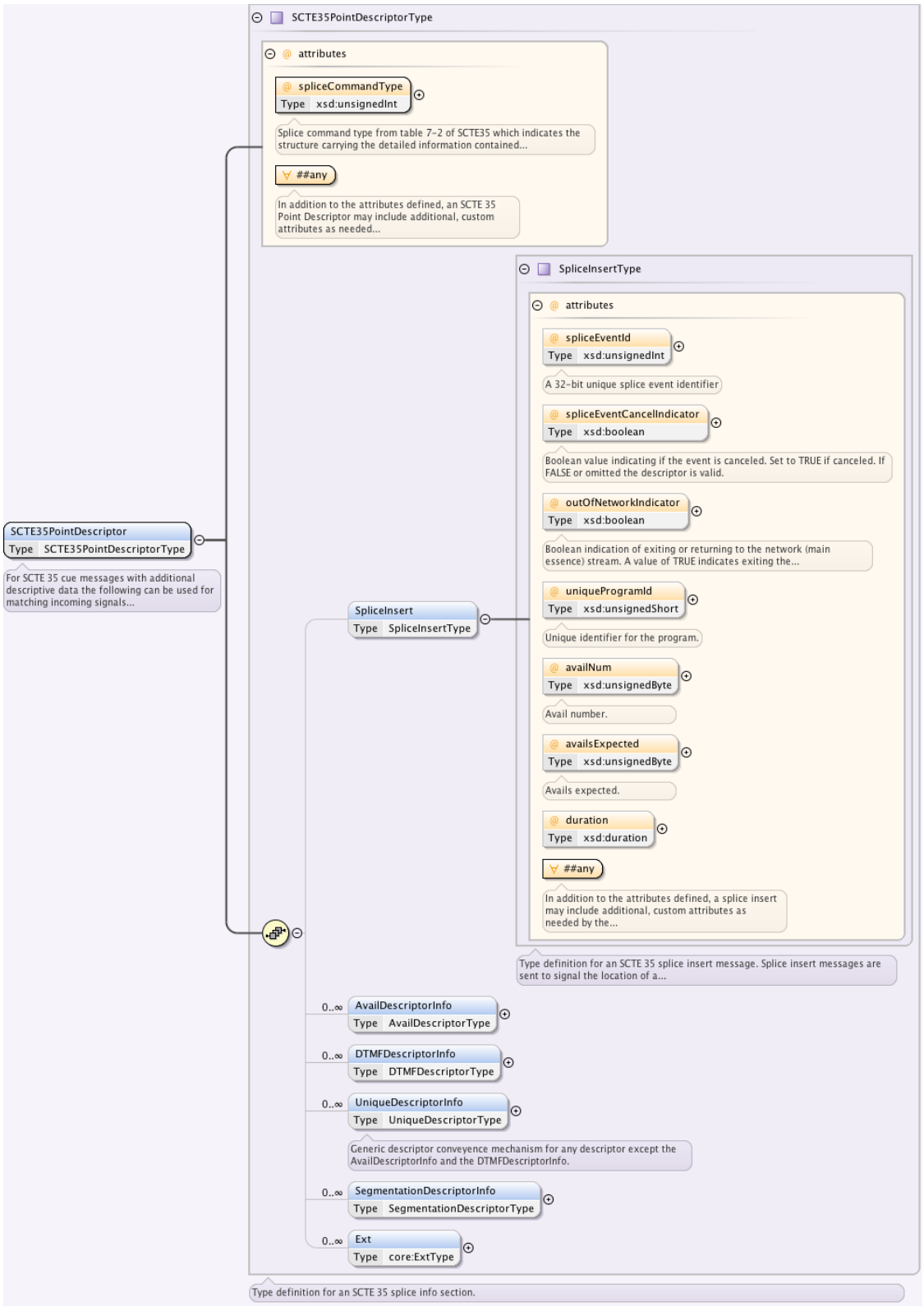

Figure 17 - SCTE35PointDescriptor XML Schema

8.4.1 SignalProcessingEvent Semantics

8.4.1.1 AcquiredSignal Element

Table 16 - AcquiredSignal Attributes

Name	Required	Description
acquisitionPointIdentity	Y	A required string, typically a system-wide unique string, identifying the transcoder at a specific site on a specific channel/network feed.
acquisitionSignalID	Y	A required string, typically a GUID, generated by the transcoder identifying the signal point being confirmed.
signalPointID	N	Optional string passed by the signal process system identifying the confirmed signal.
acquisitionTime	N	Optional UTC Zulu combined date and time encoding of the wall clock when the signal was captured (e.g., the SCTE 35 splice_info_section() acquisition time). For SCTE 35, this time value is ***not*** the splice time as referenced by a cue time or time_signal(), but rather the date and time of when the signal was received/encountered by the acquisition transcoder. See Section 6.2 for additional format information.

Table 17 - AcquiredSignal Elements

Name		Required	Description
Choice	UTCPoint	Y	Required UTC Zulu combined date and time encoding of the wall clock for the signal point (e.g., the splice time as referenced by the SCTE 35 splice_time). This UTC time value typically SHOULD NOT match the acquisitionTime value. See Section 6.2 for additional format information.
	NPTPoint	N	NPTPoint is not used with SignalProcessingEvent messages.
StreamTimes		N	StreamTimes contains one or more time values that are generated and interpreted by the underlying subsystem. While implementation-specific, this data is intended to be passed as is through intervening systems unaltered.
Choice	SCTE35PointDescriptor	N	Optional data from an SCTE 35 marker. See Section 8.4.1.2.
	BinaryData	N	Optional data from the SCTE 35 cue message (or other signal) in Base64 encoded format. See Section 8.4.1.3 for additional information.

8.4.1.2 SCTE35PointDescriptor

Table 18 - SCTE35PointDescriptor Attributes

Name	Required	Description
spliceCommandType	Y	Required splice command type from SCTE 35 table 7-2 indicating the structure carrying the detailed information contained within this object.

Table 19 - SCTE35PointDescriptor Elements

Name	Required	Description
SpliceInsert	N	See Section 8.4.1.2.1.
DTMFDescriptorInfo	N	As defined by [SCTE 35].

Name	Required	Description
SegmentationDescriptorInfo	N	As defined by [SCTE 35].
UniqueDescriptorInfo	N	As defined by [SCTE 35].
AvailDescriptorInfo	N	As defined by [SCTE 35].

8.4.1.2.1 SpliceInsert

Table 20 - SpliceInsert Attributes

Name	Required	Description
spliceEventID	N	Optional 32-bit unsigned integer from an SCTE 35 splice_insert() signal
spliceEventCancelIndicator	N	Optional Boolean to indicate if the event is canceled. If FALSE or omitted the event is valid
duration	N	Optional UTC encoded signal region (e.g., break/avail/pod) duration as indicated by the SCTE 35 break_duration()/duration field if present or an equivalent source. For example, a two-minute avail would be encoded as "PT2M". See Section 6.3 for additional format information.
uniqueProgramID	N	Optional 16-bit unsigned integer from the SCTE 35 splice_insert() signal.
availNum	N	Optional 8-bit unsigned integer from the SCTE 35 splice_insert() signal.
availsExpected	N	Optional 8-bit unsigned integer from the SCTE 35 splice_insert() signal.
outOfNetworkIndicator	N	Optional Boolean where the value of true indicates the opportunity to exit from the network feed. When set to the value of false, the signal point indicates an opportunity to return to the network feed

8.4.1.3 BinaryData

BinaryData — Contains optional sequence of Base64 Binary coded data and a string identifying the data type. See Section 6.4 for additional format information.

For SCTE 35 cue messages the signal acquisition system SHALL process the message as follows:

1. Extract the entire SCTE 35 *splice_info_section* starting at the *table_id* and ending with the *CRC_32*. (see table 7-1 in [SCTE 35]).
2. Base64 encode the value.
3. Store in the BinaryData element for transmission to the signal processing system.

8.4.2 SignalProcessingEvent Examples

To improve readability, all XML schemas in this section are assumed to have the following namespace declaration:

```
xsi:schemaLocation="urn:cablelabs:iptvservices:esam:xsd:signal:1 OC-SP-ESAM-
API-I0x-Signal.xsd"
xmlns="urn:cablelabs:iptvservices:esam:xsd:signal:1"
xmlns:sig="urn:cablelabs:md:xsd:signaling:3.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

8.4.2.1 SCTE 35 SpliceInsert (parsed)

The transcoder forwards a parsed SCTE 35 splice insert marker to the POIS using the following syntax.

```
<SignalProcessingEvent>
        <!-- acquisitionSignalID -->
        <!-- GUID: 4A6A94EE-62FA-11E1-B1CA-882F4824019B -->
        <AcquiredSignal acquisitionPointIdentity="ESPN_East_Acquisition_Point_1"
acquisitionSignalID="4A6A94EE-62FA11E1B1CA882F4824019B" acquisitionTime="2012-09-18T10:14:26Z">
```

```
                <sig:UTCPoint utcPoint="2012-09-18T10:14:34Z"/>
                <sig:SCTE35PointDescriptor spliceCommandType="5">
                        <sig:SpliceInsert spliceEventID="344568691" outOfNetworkIndicator="true"
uniqueProgramID="55355" duration="PT1M0S"/>
                </sig:SCTE35PointDescriptor>
                <sig:StreamTimes>
                        <sig:StreamTime timeType="PTS" timeValue="4452723280"/>
                </sig:StreamTimes>
        </AcquiredSignal>
</SignalProcessingEvent>
```

8.4.2.2 SCTE 35 TimeSignal (Advertisement)

The transcoder forwards a parsed event for an ad marker defined with SCTE 35 TimeSignal command with a segmentation descriptor using the following syntax:

```
<SignalProcessingEvent>
        <!-- acquisitionSignalID -->
        <!-- GUID: 4A6A94EE-62FA-11E1-B1CA-882F4824019B -->
        <AcquiredSignal acquisitionPointIdentity="ESPN_East_Acquisition_Point_1"
acquisitionSignalID="4A6A94EE-62FA11E1B1CA882F4824019B" acquisitionTime="2012-09-18T10:14:26Z">
                <sig:UTCPoint utcPoint="2012-09-18T10:14:34Z"/>
                <!-- spliceCommandType of 6 is a time signal command -->
                <sig:SCTE35PointDescriptor spliceCommandType="6">
                        <!-- Segmentation descriptor -->
                        <!-- upidType of 9 is an ADI identifier -->
                        <!-- UPID is hex encoding of SIGNAL:gFkRZZobSm+Nd7f5ccW1lA== -->
                        <!-- segmentTypeID of 50 is PlacementOpportunity Start -->
                        <sig:SegmentationDescriptorInfo segmentEventID="99790150" upidType="9"
upid="5349474e414c3a67466b525a5a6f62536d2b4e64376635636357316c413d3d" segmentTypeID="50"
segmentNum="0" segmentsExpected="0" duration="PT1M0S"/>
                </sig:SCTE35PointDescriptor>
                <sig:StreamTimes>
                        <sig:StreamTime timeType="PTS" timeValue="4452723280"/>
                </sig:StreamTimes>
        </AcquiredSignal>
</SignalProcessingEvent>
```

8.4.2.3 SCTE 35 TimeSignal (Blackout Candidate)

The transcoder forwards an event for a program start boundary defined with a parsed SCTE 35 TimeSignal with a segmentation descriptor identifying the program using the following syntax:

```
<SignalProcessingEvent>
        <!-- acquisitionSignalID -->
        <!-- GUID: 4A6A94EE-62FA-11E1-B1CA-882F4824019B -->
        <AcquiredSignal acquisitionPointIdentity="ESPN_East_Acquisition_Point_1"
acquisitionSignalID="4A6A94EE62FA11E1B1CA882F4824019B" acquisitionTime="2012-09-18T10:14:26Z">
                <sig:UTCPoint utcPoint="2012-09-18T10:14:34Z"/>
                <!-- spliceCommandType of 6 is a time signal command -->
                <sig:SCTE35PointDescriptor spliceCommandType="6">
                        <!-- Segmentation descriptor -->
                        <!-- upidType of 10 is an EIDR -->
                        <!-- UPID is hex encoding of 10.5240/DA73-37DD-11E7-9394-DE6D-U -->
                        <!-- segmentTypeID of 16 is Program Start -->
                        <sig:SegmentationDescriptorInfo segmentEventID="99790150" upidType="10"
upid="31302e353234302f444137332d333744442d313145372d393339342d444536442d55" segmentTypeID="16"
segmentNum="1" segmentsExpected="1"/>
```

```
                </sig:SCTE35PointDescriptor>
                <sig:StreamTimes>
                        <sig:StreamTime timeType="PTS" timeValue="4452723280"/>
                </sig:StreamTimes>
        </AcquiredSignal>
</SignalProcessingEvent>
```

8.4.2.4 SCTE 35 Binary

The transcoder forwards an unparsed SCTE 35 event in binary format to the POIS:

```
<SignalProcessingEvent>
        <!-- AcquisitionSignalID 168fdcf8-edbb-42df-8acb-99f4c8ae3c2e-->
        <AcquiredSignal acquisitionPointIdentity="ESPN_East_Acquisition_Point_1"
acquisitionSignalID="168fdcf8edbb42df8acb99f4c8ae3c2e">
                <!-- Time derived by splice_time in splice_insert with pts_adjustment from splice_info_section
applied -->
                <sig:UTCPoint utcPoint="2011-04-22T16:16:42.123Z"/>
                <!-- Information from the splice_insert message in Base64 derived from HEX
F00F0500028CE87FCFFED58923BF556600000000-->
                <sig:BinaryData signalType="SCTE35">
/D//AH/////////wDwUAAozof8/+1Ykjv1VmAAAAAP/////////</sig:BinaryData>
                <!-- Stream times derived by the SPE -->
                <sig:StreamTimes>
                        <sig:StreamTime TimeType="PTS" TimeValue="5022471834"/>
                </sig:StreamTimes>
        </AcquiredSignal>
</SignalProcessingEvent>
```

The following table describes the BinaryData. The section references refer to SCTE 35 2011 (see [SCTE 35]).

BinaryData:
FC3FFF007FFFFFFFFFFFFF00F0500028CE87FCFFED58923BF556600000000FFFFFFFFFFFFFFFF

Base64 encoding: /D//AH/////////wDwUAAozof8/+1Ykjv1VmAAAAAP/////////

Fields labeled ”Not processed” in the Value column are not processed by the signal processing system. These fields SHALL be present in the SignalProcessingEvent binary data from the signal acquisition system to the signal processing system. In the example, 1’s were used to fill the “Not processed” values. In actual practice those fields SHALL have values that comply with SCTE 35 2011 (see [SCTE 35]).

Table 21 - SCTE 35 Binary Data

Field	Length (bits)	Value	Notes
splice_info_section (S 7.2)			
table_id	8	Not processed	0XFC is the assigned table ID for the splice_info_section
Section_syntax_indicator	1	Not processed	Always 0
private_indicator	1	Not processed	Always 0
Reserved	2	Not processed	
section_length	12	Not processed	
protocol_version	8	Not processed	0 is the only supported value
encrypted_packet	1	Not processed	Only unencrypted data passed
encryption_algorithm	6	Not processed	Usage is not defined for unencrypted data
pts_adjustment	33	Not processed	
cw_index	8	Not processed	Usage is not defined for unencrypted data
reserved	12	Not processed	
splice_command_length	12	0x00F	15 bytes of data for splice_insert()
splice_command_type	8	0x05	splice_insert()
splice_insert (S 7.3.3)			
splice_event_id	32	0x00028CE8	Splice event identifier
splice_event_cancel_indicator	1	0	Not a cancel of a previous send cue message
reserved	7	1111111	All 1's
Out_of_network_indicator	1	1	Out point indicator
program_splice_flag	1	1	Program splice
duration_flag	1	0	No duration supplied
splice_immediate_flag	1	0	PTS supplied for outpoint
reserved	4	0xF	All 1's
Splice_time() (S 7.4.1)			
time_specified_flag	1	1	Time is supplied
reserved	6	111111	All 1's
pts_time	33	0 bit + 0xD58923BF	PTS time of out point (implementation applies pts_adjustment if present)
unique_program_id	16	5566	
avail_num	8	0x00	
avails_expected	8	0x00	
descriptor_loop_length	16	0x0000	0 (no descriptor follows)
Alignment_stuffing	N x 8	Not processed	
E_CRC_32	32	Not processed	
CRC_32	32	Not processed	

8.5 SignalProcessingNotification

Similarly to the event message structure, the notification message is also a wrapper for the SCTE 35 *AcquiredSignal* type. The notification message contains multiple acquired signals and condition points.

The SignalProcessingNotification element is used for both synchronous and asynchronous processing. For synchronous processing, the SignalProcessingNotification is sent in response to a SignalProcessingEvent. For asynchronous processing, the SignalProcessingNotification is sent in response to a UriProcessingRequest for specific content (see Section 7.7) or as a request to a processing element to initiate processing of specified content.

The notification message must explicitly define any updates to signals using the ResponseSignal element with the associated action element. The transcoder SHALL NOT be required to compose or infer a new or modified signal based on any other information (e.g., SignalPointID). Therefore, the SignalPointID must be explicitly provided as part of a new or updated signal in the notification message if it is expected to be retrieved downstream by another device. Proprietary means SHALL NOT be devised to circumvent this requirement.

If the signal acquisition system does not recognize or support the returned attribute/element or its semantics, the signal acquisition system is to ignore the data.

The following XML payload is returned:

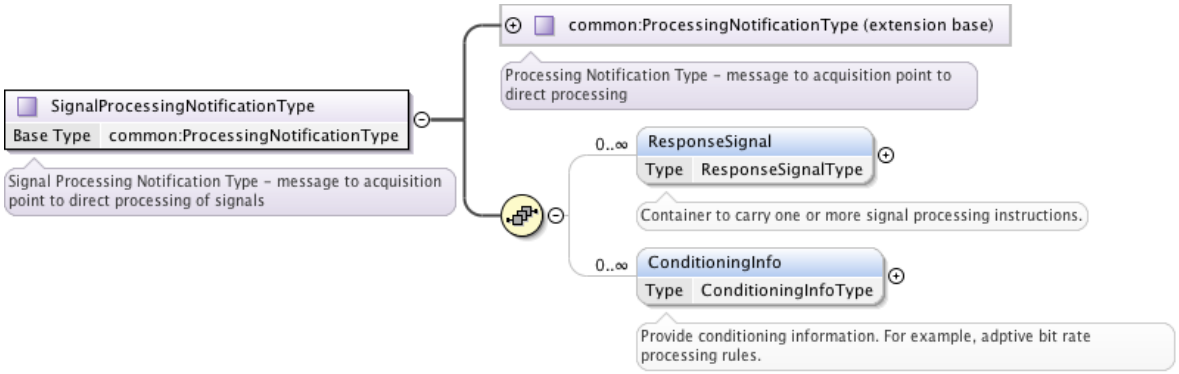

Figure 18 - SignalProcessingNotification XML Schema

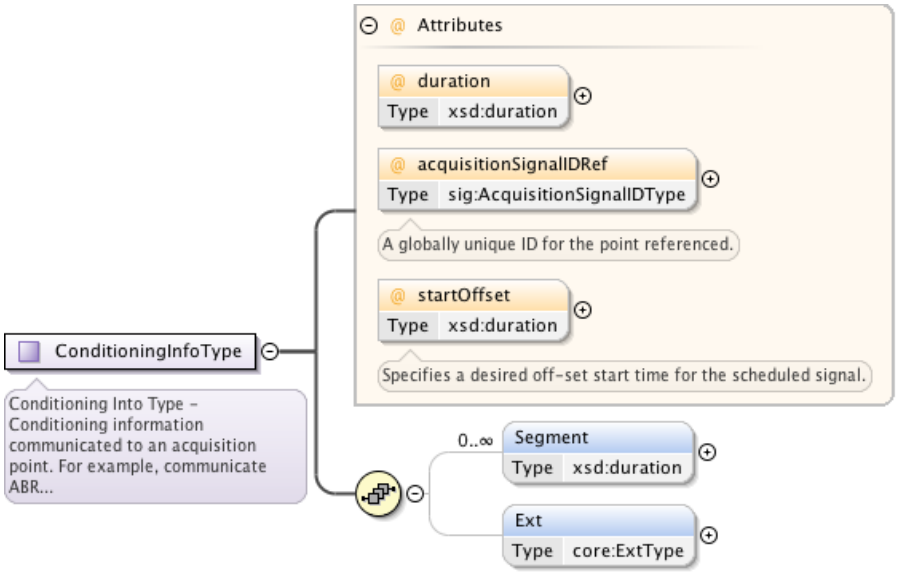

Figure 19 - ConditioningInfo XML Schema

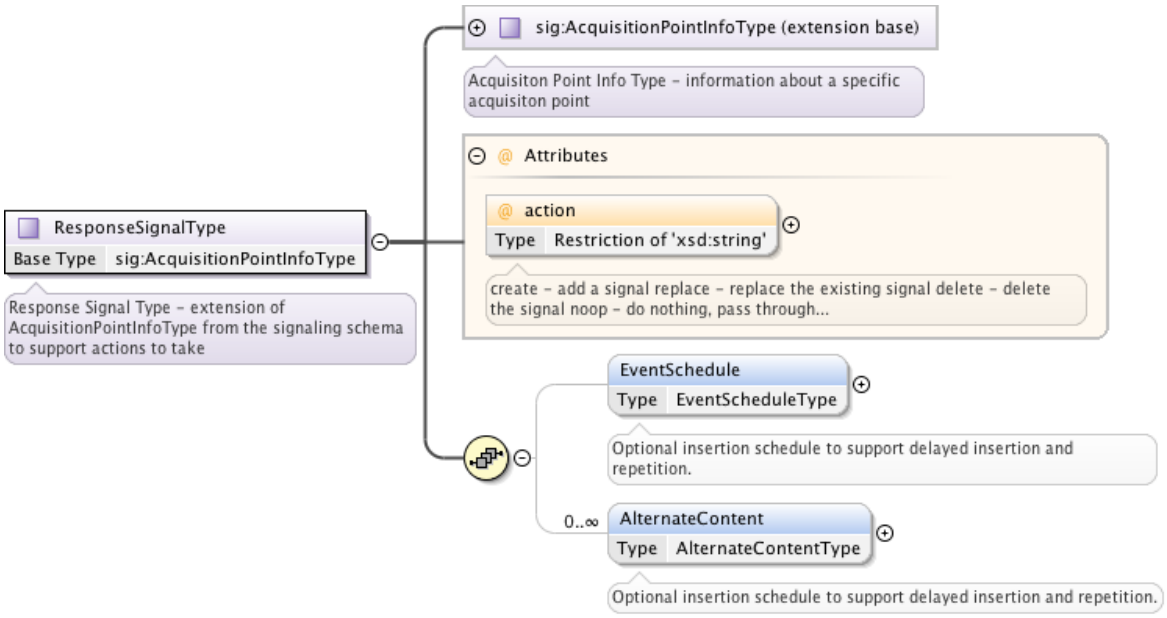

Figure 20 - ResponseSignal XML Schema

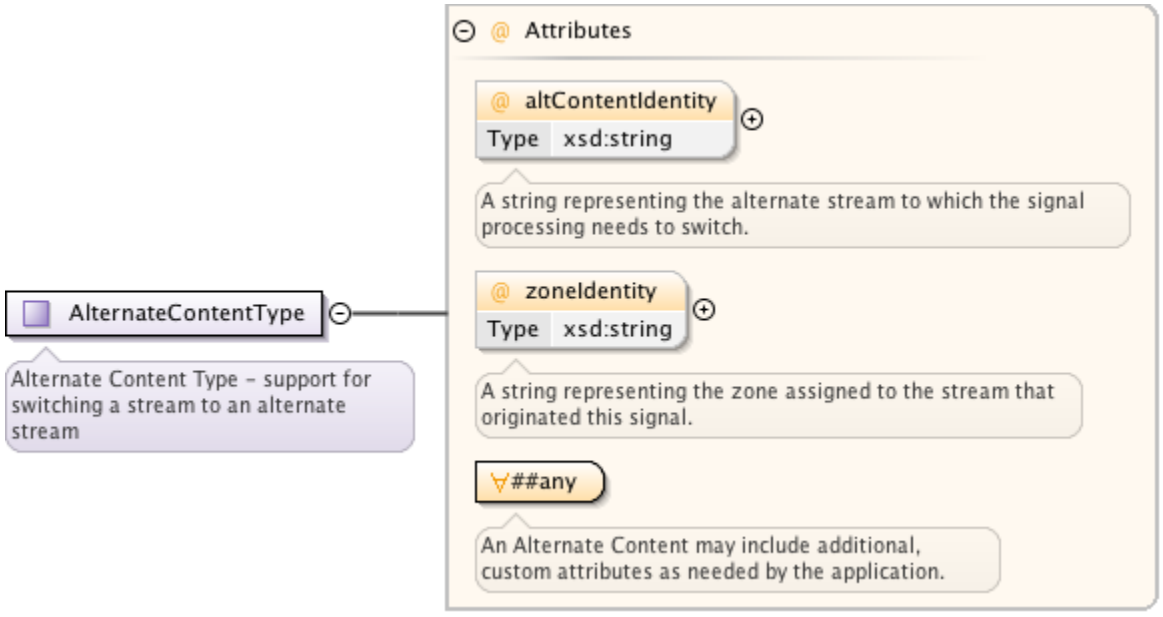

Figure 21 - AlternateContent XML Schema

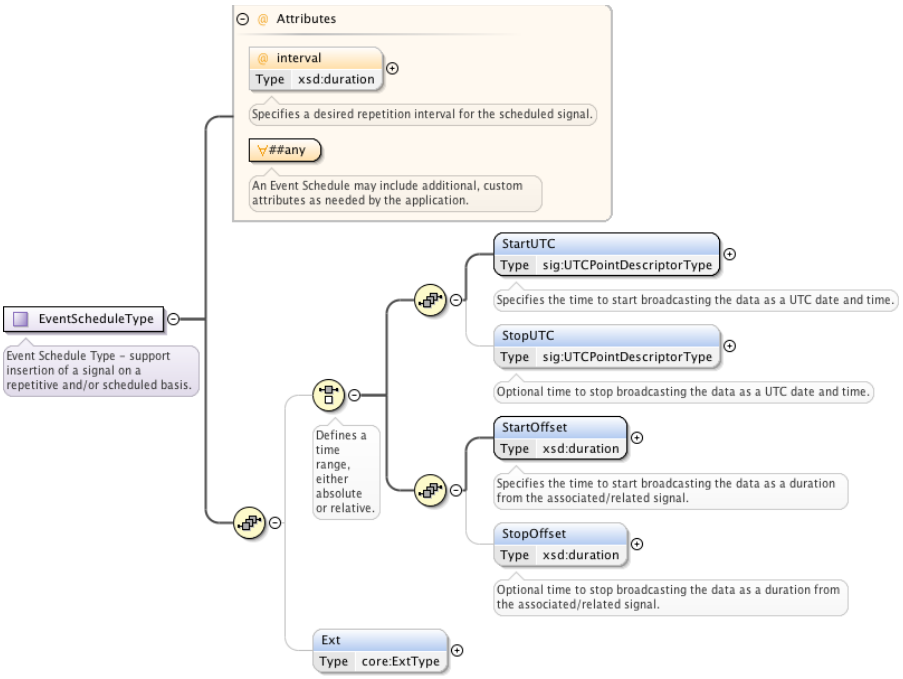

Figure 22 - EventSchedule XML Schema

8.5.1 SignalProcessingNotification Semantics

Table 22 - SignalProcessingNotification Attributes

Name	Required	Description
acquisitionPointIdentity	N	A unique identity of the acquisition point identifying the signal acquisition system (ex. transcoder) at a specific site on a specific channel/network feed for all of the contained ResponseSignal or AlternateContentEvent elements. If the acquisitionPointIdentity attribute is not present in the SignalProcessingNotification element, the target is determined by acquisitionPointIdentity attribute of each ResponseSignal or AlternateContentEvent element (and MAY be different for each).

8.5.1.1 ResponseSignal

This element extends the *AcquisitionPointInfoType (see* [CONTENT 3.0]*)* with the following components:

Table 23 - ResponseSignal Attributes

Name	Required	Description
acquisitionPointIdentity	N	A unique identity of the acquisition point identifying the signal acquisition system (ex. transcoder) at a specific site on a specific channel/network feed. If the acquisitionPointIdentity attribute is present in the SignalProcessingNotification element, then the value (if any) of this attribute SHALL be ignored. If the acquisitionPointIdentity is not present in the SignalProcessingNotification element and is also not present in the ResponseSignal element, the ResponseSignal applies to all channels/network feeds that are being processed by the signal acquisition systems that are part of a device.
acquisitionSignalID	N	Inherited from AcquisitionPointInfoType.

Name	Required	Description
action	Y	Specifies the action to be taken: **create** - add a signal **replace** - replace the existing signal **delete** - delete the signal **noop** - do nothing, pass through **private:*** - support private actions A **create** action for an existing signal SHALL cause the creation of an additional signal. Receipt of **delete** or **noop** actions for an unknown signal is invalid and MAY be ignored. Receipt of a **replace** action for an unknown signal should be interpreted as **create**.

Table 24 - ResponseSignal Elements

Name	Required	Description
EventSchedule	N	Optional element to support repetition of a signal.
AlternateContent	N	Optional insertion schedule to support delayed insertion and repetition.

For synchronous processing, the UTCPoint Element SHALL be present and specify the UTC position for the signal to create. For asynchronous processing, the NPTPoint Element SHALL be present and specify the NPT offset per [CEP 3.0] for the signal position.

If the action is create and the ResponseSignal contains an SCTE 35, the Signal Acquisition System SHOULD insert the new SCTE 35 no less than four (4) seconds prior to the indicated UTCPoint splice time per SCTE 35 practice.

8.5.1.2 Alternate Content

Table 25 - ResponseSignal Attributes

Name	Required	Description
altContentIdentity	N	Optional: A string representing the alternate stream to which the signal processing needs to switch. If altContentIdentity is an empty string or the altContentIdentity attribute is not present, the signal processing needs to switch to the default stream.
zoneIdentity	N	Optional: A string that represents the location of this stream. Examples may include: Ad Zone or VIRD.

8.5.1.3 ConditioningInfo

Table 26 - ConditioningInfo Attributes

Name	Required	Description
duration	N	An ISO Duration encoded total signal region duration, typically a break/pod/avail duration, suggesting where the transcoder should create an I/IDR frame for a seamless splice return. For example, a one-minute avail is encoded as "PT1M", a 30 second break is encoded as "PT30S", and a 30500 millisecond break would be coded as "PT30.5S". See Section 6.3 for additional format information.
acquisitionSignalIDRef	N	Optional reference to an acquisitionsignalID identifying the starting point for the conditioning info.

Name	Required	Description
StartOffset	N	An ISO Duration indicating the chronological order of the conditioning points.

Table 27 - ConditioningInfo Elements

Name	Required	Description
Segment	N	Condition points for subdividing the break into smaller units (usually used for HLS support). Total duration of segments SHALL equal the duration provided for the condition point.

8.5.1.4 EventSchedule

Table 28 - EventScheduleType Attributes

Name	Required	Description
Interval	N	Optional repetition interval for a signal.

EventSchedule supports a choice of two scheduling models:

- Absolute start and end UTC times
- Relative (to signal time) start and end offsets

Table 29 - EventScheduleType Elements

Name	Required	Description
StartUTC	Y	SHALL be specified if using absolute timing.
EndUTC	N	Optional end time if StartUTC is specified.
StartOffset	Y	SHALL be specified if using relative timing.
EndOffset	N	Optional end offset if StartOffset is specified.

8.5.2 *SignalProcessingNotification* Examples

To improve readability, all XML schemas in this section are assumed to have the following namespace declaration:

```
xsi:schemaLocation="urn:cablelabs:iptvservices:esam:xsd:signal:1 OC-SP-ESAM-
API-I0x-Signal.xsd"
xmlns="urn:cablelabs:iptvservices:esam:xsd:signal:1"
xmlns:sig="urn:cablelabs:md:xsd:signaling:3.0"
xmlns:core="urn:cablelabs:md:xsd:core:3.0"
xmlns:content="urn:cablelabs:md:xsd:content:3.0"
xmlns:common="urn:cablelabs:iptvservices:esam:xsd:common:1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

8.5.2.1 Out-of-Band Notification

The POIS notifies the transcoder to insert a signal indicating a program start with the additional instructions to repeat every 5 seconds for the next two hours. This notification message is not generated as a response to an event sent by the transcoder.

```
<SignalProcessingNotification acquisitionPointIdentity="ESPN_East_Acquisition_Point_1">
        <!-- Insert a program start boundary -->
        <!-- GUID: 4A6A94EE-62FA-11E1-B1CA-882F4824019B -->
```

```
        <ResponseSignal action="create" acquisitionPointIdentity="ESPN_East_Acquisition_Point_1"
acquisitionSignalID="4A6A94EE62FA11E1B1CA882F4824019B">
                <sig:UTCPoint utcPoint="2012-09-18T10:00:00Z"/>
                <!-- Time Signal Command -->
                <sig:SCTE35PointDescriptor spliceCommandType="6">
                        <!-- Segmentation descriptor -->
                        <!-- upidType of 10 is an EIDR -->
                        <!-- UPID is hex encoding of 10.5240/DA73-37DD-11E7-9394-DE6D-U -->
                        <!-- segmentTypeID of 16 is Program Start -->
                        <sig:SegmentationDescriptorInfo segmentEventID="99790150" upidType="10"
upid="31302e353234302f444137332d333744442d313145372d393339342d444536442d55" segmentTypeID="16"
segmentNum="1" segmentsExpected="1"/>
                </sig:SCTE35PointDescriptor>
        </ResponseSignal>
        <!-- Insert repeating program identification signals -->
        <!-- GUID: 11b98ada-7cc0-4152-bc5c-1d2ab2909210 -->
        <ResponseSignal action="create" acquisitionPointIdentity="ESPN_East_Acquisition_Point_1"
acquisitionSignalID="11b98ada7cc04152bc5c1d2ab2909210">
                <sig:UTCPoint utcPoint="2012-09-18T10:00:05Z"/>
                <!-- Time Signal Command -->
                <sig:SCTE35PointDescriptor spliceCommandType="6">
                        <!-- Segmentation descriptor -->
                        <!-- upidType of 10 is an EIDR -->
                        <!-- UPID is hex encoding of 10.5240/DA73-37DD-11E7-9394-DE6D-U -->
                        <!-- segmentTypeID of 1 is Content Identification -->
                        <sig:SegmentationDescriptorInfo segmentEventID="99790150" upidType="10"
upid="31302e353234302f444137332d333744442d313145372d393339342d444536442d55" segmentTypeID="1"
segmentNum="0" segmentsExpected="0"/>
                </sig:SCTE35PointDescriptor>
                <!-- Insert the signal every 5 seconds for the scheduled 2 hour duration of the program -->
                <EventSchedule interval="PT5S">
                        <StartUTC utcPoint="2012-09-18T10:00:05Z"/>
                        <StopUTC utcPoint="2012-09-18T12:00:00Z"/>
                </EventSchedule>
        </ResponseSignal>

</SignalProcessingNotification>
```

8.5.2.2 Replace existing signal

The POIS notifies the transcoder that the signal identified by the acquisitionSignalID has to be replaced with the one included in the notification payload. The StatusCode optional element MAY include additional information about the directive.

```
<SignalProcessingNotification>
        <!-- acquisitionSignalID - replace the splice insert message-->
        <!-- GUID: 4A6A94EE-62FA-11E1-B1CA-882F4824019B -->
        <!-- provide the signal point ID  -->
        <!-- GUID: 805911659a1b4a6f8d77b7f971c5b594== -->
        <!-- Base64 representation: gFkRZZobSm+Nd7f5ccW1lA -->
        <common:StatusCode classCode="0">
                <core:Note>insert normalized signal</core:Note>
        </common:StatusCode>
        <ResponseSignal action="replace" acquisitionPointIdentity="ESPN_East_Acquisition_Point_1"
acquisitionSignalID="4A6A94EE62FA11E1B1CA882F4824019B" acquisitionTime="2006-05-04T18:13:51.0Z"
signalPointID="gFkRZZobSm+Nd7f5ccW1lA==">
                <sig:UTCPoint utcPoint="2006-05-04T18:13:51.0Z"/>
```

```
                <!-- Time Signal Command -->
                <sig:SCTE35PointDescriptor spliceCommandType="06">
                        <!-- Attach a signal ID to the out point -->
                        <sig:SegmentationDescriptorInfo segmentTypeId="52" segmentNum="0"
segmentsExpected="0" upidType="9"
upid="5349474e414c3a67466b525a5a6f62536d2b4e64376635636357316c41"/>
                        <!-- UPID is hex encoding of SIGNAL:gFkRZZobSm+Nd7f5ccW1lA -->
                </sig:SCTE35PointDescriptor>
                <sig:StreamTimes>
                        <sig:StreamTime timeType="PTS" timeValue="4452723280"/>
                </sig:StreamTimes>
        </ResponseSignal>
        <ConditioningInfo duration="PT1M30S"/>
</SignalProcessingNotification>
```

8.5.2.3 Delete current signal

The POIS instructs the transcoder to delete the current signal because it was unable to confirm the validity.

```
<SignalProcessingNotification>
        <!-- acquisitionSignalID - delete the signal -->
        <!-- GUID: 4A6A94EE-62FA-11E1-B1CA-882F4824019B -->
        <common:StatusCode classCode="0">
                <core:Note>unconfirmed signal</core:Note>
        </common:StatusCode>
        <ResponseSignal action="delete" acquisitionPointIdentity="ESPN_East_Acquisition_Point_1"
acquisitionSignalID="4A6A94EE62FA11E1B1CA882F4824019B">
                <sig:UTCPoint utcPoint="2006-05-04T18:13:51.0Z"/>
        </ResponseSignal>

</SignalProcessingNotification>
```

8.5.2.4 Cancel existing signal

As described in Section 8.3.3, the POIS MAY instruct the transcoder to repeat a signal every 5 seconds for an extended duration. This is commonly used during blackout scenarios, but may have other applications as well. When blackout alternative content is no longer required, the POIS sends a cancel to the transcoder ending the signal repetition. To cancel a previously confirmed content identification signal, both the signal acquisition system and the signal processing system need to reference the acquisitionSignalId of the previously confirmed signal. In other words, the systems need to be aware of the state of each signal that has been confirmed.

The following use case replaces the previously repetitive content identification ResponseSignal with a one-time content identification with a cancel indicator. In this use case, any downstream system receiving the signal resulting from the updated ResponseSignal knows that the content identification has ended based on explicit signaling.

```
<SignalProcessingNotification acquisitionPointIdentity="ESPN_East_Acquisition_Point_1">
        <!-- Cancel previously inserted repeating program identification signals by replacing the previously sent
ResponseSignal -->
        <!-- GUID: 11b98ada-7cc0-4152-bc5c-1d2ab2909210 -->
        <ResponseSignal action="replace" acquisitionPointIdentity="ESPN_East_Acquisition_Point_1"
acquisitionSignalID="11b98ada7cc04152bc5c1d2ab2909210">
                <sig:UTCPoint utcPoint="2012-09-18T10:00:05Z"/>
                <!-- Time Signal Command -->
                <sig:SCTE35PointDescriptor spliceCommandType="6">
                        <!-- Segmentation descriptor -->
                        <!-- upidType of 10 is an EIDR -->
                        <!-- UPID is hex encoding of 10.5240/DA73-37DD-11E7-9394-DE6D-U -->
                        <!-- segmentTypeID of 1 is Content Identification -->
```

```
                    <sig:SegmentationDescriptorInfo segmentEventID="99790150"
segmentationEventCancelIndicator="true" upidType="10"
upid="31302e353234302f444137332d333744442d313145372d393339342d444536442d55" segmentTypeID="1"
segmentNum="0" segmentsExpected="0"/>
                </sig:SCTE35PointDescriptor>
        </ResponseSignal>
</SignalProcessingNotification>
```

An alternative implementation would be to send a delete referencing the acquisitionSignalID of the ResponseSignal. Unlike the previous use case where there is explicit signaling, the downstream implementation would need to detect loss of content identification signaling and behave accordingly.

```
<SignalProcessingNotification acquisitionPointIdentity="ESPN_East_Acquisition_Point_1">
        <!-- Cancel previous inserted repeating program identification signals -->
        <!-- GUID: 11b98ada-7cc0-4152-bc5c-1d2ab2909210 -->
        <ResponseSignal action="delete" acquisitionPointIdentity="ESPN_East_Acquisition_Point_1"
acquisitionSignalID="11b98ada-7cc04152bc5c1d2ab2909210">
                <sig:UTCPoint utcPoint="2012-09-18T10:00:05Z"/>
        </ResponseSignal>
</SignalProcessingNotification>
```

Other alternatives that could be applied are to send a segmentation type id of "Program End" (0x11) or "Program Breakaway" (0x13). See [SCTE 67] for additional program centric use cases.

8.5.2.5 Condition Stream for Downstream ABR Insertion

The POIS provides the transcoder with duration information for a multi-spot break. The break below is 45 seconds long, composed of two spots of duration 30 and 15 seconds. The first segment starts at PTS=4452723280 and the second follows immediately after the first (this would be PTS 4455423280 with corresponding UTCPoint 2006-05-04T18:14:26.0Z).

The 15-second spot is further specified to be composed of two segments. The segmentation of the 30 second spot is unspecified (and presumably determined but system configurations or other out-of-band agreements).

```
<SignalProcessingNotification>
        <ResponseSignal action="create" acquisitionPointIdentity="ESPN_East_Acquisition_Point_1"
acquisitionSignalID="SmqU7mL6EeGxyogvSCQBmw==" acquisitionTime="2006-05-04T18:13:51.0Z"
signalPointID="gFkRZZobSm+Nd7f5ccW1lA==">
                <sig:UTCPoint utcPoint="2006-05-04T18:13:56.0Z"/>
                <sig:StreamTimes>
                        <sig:StreamTime timeType="PTS" timeValue="4452723280"/>
                </sig:StreamTimes>
        </ResponseSignal>
        <ConditioningInfo acquisitionSignalIDRef="SmqU7mL6EeGxyogvSCQBmw==" duration="PT30S"/>
        <ConditioningInfo acquisitionSignalIDRef="SmqU7mL6EeGxyogvSCQBmw==" duration="PT15S"
startOffset="PT30S">
          <Segment>PT10S</Segment>
          <Segment>PT5S</Segment>
        </ConditioningInfo>
</SignalProcessingNotification>
```

8.5.2.6 Batch Conditioning Request for Downstream ABR Insertion

The POIS provides the transcoder with NPT positional data for pre-roll and mid-roll insertion opportunities for a source file. The pre-roll is delineated by 2 signalPointID values, both with an NPT value of “BOS”. The mid-roll is delineated by 2 signalPointID values for the NPT range 732.537 to 792.537.

This message can be sent from the POIS to the transcoder or sent in response to a UriProcessingRequest (see Section 7.7.2.1).

```
<?xml version="1.0" encoding="UTF-8"?>
<SignalProcessingNotification acquisitionPointIdentity="VOD Processing_Point_1">
        <!-- Batch information -->
        <!-- TERSE DATA - MovieType allows lots of data about the asset -->
        <common:BatchInfo batchId="27b777c1ee9941479677b56203953300">
                <common:Source xsi:type="content:MovieType"
uriId="provider.com/Asset/UNVA2001081701004002">
                        <content:SourceUrl>The_Titanic.mpg</content:SourceUrl>
                </common:Source>
                <!-- Destination data can also be applied -->
        </common:BatchInfo>
        <!-- start of first region of interest - starts at BOS (NPT 0) -->
        <ResponseSignal acquisitionPointIdentity="mso.com/VODProcessingPt/1"
acquisitionSignalID="s/AhUqctQje0RpuvHAuvNQ==" signalPointID="OjwgkQp4RwmdJc2u46vrmw=="
action="create">
                <sig:NPTPoint nptPoint="BOS"/>
                <sig:SCTE35PointDescriptor spliceCommandType="06">
                        <!-- Attach a signal ID to the out point -->
                        <sig:SegmentationDescriptorInfo segmentTypeId="52" segmentNum="0"
segmentsExpected="0" upidType="9"
upid="5349474E414C3A4F6A77676B51703452776D644A633275343676726D773D3D"/>
                        <!-- UPID is hex encoding of SIGNAL:OjwgkQp4RwmdJc2u46vrmw== -->
                </sig:SCTE35PointDescriptor>
        </ResponseSignal>
        <!-- end of first region of interest - ends at BOS (NPT 0) -->
        <ResponseSignal acquisitionPointIdentity="mso.com/VODProcessingPt/1"
acquisitionSignalID="UR01sqCVSgydxKhzbQizyQ==" signalPointID="mLSVqLc5RqKJGVAZ1LevaQ=="
action="create">
                <sig:NPTPoint nptPoint="BOS"/>
                <sig:SCTE35PointDescriptor spliceCommandType="06">
                        <!-- Attach a signal ID to the in point -->
                        <sig:SegmentationDescriptorInfo segmentTypeId="53" segmentNum="0"
segmentsExpected="0" upidType="9"
upid="5349474E414C3A6D4C5356714C633552714B4A4756415A314C657661513D3D"/>
                        <!-- UPID is hex encoding of SIGNAL:mLSVqLc5RqKJGVAZ1LevaQ== -->
                </sig:SCTE35PointDescriptor>
        </ResponseSignal>
        <!-- start of second region of interest - starts at NPT 732.537-->
        <ResponseSignal acquisitionPointIdentity="mso.com/VODProcessingPt/1"
acquisitionSignalID="4CFXb/7CRl6Q/8j6LyA8fw==" signalPointID="Y8o0D3zpTxS0LT1ew+wuiw=="
action="create">
                <sig:NPTPoint nptPoint="732.537"/>
                <sig:SCTE35PointDescriptor spliceCommandType="06">
                        <!-- Attach a signal ID to the in point -->
                        <sig:SegmentationDescriptorInfo segmentTypeId="52" segmentNum="0"
segmentsExpected="0" upidType="9"
upid="5349474E414C3A59386F3044337A70547853304C543165772B777569773D3D"/>
                        <!-- UPID is hex encoding of SIGNAL:Y8o0D3zpTxS0LT1ew+wuiw== -->
                </sig:SCTE35PointDescriptor>
        </ResponseSignal>
        <!-- end of second region of interest - starts at NPT 792.537-->
        <ResponseSignal acquisitionPointIdentity="mso.com/VODProcessingPt/1"
acquisitionSignalID="gIho/mMRTFefPKEOkDdP9w==" signalPointID="6Wiqv1cCQumJctagOKS+ug=="
action="create">
                <sig:NPTPoint nptPoint="792.537"/>
                <sig:SCTE35PointDescriptor spliceCommandType="06">
```

```
                        <!-- Attach a signal ID to the in point -->
                        <sig:SegmentationDescriptorInfo segmentTypeId="53" segmentNum="0"
segmentsExpected="0" upidType="9"
upid="5349474E414C3A3657697176316343517 56D4A637461674F4B532B75673D3D"/>
                        <!-- UPID is hex encoding of SIGNAL:6Wiqv1cCQumJctagOKS+ug== -->
                </sig:SCTE35PointDescriptor>
        </ResponseSignal>
        <ConditioningInfo startOffset="PT0S" acquisitionSignalIDRef="s/AhUqctQje0RpuvHAuvNQ=="
duration="PT0S">
                <Segment>PT0S</Segment>
        </ConditioningInfo>
        <ConditioningInfo startOffset="PT732.537S" acquisitionSignalIDRef="4CFXb/7CRl6Q/8j6LyA8fw=="
duration="PT60S">
                <Segment>PT60S</Segment>
        </ConditioningInfo>
</SignalProcessingNotification>
```

8.5.2.7 Alternate Content Response

```
<SignalProcessingNotification>
        <ResponseSignal
        action="create"
        acquisitionPointIdentity="ESPN_East_HD"
        acquisitionSignalID="4A6A94EE-62FA-11E1-B1CA-882F4824019B"
        signalPointID="10D65A82-631E-11E1-B5B7-7A5A4824019B">
                <sig:UTCPoint utcPoint="2013-07-06T10:14:34Z" />
                <AlternateContent zoneIdentity="ESPN_East_HD_Z01"
                altContentIdentity="ESPN_SLATE" />
        </ResponseSignal>
</SignalProcessingNotification>
```

9 MANIFEST CONFIRMATION AND CONDITIONING API

The manifest confirmation and conditioning API is for use between an ABR fragmenter and the manifest confirmation and conditioning service. The fragmenter provides a manifest confirmation by sending an HTTP POST request with the defined JSON or XML payload and it receives conditioning information in the response's payload.

The manifest confirmation input parameters provide as much information as possible regarding the signal point triggering the confirmation. An example signal point may be a splice out/exit point as indicated by an SCTE 35 splice_info_section()/splice_insert() command or it could be an SCTE 35 segmentation_descriptor().

The return payload provides information about how the fragmenter might condition the ABR stream and associated manifest files, and it provides auxiliary data to be inserted for downstream signal point identification.

The manifest API supports two different forms of HLS manifest conditioning: media segment modification mode (segment modify) and media segment replacement mode (segment replace). The fragmenter submits the same input payload to an endpoint URL and receives the appropriate response.

Media segment modification mode, i.e., segment modify, describes the output manifest changes when only a subset of the HLS media segments are enhanced.

Media segment replacement mode, i.e., segment replace, indicates the API returned a set of media segments that are to directly replace an existing region of media segments.

Each mode has its positives and negatives and the selection and usage are outside the scope of this document.

If the caller does not support a returned attribute's usage, the attribute is to be ignored. For example, if the caller does not support the Microsoft Smooth sparse track/fragment insertion, the attribute is ignored.

9.1 Manifest Conditioning Use Cases

The following use cases are meant as a guideline for defining the schemas for the event and notification elements and are not intended to be an exhaustive representation of the system usage. The schemas are designed with extensibility features that are intended to support yet-to-be-defined signaling models and formats.

9.1.1 Smooth conditioning for ads insertion

The fragmenter generates a *ManifestConfirmConditionEvent* element containing one or more confirmed signals and forwards it to the POIS endpoint. The POIS responds with a *ManifestConfirmConditionNotification* message, which carries the payload to be inserted into the appropriate sparse tracks.

9.1.2 HLS conditioning for ads insertion

The fragmenter generates a *ManifestConfirmConditionEvent* element containing one or more confirmed signals and forwards it to the POIS endpoint. The POIS responds with a *ManifestConfirmConditionNotification* message, which carries the instructions on how to manipulate the appropriate segments with the HLS manifest.

9.1.3 Conditioning for blackouts

The fragmenter generates a *ManifestConfirmConditionEvent* element containing one or more confirmed signals and forwards it to the POIS endpoint. The POIS responds with a *ManifestConfirmConditionNotification* message, which carries security metadata and optionally a payload to be inserted into the appropriate sparse tracks.

9.1.4 Support multi-format conditioning

The fragmenter generates a ManifestConfirmConditionEvent element containing one or more confirmed signals and in the StreamTime element indicates the required format. The POIS responds with a ManifestConfirmConditionNotification message, which carries manifest conditioning instructions for all the requested formats.

9.1.5 Support VOD processing

The fragmenter can generate a ManifestConfirmConditionEvent which contains all of the confirmed signals that may be present in a VOD asset. When creating the AcquiredSignals, the fragmenter can convert the UTC time to NPT time relative to the start of the file. The POIS responds with a ManifestConfirmConditionNotification message, which carries manifest conditioning instructions for all the requested formats. Optionally, the ManifestConfirmConditionNotification message may include a BatchInfo element, which can be referenced during the processing of the Manifest updates.

9.2 ManifestConfirmConditionEvent

The following XML payload is submitted to the endpoint URI:

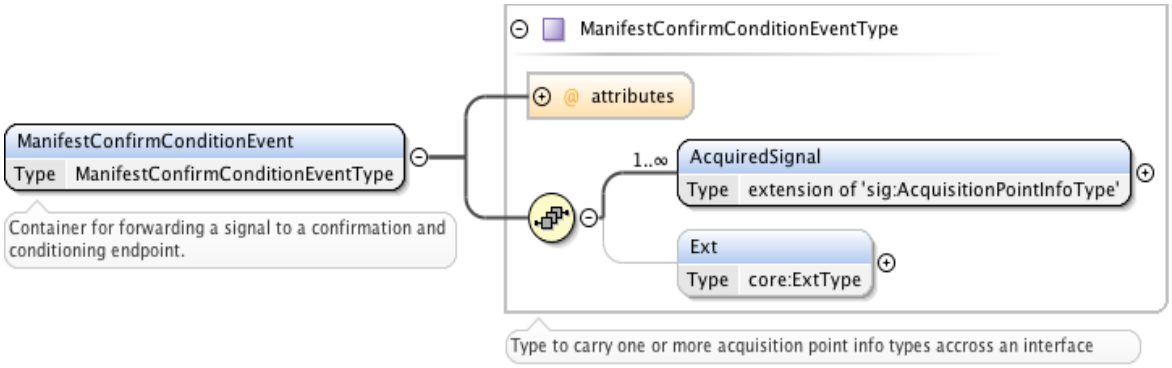

Figure 23 - ManifestConfirmConditionEvent XML Schema

9.2.1 *ManifestConfirmConditionEvent* Semantics

Semantics for the manifest confirmation and condition events are the same as acquired signal. If necessary, ManifestConfirmConditionEventType can be extended to include additional attributes and elements.

9.2.2 *ManifestConfirmConditionEvent* Examples

To improve readability, all XML schemas in this section are assumed to have the following namespace declaration:

```
xmlns:core="urn:cablelabs:md:xsd:core:3.0"
xmlns:sig="urn:cablelabs:md:xsd:signaling:3.0"
xmlns= "urn:cablelabs:iptvservices:esam:xsd:manifest:1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

9.2.2.1 SCTE 35 SpliceInsert

The fragmenter forwards an SCTE 35 splice insert marker to the POIS using the following syntax.

```
<ManifestConfirmConditionEvent>
    <AcquiredSignal
        acquisitionPointIdentity="ESPN_East_Acquisition_Point_1"
        acquisitionSignalID="4A6A94EE-62FA-11E1-B1CA-882F4824019B"
        acquisitionTime="2013-11-19T10:14:34Z">
        <sig:UTCPoint utcPoint="2012-09-18T10:14:34Z" />
        <sig:SCTE35PointDescriptor spliceCommandType="05">
            <sig:SpliceInsert
                spliceEventID="344568691"
                outOfNetworkIndicator="true"
                uniqueProgramID="55355"
                duration="PT1M0S"/>
        </sig:SCTE35PointDescriptor>
        <sig:StreamTimes>
            <sig:StreamTime timeType="PTS" timeValue="4452723280" />
```

```
            <sig:StreamTime timeType="HLS" timeValue="4452723280" />
        </sig:StreamTimes>
    </AcquiredSignal>
</ManifestConfirmConditionEvent>
```

9.2.2.2 SCTE 35 TimeSignal

The fragmenter forwards an event for an ad marker defined with SCTE 35 TimeSignal using the following syntax:

```
<ManifestConfirmConditionEvent>
    <AcquiredSignal
        acquisitionPointIdentity="ESPN_East_Acquisition_Point_1"
        acquisitionSignalID="4A6A94EE-62FA-11E1-B1CA-882F4824019B"
        acquisitionTime="2013-11-19T10:14:34Z">
        <sig:UTCPoint utcPoint="2013-07-06T10:14:34Z" />
        <sig:SCTE35PointDescriptor spliceCommandType="06">
            <sig:SegmentationDescriptorInfo
                segmentEventID="99790150"
                upidType="9"
                upid="784B7944684C70786779455764446950596A78"
                segmentTypeID="50"
                segmentNum="0"
                segmentsExpected="0"
                duration="PT1M0S"/>
        </sig:SCTE35PointDescriptor>
        <sig:StreamTimes>
            <sig:StreamTime timeType="PTS" timeValue="4452723280" />
            <sig:StreamTime timeType="HSS" timeValue="4452723280" />
        </sig:StreamTimes>
    </AcquiredSignal>
</ManifestConfirmConditionEvent>
```

9.2.2.3 SCTE 35 Binary

The fragmenter forwards an unparsed SCTE 35 event in binary format to the POIS:

```
<ManifestConfirmConditionEvent>
    <AcquiredSignal
        acquisitionPointIdentity="ESPN_East_Acquisition_Point_1"
        acquisitionSignalID="4A6A94EE-62FA-11E1-B1CA-882F4824019B"
        acquisitionTime="2013-11-19T10:14:34Z">
        <sig:UTCPoint utcPoint="2013-07-06T10:14:34Z" />
        <sig:BinaryData signalType="SCTE35">
             cEJwQUthSWpMdmhic1Jl
        </sig:BinaryData>
        <sig:StreamTimes>
            <sig:StreamTime timeType="PTS" timeValue="4452723280" />
            <sig:StreamTime timeType="HDS" timeValue="4452723280" />
        </sig:StreamTimes>
    </AcquiredSignal>
</ManifestConfirmConditionEvent>
```

9.3 ManifestConfirmConditionNotification

The following response XML payload is returned:

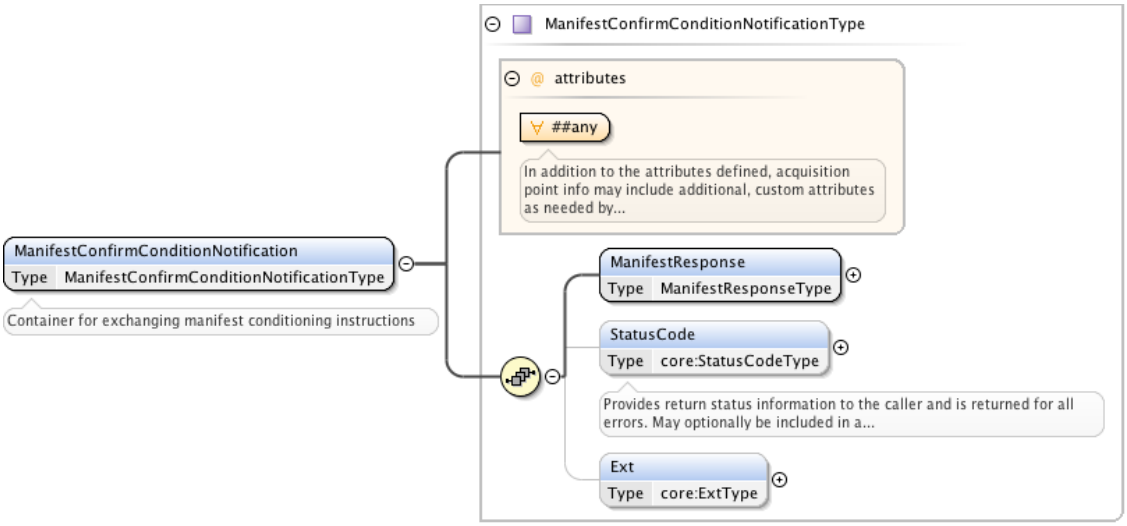

Figure 24 - ManifestConfirmConditionNotification XML Schema

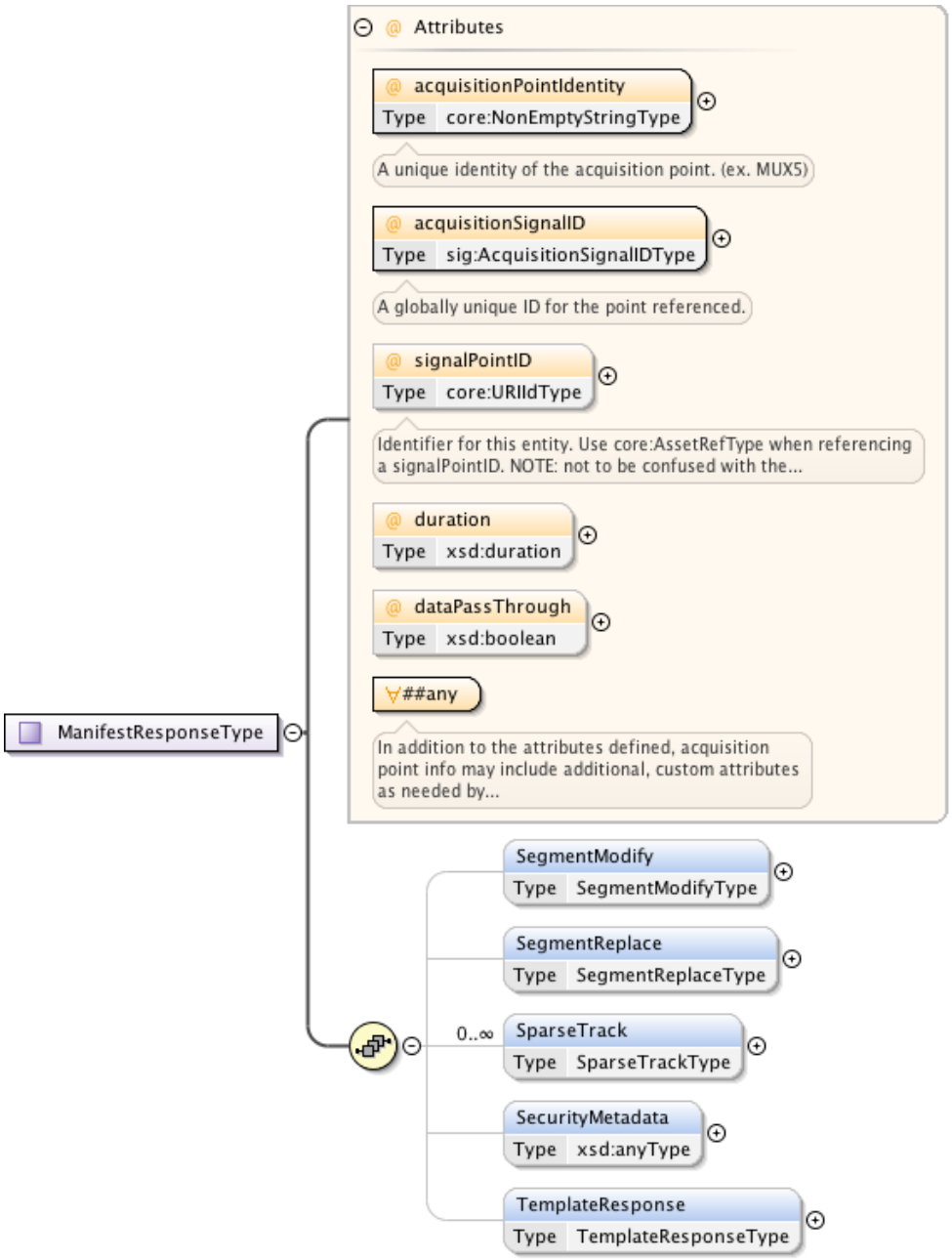

Figure 25 - ManifestResponse XML Schema

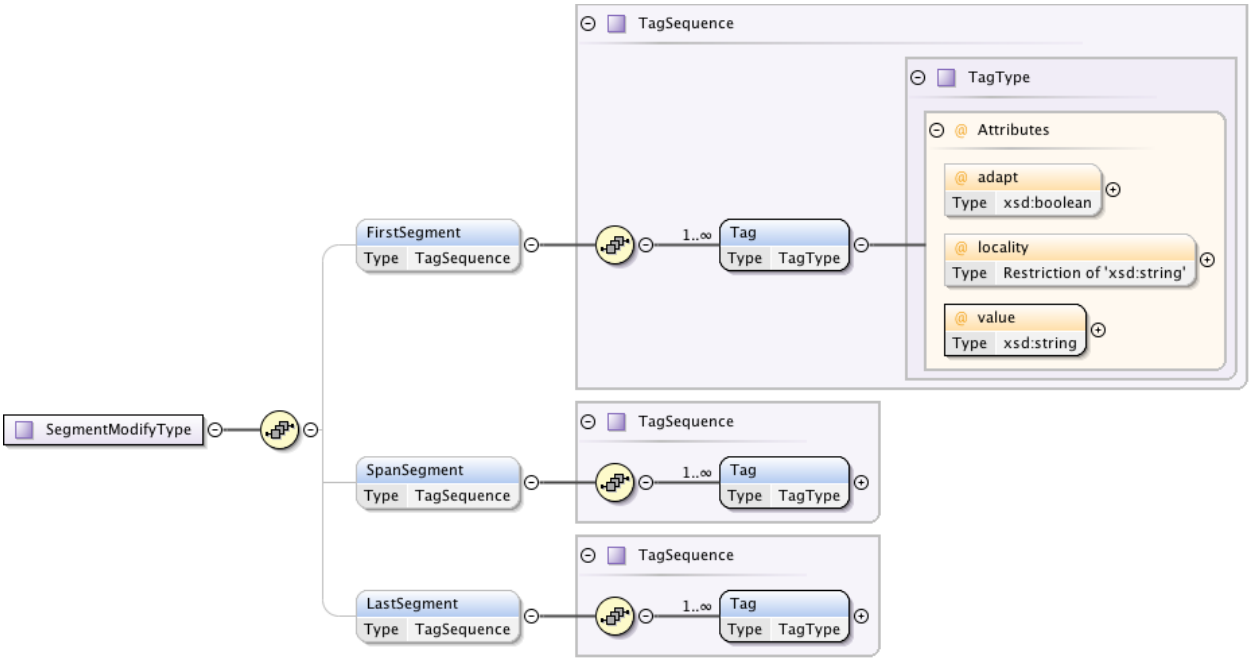

Figure 26 - SegmentModify XML Schema

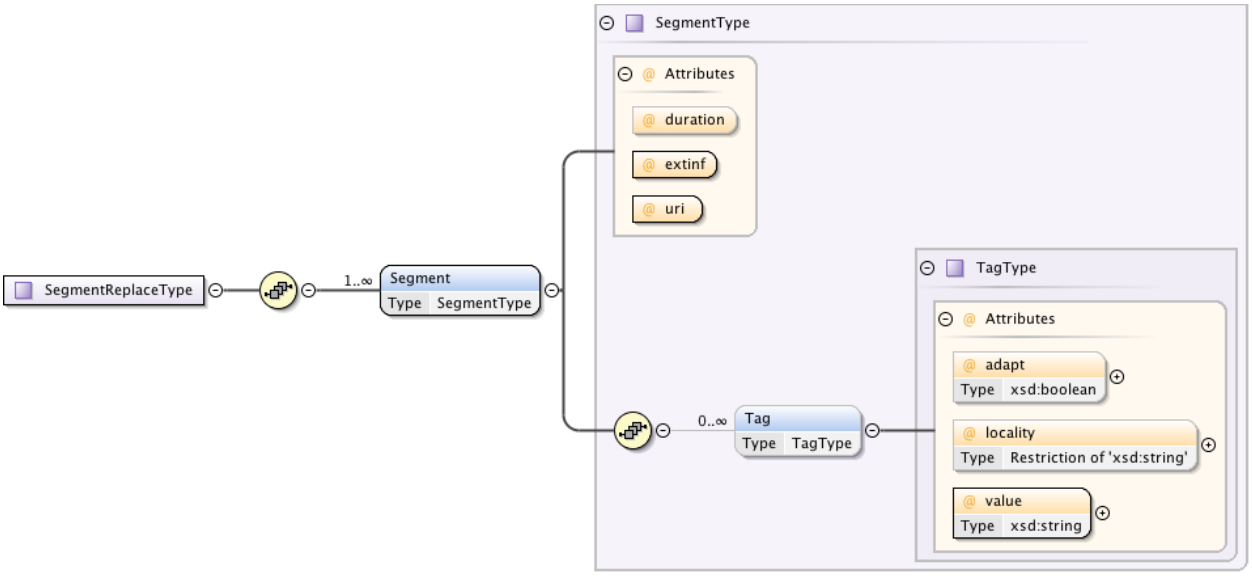

Figure 27 - SegmentReplace XML Schema

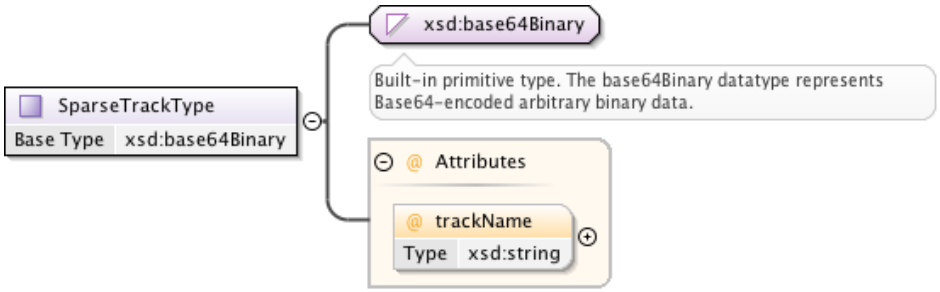

Figure 28 - SparseTrack XML Schema

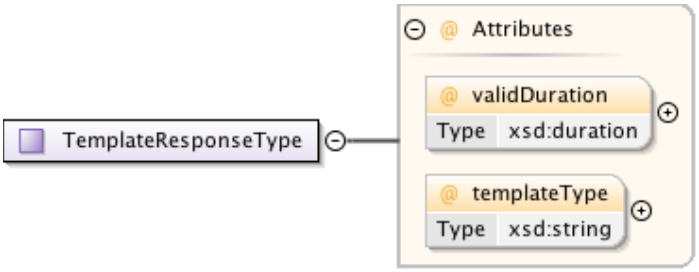

Figure 29 - TemplateResponse XML Schema

9.3.1 ManifestConfirmConditionNotification Semantics

If the fragmenter does not recognize or support the returned attribute/element or its semantics, then the fragmenter is to ignore the data.

Table 30 - ManifestConfirmConditionNotification Elements

Name	Required	Description
ManifestResponse	Y	See Section 9.3.2
Status Code	N	See Section 6.1

9.3.2 *ManifestResponse* Semantics

If the fragmenter does not recognize or support the returned attribute/element or its semantics, then the fragmenter is to ignore the data.

Table 31 - ManifestResponse Attributes

Name	Required	Description
acquisitionPointIdentity	Y	A required string, typically a system-wide unique string, identifying the transcoder at a specific site on a specific channel/network feed.
acquisitionSignalID	Y	A required string, typically a GUID, generated by the transcoder identifying the signal point being confirmed.
signalPointID	N	Optional string generated by the POIS identifying the confirmed signal.
dataPassThrough	N	Optional Boolean value that indicates if the data needs to be forwarded unmodified to the downstream systems. For example, in HLS applications if this parameter is true, then the fragmenter will pass the SCTE 35 signal in the data PID of the transport stream.

Name	Required	Description
duration	N	Optional UTC encoded signal region (e.g., break/avail) duration. See Section 6.3 for additional format information.

Table 32 - ManifestResponse Elements

Name	Required	Description
SegmentModify	N	The returned object when the manifest file is to be altered in media segment modify mode. In media segment modify mode, only a limited set of media segments are altered. The first media segment associated with the signal starting MAY be modified, zero or more "middle" media segments MAY be altered, and the last media segment of a region MAY be enhanced. That is, the splice exit/out media segment MAY be updated (i.e., signal start point), the splice return/in media segment MAY be altered (i.e., signal region end), and if the optional spanSegment object is present, each media segment following the first media segment is modified until the last media segment is encountered. The spanSegment information assists when a signal region spans two or more manifest files. This object is typically only used for live event manifest generation. VOD manifest files are expected to be self-contained and SHOULD NOT use the spanSegment object. For clarification, the use of SegmentModify does not entail modifying the associated URI but rather modifying the media tags only. With that, the media tags in question are bound by the duration of the event signal in the request message. A spanSegment object typical use is in scenarios where an "exit" from the network tagged in the conditioned manifest is in the past of a live stream and a downstream consumer of the conditioned manifest must be made aware by further tagging of the media segments.
SegmentReplace	N	One or more media segments (i.e., a JSON segments array entry or an XML Segment element) to *replace* the existing media segments starting at the signal point thru the end of the signal region. Both the extinf attribute and the uri attribute SHALL be present in each media segment. The extinf attribute supplies data for the media segment starting "#EXTINF" tag line. The uri attribute is used as the media segment's URI component. Each media segment is substituted for an existing media segment in sequence.
SparseTrack	N	Optional data to be embedded in an ABR Microsoft Smooth sparse track. The data blob is an adaptable string similar to the tags/value attribute.
SecurityMetadata	N	Optional security data to provide instructions for encryption key rotations and/or data to be forwarded downstream.
TemplateResponse	N	Optional: Indicates that this response is a template response. The template MAY be used with an SCTE 35 message to create the manifest conditioning for the SCTE 35 message.

9.3.2.1 SegmentModify

Table 33 - SegmentModify Elements

Name	Required	Description
FirstSegment	Y	One or more manifest lines that are inserted at the signal point start media segment (i.e., the signal splice start location). The media segment's start record marker (i.e., #EXTINF line) and URI are not altered. If the signal region duration (as signaled in ManifestResponse@duration) is zero, the manifest lines are inserted before the media segment immediately following the signal point, and all Tags SHALL have a (possibly implicit) locality of "before". Tag element: See Section 9.3.3.
SpanSegment	N	Optional and typically only used in live streaming mode. One or more manifest lines that are inserted for each media segment between the first media segment and the last media segment, excluding the first and last media segments, which are independently specified using the firstSegment and lastSegment objects. Tag element: See Section 9.3.3.
LastSegment	Y	One or more manifest lines that are inserted with the last media segment identified as the end location. The media segment's start record marker (i.e., #EXTINF line) and URI are not altered. If the signal region duration (as signaled in ManifestResponse@duration) is zero, the manifest lines are inserted after the media segment immediately preceding the signal point, and all Tags SHALL have a locality of "after". Tag element: See Section 9.3.3.

9.3.2.2 SegmentReplace

Table 34 - SegmentReplace Elements

Name	Required	Description
Segment	Y	A single media segment consisting of the extinf attribute, a URI, and zero or more tag lines.

9.3.2.2.1 Segment

Table 35 - Segment Attributes

Name	Required	Description
duration	N	Optional UTC encoded segment duration. See Section 6.3 for additional format information.
extinf	N	The #EXTINF line data which the fragmenter is to append to the string "#EXTINF:" which is assumed to be provided by the fragmenter. Thus, a common attribute value is "8," and the fragmenter outputs "#EXTINF:8," for the media segment.
Uri	N	The URI to be adapted by the fragmenter and inserted as the URI component of a manifest record. Every URI attribute value is to be modified by the fragmenter before placement into the manifest file using the substitution keywords defined in Section 9.3.3.

Table 36 - Segment Elements

Name	Required	Description
Tag	N	The manifest tag lines to be inserted. See Section 9.3.3 for additional information.

9.3.2.3 SparseTrack

Table 37 - SparseTrack Attributes

Name	Required	Description
trackName	N	Optional track name identifier. If omitted, the first sparse track is to be assumed.
value	N	Required string placed into the Smooth sparse track. Section 9.3.3 contains the substitution keyword and description table.

9.3.2.4 Template Response

Table 38 - Template Response Attributes

Name	Required	Description
validDuration	N	Optional ISODuration type which indicates for how long this template response is valid. After the duration expires, the packager will call out to the POIS upon receipt of the next SCTE 35 message. If not present, the template will always be valid.
templateType	N	Optional string that indicates to what type of SCTE 35 message this template can be applied. If this value is not present, the template will be applied to all SCTE 35 messages. Valid values are in Table 39.

Table 39 - TemplateType Attribute Values

Value	Description
NETWORK_OUT	The template will be applied if the SCTE 35 message indicates a network out condition.
NETWORK_IN	The template will be applied if the SCTE 35 message indicates a network in condition.
ANY	The template will be applied to all SCTE 35 messages.
PRIVATE	Custom type attribute.

9.3.2.4.1 Template Semantics

The TemplateResponse will be able to use macro substitutions in the ManifestResponse which instruct the packager about how to condition the manifests using the SCTE 35 message as a source. These macros are delimited by the '$' character in the notification response. Here is a list of the possible macros along with the origin of the data for that macro. If a specified macro element is not present in the SCTE 35 message, that element of the notification SHOULD be ignored.

- *acquisitionPointIdentity* Configured on the packager
- *acquisitionSignalID* Randomly generated per signal

- *segmentationEventId* Extracted from segmentation_event_id in the segmentation descriptor
- *segmentationUpid* Extracted from segmentation_upid() name in the segmentation descriptor
- *duration* Extracted from segmentation_duration in the segmentation descriptor
- *hdsDuration* Duration expressed as fractional seconds
- *availNum* Extracted from splice insert command
- *availExpected* Extracted from splice insert command
- *utcPoint* Expected wall clock time of signal point in UTC timestamp (XML dateTime)
- *ptsTime* Expected PTS value of signal point
- *smoothTime* Expected Smooth timestamp of signal point
- *hdsTime* Expected signal point time expressed as fractional seconds
- *binarySignal* Base64 encoded representation of the SCTE 35 signal

9.3.3 Tag Semantics

Tag (XML sequence)—One or more manifest tags (lines) to be directly inserted into the output manifest file. Each line is explicitly controlled, via the "adapt" attribute, as to whether the line element is to be placed directly into the manifest file unaltered or if the fragmenter is to enhance the returned attribute value. If the line is to be altered/enhanced by the fragmenter prior to placement into the manifest file, the "adapt" attribute is set to the "true" value. The fragmenter fills in the placeholder substitution keyword framed by the start substitution delimiter '${' and ending delimiter '}'. Thus, the full substitution sequence is ${keyword}. Table 40 lists the substitution keywords and their descriptions. The line insertion location is controlled by the "locality" attribute and lines having the same locality are positioned in returned document order.

Table 40 - Manifest Line Substitution Keywords

Substitution Keyword	Value Description
timeFromSignal	The total amount of time passed since the signal point was encountered. The value is formatted as an ISODuration. The value is typically the sum of media segment #EXTINF duration values inclusive of the signal region ending segment.
segmentID	The original media segment URI.
streamID	The configured streamId value.

9.3.3.1 Tag Attributes

Table 41 - Tag Attributes

Name	Required	Description
adapt	N	Optional Boolean indicating if the value attribute's string is to be modified by the fragmenter before placement into the manifest file. If the attribute is omitted, the default value is "false."
locality	N	Optional string specifying the line location relative to the media segment. Table 42 lists the possible values along with a location description. The attribute values SHALL appear exactly as they do in Table 42. If the attribute is omitted, the default value is "before".
value	N	Required string, typically starting with "#EXT", placed into the manifest file.

Table 42 - locality Attribute

Value	Description
before	The line is to be placed before the media segment's #EXTINF tag line.
within	The line is to be placed in between the media segment's start #EXTINF line and the media segment's ending URI line.
after	The line is to be placed after the media segment's URI line.

9.3.4 Notification Examples

To improve readability, all XML schemas in this section are assumed to have the following namespace declaration:

```
xmlns:core="urn:cablelabs:md:xsd:core:3.0"
xmlns:sig="urn:cablelabs:md:xsd:signaling:3.0"
xmlns="urn:cablelabs:iptvservices:esam:xsd:manifest:1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

9.3.4.1 HLS Segment Modify

```
<ManifestConfirmConditionNotification>
<ManifestResponse
   acquisitionPointIdentity="ESPN_East_Acquisition_Point_1"
   acquisitionSignalID="4A6A94EE-62FA-11E1-B1CA-882F4824019B"
   signalPointID="10D65A82-631E-11E1-B5B7-7A5A4824019B"
   duration="PT1M0S"
   dataPassThrough="true">
   <SegmentModify>
        <FirstSegment>
            <Tag value="#EXT-X-SPLICE-EXIT:
                        SpliceDescriptors=0x786df876dffa87687"/>
            <Tag value="#EXT-X-DISCONTINUITY"/>
        </FirstSegment>
        <SpanSegment>
            <Tag adapt="true" value="#EXT-X-SPLICE-SPAN:
    SpliceDescriptors=0x786df876dffa87687,TimeFromSignal=${timeFromSignal}"/>
        </SpanSegment>
        <LastSegment>
            <Tag adapt="true" value="#EXT-X-SPLICE-
SPAN:SpliceDescriptors=0x786df876dffa87687,TimeFromSignal=${timeFromSignal}"/>
            <Tag locality="after" value="#EXT-X-SPLICE-
RETURN:SpliceDescriptors=0x786df876dffa87687"/>
            <Tag locality="after" value="#EXT-X-DISCONTINUITY"/>
        </LastSegment>
    </SegmentModify>
</ManifestResponse>
</ManifestConfirmConditionNotification>
```

9.3.4.2 HLS Segment Replace

```
<ManifestConfirmConditionNotification>
<ManifestResponse
   acquisitionPointIdentity="ESPN_East_Acquisition_Point_1"
   acquisitionSignalID="4A6A94EE-62FA-11E1-B1CA-882F4824019B"
   signalPointID="10D65A82-631E-11E1-B5B7-7A5A4824019B"
   duration="PT1M0S"
   dataPassThrough="true">
    <SegmentReplace>
        <Segment extinf="10,"
uri="http://ADMServer/decisions?segment=${segmentID}&amp;spliceDescriptors=0x0
0085448495300112233&amp;net=CNN&amp;time=2010-12-17T11:12:42.123Z">
            <Tag value="#EXT-X-SPLICE-
EXIT:spliceDescriptors=0x00085448495300112233"/>
```

```
            <Tag value="#EXT-X-DISCONTINUITY"/>
        </Segment>
        <Segment extinf="10,"
uri="http://ADMServer/decisions?segment=${segmentID}&amp;spliceDescriptors=0x0
0085448495300112233&amp;net=CNN&amp;time=2010-12-17T11:12:42.123Z"/>
        <Segment extinf="10,"
uri="http://ADMServer/decisions?segment=${segmentId}&amp;spliceDescriptors=0x0
0085448495300112233&amp;net=CNN&amp;time=2010-12-17T11:12:42.123Z">
            <Tag locality="after" value="#EXT-X-SPLICE-
RETURN:spliceDescriptors=0x0008544849530011223 3"/>
            <Tag locality="after" value="#EXT-X-DISCONTINUITY"/>
        </Segment>
    </SegmentReplace>
</ManifestResponse>

</ManifestConfirmConditionNotification>
```

9.3.4.3 HSS Sparse Track

```
<ManifestConfirmConditionNotification>
<ManifestResponse
   acquisitionPointIdentity="ESPN_East_Acquisition_Point_1"
   acquisitionSignalID="4A6A94EE-62FA-11E1-B1CA-882F4824019B">
    <SparseTrack
trackName="ad_marker">PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiPz4KPEFj
cXVpcmVkU2lnbmFsIHhtbG5zPSJodHRwOi8vd3d3LmNvbWNhc3QuY29tL3NjaGVtYXMvTkdPRC9TaW
duYWwvMjAxMC9SMVYwIiB4bWxuczp4c2k9Imh0dHA6Ly93d3cudzMub3JnLzIwMDEvWE1MU2NoZW1h
LWluc3RhbmNlIiB4c2k6c2NoZW1hTG9jYXRpb249Imh0dHA6Ly93d3cuY29tY2FzdC5jb20vc2NoZW
1hcy9OR09EL1NpZ25hbC8yMDEwL1IxVjAgQ0MtTkdPRC1TSUdOQUxJTkctUjFWMC0xMDEyMTAueHNk
Ij4KCTxBY3F1aXNpdGlvblBvaW50SW5mbyBBY3F1aXNpdGlvblBvaW50SWRlbnRpdHk9IkUhQ01DU0
EzIiBBY3F1aXNpdGlvblNpZ25hbElEPSI1YjQ4ZjdmZi1hMTJlLTQ0ZWEtOGIxZC1iODA5OGZjZDEw
ZmEiLz4KCTxVVENPZlN3aXRjaFBvaW50IFVUQ1BvaW50PSIyMDEyLTAyLTI5VDIxOjE5OjA5WiIvPg
oJPFN0cmVhbVRpbWVzPgoJCTxTdHJlYW1UaW1lIFRpbWVUeXBlPSJTbW9vdGgiIFRpbWVWYWx1ZT0i
MjM3NTk0NDAxMDg4OCIvPgoJCTxTdHJlYW1UaW1lIFRpbWVUeXBlPSJQVFMiIFRpbWVWYWx1ZT0iND
IwMzYyNjkxNCIvPgoJPC9TdHJlYW1UaW1lcz4KPC9BY3F1aXJlZFNpZ25hbD4K
    </SparseTrack>
</ManifestResponse>
</ManifestConfirmConditionNotification>
```

9.3.4.4 HSS Blackout (with Key Rotation)

```
<ManifestConfirmConditionNotification>
<ManifestResponse
   acquisitionPointIdentity="ESPN_East_Acquisition_Point_1"
   acquisitionSignalID="4A6A94EE-62FA-11E1-B1CA-882F4824019B">
    <SparseTrack
trackName="ad_marker">PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiPz4KPEFj
cXVpcmVkU2lnbmFsIHhtbG5zPSJodHRwOi8vd3d3LmNvbWNhc3QuY29tL3NjaGVtYXMvTkdPRC9TaW
duYWwvMjAxMC9SMVYwIiB4bWxuczp4c2k9Imh0dHA6Ly93d3cudzMub3JnLzIwMDEvWE1MU2NoZW1h
LWluc3RhbmNlIiB4c2k6c2NoZW1hTG9jYXRpb249Imh0dHA6Ly93d3cuY29tY2FzdC5jb20vc2NoZW
1hcy9OR09EL1NpZ25hbC8yMDEwL1IxVjAgQ0MtTkdPRC1TSUdOQUxJTkctUjFWMC0xMDEyMTAueHNk
Ij4KCTxBY3F1aXNpdGlvblBvaW50SW5mbyBBY3F1aXNpdGlvblBvaW50SWRlbnRpdHk9IkUhQ01DU0
EzIiBBY3F1aXNpdGlvblNpZ25hbElEPSI1YjQ4ZjdmZi1hMTJlLTQ0ZWEtOGIxZC1iODA5OGZjZDEw
ZmEiLz4KCTxVVENPZlN3aXRjaFBvaW50IFVUQ1BvaW50PSIyMDEyLTAyLTI5VDIxOjE5OjA5WiIvPg
oJPFN0cmVhbVRpbWVzPgoJCTxTdHJlYW1UaW1lIFRpbWVUeXBlPSJTbW9vdGgiIFRpbWVWYWx1ZT0i
MjM3NTk0NDAxMDg4OCIvPgoJCTxTdHJlYW1UaW1lIFRpbWVUeXBlPSJQVFMiIFRpbWVWYWx1ZT0iND
IwMzYyNjkxNCIvPgoJPC9TdHJlYW1UaW1lcz4KPC9BY3F1aXJlZFNpZ25hbD4K
    </SparseTrack>
    <SecurityMetadata>
z4KPEFjcXVpcmVkU2lnbmFsIHhtbG5zPSJodHRwOi8vd3d3LmNvbWNhc3QuY29tL3NjaGVtYXMvTkd
PRC9TaWduYWwvMjAxMC9SMVYwIiB4bWxuczp4c2k9Imh0dHA6Ly93d3cudzMub3JnLzIwMDEv
    </SecurityMetadata>
</ManifestResponse>
</ManifestConfirmConditionNotification>
```

10 EXAMPLE SIGNALING DIAGRAMS

The messages described throughout this section can be sent or received synchronously or asynchronously. In order to provide additional clarity regarding this concept, the following example signaling flows have been provided.

10.1 In-band Signaling Example

In the following example, the content provider provides an in-band SCTE 35 mark signaling a necessary change to alternate content for an upcoming time and duration, among other details. Upon receiving the in-band signal, the transcoder sends a *SignalProcessingEvent* to the Signal Processor (e.g., POIS). The Signal Processor responds with a synchronous *SignalProcessingNotification*, which MAY include instructions to modify or replace the in-band signal. The signal is sent downstream to the fragmenter, which sends a *ManifestConfirmConditionEvent* to the Signal Processor, which responds with a synchronous *ManifestConfirmConditionNotification*.

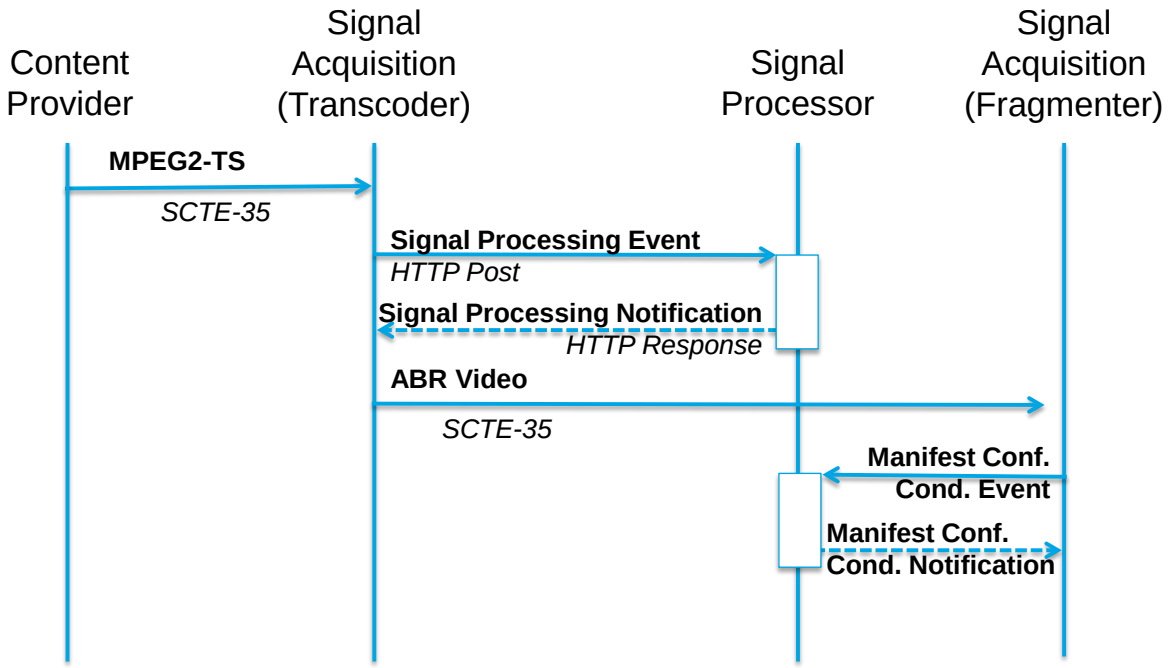

Figure 30 - In-band Signaling Example

10.2 Out-of-band Signaling Example

In the following example, the content provider notifies the content distributor of an upcoming necessary change to alternate content along with the time and duration, among other details. Prior to the specified time, the Signal Processor sends an asynchronous *SignalProcessingNotification* to the transcoder. The transcoder inserts the appropriate signal within the video. The signal is sent downstream to the fragmenter, which sends a *ManifestConfirmConditionEvent* to the Signal Processor, which responds with a *ManifestConfirmConditionNotification*.

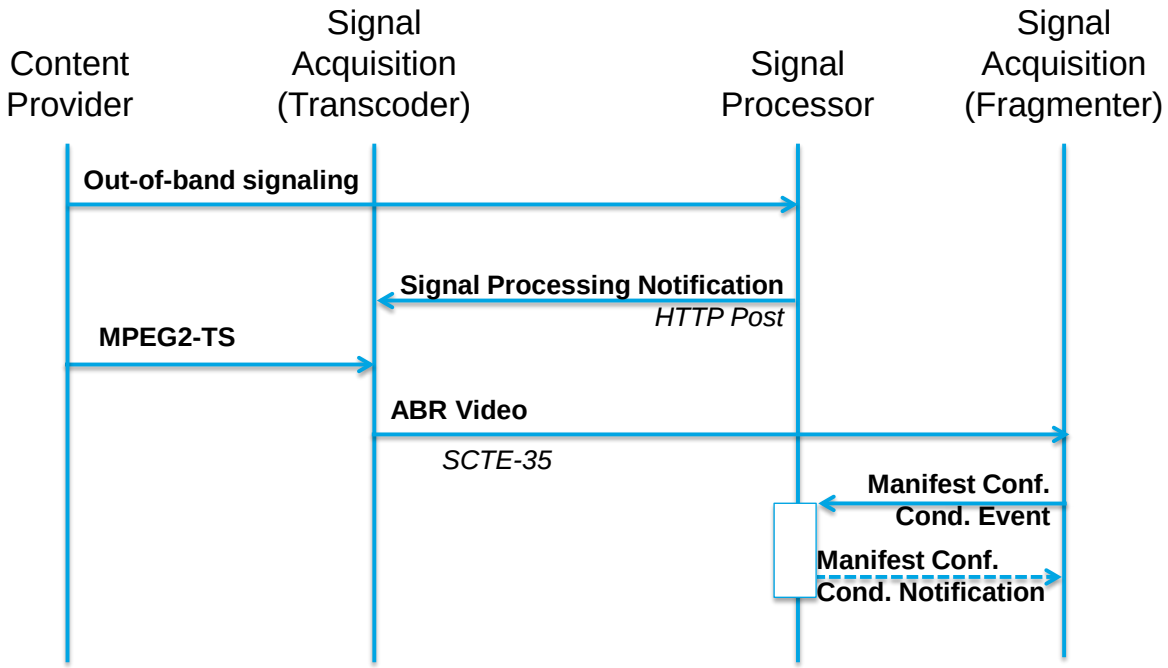

Figure 31 - Out-of-band Signaling Example

10.3 Out-of-band Manifest Conditioning Example

In the following example, the content provider notifies the content distributor of an upcoming necessary change to alternate content along with the time and duration, among other details. At the specified time, the Signal Processor sends an asynchronous *ManifestConfirmConditionNotification* to the fragmenter. After receiving the notification, the fragmenter modifies the manifest or sparse track appropriately.

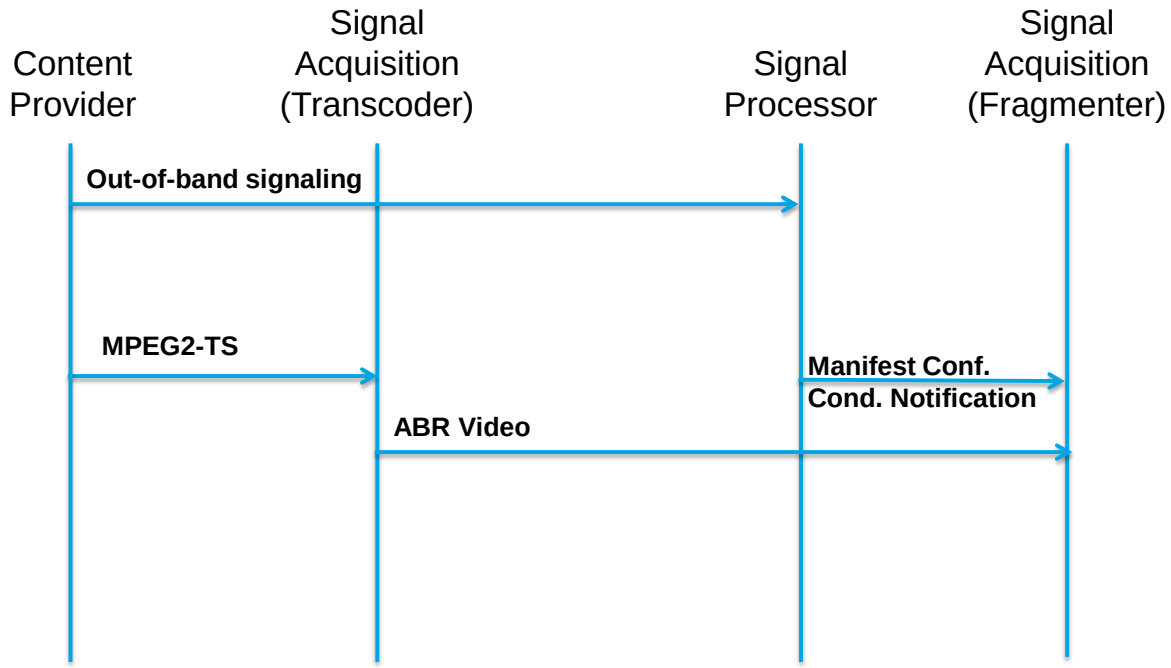

Figure 32 - Out-of-band Manifest Conditioning Example

Annex A ESAM Normative Schemas

The messages, elements, and attributes that make up this specification are not explicitly defined in this document. The XML schemas, provided separately, are a normative part of this specification and SHALL be adhered to.

No messages representing the interfaces defined in the schema are considered conformant unless they are valid according to the schema documents. In the case where the written normative specification and the normative schema document conflict, the specification SHALL take precedence. Furthermore, there are additional requirements described in the schema annotations that are not enforceable by schema validation mechanisms. These requirements SHALL also be strictly observed.

The inclusion of a normative XML schema document does not require or imply the specific use of the schema nor a requirement that a message be validated.

A.1 ESAM Common Schema

The normative version of the ESAM Common schema file is published as a part of this release with the name OC-SP-ESAM-API-I03-Common.xsd.

A.2 ESAM Signal Schema

The normative version of the ESAM Signal schema file is published as a part of this release with the name OC-SP-ESAM-API-I03-Signal.xsd.

A.3 ESAM Manifest Schema

The normative version of the ESAM Manifest schema file is published as a part of this release with the name OC-SP-ESAM-API-I03-Manifest.xsd.

Appendix I Acknowledgements

We wish to thank the participants contributing directly to this document:

Arris: Guy Cherry

Elemental Technologies: Jesse Rosenzweig

Envivio: Alex MacAulay

Comcast: Allen Broome, Ted Dawson, Francesco Dorigo, Kevin Flanagan, and Walt Michel

Time Warner Cable: Chuck Hasek

CableLabs: David Agranoff, Don Burt, and Daryl Malas

Appendix II Revision History

The following ECNs were incorporated into version I02 of this specification:

ECN Identifier	Author	Accepted Date	Description
ESAM-API-N-13.1821-2	Michel	4/4/2013	Add support for batch processing (e.g., VOD Content) and element recovery
ESAM-API-N-13.1856-2	Agranoff	9/3/2013	Update Metadata references
ESAM-API-N-13.1857-2	Agranoff	9/3/2013	Clarify transmission of SignalPointID between Transcoder and Packager/Fragmenter
ESAM-API-N-13.1858-1	Agranoff	9/3/2013	Clarification of LastSegment and guidelines for inserting a new signal at transcoder
ESAM-API-N-13.1859-1	McBride	9/3/2013	Add support for template response to manifest confirmation API
ESAM-API-N-13.1860-1	Flanagan	9/27/2013	Linear Stream Switching Interface
ESAM-API-N-13.1865-1	Agranoff	9/27/2013	Additional Schema changes

The following ECN was incorporated into I03 of this specification:

ECN Identifier	Author	Accepted Date	Description
ESAM-API-N-13.1868-2	Agranoff	10/25/13	Fixes for ESAM Signaling Schema