Errata Notice on Schema Locations

March 17, 2022

This standard makes use of namespace locations with a form of http://www.scte.org/schemas/xyz/*, where “xyz” is the location of the specific schema being referenced. Due to limitations on the current SCTE website, those specific locations are not available.

To find such schemas:

1. Go to the standards download page at https://www.scte.org/standards/library/catalog/
2. Search for the standard number (xyz in the above example)
3. Select the document from the table
4. Scroll to the “Supporting Documentation” section of the document webpage.

The schema will be listed within the Supporting Documentation section.

This notice will be removed once the exact namespace values are functional.

SCTE · ISBE

STANDARDS

Digital Video Subcommittee

AMERICAN NATIONAL STANDARD

ANSI/SCTE 130-2 2020

Digital Program Insertion–Advertising Systems Interfaces

Part 2

Core Data Elements

NOTICE

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

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

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

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

140 Philips Road
Exton, PA 19341

Table of Contents

List of Figures

Title **Page Number**

List of Tables

Title **Page Number**

Digital Program Insertion—Advertising Systems Interfaces Part 2—Core Data Elements

1.0 SCOPE

This document, SCTE 130 Part 2, describes the Digital Program Insertion - Advertising Systems Interfaces' core messaging and data types using extensible markup language (XML), XML Namespaces, and XML Schema. Core messaging includes the extensible message schemas, the common SCTE 130 message attributes, and the required SCTE 130 messages. The core data types are XML attributes and XML elements that may be used in any SCTE 130 message element or within any SCTE 130 element definition.

2.0 REFERENCES

2.1 Normative references

The following standards contain provisions that, through reference in this text, constitute provisions of this standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent edition of the standards indicated below.

[XML]	W3C Recommendation, "Extensible Markup Language (XML) 1.0 (Fourth Edition)", Tim Bray, et al, 16 August 2006, http://www.w3.org/TR/2006/REC-xml-20060816/.
[XMLNamespaces]	W3C Recommendation, "Namespaces In XML (Second Edition)", Tim Bray, et al, 16 August 2006, http://www.w3.org/TR/2006/REC-xml-names-20060816/.
[XMLSchemaP1]	W3C Recommendation, "XML Schema Part 1: Structures (Second Edition)", H. Thompson, et al, 28 October 2004, http://www.w3.org/TR/xmlschema-1/.
[XMLSchemaP2]	W3C Recommendation, "XML Schema Part 2: Datatypes (Second Edition)", P. Biron, et al, 28 October 2004, http://www.w3.org/TR/xmlschema-2/.
[XMLInfoSet]	W3C Recommendation, "XML InfoSet (Second Edition)", John Cowan, Richard Tobin, 4 February 2004, http://www.w3.org/TR/2004/REC-xml-infoset-20040204/.
[SCTE35]	ANSI/SCTE 35 2019r1—Digital Program Insertion Cueing Message for Cable.

[SCTE118-3]	ANSI/SCTE 118-3 2019—Program Specific Ad Insertion—Traffic System to Ad Insertion System File Format Specification.
[CLADI1-1]	CableLabs Asset Distribution Interface Specification Version 1.1: MD-SP-ADI1.1-I04-060505, http://www.cablelabs.com/specifications/MD-SP-ADI1.1-I04-060505.pdf
[SCTE236]	ANSI/SCTE 236 2017—Content Metadata.
[RFC2616]	Part 3 of Hypertext Transfer Protocol -- HTTP/1.1
[RFC3986]	RFC 3986, "Uniform Resource Identifier (URI): Generic Syntax", January 2005, www.ietf.org/rfc/rfc3986.txt
[IEEE802]	IEEE Std 802-2001—IEEE Standard for Local and Metropolitan Area Networks: Overview and Architecture-IEEE Computer Society, approved 6 December 2001, http://standards.ieee.org/getieee802/802.html

.

2.2 Informative References

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

[XMLSchemaP0]	W3C Recommendation, "XML Schema Part 0: Primer (Second Edition)", D. Fallside, et al, 28 October 2004, http://www.w3.org/TR/xmlschema-0/.
[RFC2046]	Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types
[RFC6838]	Media Type Specifications and Registration Procedures
[RFC4122]	RFC 4122 "The Universally Unique IDentifier (UUID) URN Namespace, July 2005, www.ietf.org/rfc/rfc4122.txt.
[SCTE67]	SCTE 67—Recommended Practice for Digital Program Insertion for Cable

[SCTE118-1]	ANSI/SCTE 118-1—Program-Specific Ad Insertion - Data Field Definitions, Functional Overview and Application Guidelines.
[SCTE118-2]	ANSI/SCTE 118-2—Program-Specific Ad Insertion—Content Provider to Traffic Communication Applications Data Model.
[CLVOD1-1]	CableLabs Video-on-Demand Content Specification 1.1: MD-SP-VOD-CONTENT1.1-I05-060831, http://www.cablelabs.com/specifications/MD-SP-VOD-CONTENT1.1-I05-060831.pdf.
[SCTE130-1]	SCTE 130-1: Digital Program Insertion—Advertising Systems Interfaces Part 1—Overview.
[SCTE130-3]	SCTE 130-3: Digital Program Insertion—Advertising Systems Interfaces Part 3—Ad Management Service (ADM) Interface.
[SCTE130-7]	SCTE 130-7: Digital Program Insertion—Advertising Systems Interfaces Part 7—Message Transport.

3.0 COMPLIANCE NOTATION

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

4.0 DEFINITION OF TERMS

Throughout this standard the terms below have specific meanings. Because some of the terms are defined in other SCTE documents having very specific technical meanings, the reader is referred to the original source for their definition. For terms defined by this standard, brief definitions are given below.

content: The video, audio, and data streams taken together as a single identifiable unit. Content *may* refer to the original entertainment (programming) content, an ad spot, an interactive or enhanced application asset, or any other similar asset.

default endpoint: The endpoint where messages are delivered in the absence of a message specific endpoint designation.

endpoint: An address, a Uniform Resource Identifier (URI), or a specific location where a logical service function or functions ***shall*** be found and consumed via message exchange.

event: A general term indicating something has happened or occurred.

logical service: A well-defined, self-contained set of functions accessible via one or more endpoints. The logical service has some type of underlying computer system that supports message communication.

message: The unit of communication between two logical services.

program: A time-bounded collection of video, audio, and data streams.

registration-established service channel: A service channel duration commencing with a successful registration and continuing until termination through deregistration.

scope of uniqueness: Uniqueness is context relative and for this specification's purpose ***shall*** be defined by one of following: global, service channel or element.

- **global uniqueness:** Global or universally unique and at no other time ***shall*** the item be compromised, reused, or otherwise taken to have more than one meaning. Enforcement of uniqueness as well as creation of globally unique identifiers is outside the scope of this specification and RFC 4122 is recommended. See [RFC4122] for additional information.

- **service channel uniqueness:** Uniqueness scoped by the @identity attribute and the service channel and at no other time ***shall*** the item be compromised, reused, or otherwise taken to have more than one meaning. XML messages ***shall*** be service channel unique and a message ***shall not*** be compromised or reused for the duration of the service channel. Service channel uniqueness is relative only to the endpoints where the message exchange is occurring and within the identity domain of the two endpoints involved in the exchange. Enforcement of uniqueness as well as creation of identity unique identifiers is outside the scope of this specification and RFC 4122 is recommended. See [RFC4122] for additional information.

- **element uniqueness:** Generally, XML elements ***shall*** be unique according to existing XML compliance where the element's distinctiveness is unambiguous and unique relative to its immediate spatial relationship to other elements.

service channel: A message communication path between two logical services.

5.0 ABBREVIATIONS

This document uses the following abbreviations:

ADS: Ad Decision Service.

ADM: Ad Management Service

CIS: Content Information Service

HA: High Availability

IANA: Internet Assigned Numbers Authority – See [RFC6838].

POIS: Placement Opportunity Information Service

SIS: Subscriber Information Service

UPID: Unique Program Identifier—See [SCTE35].

URI: Uniform Resource Identifier—See [RFC3986].

UUID: Universally Unique Identifier—See [RFC4122].

XML: Extensible Markup Language —See [XML].

6.0 INTRODUCTION

This document defines the following:

- The extensible SCTE 130 message structure.
- A set of required messages that ***shall*** be implemented by all SCTE 130 compliant logical services.
- A common set of XML elements (i.e., core elements) which *may* appear in any SCTE 130 message or within any SCTE 130 defined element.
- Common XML attributes used by the core data elements which *may* also be used by any other element.

All externally defined SCTE 130 messages ***shall*** contain the appropriate common message attributes as defined herein.

This specification also defines requirements for the SCTE 130 message transport without specifying the actual transport implementation. SCTE 130 Part 7 [SCTE130-7] defines the normative transport specification.

6.1 Document Organization

This document provides a structured, logical approach to the core aspects of SCTE 130. Each subsequent section focuses on a particular specification aspect and the document's presentation order provides a top-down, conceptual introduction necessary to implement SCTE 130.

Section 7 explains the document's notational conventions. Section 8 defines the XML namespace usage and the applicable XML semantics. Section 9 introduces the message concepts and the core XML attributes which are the foundation of every SCTE 130 message. This section also defines the message exchange characteristics common to all SCTE 130 message interactions.

Section 9 and its subsequent sections often *may* reference XML attributes and elements defined later in the document. The reader *may* occasionally need to

reference other sections of this document in order to find the complete explanation of a syntactic component.

Section 10 defines messages that ***shall*** be supported by every SCTE 130 logical service. Section 11 presents the core XML attributes followed by the principal specification XML elements with each group presented in alphabetical order. Appendix A concludes the document with the normative status code values, their meanings, and their message usage applicability.

7.0 NOTATIONAL CONVENTIONS

7.1 Normative XML Schema

Descriptions of messages, elements, and attributes are normative and, when combined with the normative XML schema document (provided separately), comprise the full normative specification. Non-normative schema illustrations and instance examples are included herein for informational purposes only. Any real or implied usage, semantics, or structure indicated by the schema illustrations and examples ***shall not*** be considered part of the specification.

No messages representing the interfaces defined in the schema are considered conformant unless they are valid according to the schema document. Additionally, other SCTE 130 standard normative parts *may* impose additional rules or restrictions that ***shall*** be adhered to in order for the messages to be considered conformant to those parts.

In the case where the written normative specification (for example, this document) and the normative schema document (i.e., the separately provided XML ‘xsd’ file) conflict, the written normative specification ***shall*** take precedence over the XML schema document.

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.

7.2 Document Conventions

XML elements *may* be listed in the format “element” or “prefix:element” without the double-quotes where “prefix” denotes the appropriate namespace defined in Section 8 and “element” denotes the name of the element. A slash “/” denotes that a child XML element or XML attribute follows.

XML attributes are listed in the format “@attribute” or “@prefix:attribute” without the double-quotes where “prefix” denotes the appropriate namespace defined in Section 8. An “@” character without the double-quotes denotes an XML attribute as opposed to an XML element, and “attribute” denotes the name of the attribute.

When describing concrete XML schemas, an element wildcard (<xsd:any/>) is represented by the notation ##any. The ##any element refers to all namespaces including the namespace defined by this document. Any SCTE 130 or alternative namespace element *may* be included.

The constrained wildcard element <xsd:any namespace="##other"/> is represented by ##other. The ##other element refers to all other namespaces except the namespace defined by this document. Any non-SCTE 130 Part 2 namespace element *may* be included.

An attribute wildcard (<xsd:anyAttribute/>) is represented by the notation @##any (or 'any ##any' within the schema diagrams). The ##any attribute indicates that any SCTE 130 defined or alternatively defined attribute *may* be included.

Figure 1 explains the schema illustration technique used throughout the SCTE 130 specifications. Symbols and their meanings are explained within the figure.

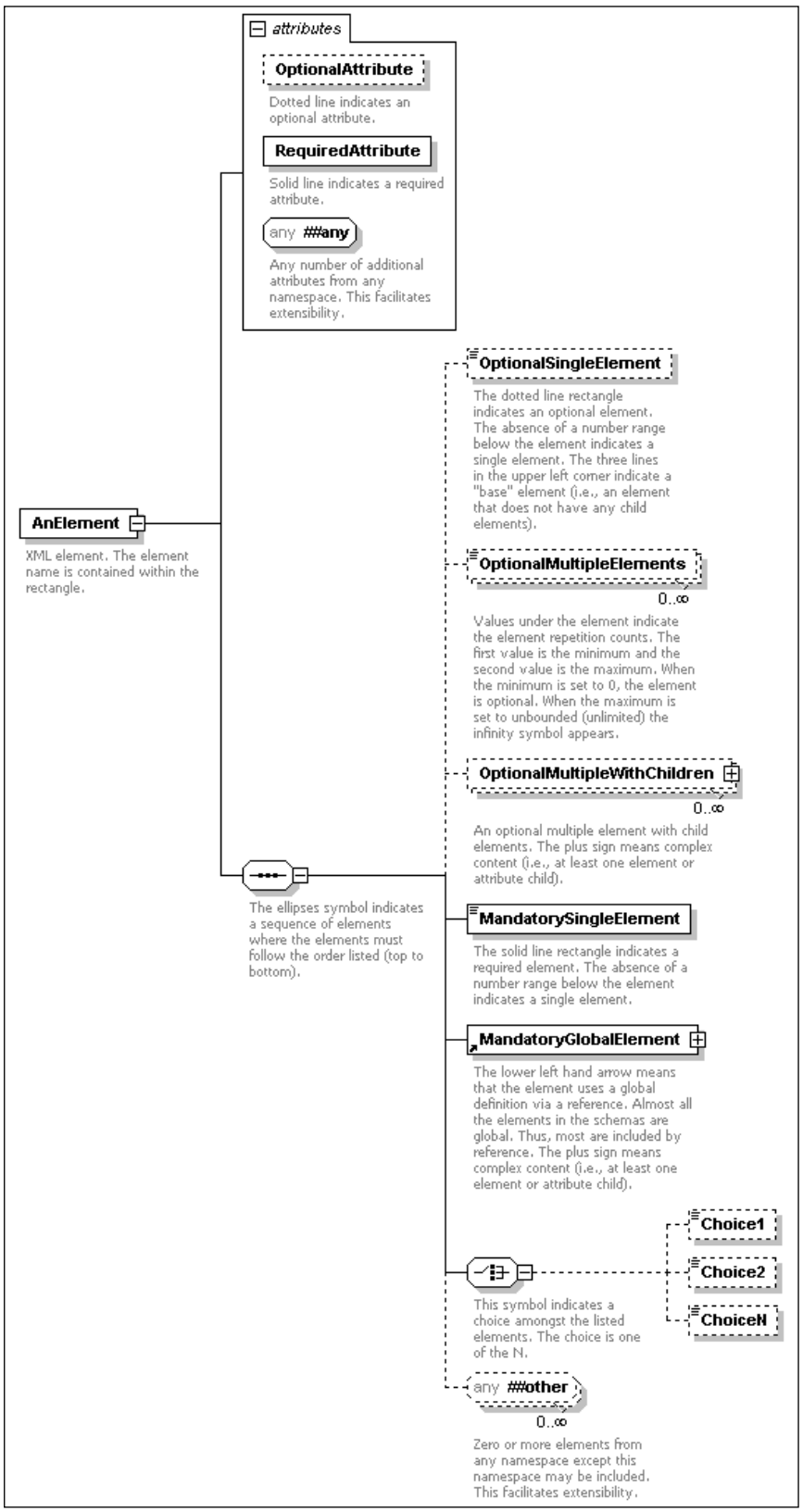

Figure 1: Schema Illustration Explanation

8.0 XML NAMESPACES

This document uses the prefix *'core'* for the interface associated with this specification's XML namespace URI. This URI ***shall*** be used by all implementations applying this specification. Table 1 lists the prefix, the corresponding namespace, and a description of the defining specification.

Standard	XML Schema Prefix	XML Schema Elements	Value
2020 (latest) SCTE 130 Part 2 (i.e., this document)	core	Schema namespace	http://www.scte.org/schemas/130-2/2008a/core[1]
		Schema version attribute	20200321
		Schema filename	SCTE_130-2_core_20200321.xsd
XML foundation. See [XMLSchemaP1].	xsd	Schema namespace	http://www.w3.org/2001/XMLSchema

Table 1: XML Namespace Declarations

The namespace is formatted per the SCTE recommendation and includes the SCTE 130 version through the inclusion of a date, which *may* minimally be a specification year identifier.

8.1 Unknown/Unrecognized/Unsupported XML Elements and Attributes

Generally, unknown, unrecognized or unsupported XML elements and attributes contained within SCTE 130 messages and elements *should* be ignored during message processing. Specifically, these are elements or attributes which the implementation does not understand or expect. The logical service *should* generate an appropriate response message for any element or attribute in this categorization. For example, the return message might contain an informational Note element. Ideally, validation *should* be utilized to identify and report prohibited constructs through the

[1] While this specification has a ratified year of 2020, the XML schema/XSD namespace has a year of 2008a, which is the year the XSD and this specification's syntax was initially ratified. All subsequent changes have been backwards compatible and thus, the namespace has not changed.

appropriate mechanisms. Elements and attributes that are prohibited by a namespace *should* generate an appropriate error response message.

8.2 Element Order

Element order is constrained by the schemas and must be preserved throughout processing of the XML document. In particular, the order of elements affects the end result of the processing. Consequently, an implementation failing to preserve the order *may* cause incorrect processing results. Subsequently, the process of producing an abstract XML Information Set (InfoSet) from a concrete XML document, e.g., by parsing it, ***shall*** always result in the same abstract InfoSet, with the same element order per XML InfoSet. (See [XMLInfoSet] for additional information.) Any intermediary processing *may* enhance the XML document but it ***shall not*** alter the abstract InfoSet element order (i.e., the XML elements comprising the document ***shall*** stay in document order).

8.3 Language Identification

In document processing, it is often useful to identify the natural or formal language in which the content is written. A special attribute named xml:lang *may* be inserted in documents to specify the language used in the contents and attribute values of any element in an XML document. See [XML] for the allowed values. Typically, the xml:lang attribute is utilized with an xsd:string type.

9.0 SCTE 130 MESSAGE BASICS

A logical service is a self-contained set of functions accessible via one or more endpoints where each function is accessed via an XML document exchange. An endpoint identifies where a function *may* be found and consumed and multiple functions *may* be found at the same endpoint. Endpoint discovery is outside the scope of this specification. A message is the unit of communication between two logical services. All SCTE 130 identified logical services exchange information based on a common XML schema defined herein.

The XML document root element for all communication exchanges is a specific form of request, response, notification or acknowledgement element referred to as a message. Communication involves a two-way message exchange of a specific paired message set as illustrated in Figure 2. The possible exchanged message sets ***shall*** be either a request/response message pair or a notification/acknowledgement message pair. All other message combinations ***shall not*** be allowed. When a request or notification message is received by an SCTE 130 logical service, then the appropriately paired response or acknowledgement message ***shall*** be returned.

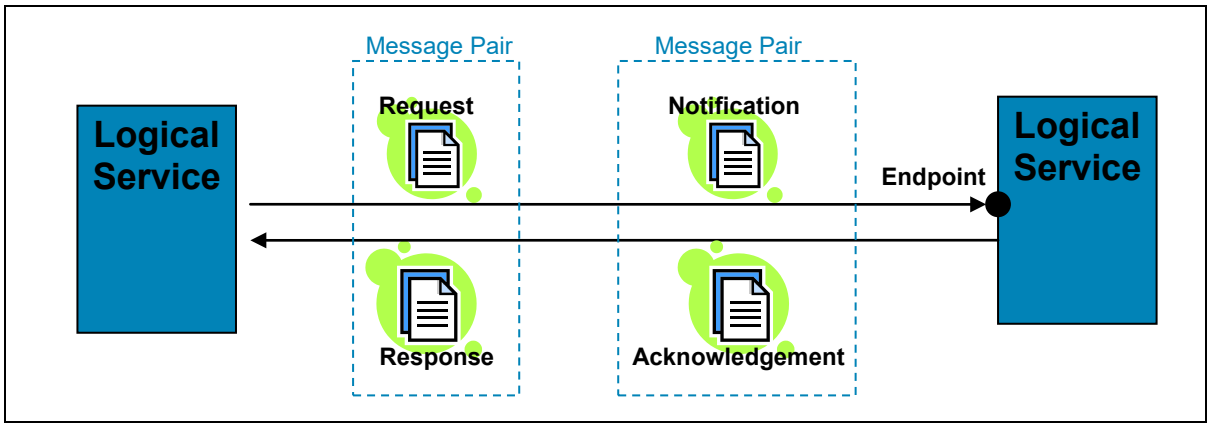

Figure 2: Basic Communication Overview

Figure 3 illustrates a single logical service *may* be simultaneously communicating with multiple other logical services.

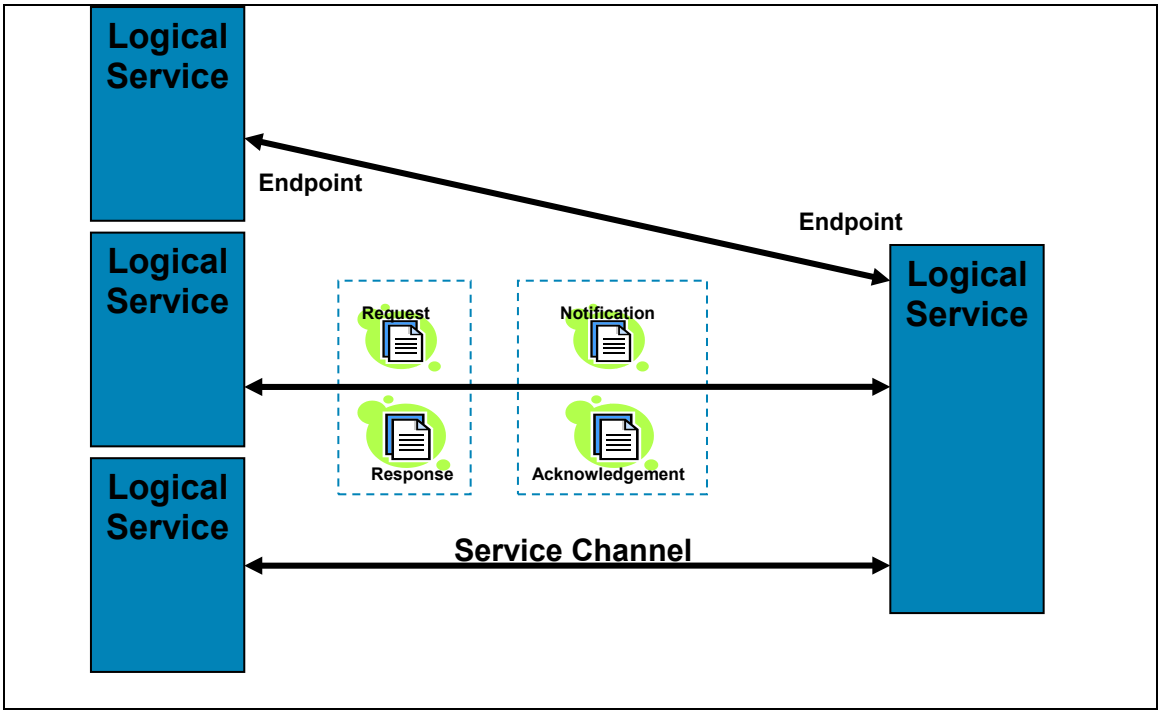

Figure 3: Service Channel Illustrated

A service channel is the message communication path between two logical services which exists for one of the following durations: the duration marked by the first message of a service registration and ending with the last message of service deregistration or the single request/response message transaction where the messages are permitted outside the registration-established service channel scope.

Once successfully constructed, the registration-established service channel ***shall*** remain active until terminated via deregistration. Normal termination of a service channel ***shall***

occur via a deregistration message exchange. Abnormal termination (e.g., a system failure, etc.) *may* require alternative deregistration mechanisms outside the scope of this specification. (Note: SCTE 130 Part 7 provides additional details. See [SCTE130-7] for supplementary information.)

Since an SCTE 130 message element is the XML document root element, all SCTE 130 messages ***shall*** share a common base syntactic structure as defined in the following sections.

9.1 Common Schema for All Messages

All SCTE 130 message XML elements ***shall*** share a common attribute definition though individual messages *may* contain additional attributes. Refer to the individual message schemas for specific details. Figure 4 illustrates the SCTE 130 message common attribute schema.

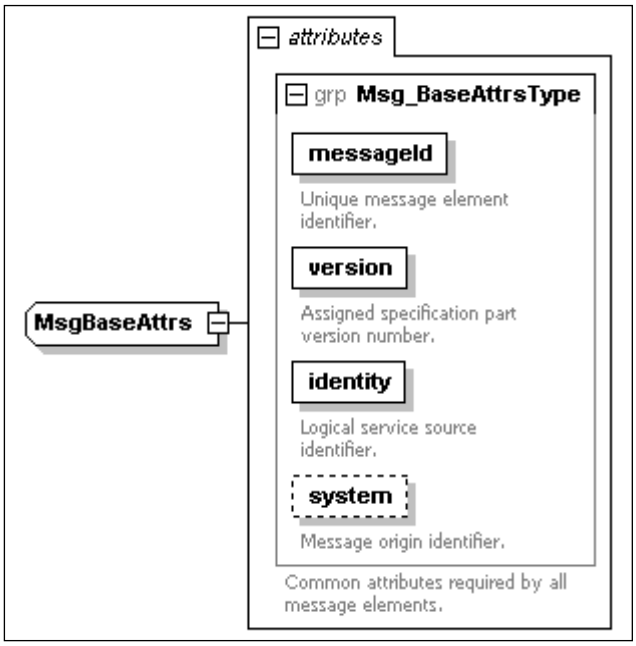

Figure 4: SCTE 130 Message Common Attribute Schema

A single SCTE 130 message ***shall*** be identifiable via its @messageId attribute. The individual SCTE 130 message specification version ***shall*** be identified via the @version attribute. The logical service message origin ***shall*** be specified using the @identity attribute and a further refinement *may* be provided via the @system attribute.

9.1.1 Semantic Definitions for the SCTE 130 Message Common Attributes

@messageId [Required, messageIdAttrType]—The message identifier. Every SCTE 130 message instance ***shall*** have a service channel unique value. The

@messageId *should* be a Universally Unique Identifier as defined by RFC 4122 (see [RFC4122]). See Section 11.2.4 for additional information.

@version [Required, versionAttrType]—The SCTE 130 message specification version number. The value ***shall*** be specified by the individual SCTE 130 specification parts. For all messages defined herein, the value ***shall*** be "1.2". See Section 11.2.9 for the attribute's type definition.

@identity [Required, identityAttrType]—The origin logical service identifier. A globally unique identifier associated with the logical service. See Section 11.2.1 for additional information.

@system [Optional, systemAttrType]—The message source identifier. See Section 11.2.8 for additional information.

9.2 Request and Notification Messages Base Schema

Figure 5 and Figure 6 illustrate the SCTE 130 request and notification messages base schema.

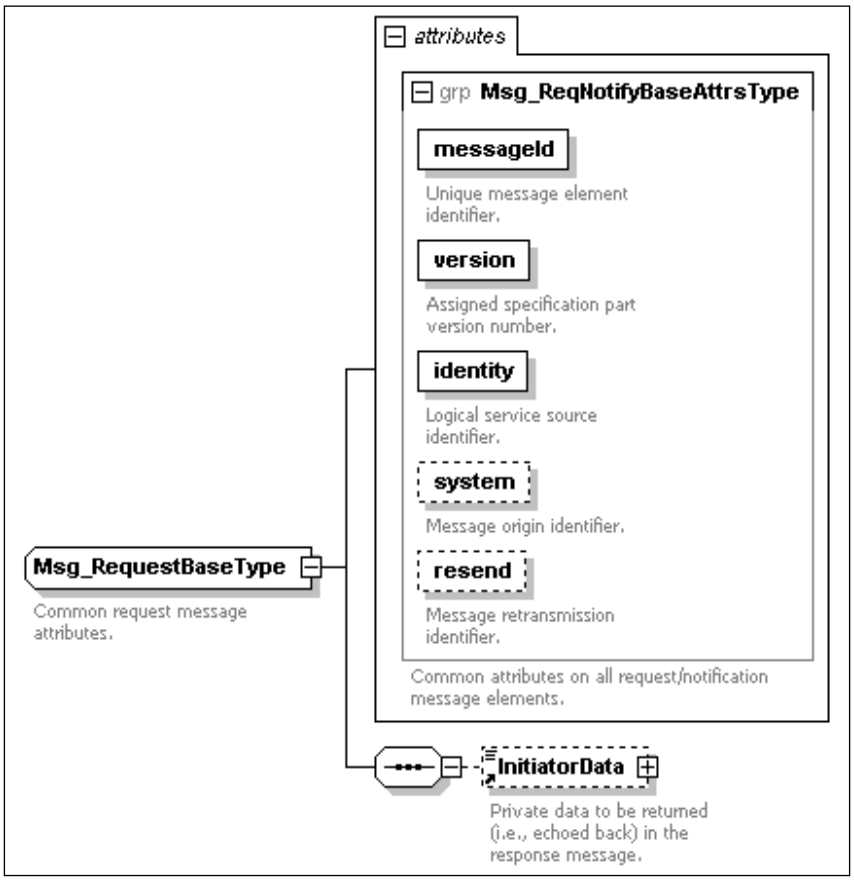

Figure 5: Request Message Base Schema

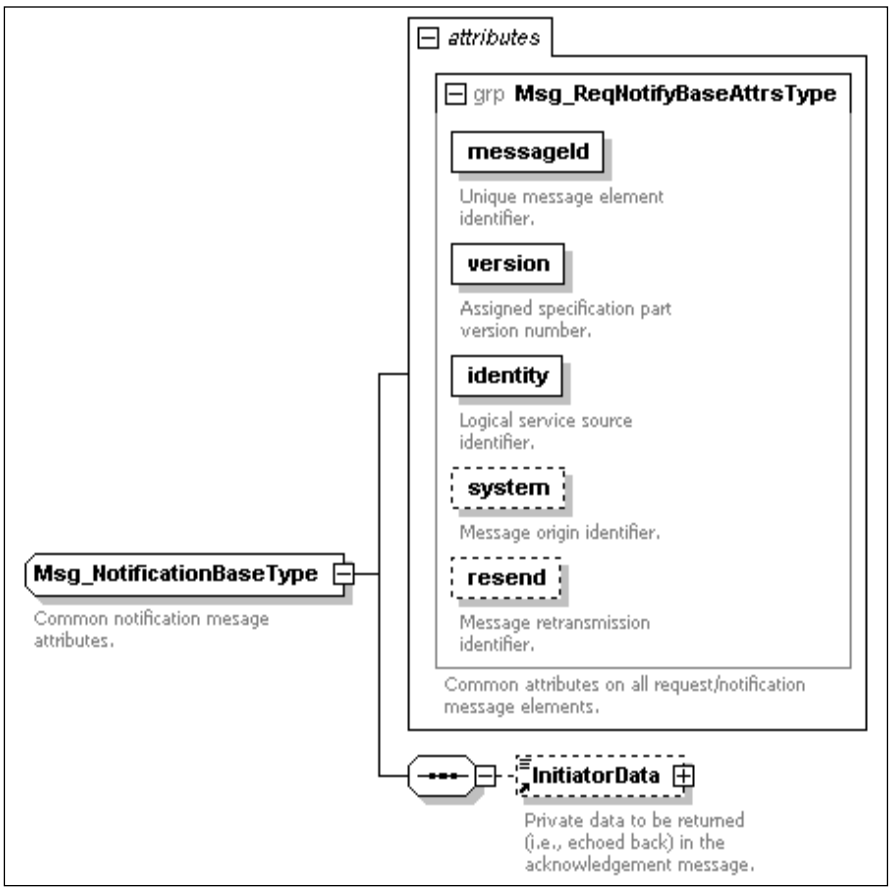

Figure 6: Notification Message Base Schema

9.2.1 Semantic Definitions for the Request and Notification Messages Base Schema

@messageId [Required, messageIdAttrType]—The message identifier. See Section 9.1.1 for additional information.

@version [Required, versionAttrType]—The message specification version. See Section 9.1.1 for additional information.

@identity [Required, identityAttrType]—The origin logical service identifier. See Section 9.1.1 for additional information.

@system [Optional, systemAttrType]—The message source identifier. See Section 9.1.1 for additional information.

@resend [Optional, resendAttrType]—The message identifier of a previously sent original message (i.e., the @messageId of a previous message) for which this message is a retransmit. See Section 9.4.7 and Section 11.2.7 for additional information.

InitiatorData [Optional]—The InitiatorData element contains implementation specific private data. If the element is present in the request or notification

message, the corresponding response or acknowledgement message ***shall*** include an exact copy of the original element. Figure 9 illustrates the linkage. If the InitiatorData element is omitted from the original request or notification message, the element ***shall not*** be present in the corresponding response or acknowledgement message. See Section 11.14 for additional information.

9.3 Response and Acknowledgement Messages Base Schema

Figure 7 and Figure 8 illustrate the SCTE 130 response and acknowledgement messages base schema.

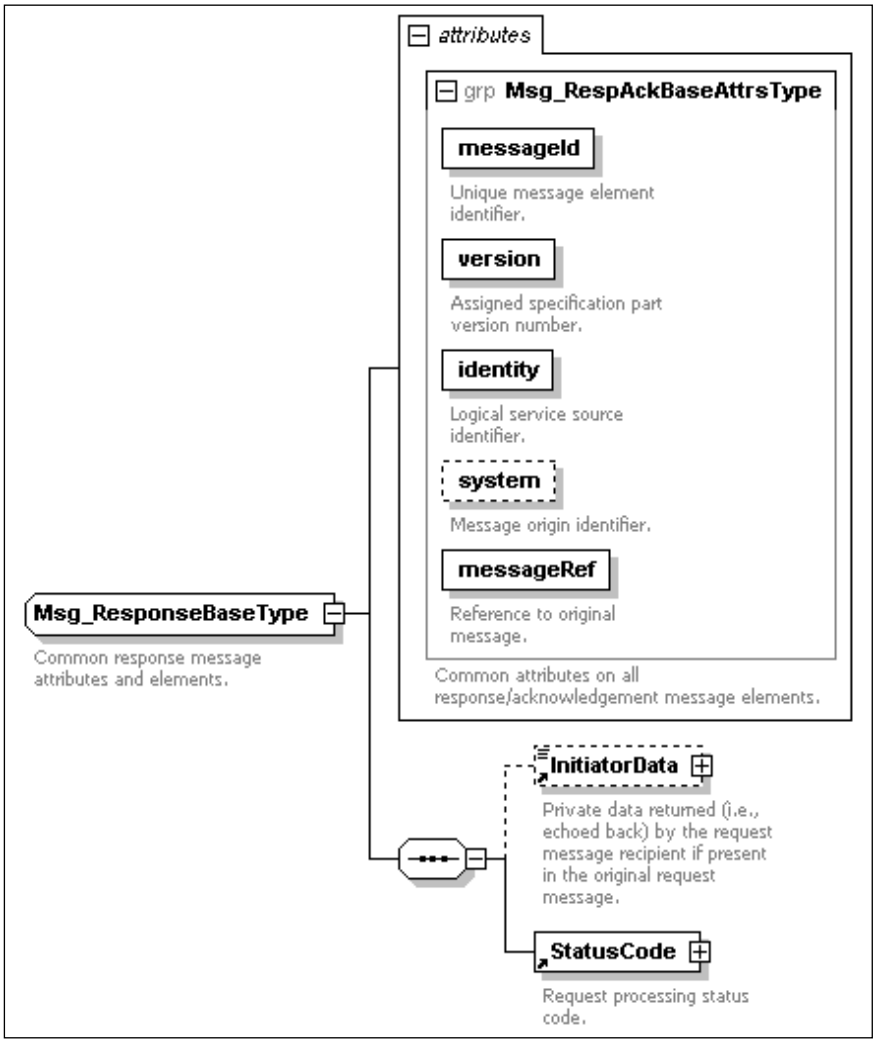

Figure 7: Response Message Base Schema

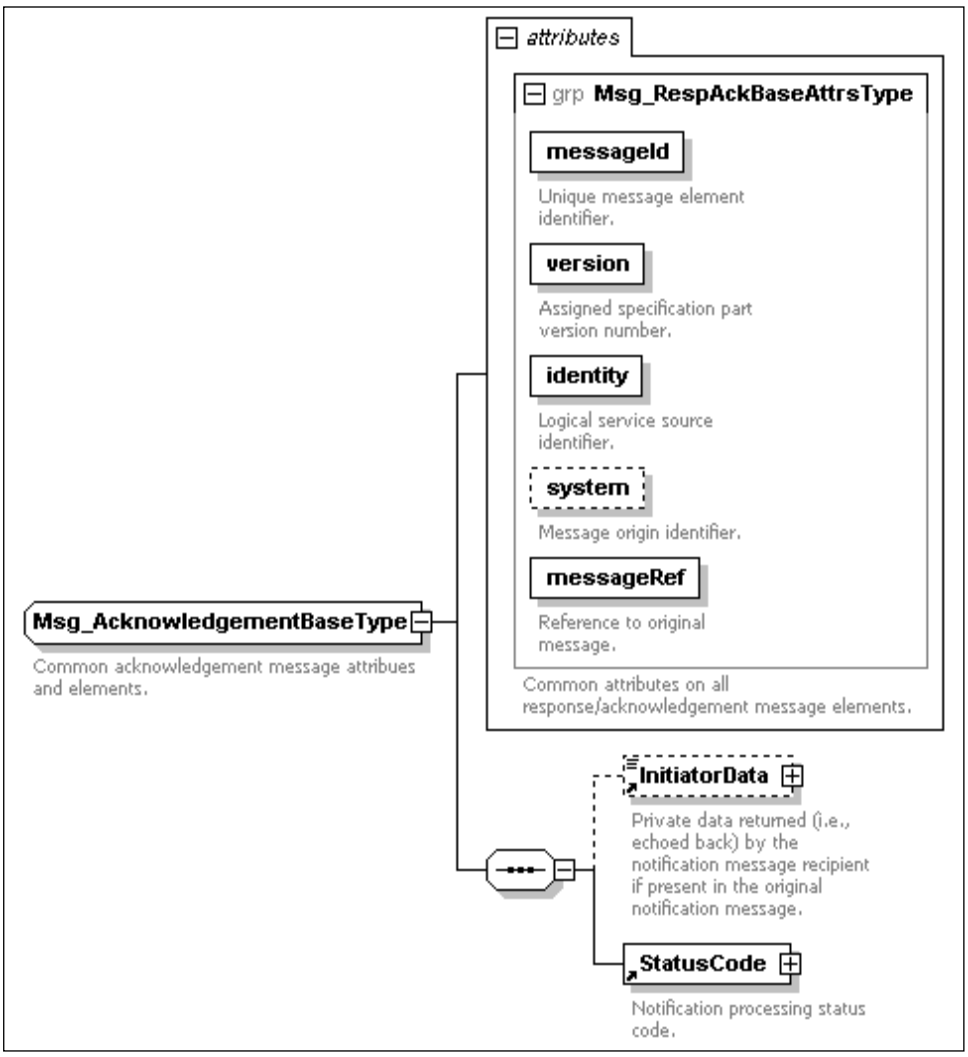

Figure 8: Acknowledgement Message Base Schema

9.3.1 Semantic Definitions for the Response and Acknowledgement Messages Base Schema

@messageId [Required, messageIdAttrType]—The message identifier. See Section 9.1.1 for additional information.

@version [Required, versionAttrType]—The message specification version. See Section 9.1.1 for additional information.

@identity [Required, identityAttrType]—The origin logical service identifier. See Section 9.1.1 for additional information.

@system [Optional, systemAttrType]—The message source identifier. See Section 9.1.1for additional information.

@messageRef [Required, messageRefAttrType]—An attribute in all response and acknowledgement messages which references the paired request or notification message. The attribute's value is the request or notification message's @messageId attribute value from the paired SCTE 130 message. See Section 11.2.5 for additional information. Figure 9 illustrates the linkage.

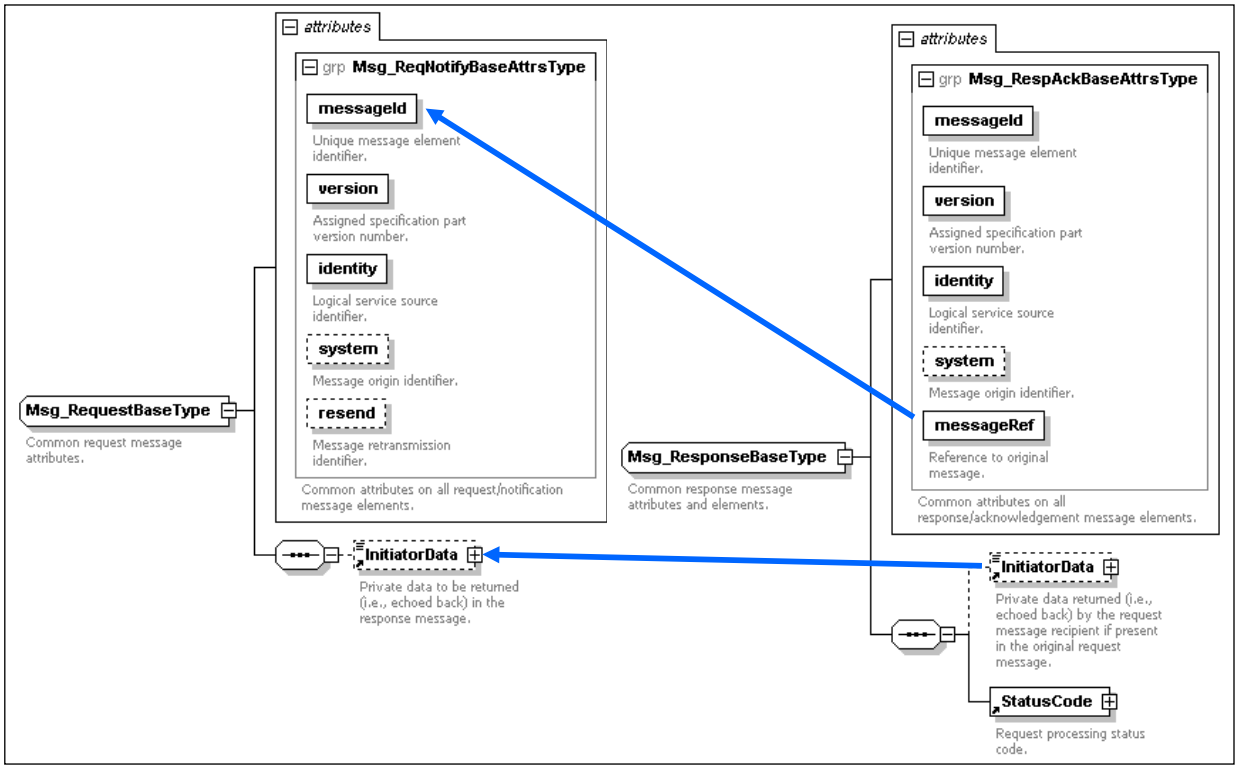

Figure 9: @messageRef and InitiatorData Paired Message Linkage

InitiatorData [Optional]—The InitiatorData element ***shall*** be present if the initiating request or notification message contained the element and the element ***shall*** be an exact copy of the original element (i.e., the element is being echoed back). Figure 9 illustrates the linkage. If the InitiatorData element is omitted from the original request or notification message then the element ***shall not*** be present in the corresponding response or acknowledgement message. See Section 11.14 for additional information.

StatusCode [Required]—An applicable processing status code. See Section 11.19 for additional information.

9.4 SCTE 130 Message Characterizations

9.4.1 Transport Mechanisms

SCTE 130 requires a reliable transport for delivery of all messages. This document does not define the specific mechanism or protocol for transporting messages nor does it restrict the choice of transport mechanisms in any way. (Note: SCTE 130 Part 7 [SCTE130-7] defines the transport details.)

SCTE 130 does not expressly require, prohibit, or mandate that a message exchange occur directly between two SCTE 130 logical services. Messages *may* be passed and/or routed through intermediaries as long as such activity does not compromise the SCTE 130 message specification defined herein or any other SCTE 130 specification part.

When there is an intermediary system between a pair of logical services, such as illustrated in Figure 10, there ***shall*** be two independent service channels. One service channel ***shall*** exist between the original requesting system and the intermediary; the second service channel ***shall*** exist between the intermediary and the original responding system. The actions taken within the intermediary system are outside the scope of this specification and there is no relationship between the two service channels within the context of this specification (i.e., this specification does not impose restrictions or constraints on the actions taken by the intermediary outside of the service channels). However, the intermediary ***shall*** ensure that each service channel follows the specification requirements.

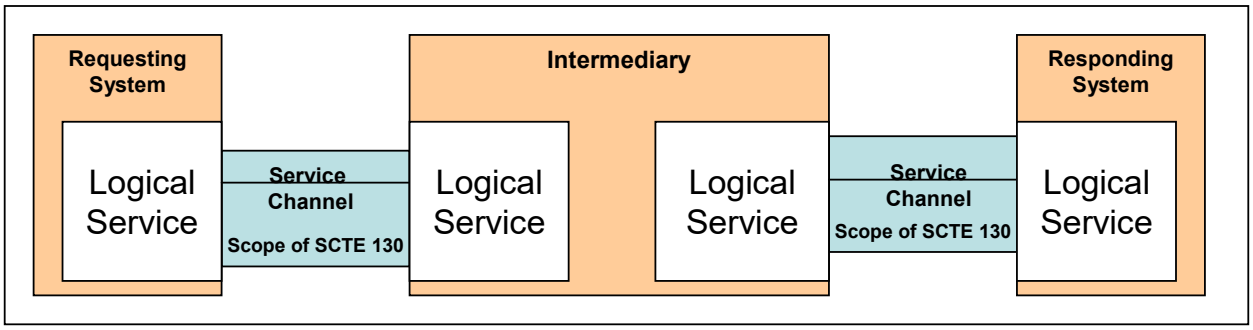

Figure 10: Intermediary Logical Service

9.4.2 Message Order

This document does not define any mechanisms for sending or processing messages in a specific order. It is the responsibility of the message sending system and the selected transport to send them in such a manner as to facilitate processing in an appropriate order. (Note: SCTE 130 Part 7 [SCTE130-7] defines transport details.)

9.4.3 Multiple Messages

As supported by the transport protocol, a message initiator or message respondent *may* have multiple messages to the same logical service endpoint outstanding or in transit simultaneously. Consequently, there *may* be multiple

messages “in flight” concurrently. (Note: SCTE 130 Part 7 [SCTE130-7] defines the transport details.)

9.4.4 Message Timeliness

This document does not specify a maximum response or acknowledgement time limit. SCTE 130 Part 7 [SCTE130-7] *may* specify a transport specific timeout value. Though SCTE 130 requires a reliable transport, implementations *should* be prepared to handle the case when no paired response or acknowledgement message returns in a timely manner. Subsequent error behavior and recovery is outside the scope of this specification.

9.4.5 Message Specification Versioning

The specific SCTE 130 specification version for an SCTE 130 message ***shall*** be identified as a combination of both the SCTE 130 part specific XML namespace (which contains a revision date identifier) and the SCTE 130 message’s @version attribute. Using this identification schema, forward compatible transforms *may* be defined in order to facilitate future specification compatibility.

9.4.6 Message Error Handling

This document defines a single mechanism for communicating request or notification message processing errors via the appropriate paired response message. The paired response or acknowledgment message describes the original message processing result which includes message handling, execution, and processing. The result is supplied via the StatusCode element and it *may* specify one or more errors, warnings, or informational descriptions via the Note elements.

In the case of an error in the response or acknowledgement message, the response or acknowledgement message receiver *may* optionally use the service status message exchange (i.e., ServiceStatusNotification and ServiceStatusAcknowledgement) to inform the response/acknowledgment sender of an error (for example a malformed response or acknowledgement message). The response/acknowledgement message receiver is not required to inform the sender of an error.

Additionally, alternate reporting mechanisms *may* be defined by the individual specification parts for specific cases (such as the SCTE 130 Part 3 PlacementStatusNotification message—see [SCTE130-3]). Refer to the individual specifications for additional information.

Asynchronous or unsolicited event reporting (including error, warning, or informational events) not occurring as the direct result of request or notification message processing ***shall*** be signaled using the service status message exchange

(i.e., a ServiceStatusNotification and ServiceStatusAcknowledgement message transaction).

9.4.7 High Availability and Message Retransmission

Every logical service endpoint ***shall*** adhere to and implement the capabilities described in this section in support of intra-service high availability (HA) regardless of whether or not the endpoint implements a high availability feature set. By mandating support of these capabilities, the specification enables HA and non-HA implementations to interoperate.

Every message ***shall*** support an optional, opaque InitiatorData element. (See Section 11.14 for additional information on the InitiatorData element.) The InitiatorData element *may* be used to carry arbitrary data which ***shall*** be populated by the sender of the request or notification message and echoed back by the respondent in the corresponding response or acknowledgment message. The respondent's InitiatorData element ***shall*** be an exact copy of the InitiatorData element from the corresponding request or notification message when present in the paired message. The respondent ***shall*** include the InitiatorData element in any response or acknowledgement message where the corresponding request or notification message contained an InitiatorData element.

A request or notification message *may* contain an @resend attribute, which indicates that this message is a retransmission of a previous message. The @resend attribute value ***shall*** contain the originally transmitted message's @messageId attribute value. See Section 11.2.7 for additional information on the @resend attribute. The retransmitted response or acknowledgement message ***shall*** have its @messageRef attribute set to the retransmitted request or notification message's @messageId attribute value and this attribute ***shall not*** be set to the @resend attribute value (i.e., the paired response message's @messageRef ***shall*** always be the @messageId attribute value of its paired mate).

The following sections describe the message initiator and message respondent capabilities and constraints.

9.4.7.1 Message Initiator

The message pair initiator ***shall*** be responsible for determining the appropriate course of action *should* the message pair's response or acknowledgement message not be received by the initiator in a timely manner. The definition of "received in a timely manner" is implementation dependent and outside the scope of this specification. Typically, the initiator *may* choose between the following two actions if the response or acknowledgement message is not received:

- Retransmit the request or notification message

The initiator ***shall*** retransmit the request or notification message after an implementation dependent timeout. The request or notification message's @resend attribute ***shall*** contain the original message's @messageId attribute value for which this message is a retransmission. This attribute's presence in combination with its value allows for resend detection and correlation at the recipient. The other elements of the retransmitted message *should* be the same, but *may* differ due to changes in the initiator between the original transmission and the subsequent re-transmission (e.g., initiator failover, time elapsing, etc.). The message recipient's response is implementation specific and outside the scope of this standard. However, some behavioral constraints and possible implementation choices are outlined in Section 9.4.7.2.

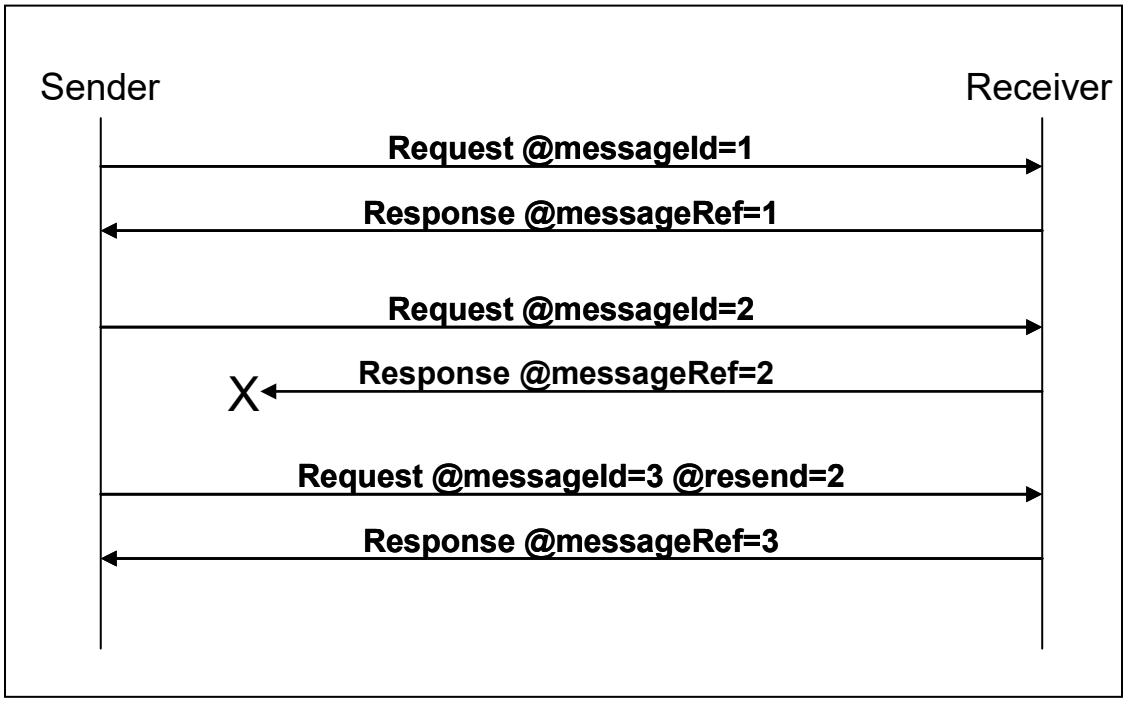

Figure 11: Example Resend Message Sequence

- Abandon the request or notification message

The initiator ***shall*** behave as if the message was never sent. The initiator ***shall*** be responsible for handling any subsequent messages for the abandoned message pair including, if necessary, initiating additional message pair(s) to signal the abandonment upon receipt of a message corresponding to the abandoned message pair.

For example, an ADM sends an adm:PlacementRequest message and the ADS responds with an adm:PlacementResponse message. The ADM does not receive the adm:PlacementResponse message and subsequently decides to abandon the placement operation. The ADM, as part of its abandonment processing, *should* send an adm:PlacementStatusNotification indicating termination (i.e. send an "endAll"). However before the ADM completes the abandonment signaling, the ADS sends an adm:PlacementUpdateNotification message to the ADM referring to the abandoned adm:PlacementRequest message. The ADM *should* respond with a status of success as the "endAll" notifies the ADS of the abandonment.

9.4.7.2 Message Respondent

Each logical service endpoint ***shall*** support receipt of a retransmitted request or notification message when functioning as the respondent in a logical service pair. Upon detection of a retransmitted request or notification message via the @resend attributes presence, the respondent ***shall*** transmit the proper implementation specific response applicable to the received retransmission message. (Possible meanings for the phrase "proper implementation specific response" are covered in subsequent paragraphs herein.)

The respondent's retransmit message processing ***shall not*** cause the respondent to maintain two or more states – one for the original message and one for each received retransmitted message. (In this case, state means data within the respondent directly related to the request or notification message processing.) The respondent ***shall not*** initiate new state if previous state exists, but rather replace or update the existing state based on processing the received request or notification message. The key requirement is that any state maintained by the respondent from the original message processing ***shall*** be replaced or updated by the state resulting from subsequent processing of the retransmitted request or notification message.

More specific handling of a retransmitted message is outside the scope of the specification; however, some of the possible choices include:

- Existing state not found by the respondent, return error with the StatusCode element's @detail attribute having a value "unknown message reference."
- Existing state not found by the respondent. Treat the message as a new request or notification message and process it accordingly including sending the appropriate response or acknowledgement message.

- Existing state found, update or replace the state based on processing of the retransmitted request or notification message and return the appropriate equivalent processed response or acknowledgment message. The retransmitted response or acknowledgement message *should* be similar to, but ***shall not*** be required to be an exact copy of the original response or acknowledgement message. Changes in the retransmitted request or notification message and/or changes in the respondent *may* cause a different response or acknowledgment message to be produced.

  For example, the receipt of a retransmitted adm:PlacementRequest message results in the transmission of an adm:PlacementResponse message appropriate to the received adm:PlacementRequest message's contents. The new adm:PlacementResponse message *should* be similar to, but is not be required to be, an exact copy of the original adm:PlacementResponse message. In particular, it *should* be noted that the resent adm:PlacementRequest message *may* not be an exact copy of the original and thus, *may* require a different adm:PlacementResponse and/or the state of the respondent *may* have changed requiring a change in the adm:PlacementResponse message. In all cases, the adm:PlacementResponse message ***shall*** be valid in the context of the retransmitted adm:PlacementRequest. Again, the key requirement is that any state maintained by the respondent from the original adm:PlacementRequest message ***shall*** be replaced or updated by the state resulting from processing the retransmitted adm:PlacementRequest message.

- Existing state found, remove state and return error with the StatusCode element's @detail attribute having a value "resend forced abandonment."

In the case of an error return (i.e., the StatusCode element's @statusCode attribute is set to error and the @detail attribute is set to one of the above values or an appropriate value from Table 6), the respondent ***shall not*** maintain any state related to either the original or the retransmitted message. The message initiator, upon receipt of the previously referenced error codes, ***shall*** abandon further attempts to retransmit the message and ***shall*** consider the message exchange in error. Thus, both the message initiator and the message respondent are in sync since neither has state related to the original or retransmitted messages.

9.4.8 List Registration

For all SCTE 130 specification parts, the list registration function ***shall*** always return the active or accepted registration message(s) which *may* be a

retransmitted version of the original registration message. The list function respondent ***shall not*** be required to return an exact or identical copy of the original registration message (i.e., the returned data ***shall not*** be required to be an exact copy of the "on the wire" registration message). The list function respondent *may* return a syntactically and semantically valid recoded version of the original registration message. The recoded message, which ***shall*** be returned as an element in the list registration response message, ***shall*** include all the original information including all present attributes and/or elements and all elements contained within the Ext element. The returned message element *may* optionally include XML style comments from the active registration message.

Registration messages ***shall*** always be referenced by the original registration message's @messageId attribute (i.e., the lookup identifier ***shall*** always be the original message's @messageId attribute value which in a retransmitted registration message ***shall*** be the @resend attribute's value). When identifying a specific message via the @registrationRef attribute in the list registration and deregister functions, the @registrationRef attribute's value ***shall*** always be the original registration message's @messageId attribute value. Thus, the same identifier ***shall*** be used for all functions regardless of retransmission processing.

9.5 Addressing

SCTE 130 uses common, well-defined address formats whenever possible. The following sections describe these formats.

9.5.1 Internet Protocol version 4 (IPv4) Address

An Internet Protocol (IP) version 4 address ***shall*** conform to the format and characteristics as defined by RFC 3986. See [RFC3986] for additional details. Typically, the address is represented in dot decimal notation (also known as dotted quad notation, i.e., nnn.nnn.nnn.nnn). Example 1 illustrates a few valid instances.

```
A basic IPv4 address.
171.70.222.82

An IPv4 address in a URL.
http://171.70.22.82
```

Example 1

9.5.2 Internet Protocol version 6 (IPv6) Address

An Internet Protocol (IP) version 6 address ***shall*** conform to the format and characteristics as defined by RFC 3986. See [RFC3986] for additional details. Typically, the address is written as eight groups of four hexadecimal digits having the form of xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx, where each x

represents a single hex character of the 128-bit address. Per RFC 3986, use of the standard "::" *may* be used to suppress repeating zeros. Example 2 illustrates a few valid instances. See [RFC3986] for additional examples.

The following 6 addresses are all equivalent. 2001:0db8:0000:0000:0000:0000:1428:57ab 2001:0db8:0000:0000:0000::1428:57ab 2001:0db8:0:0:0:0:1428:57ab 2001:0db8:0:0::1428:57ab 2001:0db8::1428:57ab 2001:db8::1428:57ab An IPv6 address in a URL. (Note the required brackets around the IPv6 address.) http://[2001:0db8:85a3:08d3:1319:8a2e:0370:7344]

Example 2

9.5.3 IPv4 and IPv6 Port Identifier

An IPv4 or an IPv6 address *may* optionally include a port identifier as part of the address. Port identification ***shall*** conform to the format and characteristics as defined by RFC 3986. See [RFC3986] for additional details. The individual specifications ***shall*** indicate if a port value is optional or required as appropriate. Example 3 illustrates a few valid instances.

An IPv4 address with a port identifier. 171.70.222.82:8080 An IPv4 address in a URL that includes a port identifier. http://171.70.222.82:8080 An IPv4 address in a URL that includes a port identifier and an endpoint. http://64.13.147.67:8080/scte130/index.jsp An IPv6 address with a port identifier. [2001:0db8:85a3:08d3:1319:8a2e:0370:7344]:443 An IPv6 address included in a URL that includes a port identifier. https://[2001:0db8:85a3:08d3:1319:8a2e:0370:7344]:443 An IPv6 address included in a URL that includes a port identifier and an endpoint. https://[2001:0db8:85a3:08d3:1319:8a2e:0370:7344]:443/scte130/index.jsp

Example 3

9.5.4 IEEE Media Access Control (MAC) Address

A 6-byte IEEE MAC address ***shall*** conform to the format and characteristics of hexadecimal representation per IEEE 802. See [IEEE802] for the hexadecimal representation definition and additional information. The value format is xx-xx-xx-xx-xx-xx, where each x represents a single hex character (0 through 9 and 'A' through 'F' or 'a' through 'f'). The uppercase hexadecimal digits 'A' through 'F' are equivalent to the lowercase digits 'a' through 'f', respectively. Each pair of hex characters, which represent an octet, are separated by a hyphen. (Note: The required separator ***shall*** be a hyphen and ***shall not*** be a colon per IEEE 802.) Leading zeros ***shall not*** be omitted. Example 4 illustrates a few valid instances.

The following four MAC addresses are all equivalent. 08-00-69-02-01-FC 08-00-69-02-01-fc 08-00-69-02-01-Fc 08-00-69-02-01-fC

Example 4

10.0 SCTE 130 PART 2 MESSAGES

Table 2 identifies the SCTE 130 Part 2 messages defined herein that ***shall*** be supported and implemented by all SCTE 130 logical services.

Message	Description
ServiceCheckRequest	Request for peer or endpoint health
ServiceCheckResponse	Response containing current health state
ServiceStatusNotification	Event notification
ServiceStatusAcknowledgement	Event notification receipt and processing confirmation

Table 2: SCTE 130 Part 2 Messages

10.1 Service Check Messages

Any SCTE 130 logical service *may* check on its peer's or an endpoint's health by sending a ServiceCheckRequest message at any time. The ServiceCheckRequest message *may* be sent to any endpoint declared in a Callback element using an Address element. The respondent ***shall*** return a ServiceCheckResponse message providing its operational status using the StatusCode element. Additional descriptive information *may* be supplied using the Note elements of the StatusCode element. Figure 12 illustrates this message exchange.

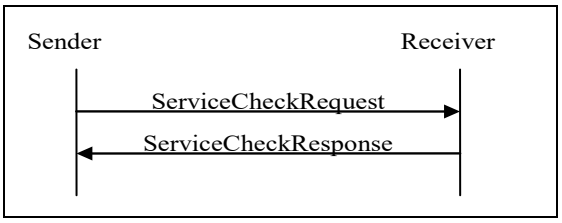

Figure 12: Service Check Message Exchange

A ServiceCheckRequest *may* be sent to any endpoint at any time including during an active registration request (i.e., the message *may* be sent to any endpoint declared using an Address element in the Callback element). Thus, any logical service registering with another logical service using the SCTE 130 part specific registration process ***shall*** be prepared to receive the ServiceCheckRequest message and to respond with a ServiceCheckResponse message. The ServiceCheckRequest and ServiceCheckResponse message schemas are defined in the following sections.

10.1.1 ServiceCheckRequest Message Schema

Figure 13 illustrates the ServiceCheckRequest message schema.

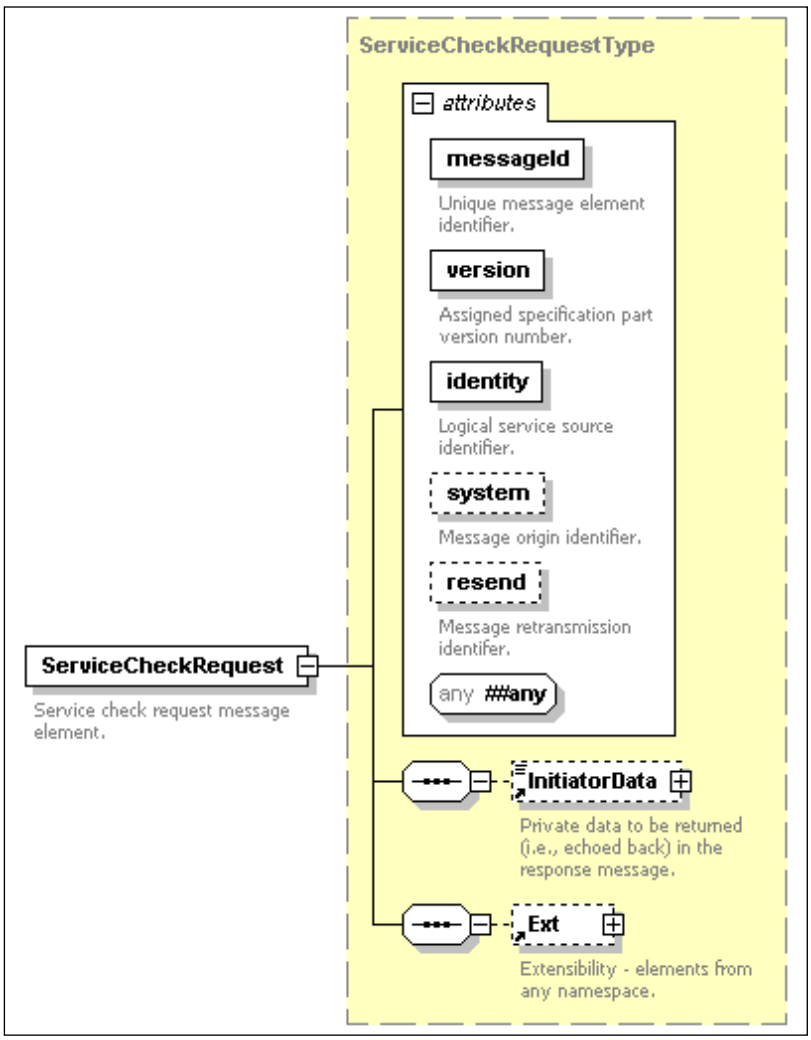

Figure 13: ServiceCheckRequest Message Schema

The ServiceCheckRequest message *may* be an empty element.

10.1.1.1 Semantic Definitions for the ServiceCheckRequest Message

@messageId [Required, messageIdAttrType]—The message identifier. See Section 9.1.1 for additional information.

@version [Required, versionAttrType]—The message specification version. See Section 9.1.1 for additional information.

@identity [Required, identityAttrType]—The origin logical service identifier. See Section 9.1.1 for additional information.

@system [Optional, systemAttrType]—The message source identifier. See Section 9.1.1 for additional information.

@resend [Optional, resendAttrType]—The message is a retransmit of a previous message identified by the supplied message identifier. See Section 9.2.1 for additional information.

@##any [Optional]—Any additional attribute from any namespace.

InitiatorData [Optional]—The InitiatorData element contains implementation specific private data which ***shall*** be returned in the ServiceCheckResponse message. See Section 9.2.1 for additional information.

Ext [Optional]—A container for any additional elements from any namespace. See Section 11.12 for additional information.

10.1.2 ServiceCheckResponse Message Schema

Figure 14 illustrates the ServiceCheckResponse element schema sent following the reception and processing of a ServiceCheckRequest message.

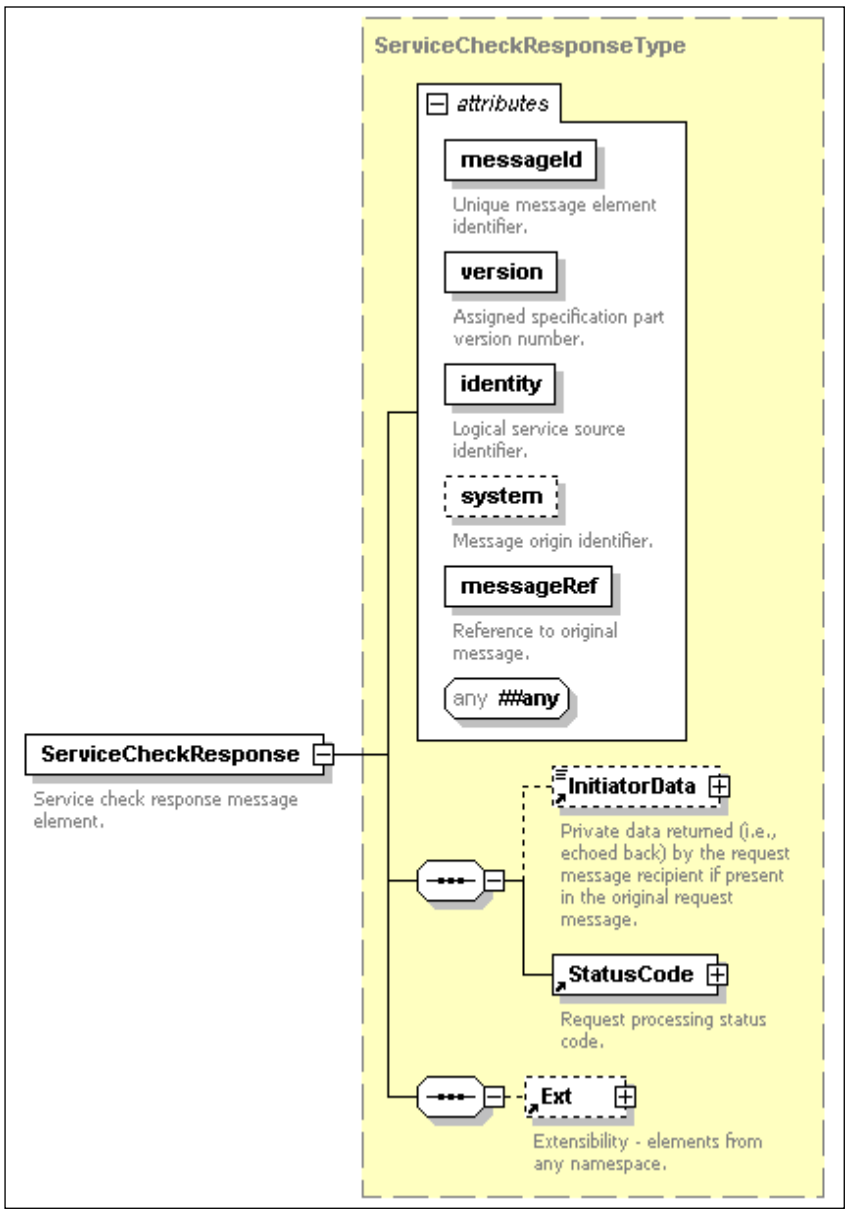

Figure 14**: ServiceCheckResponse Message Schema**

10.1.2.1 Semantic Definitions for the ServiceCheckResponse Message

@messageId [Required, messageIdAttrType]—The message identifier. See Section 9.1.1 for additional information.

@version [Required, versionAttrType]—The message specification version. See Section 9.1.1 for additional information.

@identity [Required, identityAttrType]—The origin logical service identifier. See Section 9.1.1 for additional information.

@system [Optional, systemAttrType]—The message source identifier. See Section 9.1.1 for additional information.

@messageRef [Required, messageRefAttrType]—A reference to the SCTE 130 ServiceCheckRequest message element initiating this message exchange. The value ***shall*** be the ServiceCheckRequest message's @messageId attribute value. See Section 9.3.1 for additional information.

@##any [Optional]—Any additional attribute from any namespace.

InitiatorData [Optional]—The InitiatorData element ***shall*** be an exact copy of the ServiceCheckRequest message's InitiatorData element and ***shall*** only be present if found in the paired request message. See Section 9.3.1 for additional information.

StatusCode [Required]—An applicable processing status code specific to the ServiceCheckRequest message processing. See Section 11.19 for additional information. The StatusCode element ***shall*** contain the service health status.

Ext [Optional]—A container for any additional elements from any namespace. See Section 11.12 for additional information.

10.1.3 Service Check Message Examples

A ServiceCheckRequest message (Example 5) with a positive (i.e., success) ServiceCheckResponse message (Example 6).

```
<ServiceCheckRequest messageId="8FA2C7BD-2751-4D18-A2C6-372D8CE5F100"
version="1.2" identity="ADMLogicalService_1170FEF4-C19B-450E-B624-
421B23F525F4" system="adm"/>
```

Example 5

```
<ServiceCheckResponse messageId="8FA2C7BD-2751-4D18-A2C6-372D8CE5F105"
version="1.2" identity="ADSLogicalService_1170FEF4-C19B-450E-B624-
421B23F525F5" system="ads" messageRef="8FA2C7BD-2751-4D18-A2C6-
372D8CE5F100">
  <StatusCode class="0"/>
</ServiceCheckResponse>
```

Example 6

A negative ServiceCheckResponse message (Example 7) assuming the same request message as Example 5.

```
<ServiceCheckResponse messageId="8FA2C7BD-2751-4D18-A2C6-372D8CE5F106"
version="1.2" identity="ADSLogicalService_1170FEF4-C19B-450E-B624-
```

```
421B23F525F5" system="ads" messageRef="8FA2C7BD-2751-4D18-A2C6-
372D8CE5F100">
  <StatusCode class="2" detail="10">
    <Note>Warning. Network connection lost.</Note>
    <Note>Lost contact with the Content Information Service (CIS).</Note>
  </StatusCode>
</ServiceCheckResponse>
```

Example 7

A ServiceCheckRequest message (Example 8) with a positive (i.e., success) ServiceCheckResponse message (Example 9) illustrating the InitiatorData elements inclusion.

```
<ServiceCheckRequest messageId="8FA2C7BD-2751-4D18-A2C6-372D8CE5F130"
version="1.2" identity="ADMLogicalService_1170FEF4-C19B-450E-B624-
421B23F525F4" system="adm">
  <InitiatorData secret="true">Hidden secrets.</InitiatorData>
</ServiceCheckRequest>
```

Example 8

```
<ServiceCheckResponse messageId="8FA2C7BD-2751-4D18-A2C6-372D8CE5F135"
version="1.2" identity="ADSLogicalService_1170FEF4-C19B-450E-B624-
421B23F525F5" system="ads" messageRef="8FA2C7BD-2751-4D18-A2C6-
372D8CE5F130">
  <InitiatorData secret="true">Hidden secrets.</InitiatorData>
  <StatusCode class="0"/>
</ServiceCheckResponse>
```

Example 9

10.2 Service Status Messages

Any SCTE 130 logical service *may* at anytime notify its peers of a status or health change by sending a ServiceStatusNotification message. The respondent ***shall*** return a ServiceStatusAcknowledgement message indicating notification message receipt and providing a message processing status. Additional descriptive information *may* be supplied using Note elements in either message. Figure 15 illustrates this message exchange.

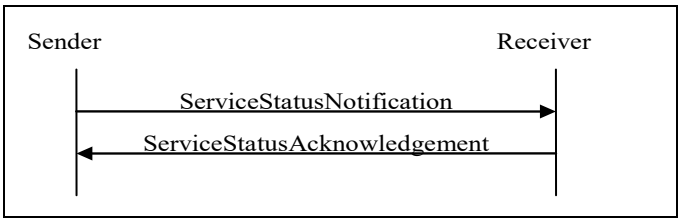

Figure 15: Service Status Message Exchange

The ServiceStatusNotification message provides a mechanism which minimally does the following:

- Provides unsolicited informational status
- Facilitates general event notification

The following sections illustrate the service status message schemas.

10.2.1 ServiceStatusNotification Message Schema

Figure 16 illustrates the ServiceStatusNotification message's schema.

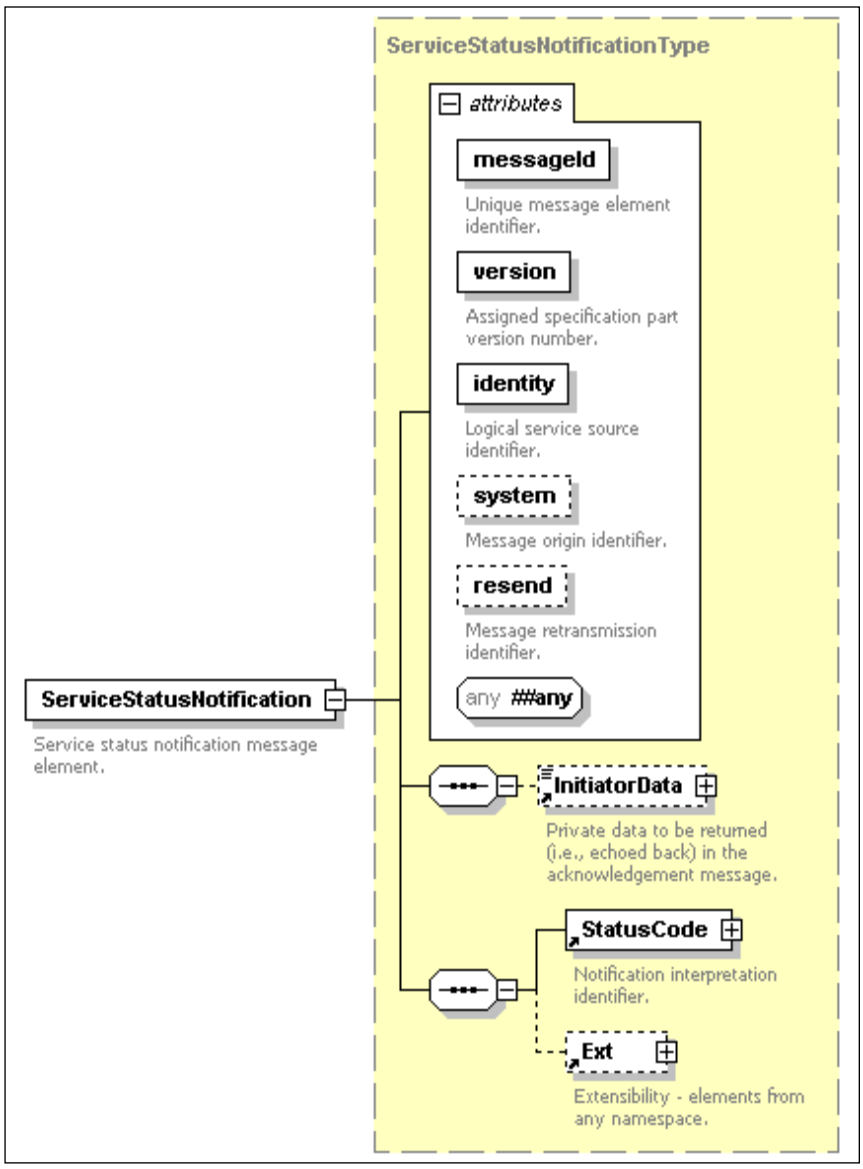

Figure 16: ServiceStatusNotification Message Schema

The ServiceStatusNotification message *may* be sent at any time facilitating unsolicited notifications. For example, notifications might include low disk space, a catastrophic hardware failure, or loss of communication with a separate subsystem such as the CIS. One or more Note elements *may* be included in the message providing detailed description text applicable to the supplied StatusCode element.

10.2.1.1 Semantic Definitions for the ServiceStatusNotification Message

@messageId [Required, messageIdAttrType]—The message identifier. See Section 9.1.1 for additional information.

@version [Required, versionAttrType]—The message specification version. See Section 9.1.1 for additional information.

@identity [Required, identityAttrType]—The origin logical service identifier. See Section 9.1.1 for additional information.

@system [Optional, systemAttrType]—The message source identifier. See Section 9.1.1 for additional information.

@resend [Optional, resendAttrType]—The message is a retransmit of a previous message identified by the supplied message identifier. See Section 9.2.1 for additional information.

@##any [Optional]—Any additional attribute from any namespace.

InitiatorData [Optional]—The InitiatorData element contains implementation specific private data which ***shall*** be returned in the ServiceStatusAcknowledgement message. See Section 9.2.1 for additional information.

StatusCode [Required]—An applicable notification status code specific to the ServiceStatusNotification message. See Section 11.19 for additional information.

Ext [Optional]—A container for any additional elements from any namespace. See Section 11.12 for additional information.

10.2.2 ServiceStatusAcknowledgement Message Schema

Figure 17 illustrates the ServiceStatusAcknowledgement element schema sent following the reception and processing of a ServiceStatusNotification message.

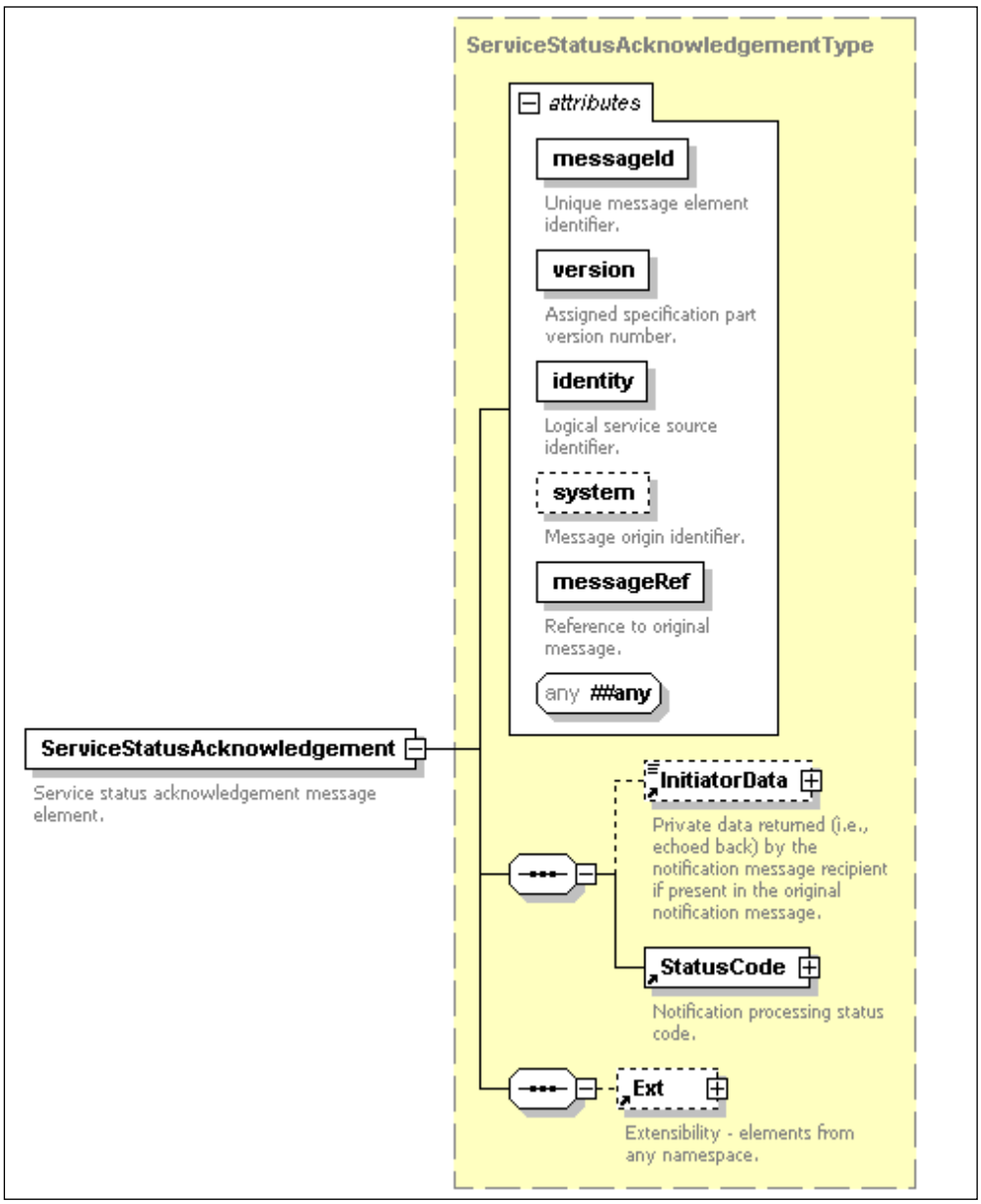

Figure 17: ServiceStatusAcknowledgement Message Schema

10.2.2.1 Semantic Definitions for the ServiceStatusAcknowledgement Message

@messageId [Required, messageIdAttrType]—The message identifier. See Section 9.1.1 for additional information.

@version [Required, versionAttrType]—The message specification version. See Section 9.1.1 for additional information.

@identity [Required, identityAttrType]—The origin logical service identifier. See Section 9.1.1 for additional information.

@system [Optional, systemAttrType]—The message source identifier. See Section 9.1.1 for additional information.

@messageRef [Required, messageRefAttrType]—A reference to the SCTE 130 ServiceStatusNotification message element initiating this message exchange. The value ***shall*** be the ServiceStatusNotification message's @messageId attribute value. See Section 9.3.1 for additional information.

@##any [Optional]—Any additional attribute from any namespace.

InitiatorData [Optional]—The InitiatorData element ***shall*** be an exact copy of the ServiceStatusNotification message's InitiatorData element and ***shall*** only be present if found in the paired notification message. See Section 9.3.1 for additional information.

StatusCode [Required]—An applicable status code specific to the ServiceStatusNotification message processing. See Section 11.19 for additional information.

Ext [Optional]—A container for any additional elements from any namespace. See Section 11.12 for additional information.

10.2.3 Service Status Message Examples

A notification (Example 10) and acknowledgement (Example 11) message sequence.

```
<ServiceStatusNotification messageId="87648E84-9D5E-11DB-96CA-005056C00009"
version="1.2" identity="ADMLogicalService_1170FEF4-C19B-450E-B624-
421B23F525F4" system="adm">
  <StatusCode class="3">
    <Note>Information.</Note>
    <Note>Low disk space.</Note>
  </StatusCode>
</ServiceStatusNotification>
```

Example 10

```
<ServiceStatusAcknowledgement messageId="87648E84-9D5E-11DB-96CA-
005056C0000a" version="1.2" identity="ADSLogicalService_1170FEF4-C19B-450E-
B624-421B23F525F5" system="ads" messageRef="87648E84-9D5E-11DB-96CA-
005056C00009">
  <StatusCode class="0"/>
</ServiceStatusAcknowledgement>
```

Example 11

An error notification message (Example 12) and acknowledgement message (Example 13).

```
<ServiceStatusNotification messageId="87648E84-9D5E-11DB-96CA-005056C0000C"
version="1.2" identity="ADSLogicalService_1170FEF4-C19B-450E-B624-
421B23F525F5" system="ads">
  <StatusCode class="1">
    <Note>Disk failure.</Note>
    <Note>The system is going down now. Bye-bye.</Note>
  </StatusCode>
</ServiceStatusNotification>
```

Example 12

```
<ServiceStatusAcknowledgement messageId="87648E84-9D5E-11DB-96CA-
005056C0000D" version="1.2" identity="ADMLogicalService_1170FEF4-C19B-450E-
B624-421B23F525F4" system="adm" messageRef="87648E84-9D5E-11DB-96CA-
005056C0000C">
  <StatusCode class="0"/>
</ServiceStatusAcknowledgement>
```

Example 13

11.0 SCTE 130 CORE ATTRIBUTE TYPES AND ELEMENTS

The following sections define the SCTE 130 core types including common attribute types and element definitions which *may* appear in any SCTE 130 XML element definition. The common attribute definitions *may* be utilized by any element including the root message elements.

11.1 Semantic Definitions for SCTE 130 Core Types

11.1.1 dateTimeTimezoneType Simple Type

dateTimeTimezoneType [xsd:dateTime]—This xsd:dateTime type ***shall*** include a timezone indicator as defined by [XMLSchemaP2].

11.1.2 nonEmptyStringType Simple Type

nonEmptyStringType [xsd:string]—This xsd:string type excludes an empty string as an acceptable value (i.e., string lengths ***shall*** be greater than zero).

11.2 Semantic Definitions for SCTE 130 Core Attribute Types

11.2.1 identityAttrType Attribute Type

identityAttrType [nonEmptyStringType]—This attribute type, typically referred to as the @identity attribute, represents a globally unique string identifying a set (or cluster) of system objects as a single logical service. The

identityAttrType *should* be a Universally Unique Identifier as defined by RFC4122 [RFC4122] and the string ***shall not*** be empty.

The logical service blocks of Section 9.0, Figure 2 and Figure 3, represent many possible different physical implementations. In the most basic form, a logical service block represents a single system or a single physical server as illustrated in Figure 18. In this case, the logical service block's named identity (i.e., the @identity attribute's value) and the physical system map one-to-one.

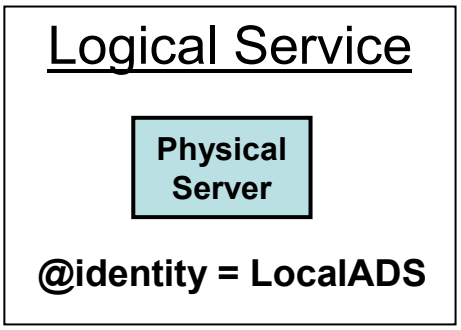

Figure 18: Basic @identity Attribute Mapping

More complicated implementations *may* be utilized and Figure 19 illustrates additional possible implementations but not all possible implementations.

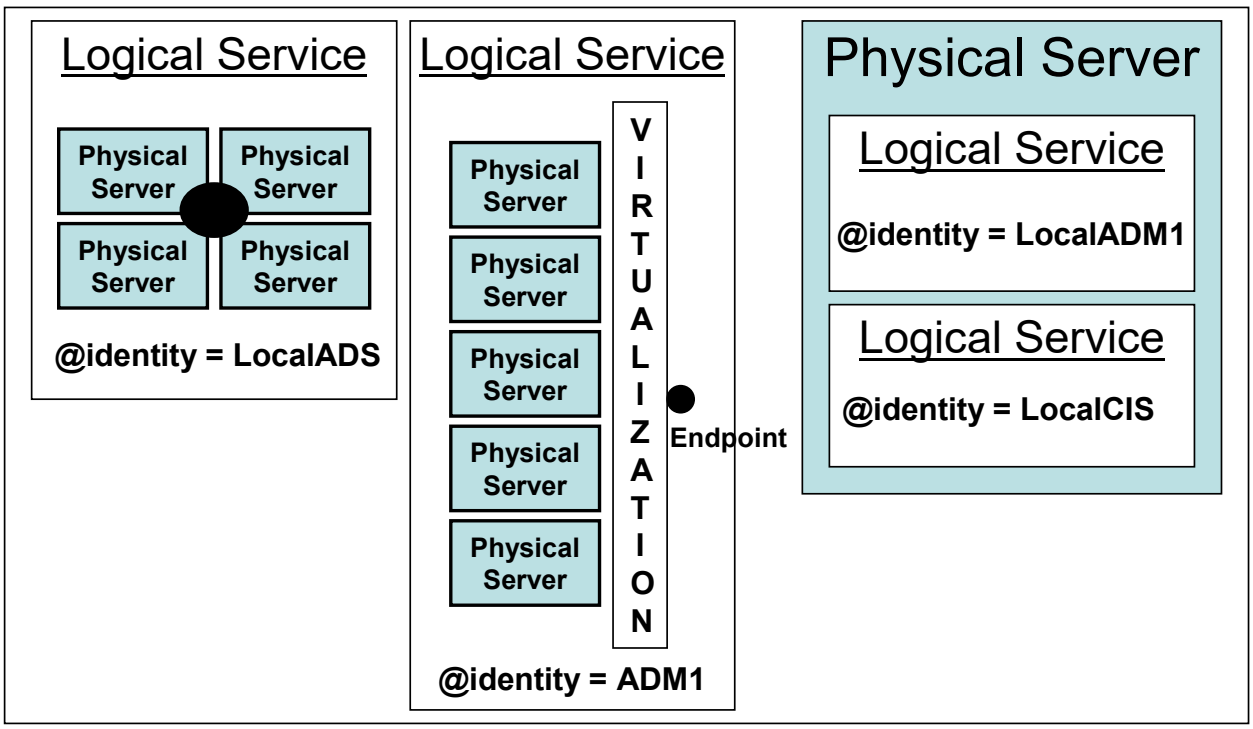

Figure 19: Complex @identity Attribute Mappings

In Figure 19, the @identity attribute value ***shall*** be the same for all physical systems/servers comprising a logical service. For example in the left hand side implementation where the logical service has the @identity attribute value of LocalADS, the physical servers are interconnected but the placement services *may* be assigned to a unique server. In the middle logical service identified as ADM1, the physical servers are located behind a virtualizing entity. The virtualization entity sources a single logical service endpoint for all the physical servers comprising the single logical service. At the far right of the illustration, the physical server is hosting two logical services identified as LocalADM1 and LocalCIS respectively. These are just examples and many other configurations and implementations are possible. The important concept to understand is that the @identity attribute identifies a logical service without reference to a physical implementation and it is the @system attribute that *may* be used to identify a unique server within the logical service. The identityAttrType virtualizes the implementation into single identifiable entity.

11.2.2 idAttrType Attribute Type

idAttrType [nonEmptyStringType]—This attribute type, generally used as the @id attribute, is a unique identifier for the element. The element ***shall*** have a unique value scoped by the element containing it (i.e., element uniqueness). The idAttrType *may* be a Universally Unique Identifier as defined by RFC 4122 [RFC4122] and the string ***shall not*** be empty.

11.2.3 mediaAvailableAttrType Attribute Type

mediaAvailableAttrType [xsd:boolean]—This attribute type, typically used as the @mediaAvailable attribute, indicates asset accessibility. The value true indicates the asset is available and the value false indicates the asset is not available. The attribute's omission indicates asset availability is unknown.

11.2.4 messageIdAttrType Attribute Type

messageIdAttrType [nonEmptyStringType]—This attribute type, referenced as the @messageId attribute, is a service channel unique identifier for a message. Every SCTE 130 message instance ***shall*** have a service channel unique value and ***shall not*** be empty. The attribute *should* be a Universally Unique Identifier as defined by RFC 4122 [RFC4122].

11.2.5 messageRefAttrType Attribute Type

messageRefAttrType [messageIdAttrType]—This attribute type, used as the @messageRef attribute, is a reference to an original message via the original message's messageIdAttrType and ***shall not*** be empty.

11.2.6 registrationRefAttrType Attribute Type

registrationRefAttrType [messageIdAttrType]—This attribute type, typically referenced as the @registrationRef attribute, is an exact copy of the original registration message @messageId attribute and provides a linkage to a registration message. The string ***shall not*** be empty. The value ***shall*** never be the @messageId attribute value for a message including the @resend attribute (i.e., @registrationRef ***shall*** be set to the @resend attribute's value as it is the same as the original registration message's @messageId attribute's value).

11.2.7 resendAttrType Attribute Type

resendAttrType [messageIdAttrType]—This attribute type, typically referenced as the @resend attribute, indicates whether a message is a retransmission (resend) of a previous message. The attribute's presence indicates the message is a retransmission and the attribute's value is the corresponding original message's @messageId attribute for which the message is a retransmission. The attribute's omission indicates the message is an original transmission. The retransmitted (resent) message is not required to be an exact copy of the original message. However, the differences *should* be minimized. See Section 9.4.7 for additional information.

11.2.8 systemAttrType Attribute Type

systemAttrType [nonEmptyStringType]—This attribute type, generally referred to as the @system attribute, identifies the originating source and the string ***shall not*** be empty.

11.2.9 versionAttrType Attribute Type

versionAttrType [nonEmptyStringType]—This attribute type, referenced as the @version attribute, is the SCTE 130 message specification version number. The value ***shall*** be specified by the individual SCTE 130 specification parts and ***shall not*** be empty. Refer to the SCTE 130 specification where the message is defined for the currently specified value. Refer to Section 9.1.1 for this part's assigned value (i.e., the SCTE 130 Part 2value which is this document).

11.3 Address Element

The Address element contains endpoint information. Priority order of the Address elements within a container element is indirectly specified by the element document order. The XML value interpretation *may* be specified using the optional @type attribute.

Though the processing rules are not specified here as to how the Address elements are to be used, one *may* consider supplying multiple Address elements as alternate endpoints *should* the first Address element not return in a timely manner. See SCTE 130 Part 7 [SCTE130-7], which supplies additional information and usage criterion.

11.3.1 Address Element Schema

Figure 20 illustrates the Address element's schema.

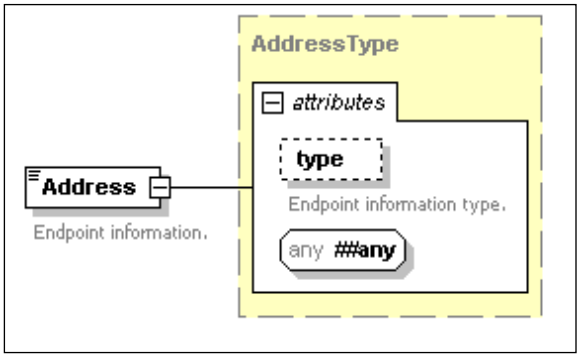

Figure 20: Address Element Schema

11.3.1.1 Semantic Definitions for the Address Element

@type [Optional, nonEmptyStringType]—An attribute uniquely identifying the interpretation of the element's value and the value ***shall not*** be empty. SCTE 130 Part 7 [SCTE130-7] supplies additional information and examples including strings such as: HTTP, SOAP, etc.

@##any [Optional]—Any additional attribute from any namespace.

The Address element's value is of type xsd:string and specifies all or a subset of the connection endpoint identification. The Address element *may* be empty (if all of the data is supplied as attributes). The string format *may* be implied or specified based on the @type attribute value. The string format ***shall*** conform to Section 9.5 as applicable. SCTE 130 Part 7 [SCTE130-7] supplies additional format details.

11.3.2 Address Element Examples

```
<Address type="HTTP">http://25.35.45.55:80/SCTE130</Address>
<Address type="SOAP">adselector1.ads.com/default</Address>
<Address>www.foobar.com/scte130-2/PlacementReq</Address>
```

11.4 AdType Element

The AdType element identifies a specific ad form with respect to its presentation characteristics and effects on the media stream such as graphical overlay. The definition of presentation characteristics and effects on the media stream are outside the scope of this specification.

11.4.1 AdType Element Schema

Figure 21 illustrates the AdType element's schema.

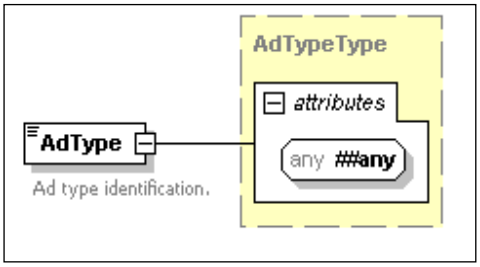

Figure 21: AdType Element Schema

11.4.1.1 Semantic Definitions for the AdType Element

@##any [Optional]—Any additional attribute from any namespace.

The element's value is of type nonEmptyStringType and it describes the ad's type. The value ***shall not*** be empty.

11.3.3 AdType Element Examples

```
<AdType>Graphic Overlay</AdType>
```

11.5 AssetRef Element

The AssetRef element identifies a unique content asset (i.e., entertainment/programming, ad, etc.) through a provider identifier and an asset identifier with the later being unique within the provider reference. Typically, this element is used to identify CableLabs ADI assets or SCTE 236 standard assets. For ADI information, see [CLADI1-1], and [SCTE236].

11.5.1 AssetRef Element Schema

Figure 22 illustrates the AssetRef element's schema.

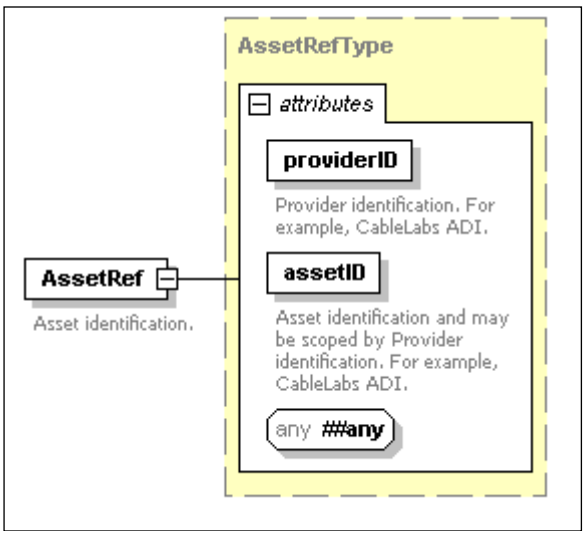

Figure 22: AssetRef Element Schema

11.5.1.1 Semantic Definitions for the AssetRef Element

@providerID [Required, nonEmptyStringType—An attribute matching the provider of the asset as defined by the content data model. The value ***shall not*** be empty. Typically, the value matches either the [CLADI1-1] specification or the [SCTE236] standard identifying the provider of the asset. For example, in ADI this attribute *may* be set to a registered Internet domain name restricted to at most 20 lower-case characters and belonging to the provider.

@assetID [Required, nonEmptyStringType]—An attribute matching the provider of the asset as defined by the content data model and the value ***shall not*** be empty. Typically, the value matches either the [CLADI1-1] specification or the [SCTE236] standard identifying the asset uniquely within the provider's assetID space. For example, in ADI, this attribute *may* be an ASCII string of 20 characters with the first four being Alpha and the remaining 16 being numeric.

@##any [Optional]—Any additional attribute from any namespace.

The AssetRef element's value ***shall*** be empty.

11.5.2 AssetRef Element Examples

```
<AssetRef assetID="ABCD0123456789012345" providerID="example.com"/>
```

11.6 Callout Element

The Callout element specifies a logical service's message reception endpoints. The endpoints *may* be specified as a single collective aggregation, on a per message type basis via the @message attribute, or as a combination of the two techniques.

A single message exchange endpoint *may* be defined to handle all logical service messaging. If a single endpoint is handling all messaging, there ***shall*** be only one Callout element present in the defining specification announcement message and the Callout element's @message attribute ***shall not*** be used. The Callout element omitting the @message attribute is referred to as the default endpoint.

If independent endpoints are desired, the Callout element's @message attribute ***shall*** be used and each Callout element ***shall*** contain a message name as specified by the defining logical service. For an example, see SCTE130 Part 3 [SCTE130-3] the adm:ListADMServicesResponse message.

If independent message exchange endpoints are desired for only a subset of endpoints, the default endpoint *may* be used in conjunction with one or more additional Callout elements. All message endpoints not specifically referenced by an @message attribute ***shall*** be available through the default endpoint. This behavior allows a logical service to provide specific, message exchange endpoints for one or more Callout endpoints while utilizing a single, general purpose endpoint for all other messaging. If this description technique is used, there ***shall*** only be a single Callout element omitting the @message attribute for the callout sequence as detailed by the individual specifications (i.e., there ***shall*** be a maximum of one default endpoint for the applicable callout sequence).

If no default endpoint is supplied and only a subset of the endpoints are provided in the message, the unlisted endpoints ***shall*** be discovered by a different, unspecified mechanism which is outside the scope of this specification.

Within the Callout element, the logical service *may* provide one or more Address elements. The Address element describes a specific endpoint. The processing rules for the Address elements are not specified here and are outside the scope of this specification. Each listed Address element ***shall*** be prepared to receive and respond to a ServiceCheckRequest message and the returned status ***shall*** be applicable to the queried endpoint. When a Callout element contains more than one Address element, at least one Address element specified reception endpoint ***shall*** successfully respond to the ServiceCheckRequest message. The peer *may* reject the Callout element containing message if all Address elements fail to successfully respond. A peer ***shall not*** reject the containing message if only a subset of the Address elements successfully respond to the ServiceCheckRequest message.

11.6.1 Callout Element Schema

Figure 23 illustrates the Callout element's schema.

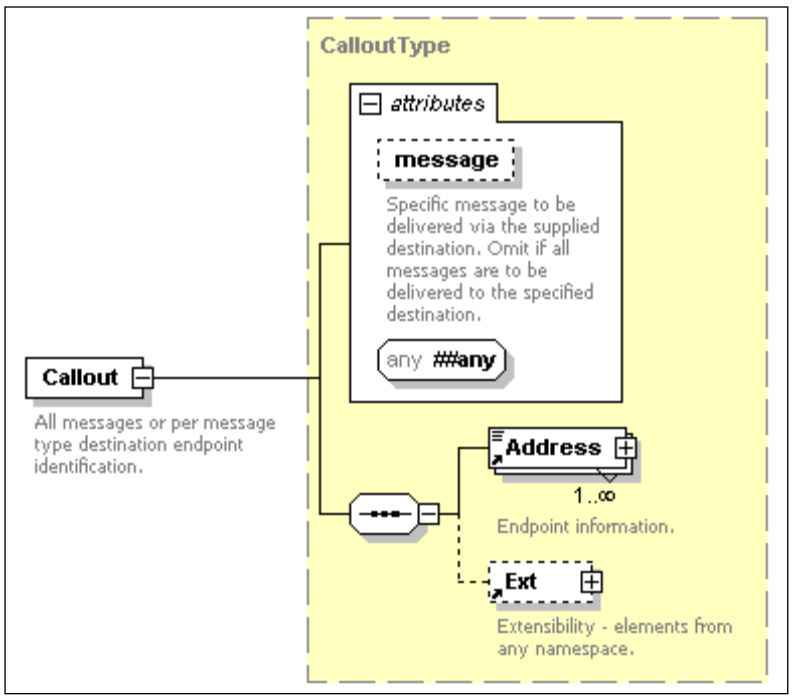

Figure 23: Callout Element Schema

11.6.1.1 Semantic Definitions for the Callout Element

@message [Optional, nonEmptyStringType]—An attribute identifying the message exchange pair being specified. The attribute values are defined within the individual specification parts where the element is utilized. If the attribute is omitted, the Callout element is being declared as the default endpoint. The value ***shall not*** be empty.

@##any [Optional]—Any additional attribute from any namespace.

Address [Required]—One or more elements each specifying an endpoint. See Section 11.3 for additional information.

Ext [Optional]—A container for any additional elements from any namespace. See Section 11.12 for additional information.

The Callout element ***shall not*** be empty.

11.6.2 Callout Element Examples

Example 14 illustrates a default endpoint declaration. All communication is expected to use this interface.

```
<Callout>
  <Address>101.5.3.33:5786</Address>
</Callout>
```

Example 14

Example 15 illustrates a message specific endpoint declaration.

```
 <Callout message="PlacementStatusNotification">
  <Address>adselector1.ads.com</Address>
</Callout>
```

Example 15

Example 16 illustrates a combination of the two techniques. A default endpoint is declared along with two specific message interfaces.

```
<Callout>
  <Address type="HTTP">example.com:80/DefaultMessageHandler</Address>
</Callout>
<Callout message="MessageInterface1">
  <Address type="HTTP">example.com:80/MessageInterface1</Address>
  <Address type="HTTP">example.com:8080/MessageInterface1</Address>
</Callout>
<Callout message="MessageInterface2">
  <Address type="HTTP">example.com:33423/MessageInterface2</Address>
</Callout>
```

Example 16

11.7 Content Element

The Content element describes assets such as entertainment (programming) and ad content. The element is partitioned into three sequences. The first sequence provides content identification. The second sequence *may* be used for content location and the third sequence provides content related information.

11.7.1 Content Element Schema

Figure 24 illustrates the Content element's schema.

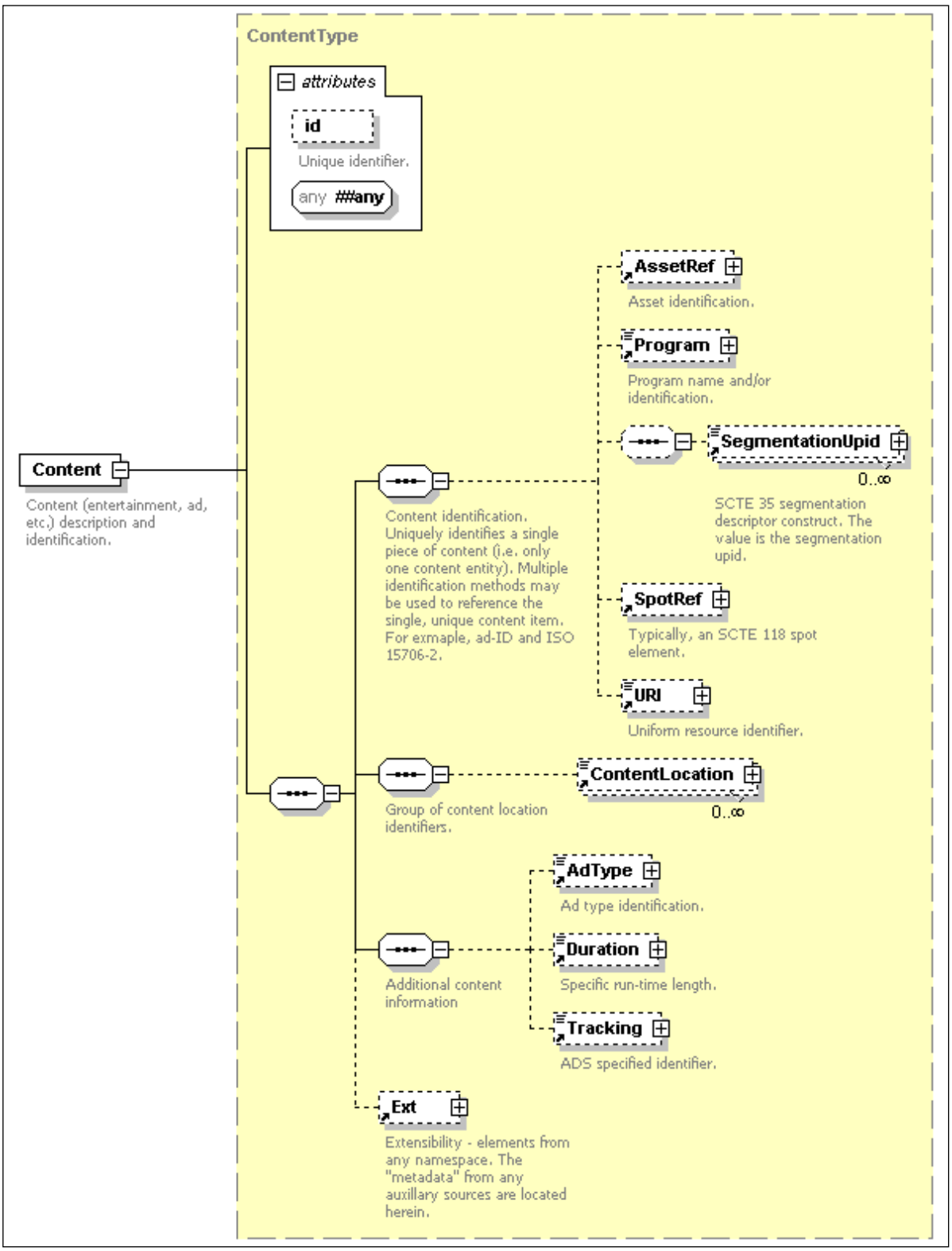

Figure 24: Content Element Schema

11.7.1.1 Semantic Definitions for the Content Element

@id [Optional, idAttrType]—A element unique identifier which *may* be used to identity this Content element.

@##any [Optional]—Any additional attribute from any namespace.

AssetRef [Optional]—Content asset identification typically used with an ADI data model. See Section 11.5 for additional information.

Program [Optional]—Content asset identification typically used when an SCTE 35 splice_insert() section identifies the asset. See Section 11.16 for additional information.

SegmentationUpid [Optional]—Zero or more elements identifying a content asset. Typically, this element is used when an SCTE 35 [SCTE35] segmentation_descriptor() identifies the asset. See Section 11.17 for additional information.

SpotRef [Optional]—Content asset identification typically used for an ad spot asset. Typically, this element is used when an SCTE 118 Spot element [SCTE118-3] identifies the asset. See Section 11.18 for additional information.

URI [Optional]—Content asset identification via a uniform resource identifier (URI). See Section 11.21 for additional information.

ContentLocation [Optional]—Zero or more asset location specifiers. See Section 11.9 for additional information.

AdType [Optional]—An element which *should* only be present when the element references an ad asset. The element describes the ad spot type. See Section 11.4 for additional information.

Duration [Optional]—The content run-time length. See Section 11.11 for additional information.

Tracking [Optional]—An externally assigned identifier. This element is typically supplied by an ADS when the Content element is provided as part of a placement description. See Section 11.20 for additional information.

Ext [Optional]—A container for any additional elements from any namespace. See Section 11.12 for additional information.

The Content element's value *should not* be empty.

11.7.2 Content Element Examples

```
<Content>
  <Program>Three Guy Jokes Gone Bad</Program>
</Content>

<Content>
  <AssetRef providerID="theadshop.com" assetID="ADCO123456789012345"/>
  <SpotRef trafficId="893" spotId="TheSpot"/>
  <AdType>Telescoping</AdType>
</Content>

<Content>
  <AssetRef providerID="example.com" assetID="ADCO0987654321098765"/>
  <AdType>Overlay</AdType>
  <Duration>PT30.000S</Duration>
  <Tracking>MyTrackingID 839839839</Tracking>
</Content>

<Content>
  <!--Two Segmentation UPIDs coordinating references to the same piece of content.-->
  <SegmentationUpid type="6"><!—ISO 15706-2 goes here.--></SegmentationUpid>
  <SegmentationUpid type="3"><!--Advertising Digital Identification goes here.--
></SegmentationUpid>
</Content>
```

11.8 ContentDataModel Element

The ContentDataModel element facilitates expressing the general content data model and a specific revision as desired.

11.8.1 ContentDataModel Element Schema

Figure 25 illustrates the ContentDataModel element's schema.

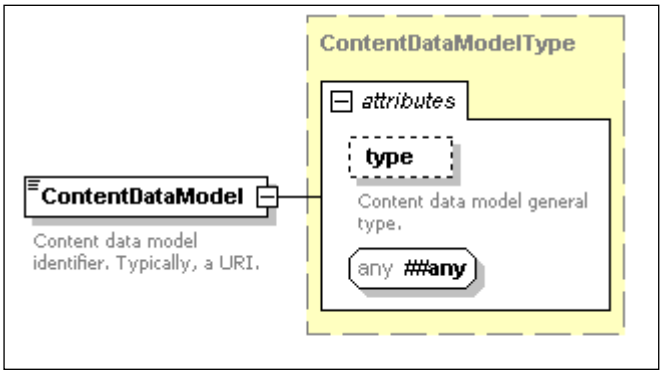

Figure 25: ContentDataModel Element Schema

11.8.1.1 Semantic Definitions for the ContentDataModel Element

@type [Optional, nonEmptyStringType]—A string that ***shall not*** be empty generalizing the data model. Table 3 lists the defined values, which *may* be extended by private agreement outside the scope of this specification. The values ***shall*** appear exactly as they appear in Table 3.

Value	Description
CLADI_1.1	Model represented by the CableLabs ADI 1.1 specification.
CLMD_3.0	Model represented by the CableLabs Metadata 3.0 specification. Use the value SCTE236 as this value remains only for backward compatibility.
SCTE118-2	Model represented by the SCTE 118-2 standard.
SCTE118-3	Model represented by the SCTE 118-3 standard.
SCTE236	Model represented by the SCTE 236 standard.
…	User defined and outside the scope of this specification. The string ***shall*** be prefixed with the text “private:”.

Table 3: ContentDataModel Element’s @type Attribute Defined Values

@##any [Optional]—Any additional attribute from any namespace.

The XML value is of type nonEmptyStringType and ***shall not*** be empty. The string represents precise identification of the content data model which might include the revision. The string *should* be a URI. See [RFC3986] for additional information.

11.8.2 ContentDataModel Element Examples

```
 <ContentDataModel
type="SCTE236">http://www.scte.org/schemas/236/2017</ContentDataModel>

 <ContentDataModel
type="CLADI_1.1">http://www.cablelabs.com/InsertTheRevisionDate/ADI1.1</ContentD
ataModel>

 <ContentDataModel
type="CLMD_3.0">urn:cablelabs:md:xsd:content:3.0</ContentDataModel>

 <ContentDataModel type="SCTE118-3">http://www.scte.org/schemas/118-
3/2013</ContentDataModel>
```

11.9 ContentLocation Element

The ContentLocation element facilitates specifying an asset's place. The element value *may* be any valid URI. See [RFC3986] for additional information.

11.9.1 ContentLocation Element Schema

Figure 26 illustrates the ContentLocation element's schema.

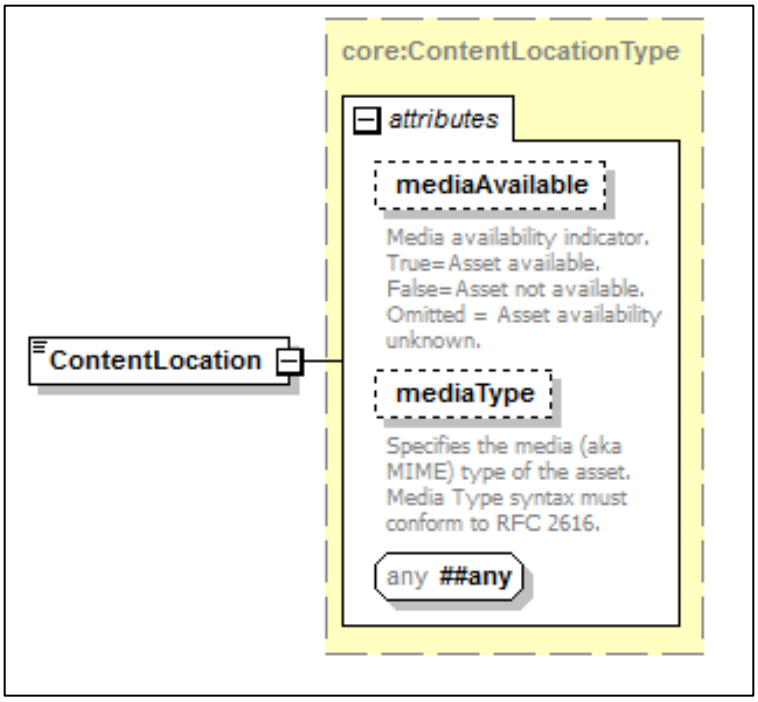

Figure 26: ContentLocation Element Schema

11.9.1.1 Semantic Definitions for the ContentLocation Element

@mediaAvailable [Optional, mediaAvailableAttrType]—Media accessibility indicator. True indicates the asset is available. False indicates the asset is not available. Attribute omission indicates the asset's availability is unknown. See Section 11.2.3 for additional information.

@mediaType[Optional, xsd:normalizedString] – Media Type. This optional attribute *may* be used to indicate the type of media available at the content location. When present, it specifies the Internet Media Type (aka MIME type) of the asset and ***shall*** conform to the Media Type syntax as described in RFC 2616 Section 3.7 [RFC2616]. Media Type values *should* conform to RFC 2046 [RFC2046] and *should* be registered with IANA as per RFC 6838 [RFC6838]. Use of unregistered media types is discouraged.

@##any [Optional]—Any additional attribute from any namespace.

The XML value is of type xsd:anyURI and *may* be empty when only the @mediaAvailable attribute is carried. The string represents the content asset's location. See [RFC3986] for additional information.

11.9.2 ContentLocation Element Examples

```
<ContentLocation mediaAvailable="true" mediaType="video/mpeg">
  ftp://somehost/ReadyToPlay/asset.mpeg
</ContentLocation>

<ContentLocation mediaAvailable="false"/>

<ContentLocation>ftp://somehost/asset.mpeg</ContentLocation>
```

The first example element provides an indication that the media is currently accessible, a mediaType and a URL at which to access the asset.

The second example element, which has the @mediaAvailable attribute set to the value false, indicates the content is known not to be accessible. In this case, no URL has been provided (though one could have been supplied.)

The final example element provides a URL without specifying the media availability (i.e., the media accessibility is unknown due to the omission of the @mediaAvailable attribute type). Because of the media unknown state indication, media access *may* result in an error.

11.10 CurrentDateTime Element

The CurrentDateTime element provides the current date and time value which ***shall*** include a timezone indicator. See Section 11.1.1 for additional information.

11.10.1 CurrentDateTime Element Schema

Figure 27 illustrates the CurrentDateTime element's schema.

Figure 27: CurrentDateTime Element Schema

11.10.1.1 Semantic Definitions for the CurrentDateTime Element

The CurrentDateTime element's value is of type core:dateTimeTimezoneType and ***shall not*** be empty. See Section 11.1.1 for additional information.

11.10.2 CurrentDateTime Element Examples

```
<CurrentDateTime>2007-01-05T12:30:36.0Z</CurrentDateTime>
```

11.11 Duration Element

The Duration element describes a run-time in years, months, days, hours, minutes, seconds, and milliseconds.

11.11.1 Duration Element Schema

Figure 28 illustrates the Duration element's schema.

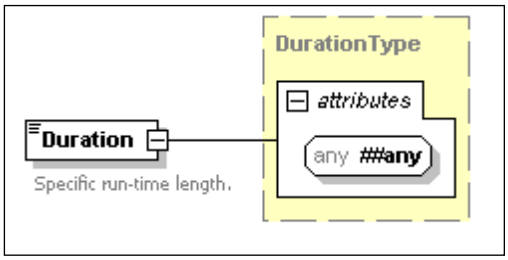

Figure 28: Duration Element Schema

11.11.1.1 Semantic Definitions for the Duration Element

@##any [Optional]—Any additional attribute from any namespace.

The Duration element's value is of type xsd:duration and is formatted as PnYnMnDTnHnMn.nnnS.

11.11.2 Duration Element Examples

```
<Duration>PT30S</Duration><!--30 seconds-->
<Duration>PT29.666S</Duration><!--29 seconds and 666 milliseconds-->
<Duration>PT01H00M05.000S</Duration><!--1 hour and 5 seconds-->
<Duration>PT03H04M05.333S</Duration><!--3 hours, 4 minutes, 5 seconds, and 333 milliseconds-->
```

11.12 Ext Element

The Ext (extensibility) element allows zero or more elements from any namespace to be included. This element facilitates expansion, customization, and extensibility of the specification. Encapsulating elements from external namespaces into a single element allows filters, transforms, and other operations to be applied easily.

11.12.1 Ext Element Schema

Figure 29 illustrates the Ext element's schema.

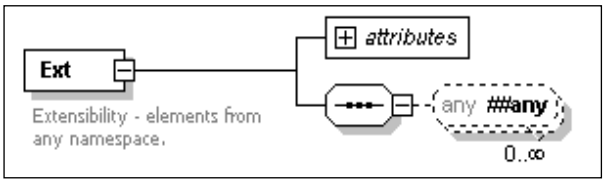

Figure 29: Ext Element Schema

11.12.1.1 Semantic Definitions for the Ext Element

@##any [Optional]—Any additional attribute from any namespace.

##any[Optional]—Zero or more elements from any namespace. (Zero elements are allowed as all the data *may* be included via attributes.)

11.12.2Ext Element Examples

```
<Ext>
  <!--Elements from any namespaces go here.-->
</Ext>
```

11.13 ExternalStatusCode Element

The ExternalStatusCode element allows for a detailed status code to be supplied from an external (non SCTE 130) specification. The element identifies the status code source and *may* be augmented with optional descriptive text.

11.13.1ExternalStatusCode Schema

Figure 30 illustrates the ExternalStatusCode element's schema.

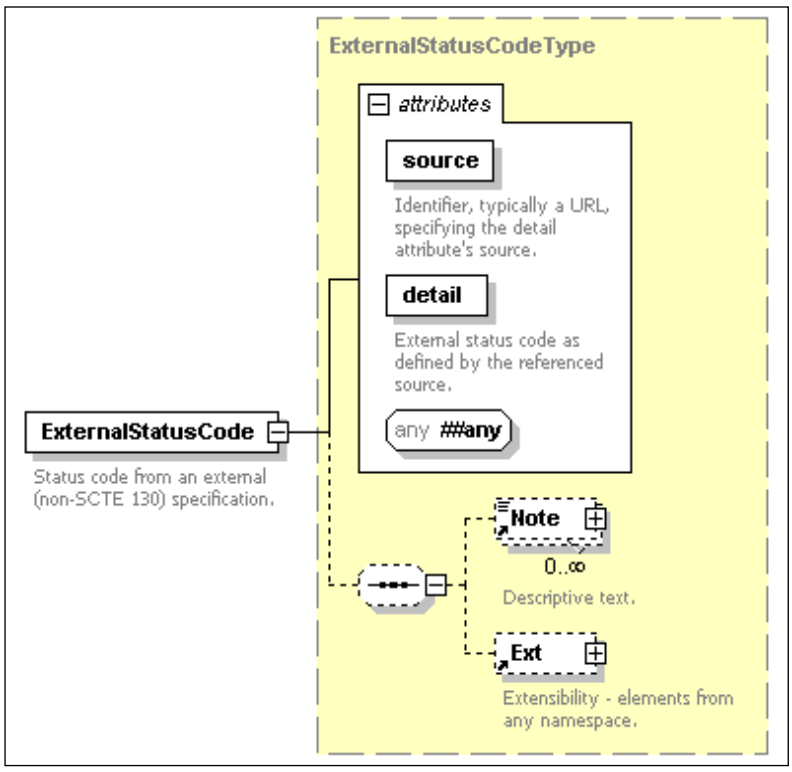

Figure 30: ExternalStatusCode Element Schema

11.13.1.1 Semantic Definitions for the ExternalStatusCode Element

@source [Required, nonEmptyStringType]—Identification of the external status code source. Typically, the value *should* be a uniform resource identifier (URI), see [RFC3986], and ***shall not*** be empty. Table 4 lists the defined values which *may* be extended by private agreement outside the scope of this specification.

@source Value	Description
http://www.scte.org/schemas/35/2013	SCTE 35 [SCTE35]
http://www.scte.org/schemas/118-3/2012	SCTE 118-3 [SCTE118-3]
…	User defined and outside the scope of this specification.

Table 4: ExternalStatusCode Element's @source Attribute Values

@detail [Required, nonEmptyStringType]—A non-empty external status code value.

@##any [Optional]—Any additional attribute from any namespace.

Note [Optional]—Zero or more Note elements where each Note element contains descriptive text. See Section 11.15 for additional information regarding the Note element.

Ext [Optional]—A container for any additional elements from any namespace. See Section 11.12 for additional information.

11.13.2 ExternalStatusCode Element Examples

```
<ExternalStatusCode source="http://www.scte.org/schemas/118-3/2006" detail="3"/> <!--Failed, bypass on-->

<ExternalStatusCode source="http://www.scte.org/schemas/118-3/2006" detail="16">
  <Note>Failed, Operator Error.</Note>
  <Note>RTFM...</Note>
</ExternalStatusCode>
```

11.14 InitiatorData Element

The InitiatorData element provides carriage for privately defined attributes and a string that ***shall*** be returned exactly as received (i.e., echoed back). The logical service endpoint respondent ***shall*** return an exact copy of this element when received. The element's contents are opaque to the respondent logical services since the data is implementation specific relative to the originating logical service. Example element usage includes providing high availability services or other value-added features.

11.14.1 InitiatorData Element Schema

Figure 31 illustrates the InitiatorData element's schema.

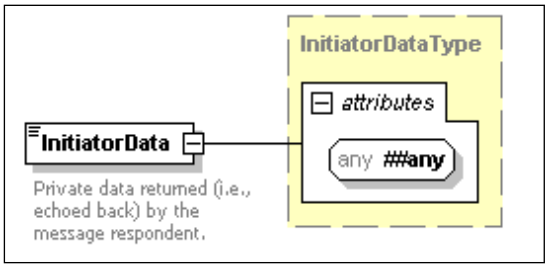

Figure 31: InitiatorData Element Schema

11.14.1.1 Semantic Definitions for the InitiatorData Element

@##any [Optional]—Any additional attribute from any namespace.

The InitiatorData element's value is of type xsd:string and *may* be empty (if all the data is provided as attributes).

11.14.2 InitiatorData Element Examples

```
<InitiatorData secret="Can't tell">
  Company specific secret sauce goes here.
</InitiatorData>
```

11.15 Note Element

The Note element carries descriptive text.

11.15.1 Note Element Schema

Figure 32 illustrates the Note element's schema.

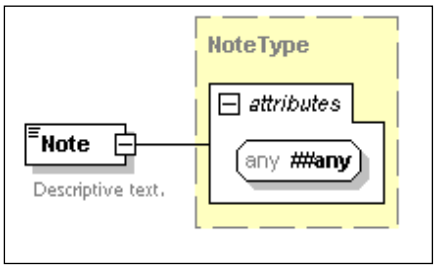

Figure 32: Note Element Schema

11.15.1.1 Semantic Definitions for the Note Element

@##any [Optional]—Any additional attribute from any namespace.

The Note element's value is of type nonEmptyStringType and ***shall not*** be empty.

11.15.2 Note Element Examples

```
<Note>Three guys go into a bar...</Note>
```

11.16 Program Element

The Program element provides the program name and/or a unique program identifier.

11.16.1 Program Element Schema

Figure 33 illustrates the Program element's schema.

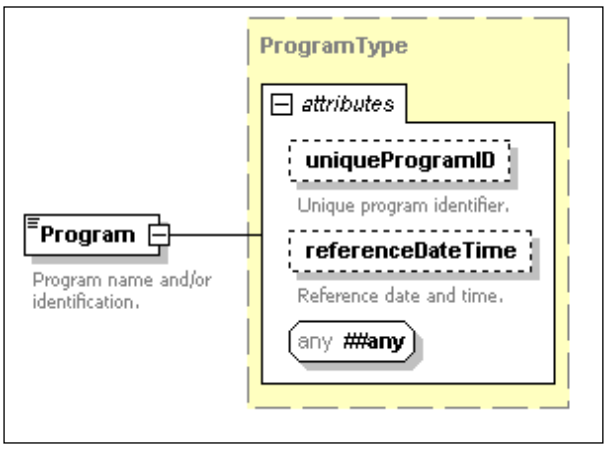

Figure 33: Program Element Schema

11.16.1.1 Semantic Definitions for the Program Element

@uniqueProgramID [Optional, xsd:nonNegativeInteger]—An attribute uniquely identifying the program.

@referenceDateTime [Optional, core:dateTimeTimezoneType]—An attribute identifying when the @uniqueProgramID attribute was established (i.e., contextual reference). This attribute *should* only be used when the @uniqueProgramID attribute is present. See Section 11.1.1 for additional information.

@##any [Optional]—Any additional attribute from any namespace.

The Program element's value is of type xsd:string and *may* be empty.

11.16.2Program Element Examples

```
<Program>TheBestProgram</Program>
<Program uniqueProgramID="39030"/>
<Program uniqueProgramID="9389">UglyBetty</Program>
<Program uniqueProgramID="8389" referenceDateTime="2007-05-
05T09:33:55Z">UglyBetty</Program>
```

11.17 SegmentationUpid Element

The SegmentationUpid element corresponds to the SCTE 35 segmentation_descriptor(). See [SCTE35] for additional information. The element facilitates in-band content asset identification (either entertainment/programming or ad).

11.17.1SegmentationUpid Element Schema

Figure 34 illustrates the SegmentationUpid element's schema.

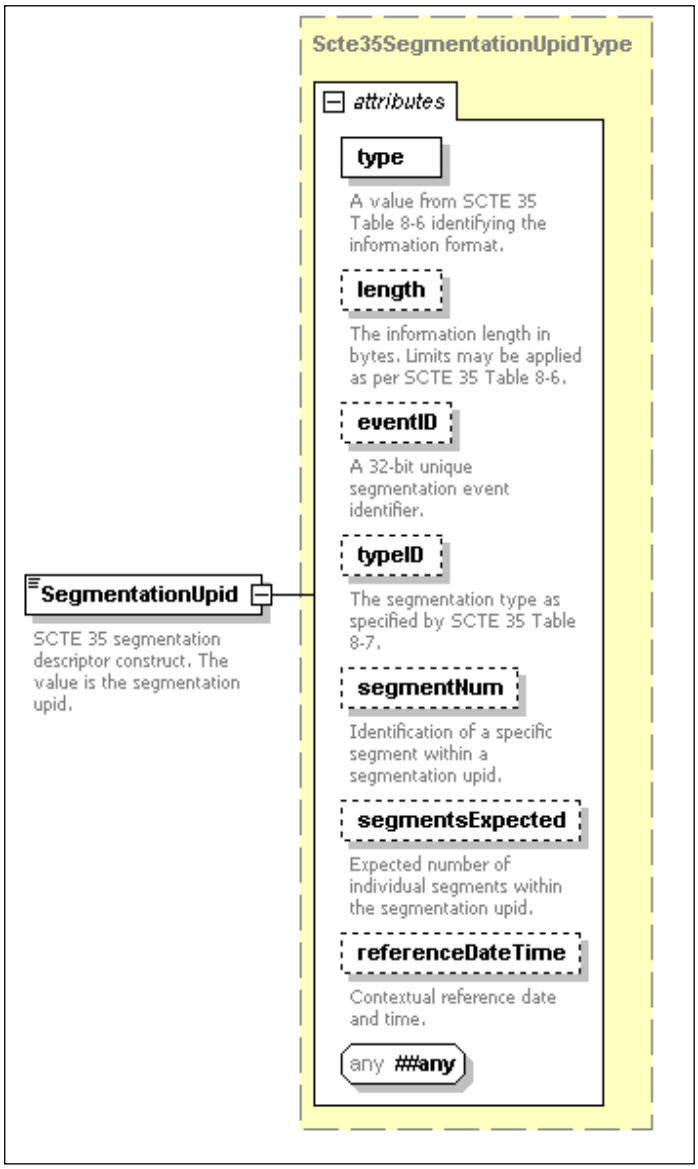

Figure 34: SegmentationUpid Element Schema

11.17.1.1 Semantic Definitions for the SegmentationUpid Element

@type [Required, xsd:unsignedByte]—Any valid value from SCTE 35 Table 8-6 Type column where the attribute maps to the SCTE 35 segmentation_upid_type bit field. See [SCTE35] for additional information.

@length [Optional, xsd:unsignedByte]—Any valid value from SCTE 35 Table 8-6 Length Bytes column and the @length attribute's value is the binary data length. The @length value is dependent upon the @type

value and maps to the SCTE 35 segmentation_upid_length bit field. See [SCTE35] for additional information.

@eventID [Optional, xsd:unsignedInteger]—The SCTE 35 segmentation_event_id bit field. See [SCTE35] for additional information.

@typeID [Optional, xsd:unsignedByte]—Any valid value from SCTE 35 Table 8-7 which maps to the segmentation_type_id bit field. See [SCTE35] for additional information.

@segmentNum [Optional, xsd:unsignedByte]—An attribute conformant to the SCTE 35 segment_num bit field description. See [SCTE35] for additional information.

@segmentsExpected [Optional, xsd:unsignedByte]—An attribute conformant to the SCTE 35 segments_expected bit field description. See [SCTE35] for additional information.

@referenceDateTime [Optional, core:dateTimeTimezoneType]—The date and time providing contextual reference. See Section 11.1.1 for additional information.

@##any [Optional]—Any additional attribute from any namespace.

The SegmentationUpid element's value is of type xsd:hexBinary and contains the SCTE 35 segmentation_upid bit field. The value *should not* be empty. The value is specific to the @type attribute and ***shall*** meet the requirements as specified in SCTE 35. See [SCTE35] for additional information.

11.17.2 SegmentationUpid Element Examples

```
<SegmentationUpid type="1">89</SegmentationUpid>
<SegmentationUpid type="6" length="12" eventID="8" typeID="32" segmentNum="2" segmentsExpected="4">188166C7342065419F3A0245</SegmentationUpid><!--96-bit ISO 15706-2-->
```

11.18 SpotRef

The SpotRef element corresponds to a subset of the SCTE 118-3 schedule's spot element. The element facilitates content asset identification (typically an ad). Refer to [SCTE118-3] for additional information.

11.18.1 SpotRef Element Schema

Figure 35 illustrates the SpotRef element's schema.

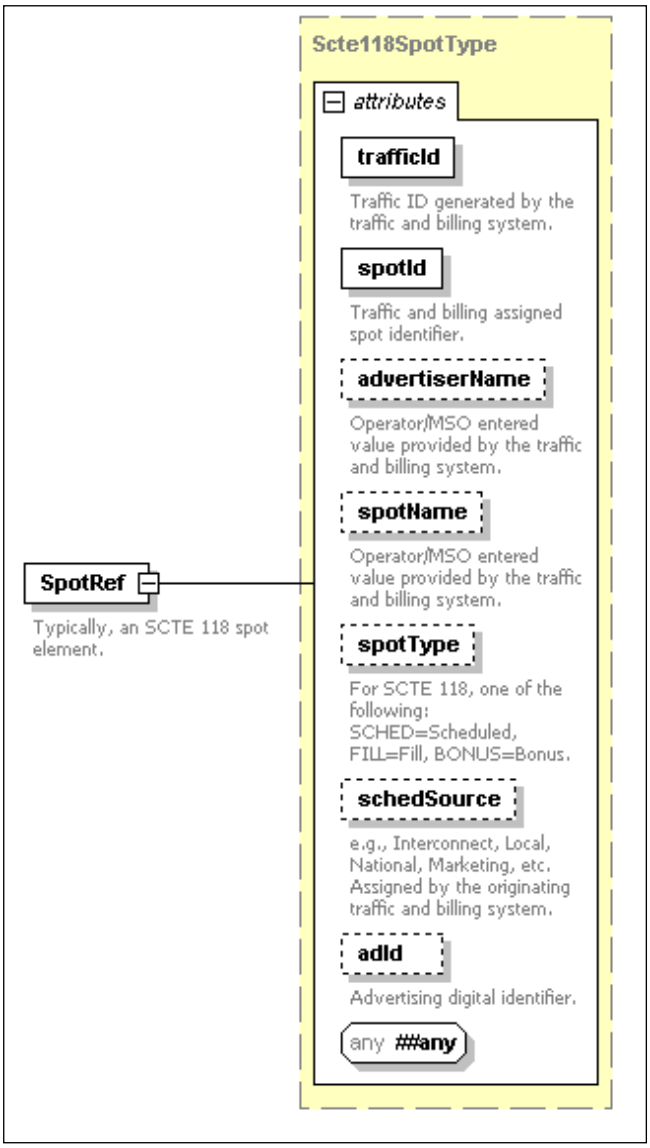

Figure 35: SpotRef Element Schema

11.18.1.1 Semantic Definitions for the SpotRef Element

@trafficId [Required, xsd:integer]—A unique ID generated by the traffic and billing system for tracking a specific ad spot instance. See [SCTE118-3] for additional information.

@spotId [Required, nonEmptyStringType]—A non-empty string generated by the traffic and billing system for spot identification. See [SCTE118-3] for additional information.

@advertiserName [Optional, nonEmptyStringType]—A non-empty string representing the operator entered advertiser name value provided by the traffic and billing system. See [SCTE118-3] for additional information.

@spotName [Optional, nonEmptyStringType]—A non-empty string representing the operator entered spot identification name provided by the traffic and billing system. See [SCTE118-3] for additional information.

@spotType [Optional, nonEmptyStringType]—A non-empty string representing the spot type. From SCTE 118-3, the values might be SCHED, FILL, or BONUS. See [SCTE118-3] for additional information.

@schedSource [Optional, nonEmptyStringType]—A non-empty string identifying the scheduling source as assigned by the originating traffic and billing system. See [SCTE118-3] for additional information.

@adId [Optional, nonEmptyStringType]—A non-empty string supplying the advertising digital identifier. See [SCTE118-3] for additional information.

@##any [Optional]—Any additional attribute from any namespace.

The SpotRef element's value ***shall*** be empty.

11.18.2 SpotRef Element Examples

```
<SpotRef trafficId="339" spotId="TheSpot"/>
<SpotRef trafficId="8989" spotId="SpotIdentifier1" advertiserName="JoesShoes"
spotName="SmellyFeet" spotType="SCHED" schedSource="Local"
adId="AdIDGoesHere"/>
```

11.19 StatusCode Element

The StatusCode element provides a general status classification value using the @class attribute and a specific detail value when applicable.

11.19.1 StatusCode Element Schema

Figure 36 illustrates the StatusCode element's schema.

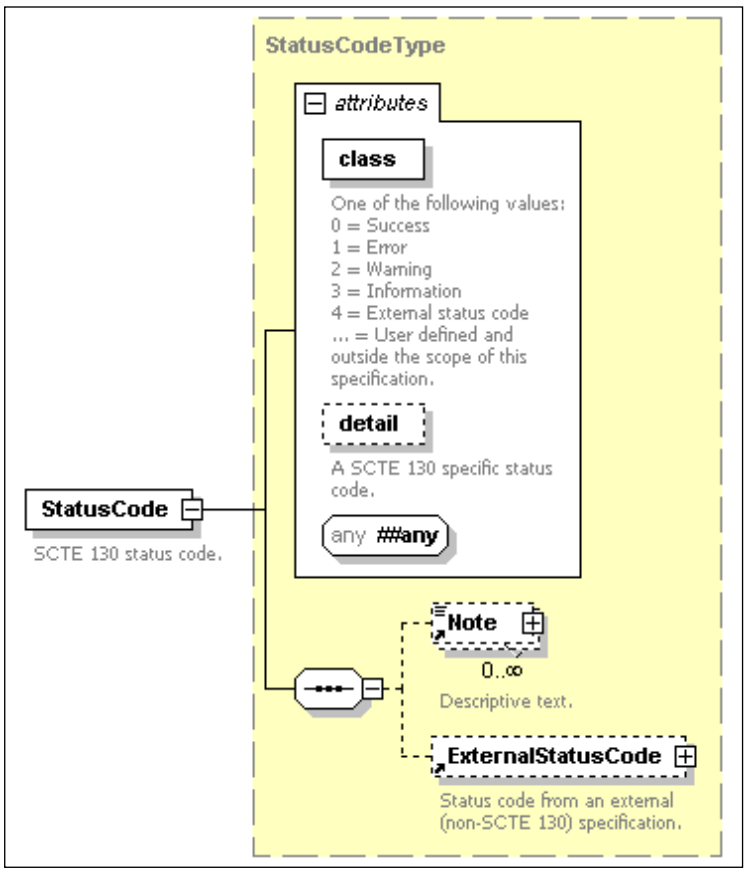

Figure 36: StatusCode Element Schema

11.19.1.1 Semantic Definitions for the StatusCode Element

@class [Required, xsd:nonNegativeInteger]—The value equates to one of the Table 5 specified @class attribute values which *may* be extended by private agreement outside the scope of this specification. If the @class attribute does not contain a success value, either the @detail attribute and/or the ExternalStatusCode element *should* be used to communicate the reason. Additionally, one or more Note elements *should* be used to communicate detailed text.

@class Attribute Value	Description
0	Success
1	Error
2	Warning
3	Information
4	Deferred to the ExternalStatusCode element and there ***shall*** be an ExternalStatusCode element present in this StatusCode element

5	Partial success (Originally defined in Part 8 — Appendix F Mutable API.)
6	Batch cancelled (Originally defined in Part 8 — Appendix F Mutable API.)
…	User defined and outside the scope of this specification.

Table 5: StatusCode Element @class Attribute Values

@detail [Optional, xsd:nonNegativeInteger]—The applicable detail status code value from Appendix A Table 6.

@##any [Optional]—Any additional attribute from any namespace.

Note [Optional]—Zero or more Note elements where each Note element contains descriptive text. See Section 11.15 for additional information regarding the Note element.

ExternalStatusCode [Optional]—A container for a status code from an external source. See Section 11.13 for additional information.

11.19.2StatusCode Element Examples

```
<StatusCode class="0"/> <!--Success-->

<StatusCode class="1" detail="2"/>

<StatusCode class="1" detail="5">
  <Note>Ambiguous details.</Note>
  <Note>Supplied identifier did not match.</Note>
</StatusCode>

<StatusCode class="3"> <!--Information-->
  <Note>Contact with the CIS has been established.</Note>
</StatusCode>

<StatusCode class="4">
  <ExternalStatusCode source="http://www.scte.org/schemas/118-3/2006" detail="20">
    <Note>Failed. No ad copy in inserter.</Note>
    <Note>Could not find the file fubar.mpg.</Note>
  </ExternalStatusCode>
</StatusCode>
```

11.20 Tracking Element

The Tracking element provides carriage for privately defined attributes and data which ***shall*** be returned in normatively specified container elements. The returned Tracking element ***shall*** be an exact copy of the received original (i.e., the element is echoed back). The element's usage and return requirements are defined explicitly by the including specification. (For example, see [SCTE130-3].) The internal element information is opaque to all other logical services as the data is implementation specific to the originating logical service.

An example Tracking element usage is as a unique identifier for a content asset. Typically, the value is assigned by an ADS to track a specific ad asset instance. For example, the Tracking element *may* be present in an adm:Placement element's core:Content element. SCTE 130 Part 3 mandates that the Tracking element always be provided when that specific core:Content element instance is thereafter referenced. Consequently in this example, if the Tracking element is provided in a core:Content element contained within an adm:Placement element and the same core:Content element is later used in an ADM named event element, the Tracking element is then provided either directly or indirectly in the named event element as noted in SCTE 130 Part 3 (see [SCTE130-3]. The Tracking element is not limited to this usage but rather this example describes one possible usage.

11.20.1 Tracking Element Schema

Figure 37 illustrates the Tracking element's schema.

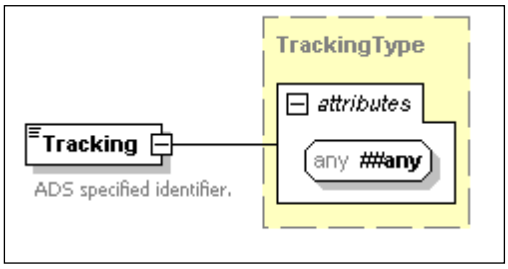

Figure 37: Tracking Element Schema

11.20.1.1 Semantic Definitions for the Tracking Element

@##any [Optional]—Any additional attribute from any namespace.

The Tracking element's value is of type xsd:string and *should not* be empty (but *may* be if all the data is provided as attributes).

11.20.2 Tracking Element Examples

```
<Tracking>TrackingTagExampleValue 123893 9893 8939398 993</Tracking>
<Tracking>Put any string desired here.</Tracking>
```

11.21 URI Element

The URI (Uniform Resource Identifier) element is used for identification, typically content identification (i.e., a name or identifier associated with an asset). See [RFC3986] for additional information.

11.21.1URI Element Schema

Figure 38 illustrates the URI element's schema.

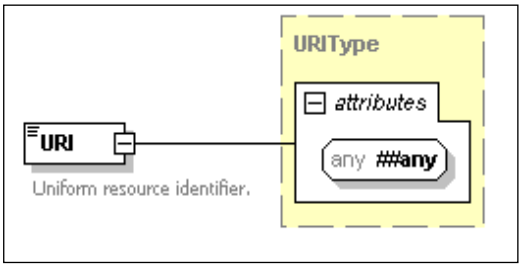

Figure 38: URI Element Schema

11.21.1.1 Semantic Definition for the URI Element

@##any [Optional]—Any additional attribute from any namespace.

The URI element's value is of type xsd:anyURI and ***shall not*** be empty. See [RFC3986] for additional information.

11.21.2URI Element Examples

```
<URI>Any valid URI/URL/URN</URI>
<URI>ftp://somehost/asset.mpeg</URI>
```

APPENDIX A: (NORMATIVE) STATUSCODE ELEMENT @DETAIL ATTRIBUTE VALUES

Table 6 contains applicable values for the StatusCode element's @detail attribute. A checkmark (✓) in the ServiceCheckResponse (SCR), the ServiceStatusNotification (SSN), or the ServiceStatusAcknowledgment (SSA) column indicates when the code *may* be used.

The @detail value ***shall*** be formed by preceding the last 3 numeric digits by the specification part with the exception of this document. Thus, the concatenated format is <part number><###> where each # represents a digit. The exception is for this document which uses the values 0 to 2999.

Note: Private implementation specific error codes ***shall*** be carried by an ExternalStatusCode element. See Section 11.13 for additional information.

@detail Value	Description	SCR	SSN	SSA	Comment
0	Reserved				
1	Incomplete message	✓	✓	✓	
2	Message validation failed	✓	✓	✓	Includes a malformed message.
3	Registration overlap				
4	Query failed				
5	Ambiguous details	✓		✓	
6	Unsupported protocol				
7	Network address does not exist		✓		
8	Network address/port in use	✓	✓		
9	Duplicate message id	✓	✓	✓	
10	Network connection lost	✓	✓		
11	Resource not found	✓	✓		
12	Not supported				
13	Not authorized				
14	Unknown message reference	✓		✓	
15	Resend forced abandonment	✓		✓	
16	Out of resources	✓	✓	✓	
17	Timeout	✓	✓	✓	
18	General error	✓	✓	✓	
19 to 2999	Reserved for Part 2				

3000 to 3999	Reserved for Part 3				Part 3 specific errors.
4000 to 4999	Reserved for Part 4				Part 4 specific errors.
5000 to 5999	Reserved for Part 5				Part 5 specific errors.
6000 to 6999	Reserved for Part 6				Part 6 specific errors.
7000 to 7999	Reserved for Part 7				Part 7 specific errors.
8000 to 8999	Reserved for Part 8				Part 8 specific errors.
10000 to 10999	Reserved for Part 10				Part 10 specific errors.

Table 6: StatusCode Element @detail Attribute Values