International Telecommunication Union

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

J.380.2

(11/2011)

SERIES J: CABLE NETWORKS AND TRANSMISSION OF TELEVISION, SOUND PROGRAMME AND OTHER MULTIMEDIA SIGNALS

Digital transmission of television signals

Digital program insertion – Advertising systems interfaces – Core data elements

Recommendation ITU-T J.380.2

ITU-T

International Telecommunication Union

Recommendation ITU-T J.380.2

Digital program insertion – Advertising systems interfaces – Core data elements

Summary

Recommendation ITU-T J.380.2 describes the digital program insertion advertising systems interfaces' core messaging and data types, using extensible mark-up language (XML), XML namespaces, and XML schema.

History

Edition	Recommendation	Approval	Study Group
1.0	ITU-T J.380.2	2011-11-13	9

FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications, information and communication technologies (ICTs). The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

Compliance with this Recommendation is voluntary. However, the Recommendation may contain certain mandatory provisions (to ensure, e.g., interoperability or applicability) and compliance with the Recommendation is achieved when all of these mandatory provisions are met. The words "shall" or some other obligatory language such as "must" and the negative equivalents are used to express requirements. The use of such words does not suggest that compliance with the Recommendation is required of any party.

INTELLECTUAL PROPERTY RIGHTS

ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, ITU had not received notice of intellectual property, protected by patents, which may be required to implement this Recommendation. However, implementers are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database at http://www.itu.int/ITU-T/ipr/.

Table of Contents

List of Figures

Page

List of Tables

Introduction

This Recommendation defines the following:

– The extensible ITU-T J.380 message structure.

– A set of required messages that shall be implemented by all ITU-T J.380 compliant logical services.

– A common set of XML elements (i.e., core elements) which may appear in any ITU-T J.380 message or within any ITU-T J.380 defined element.

– Common XML attributes used by the core data elements which may also be used by any other element.

All externally defined ITU-T J.380 messages shall contain the appropriate common message attributes as defined herein.

This Recommendation also defines requirements for the ITU-T J.380 message transport without specifying the actual transport implementation. [ITU-T J.380.7] is expected to define the normative transport specification.

Recommendation organization

This Recommendation provides a structured, logical approach to the core aspects of the Recommendation ITU-T J.380.x series. Each subsequent clause focuses on a particular specification aspect and the document's presentation order provides a top-down, conceptual introduction necessary to implement Recommendation ITU-T J.380.

Clause 5 explains the document's notational conventions. Clause 6 defines the XML namespace usage and the applicable XML semantics. Clause 7 introduces the message concepts and the core XML attributes which are the foundations of every ITU-T J.380 message. This clause also defines the message exchange characteristics common to all ITU-T J.380 message interactions.

Clause 7 and its subsequent clauses often may reference XML attributes and elements defined later in the document. The reader may occasionally need to reference other clauses of this document in order to find the complete explanation of a syntactic component.

Clause 8 defines messages that shall be supported by every ITU-T J.380 logical service. Clause 9 presents the core XML attributes followed by the principal specification XML elements with each group presented in alphabetical order. Annex A concludes the document with the normative status code values, their meanings, and their message usage applicability.

Recommendation ITU-T J.380.2

Digital program insertion – Advertising systems interfaces – Core data elements

1 Scope

Recommendation ITU-T J.380.2 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 ITU-T J.380 message attributes, and the required ITU-T J.380 messages. The core data types are XML attributes and XML elements which may be used in any ITU-T J.380.x message element or within an ITU-T J.380.x element definition.

2 References

The following ITU-T Recommendations and other references contain provisions, which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

[ITU-T J.380.3] Recommendation ITU-T J.380.3 (2011), *Digital program insertion – Advertising systems interfaces – Management interface*.

[ITU-T J.380.4] Recommendation ITU-T J.380.4 (2011), *Digital program insertion – Advertising systems interfaces – Content information service.*

[ITU-T J.380.5] Recommendation ITU-T J.380.5 (2011), *Digital program insertion – Advertising systems interfaces – Placement opportunity information service.*

[ITU-T J.380.6] Recommendation ITU-T J.380.6 (2011), *Digital program insertion – Advertising systems interfaces – Subscriber information service.*

[ITU-T J.380.7] Recommendation ITU-T J.380.7 (2011), *Digital program insertion – Advertising systems interfaces – Message transport*.

[ITU-T J.380.x] Recommendation ITU-T J.380.x-series (2011), *Digital program insertion – Advertising systems interfaces.*

[IEEE 802] IEEE Std 802-2001 – *IEEE Standard for Local and Metropolitan Area Networks: Overview and Architecture-IEEE Computer Society*.
<http://standards.ieee.org/getieee802/802.html>

[IETF RFC 3986] IETF RFC 3986 (2005), *Uniform Resource Identifier (URI): Generic Syntax*.
<www.ietf.org/rfc/rfc3986.txt>

[SCTE 35] ANSI/SCTE 35-2007, *Digital Program Insertion Cueing Message for Cable.*

[SCTE 118-3] ANSI/SCTE 118-3-2006, *Program Specific Ad Insertion – Traffic System to Ad Insertion System File Format Specification*.

[SCTE 130-2/core schema] ANSI/SCTE 130-2-2008, *Digital Program Insertion – Advertising Systems Interfaces Part 2 Core Data Elements* – core schema.
<http://www.scte.org/schemas/130-2/SCTE_130-2_2008a.xsd>

[XML] W3C Recommendation (2006), *Extensible Markup Language (XML) 1.0 (Fourth Edition)*.
<http://www.w3.org/TR/2006/REC-xml-20060816/>

[XMLInfoSet] W3C Recommendation (2004), *XML Information Set (Second Edition)*.
<http://www.w3.org/TR/2004/REC-xml-infoset-20040204/>

[XMLNamespaces] W3C Recommendation (2006), *Namespaces In XML 1.0 (Second Edition)*.
<http://www.w3.org/TR/2006/REC-xml-names-20060816/>

[XMLSchemaP1] W3C Recommendation (2004), *XML Schema Part 1: Structures (Second Edition)*.
<http://www.w3.org/TR/xmlschema-1/>

[XMLSchemaP2] W3C Recommendation (2004), *XML Schema Part 2: Datatypes (Second Edition)*.
<http://www.w3.org/TR/xmlschema-2/>

3 Definitions

3.1 Terms defined elsewhere

None.

3.2 Terms defined in this Recommendation

This Recommendation defines the following terms:

3.2.1 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.

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

3.2.3 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.

3.2.4 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.

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

3.2.6 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. The enforcement of uniqueness as well as the creation of globally unique identifiers is outside the scope of this Recommendation and [b-IETF RFC 4122] is recommended.

3.2.7 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.

3.2.8 message: The unit of communication between two logical services.

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

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

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

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

3.2.13 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 the creation of identity unique identifiers is outside the scope of this Recommendation and [b-IETF RFC 4122] is recommended.

4 Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms:

ADM Ad Management Service

ADS Ad Decision Service

CIS Content Information Service

HA High Availability

POIS Placement Opportunity Information Service

SIS Subscriber Information Service

UPID Unique Program Identifier

URI Uniform Resource Identifier

UUID Universally Unique Identifier

XML Extensible Markup Language

5 Conventions

5.1 Normative XML schema

Descriptions of messages, elements, and attributes are normative and, when combined with the normative XML schema document (see clause 2), 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 ITU-T J.380 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 Recommendation) 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.

5.2 Recommendation 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 clause 6 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 clause 6. 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 ITU-T J.380 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 Recommendation. Any ITU-T J.380.2 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 ITU-T J.380 defined or alternatively defined attribute may be included.

Figure 1 explains the schema illustration technique used throughout [ITU-T J.380.x]. Symbols and their meanings are explained within Figure 1.

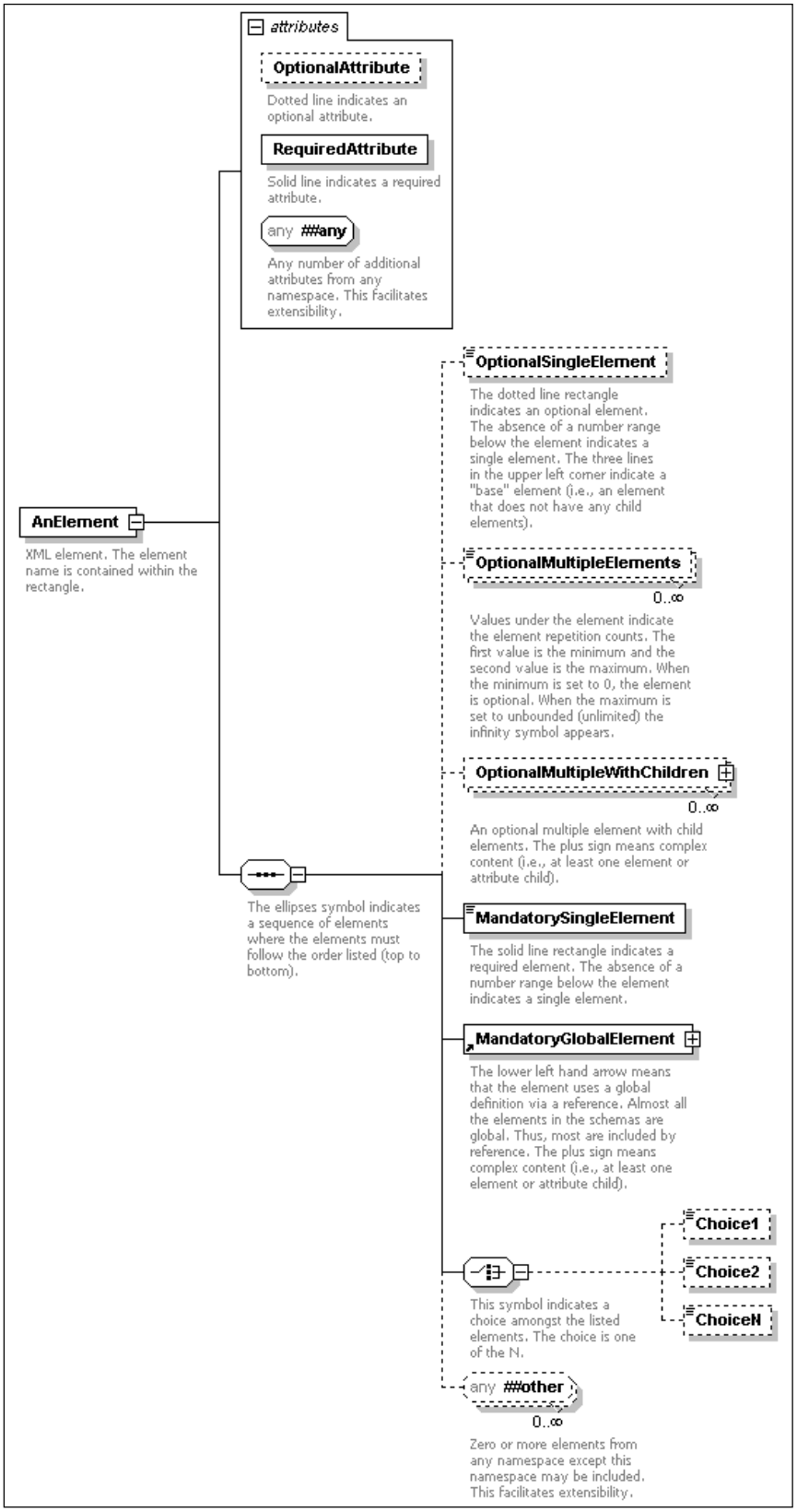

Figure 1 – Schema illustration explanation

6 XML namespaces

This Recommendation 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 Recommendation. Table 1 lists the prefix, the corresponding namespace, and a description of the defining Recommendation.

Table 1 – XML namespace declarations

Prefix	Namespace	Description
core	http://www.scte.org/schemas/130-2/2008a/core	ITU-T J.380.2 (i.e., this document)
xsd	http://www.w3.org/2001/XMLSchema	XML foundation. See [XMLSchemaP1].

The namespace is formatted as per the SCTE recommendation and includes the ITU-T J.380 version through the inclusion of a date which may minimally be a specification year identifier.

6.1 Unknown/unrecognized/unsupported XML elements and attributes

Generally, unknown, unrecognized or unsupported XML elements and attributes contained within ITU-T J.380 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 appropriate mechanisms. Elements and attributes which are prohibited by a namespace should generate an appropriate error response message.

6.2 Element order

Element order is constrained by the schemas and must be preserved throughout the 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).

6.3 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.

7 ITU-T J.380 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 Recommendation and may be addressed by [ITU-T J.380.7]. A message is the unit of communication between two logical services. All ITU-T J.380 identified logical services exchange information based on a common XML schema are 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 ITU-T J.380 logical service, then the appropriately paired response or acknowledgement message shall be returned.

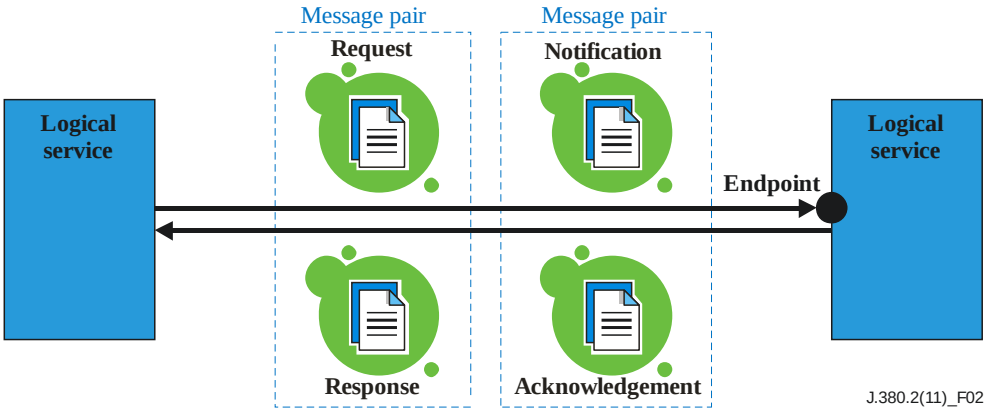

Figure 2 – Basic communication overview

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

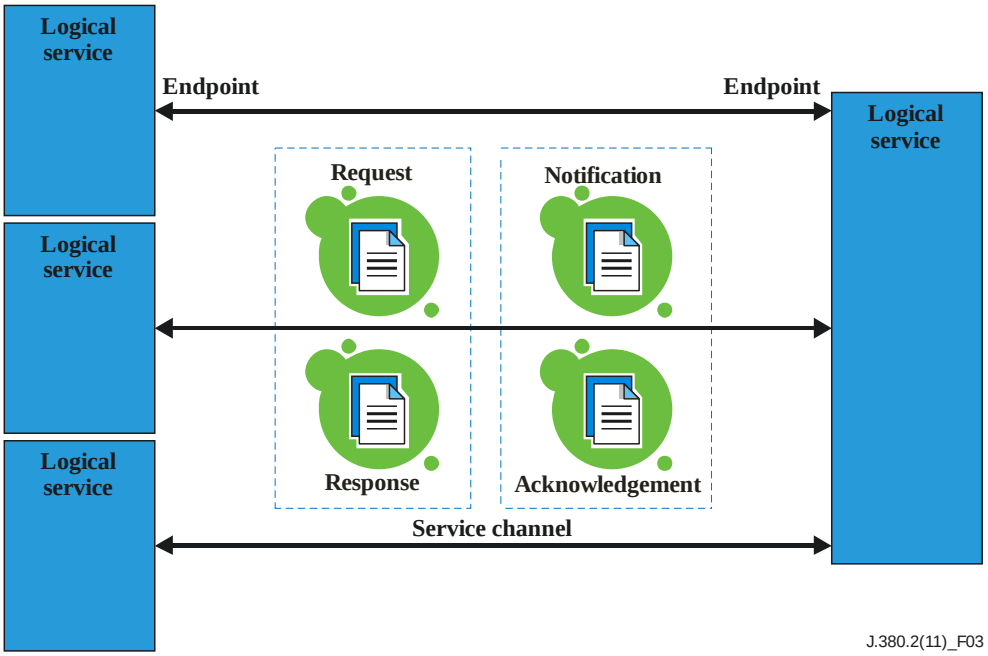

Figure 3 – Service channel illustrated

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 Recommendation.

NOTE – [ITU-T J.380.7] may provide additional details.

Since an ITU-T J.380 message element is the XML document root element, all ITU-T J.380 messages shall share a common base syntactic structure as defined in the following clauses.

7.1 Common schema for all messages

All ITU-T J.380 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 ITU-T J.380 message common attribute schema.

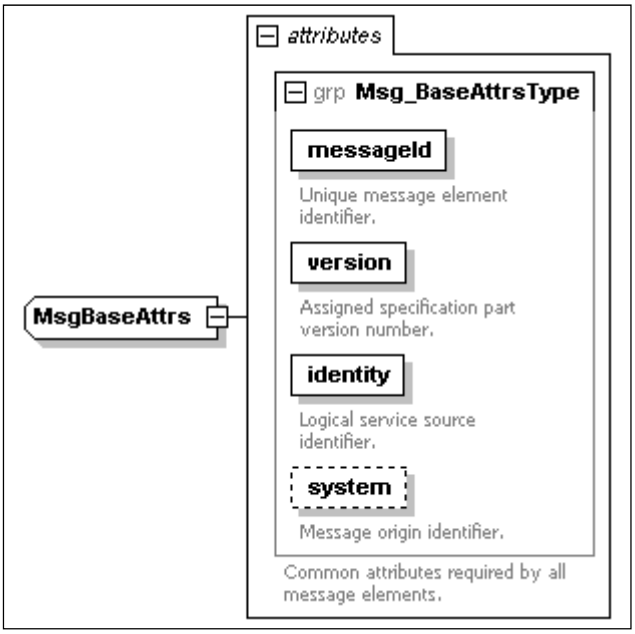

Figure 4 – ITU-T J.380 message common attribute schema

A single ITU-T J.380 message shall be identifiable via its @messageId attribute. The individual ITU-T J.380 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.

7.1.1 Semantic definitions for the ITU-T J.380 message common attributes

@messageId [Required, messageIdAttrType] – The message identifier. Every ITU-T J.380 message instance shall have a service channel unique value. The @messageId should be a universally unique identifier as defined by [b-IETF RFC 4122]. See clause 9.2.4 for additional information.

@version [Required, versionAttrType] – The ITU-T J.380 message specification version number. The value shall be specified by the individual ITU-T J.380 specification parts. For all messages defined herein, the value shall be "1.1". See clause 9.2.9 for the attribute's type definition.

@identity [Required, identityAttrType] – The origin logical service identifier. A globally unique identifier associated with the logical service. See clause 9.2.1 for additional information.

@system [Optional, systemAttrType] – The message source identifier. See clause 9.2.8 for additional information.

7.2 Request and notification messages base schema

Figures 5 and 6 illustrate the ITU-T J.380 request and notification messages base schema.

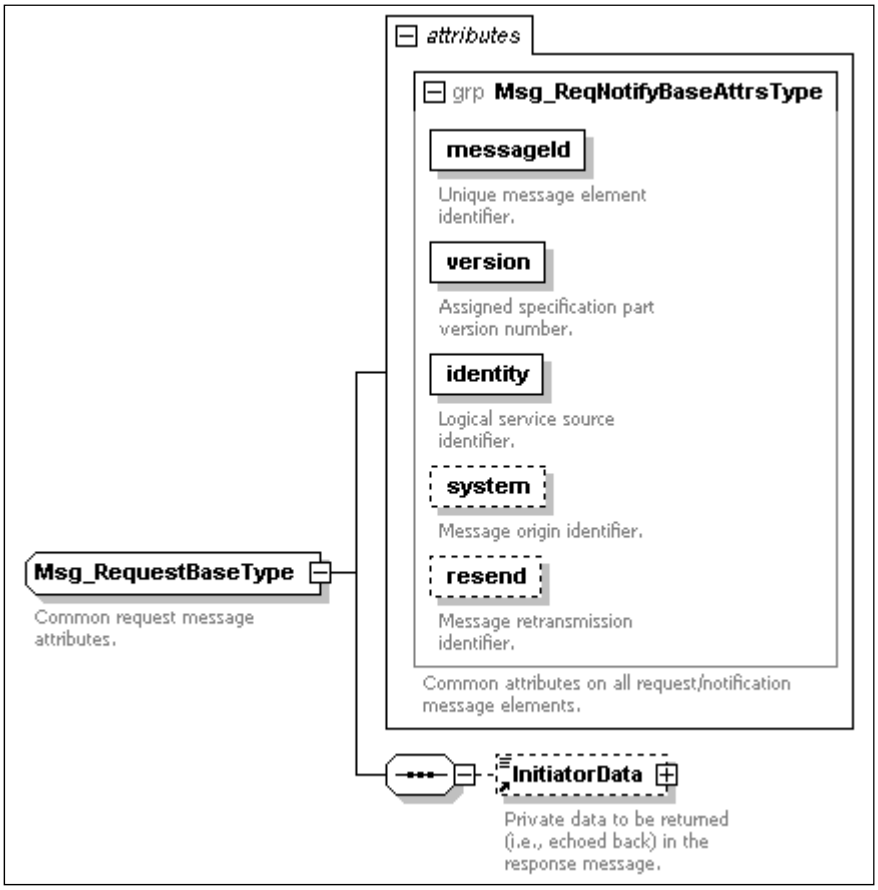

Figure 5 – Request message base schema

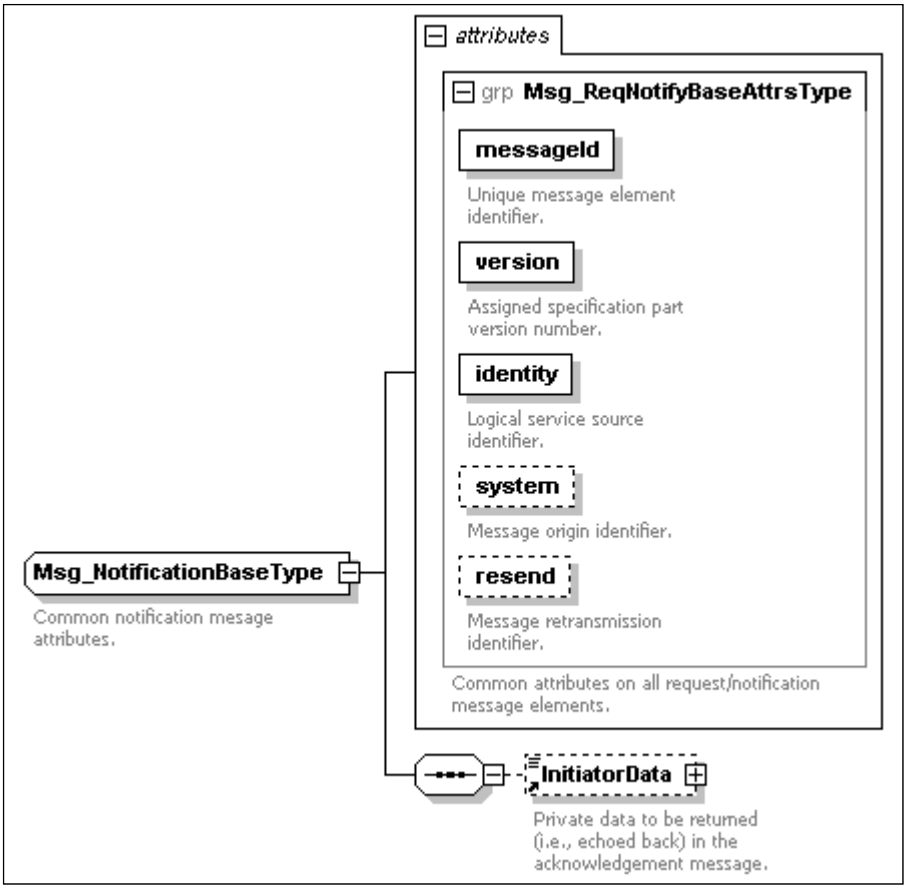

Figure 6 – Notification message base schema

7.2.1 Semantic definitions for the request and notification messages base schema

@messageId [Required, messageIdAttrType] – The message identifier. See clause 7.1.1 for additional information.

@version [Required, versionAttrType] – The message specification version. See clause 7.1.1 for additional information.

@identity [Required, identityAttrType] – The origin logical service identifier. See clause 7.1.1 for additional information.

@system [Optional, systemAttrType] – The message source identifier. See clause 7.1.1 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 clauses 7.4.7 and 9.2.7 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 clause 9.14 for additional information.

7.3 Response and acknowledgement messages base schema

Figures 7 and 8 illustrate the ITU-T J.380 response and acknowledgement messages base schema.

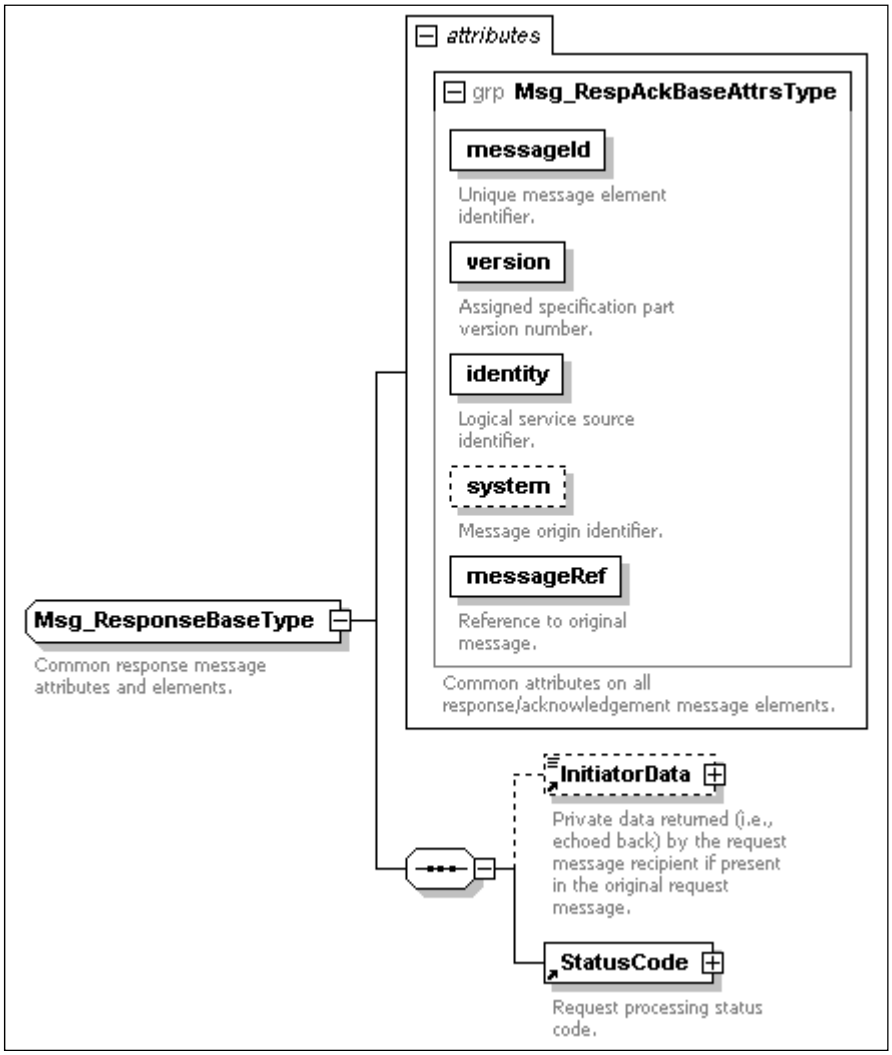

Figure 7 – Response message base schema

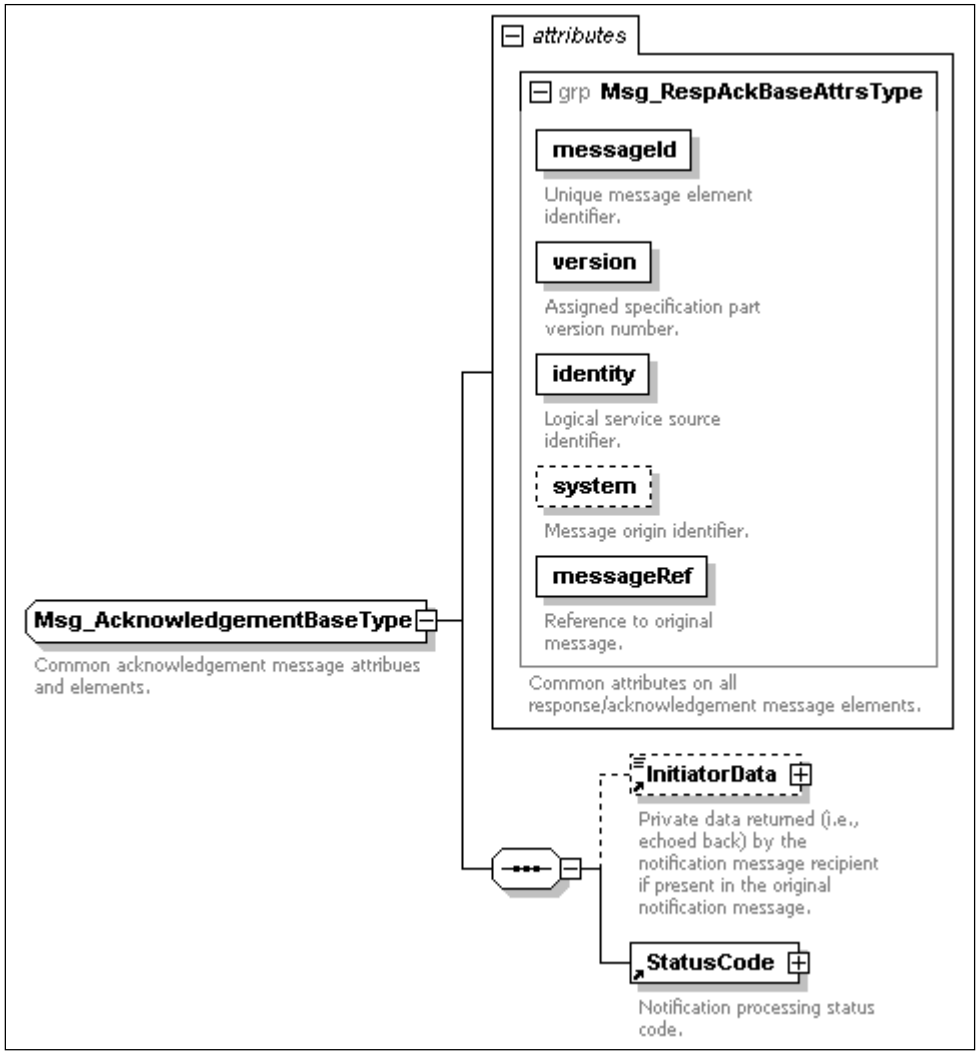

Figure 8 – Acknowledgement message base schema

7.3.1 Semantic definitions for the response and acknowledgement messages base schema

@messageId [Required, messageIdAttrType] – The message identifier. See clause 7.1.1 for additional information.

@version [Required, versionAttrType] – The message specification version. See clause 7.1.1 for additional information.

@identity [Required, identityAttrType] – The origin logical service identifier. See clause 7.1.1 for additional information.

@system [Optional, systemAttrType] – The message source identifier. See clause 7.1.1 for 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 ITU-T J.380 message. See clause 9.2.5 for additional information. Figure 9 illustrates the linkage.

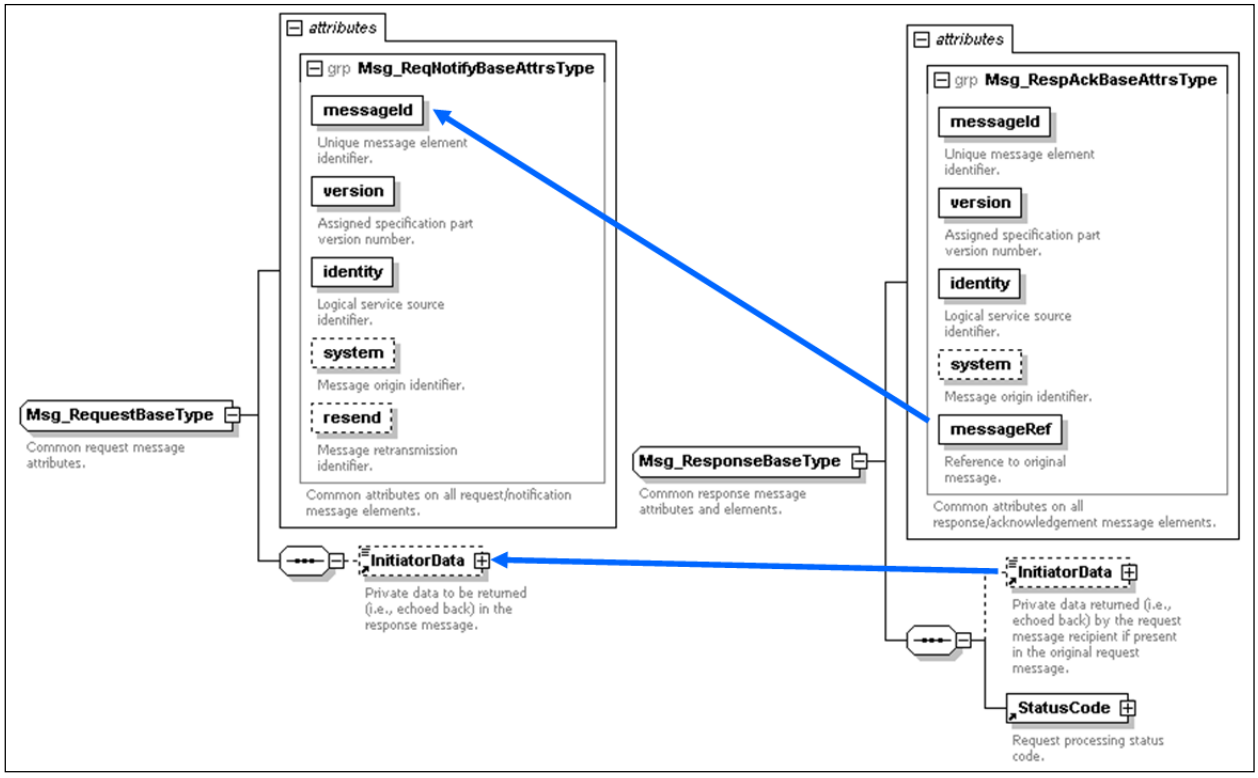

Figure 9 – @messageRef and InitiatorData paired message linkage

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 clause 9.14 for additional information.

StatusCode [Required] – An applicable processing status code. See clause 9.19 for additional information.

7.4 ITU-T J.380 message characterizations

7.4.1 Transport mechanisms

ITU-T J.380 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 – [ITU-T J.380.7] is expected to define the transport details.

ITU-T J.380 does not expressly require, prohibit, or mandate that a message exchange occurs directly between two ITU-T J.380 logical services. Messages may be passed and/or routed through intermediaries as long as such activity does not compromise the ITU-T J.380 message specification defined herein or any other ITU-T J.380 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 Recommendation and there is no relationship between the two service channels within the context of this Recommendation (i.e., this Recommendation 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.

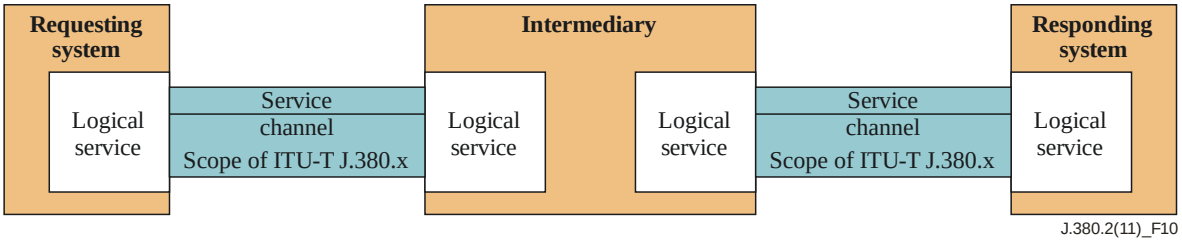

Figure 10 – Intermediary logical service

7.4.2 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 – [ITU-T J.380.7] is expected to define transport details.

7.4.3 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 – [ITU-T J.380.7] is expected to define the transport details.

7.4.4 Message timeliness

This document does not specify a maximum response or acknowledgement time limit. [ITU-T J.380.7] may specify a transport specific timeout value. Though a reliable transport is required by J.DPI-ASI, implementations should be prepared to handle the case when no paired response or acknowledgement message returns in a timely manner. Subsequent error behaviour and recovery is outside the scope of this Recommendation.

7.4.5 Message specification versioning

The specific ITU-T J.380 Recommendation version for an ITU-T J.380 message shall be identified as a combination of both the ITU-T J.380 part specific XML namespace (which contains a revision date identifier) and the ITU-T J.380 message's @version attribute. Using this identification schema, forward compatible transforms may be defined in order to facilitate future specification compatibility.

7.4.6 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 Recommendation parts for specific cases (such as ITU-T J.380.3 PlacementStatusNotification message). Refer to the individual Recommendations 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 signalled using the service status message exchange (i.e., a ServiceStatusNotification and ServiceStatusAcknowledgement message transaction).

7.4.7 High availability and message retransmission

Every logical service endpoint shall adhere to and implement the capabilities described in this clause 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 clause 9.14 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 clause 9.2.7 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 clauses describe the message initiator and message respondent capabilities and constraints.

7.4.7.1 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 Recommendation. 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 (Figure 11)

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 by 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 Recommendation. However, some behavioural constraints and possible implementation choices are outlined in clause 7.4.7.2.

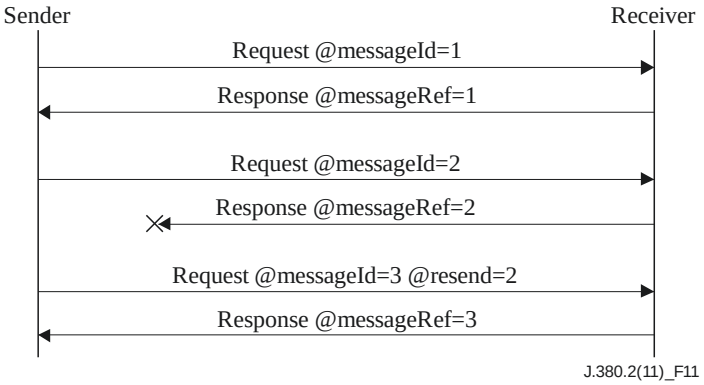

Figure 11 – Example resend message sequence

– 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.

7.4.7.2 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 a new state if a 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 this Recommendation; 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 A.1), 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.

7.4.8 List registration

For [ITU-T J.380.x], 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 re-coded 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.

7.5 Addressing

[ITU-T J.380.x] uses common, well-defined address formats whenever possible. The following clauses describe these formats.

7.5.1 Internet protocol version 4 (IPv4) address

An Internet protocol (IP) version 4 address shall conform to the format and characteristics as defined by [IETF RFC 3986]. Typically, the address is represented in dot decimal notation (also known as dotted quad notation, i.e., nnn.nnn.nnn.nnn). Figure 12 illustrates a few valid instances.

```
A basic IPv4 address.
171.70.222.82

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

Figure 12 – Example 1

7.5.2 Internet protocol version 6 (Ipv6) address

An Internet protocol (IP) version 6 address shall conform to the format and characteristics as defined by [IETF RFC 3986]. 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. As per [IETF RFC 3986], use of the standard "::" may be used to suppress repeating zeros. Figure 13 illustrates a few valid instances. See [IETF RFC 3986] for additional examples.

```
A basic Ipv4 address.
171.70.222.82

An Ipv4 address in a URL.
http://171.70.22.82
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]
```

Figure 13 – Example 2

7.5.3 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 IETF RFC 3986. See [IETF RFC 3986] for additional details. The individual specifications shall indicate if a port value is optional or required as appropriate. Figure 14 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/J.DPI-ASI/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/J.DPI-ASI/index.jsp
```

Figure 14 – Example 3

7.5.4 IEEE media access control (MAC) address

A 6 byte IEEE MAC address shall conform to the format and characteristics of hexadecimal representation as per [IEEE 802]. See [IEEE 802] 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 as per [IEEE 802].

Leading zeros shall not be omitted. Figure 15 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
```

Figure 15 – Example 4

8 ITU-T J.380.2 messages

Table 2 identifies the ITU-T J.380.2 messages defined herein that shall be supported and implemented by all ITU-T J.380 logical services.

Table 2 – ITU-T J.380.2 messages

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

8.1 Service check messages

Any ITU-T J.380 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 16 illustrates this message exchange.

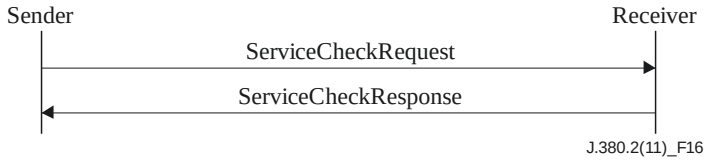

Figure 16 – Service check message exchange

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 ITU-T J.380 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 clauses.

8.1.1 ServiceCheckRequest message schema

Figure 17 illustrates the ServiceCheckRequest message schema.

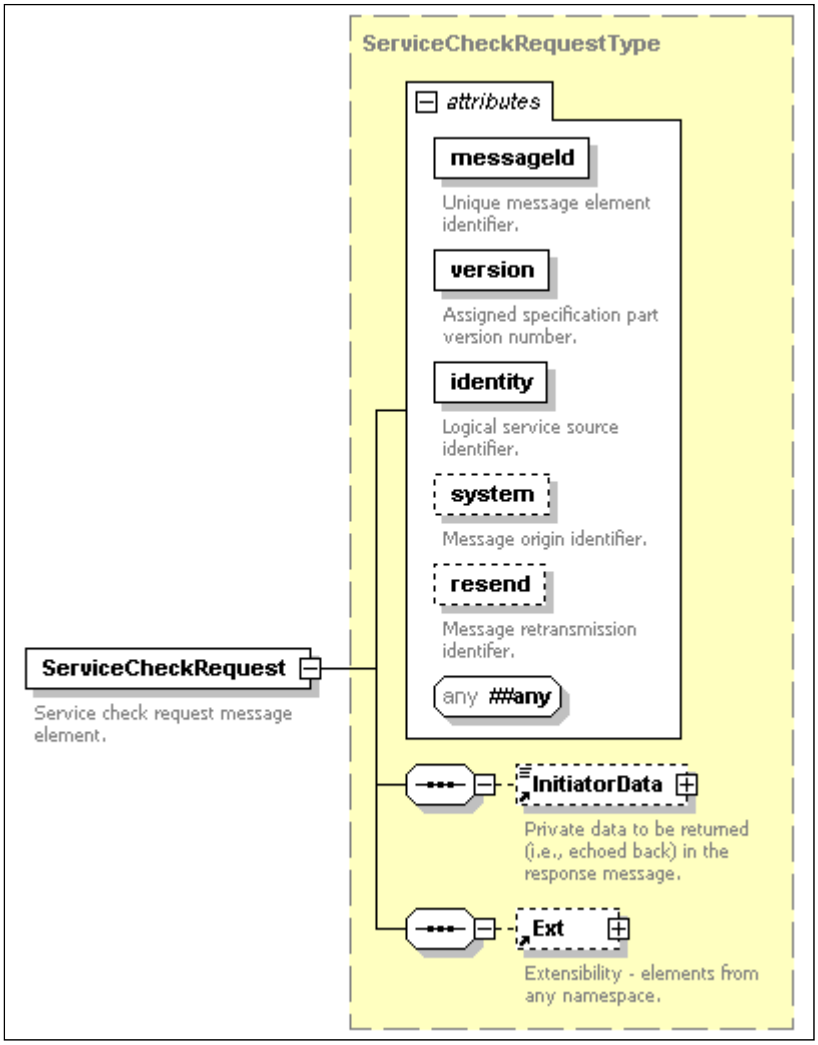

Figure 17 – ServiceCheckRequest message schema

The ServiceCheckRequest message may be an empty element.

8.1.1.1 Semantic definitions for the ServiceCheckRequest message

@messageId [Required, messageIdAttrType] – The message identifier. See clause 7.1.1 for additional information.

@version [Required, versionAttrType] – The message specification version. See clause 7.1.1 for additional information.

@identity [Required, identityAttrType] – The origin logical service identifier. See clause 7.1.1 for additional information.

@system [Optional, systemAttrType] – The message source identifier. See clause 7.1.1 for additional information.

@resend [Optional, resendAttrType] – The message is a retransmit of a previous message identified by the supplied message identifier. See clause 7.2.1 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 clause 7.2.1 for additional information.

Ext [Optional] – A container for any additional elements from any namespace. See clause 9.12 for additional information.

8.1.2 ServiceCheckResponse message schema

Figure 18 below illustrates the ServiceCheckResponse element schema sent following the reception and processing of a ServiceCheckRequest message.

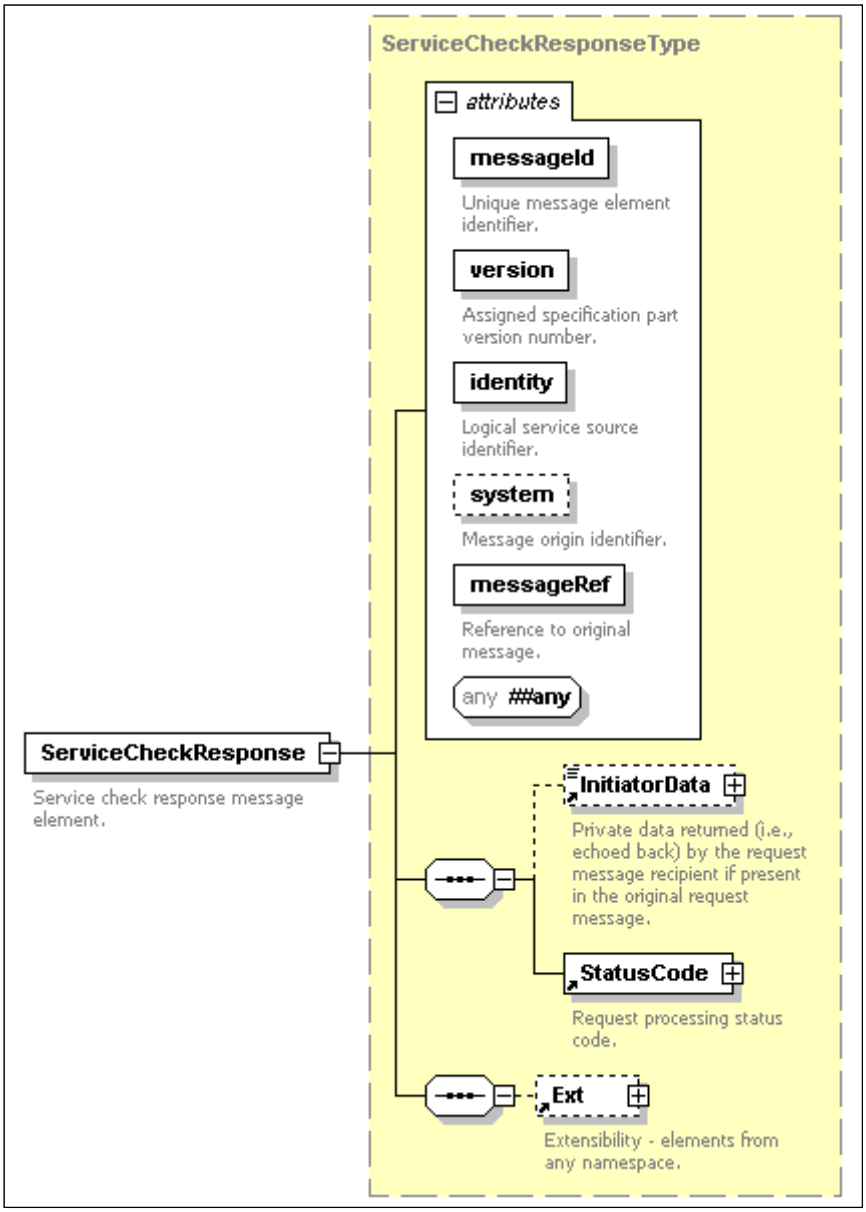

Figure 18 – ServiceCheckResponse message schema

8.1.2.1 Semantic definitions for the ServiceCheckResponse message

@messageId [Required, messageIdAttrType] – The message identifier. See clause 7.1.1 for additional information.

@version [Required, versionAttrType] – The message specification version. See clause 7.1.1 for additional information.

@identity [Required, identityAttrType] – The origin logical service identifier. See clause 7.1.1 for additional information.

@system [Optional, systemAttrType] – The message source identifier. See clause 7.1.1 for additional information.

@messageRef [Required, messageRefAttrType] – A reference to the ITU-T J.380 ServiceCheckRequest message element initiating this message exchange. The value shall be the ServiceCheckRequest message's @messageId attribute value. See clause 7.3.1 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 clause 7.3.1 for additional information.

StatusCode [Required] – An applicable processing status code specific to the ServiceCheckRequest message processing. See clause 9.19 for additional information. The StatusCode element shall contain the service health status.

Ext [Optional] – A container for any additional elements from any namespace. See clause 9.12 for additional information.

8.1.3 Service check message examples

A ServiceCheckRequest message (Figure 19) with a positive (i.e., success) ServiceCheckResponse message (Figure 20).

A negative ServiceCheckResponse message (Figure 21) assuming the same request message as Example 5.

```
<ServiceCheckRequest messageId="8FA2C7BD-2751-4D18-A2C6-372D8CE5F100" version="1.1"
identity="ADMLogicalService_1170FEF4-C19B-450E-B624-421B23F525F4" system="adm"/>
```

Figure 19 – Example 5

```
<ServiceCheckResponse messageId="8FA2C7BD-2751-4D18-A2C6-372D8CE5F105" version="1.1"
identity="ADSLogicalService_1170FEF4-C19B-450E-B624-421B23F525F5" system="ads"
messageRef="8FA2C7BD-2751-4D18-A2C6-372D8CE5F100">
  <StatusCode class="0"/>
</ServiceCheckResponse>
```

Figure 20 – Example 6

```
<ServiceCheckResponse messageId="8FA2C7BD-2751-4D18-A2C6-372D8CE5F106" version="1.1"
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>
```

Figure 21 – Example 7

A ServiceCheckRequest message (Figure 22) with a positive (i.e., success) ServiceCheckResponse message (Figure 23) illustrating the InitiatorData elements inclusion.

```
<ServiceCheckRequest messageId="8FA2C7BD-2751-4D18-A2C6-372D8CE5F130" version="1.1"
identity="ADMLogicalService_1170FEF4-C19B-450E-B624-421B23F525F4" system="adm">
  <InitiatorData secret="true">Hidden secrets.</InitiatorData>
</ServiceCheckRequest>
```

Figure 22 – Example 8

```
<ServiceCheckResponse messageId="8FA2C7BD-2751-4D18-A2C6-372D8CE5F135" version="1.1"
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>
```

Figure 23 – Example 9

8.2 Service status messages

Any ITU-T J.380 logical service may at any time 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 24 illustrates this message exchange.

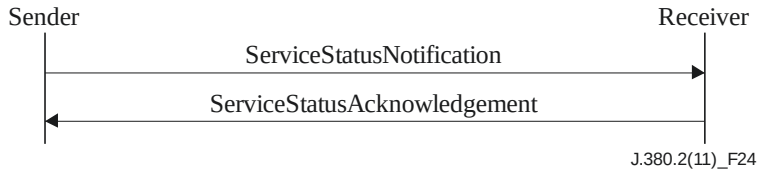

Figure 24 – Service status message exchange

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

– Provides unsolicited informational status

– Facilitates general event notification

The following clauses illustrate the service status message schemas.

8.2.1 ServiceStatusNotification message schema

Figure 25 illustrates the ServiceStatusNotification message's schema.

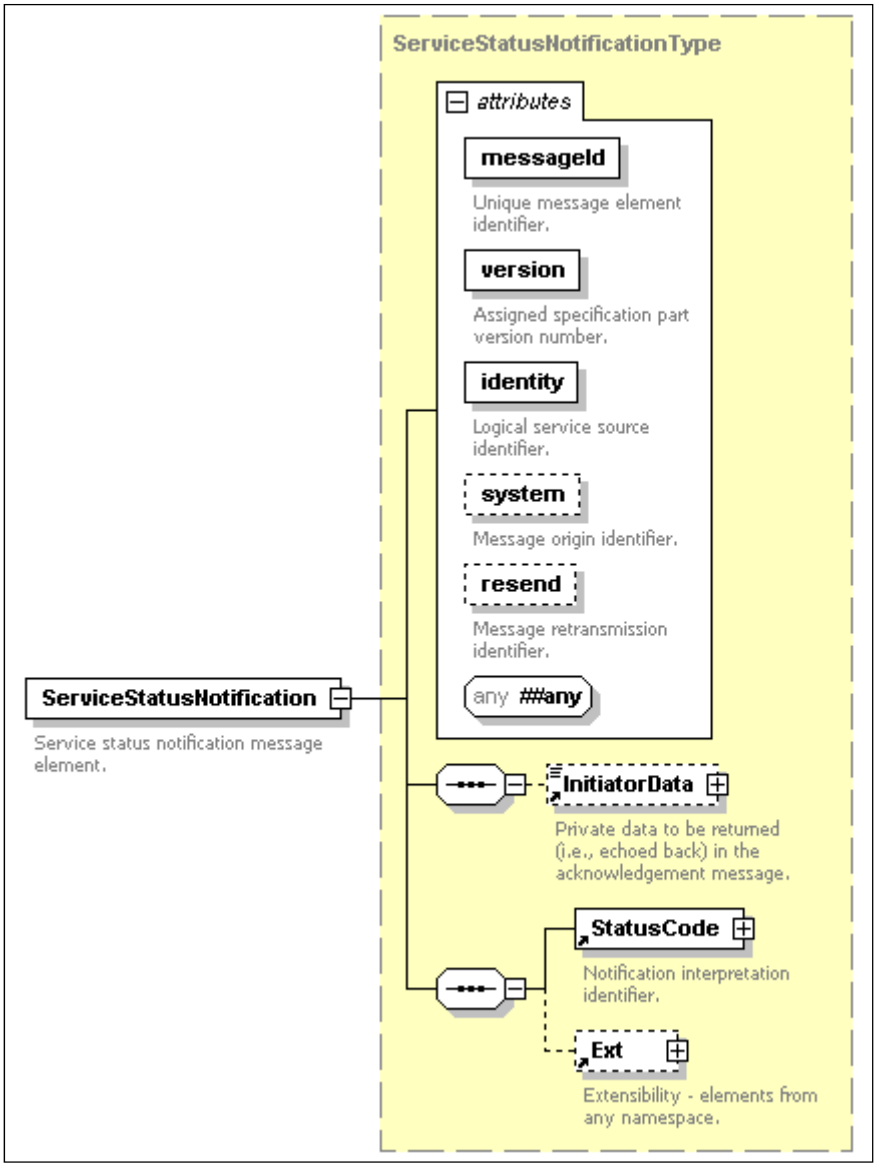

Figure 25 – ServiceStatusNotification message schema

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.

8.2.1.1 Semantic definitions for the ServiceStatusNotification message

@messageId [Required, messageIdAttrType] – The message identifier. See clause 7.1.1 for additional information.

@version [Required, versionAttrType] – The message specification version. See clause 7.1.1 for additional information.

@identity [Required, identityAttrType] – The origin logical service identifier. See clause 7.1.1 for additional information.

@system [Optional, systemAttrType] – The message source identifier. See clause 7.1.1 for additional information.

@resend [Optional, resendAttrType] – The message is a retransmit of a previous message identified by the supplied message identifier. See clause 7.2.1 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 clause 7.2.1 for additional information.

StatusCode [Required] – An applicable notification status code specific to the ServiceStatusNotification message. See clause 9.19 for additional information.

Ext [Optional] – A container for any additional elements from any namespace. See clause 9.12 for additional information.

8.2.2 ServiceStatusAcknowledgement message schema

Figure 26 illustrates the ServiceStatusAcknowledgement element schema sent following the reception and processing of a ServiceStatusNotification message.

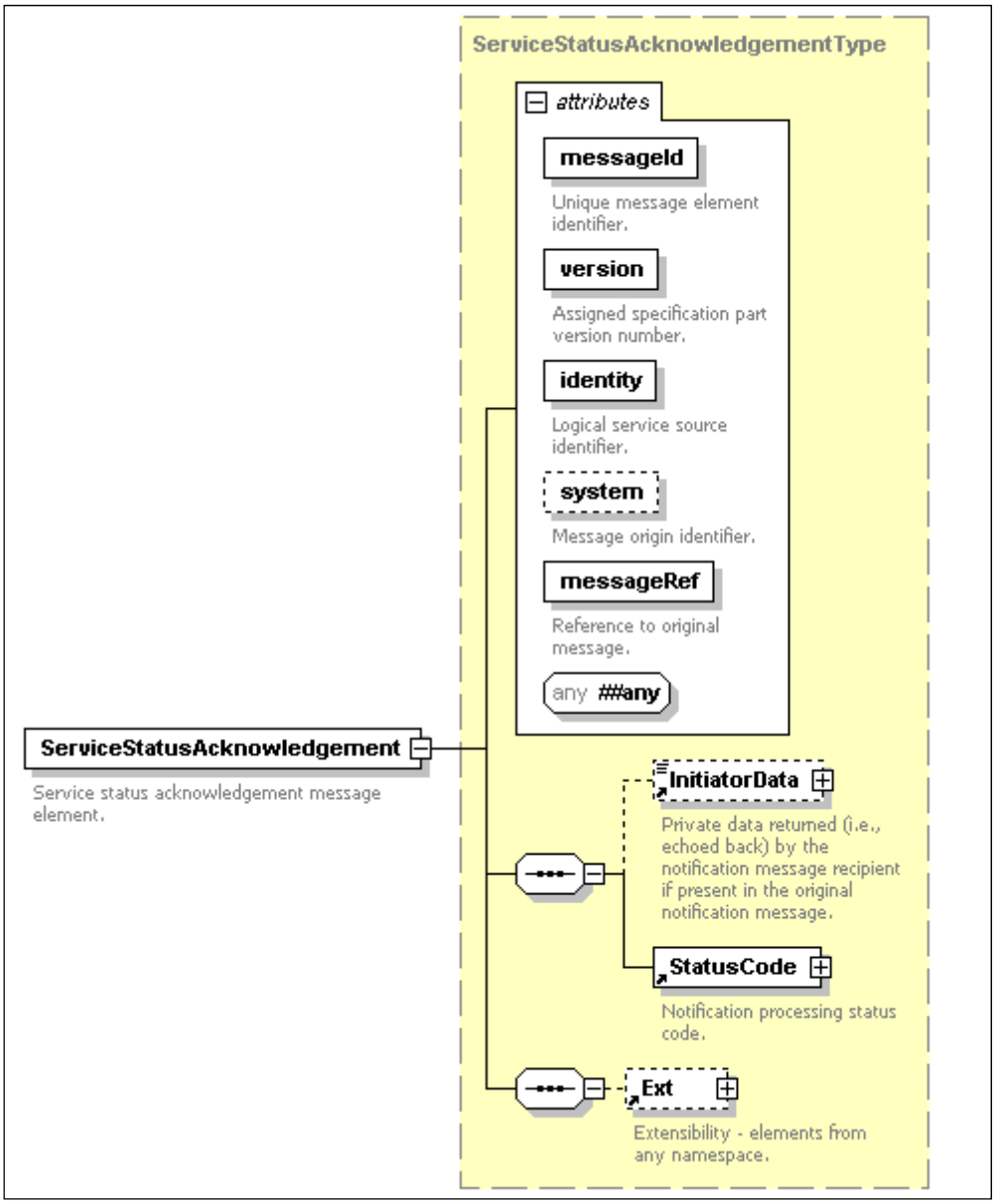

Figure 26 – ServiceStatusAcknowledgement message schema

8.2.2.1 Semantic Definitions for the ServiceStatusAcknowledgement Message

@messageId [Required, messageIdAttrType] – The message identifier. See clause 7.1.1 for additional information.

@version [Required, versionAttrType] – The message specification version. See clause 7.1.1 for additional information.

@identity [Required, identityAttrType] – The origin logical service identifier. See clause 7.1.1 for additional information.

@system [Optional, systemAttrType] – The message source identifier. See clause 7.1.1 for additional information.

@messageRef [Required, messageRefAttrType] – A reference to the ITU-T J.380 ServiceStatusNotification message element initiating this message exchange. The value shall be the ServiceStatusNotification message's @messageId attribute value. See clause 7.3.1 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 clause 7.3.1 for additional information.

StatusCode [Required] – An applicable status code specific to the ServiceStatusNotification message processing. See clause 9.19 for additional information.

Ext [Optional] – A container for any additional elements from any namespace. See clause 9.12 for additional information.

8.2.3 Service status message examples

A notification (Figure 27) and an acknowledgement (Figure 28) message sequence.

An error notification message (Figure 29) and an acknowledgement message (Figure 30).

```
<ServiceStatusNotification messageId="87648E84-9D5E-11DB-96CA-005056C00009" version="1.1"
identity="ADMLogicalService_1170FEF4-C19B-450E-B624-421B23F525F4" system="adm">
  <StatusCode class="3">
    <Note>Information.</Note>
    <Note>Low disk space.</Note>
  </StatusCode>
</ServiceStatusNotification>
```

Figure 27 – Example 10

```
<ServiceStatusAcknowledgement messageId="87648E84-9D5E-11DB-96CA-005056C0000a" version="1.1"
identity="ADSLogicalService_1170FEF4-C19B-450E-B624-421B23F525F5" system="ads"
messageRef="87648E84-9D5E-11DB-96CA-005056C00009">
  <StatusCode class="0"/>
</ServiceStatusAcknowledgement>
```

Figure 28 – Example 11

```
<ServiceStatusNotification messageId="87648E84-9D5E-11DB-96CA-005056C0000C" version="1.1"
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>
```

Figure 29 – Example 12

```
<ServiceStatusAcknowledgement messageId="87648E84-9D5E-11DB-96CA-005056C0000D" version="1.1"
identity="ADMLogicalService_1170FEF4-C19B-450E-B624-421B23F525F4" system="adm"
messageRef="87648E84-9D5E-11DB-96CA-005056C0000C">
  <StatusCode class="0"/>
</ServiceStatusAcknowledgement>
```

Figure 30 – Example 13

9 ITU-T J.380 core attribute types and elements

The following clauses define the ITU-T J.380 core types including common attribute types and element definitions which may appear in any ITU-T J.380 XML element definition. The common attribute definitions may be utilized by any element including the root message elements.

9.1 Semantic definitions for ITU-T J.380 core types

9.1.1 dateTimeTimezoneType simple type

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

9.1.2 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).

9.2 Semantic definitions for ITU-T J.380 core attribute types

9.2.1 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 [b-IETF RFC 4122] and the string shall not be empty.

The logical service blocks of clause 7, 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 31. 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.

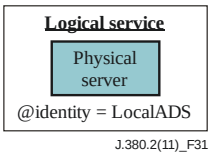

Figure 31 – Basic @identity attribute mapping

More complicated implementations may be utilized and Figure 32 illustrates additional possible implementations but not all possible implementations.

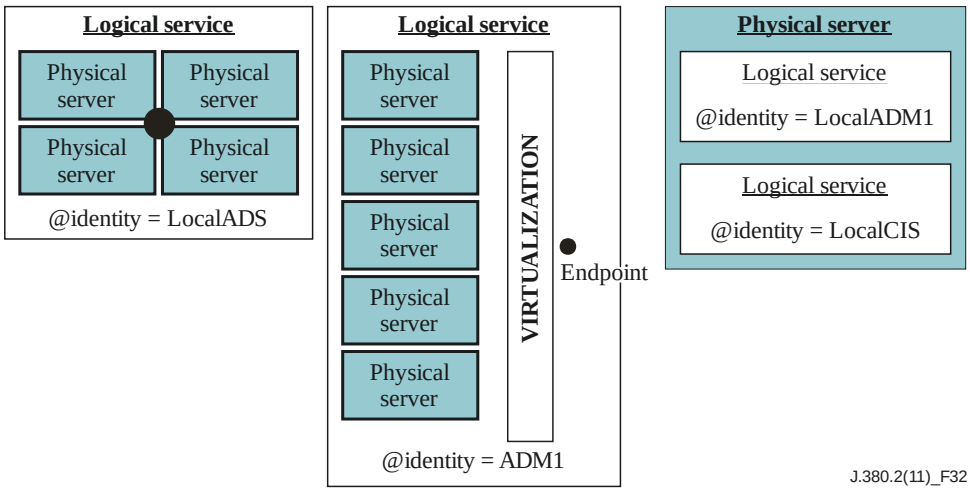

Figure 32 – Complex @identity attribute mappings

In Figure 32, 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 a single identifiable entity.

9.2.2 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 [b-IETF RFC 4122] and the string shall not be empty.

9.2.3 mediaAvailableAttrType attribute type

mediaAvailableAttrType [xsd:29ventid] – 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.

9.2.4 messageIdAttrType attribute type

messageIdAttrType [nonEmptyStringType] – This attribute type, referenced as the @messageId attribute, is a service channel unique identifier for a message. Every ITU-T J.380 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 [b-IETF RFC 4122].

9.2.5 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.

9.2.6 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).

9.2.7 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 clause 7.4.7 for additional information.

9.2.8 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.

9.2.9 versionAttrType attribute type

versionAttrType [nonEmptyStringType] – This attribute type, referenced as the @version attribute, is the ITU-T J.380 message specification version number. The value shall be specified by the individual parts of [ITU-T J.380.x] and shall not be empty. Refer to [ITU-T J.380.x] Recommendation where the message is defined for the currently specified value. Refer to clause 7.1.1 for the ITU-T J.380.2 assigned value.

9.3 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 [ITU-T J.380.7] which may supply additional information and usage criterion.

9.3.1 Address element schema

Figure 33 illustrates the Address element's schema.

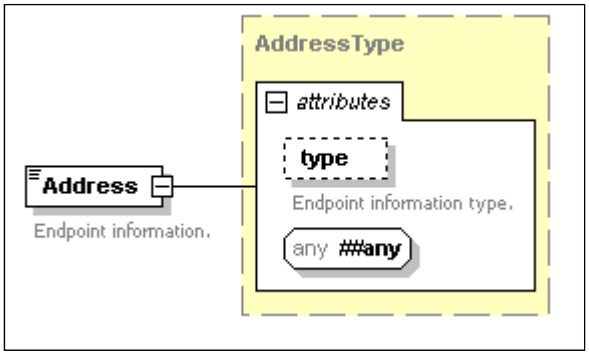

Figure 33 – Address element schema

9.3.1.1 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. [ITU-T J.380.7] may supply additional information and examples may include 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 clause 7.5 as applicable. [ITU-T J.380.7] may supply additional format details as appropriate, as these may be outside the scope of this Recommendation.

9.3.2 Address element examples

```
<Address type="HTTP">http://25.35.45.55:80/J.DPI-ASI</Address>
<Address type="SOAP">adselector1.ads.com/default</Address>
<Address>www.foobar.com/J.380.2/PlacementReq</Address>
```

9.4 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 Recommendation.

9.4.1 AdType element schema

Figure34 illustrates the AdType element's schema.

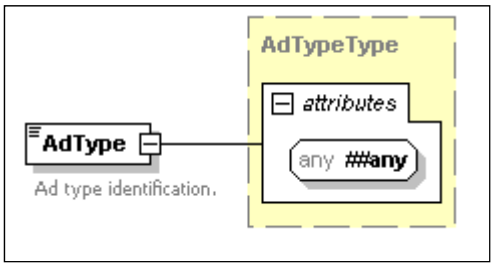

Figure 34 – AdType element schema

9.4.1.1 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.

9.4.2 AdType element examples

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

9.5 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. For ADI information, see [b-CLAD I1-1] and [b-CLAD I1-2].

9.5.1 AssetRef element schema

Figure 35 illustrates the AssetRef element's schema.

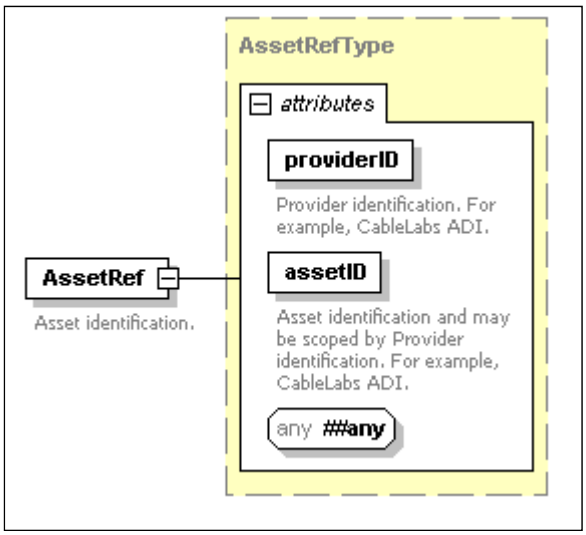

Figure 35 – AssetRef Element Schema

9.5.1.1 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 ADI 1.1 or ADI 2.0 specification 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 ADI 1.1 or ADI 2.0 specification 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.

9.5.2 AssetRef element examples

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

9.6 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 the adm:ListADMServicesResponse message in [ITU-T J.380.3].

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 behaviour 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 Recommendation.

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 Recommendation. 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.

9.6.1 Callout element schema

Figure 36 illustrates the Callout element's schema.

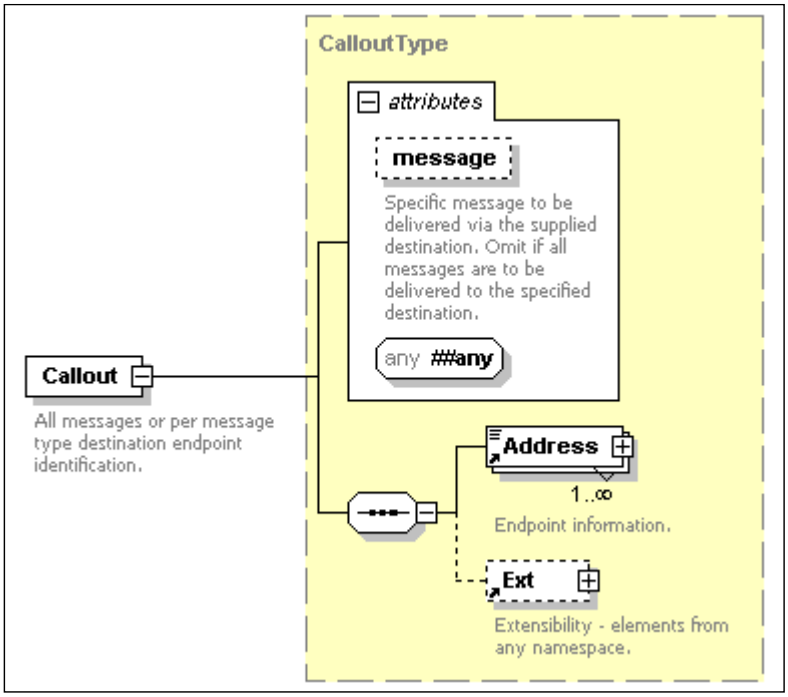

Figure 36 – Callout element schema

9.6.1.1 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 clause 9.3 for additional information.

Ext [Optional] – A container for any additional elements from any namespace. See clause 9.12 for additional information.

The Callout element shall not be empty.

9.6.2 Callout element examples

Figure 37 illustrates a default endpoint declaration. All communication is expected to use this interface.

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

Figure 37 – Example 14

Figure 38 illustrates a message specific endpoint declaration.

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

Figure 38 – Example 15

Figure 39 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>
```

Figure 39 – Example 16

9.7 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.

9.7.1 Content element schema

Figure 40 illustrates the Content element's schema.

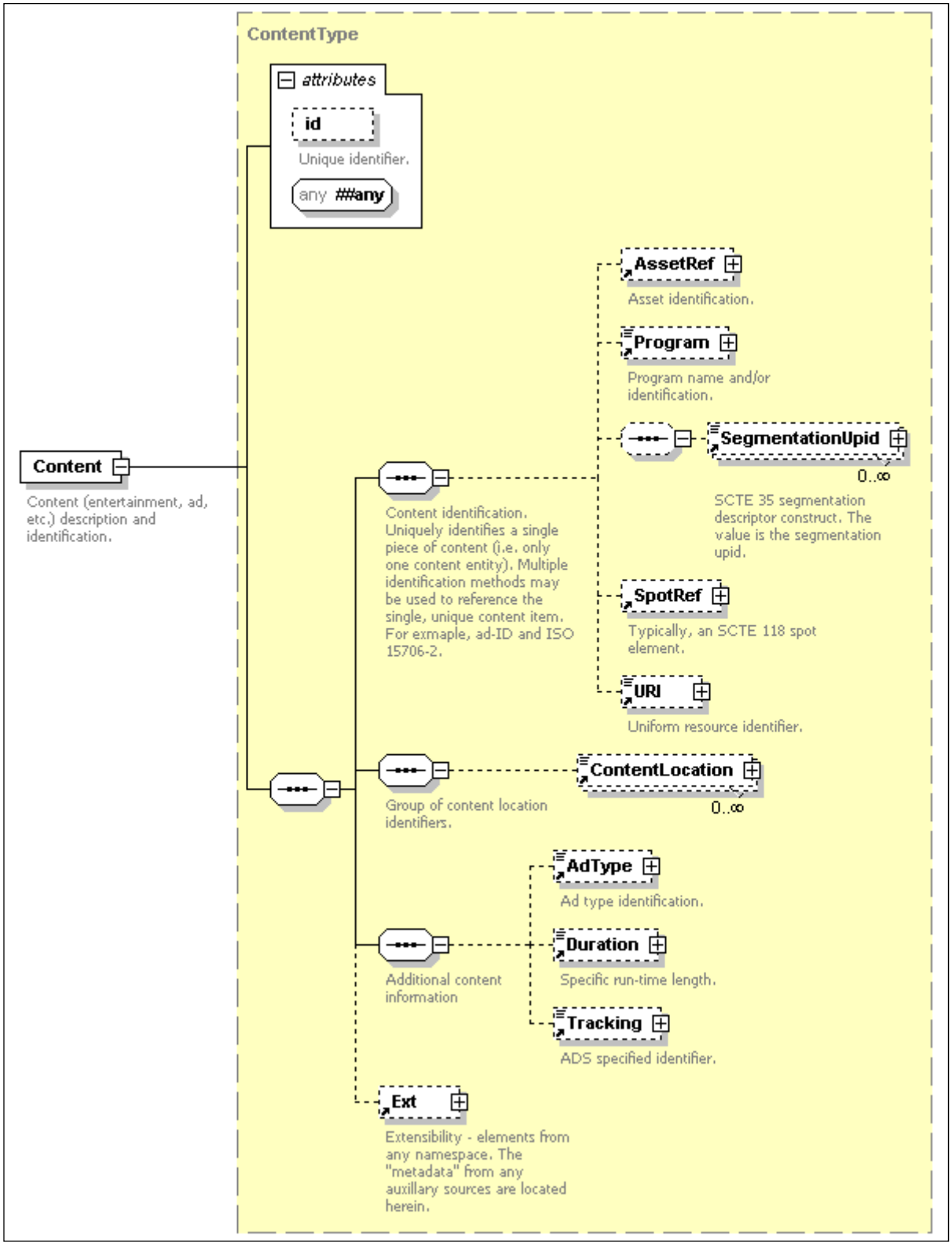

Figure 40 – Content element schema

9.7.1.1 Semantic definitions for the Content element

@id [Optional, idAttrType] – An 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 clause 9.5 for additional information.

Program [Optional] – Content asset identification typically used when an SCTE 35 splice_insert() section identifies the asset. See clause 9.16 for additional information.

SegmentationUpid [Optional] – Zero or more elements identifying a content asset. Typically, this element is used when an SCTE 35 segmentation_descriptor() identifies the asset. See clause 9.17 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 identifies the asset. See clause 9.18 for additional information.

URI [Optional] – Content asset identification via a uniform resource identifier (URI). See clause 9.21 for additional information.

ContentLocation [Optional] – Zero or more asset location specifiers. See clause 9.9 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 clause 9.4 for additional information.

Duration [Optional] – The content run-time length. See clause 9.11 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 clause 9.20 for additional information.

Ext [Optional] – A container for any additional elements from any namespace. See clause 9.12 for additional information.

The Content element's value should not be empty.

9.7.2 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>
```

9.8 ContentDataModel element

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

9.8.1 ContentDataModel element schema

Figure 41 illustrates the ContentDataModel element's schema.

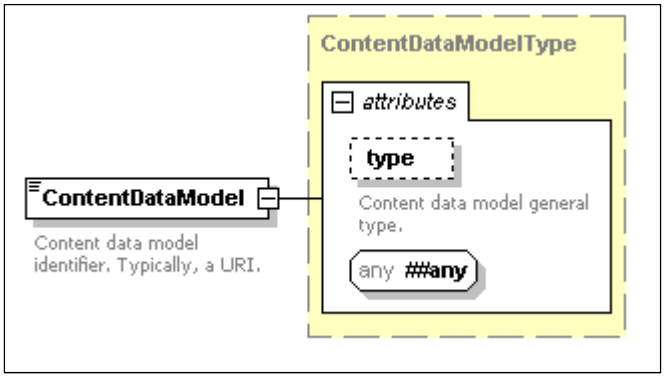

Figure 41 – ContentDataModel element schema

9.8.1.1 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 Recommendation. The values shall appear exactly as they appear in Table 3.

Table 3 – ContentDataModel element's @type attribute defined values

Value	Description
CLADI_1.1	Model represented by the CableLabs ADI 1.1 specification.
CLADI_2.0	Model represented by the CableLabs ADI 2.0 specification.
SCTE118-2	Model represented by [b-SCTE 118-2].
SCTE118-3	Model represented by [SCTE 118-3].
…	User defined and outside the scope of this Recommendation. The string shall be prefixed with the text "private:".

@##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 [IETF RFC 3986] for additional information.

9.8.2 ContentDataModel element examples

```
  <ContentDataModel type="CLADI_2.0">http://www.cablelabs.com/2006-05-05/ADI2</ContentDataModel>

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

  <ContentDataModel type="SCTE118-3">http://www.scte.org/schemas/118-3/2006</ContentDataModel>
```

9.9 ContentLocation element

The ContentLocation element facilitates specifying an asset's place. The element value may be any valid URI. See [IETF RFC 3986] for additional information.

9.9.1 ContentLocation element schema

Figure 42 illustrates the ContentLocation element's schema.

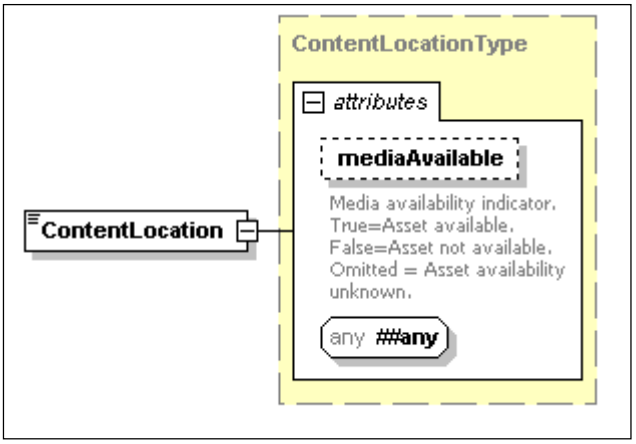

Figure 42 – ContentLocation element schema

9.9.1.1 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 clause 9.2.3 for additional information.

@##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 [IETF RFC 3986] for additional information.

9.9.2 ContentLocation element examples

```
<ContentLocation mediaAvailable="true">
  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 and a location URI.

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 URI has been provided (though one could have been supplied.)

The final example element provides a location URI 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.

9.10 CurrentDateTime element

The CurrentDateTime element provides the current date and time value which shall include a timezone indicator. See clause 9.1.1 for additional information.

9.10.1 CurrentDateTime element schema

Figure 43 illustrates the CurrentDateTime element's schema.

Figure 43 – CurrentDateTime element schema

9.10.1.1 Semantic definitions for the CurrentDateTime element

The CurrentDateTime element's value is of type core:dateTimeTimezoneType and shall not be empty. See clause 9.1.1 for additional information.

9.10.2 CurrentDateTime element examples

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

9.11 Duration element

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

9.11.1 Duration element schema

Figure 44 illustrates the Duration element's schema.

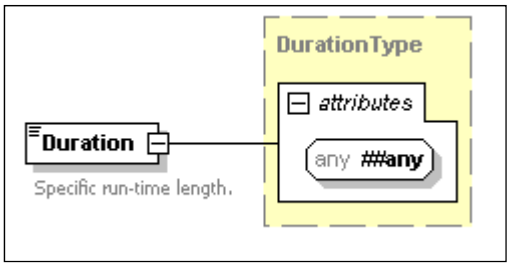

Figure 44 – Duration element schema

9.11.1.1 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.

9.11.2 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→
```

9.12 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.

9.12.1 Ext element schema

Figure 45 illustrates the Ext element's schema.

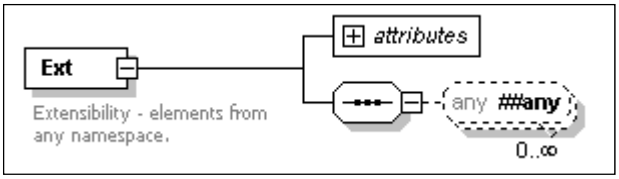

Figure 45 – Ext element schema

9.12.1.1 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.)

9.12.2 Ext element examples

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

9.13 ExternalStatusCode element

The ExternalStatusCode element allows for a detailed status code to be supplied from an external (non ITU-T J.380) specification. The element identifies the status code source and may be augmented with optional descriptive text.

9.13.1 ExternalStatusCode schema

Figure 46 illustrates the ExternalStatusCode element's schema.

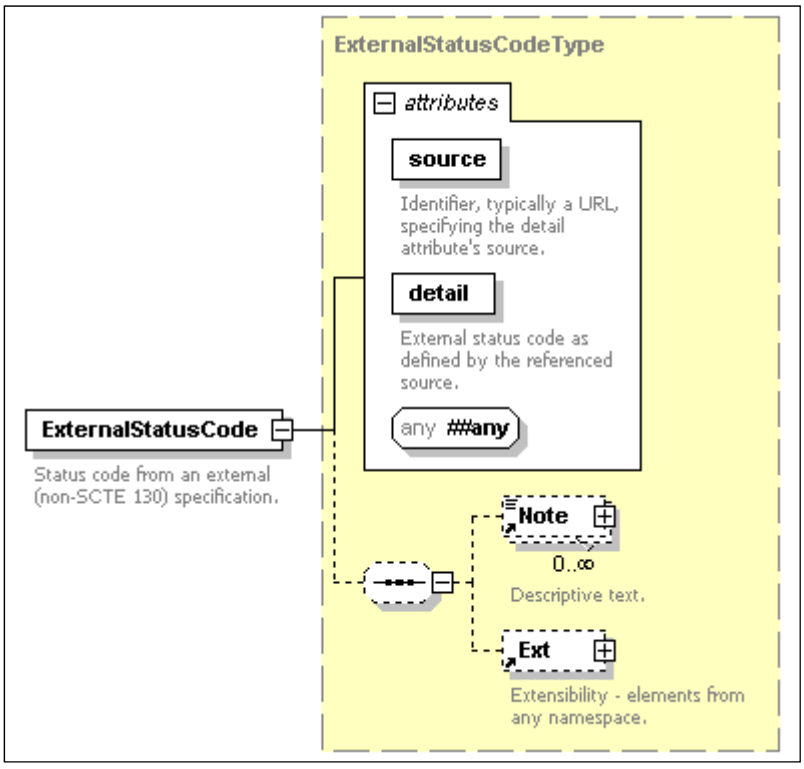

Figure 46 – ExternalStatusCode element schema

9.13.1.1 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 [IETF RFC 3986], and shall not be empty. Table 4 lists the defined values which may be extended by private agreement outside the scope of this Recommendation.

Table 4 – ExternalStatusCode element's @source attribute values

@source Value	Description
http://www.scte.org/schemas/30/2007	SCTE 30
http://www.scte.org/schemas/118-3/2006	SCTE 118-3
…	User defined and outside the scope of this Recommendation.

@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 clause 9.15 for additional information regarding the Note element.

Ext [Optional] – A container for any additional elements from any namespace. See clause 9.12 for additional information.

9.13.2 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>
```

9.14 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.

9.14.1 InitiatorData element Schema

Figure 47 illustrates the InitiatorData element's schema.

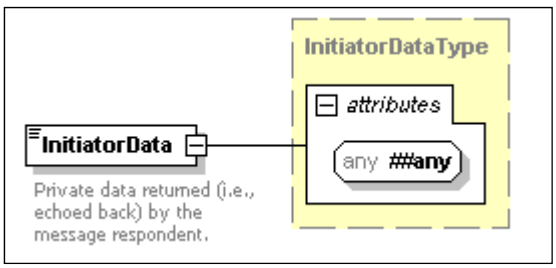

Figure 47 – InitiatorData element schema

9.14.1.1 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).

9.14.2 InitiatorData element examples

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

9.15 Note element

The Note element carries descriptive text.

9.15.1 Note element schema

Figure 48 illustrates the Note element's schema.

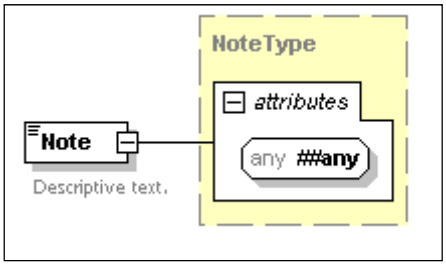

Figure 48 – Note element schema

9.15.1.1 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.

9.15.2 Note element examples

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

9.16 Program element

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

9.16.1 Program element schema

Figure 49 illustrates the Program element's schema.

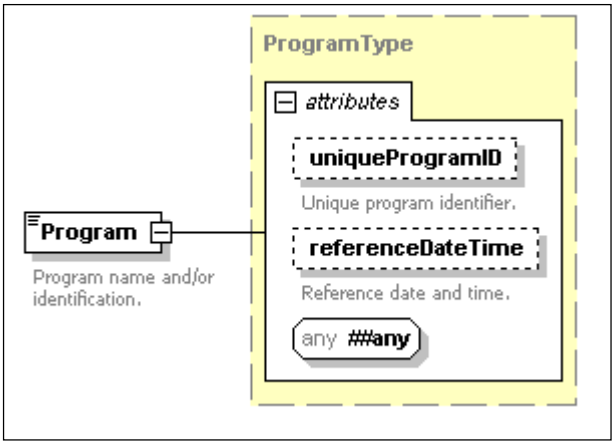

Figure 49 – Program element schema

9.16.1.1 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 clause 9.1.1 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.

9.16.2 Program element examples

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

9.17 SegmentationUpid element

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

9.17.1 SegmentationUpid element schema

Figure 50 illustrates the SegmentationUpid element's schema.

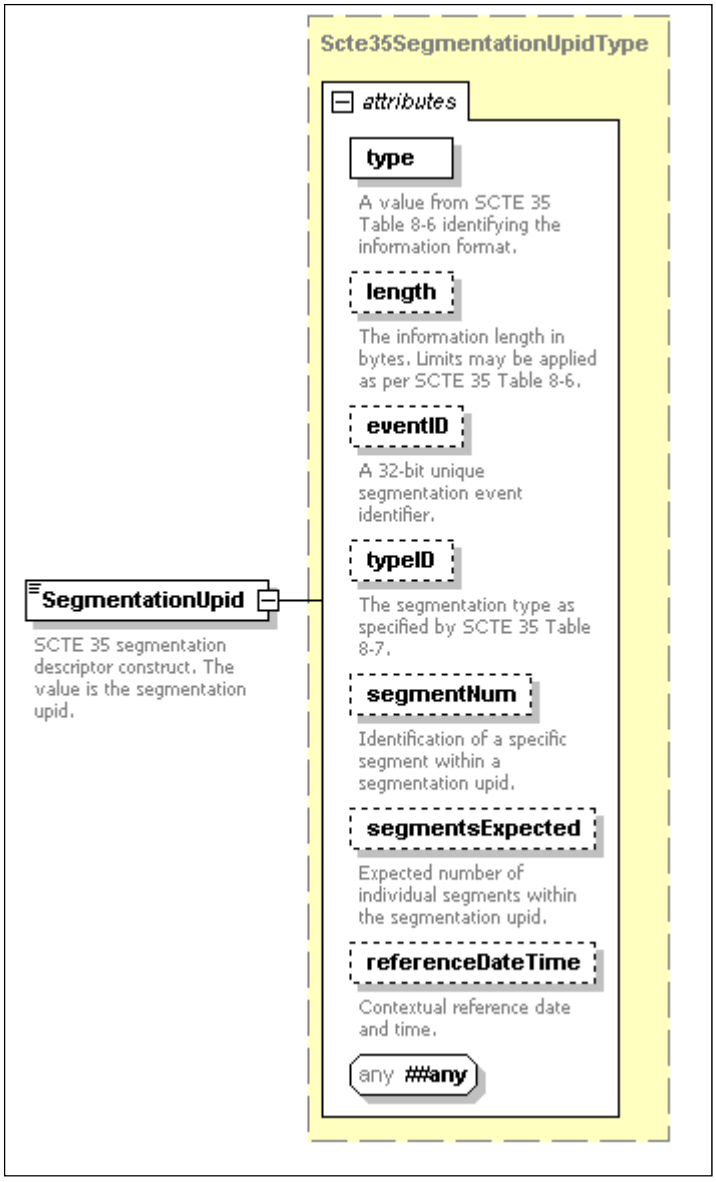

Figure 50 – SegmentationUpid element schema

9.17.1.1 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 [SCTE 35] 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 [SCTE 35] for additional information.

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

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

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

@referenceDateTime [Optional, core:dateTimeTimezoneType] – The date and time providing contextual reference. See clause 9.1.1 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 [SCTE 35] for additional information.

9.17.2 SegmentationUpid element examples

```
  <SegmentationUpid type="1">89</SegmentationUpid>
  <SegmentationUpid type="6" length="12" 46ventide="8" typeID="32" segmentNum="2"
segmentsExpected="4">188166C7342065419F3A0245</SegmentationUpid><!—96-bit ISO 15706-2→
```

9.18 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 [SCTE 118-3] for additional information.

9.18.1 SpotRef element schema

Figure 51 illustrates the SpotRef element's schema.

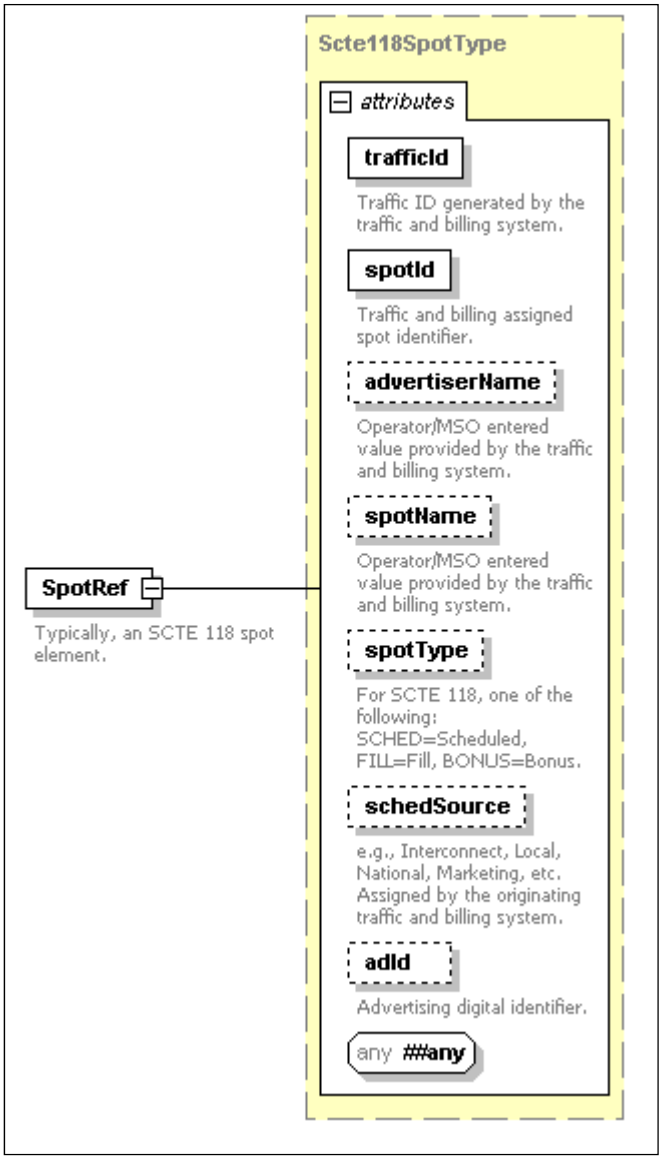

Figure 51 – SpotRef element schema

9.18.1.1 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 [SCTE 118-3] for additional information.

@spotId [Required, nonEmptyStringType] – A non-empty string generated by the traffic and billing system for spot identification. See [SCTE 118-3] 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 [SCTE 118-3] 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 [SCTE 118-3] 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 [SCTE 118-3] for additional information.

@schedSource [Optional, nonEmptyStringType] – A non-empty string identifying the scheduling source as assigned by the originating traffic and billing system. See [SCTE 118-3] for additional information.

@adId [Optional, nonEmptyStringType] – A non-empty string supplying the advertising digital identifier. See [SCTE 118-3] for additional information.

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

The SpotRef element's value shall be empty.

9.18.2 SpotRef element examples

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

9.19 StatusCode element

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

9.19.1 StatusCode element schema

Figure 52 illustrates the StatusCode element's schema.

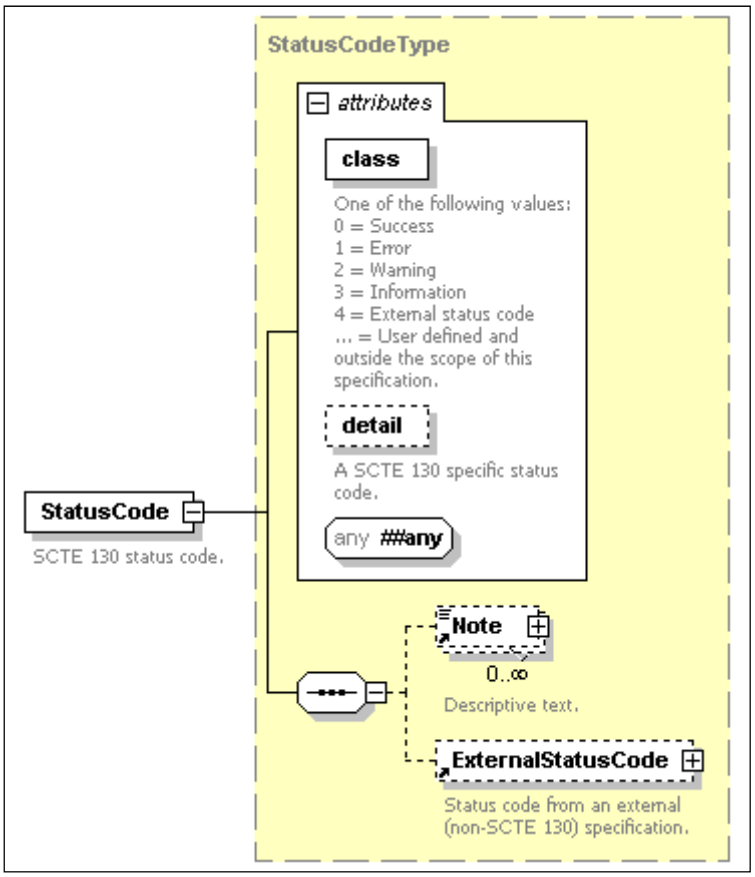

Figure 52 – StatusCode element schema

9.19.1.1 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

Recommendation. 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.

Table 5 – StatusCode element @class attribute values

@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
…	User defined and outside the scope of this Recommendation.

@detail [Optional, xsd:nonNegativeInteger] – The applicable detail status code value from Table A.1.

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

Note [Optional] – Zero or more Note elements where each Note element contains descriptive text. See clause 9.15 for additional information regarding the Note element.

ExternalStatusCode [Optional] – A container for a status code from an external source. See clause 9.13 for additional information.

9.19.2 StatusCode 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>
```

9.20 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 Recommendation. (For example, see

[ITU-T J.380.3].) 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. [ITU-T J.380.3] 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 [ITU-T J.380.3]. The Tracking element is not limited to this usage but rather this example describes one possible usage.

9.20.1 Tracking element schema

Figure 53 illustrates the Tracking element's schema.

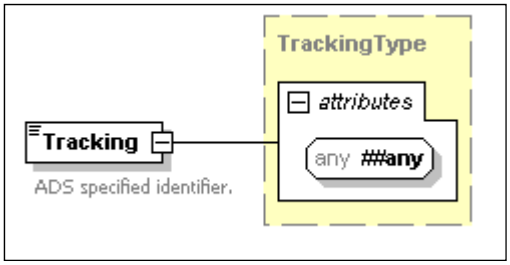

Figure 53 – Tracking Element Schema

9.20.1.1 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).

9.20.2 Tracking element examples

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

9.21 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 [IETF RFC 3986] for additional information.

9.21.1 URI element schema

Figure 54 illustrates the URI element's schema.

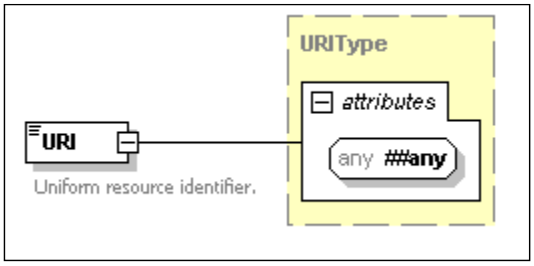

Figure 54 – URI element schema

9.21.1.1 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 [IETF RFC 3986] for additional information.

9.21.2 URI element examples

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

Annex A

StatusCode element @detail attribute values

(This annex forms an integral part of this Recommendation.)

Table A.1 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 clause 9.13 for additional information.

Table A.1 – StatusCode element @detail attribute values

@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 ITU-T J.380.2				

Table A.1 – StatusCode element @detail attribute values

@detail Value	Description	SCR	SSN	SSA	Comment
3000 to 3999	Reserved for ITU-T J.380.3				ITU-T J.380.3 specific errors.
4000 to 4999	Reserved for ITU-T J.380.4				ITU-T J.380.4 specific errors.
5000 to 5999	Reserved for ITU-T J.380.5				ITU-T J.380.5 specific errors.
6000 to 6999	Reserved for ITU-T J.380.6				ITU-T J.380.6 specific errors.
7000 to 7999	Reserved for ITU-T J.380.7				ITU-T J.380.7 specific errors.

Bibliography

[b-ITU-T J.380.1] ITU-T J.380.1 (2011), *Digital Program Insertion – Advertising Systems Interfaces – Advertising systems overview*.

[b-IETF RFC 4122] IETF RFC 4122 (2005), A *Universally Unique IDentifier (UUID) URN Namespace*.

[b-SCTE 67] SCTE 67-2006, *Digital Program Insertion Cueing Message for Cable – Interpretation for SCTE 35*.

[b-SCTE 118-1] ANSI/SCTE 118-1-2006, *Program-Specific Ad Insertion – Data Field Definitions, Functional Overview and Application Guidelines*.

[b-SCTE 118-2] ANSI/SCTE 118-2-2007, *Program-Specific Ad Insertion – Content Provider to Traffic Communication Applications Data Model*.

[b-XMLSchemaP0] W3C Recommendation XML Schema Part 0 (2004), *XML Schema Part 0: Primer Second Edition*.
<http://www.w3.org/TR/xmlschema-0/>

[b-CLAD I1-1] CableLabs® (2006), *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>

[b-CLAD I2-0] CableLabs® (2007), *ADI 2.0 Specification Asset Structure: MD-SP-ADI2.0-AS-I03-070105*.
<http://www.cablelabs.com/specifications/MD-SP-ADI2.0-AS-I03-070105.pdf>

[b-CLADVS 2-0] CableLabs® (2007), *Advertising Distribution 2.0 Specification: MD-SP-ADVS2.0-I01-070105*.
<http://www.cablelabs.com/specifications/MD-SP-ADVS2.0-I01-070105.pdf>

[b-CLAIM 2-0] CableLabs® (2006), ADI 2.0 Specification Asset Inventory Messages: MD-SP-ADI2.0-AIM-I02-060505.
<http://www.cablelabs.com/specifications/MD-SP-ADI2.0-AIM-I02-060505.pdf>

[b-CLVOD 1-1] CableLabs® (2006), *Video-on-Demand Content Specification Version 1.1: MD-SP-VOD-CONTENT1.1-I05-060831*.
<http://www.cablelabs.com/specifications/MD-SP-VOD-CONTENT1.1-I05-060831.pdf>

[b-CLVOD 2-0] CableLabs® (2007), *Video-On-Demand Content Specification Version 2.0: MD-SP-VOD-CONTENT2.0-I02-070105*.
<http://www.cablelabs.com/specifications/MD-SP-VOD-CONTENT2.0-I02-070105.pdf>

SERIES OF ITU-T RECOMMENDATIONS

Series A	Organization of the work of ITU-T
Series D	General tariff principles
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media, digital systems and networks
Series H	Audiovisual and multimedia systems
Series I	Integrated services digital network
Series J	**Cable networks and transmission of television, sound programme and other multimedia signals**
Series K	Protection against interference
Series L	Construction, installation and protection of cables and other elements of outside plant
Series M	Telecommunication management, including TMN and network maintenance
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Terminals and subjective and objective assessment methods
Series Q	Switching and signalling
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	Terminals for telematic services
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	Data networks, open system communications and security
Series Y	Global information infrastructure, Internet protocol aspects and next-generation networks
Series Z	Languages and general software aspects for telecommunication systems

Printed in Switzerland
Geneva, 2012