



***Society of Cable
Telecommunications
Engineers***

**ENGINEERING COMMITTEE
Digital Video Subcommittee**

AMERICAN NATIONAL STANDARD

ANSI/SCTE 130-4 2009

Digital Program Insertion–Advertising Systems Interfaces

Part 4

Content Information Service (CIS)

NOTICE

The Society of Cable Telecommunications Engineers (SCTE) Standards are intended to serve the public interest by providing specifications, test methods and procedures that promote uniformity of product, interchangeability and ultimately the long term reliability of broadband communications facilities. These documents shall not in any way preclude any member or non-member of SCTE from manufacturing or selling products not conforming to such documents, nor shall the existence of such standards preclude their voluntary use by those other than SCTE members, whether used domestically or internationally.

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

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

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

All Rights Reserved

© Society of Cable Telecommunications Engineers, Inc. 2009
140 Philips Road
Exton, PA 19341

TABLE OF CONTENTS

1.0 SCOPE	1
2.0 REFERENCES	1
2.1 NORMATIVE REFERENCES	1
2.1.1 Normative Reference Acquisition	1
2.2 INFORMATIVE REFERENCES	2
2.2.1 Informative Reference Acquisition.....	2
3.0 COMPLIANCE NOTATION	2
4.0 DEFINITIONS AND ACRONYMS	3
5.0 ABBREVIATIONS	3
6.0 CONTENT INFORMATION SERVICE OVERVIEW (INFORMATIVE)	3
7.0 NOTATIONAL CONVENTIONS	5
7.1 NORMATIVE XML SCHEMA	5
7.2 DOCUMENT CONVENTIONS	5
8.0 PROCESSING CONVENTIONS	5
8.1 UNKNOWN/UNRECOGNIZED/UNSUPPORTED XML ELEMENTS AND ATTRIBUTES	5
9.0 XML NAMESPACES	5
10.0 CIS MESSAGES	7
10.1 @VERSION ATTRIBUTE.....	8
10.2 REQUEST BASE MESSAGE	8
10.3 RESPONSE BASE MESSAGE	8
10.4 NOTIFICATION BASE MESSAGE	8
10.5 ACKNOWLEDGEMENT BASE MESSAGE	8
10.6 MESSAGES REQUIRING CONTENT NOTIFICATION REGISTRATION	8
10.7 DEFAULT LOGICAL SERVICE CHANNEL ENDPOINT	8
10.8 CIS MESSAGE EXCHANGE.....	9
10.9 LIST SUPPORTED FEATURES REQUEST AND RESPONSE.....	9
10.9.1 ListSupportedFeaturesRequest Message	10
10.9.2 ListSupportedFeaturesResponse Message.....	11
10.10 LIST CONTENT NOTIFICATION REGISTRATION REQUEST AND RESPONSE	14
10.10.1 ListContentNotificationRegistrationRequest Message.....	14
10.10.2 ListContentNotificationRegistrationResponse Message	16
10.11 CONTENT NOTIFICATION REGISTRATION REQUEST AND RESPONSE	18
10.11.1 ContentNotificationRegistrationRequest Message.....	19
10.11.2 ContentNotificationRegistrationResponse Message	22
10.12 CONTENT NOTIFICATION AND ACKNOWLEDGEMENT	24
10.12.1 ContentNotification Message.....	24
10.12.2 ContentNotificationAcknowledgement Message.....	26
10.13 CREATE CURSOR REQUEST AND RESPONSE.....	28
10.13.1 CreateCursorRequest Message.....	28
10.13.2 CreateCursorResponse Message	30
10.14 CANCEL CURSOR REQUEST AND RESPONSE	32
10.14.1 CancelCursorRequest Message	32

10.14.2	<i>CancelCursorResponse Message</i>	34
10.15	CONTENT QUERY REQUEST AND RESPONSE.....	36
10.15.1	<i>ContentQueryRequest Message</i>	36
10.15.2	<i>ContentQueryResponse Message</i>	38
10.16	CONTENT NOTIFICATION DEREGISTER REQUEST AND RESPONSE.....	40
10.16.1	<i>ContentNotificationDeregisterRequest Message</i>	40
10.16.2	<i>ContentNotificationDeregisterResponse Message</i>	42
10.17	DEREGISTRATION NOTIFICATION AND ACKNOWLEDGEMENT.....	44
10.17.1	<i>DeregistrationNotification Message</i>	44
10.17.2	<i>DeregistrationAcknowledgement Message</i>	46
10.18	SERVICE CHECK SUPPORT.....	48
10.19	SERVICE STATUS SUPPORT.....	48
11.0	CIS ELEMENT DETAILS.....	49
11.1	BASIC AND ADVANCED QUERIES.....	49
11.2	CONTENTNOTIFICATIONSELECTOR.....	50
11.3	CONTENTQUERY.....	51
11.4	CURSOR.....	52
11.5	CONTENTQUERYRESULT.....	53
11.6	QUERYFILTER.....	54
11.7	FILTERELEMENT.....	55
11.8	BASICQUERYRESULTLIST.....	56
11.9	ADVANCEDFILTERELEMENT.....	57
11.10	ADVANCEDQUERYRESULTLIST.....	58
11.11	ADVANCEDQUERYRESULT.....	58
11.12	DATAMODELLIST.....	60
11.13	ADVANCEDQUERYLANGUAGELIST.....	60
11.14	ADVANCEDQUERYLANGUAGE.....	61
12.0	SCTE 130 PART 4 CIS ATTRIBUTE TYPES.....	61
12.1	SEMANTIC DEFINITIONS FOR SCTE 130 PART 4 CIS ATTRIBUTE TYPES.....	61
12.1.1	<i>queryIdAttrType Attribute Type</i>	61
12.1.2	<i>queryIdRefAttrType Attribute Type</i>	61
12.1.3	<i>cursorIdAttrType Attribute Type</i>	62
12.1.4	<i>cursorIdRefAttrType Attribute Type</i>	62
12.1.5	<i>queryFilterOpTypeEnumeration Attribute Type</i>	62
12.1.6	<i>filterElementNameType Attribute Type</i>	62
12.1.7	<i>filterElementValueType Attribute Type</i>	62
12.1.8	<i>queryLanguageAttrType Attribute Type</i>	63
12.1.9	<i>expandOutputAttrType Attribute Type</i>	63
13.0	BASIC QUERIES AND REGULAR EXPRESSIONS.....	64
13.1	REGULAR EXPRESSIONS AND WILDCARDS.....	64
A.	APPENDIX A (NORMATIVE) STATUSCODE ELEMENT @DETAIL ATTRIBUTE VALUES.....	67
B.	APPENDIX B (INFORMATIVE) COMPLEX QUERIES AND EXPANDED OUTPUT.....	68
B.1	MULTIPLE FILTER ELEMENTS.....	68
B.2	EXPANDED OUTPUT.....	70
C.	APPENDIX C (INFORMATIVE) ADVANCED QUERIES.....	73
C.1	ADVANCED QUERIES.....	73
D.	APPENDIX D (INFORMATIVE) CURSORS.....	74
D.1	CREATING CURSORS.....	74

D.2	WALKING CURSORS	75
D.3	CANCELING EXISTING CURSORS	76
E.	APPENDIX E (INFORMATIVE) COMPLETE MESSAGE EXAMPLES	78
E.1	LIST SUPPORTED FEATURES REQUEST AND RESPONSE	78
E.2	CONTENT QUERY REQUEST AND RESPONSE	80
E.3	CONTENT NOTIFICATION REGISTRATION REQUEST	81
E.4	CONTENT NOTIFICATION	82
F.	APPENDIX F (NORMATIVE) WSDL.....	84
	<i>F.1 WSDL TARGET NAMESPACE URI FORMAT</i>	<i>84</i>
	<i>F.2 WSDL DESCRIPTION</i>	<i>85</i>

LIST OF FIGURES

Figure 1 – CIS System Overview	4
Figure 2 - CIS Message Exchange	9
Figure 3 - ListSupportedFeaturesRequest–XML Schema	10
Figure 4 - ListSupportedFeaturesResponse–XML Schema	12
Figure 5 - ListContentNotificationRegistrationRequest XML Schema	15
Figure 6 - ListContentNotificationRegistrationResponse XML Schema	17
Figure 7 - ContentNotificationRegistrationRequest–XML Schema	20
Figure 8 - ContentNotificationRegistrationResponse	23
Figure 9 - ContentNotification–XML Schema	25
Figure 10 - ContentNotificationAcknowledgement XML Schema	27
Figure 11 - CreateCursorRequest XML Schema	29
Figure 12 - CreateCursorResponse XML Schema	31
Figure 13 - CancelCursorRequest - XML Schema	33
Figure 14 - CancelCursorResponse - XML Schema	35
Figure 15 - ContentQueryRequest–XML Schema	37
Figure 16 - ContentQueryResponse–XML Schema	39
Figure 17 - ContentNotificationDeregisterRequest	41
Figure 18 - ContentNotificationDeregisterResponse XML Schema	43
Figure 19 - DeregistrationNotification–XML Schema	45
Figure 20 - DeregistrationAcknowledgement–XML Schema	47
Figure 21 - ContentNotificationSelector XML Schema	50
Figure 22 - ContentQuery–XML Schema	51
Figure 23 - Cursor–XML Schema	52
Figure 24 - ContentQueryResult–XML Schema	53
Figure 25 - QueryFilter–XML Schema	54
Figure 26 - FilterElement–XML Schema	56

Figure 27 - BasicQueryResultList–XML Schema	56
Figure 28 - AdvancedFilterElement–XML Schema	57
Figure 29 - AdvancedQueryResultList–XML Schema	58
Figure 30 - AdvancedQueryResult - XML Schema	59
Figure 31 - DataModelList – XML Schema	60
Figure 32 - AdvancedQueryLanguageList – XML Schema	60
Figure 33 - AdvancedQueryLanguage XML Schema	61

LIST OF TABLES

Table 1: XML Namespace Declarations	6
Table 2: CIS Top Level Messages	7
Table 3: ListSupportedFeatures core:Callout @message values	13
Table 4: ContentNotificationRegistrationResponse core:Callout @message values	21
Table 5: ContentNotification@Type values	24
Table 6: CIS Elementary Message Details	49
Table 7: Advanced Query Languages	49
Table 8: BasicQueryResultList Child Elements	57
Table 9: AdvancedQueryResult CDATA contents	59
Table 10: QueryFilterOpTypeEnumeration Values	62
Table 11: CIS Regular Expressions	65
Table 12: Regular Expression Examples	66
Table 13: StatusCode Details	67
Table 14: Part 2 and Part 4 StatusCode Detail Usage	68

Digital Program Insertion—Advertising Systems Interfaces Part 4—Content Information Service

1.0 SCOPE

This document, SCTE 130 Part 4, describes the Digital Program Insertion Advertising Systems Interfaces' CIS (Content Information Service) messaging and data type specification using XML, XML Namespaces, and XML Schema.

2.0 REFERENCES

2.1 Normative References

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

[SCTE130-2]	SCTE 130-2 2008a: Digital Program Insertion—Advertising Systems Interfaces Part 2 - Core Data Elements
[SCTE130-7]	SCTE 130-7 2009: Digital Program Insertion – Advertising Systems Interfaces Part 7 – Message Transport.
[SCTE118-3]	ANSI/SCTE 118-3 2006 – Program-Specific Ad Insertion – Traffic System to Ad Insertion System File Format Specification
[W3C - XPath]	XPath–W3C REC: XML Path Language (XPath) Version 1.0. November 16, 1999
[W3C – Xquery]	XQuery–W3C REF: An XML Query Language (XQuery) Version 1.0. January 23, 2007

All normative references found in [SCTE130-2] are included and apply to this document. See [SCTE130-2] for additional information.

2.1.1 Normative Reference Acquisition

2.1.1.1 SCTE Standards: United States of America

Society of Cable Telecommunications Engineers Inc., 140 Philips Road, Exton, PA 19341; Telephone 800-542-5040; Facsimile: 610-363-5898; E-mail: standards@scte.org; URL: <http://www.scte.org>.

2.1.1.2 W3C Standards

MIT, 32 Vassar Street, Room 32-G515, Cambridge, MA 02139, USA;
Telephone: +1.617.258.5999; <http://www.w3.org>.

2.2 Informative References

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

[SCTE130-1]	SCTE 130-1 2008: Digital Program Insertion—Advertising Systems Interfaces Part 1—Overview
-------------	---

2.2.1 Informative Reference Acquisition

2.2.1.1 SCTE Standards: United States of America

See Section 2.1.1.1.

3.0 COMPLIANCE NOTATION

“SHALL”	This word or the adjective “REQUIRED” means that the item is an absolute requirement of this specification.
“SHALL NOT”	This phrase means that the item is an absolute prohibition of this specification.
“SHOULD”	This word or the adjective “RECOMMENDED” means that there may exist valid reasons in particular circumstances to ignore this item, but the full implications should be understood and the case carefully weighed before choosing a different course.
“SHOULD NOT”	This phrase means that there may exist valid reasons in particular circumstances when the listed behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.
“MAY”	This word or the adjective “OPTIONAL” means that this item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because it enhances the product, for example; another vendor may omit the same item.

4.0 DEFINITIONS AND ACRONYMS

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

All [SCTE130-2] definitions are included herein. See [SCTE130-2] for additional information

Cursor: A temporary CIS construct containing static data. CIS clients may create and access cursor information using the standard query mechanisms described in this document.

5.0 ABBREVIATIONS

All [SCTE130-2] abbreviations are included herein. See [SCTE130-2] for additional information.

CDATA: (Character **DATA**) XML data that is not parsed. CDATA carries markup examples that would otherwise be interpreted as XML because of the tags.

6.0 CONTENT INFORMATION SERVICE OVERVIEW (INFORMATIVE)

A CIS provides asset metadata query and notification services, including media availability if known, to service consumers. Using the interfaces defined in this specification, service consumers may retrieve detailed information about assets referenced by a CIS.

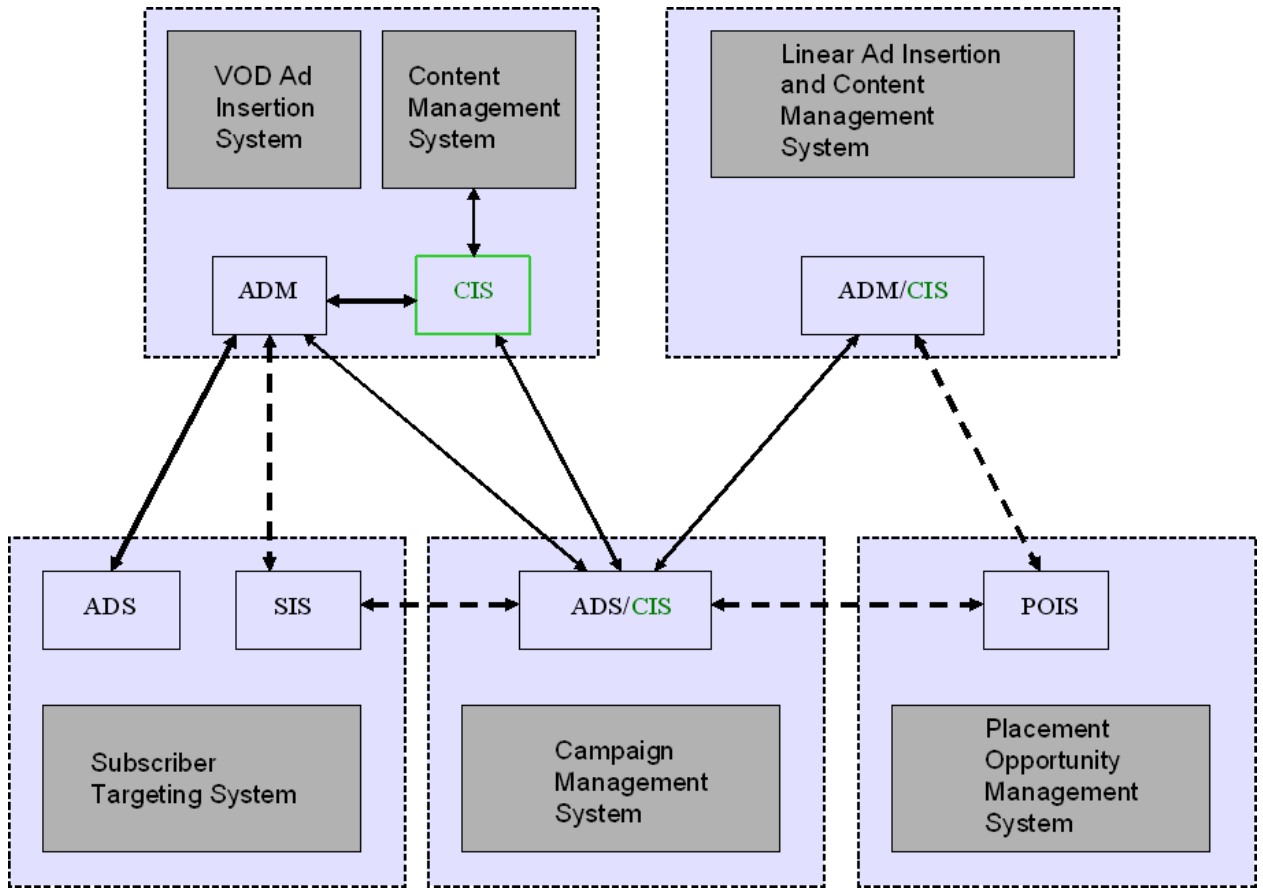


Figure 1 – CIS System Overview

Figure 1 provides examples of several possible CIS implementations with respect to the SCTE 130 logical services. Locations for CIS logical services are not limited to physical content stores, but may also be implemented on top of any physical instances containing asset information.

A CIS answers queries concerning its associated physical content store(s) and issues notification messages when physical content store changes are detected. The interface between the CIS and the physical content store is outside the scope of this specification.

The existence of a CIS implementation, either as a standalone system or as a logical service coexisting with other services or systems, does not imply physical presence of referenced assets. Asset availability may be signaled by the CIS through specific attribute values in the query and content notification response messages. See Sections 11.8 and 11.11 for additional details on asset availability indication.

This specification describes the message sets used to form basic and/or advanced queries against CIS fronted asset metadata. Basic queries leverage a limited key/value regular expression grammar while advanced queries support industry standard XML parsing languages and other user defined query languages. See section 11.14 for additional

information on advanced query language support and section 11.1 for additional information on Basic and Advanced query mechanisms.

This specification also describes cursor based operations for both basic and advanced queries. Consumers may request the creation of temporary static, cursor based information with specified lifetimes, and then iterate over the information until the cursor life cycle completes (i.e., expires). See section 11.4 for additional information on cursors.

This specification includes a notification model for CIS to consumer messaging. The model includes associated notification management functions such as registration, deregistration and active registration listing.

Part 4 also facilitates the expression of CIS supported data models(s). The examples provided herein assume that the CIS supports a data model similar to the CableLabs Asset Distribution Interface Specification (ADI), Version 1.1. See the [CLADI1-1] normative reference from [SCTE130-2].

7.0 NOTATIONAL CONVENTIONS

7.1 Normative XML Schema

See [SCTE130-2] for information.

7.2 Document Conventions

This specification utilizes the same document conventions as SCTE 130 Part 2. See [SCTE130-2] for conventions and XML schema illustration nomenclature explanations.

8.0 PROCESSING CONVENTIONS

8.1 Unknown/Unrecognized/Unsupported XML Elements and Attributes

See [SCTE130-2] for information.

9.0 XML NAMESPACES

This specification uses the ‘cis’ prefix, as described in Table 1, for the interface associated with the specific XML namespace URI that shall be used by all implementations. Table 1 lists the prefix, the corresponding namespace, and a description of the defining specification used herein.

Prefix	Namespace	Description
core	http://www.scte.org/schemas/130-2/2008a/core	See [SCTE130-2]
cis	http://www.scte.org/schemas/130-4/2008a/cis	SCTE 130 Part 4

xsd	http://www.w3.org/2001/XMLSchema	See [XMLSchemaP1] and [XMLSchemaP2] from [SCTE130-2]
-----	---	--

Table 1: XML Namespace Declarations

Unless otherwise stated, all references to XML elements illustrated in this document are from the 'cis' namespace. Elements from other namespaces will be prefixed with the name of the external namespace, e.g. <core:XXXX>.

10.0 CIS MESSAGES

The following topics are covered by [SCTE130-2] and this specification considers all aspects defined therein to be normative and applicable herein. See [SCTE130-2] for additional information.

- Message format
- XML message carriage
- Transport mechanisms
- Message error handling

The CIS message interface shall include the messages defined in [SCTE130-2]. Table 2 identifies additional SCTE 130 Part 4 CIS specific messages.

Message	Description
ListSupportedFeaturesRequest	Request to retrieve a list of CIS supported features
ListSupportedFeaturesResponse	Response to ListSupportedFeaturesRequest
ListContentNotificationRegistrationRequest	Request to list existing registrations
ListContentNotificationRegistrationResponse	Response to ListContentNotificationRegistrationRequest
ContentNotificationRegistrationRequest	Registration request for content notification
ContentNotificationRegistrationResponse	Response to ContentNotificationRegistrationRequest
ContentNotification	Notification message indicating asset status change
ContentNotificationAcknowledgement	Response to ContentNotification
CreateCursorRequest	Request to create a cursor
CreateCursorResponse	Response to CreateCursorRequest
CancelCursorRequest	Request to cancel an existing cursor
CancelCursorResponse	Response to CancelCursorRequest
ContentQueryRequest	Request to acquire records from the CIS
ContentQueryResponse	Response to ContentQueryRequest
ContentNotificationDeregisterRequest	Request to de-register a previously accepted registration
ContentNotificationDeregisterResponse	Response to ContentNotificationDeregisterRequest
DeregistrationNotification	Deregistration notification
DeregistrationAcknowledgement	Deregistration notification acknowledgement

Table 2: CIS Top Level Messages

10.1 @version Attribute

For all SCTE 130 Part 4 messages specified in Table 2, the [SCTE130-2] @version attribute shall be set to the value “1.1” for this document’s revision.

10.2 Request Base Message

All CIS top level *request* messages are derived from the core:Msg_RequestBaseType abstract base message type. See the [SCTE130-2] document for details on the attributes and elements contained in this base message.

10.3 Response Base Message

All CIS top level *response* messages are derived from the core:Msg_ResponseBaseType abstract base message type. See the [SCTE130-2] document for details on the attributes and elements contained in this base message.

10.4 Notification Base Message

All CIS top level *notification* messages are derived from the core:Msg_NotificationBaseType abstract base message type. See the [SCTE130-2] document for details on the attributes and elements contained in this base message.

10.5 Acknowledgement Base Message

All CIS top level *acknowledgement* messages are derived from the core:Msg_AcknowledgementBaseType abstract base message type. See the [SCTE130-2] document for details on the attributes and elements contained in this base message.

10.6 Messages Requiring Content Notification Registration

Registration is required for ContentNotification, ContentNotificationDeregisterRequest and DeregistrationNotification message exchanges. All other CIS messages do not require registration and may be sent at any time.

10.7 Default Logical Service Channel Endpoint

At a minimum, the CIS shall support the receipt of the ListSupportedFeaturesRequest message on the default (well known) logical service channel endpoint address for the CIS. Other CIS messages may also be offered on the default (well known) logical service channel endpoint or may be optionally offered on additional logical service channel endpoints. In order to discover all CIS provided logical service channel endpoints, clients must complete a ListSupportedFeatures transaction with the CIS.

See Section 10.9.2 for additional information on the ListSupportedFeaturesResponse message.

See [SCTE130-2] for a complete description of the terms (message, endpoint and logical service channel).

10.8 CIS Message Exchange

The following diagram illustrates a typical message exchange between a CIS client and the CIS.

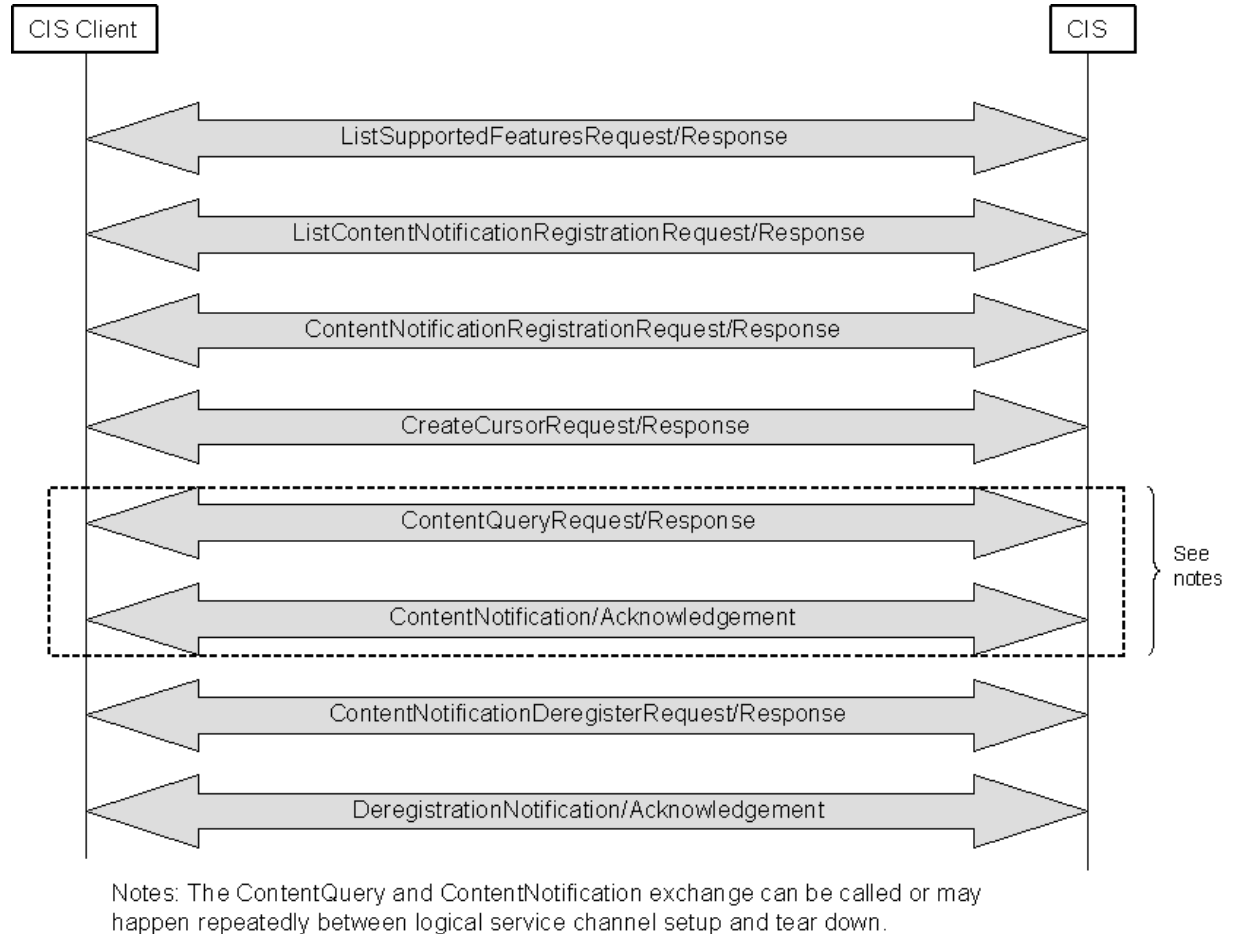


Figure 2 - CIS Message Exchange

Figure 2 illustrates all of the message exchanges that are specific to the CIS. The ServiceCheck and ServiceStatus message exchanges are not depicted in this illustration.

10.9 List Supported Features Request and Response

The ListSupportedFeatures request and response messages allow clients to inquire about CIS supported data models and advanced query languages.

10.9.1 ListSupportedFeaturesRequest Message

The ListSupportedFeatures request message allows a CIS client to inquire about the data models and advanced query languages supported by a CIS. Advanced query language support is optional and thus, the response message may or may not contain information regarding advanced query languages.

The XML schema definition for this message is illustrated in Figure 3 - ListSupportedFeaturesRequest-XML Schema.

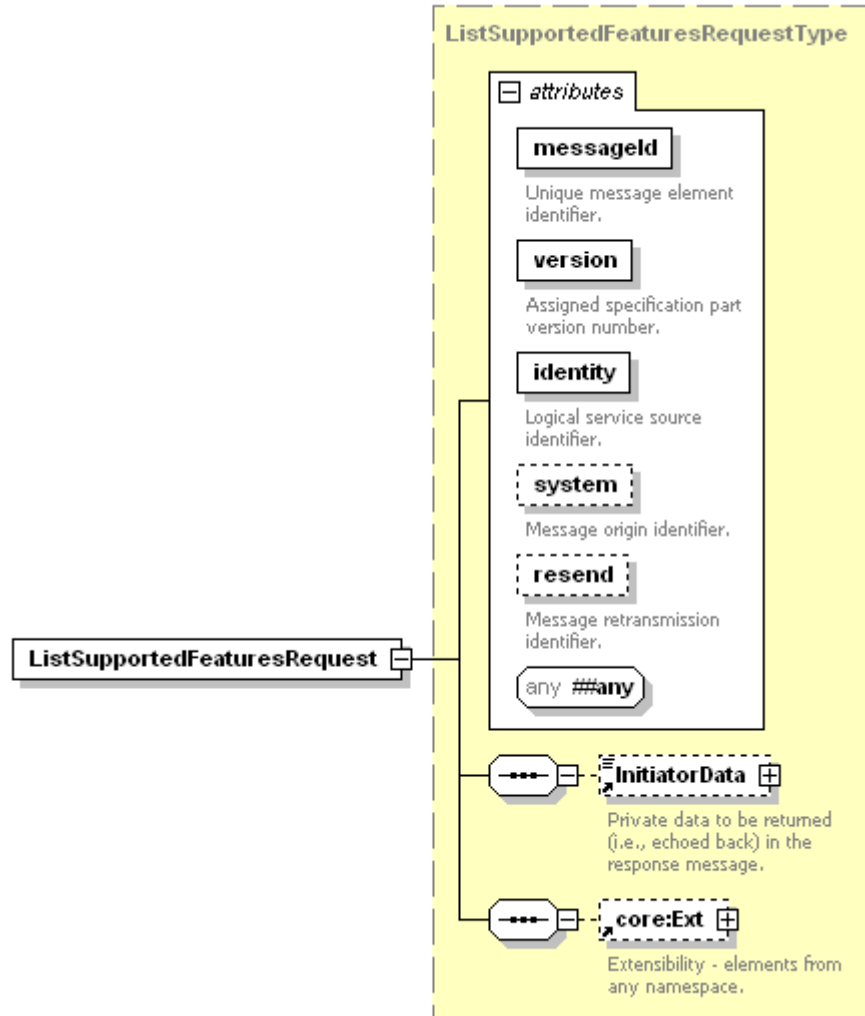


Figure 3 - ListSupportedFeaturesRequest-XML Schema

The ListSupportedFeaturesRequest message defines the following attributes and elements.

@messageId [Required, core:messageIdAttrType] - The message identifier. See [SCTE130-2] for additional information.

@version [Required, core:versionAttrType] - The message specification version. See [SCTE130-2] for additional information.

@identity [Required, core:identityAttrType] - The origin logical service identifier. See [SCTE130-2] for additional information.

@system [Optional, core:systemAttrType] - The message source identifier. See [SCTE130-2] for additional information.

@resend [Optional, core:resendAttrType] – Message retransmission identifier. See [SCTE130-2] for additional information.

@##any [Optional] - Any additional attributes from any namespace.

core:InitiatorData [Optional] – Private data that shall be returned in the ListSupportedFeaturesResponse message. See [SCTE130-2] for additional details on the core:InitiatorData element.

core:Ext [Optional] – Any additional elements from other namespaces.

10.9.2 ListSupportedFeaturesResponse Message

A successful ListSupportedFeaturesResponse message shall contain, at a minimum, a single core:Callout element containing one or more core:Address element(s). See [SCTE130-2] for additional information on the core:Callout element.

The XML schema definition for this message is illustrated in Figure 4.

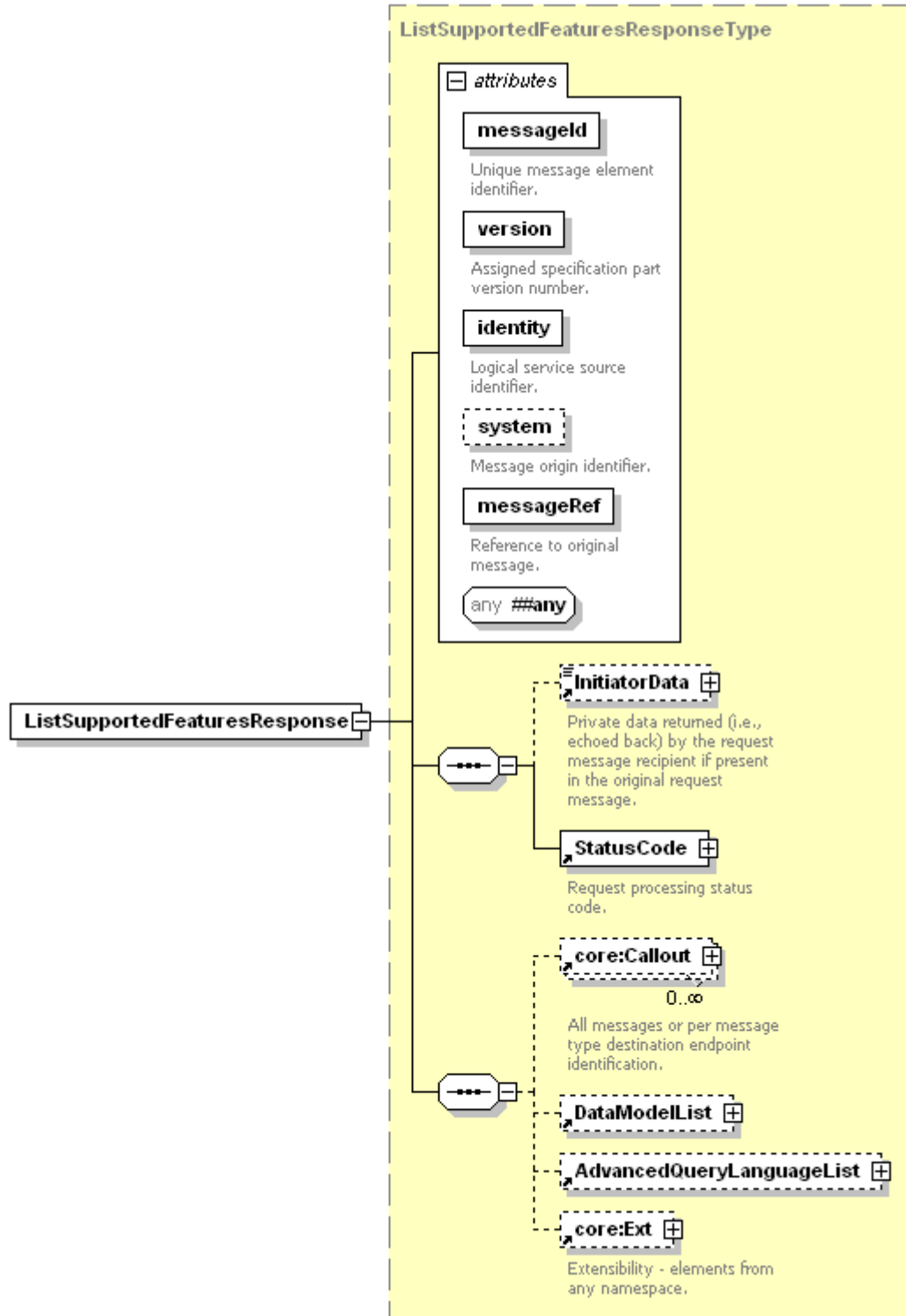


Figure 4 - ListSupportedFeaturesResponse–XML Schema

The ListSupportedFeaturesResponse message is derived from the core namespace base type core:Msg_ResponseBaseType and defines the following attributes and elements.

@messageId [Required, core:messageIdAttrType] - The message identifier. See [SCTE130-2] for additional information.

@version [Required, core:versionAttrType] - The message specification version. See [SCTE130-2] for additional information.

@identity [Required, core:identityAttrType] - The origin logical service identifier. See [SCTE130-2] for additional information.

@system [Optional, core:systemAttrType] - The message source identifier. See [SCTE130-2] for additional information.

@messageRef [Required, core:messageRefAttrType] - A reference to the ListSupportedFeaturesRequest message element initiating this message exchange. The value shall be the ListSupportedFeaturesRequest message's @messageId attribute value. See [SCTE130-2] for additional information on the core:messageRefAttrType.

@##any [Optional] – Any additional attributes.

core:InitiatorData [Optional] – Private data from the ListSupportedFeaturesRequest message. See [SCTE130-2] for additional details on the core:InitiatorData element.

core:StatusCode [Required] - An core:StatusCode element for communicating status information to the client. See [SCTE130-2] for additional information.

core:Callout [Optional] – Zero or more core:Callout elements specifying the service channel message endpoint(s). See [SCTE130-2] for additional information on core:Callout. Table 3 contains the values for the @message attribute of the core:Callout element. Values for the @message attribute should be used exactly as defined in this table.)

Value	Description
ContentNotificationRegistrationRequest	Destination endpoint for message.
ContentNotificationDeregisterRequest	Destination endpoint for message.
ListContentNotificationRegistrationRequest	Destination endpoint for message.
CreateCursorRequest	Destination endpoint for message.
CancelCursorRequest	Destination endpoint for message.
ContentQueryRequest	Destination endpoint for message.
ServiceStatusNotification	Destination endpoint for message.

Table 3: ListSupportedFeatures core:Callout @message values

DataModelList [Optional] – The DataModelList element contains a sequence of core:ContentDataModel elements indicating the data models supported by

this CIS. See the [SCTE130-2] documentation for a list of the supported data models. See Section 11.12 for additional information on the DataModelList element.

AdvancedQueryLanguageList [Optional] –The AdvancedQueryLanguageList element contains a list of the advanced query languages supported by the CIS. See Table 7 for a list of advanced query processing languages for CIS implementations that support advanced query language capabilities. See Section 11.13 for additional information on the AdvancedQueryLanguageList element.

core:Ext [Optional] –Any additional elements from other namespaces.

10.10 List Content Notification Registration Request and Response

A CIS client may inquire about current registrations by using the ListContentNotificationRegistrationRequest message. The CIS shall respond to the ListContentNotificationRegistrationRequest message with a ListContentNotificationRegistrationResponse message. This allows a CIS client to discover the active notification queries that were previously installed by one or more ContentNotificationRegistrationRequest messages.

10.10.1 ListContentNotificationRegistrationRequest Message

The ListContentNotificationRegistrationRequest message may be issued to a CIS to retrieve information about active ContentNotificationRegistrationRequest messages.

The XML schema definition for this message is illustrated in Figure 5.

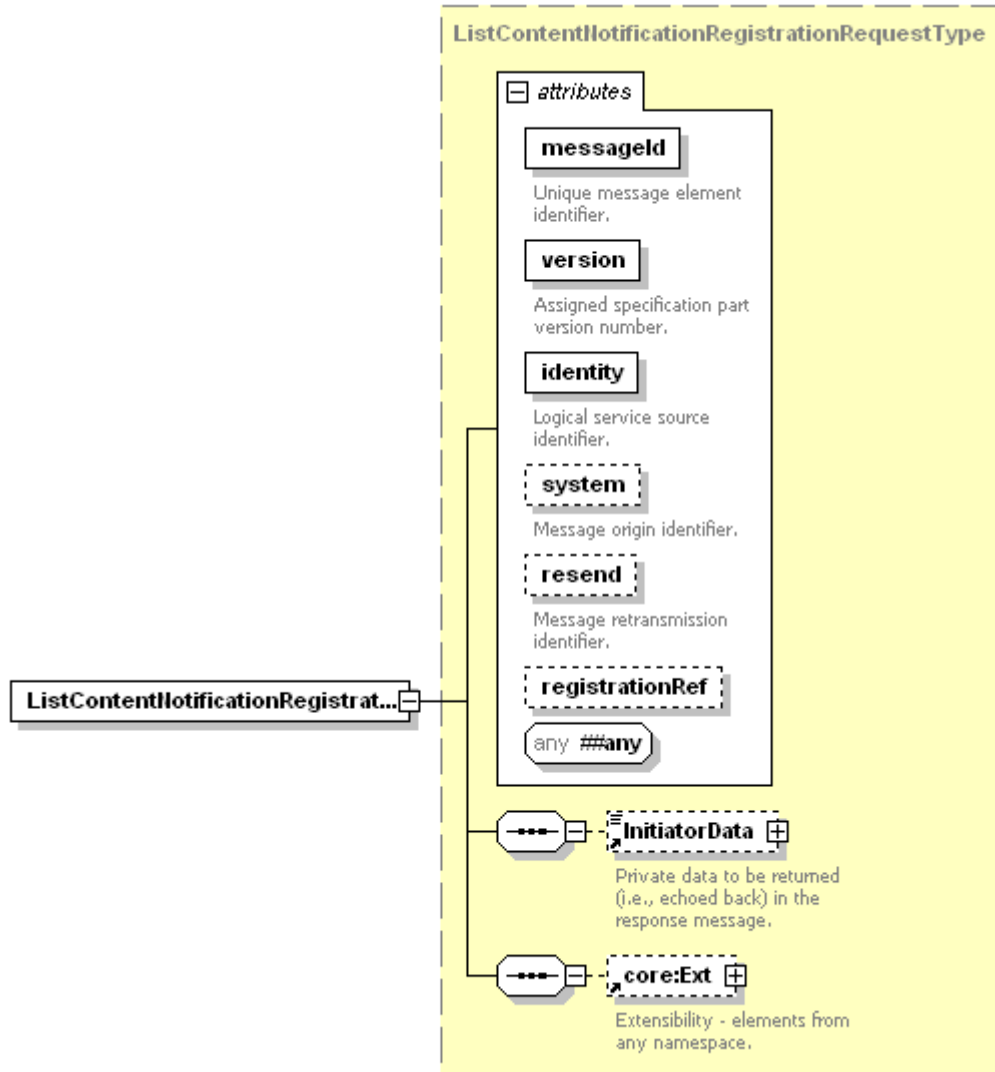


Figure 5 - ListContentNotificationRegistrationRequest XML Schema

The ListContentNotificationRegistrationRequest message is derived from the core namespace base type core:Msg_RequestBaseType and defines the following attributes and elements.

@messageId [Required, core:messageIdAttrType] - The message identifier. See [SCTE130-2] for additional information.

@version [Required, core:versionAttrType] - The message specification version. See [SCTE130-2] for additional information.

@identity [Required, core:identityAttrType] - The origin logical service identifier. See [SCTE130-2] for additional information.

@system [Optional, core:systemAttrType] - The message source identifier. See [SCTE130-2] for additional information.

@resend [Optional, core:resendAttrType] – Message retransmission identifier. See [SCTE130-2] for additional information.

@registrationRef [Optional, core:registrationRefAttrType] – A reference to an original ContentNotificationRegistrationRequest message. This attribute instructs the CIS to only list the ContentNotificationRegistrationRequest message identified by this reference. The value shall always be the ContentNotificationRegistrationRequest message's @messageId attribute value, which in a retransmitted registration message, shall be the @resend attribute's value.

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

core:InitiatorData [Optional] – Private data that shall be returned in the ListContentNotificationRegistrationResponse message. See [SCTE130-2] for additional details on the core:InitiatorData element.

core:Ext [Optional] – Any additional elements from other namespaces.

10.10.2 ListContentNotificationRegistrationResponse Message

The ListContentNotificationRegistrationResponse message is the response pair to the previously defined ListContentNotificationRegistrationRequest message.

The XML schema definition for this message is illustrated in Figure 6.

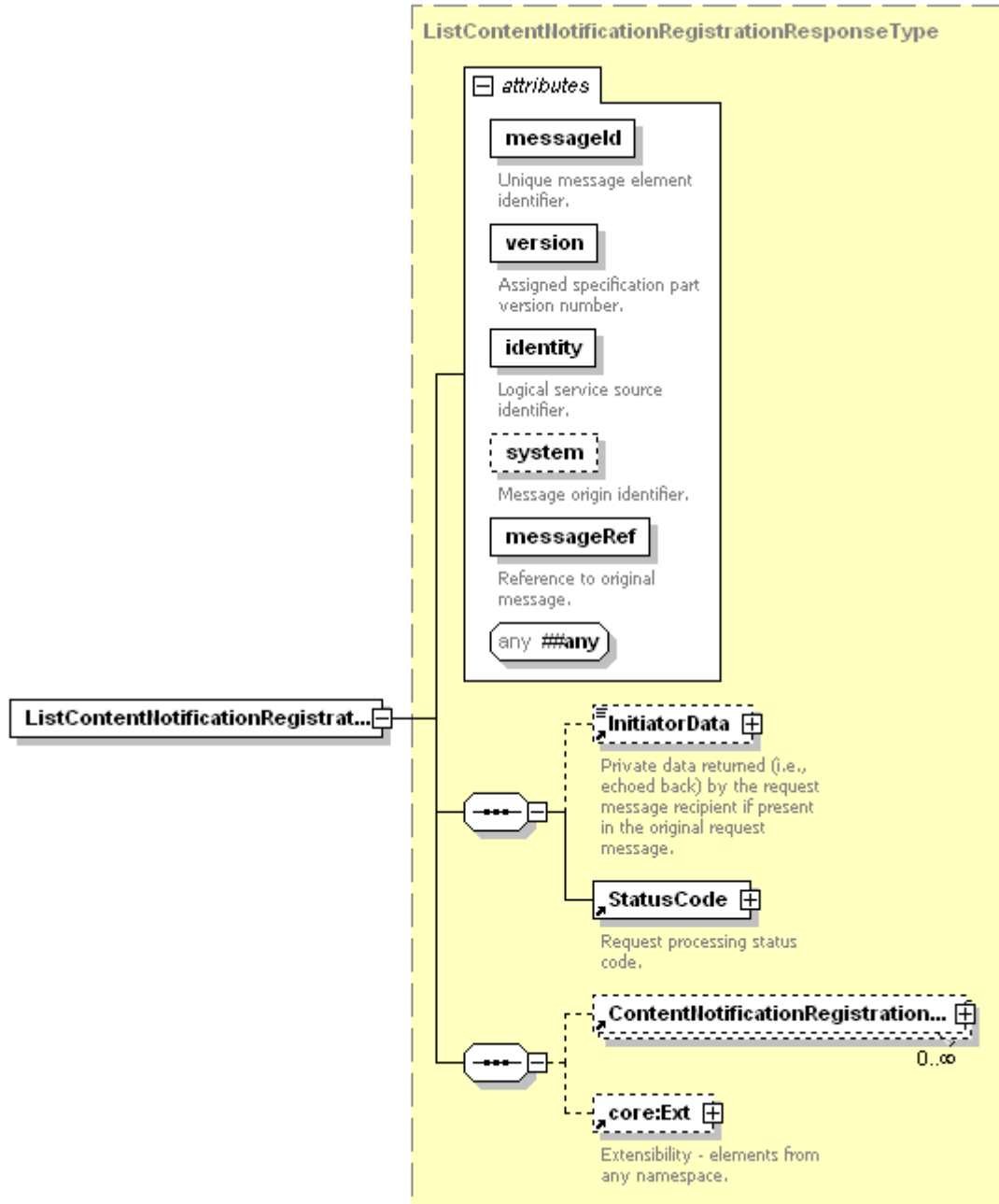


Figure 6 - ListContentNotificationRegistrationResponse XML Schema

The `ListContentNotificationRegistrationResponse` message is derived from the core namespace base type `core:Msg_ResponseBaseType` and defines the following attributes and elements.

@messageId [Required, core:messageIdAttrType] - The message identifier. See [SCTE130-2] for additional information.

@version [Required, core:versionAttrType] - The message specification version. See [SCTE130-2] for additional information.

@identity [Required, core:identityAttrType] - The origin logical service identifier. See [SCTE130-2] for additional information.

@system [Optional, core:systemAttrType] - The message source identifier. See [SCTE130-2] for additional information.

@messageRef [Required, core:messageRefAttrType] - A reference to the ListContentNotificationRegistrationRequest message element initiating this message exchange. The value shall be the ListContentNotificationRegistrationRequest message's @messageId attribute value. See [SCTE130-2] for additional information on the core:messageRefAttrType.

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

core:InitiatorData [Optional] – Private data from the ListContentNotificationRegistrationRequest message. See [SCTE130-2] for additional details on the core:InitiatorData element.

core:StatusCode [Required] – An core:StatusCode element for communicating status information to the client. See [SCTE130-2] for additional information.

ContentNotificationRegistrationRequest [Optional] – Zero or more ContentNotificationRegistrationRequest message elements. The ContentNotificationRegistrationRequest element shall be a recorded copy of the accepted registration message. The message element order does not convey any information (e.g., element order does not reflect registration order). See Section 10.11.1 for additional information on the ContentNotificationRegistrationRequest message.

core:Ext [Optional] –Any additional elements from other namespaces.

10.11 Content Notification Registration Request and Response

A CIS shall support registration for ContentNotification message delivery. The ContentNotificationRegistrationRequest message enables a client to specify notification interests relative to a selectively defined asset set. On receipt of an asset update, addition or deletion event from the underlying content store, the CIS shall send a ContentNotification message to each matching registered client.

10.11.1 ContentNotificationRegistrationRequest Message

The ContentNotificationRegistrationRequest message allows a client to specify a set of asset related notification interests. These registered interests shall be examined by the CIS relative to content store changes. If a change to the content store results in a match, a notification containing the result of the matching query shall be sent to the client in the form of a ContentNotification message.

The XML schema representation of the ContentNotificationRegistrationRequest message is illustrated in Figure 7 - ContentNotificationRegistrationRequest-XML Schema.

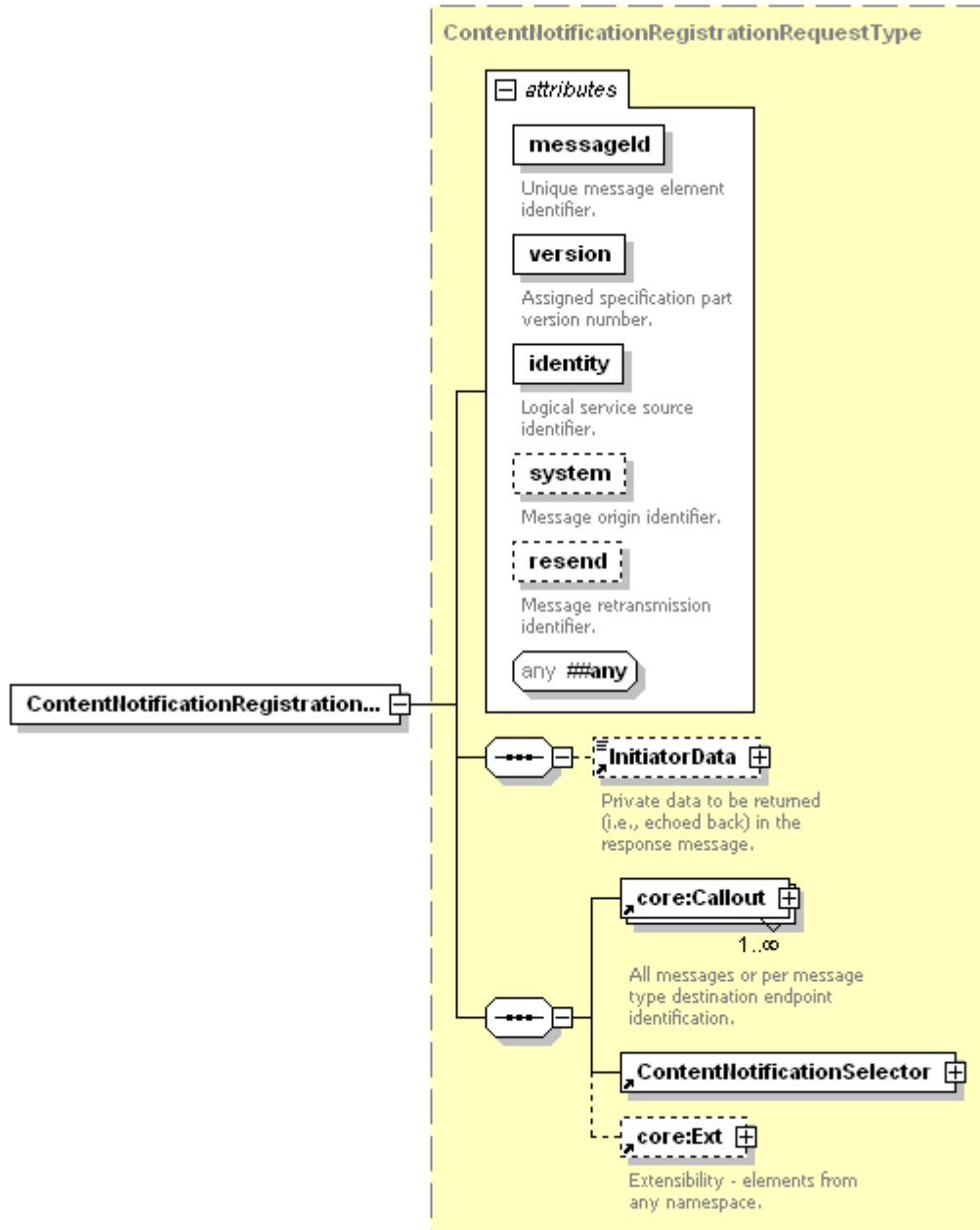


Figure 7 - ContentNotificationRegistrationRequest–XML Schema

The `ContentNotificationRegistrationRequest` message is derived from the core namespace base type `core:Msg_RequestBaseType` and defines the following attributes and elements.

@messageId [Required, core:messageIdAttrType] - The message identifier. See [SCTE130-2] for additional information.

@version [Required, core:versionAttrType] - The message specification version. See [SCTE130-2] for additional information.

@identity [Required, core:identityAttrType] - The origin logical service identifier. See [SCTE130-2] for additional information.

@system [Optional, core:systemAttrType] - The message source identifier. See [SCTE130-2] for additional information.

@resend [Optional, core:resendAttrType] – Message retransmission identifier. See [SCTE130-2] for additional information.

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

core:InitiatorData [Optional] – Private data that shall be returned in the ContentNotificationRegistrationResponse message. See [SCTE130-2] for additional details on the core:InitiatorData element.

core:Callout [Required] – The core:Callout element provides callback message and address information to the CIS. See [SCTE130-2] for a complete description of the Callout element.

Before generating a ContentNotificationRegistrationResponse, the CIS may send a core:ServiceCheckRequest message to a core:Address, in order to verify the validity of the callback endpoint. The CIS shall not fail the registration for a core:ServiceCheck message failure provided at least one Address of each Callout is reachable.

A CIS implementation shall recognize the values listed in Table 4 as values for the core:Callout @message attribute.

@message value	Description
ContentNotification	Value associated with the address endpoint where ContentNotification messages shall be sent.
ServiceStatusNotification	Value associated with the address endpoint where ServiceStatusNotification messages shall be sent.
DeregistrationNotification	Value associated with the address endpoint where DeregistrationNotification messages shall be sent.
...	User defined address endpoint outside of the scope of this specification. The string shall be prefixed with the text “private:”.

Table 4: ContentNotificationRegistrationResponse core:Callout @message values

If the @message attribute of the core:Callout element is blank, a CIS implementation shall send all unsolicited messages to the default endpoint supplied in the ContentNotificationRegistrationRequest from the client.

ContentNotificationSelector [Required] – The ContentNotificationSelector element contains all of the necessary elements required to specify a collection of asset interests for notification purposes. See Section 11.2 for additional details on the ContentNotificationSelector element.

core:Ext [Optional] – Any additional elements from other namespaces.

10.11.2 ContentNotificationRegistrationResponse Message

Upon completion of processing a ContentNotificationRegistrationRequest message, the CIS shall respond with a ContentNotificationRegistrationResponse message.

The XML schema diagram for the ContentNotificationRegistrationResponse message is as follows in Figure 8 - ContentNotificationRegistrationResponse

.

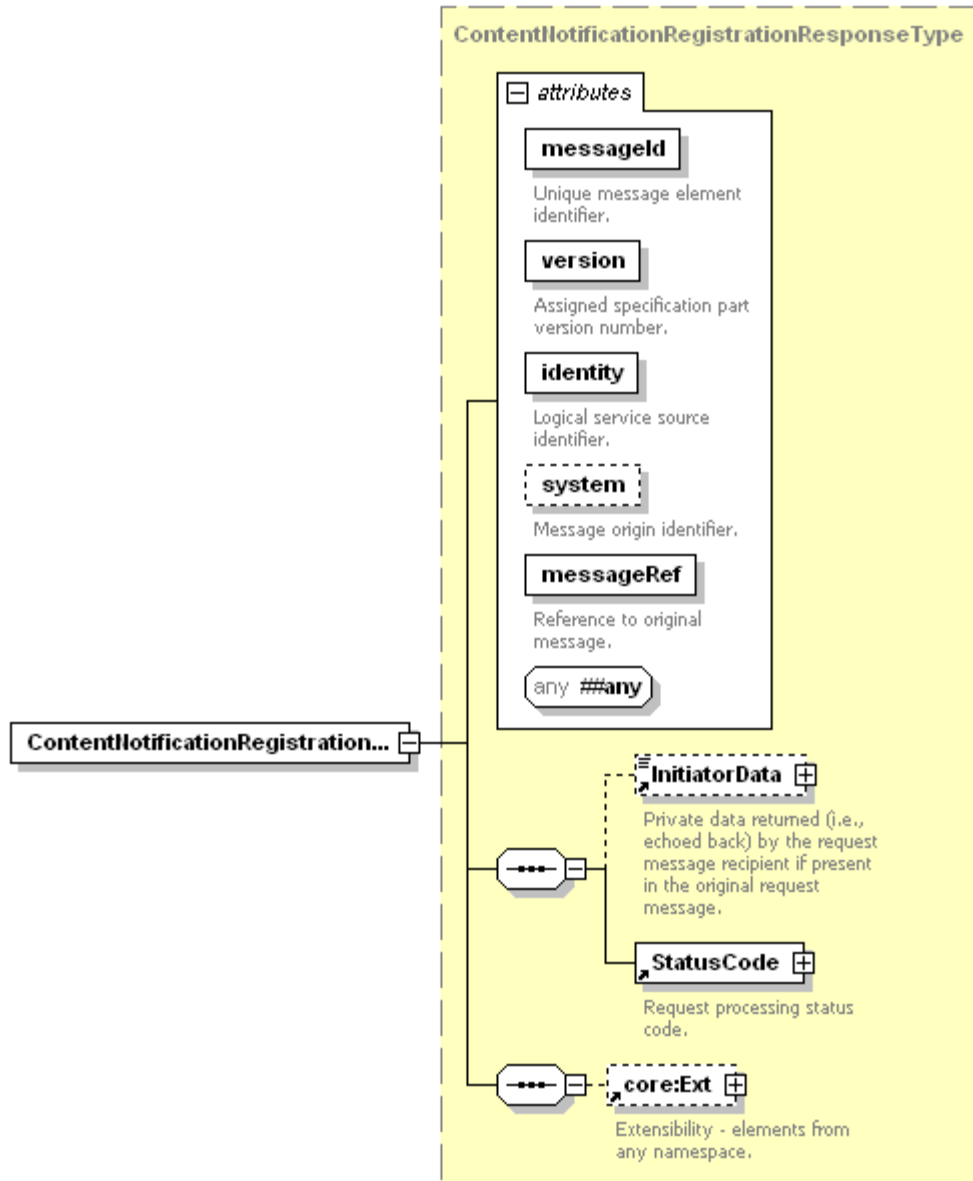


Figure 8 - ContentNotificationRegistrationResponse

The ContentNotificationRegistrationResponse is derived from the core namespace base type core:Msg_ResponseBaseType and defines the following attributes and elements.

@messageId [Required, core:messageIdAttrType] - The message identifier. See [SCTE130-2] for additional information.

@version [Required, core:versionAttrType] - The message specification version. See [SCTE130-2] for additional information.

@identity [Required, core:identityAttrType] - The origin logical service identifier. See [SCTE130-2] for additional information.

@system [Optional, core:systemAttrType] - The message source identifier. See [SCTE130-2] for additional information.

@messageRef [Required, core:messageRefAttrType] - A reference to the ContentNotificationRegistrationRequest message element initiating this message exchange. The value shall be the ContentNotificationRegistrationRequest message's @messageId attribute value. See [SCTE130-2] for additional information on the core:messageRefAttrType.

@##any [Optional] – Any additional attributes from any namespace

core:InitiatorData [Optional] – Private data from the ContentNotificationRegistrationRequest message. See [SCTE130-2] for additional details on the core:InitiatorData element.

core:StatusCode [Required] – A core:StatusCode element for communicating status information to the client. See [SCTE130-2] for additional information.

core:Ext [Optional] – Any additional elements from other namespaces.

10.12 Content Notification and Acknowledgement

A CIS shall support the exchange of ContentNotification and ContentNotificationAcknowledgement messages with registered consumers for the purpose of content asset change notification.

10.12.1 ContentNotification Message

Upon detection of a change in one or more assets referenced by the CIS, a CIS shall send a ContentNotification message to qualified, registered clients. A change to an asset includes the set of definitions described in Table 5. The values shall appear exactly as they appear in Table 5 (i.e., all in lower case)

ContentNotification@Type value	Description
update	An asset has been updated (i.e., some part of an asset has changed)
new	A new asset has been added to the collection of assets referenced by the CIS
delete	A CIS referenced asset has been deleted.
...	User defined ContentNotification types outside of the scope of this specification. The string shall be prefixed with the text "private:".

Table 5: ContentNotification@Type values

The XML schema for the ContentNotification message is illustrated in Figure 9.

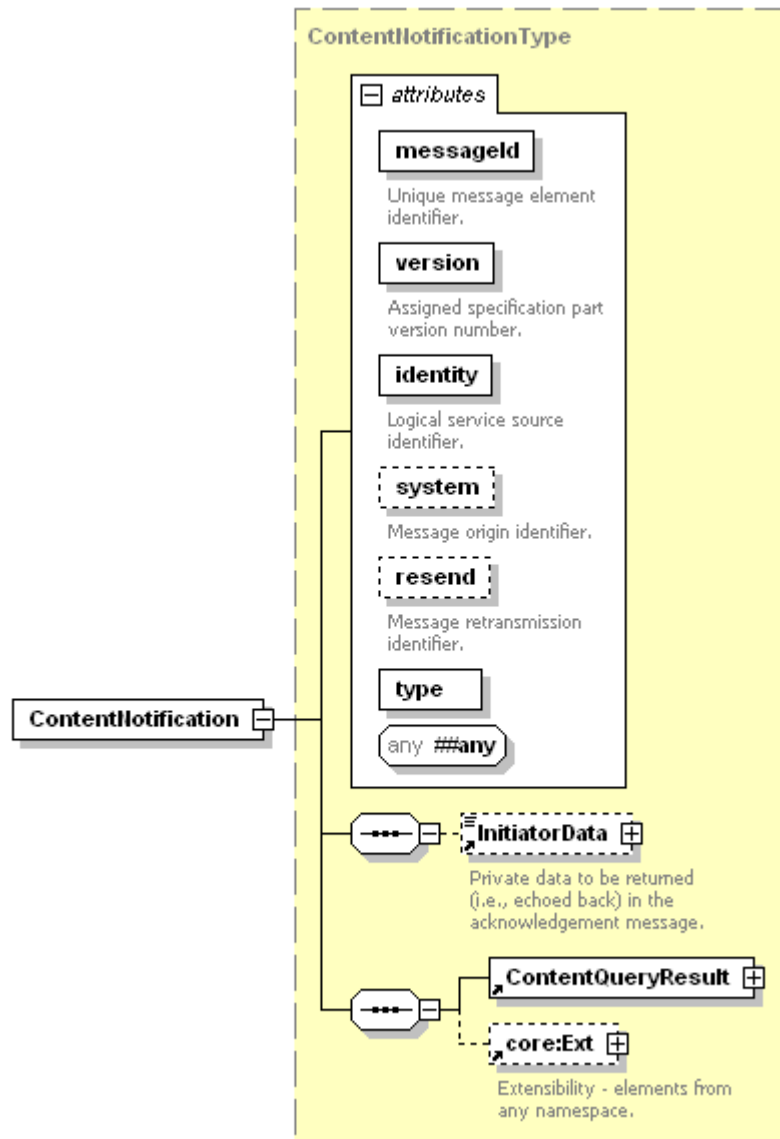


Figure 9 - ContentNotification–XML Schema

The ContentNotification message is derived from the core namespace base type core:Msg_NotificationBaseType and defines the following attributes and elements.

@messageId [Required, core:messageIdAttrType] - The message identifier. See [SCTE130-2] for additional information.

@version [Required, core:versionAttrType] - The message specification version. See [SCTE130-2] for additional information.

@identity [Required, core:identityAttrType] - The origin logical service identifier. See [SCTE130-2] for additional information.

@system [Optional, core:systemAttrType] - The message source identifier. See [SCTE130-2] for additional information.

@resend [Optional, core:resendAttrType] – Message retransmission identifier. See [SCTE130-2] for additional information.

@type [Required, cis:contentNotificationTypeEnumeration] – The @type attribute is an enumeration that shall contain one of the values listed above in Table 5.

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

core:InitiatorData [Optional] – Private data that shall be returned in the ContentNotificationAcknowledgement message. See [SCTE130-2] for additional details on the core:InitiatorData element.

ContentQueryResult [Required] - This element contains the list of assets that were subject to a change in the repository and have qualified against the client's original content notification requirements. See Section 11.5 for additional information on the ContentQueryResult element.

core:Ext [Optional] – Any additional elements from other namespaces.

10.12.2 ContentNotificationAcknowledgement Message

Upon the receipt of a ContentNotification message, a CIS client shall respond with a ContentNotificationAcknowledgement message.

The XML schema for the ContentNotificationAcknowledgement element is illustrated in Figure 10.

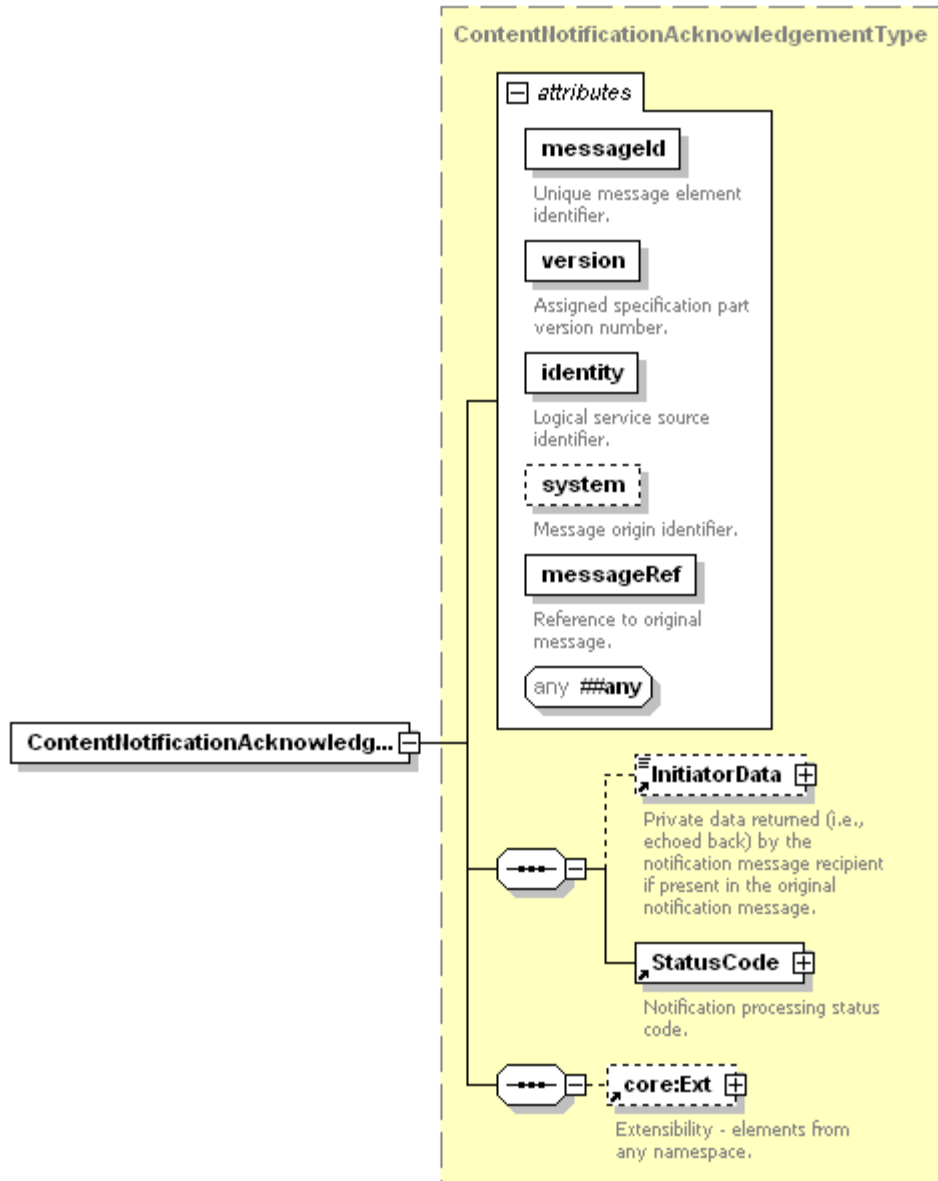


Figure 10 - ContentNotificationAcknowledgement XML Schema

The ContentNotificationAcknowledgement message is derived from the core namespace base type `core:Msg_AcknowledgementBaseType` and defines the following attributes and elements.

@messageId [Required, core:messageIdAttrType] - The message identifier. See [SCTE130-2] for additional information.

@version [Required, core:versionAttrType] - The message specification version. See [SCTE130-2] for additional information.

@identity [Required, core:identityAttrType] - The origin logical service identifier. See [SCTE130-2] for additional information.

@system [Optional, core:systemAttrType] - The message source identifier. See [SCTE130-2] for additional information.

@messageRef [Required, core:messageRefAttrType] - A reference to the ContentNotification message element initiating this message exchange. The value shall be the ContentNotification message's @messageId attribute value. See [SCTE130-2] for additional information on the core:messageRefAttrType.

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

core:InitiatorData [Optional] – Private data from the ContentNotification message. See [SCTE130-2] for additional details on the core:InitiatorData element.

core:StatusCode [Required] – A core:StatusCode element for communicating status information to the client. See [SCTE130-2] for additional information.

core:Ext [Optional] – Any additional elements from other namespaces.

10.13 Create Cursor Request and Response

A CIS shall support cursors of static asset information for both basic and advanced queries, which shall exist for a specified duration. Upon creation of a cursor on the CIS, the asset information in the cursor shall remain static relative to the referenced content store.

Cursors have a limited life span, which is first requested by the client, but may be overridden by a CIS. As part of the CreateCursorRequest message, the client shall specify a @cursorExpires date and time value attribute. This is a request to a CIS for a specific end date and value for the cursor identified by the @cursorId attribute. A CIS, in order to maintain overall system health, may choose to override a requested cursor expires end date and time value and substitute a different, implementation specific, cursor expires end date and time value. See section 11.4 for additional information on cursors.

10.13.1 CreateCursorRequest Message

The CreateCursorRequest message is used to create an instance of a static cursor on a CIS.

The XML schema for the CreateCursorRequest message is listed in **Error! Reference source not found.**

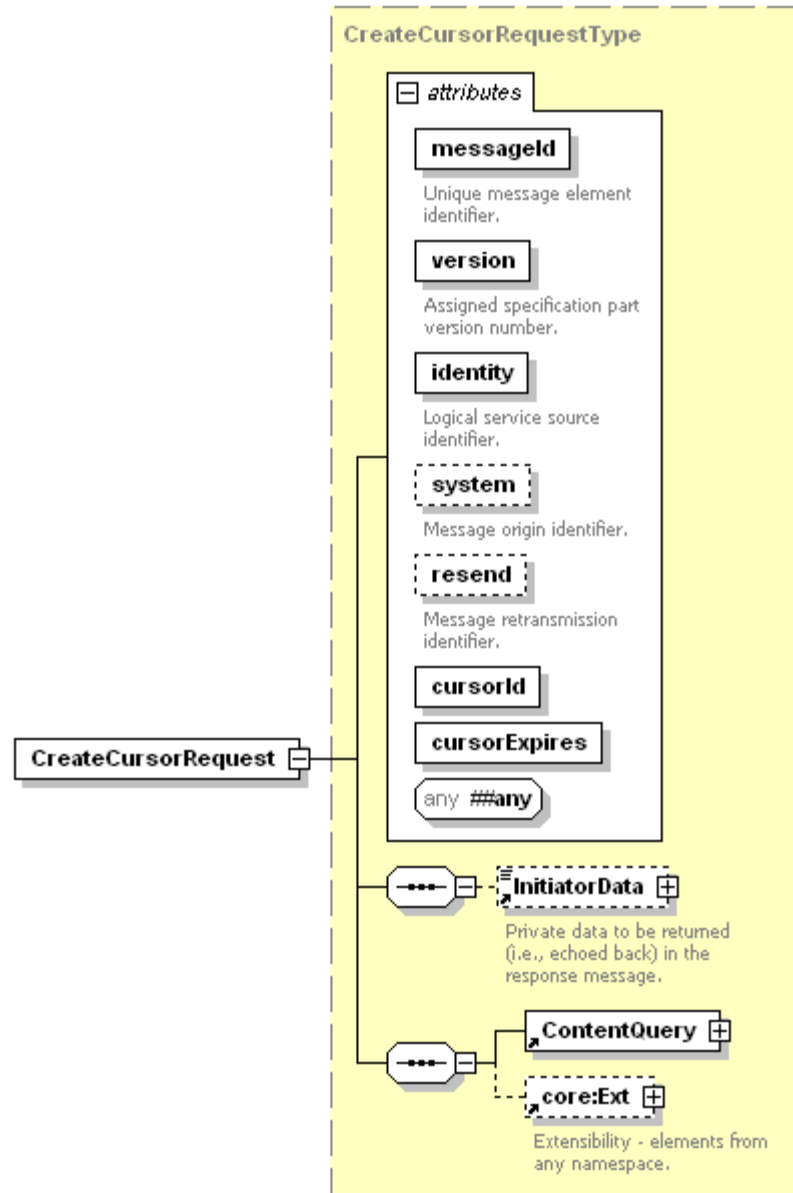


Figure 11 - CreateCursorRequest XML Schema

The CreateCursorRequest message is derived from the core namespace base type core:Msg_RequestBaseType and defines the following attributes and elements.

@messageId [Required, core:messageIdAttrType] - The message identifier. See [SCTE130-2] for additional information.

@version [Required, core:versionAttrType] - The message specification version. See [SCTE130-2] for additional information.

@identity [Required, core:identityAttrType] - The origin logical service identifier. See [SCTE130-2] for additional information.

@system [Optional, core:systemAttrType] - The message source identifier. See [SCTE130-2] for additional information.

@resend [Optional, core:resendAttrType] – Message retransmission identifier. See [SCTE130-2] for additional information.

@cursorId [Required, cis:cursorIdAttrType] – The @cursorId attribute is a client generated identifier which shall be a service channel unique value. See Section 12.1.3 for additional information on the cis:cursorIdAttrType.

@cursorExpires [Required, core:dateTimeTimezoneType] – The @cursorExpires attribute is a client request for a cursor expiration date and time value. A CIS shall not be required to create the cursor with the requested end date and time value. A CIS may override the requested end date and time value by returning an implementation specific end date and time value that better fits within the implementation’s specific design constraints.

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

core:InitiatorData [Optional] – Private data that shall be returned in the CreateCursorResponse message. See [SCTE130-2] for additional details on the core:InitiatorData element.

ContentQuery [Required] – The ContentQuery element contains the necessary attributes and elements for a CIS to execute one or more queries against its associated content store(s). Query selected items shall be added to a static cursor construct identified by the supplied @cursorId attribute. See Section 11.3 for additional information on the ContentQuery element.

core:Ext [Optional] – Any additional elements from other namespaces.

10.13.2 CreateCursorResponse Message

Upon receipt of a CreateCursorRequest message, the CIS shall attempt to create the required cursor and shall respond to the client with a CreateCursorResponse message. If the query is not successful (i.e., the core:StatusCode value does not equate to success) then the cursor shall not be established.

The XML schema for the CreateCursorResponse message is listed in Figure 12.

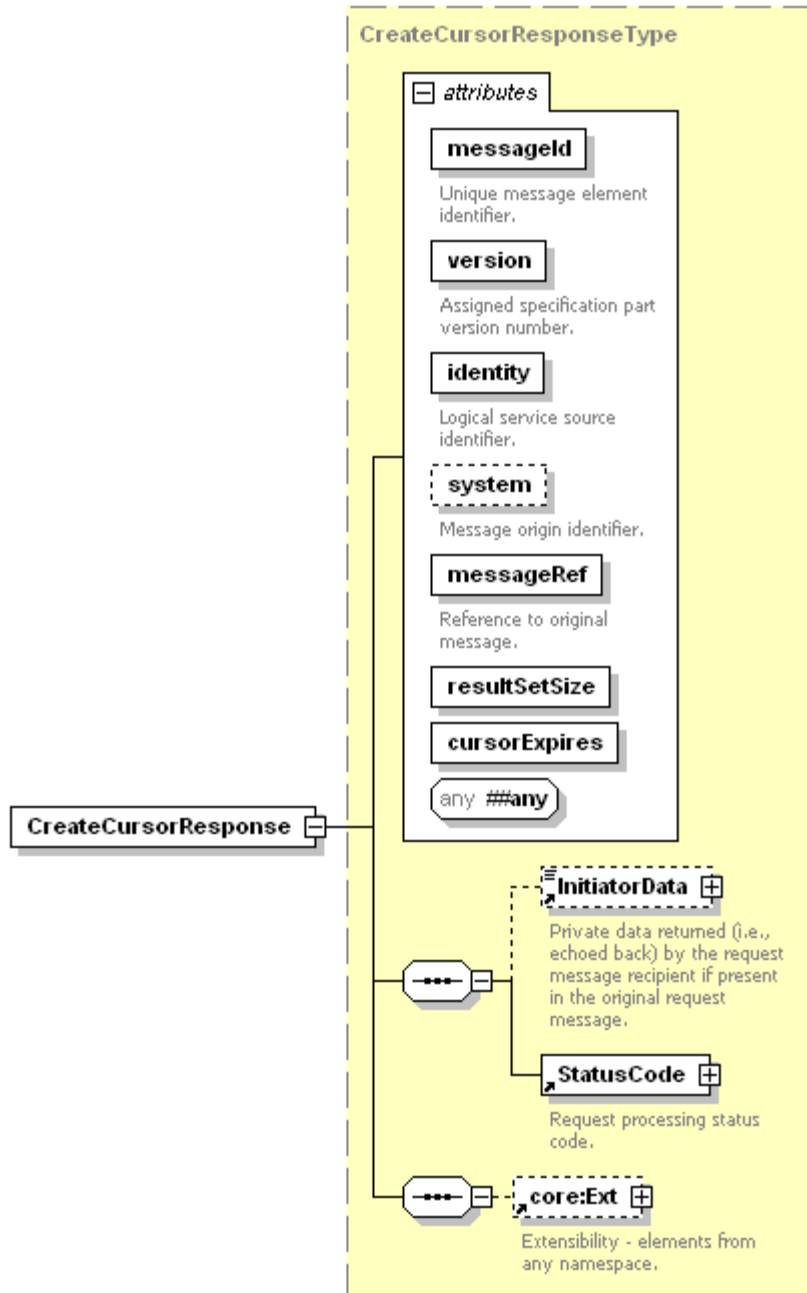


Figure 12 - CreateCursorResponse XML Schema

The CreateCursorResponse message is derived from the core namespace base type core:Msg_ResponseBaseType and defines the following attributes and elements.

@messageId [Required, core:messageIdAttrType] - The message identifier. See [SCTE130-2] for additional information.

@version [Required, core:versionAttrType] - The message specification version. See [SCTE130-2] for additional information.

@identity [Required, core:identityAttrType] - The origin logical service identifier. See [SCTE130-2] for additional information.

@system [Optional, core:systemAttrType] - The message source identifier. See [SCTE130-2] for additional information.

@messageRef [Required, core:messageRefAttrType] - A reference to the CreateCursorRequest message element initiating this message exchange. The value shall be the CreateCursorRequest message's @messageId attribute value. See [SCTE130-2] for additional information on the core:messageRefAttrType.

@resultSetSize [Required, xsd:nonNegativeInteger] – The @resultSetSize attribute is a non-negative integer, describing the number of records contained in the cursor.

@cursorExpires [Required, core:dateTimeTimezoneType] – The @cursorExpires attribute contains the CIS determined cursor expiration date and time value. The value may be either the user requested end date and time value from the CreateCursorRequest message, or a CIS specified expiration date and time value.

core:InitiatorData [Optional] – Private data from the CreateCursorRequest message. See [SCTE130-2] for additional details on the core:InitiatorData element.

core:StatusCode [Required] – An core:StatusCode element for communicating status information to the client. See [SCTE130-2] for additional information.

core:Ext [Optional] – Any additional elements from other namespaces.

10.14 Cancel Cursor Request and Response

A CIS shall allow a client to cancel an existing cursor before the expiration duration has been exceeded.

A CIS client may complete interacting with a CIS cursor before the cursor actually expires, and may choose to terminate the CIS cursor. Once a cursor has been terminated or has expired, a CIS may release resources associated with the cursor.

Any additional communications from the client that reference the canceled or expired cursor shall result in an error with the core:StatusCode @detail attribute set to the value 4001 (Cursor Undefined), as described in Table 13 StatusCode Details.

10.14.1 CancelCursorRequest Message

This message allows a CIS client to terminate a cursor before the expected cursor expiration time.

The XML schema for the CancelCursorRequest message is illustrated in Figure 13.

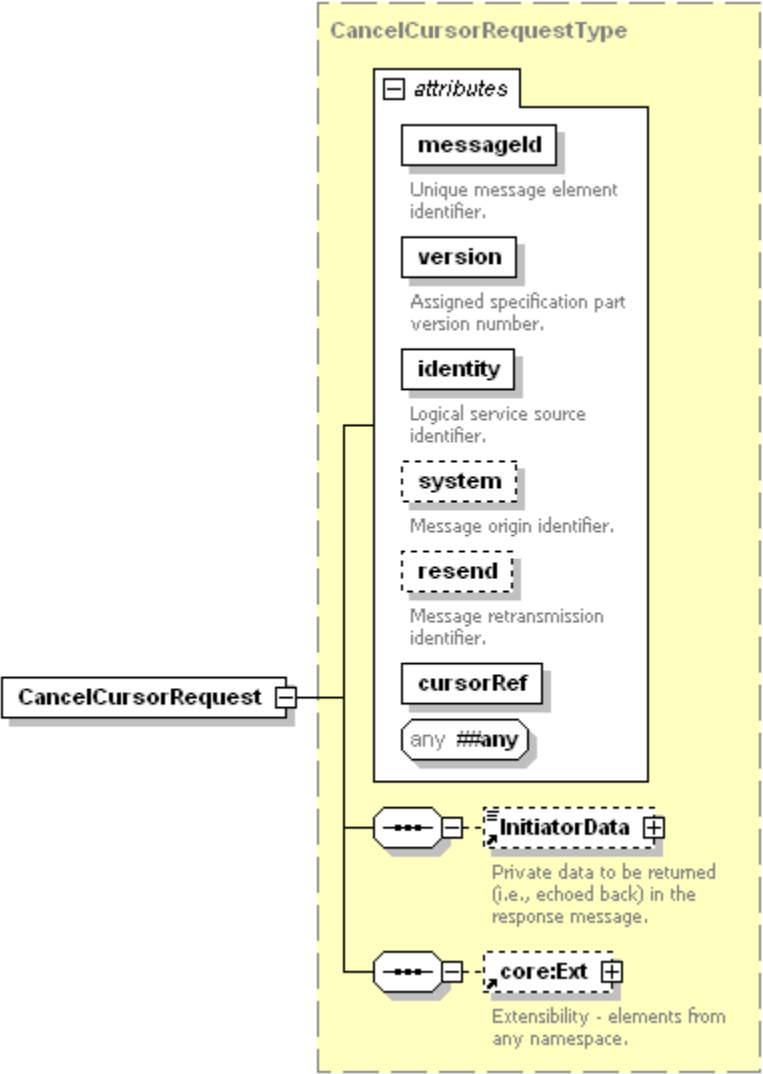


Figure 13 - CancelCursorRequest - XML Schema

The CancelCursorRequest message is derived from the core namespace base type core:Msg_RequestBaseType and defines the following attributes and elements.

@messageId [Required, core:messageIdAttrType] - The message identifier. See [SCTE130-2] for additional information.

@version [Required, core:versionAttrType] - The message specification version. See [SCTE130-2] for additional information.

@identity [Required, core:identityAttrType] - The origin logical service identifier. See [SCTE130-2] for additional information.

@system [Optional, core:systemAttrType] - The message source identifier. See [SCTE130-2] for additional information.

@resend [Optional, core:resendAttrType] – Message retransmission identifier. See [SCTE130-2] for additional information.

@cursorRef [Required, cis:cursorIdRefAttrType] – The @cursorRef attribute identifies a cursor previously created with a CreateCursorRequest message. The @cursorRef attribute shall contain the value from the @cursorId attribute of the CreateCursorRequest message. See Section 12.1.4 for additional information on cis:cursorIdRefAttrType.

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

core:InitiatorData [Optional] – Private data that shall be returned in the CancelCursorResponse message. See [SCTE130-2] for additional details on the core:InitiatorData element.

core:Ext [Optional] – Any additional elements from other namespaces.

10.14.2 CancelCursorResponse Message

Upon receipt of a CancelCursorRequest message, the CIS shall terminate the cursor identified by the @cursorRef attribute, and shall return a CancelCursorResponse message.

The XML schema for the CancelCursorResponse message is illustrated in Figure 14.

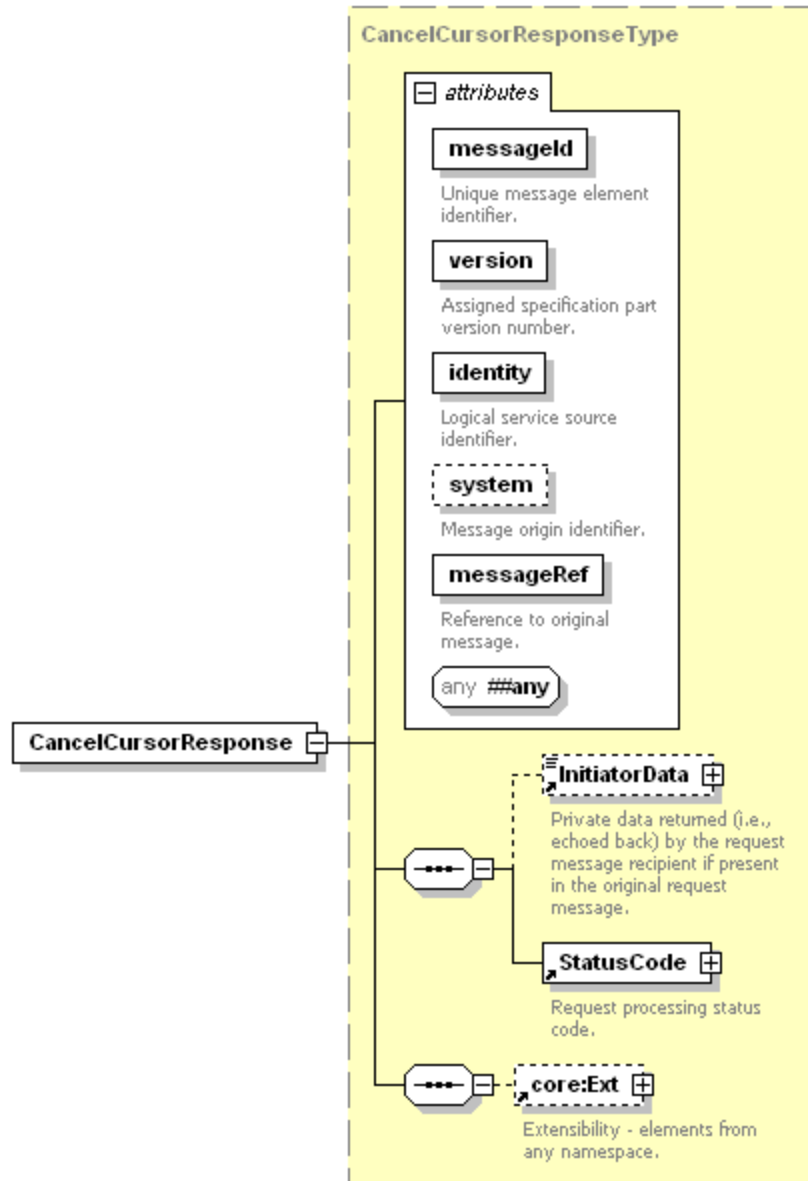


Figure 14 - CancelCursorResponse - XML Schema

The CancelCursorResponse message is derived from the core namespace base type core:Msg_ResponseBaseType and defines the following attributes and elements.

@messageId [Required, core:messageIdAttrType] - The message identifier. See [SCTE130-2] for additional information.

@version [Required, core:versionAttrType] - The message specification version. See [SCTE130-2] for additional information.

@identity [Required, core:identityAttrType] - The origin logical service identifier. See [SCTE130-2] for additional information.

@system [Optional, core:systemAttrType] - The message source identifier. See [SCTE130-2] for additional information.

@messageRef [Required, core:messageRefAttrType] - A reference to the CancelCursorRequest message element initiating this message exchange. The value shall be the CancelCursorRequest message's @messageId attribute value. See [SCTE130-2] for additional information on the core:messageRefAttrType.

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

core:InitiatorData [Optional] – Private data from the CancelCursorRequest message. See [SCTE130-2] for additional details on the core:InitiatorData element.

core:StatusCode [Required] – A core:StatusCode element for communicating status information to the client. See [SCTE130-2] for additional information.

core:Ext [Optional] – Any additional elements from other namespaces.

10.15 Content Query Request and Response

The ContentQueryRequest and ContentQueryResponse messages are used by clients to query for assets referenced by a CIS.

This message supports both basic and advanced query mechanisms as well as references to existing static cursor information.

10.15.1 ContentQueryRequest Message

The ContentQueryRequest message is the primary mechanism for a client to execute a query on a CIS. This message contains either a reference to a previously established CIS cursor or a ContentQuery element.

A ContentQueryRequest message containing a ContentQuery element shall have the query executed against all of the assets referenced by the CIS. The resulting collection of assets shall be returned in the ContentQueryResponse message.

A ContentQueryRequest message containing a Cursor element shall return in a ContentQueryResponse message all of the assets inclusive between the @startIndex and @count value within the static cursor data structure.

The ContentQueryRequest message XML schema definition is illustrated in Figure 15.

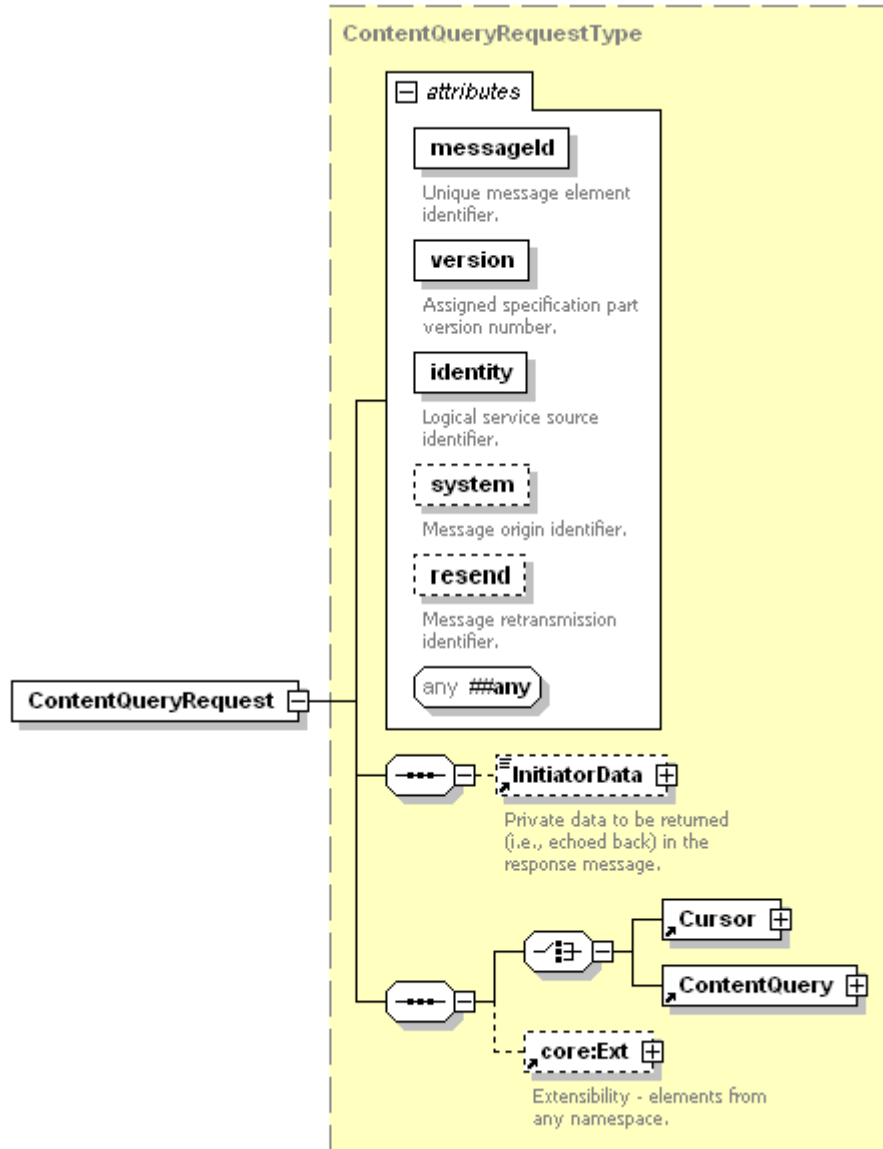


Figure 15 - ContentQueryRequest–XML Schema

The ContentQueryRequest message is derived from the core namespace base type core:Msg_RequestBaseType and defines the following attributes and elements.

@messageId [Required, core:messageIdAttrType] - The message identifier. See [SCTE130-2] for additional information.

@version [Required, core:versionAttrType] - The message specification version. See [SCTE130-2] for additional information.

@identity [Required, core:identityAttrType] - The origin logical service identifier. See [SCTE130-2] for additional information.

@system [Optional, core:systemAttrType] - The message source identifier. See [SCTE130-2] for additional information.

@resend [Optional, core:resendAttrType] – Message retransmission identifier. See [SCTE130-2] for additional information.

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

core:InitiatorData [Optional] – Private data that shall be returned in the ContentQueryResponse message. See [SCTE130-2] for additional details on the core:InitiatorData element.

Cursor [Required on choice] – The Cursor element contains an @cursorRef attribute identifying a cursor previously established using the CreateCursorRequest message. If the cursor references a canceled or expired cursor the ContentQueryResponse message shall contain an error with the core:StatusCode @detail attribute set to the value 4001 (Cursor Undefined), as described in Table 13 StatusCode Details

See Section 11.4 for details on the cis:Cursor element.

ContentQuery [Required on choice] – The ContentQuery element contains all elements and attributes required for an asset information query. The entire result set of the query is returned in the ContentQueryResponse message. See Section 11.3 for additional information on the ContentQuery element.

core:Ext [Optional] – Any additional elements from other namespaces.

10.15.2 ContentQueryResponse Message

Upon receipt of a ContentQueryRequest message, the CIS shall respond with a ContentQueryResponse message. This message contains the query results (advanced, basic or cursor) in the ContentQueryResult element.

The XML schema definition for this message is illustrated in Figure 16.

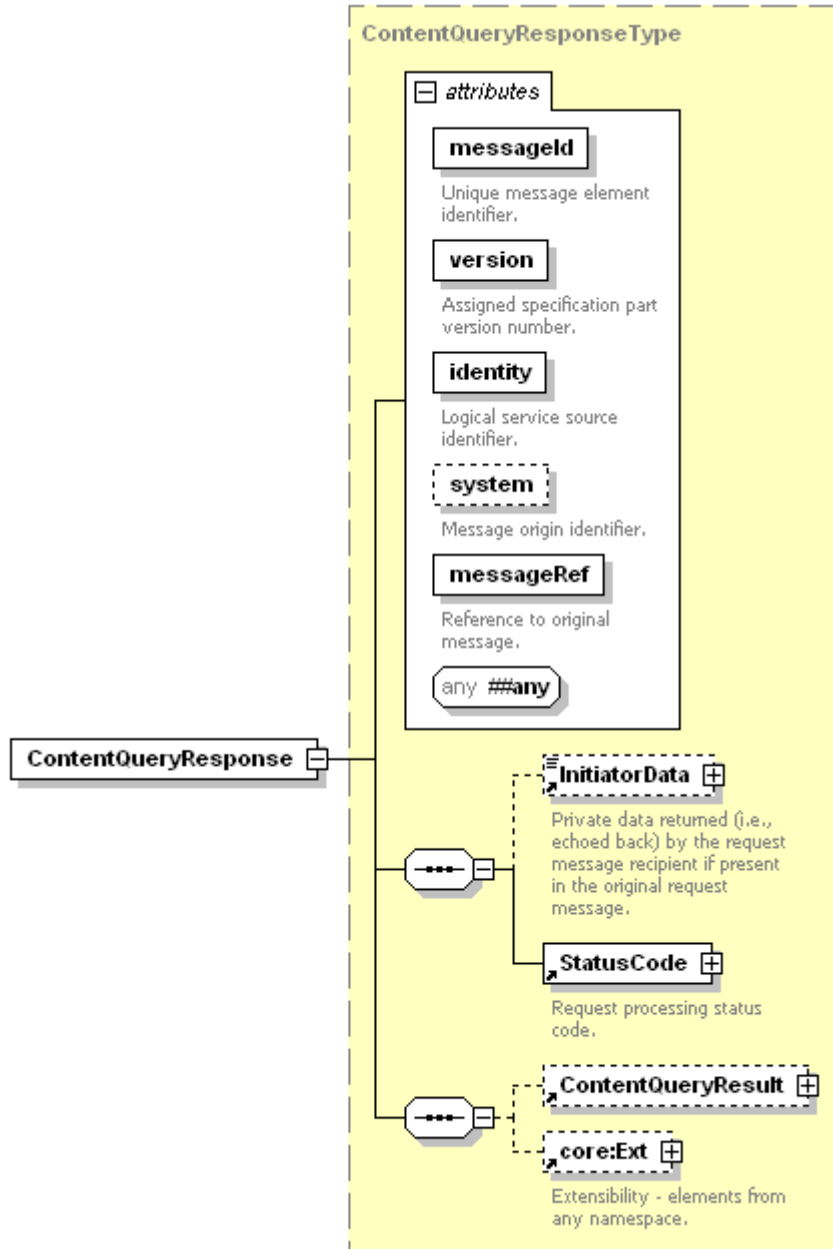


Figure 16 - ContentQueryResponse–XML Schema

The ContentQueryResponse message is derived from the core namespace base type core:Msg_ResponseBaseType and defines the following attributes and elements.

@messageId [Required, core:messageIdAttrType] - The message identifier. See [SCTE130-2] for additional information.

@version [Required, core:versionAttrType] - The message specification version. See [SCTE130-2] for additional information.

@identity [Required, core:identityAttrType] - The origin logical service identifier. See [SCTE130-2] for additional information.

@system [Optional, core:systemAttrType] - The message source identifier. See [SCTE130-2] for additional information.

@messageRef [Required, core:messageRefAttrType] - A reference to the ContentQueryRequest message element initiating this message exchange. The value shall be the ContentQueryRequest message's @messageId attribute value. See [SCTE130-2] for additional information on the core:messageRefAttrType.

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

core:InitiatorData [Optional] – Private data from the ContentQueryRequest message. See [SCTE130-2] for additional details on the core:InitiatorData element.

core:StatusCode [Required] – An core:StatusCode element for communicating status information to the client. See [SCTE130-2] for additional information.

ContentQueryResult [Optional] – The ContentQueryResult element contains the result of the execution of the query supplied in the ContentQueryRequest or the list of assets referenced in the Cursor. This element shall not be returned if the original query did not resolve to any asset references or the StatusCode element indicates an error situation. See Section 11.5 for additional information on the ContentQueryResult element.

core:Ext [Optional] –Any additional elements from other namespaces.

10.16 Content Notification Deregister Request and Response

A CIS shall allow a client to de-register a previously registered ContentNotificationRegistrationRequest message. This message exchange allows a CIS client to dynamically modify registration notifications using individual register and un-register commands.

10.16.1 ContentNotificationDeregisterRequest Message

The ContentNotificationDeregisterRequest message removes an existing content notification registration from the CIS.

The XML schema for the ContentNotificationDeregisterRequest message is illustrated in Figure 17.

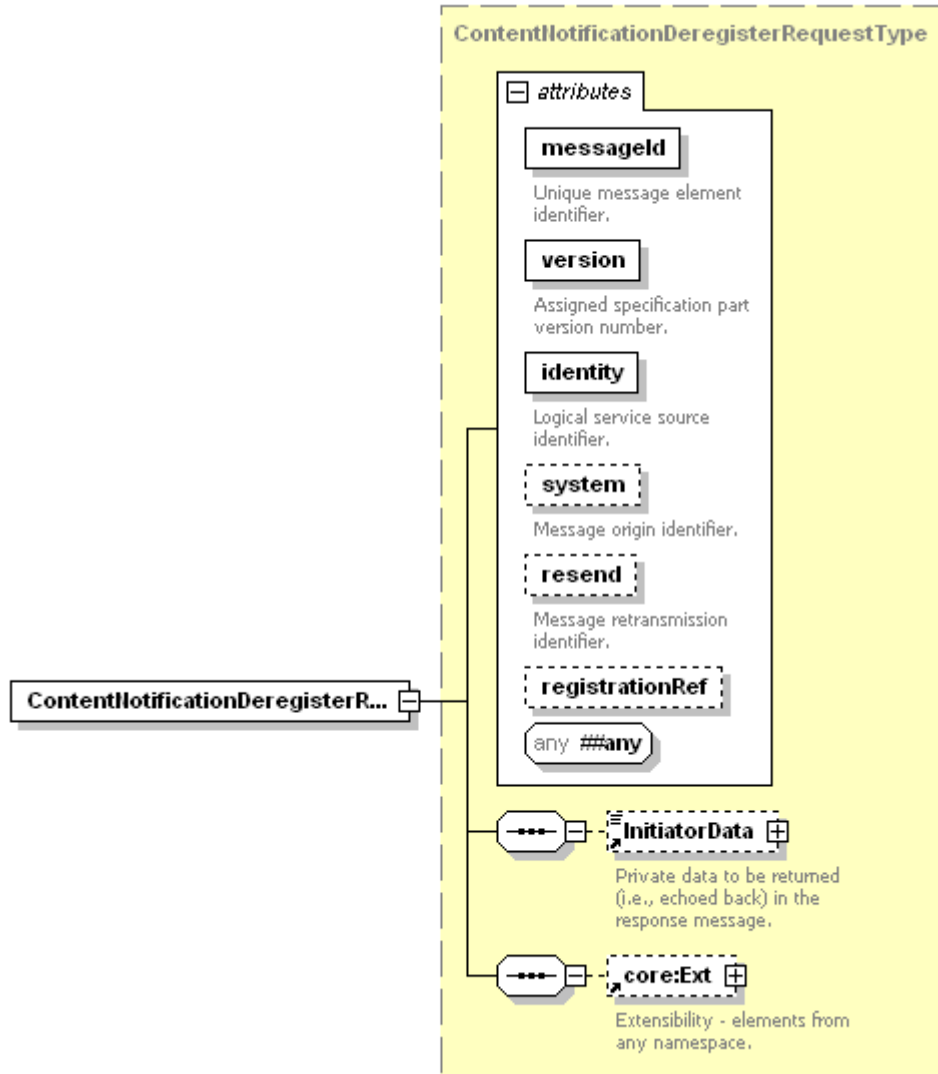


Figure 17 - ContentNotificationDeregisterRequest

The `ContentNotificationDeregisterRequest` message is derived from the core namespace base type `core:Msg_RequestBaseType` and defines the following attributes and elements.

@messageId [Required, core:messageIdAttrType] - The message identifier. See [SCTE130-2] for additional information.

@version [Required, core:versionAttrType] - The message specification version. See [SCTE130-2] for additional information.

@identity [Required, core:identityAttrType] - The origin logical service identifier. See [SCTE130-2] for additional information.

@system [Optional, core:systemAttrType] - The message source identifier. See [SCTE130-2] for additional information.

@resend [Optional, core:resendAttrType] – Message retransmission identifier. See [SCTE130-2] for additional information.

@registrationRef [Optional, core:registrationRefAttrType] – The @registrationRef identifies the original ContentNotificationRegistrationRequest message being deregistered. The value shall be the ContentNotificationRegistrationRequest message's @messageId attribute value, which in a retransmitted deregistration message, shall be the @resend attribute's value. If the @registrationRef attribute is omitted from the message, the CIS shall remove all ContentNotificationRegistrationRequest items scoped to the @identity attribute.

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

core:InitiatorData [Optional] – Private data that shall be returned in the ContentNotificationDeregisterResponse message. See [SCTE130-2] for additional details on the core:InitiatorData element.

core:Ext [Optional] – Any additional elements from other namespaces.

10.16.2 ContentNotificationDeregisterResponse Message.

Upon receipt of a ContentNotificationDeregisterRequest message from a client, the CIS shall respond with a ContentNotificationDeregisterResponse message.

The XML schema for the ContentNotificationDeregisterResponse message is illustrated in Figure 18.

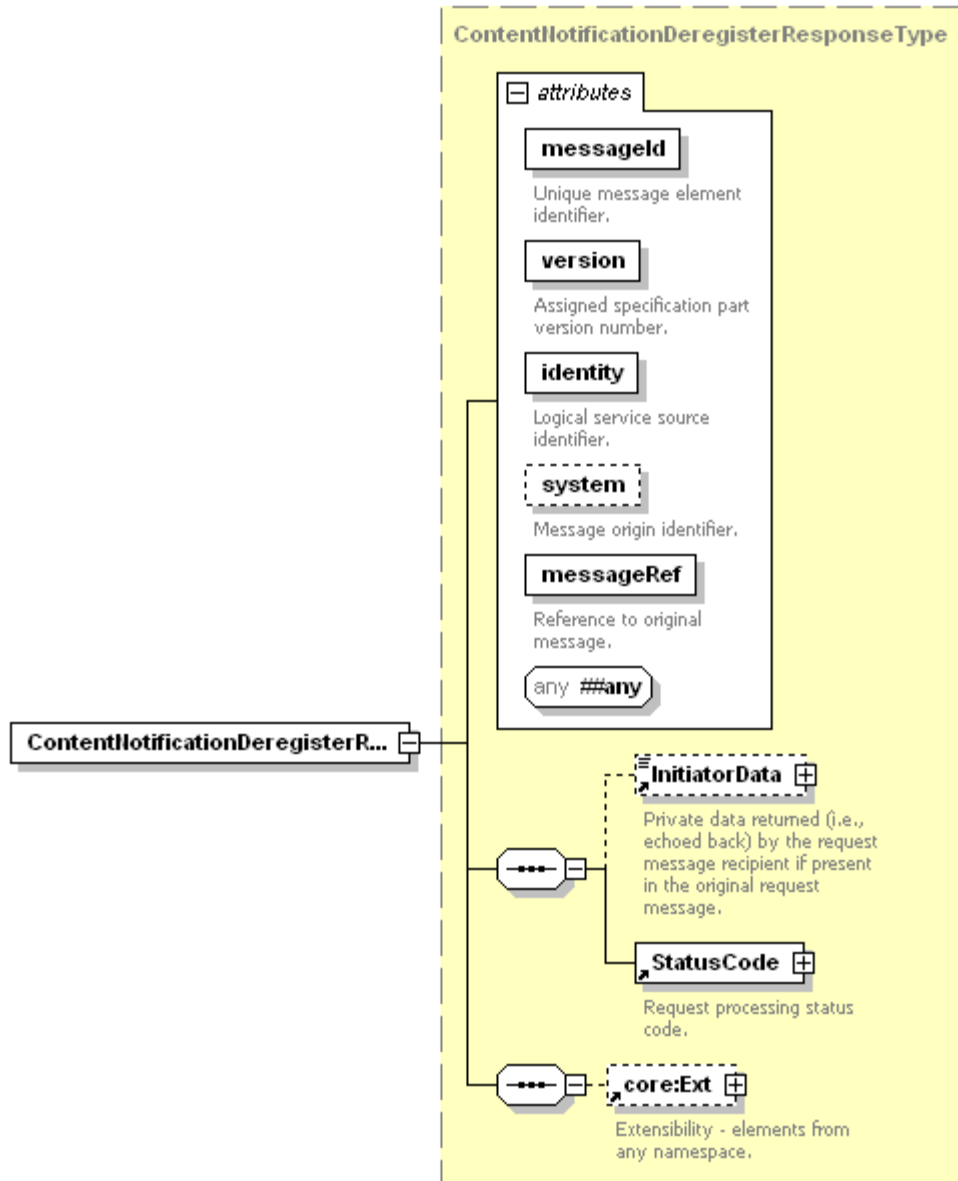


Figure 18 - ContentNotificationDeregisterResponse XML Schema

The ContentNotificationDeregisterResponse message is derived from the core namespace base type core:Msg_ResponseBaseType and defines the following attributes and elements.

@messageId [Required, core:messageIdAttrType] - The message identifier. See [SCTE130-2] for additional information.

@version [Required, core:versionAttrType] - The message specification version. See [SCTE130-2] for additional information.

@identity [Required, core:identityAttrType] - The origin logical service identifier. See [SCTE130-2] for additional information.

@system [Optional, core:systemAttrType] - The message source identifier. See [SCTE130-2] for additional information.

@messageRef [Required, core:messageRefAttrType] - A reference to the ContentNotificationRegistrationRequest message element initiating this message exchange. The value shall be the ContentNotificationRegistrationRequest message's @messageId attribute value. See [SCTE130-2] for additional information on the core:messageRefAttrType.

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

core:InitiatorData [Optional] – Private data from the ContentNotificationDeregisterRequest message. See [SCTE130-2] for additional details on the core:InitiatorData element.

core:StatusCode [Required] – An core:StatusCode element for communicating status information to the client. See [SCTE130-2] for additional information.

core:Ext [Optional] – Any additional elements from other namespaces.

10.17 Deregistration Notification and Acknowledgement

A CIS shall have the ability to deregister clients. Deregistration removes client registrations from the system and stops any content notification traffic from being sent to the deregistered client.

Upon receipt of a DeregistrationNotification message, a CIS client shall reply with a DeregistrationAcknowledgement message.

10.17.1 DeregistrationNotification Message

At any time, a CIS may issue one or more DeregistrationNotification messages to registered CIS clients. This informs the client that one or all of their active registrations (i.e., ContentNotificationRegistrationRequest messages) have been terminated and no further ContentNotifications shall be expected related to those registrations.

The XML schema for the DeregistrationNotification element is illustrated in Figure 19.

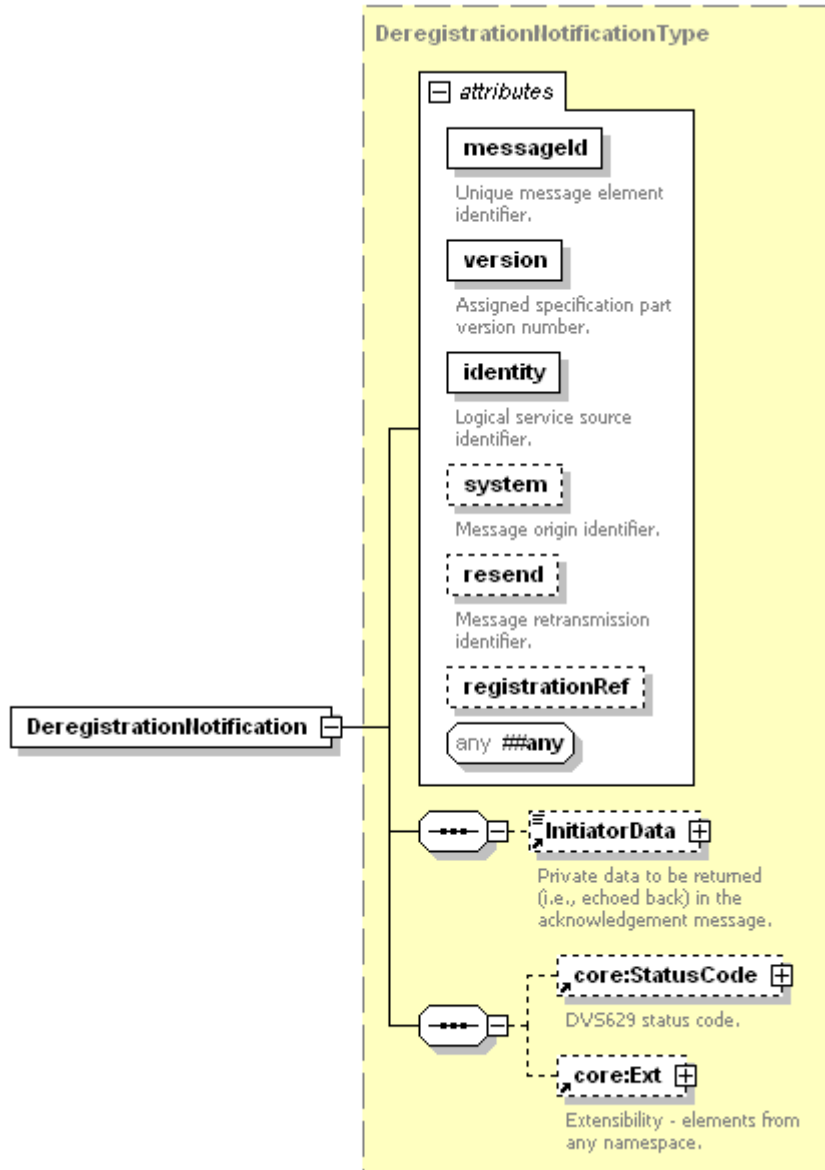


Figure 19 - DeregistrationNotification–XML Schema

The DeregistrationNotification message is derived from the core namespace base type core:Msg_NotificationBaseType and defines the following attributes and elements.

@messageId [Required, core:messageIdAttrType] - The message identifier. See [SCTE130-2] for additional information.

@version [Required, core:versionAttrType] - The message specification version. See [SCTE130-2] for additional information.

@identity [Required, core:identityAttrType] - The origin logical service identifier. See [SCTE130-2] for additional information.

@system [Optional, core:systemAttrType] - The message source identifier. See [SCTE130-2] for additional information.

@resend [Optional, core:resendAttrType] – Message retransmission identifier. See [SCTE130-2] for additional information.

@registrationRef [Optional, core:registrationRefAttrType] – When present, this attribute identifies the original ContentNotificationRegistrationRequest message that shall be deregistered. The value shall be the ContentNotificationRegistrationRequest message's @messageId attribute value, which in a retransmitted notification, shall be the @resend attribute's value. Issuing the DeregistrationNotification with this attribute informs the client that the CIS has cleared only this registration information associated with the specific ContentNotificationRegistrationRequest. If the @registrationRef attribute is absent, then all registrations associated with the specified client identity have been deregistered.

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

core:InitiatorData [Optional] – Private data that shall be returned in the DeregistrationAcknowledgement message. See [SCTE130-2] for additional details on the core:InitiatorData element.

core:StatusCode [Optional] – An optional core:StatusCode element for communicating status information to the client. See [SCTE130-2] for additional information.

core:Ext [Optional] – Any additional elements from other namespaces.

10.17.2 DeregistrationAcknowledgement Message

Upon receipt of a DeregistrationNotification message, a CIS client shall respond with a DeregistrationAcknowledgement message. This message informs the CIS that the notification message was received by the intended client and processed.

The XML schema for the DeregistrationAcknowledgement element is illustrated in Figure 20.

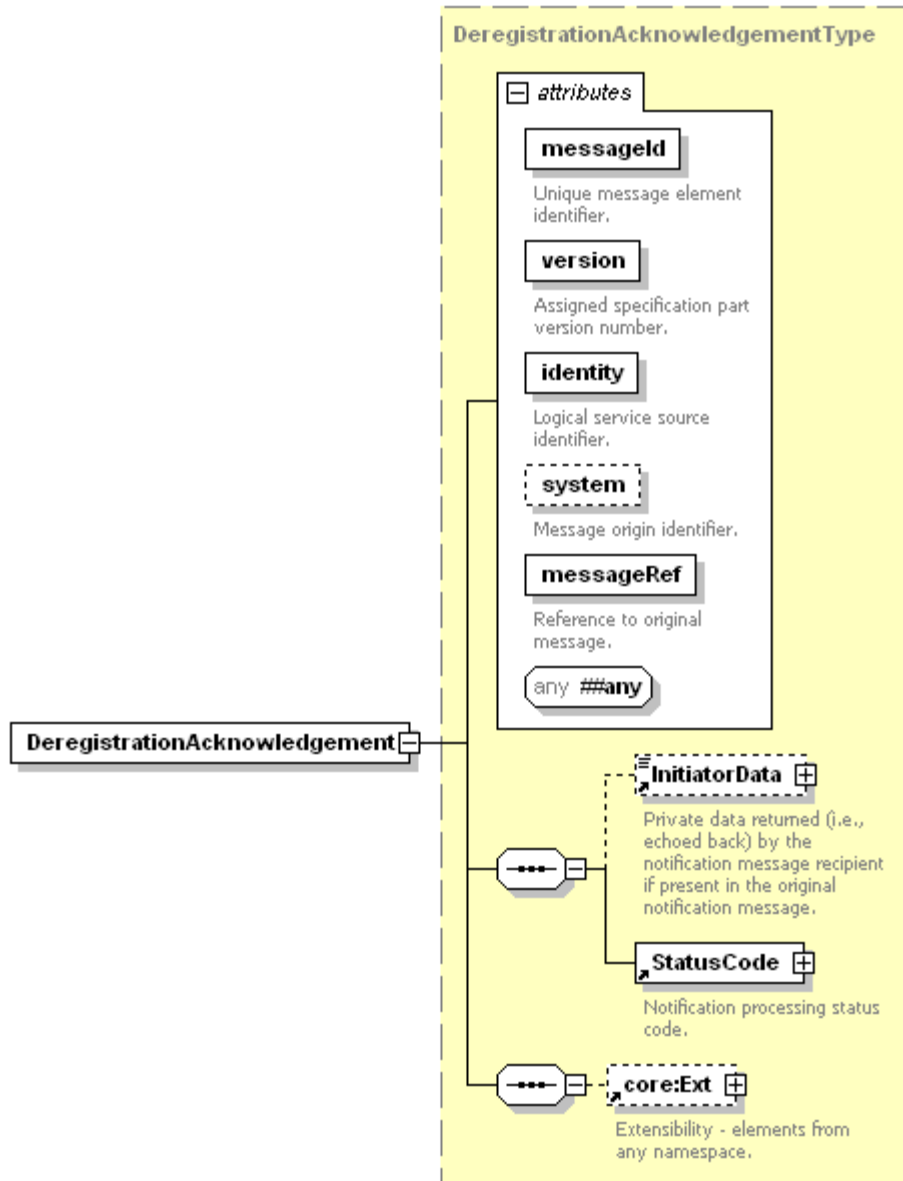


Figure 20 - DeregistrationAcknowledgement–XML Schema

The DeregistrationAcknowledgement message is derived from the core namespace base type core:Msg_AcknowledgementBaseType and defines the following attributes and elements.

@messageId [Required, core:messageIdAttrType] - The message identifier. See [SCTE130-2] for additional information.

@version [Required, core:versionAttrType] - The message specification version. See [SCTE130-2] for additional information.

@identity [Required, core:identityAttrType] - The origin logical service identifier. See [SCTE130-2] for additional information.

@system [Optional, core:systemAttrType] - The message source identifier. See [SCTE130-2] for additional information.

@messageRef [Required, core:messageRefAttrType] - A reference to the DeregistrationNotification message element initiating this message exchange. The value shall be the DeregistrationNotification message's @messageId attribute value. See [SCTE130-2] for additional information on the core:messageRefAttrType.

@##any [Optional] – Any additional attributes from any namespace

core:InitiatorData [Optional] – Private data from the DeregistrationNotification message. See [SCTE130-2] for additional details on the core:InitiatorData element.

core:StatusCode [Required] - An core:StatusCode element for communicating status information to the client. See [SCTE130-2] for additional information.

core:Ext [Optional] – Any additional elements from other namespaces.

10.18 Service Check Support

The CIS shall support the ServiceCheck message exchange, which includes the core:ServiceCheckRequest and core:ServiceCheckResponse messages as defined by [SCTE130-2].

10.19 Service Status Support

The CIS shall support the ServiceStatus message exchange, which includes the core:ServiceStatusNotification and core:ServiceStatusAcknowledgement messages as defined by [SCTE130-2].

11.0 CIS ELEMENT DETAILS

CIS elements are those that are used within the CIS top level message elements. Each of the CIS elementary messages defined in the CIS namespace are listed in Table 6 and are described in detail in subsequent document sections.

Element	Description
ContentNotificationSelector	Notification selector container
ContentQuery	ContentQueryRequest or CreateCursorRequest query container
Cursor	Cursor definition
ContentQueryResult	Result container for ContentQuery
QueryFilter	Container for FilterElements or AdvancedFilterElements
FilterElement	Individual filter item for ContentQuery
BasicQueryResultList	Core:Content container element
AdvancedFilterElement	Advanced query element for ContentQuery
AdvancedQueryResultList	AdvancedQueryResult container element
AdvancedQueryResult	Result container for advanced queries
DataModelList	Container element for core:ContentDataModel elements
AdvancedQueryLanguageList	Container element for AdvancedQueryLanguage elements
AdvancedQueryLanguage	Advanced query language descriptor.

Table 6: CIS Elementary Message Details

11.1 Basic and Advanced Queries

A CIS shall support basic query processing and should support advanced query language processing. If a CIS implementation supports advanced query language processing, then the implementation shall support both of the [W3C-XPath] and [W3C-XQuery] query languages listed in Table 7. The value shall appear exactly as they do in Table 7 (i.e., capitalized accordingly).

Query Language	Description
XPath	See [W3C-XPath]
XQuery	See [W3C-Xquery]
...	User defined query languages outside of the scope of this specification. The string shall be prefixed with the text “private:”.

Table 7: Advanced Query Languages

11.2 ContentNotificationSelector

The ContentNotificationSelector element defines a query that, when matched, generates a ContentNotification message. The matched query is identified via the @queryId attribute and the result set detail is controlled by the value of the @expandOutput attribute. The defined query may be associated with a specific content data model. See [SCTE130-2] for a more information on the core:ContentDataModel element.

The XML schema definition for this element is provided in Figure 21.

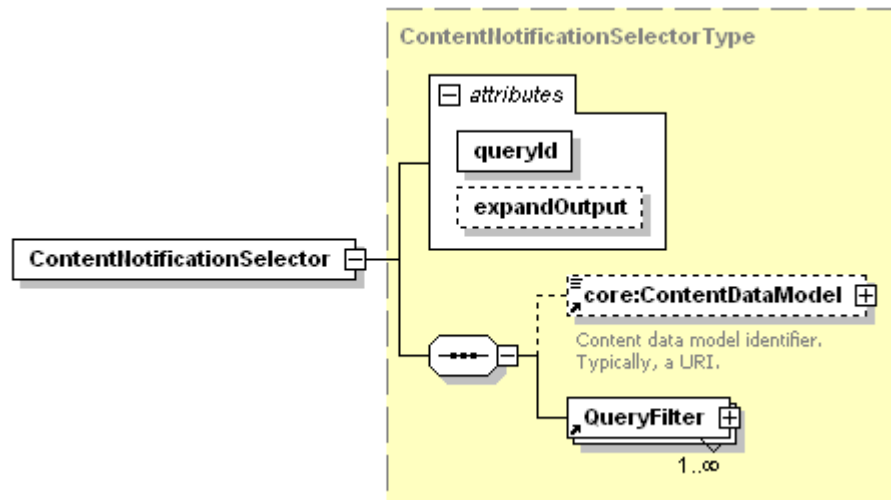


Figure 21 - ContentNotificationSelector XML Schema

The ContentNotificationSelector element contains the following attributes and elements:

@queryId [Required, cis:queryIdAttrType] – The @queryId attribute is generated by the caller and shall be returned to the caller in any resulting ContentNotification messages. See Section 12.1.1 for further details on the cis:queryIdAttrType.

@expandOutput [Optional, cis:expandOutputAttrType] – The @expandOutput attribute instructs a CIS to expand the query output to include the full text of the associated data model. The default value for the @expandOutput attribute, if missing, is 'false'. See section 12.1.9 for further details on the cis:expandOutputAttrType type.

core:ContentDataModel [Optional] – The core:ContentDataModel element contains a reference to the data model that should be used to satisfy the queries. When the core:ContentDataModel element is not present in the ContentNotificationSelector, the CIS shall use the implementations default data model to satisfy the query. See the [SCTE130-2] documentation for details on the core:ContentDataModel element.

QueryFilter [Required] – The element contents of the QueryFilter specify the notification query matching semantics. See section 11.6 for further information on the cis:QueryFilter element.

11.3 ContentQuery

The ContentQuery element contains elements specifying the semantics for a single CIS query match.

The XML schema definition for this element is provided in Figure 22.

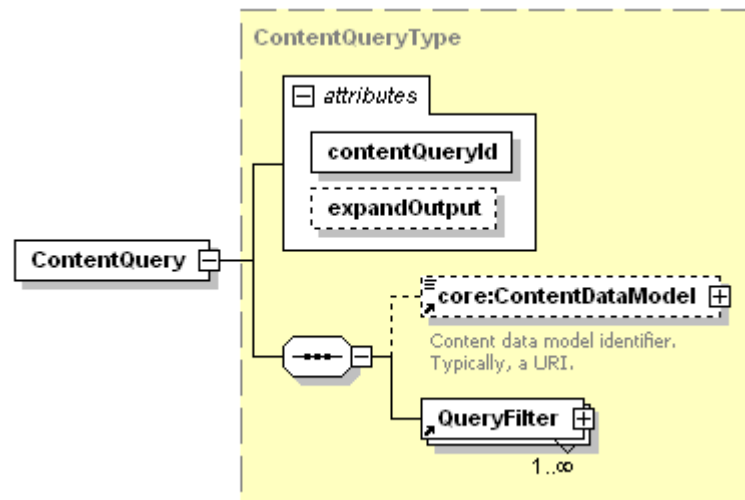


Figure 22 - ContentQuery–XML Schema

The ContentQuery element contains the following attributes and elements:

@contentQueryId [required, cis:queryIdAttrType] – The @contentQueryId attribute is a unique identifier for the ContentQuery. This identifier shall be unique within the scope of the enclosing parent element’s @identity attribute and shall not be empty. See Section 12.1.1 for further details on the cis:queryIdAttrType.

@expandOutput [Optional, cis:expandOutputAttrType] –The @expandOutput attribute indicates to a CIS that the final output should be expanded to the full data model text for basic queries, or the full text generated from the execution of an advanced query. The default value for the @expandOutput attribute, if missing, is ‘false’. See section 12.1.9 for further details on the cis:expandOutputAttrType type.

core:ContentDataModel [Optional] –The core:ContentDataModel element indicates the data model which shall be used to resolve the query. When the core:ContentDataModel element is not present in the ContentQuery element, the CIS shall use the implementation’s default data model to satisfy the query. See the [SCTE130-2] document for details.

QueryFilter [Required] –The QueryFilter element is a container element for FilterElement or AdvancedFilterElement elements. Execution of the individual FilterElement or AdvancedFilterElement elements shall be done in document order. See section 11.6 for further information on the cis:QueryFilter.

11.4 Cursor

A CIS shall support the ability for clients to create and reference static, cursor based data for both basic and advanced queries. Cursors are lists of static asset data with a limited accessibility lifetime. Once established, the data in a cursor cannot change. A cursor’s accessibility end date and time value is initially specified by the client using the CreateCursorRequest message’s @cursorExpires attribute, and finally established by the CIS via the @cursorExpires attribute in the CreateCursorResponse message. The CIS may choose to destroy a cursor and reclaim system resources at any time in order to ensure overall system health.

A CIS has final control over how long a cursor shall remain accessible. If a CIS determines that the requested cursor lifetime end date and time value exceeds some implementation specific duration limitation, then the CIS may choose to return a modified expiration end date and time value in the CreateCursorResponse message’s @cursorExpires attribute. Otherwise, a CIS shall return and use the CIS client’s initially specified @cursorExpires value.

Upon receipt of a cursor attributed CreateCursorRequest, a CIS shall create a new list of static asset data and commence a countdown on the expiration duration for the cursor. If an existing cursor with the same @cursorId value already exists, the CIS shall generate an error and return the error 4002 (Cursor Already Exists) in the core:StatusCode element of the CreateCursorResponse. See Appendix A, for additional details.

The XML schema for the Cursor element is as follows in Figure 23.

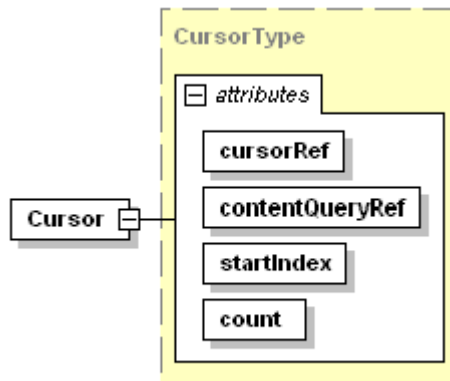


Figure 23 - Cursor–XML Schema

The Cursor element defines the following attributes:

@cursorRef [Required, cis:cursorIdRefAttrType] – The @cursorRef a reference to the @cursorId originally created as a result of a successful call to CreateCursorRequest. See Section 12.1.3 for additional information on the cis:cursorIdRefAttrType type.

@contentQueryRef [Required, cis:queryIdRefAttrType] – The @contentQueryRef attribute refers to the original @contentQueryId contained in the ContentQuery element of the CreateCursorRequest message. This attribute shall be returned to the client in the @contentQueryRef attribute of the ContentQueryResult element. See Section 12.1.2 for further information on the cis:queryIdRefAttrType.

@startIndex [Required, xsd:nonNegativeInteger] – The @startIndex attribute indicates the starting index value for the result set contained within the cursor. @startIndex values begin at the number zero (0), with zero representing the first result within the result set.

@count [Required, xsd:nonNegativeInteger] – The @count attribute indicates the item count that should be returned from the result set starting from the point indicated by the @startIndex attribute.

A CIS shall not generate an error if the @count attribute’s value is greater than the delta between the @startIndex and the end of the static data list. A CIS shall successfully return the remaining records in the static data list.

11.5 ContentQueryResult

Query results are sent back to the calling client encapsulated within a ContentQueryResult element. This element contains the immediate result set for the paired query as well as information regarding the size of the data contained in the result set.

The XML schema for ContentQueryResult is as follows in Figure 24.

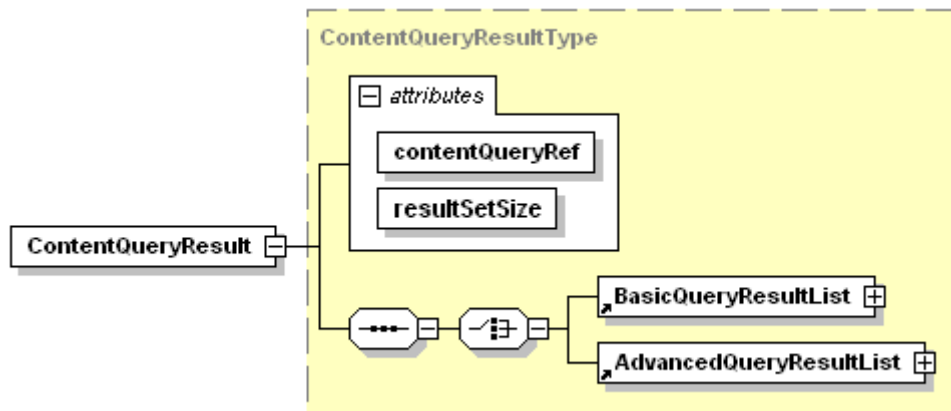


Figure 24 - ContentQueryResult–XML Schema

The ContentQueryResult element contains the following attributes and elements.

@contentQueryRef [Required, cis:queryIdRefAttrType] – The @contentQueryRef attribute contains the content query id information from the original ContentQuery@contentQueryId request message. See Section 12.1.2 for more information on the cis:queryIdRefAttrType.

@resultSetSize [Required, xsd:nonNegativeInteger] – The @resultSetSize attribute contains the total number of records returned in this result set.

BasicQueryResultList [Required on Choice] – The BasicQueryResultList element is a container element for core:Content elements. See Section 11.8 for details.

AdvancedQueryResultList [Required on Choice] – The AdvancedQueryResultList element is a container element for AdvancedQueryResult elements. See Section 11.11 for details on the AdvancedQueryResult element.

11.6 QueryFilter

The QueryFilter element is a container for FilterElement or AdvancedFilterElement elements. The filter element items within a single QueryFilter define a complete query that shall be applied to all assets referenced by a CIS on query execution.

Multiple QueryFilter elements within a single ContentQuery element may act to expand the overall output of the combined queries or may act to reduce the total amount of data returned. This behavior is controlled by the value of the @op attribute.

The XML schema element for the QueryFilter is provided below in Figure 25.

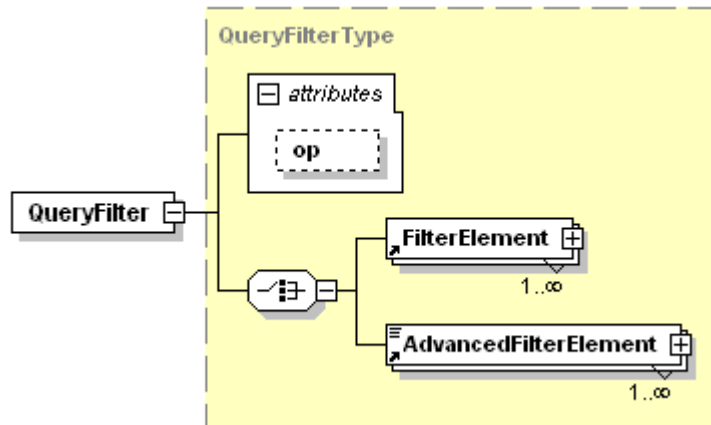


Figure 25 - QueryFilter–XML Schema

The QueryFilter contains the following attributes:

@op [Optional, cis:QueryFilterOpTypeEnumeration] – The @op attribute instructs the CIS how to process this part of the overall query with respect to the cumulative result sets from other QueryFilter elements within the same parent element. The accepted values for the @op attribute are described in Table 10. The default value for the @op attribute is ‘include’.

The QueryFilter may contain one of the two following elements:

FilterElement [Required on Choice] – The FilterElement contains name/value pair attributes that indicate to the CIS which metadata names and values for an asset should be evaluated in the query. The QueryFilter element may contain one or more FilterElements.

When more than one FilterElement is included within a QueryFilter element, a CIS shall combine (AND together) the FilterElements into a single query statement. Only records qualifying against all of the FilterElements contained within a single QueryFilter shall be part of the element’s result set, which is then added to or subtracted from the net result set for the total query. This action depends on the value of the @op attribute of the QueryFilter element.

AdvancedFilterElement [Required on Choice] – The AdvancedFilterElement contains a detailed query language expression that shall be executed against the CIS data model representation of each referenced asset. As an example, the AdvancedFilterElement may contain a complete XPath language query. This query shall be executed against each asset referenced by the CIS and, depending on the value of the @op attribute, the results are added to or subtracted from the net result set.

11.7 FilterElement

Each FilterElement element contains a name and value attribute that form a portion of a CIS query. The @name attributes refers to a particular metadata item located in the CIS supported data model. The @value attribute refers to the value referenced by the named metadata item. The @value attribute shall support regular expression processing as defined in Section 13.1, Regular Expressions and Wildcards.

The XML schema diagram for the FilterElement is as follows in Figure 26.

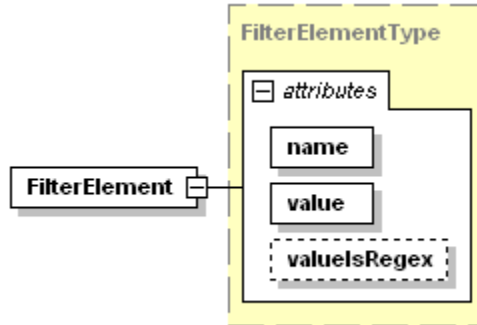


Figure 26 - FilterElement–XML Schema

The FilterElement contains the following attributes:

@name [Required, cis:filterElementNameType] – The @name attribute contains the CIS data model name for a specific metadata item from the supported data model and shall not be empty. See Section 12.1.6 for additional information.

@value [Required, cis:filterElementValueType] – The @value attribute contains the value that shall be matched against the data model metadata item specified by the @name attribute. The @value attribute shall support regular expression processing as defined in Section 13.1. See Section 12.1.7 for additional information on the cis:filterElementValueType.

@valueIsRegex [Optional, xsd:Boolean] – The @valueIsRegex attribute indicates whether the value of the @value attribute shall be treated as a regular expression or not. A value of (true) indicates that the value contained in the @value attribute shall be treated as a regular expression. A value of (false) indicates that the value of the @value attribute shall not be treated as a regular expression and shall match only the exact value (e.g., “abc” shall not match “dabcef”). If the optional @valueIsRegex attribute is missing altogether, the CIS shall default to a value of (false).

11.8 BasicQueryResultList

The BasicQueryResultList is a container of core:Content elements.

The XML schema diagram for the BasicQueryResultList is depicted in Figure 27.

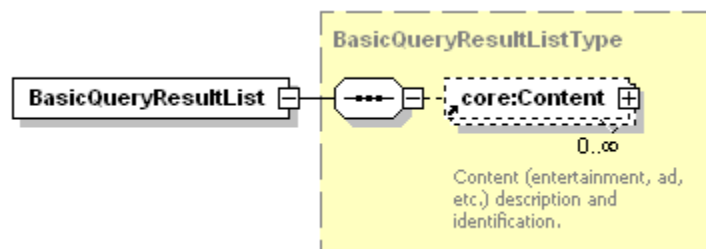


Figure 27 - BasicQueryResultList–XML Schema

For CableLabs specific data models, with the @expandOutput attribute in the ContentQuery message set to *false*, the core:Content element shall contain core:AssetRef elements. When the @expandOutput attribute is set to *true*, the core:Content element shall contain a well formed CableLabs ADI (1.1 or 2.0, depending on the data model) message within the core:Ext element of the core:Content element in addition to the core:AssetRef elements.

For [SCTE118-3] specific data models, with the @expandOutput attribute in the ContentQuery message set to *false*, the core:Content element shall contain core:Program or core:SpotRef elements. When the @expandOutput attribute is set to *true*, the core:Content element will contain [SCTE118-3] Program or Spot elements within the core:Ext element of the core:Content element in addition to the core:Program or core:SpotRef elements within the core:Content element. See Table 8 for details.

core:Content [Optional] – The core:Content element contains information on the referenced asset. This element contains a core:ContentLocation element that may describe the availability of the referenced asset and facilitates specifying an asset’s location. The core:ContentDataLocation element value may be any valid URI. See [SCTE130-2] for additional information.

DataModel	@expandOutput = false	@expandOutput = true
CLADI_1.1	core:AssetRef	core:Ext contains adi:ADI
CLADI_2.0	core:AssetRef	core:Ext contains adi:ADI2
[SCTE118-3] (program)	core:Program	core:Ext contains scte118-3:Program
[SCTE118-3] (spot)	core:SpotRef	core:Ext contains scte118-3:Spot

Table 8: BasicQueryResultList Child Elements

11.9 AdvancedFilterElement

The AdvancedFilterElement contains a query language identifier and a free form query string for use in query language processing.

The XML schema diagram for this message is as follows in Figure 28.

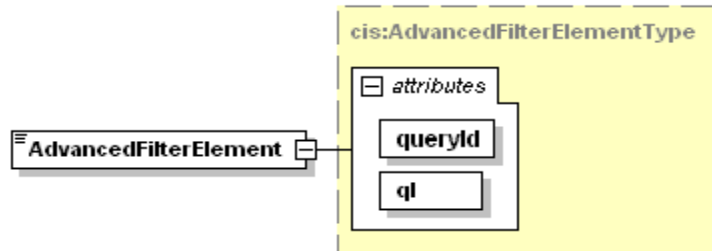


Figure 28 - AdvancedFilterElement–XML Schema

The AdvancedFilterElement contains the following attributes:

@queryId [Required, cis:queryIdAttrType] – The @queryId attribute uniquely identifies the AdvancedFilterElement within the scope of the @identity attribute from the enclosing top level parent element, and shall not be empty.

The @queryId attribute value shall be mapped to the corresponding AdvancedQueryResult @queryRef attribute when the result of the query is sent back to the caller.

See Section 12.1.1 for additional information on the cis:queryIdAttrType.

@ql [Required, cis:queryLanguageAttrType] – The @ql attribute identifies the specific query language engine that shall be used to process the query contained within the AdvancedFilterElement. See Section 12.1.8 for additional information on cis:queryLanguageAttrType.

11.10 AdvancedQueryResultList

The AdvancedQueryResultList element is a container element for AdvancedQueryResult elements.

The XML schema definition for the AdvancedQueryResultList element is as follows in Figure 29.

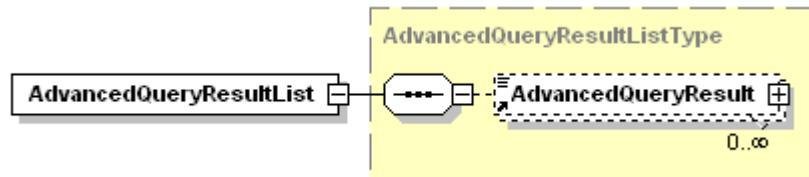


Figure 29 - AdvancedQueryResultList–XML Schema

The AdvancedQueryResultList contains zero (0) or more AdvancedQueryResult elements.

11.11 AdvancedQueryResult

The AdvancedQueryResult element contains unprocessed XML (CDATA) string data.

The XML schema definition for this element follows in Figure 30.

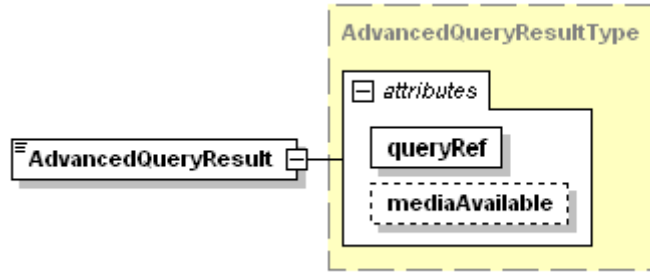


Figure 30 - AdvancedQueryResult - XML Schema

The results from an advanced query, as specified in an AdvancedFilterElement, are returned to the client using an AdvancedQueryResult element. Advanced query results returned to the client shall be returned without intermediate formatting by a CIS (i.e., as is).

When the @expandOutput attribute of the ContentQuery message is set to *false*, the AdvancedQueryResult element shall contain one or more of the elements listed in Table 9.

DataModel	@expandOutput = false
CLADI_1.1	core:AssetRef
CLADI_2.0	core:AssetRef
[SCTE118-3] (program)	core:Program
[SCTE118-3] (spot)	core:SpotRef

Table 9: AdvancedQueryResult CDATA contents

When the @expandOutput attribute of the ContentQuery message is set to *true*, the CDATA section shall contain the raw output generated from the execution of the advanced query.

The AdvancedQueryResult element contains the following attributes.

@queryRef [Required, cis:queryIdRefAttrType] – The @queryRef attribute contains the AdvancedFilterElement @queryId attribute value from the original query. This allows the client to map the result of an advanced query back to an original query source. See Section 12.1.2 for additional details on the cis:queryIdRefAttrType attribute type.

@mediaAvailable [Optional, core:mediaAvailableAttrType] – The @mediaAvailable attribute indicates the availability status of the referenced asset. See [SCTE130-2] for additional information on the core:mediaAvailableType attribute type.

11.12 DataModelList

The DataModelList is a container element for ContentDataModel elements. The ‘default’ data model for a CIS shall be the first core:ContentDataModel element returned in a DataModelList.

The XML schema definition for the DataModelList element follows in Figure 31.

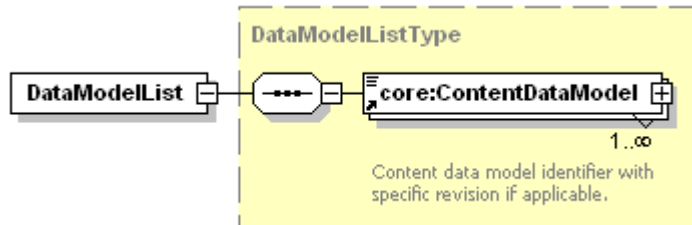


Figure 31 - DataModelList – XML Schema

There are no additional attributes defined on the DataModelList element. The DataModelList element contains the following elements:

ContentDataModel [Required]–The ContentDataModel is a container element for core:ContentDataModel elements. See the [SCTE130-2] documentation for a list of the supported data models.

11.13 AdvancedQueryLanguageList

The AdvancedQueryLanguageList is a container element for AdvancedQueryLanguage elements.

The XML schema definition for the AdvancedQueryLanguageList element is as follows in Figure 32.

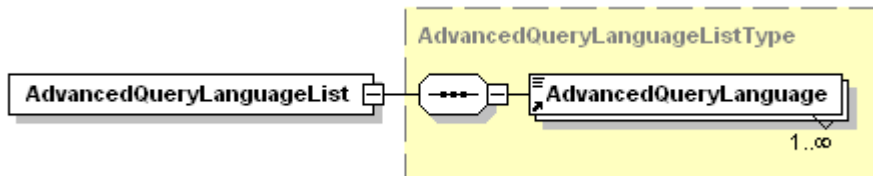


Figure 32 - AdvancedQueryLanguageList – XML Schema

There are no additional attributes defined on the AdvancedQueryLanguageList element. The AdvancedQueryLanguage element contains the following elements:

AdvancedQueryLanguage [Required] – The AdvancedQueryLanguage is a container element for describing advanced query language information AdvancedQueryLanguage.

11.14 AdvancedQueryLanguage

The AdvancedQueryLanguage element is defined in the CIS schema as type core:nonEmptyStringType. This element is used in the context of the ListSupportedFeaturesResponse message in order to return to the caller the types of advanced query languages that are supported by a CIS. See Table 7 for additional information on supported advanced query languages.

The XML schema definition for the AdvancedQueryLanguage element is as follows in Figure 33.

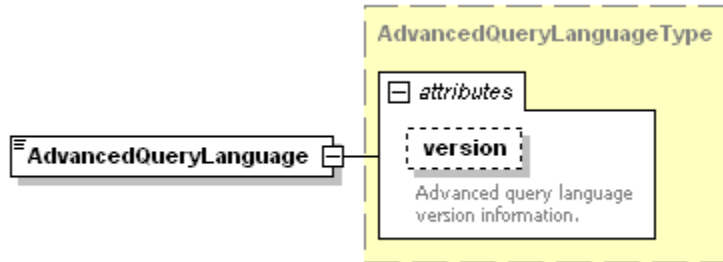


Figure 33 - AdvancedQueryLanguage XML Schema

The AdvancedQueryLanguage element contains the following attributes:

@version [Optional, core:nonEmptyStringType] – The @version attribute carries version information for the specified advanced query language.

12.0 SCTE 130 PART 4 CIS ATTRIBUTE TYPES

The following section defines the SCTE 130 Part 4 CIS attribute types that are used throughout this document.

12.1 Semantic Definitions for SCTE 130 Part 4 CIS Attribute Types

12.1.1 queryIdAttrType Attribute Type

queryIdAttrType [core:nonEmptyStringType] – This attribute type, typically referred to as the @queryId attribute, represents a unique string identifying an individual query issued from a client system and shall not be empty. The queryIdAttrType shall be unique within the scope of the client's @identity attribute.

12.1.2 queryIdRefAttrType Attribute Type

queryIdRefAttrType [cis:queryIdAttrType] – This attribute type, used as the @contentQueryRef attribute, is a reference to an original @contentQueryId attribute and shall not be empty.

12.1.3 cursorIdAttrType Attribute Type

cursorIdAttrType [core:nonEmptyStringType] – This attribute type, typically referred to as the @cursorId or @cursorRef attribute, represents a unique string identifying an individual cursor created by a client system and shall not be empty. The cursorIdAttrType shall be unique within the scope of the client’s @identity attribute.

12.1.4 cursorIdRefAttrType AttributeType

cursorIdRefAttrType [cis:cursorIdAttrType] – This attribute type, used as the @cursorRef attribute, is a reference to an original @cursorId attribute and shall not be empty.

12.1.5 queryFilterOpTypeEnumeration Attribute Type

queryFilterOpTypeEnumeration [core:nonEmptyStringType] – This attribute type, typically referred to as the @op attribute, represents the operation that is to be performed against an overall query result set relative to the items returned in an individual query result set and shall not be empty.

The two possible values for the queryFilterOpTypeEnumeration are as follows in Table 10 and shall appear exactly as they are in this table.

Enumeration Value	Description
include	The include value indicates to the CIS that the results from the execution of this query shall be uniquely appended to the final result set
exclude	The exclude value indicates to the CIS that the results of this query shall be subtracted from the final result set

Table 10: QueryFilterOpTypeEnumeration Values

The default value for the @op attribute is “include”. Records that qualify against multiple queries within the QueryFilter shall be added to the net result set only once.

12.1.6 filterElementNameType Attribute Type

filterElementNameType [core:nonEmptyStringType] – This attribute type, typically referred to as the @name attribute, represents the CIS data model name for a particular attribute or metadata item from the supported data model and shall not be empty.

12.1.7 filterElementValueType Attribute Type

filterElementValueType [core:nonEmptyStringType] – This attribute type, typically referred to as the @value attribute, represents the value that should be checked against the value of the particular data model name or metadata item identified by the @name attribute and shall not be empty. The @value attribute shall support regular expression processing. The set of regular expressions that shall be supported by the CIS is a subset of the set of regular expressions typically supported by most regular expression processing facilities. See Section 13.1, Regular Expressions and Wildcards, for additional information.

12.1.8 queryLanguageAttrType Attribute Type

queryLanguageAttrType [core:nonEmptyStringType] – This attribute type, typically referred to as the @ql attribute, is a text string that identifies the specific query language engine that shall be used to process a query and shall not be empty. The CIS shall support the advanced query languages listed in Table 7, if advanced query language support is to be provided by a CIS implementation

12.1.9 expandOutputAttrType Attribute Type

expandOutputAttrType [xsd:Boolean] – This attribute type, typically referred to as the @expandOutput attribute, is a boolean indication to the CIS to expand query results to include the full object as represented by the selected object model.

13.0 BASIC QUERIES AND REGULAR EXPRESSIONS

The ContentQuery element contains one or more QueryFilter elements. These filter elements contain name/value pairs that shall be applied to the supported CIS data model. As an example, a QueryFilter element may contain a FilterElement containing an @name attribute and a complete or wild-carded @value attribute. For instance:

```
<QueryFilter>  
  <FilterElement name="Provider_ID" value="indemand\.com" valueIsRegex="true" />  
</QueryFilter>
```

Example 1

In Example 1 above, the @name attribute of the FilterElement contains the string "Provider_ID", which maps to the Provider_ID attribute located in the data model. In this example, the data model is the CableLabs ADI 1.1 specification. See the [CLADI1-1] normative reference from [SCTE130-2].

The @value attribute contains the expected value for the @name attribute that satisfies the query. In this example the @value attribute contains the string "indemand\.com". The backslash (\) in this previous example is required to escape the next character which is a regular expression wildcard character. See 13.1 for additional information on regular expressions and wildcarding. Any assets referenced by the CIS within the [CLADI1-1] data model that contain the attribute Provider_ID="indemand.com", will be returned in the result set. The @valueIsRegex attribute indicates that the value of the @value attribute should be treated as a regular expression when processing this FilterElement.

13.1 Regular Expressions and Wildcards

Wildcarding and regular expressions allow the CIS to select a targeted set of assets using a single query. For instance:

```
<QueryFilter>  
  <FilterElement name="Provider_ID" value=".*tv\.com" valueIsRegex="true" />  
</QueryFilter>
```

Example 2

In Example 2 above, the @value attribute of the FilterElement contains the regular expression ".*tv\.com". When executed, this query shall resolve to all assets that contain

the suffix “tv.com”. For instance, the regular expression would find “itv.com”, “mtv.com” and “tv.com”.

Regular expressions are only allowed in the @value attribute of the FilterElement and are only considered regular expression if the @valueIsRegex attribute is set to (true).

The set of regular expressions that shall be supported by the CIS is a subset of the regular expressions supported by most regular expression processing engines. Table 11 lists each regular expression and a short description of what each RE (regular expression) does.

Regular Expression	Description
^ (Carat)	Match expression at the start of a line, as in ^A
\$ (Dollar)	Match expression at the end of a line, as in A\$
\ (Back slash)	Turn off the special meaning of the next character, as in \^
[] (Brackets)	Match any one of the enclosed characters, as in [aeiou]. Use hyphen “-“ for range, as in [0-9]. Lexigraphical ordering is alphabetic and numeric. Ranges are limited to [a-z] [A-Z] and [0-9]
[^]	Match any one character except those enclosed in [], as in [^0-9]
. (Period)	Match a single character of any value, except end of line
* (Asterisk)	Match zero or more of the preceding character or expression
+ (Plus)	Match one or more of the previous group or character
? (Question)	Match 0 or 1 of previous group or characters
{X,Y}	Match X to Y occurrences of the preceding
{X}	Match exactly X occurrences of the preceding
{X,}	Match X or more occurrences of the preceding
	Alternation. Allow for two regular expressions forming an OR evaluation
()	Regular expression grouping

Table 11: CIS Regular Expressions

Table 12 contains some example expressions and results based on the regular expressions defined in Table 11.

Expression	Results
“mtv”	Search for the string “mtv” within the source value
“^mtv”	Search for the string “mtv” at the beginning of the source value
“mtv\$”	Search for the string “mtv” at the end of the source value
“^mtv\$”	Search for the string “mtv” as the only content within the source value
“\^s”	Search for a value that contains the carat (^) symbol and is followed by the letter “s”
“[Mm]tv”	Search for mtv with either an upper case M or lower case m
“B[oO][bB]”	Search for BOB, BOb, Bob or BoB
“^\$”	Search for a value with no content
“[0-9][0-9]”	Search for pairs of numeric digits

“[a-zA-Z]”	Search for any value with at least one letter
“[^a-zA-Z0-9]”	Search for any value not a letter or number
“[0-9]{3}-[0-9]{4}”	Search for phone number like values: 555-1212
“^.\$”	Search for values with exactly one character
“”mtv””	Search for mtv within double quotes
“”*mtv”*”	Search for mtv with or without quotes
“^\.”	Search for any value that starts with a period “.”
“^\.[a-z][a-z]”	Search for any value starting with a period “.” and followed by two lower case letters
“^abc ^xyz	Search for any value starting with the letters a,b,c OR x,y,z
(ab)+\$	Match the group (ab) 1 or more times at the end of the line

Table 12: Regular Expression Examples

**A. APPENDIX A (NORMATIVE) STATUSCODE ELEMENT @DETAIL
ATTRIBUTE VALUES**

Table 13 contains SCTE 130 Part 4 applicable status codes for the @detail attribute.

@Detail Value	Name	Description
4001	Cursor Undefined	The requested cursor does not exist within the CIS.
4002	Cursor Already Exists	The requested cursor already exists within the CIS.

Table 13: StatusCode Details

Table 14 contains descriptive references to the SCTE130-2 and Part-4 @detail values that may be returned in CIS top level messages.

ContentNotificationRegistrationResponse = CNRR
 ContentQueryResponse = CQR
 CreateCursorResponse = CCR
 CancelCursorResponse = CNCR
 ContentNotificationDeregisterResponse = CNDR
 ListContentNotificationRegistrationResponse = LCNRR
 DeregistrationNotification = DN
 ContentNotification = CN
 ListSupportedFeaturesResponse = LSFR

Description	C N R R	C Q R	C C R	C N C R	C N D R	L C N R R	D N	C N	L S F R
Incomplete message	✓	✓	✓	✓	✓	✓			✓
Message validation failed	✓	✓	✓	✓	✓	✓			✓
Registration overlap	✓								
Query failed		✓						✓	
Ambiguous details	✓	✓	✓		✓	✓			✓
Unsupported protocol	✓								
Network address does not exist	✓						✓		
Network address/port in use	✓								
Duplicate message id	✓	✓	✓	✓	✓	✓			✓
Network connection lost	✓								
Resource not found	✓	✓		✓	✓	✓		✓	✓
Not Supported	✓	✓	✓						
Not Authorized	✓	✓	✓	✓	✓	✓			
Unknown message reference	✓	✓	✓	✓	✓	✓			✓
Resend forced abandonment	✓	✓	✓	✓	✓	✓			✓
Out of Resources	✓	✓	✓		✓	✓			✓
Timeout	✓	✓	✓	✓	✓	✓			✓
Cursor Undefined		✓		✓					
Cursor Already Exists			✓						
General error	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 14: Part 2 and Part 4 StatusCode Detail Usage

B. APPENDIX B (INFORMATIVE) COMPLEX QUERIES AND EXPANDED OUTPUT

B.1 Multiple Filter Elements

Multiple FilterElement elements may be contained within a single QueryFilter. This allows for a more detailed query to be formed. The example below illustrates this feature:

```
<QueryFilter>
  <FilterElement name="Provider_ID" value="indemand.com" valueIsRegex="false"/>
  <FilterElement name="Asset_ID" value="XXXX00000000000001" valueIsRegex="false"/>
</QueryFilter>
```

Example 3

In Example 3 above, the result of the query is a specific asset identified by a particular providerId and assetId. The operator combining two or more FilterElements within a QueryFilter is the AND operator. Both FilterElements in Example 3 are combined together to form a single query.

In Example 3 above, in order for a single record to qualify for this query, the Provider_ID must be "indemand.com" and the Asset_ID must be "XXXX000000000001", otherwise the record will be rejected. The @valueIsRegex attribute of the both FilterElements is set to (false) in Example 3. This indicates that regular expression processing will not be executed on the contents of the @value attributes. This is the default behavior if the @valueIsRegex attribute is omitted from the FilterElement altogether.

The ContentQuery element may contain one or more QueryFilter elements. The @op attribute indicates to the CIS how the individual result sets from a QueryFilter should be applied to the overall result set.

```
<ContentQuery contentQueryId="1">
  <core:ContentDataModel type="CLADI_1.1">URI</core:ContentDataModel>
  <QueryFilter>
    <FilterElement name="Provider_ID" value="indemand.com" />
    <FilterElement name="Asset_ID" value="XXXX000000000001" />
  </QueryFilter>
  <QueryFilter op="include">
    <FilterElement name="Provider_ID" value="indemand.com" />
    <FilterElement name="Asset_ID" value="XXXX000000000002" />
  </QueryFilter>
</ContentQuery>
```

Example 4

The results of this query may include up to two core:AssetRef elements, one for each asset located in the query.

In Example 5 below, an operator has been added to the second QueryFilter to act as a negating agent in order to limit the output:

```

<ContentQuery contentQueryId="1">
  <core:ContentDataModel type="CLADI_1.1">URI</core:ContentDataModel>
  <QueryFilter>
    <FilterElement name="Provider_ID" value="indemand.com" />
    <FilterElement name="Asset_ID" value="^XXXX[0-9]*" valueIsRegex="true"/>
  </QueryFilter>
  <QueryFilter op="exclude">
    <FilterElement name="Provider_ID" value="indemand.com" />
    <FilterElement name="Asset_ID" value="XXXX0000000000001" />
  </QueryFilter>
</ContentQuery>

```

Example 5

In Example 5 above, the first QueryFilter matches all indemand.com assets starting with an AssetId of "XXXX". The second QueryFilter tells the CIS to **exclude** the specific asset with assetId XXXX0000000000001 from the net result set.

B.2 Expanded Output

Query results shall be returned in the form of lists of assets or expanded results. Expanded results shall contain all of the metadata associated with the selected record(s) and defined by the selected data model.

```

<ContentQuery expandOutput="true" contentQueryId="1">
  <core:ContentDataModel type="CLADI_1.1">URI</core:ContentDataModel>
  <QueryFilter>
    <FilterElement name="Provider_ID" value="indemand.com" />
    <FilterElement name="Asset_ID" value="XXXX0000000000001" />
  </QueryFilter>
</ContentQuery>

```

Example 6

The query in Example 6 will result in the selection of a single record on the CIS. The results of this query shall be returned to the caller in the form of the full text of the CLADI_1.1 data model.

The resulting output from the query provided in Example 6 follows in Example 7:

```
<ContentQueryResult resultSetSize="1" contentQueryRef="1">
  <BasicQueryResultList>
    <core:Content>
      <core:ContentLocation mediaAvailable="true"></core:ContentLocation>
      <core:Ext>
        <ADI>
          ....
        </ADI>
      </core:Ext>
    </core:Content>
  </BasicQueryResultList>
</ContentQueryResult>
```

Example 7

Note that in Example 7 the entire output of the ADI record has been shortened to simply `<ADI>...</ADI>` for clarity and that the `core:ContentLocation` attribute indicates that the media is available, but no location URI has been provided.

In the next example, the `@expandOutput` attribute is set to `false` (the default) and the results of the query are returned as a list. Note that the `@value` attribute contains a regular expression that allows for a broad selection of assets available from a specific provider.

```
<ContentQuery expandOutput="false" contentQueryId="1">
  <core:ContentDataModel type="CLADI_1.1">URI</core:ContentDataModel>
  <QueryFilter>
    <FilterElement name="Provider_ID" value="indemand.com" />
    <FilterElement name="Asset_ID" value="^XXXX[0-9]*" valueIsRegex="true" />
  </QueryFilter>
</ContentQuery>
```

Example 8

The results of the query from Example 8 are illustrated below in Example 9.

```
<ContentQueryResult resultSetSize="2" contentQueryRef="1">
  <BasicQueryResultList>
    <core:Content>
      <core:AssetRef providerID="indemand.com" assetID="XXXX00000000000001" />
      <core:ContentLocation mediaAvailable="true">
        file://ContentStore/file1.mpg
      </core:ContentLocation>
    </core:Content>
    <core:Content>
      <core:AssetRef providerID="indemand.com" assetID="XXXX00000000000002" />
      <core:ContentLocation mediaAvailable="true">
        file://ContentStore/file2.mpg
      </core:ContentLocation>
    </core:Content>
  </BasicQueryResultList>
</ContentQueryResult>
```

Example 9

Note that in example 9, the core:ContentLocation element indicates that all of the assets are available and a URI has been supplied indicating their location.

C. APPENDIX C (INFORMATIVE) ADVANCED QUERIES

C.1 Advanced Queries

The CIS should support advanced query mechanisms. In order to support advanced queries, the QueryFilter element accepts the inclusion of one or more AdvancedFilterElement elements. This element contains a string schema type with a CDATA section containing the raw statement of the advanced query.

The results of an advanced query shall be returned in an AdvancedQueryResult element contained within a ContentQueryResult. The following example illustrates an advanced query construct.

```
<QueryFilter>
  <AdvancedFilterElement queryId="id-123" ql="XPath">
    <![CDATA[ /ADI/Metadata/AMS ]]>
  </AdvancedFilterElement>
</QueryFilter>
```

Example 10

In Example 10 above, the query language @ql is specified as XPath. The actual XPath expression is contained within the CDATA section of the AdvancedFilterElement.

Note that the @queryId attribute in the AdvancedQueryElement is reflected back to the caller in the @queryRef attribute of the AdvancedQueryResult element. The results of the previous query are illustrated below in Example 11:

```
<ContentQueryResult resultSetSize="1" contentQueryRef="1">
  <AdvancedQueryResultList>
    <AdvancedQueryResult queryRef="id-123" mediaAvailable="true">
      <![CDATA[
        <ADI>
          <Metadata>
            <AMS . . . />
          </Metadata>
        </ADI>
      ]]>
    </AdvancedQueryResult>
  </AdvancedQueryResultList>
</ContentQueryResult>
```

Example 11

D. APPENDIX D (INFORMATIVE) CURSORS

D.1 Creating Cursors

Cursors are created by using the CreateCursorRequest message. An example of this request can be seen in Example 12.

```
<CreateCursorRequest messageId="1" version="1.1" system="CIS" cursorId="cursor-1"
cursorExpires="2007-08-10T12:00:00Z" identity="40DA910E-01AF-5050-C7EA-
5D7B4A475311">
  <ContentQuery contentQueryId="query-1" expandOutput="false">
    <core:ContentDataModel type="CLADI_11">URI</core:ContentDataModel>
    <QueryFilter>
      <FilterElement name="Provider_ID" value=".*\\.com"
valueIsRegex="true" />
    </QueryFilter>
  </ContentQuery>
</CreateCursorRequest>
```

Example 12

In Example 12 above, the CreateCursorRequest message contains a @cursorId attribute and a @cursorExpires attribute. The CIS will associate the @cursorId value with the physical cursor instance so that subsequent calls from the client can refer back to the same physical cursor instance.

The @cursorExpires attribute contains an core:dateTimeTimezoneType, which is a request to the CIS to give the new cursor construct a particular life span. The CIS may choose to ignore the @cursorExpires date request and substitute an implementation specific cursor end date and time value instead. When this substitution occurs, the new cursor expiration end date and time value is returned to the caller in the CreateCursorResponse message.

The CIS, upon receipt of the CreateCursorRequest, shall create the cursor construct, accept or adjust the cursor duration and then execute the supplied ContentQuery in order to fill the cursor. The data within the cursor shall remain static for the life time of the cursor.

Once the cursor has been constructed, the CIS will respond to the caller with a CreateCursorResponse message. This message will contain a reference to the original CreateCursorRequest message, a result set size, and the final value for the cursor expiration duration.

Example 13 contains an illustration of the CreateCursorResponse message.

```
<CreateCursorResponse messageId="resp-1" version="1.1" system="CIS" messageRef="1"
resultSetSize="100" cursorExpires="2007-08-10T12:00:00.0Z" identity="40DA910E-01AF-
5050-C7EA-5D7B4A475312">
  <core:StatusCode class="0"/>
</CreateCursorResponse>
```

Example 13

Example 13 above, indicates that the creation of the cursor was successful on the CIS and that the cursor information, identified by @cursorId (cursor-1, from the CreateCursorRequest) will be available on the CIS until the end date specified by the @cursorExpires attribute. In this case, the CIS decided that the cursor expiration date and time value was within acceptable bounds and the same value for the expiration was returned to the caller.

The CreateCursorResponse message also indicates the total number of data items contained in the cursor with the @resultSetSize attribute. In this case, 100 data items are contained in the cursor.

D.2 Walking Cursors

Adding cursor information to the ContentQueryRequest message allows for the selection of data with cursor like control over the static assets contained in the cursor.

```
<ContentQueryRequest messageId="2" version="1.1" system="CIS"
identity="40DA910E-01AF-5050-C7EA-5D7B4A475311">
  <Cursor startIndex="0" count="10" cursorRef="cursor-1" contentQueryRef="query-
1"/>
</ContentQueryRequest>
```

Example 14

In Example 14 above, the starting index for the query is zero (0) and the ending count is ten (10) for a total of 10 objects to be returned. Not specifying a @count attribute shall cause the CIS to return all assets from the starting index to the end of the cursor's record list.

Note that the cursor contains a reference to the original contentQueryId (contentQueryRef) and cursorId from the original CreateCursorRequest. (See section 10.15.1).

If the requested cursor has timed out (expired), the ContentQueryResponse will contain a StatusCode indicating a class “Error” and Code (Cursor Undefined). See Appendix A, Section A for additional details.

To continue to iterate over an existing cursor, the @startIndex attribute of the cursor must be modified. Example 15 below, illustrates a continuation of the cursor walk from Example 14.

```
<ContentQueryRequest messageId="2" version="1.1" system="CIS"
  identity="40DA910E-01AF-5050-C7EA-5D7B4A475311">
  <Cursor startIndex="10" count="10" cursorRef="cursor-1" contentQueryRef="query-
  1"/>
</ContentQueryRequest>
```

Example 15

In this example, the @startIndex has been reset to 10 and the @count remains the same. The results from this query will contain cursor data items numbered 10 through 19.

D.3 Canceling Existing Cursors

Once a client has finished working with a cursor, the cursor may be canceled before the duration time of the cursor has expired. Example 16 illustrates a complete CancelCursorRequest message.

```
<CancelCursorRequest messageId="1" version="1.1" system="client1" cursorRef="cursor-1"
  identity="40DA910E-01AF-5050-C7EA-5D7B4A475311">
</CancelCursorRequest>
```

Example 16

Upon receipt of a CancelCursorRequest, the CIS shall remove the requested cursor for the specified system if the cursor exists. Example 17 illustrates the expected CancelCursorResponse message.

```
<CancelCursorResponse messageId="2" version="1.1" system="CIS" messageRef="1"
  identity="40DA910E-01AF-5050-C7EA-5D7B4A475312">
  <core:StatusCode class="0"/>
</CancelCursorResponse>
```

Example 17

In Example 17, the StatusCode element indicates that the requested cursor was successfully removed from the CIS.

If the cursor identified in the `CancelCursorRequest` message has already expired before the `CancelCursorRequest` message arrives, the `StatusCode` in the `CancelCursorResponse` message should not indicate an error.

E. APPENDIX E (INFORMATIVE) COMPLETE MESSAGE EXAMPLES

The following sections contain a selection of complete examples of CIS top level messages.

E.1 List Supported Features Request and Response

The ListSupportedFeaturesRequest is the only service endpoint that is required to be available on the well known address for the CIS. All other service endpoints may also be available on the well known CIS address or available only on other, more specific, endpoint addresses. The ListSupportedFeaturesResponse message may contain a set of core:Callout elements which may include the additional addresses for specific services.

```
<ListSupportedFeaturesRequest messageId="acs-342" system="sys-1" version="1.1"
  identity="40DA910E-01AF-5050-C7EA-5D7B4A475311">
</ListSupportedFeaturesRequest>
```

Example 18

Example 18 contains an example of a ListSupportedFeaturesRequest message.

```
<ListSupportedFeaturesResponse messageId="sca-342" system="sys-cis" version="1.1"
  identity="40DA910E-01AF-5050-C7EA-5D7B4A475312" messageRef="acs-342">
  <core:StatusCode class="0"/>
  <core:Callout>
    <core:Address type="SOAP">http://10.250.30.22/ADSServer</core:Address>
  </core:Callout>
  <DataModelList>
    <core:ContentDataModel type="CLADI_1.1">URI </core:ContentDataModel>
  </DataModelList>
  <AdvancedQueryLanguageList>
    <AdvancedQueryLanguage version="1.0">XPath</AdvancedQueryLanguage>
    <AdvancedQueryLanguage version="1.0">XQuery</AdvancedQueryLanguage>
  </AdvancedQueryLanguageList>
</ListSupportedFeaturesResponse>
```

Example 19

Example 19 contains an example of a ListSupportedFeaturesResponse. The single core:Callout element does not include an @message attribute. This indicates that all CIS service channel endpoints are available through this well known CIS address endpoint. Example 20 contains a ListSupportedFeaturesResponse that does contain core:Callout elements for several specific CIS service channel endpoints.

```
<ListSupportedFeaturesResponse messageId="sca-342" system="sys-cis" version="1.1"
```

```

identity="40DA910E-01AF-5050-C7EA-5D7B4A475312" messageRef="acs-342">
<core:StatusCode class="0"/>
<core:Callout>
  <core:Address type="SOAP">http://10.250.30.22/ADSServer</core:Address>
</core:Callout>
<core:Callout message="ContentNotificationRegistrationRequest">
  <core:Address type="SOAP">http://10.250.30.23/ADSServer</core:Address>
</core:Callout>
<core:Callout message="ContentNotificationDeregisterRequest">
  <core:Address type="SOAP">http://10.250.30.24/ADSServer</core:Address>
</core:Callout>
<DataModelList>
  <core:ContentDataModel type="CLADI_1.1">URI </core:ContentDataModel>
</DataModelList>
<AdvancedQueryLanguageList>
  <AdvancedQueryLanguage version="1.0">XPath</AdvancedQueryLanguage>
  <AdvancedQueryLanguage version="1.0">XQuery</AdvancedQueryLanguage>
</AdvancedQueryLanguageList>
</ListSupportedFeaturesResponse>

```

Example 20

Example 20 contains three core:Callout elements. The first core:Callout element is the default core:Callout element. This element contains the default address for all CIS service channel message endpoints. Two additional core:callout elements in this example indicate that the service channel endpoints for the ContentNotificationRegistrationRequest and ContentNotificationDeregisterRequest messages are located on specific addresses, different from that of the default address(s). All other messages that are not specifically listed in the selection of core:Callout elements shall be available through the default core:Callout address(s). See Table 4 for a list of all CIS service channel message endpoints.

E.2 Content Query Request and Response

The ContentQueryRequest is the workhorse of the CIS system. This message provides clients with an endless number of flexible query alternatives.

Example 21 uses the basic query mechanism to request a complete listing of records from the CIS:

```
<ContentQueryRequest messageId="acs-342" system="sys-1" version="1.1"
identity="40DA910E-01AF-5050-C7EA-5D7B4A475311">
  <ContentQuery contentQueryId="1">
    <core:ContentDataModel type="CLADI_1.1">URI </core:ContentDataModel>
    <QueryFilter>
      <FilterElement name="Provider_ID" value=".*" valueIsRegex="true"/>
    </QueryFilter>
  </ContentQuery>
</ContentQueryRequest>
```

Example 21

The result of the previous query is illustrated in Example 22:

```
<ContentQueryResponse messageId="sca-342" system="sys-cis" version="1.1"
identity="40DA910E-01AF-5050-C7EA-5D7B4A475312" messageRef="acs-342">
  <core:StatusCode class="0"/>
  <ContentQueryResult resultSetSize="100" contentQueryRef="1">
    <BasicQueryResultList>
      <core:Content>
        <core:AssetRef assetID="XXX..." providerID="com.com"/>
        <core:ContentLocation mediaAvailable="true">URI</core:ContentLocation>
      </core:Content>
      ...
    </BasicQueryResultList>
  </ContentQueryResult>
</ContentQueryResponse>
```

Example 22

E.3 Content Notification Registration Request

The following is a typical ContentNotificationRegistrationRequest example. Note that this example includes the callout instruction for future content notification messages:

```
<ContentNotificationRegistrationRequest messageId="acs-342" system="sys-1" version="1.1"
identity="40DA910E-01AF-5050-C7EA-5D7B4A475311">
  <core:Callout message="ContentNotification">
    <core:Address type="SOAP">http://10.250.30.22/ADSServer</core:Address>
  </core:Callout>
  <ContentNotificationSelector queryId="query-1">
    <core:ContentDataModel type="CLADI_1.1">URI </core:ContentDataModel>
    <QueryFilter>
      <FilterElement name="Provider_ID" value=".*" valueIsRegex="true"/>
    </QueryFilter>
  </ContentNotificationSelector>
</ContentNotificationRegistrationRequest>
```

Example 23

Two things of interest in Example 23 are the core:Callout@message attribute and the core:Address@type attribute. The @message attribute indicates to the CIS that messages of type (ContentNotification) should be sent to the specified address. The @type attribute of the core:Address element indicates the type of the service endpoint found on the client side. In this case, the address type is (SOAP) and the supplied address leads to a SOAP endpoint. See [SCTE130-2] for additional information on the core:Address element.

In this next example, the focus of the registration has been limited to two (2) providers:

```
<ContentNotificationRegistrationRequest messageId="acs-342" system="sys-1"
  version="1.1" identity="40DA910E-01AF-5050-C7EA-5D7B4A475311">
  <core:Callout message="ContentNotification">
    <core:Address type="SOAP">http://10.250.30.22/ADSServer</core:Address>
  </core:Callout>
  <ContentNotificationSelector queryId="query-1">
    <core:ContentDataModel type="CLADI_1.1">URI </core:ContentDataModel>
    <QueryFilter>
      <FilterElement name="Provider_ID" value="^ABC\.com$" valueIsRegex="true"/>
    </QueryFilter>
    <QueryFilter>
      <FilterElement name="Provider_ID" value="^XYZ\.com$" valueIsRegex="true"/>
    </QueryFilter>
  </ContentNotificationSelector>
</ContentNotificationRegistrationRequest>
```

Example 24

In Example 24, the only two providers that are of interest are “ABC.com” and “XYZ.com”.

Note that the two FilterElement elements in this example cannot be contained within the same QueryFilter element. FilterElement elements within a single QueryFilter are logically (ANDed) together to form a single query. In this case, no single asset on the CIS could have a Provider_ID of both (ABC.com) and (XYZ.com) at the same time.

E.4 Content Notification

ContentNotification messages are sent to registered clients when the underlying content store has been changed. Changes include the addition of new assets, the deletion of old assets or updates to existing assets.

Assets that have changed in the content store are evaluated against client registration queries. Matches are packaged up into ContentNotification messages and sent to the registered client. Clients shall respond to ContentNotification messages with ContentNotificationAcknowledgement messages.

Example 25 contains a ContentNotification message for a newly provisioned asset.

```
<ContentNotification type="new" messageId="sca-342" system="acs-1" version="1.1"
identity="40DA910E-01AF-5050-C7EA-5D7B4A475311">
  <ContentQueryResult resultSetSize="1" contentQueryRef="1">
    <BasicQueryResultList>
      <core:Content>
        <core:AssetRef assetID="XXXX000000000001" providerID="max.com"/>
        <core:ContentLocation mediaAvailable="true"></core:ContentLocation>
      </core:Content>
    </BasicQueryResultList>
  </ContentQueryResult>
</ContentNotification>
```

Example 25

The @type attribute of the ContentNotification message is set to (new) indicating that the asset listed in the BasicQueryResultList element was recently provisioned on the CIS.

In order for the consumer of the previous ContentNotification message to retrieve the complete set of metadata for this new asset, a ContentQueryRequest query must be issued for the specific asset, and the @expandOutput attribute must be set to true.

```
<ContentQueryRequest messageId="acs-342" system="sys-1" version="1.1"
identity="40DA910E-01AF-5050-C7EA-5D7B4A475311">
  <ContentQuery expandOutput="true" contentQueryId="1">
    <core:ContentDataModel type="CLADI_1.1">URI</core:ContentDataModel>
    <QueryFilter>
      <FilterElement name="Provider_ID" value="max.com" />
      <FilterElement name="Asset_ID" value="XXXX000000000001" />
    </QueryFilter>
  </ContentQuery>
</ContentQueryRequest>
```

Example 26

In Example 26, the @expandOutput attribute of the ContentQuery element has been set to true. This will result in the CIS sending back to the caller all of the metadata for the requested asset based on the supported data model. In this case the response includes some part of the CLADI 1.1 package specification. The response is illustrated in Example 27.

```

<ContentQueryResponse messageId="sca-342" system="acs-1" version="1.1"
  identity="40DA910E-01AF-5050-C7EA-5D7B4A475312" messageRef="acs-342">
  <core:StatusCode class="0"/>
  <ContentQueryResult resultSetSize="1" contentQueryRef="1">
    <BasicQueryResultList>
      <core:Content>
        <core:ContentLocation mediaAvailable="true">URI</core:ContentLocation>
        <core:Ext>
          <ADI>
            <Metadata> <AMS Asset_Name="..."/> ...</Metadata>
          </ADI>
        </core:Ext>
      </core:Content>
    </BasicQueryResultList>
  </ContentQueryResult>
</ContentQueryResponse>

```

Example 27

F. APPENDIX F (NORMATIVE) WSDL

The WSDL (Web Services Definition Language) document for SCTE 130 Part 4 contains (2) port sections within a single WSDL document. This separation of port definitions within a single WSDL document allows for the separation of functionality between client and server side endpoints.

F.1 WSDL Target Namespace URI Format

See [SCTE 130-7] for specifics on the correct format for WSDL document target namespace URIs.

Example 28 illustrates a properly formatted WSDL target namespace URI for SCTE 130 Part 4.

```
http://www.scte.org/wsdl/130-4/2008a/cis
```

Example 28: Part-4 WSDL Target Namespace URI

F.2 WSDL Description

The diagram in Figure 34 contains the complete WSDL mapping for the SCTE 130 Part 4 CIS interface.

The `wsdl:portType` section of the CIS WSDL document contains the following service definitions, binding type and input/output parameter mappings:

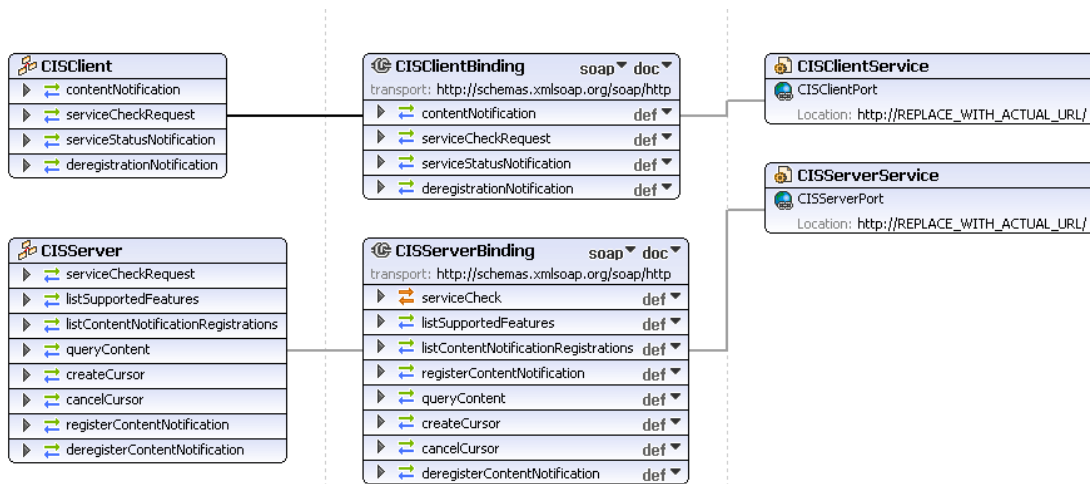


Figure 34: CIS WSDL Document

The `wsdl:portType` sections for the CIS WSDL document contain the following service definitions, binding type and input/output parameter mappings:

contentNotification [document/literal] – Service endpoint for `cis:ContentNotification` message processing.

- Input element type: `cis:ContentNotification`
- Output element type: `cis:ContentNotificationAcknowledgement`

serviceCheckRequest [document/literal] – Service endpoint for `core:ServiceCheckRequest` message processing.

- Input element type: `core:ServiceCheckRequest`

- Output element type: core:ServiceCheckResponse

This service endpoint is available on both the client port type section as well as the server port type section.

serviceStatusNotification [document/literal] – Service endpoint for core:ServiceStatusNotification message processing.

- Input element type: core:ServiceStatusNotification
- Output element type: core:ServiceStatusAcknowledgement

deregistrationNotification [document/literal] – Service endpoint for cis:DeregistrationNotification message processing.

- Input element type: cis:DeregistrationNotification
- Output element type: cis:DeregistrationAcknowledgement

listSupportedFeaturesRequest [document/literal] – Service endpoint for cis:ListSupportedFeaturesRequest message processing.

- Input element type: cis:ListSupportedFeaturesRequest
- Output element type: cis:ListSupportedFeaturesResponse

listContentNotificationRegistrations [document/literal] – Service endpoint for cis:ListContentNotificationRegistrationRequest message processing.

- Input element type: cis:ListContentNotificationRegistrationRequest
- Output element type: cis:ListContentNotificationRegistrationResponse

queryContent [document/literal] – Service endpoint for cis:ContentQueryRequest message processing.

- Input element type: cis:ContentQueryRequest
- Output element type: cis:ContentQueryResponse

createCursor [document/literal] – Service endpoint for cis:CreateCursorRequest message processing.

- Input element type: cis:CreateCursorRequest
- Output element type: cis:CreateCursorResponse

cancelCursor [document/literal] – Service endpoint for cis:CancelCursorRequest message processing.

- Input element type: cis:CancelCursorRequest
- Output element type: cis:CancelCursorResponse

registerContentNotification [document/literal] – Service endpoint for cis:ContentNotificationRegistrationRequest message processing.

- Input element type: cis: ContentNotificationRegistrationRequest
- Output element type: cis: ContentNotificationRegistrationResponse

deregisterContentNotification [document/literal] – Service endpoint for cis: ContentNotificationDeregisterRequest message processing.

- Input element type: cis: ContentNotificationDeregisterRequest
- Output element type: cis: ContentNotificationDeregisterResponse