Errata Notice on Schema Locations

March 17, 2022

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

To find such schemas:

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

The schema will be listed within the Supporting Documentation section.

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

SCTE · ISBE
STANDARDS

Digital Video Subcommittee

AMERICAN NATIONAL STANDARD

ANSI/SCTE 130-8 2020

Digital Program Insertion–Advertising Systems Interfaces Part 8–General Information Service (GIS)

NOTICE

TABLE OF CONTENTS

List of Figures

Title **Page Number**

List of Tables

List of Examples

Digital Program Insertion—Advertising Systems Interfaces Part 8—General Information Service (GIS)

1.0 SCOPE

This document, SCTE 130 Part 8, describes the Digital Program Insertion Advertising Systems Interfaces' General Information Service (GIS) 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]	ANSI/SCTE 130-2 2020 Digital Program Insertion—Advertising Systems Interfaces Part 2—Core Data Elements
[SCTE130-7]	ANSI/SCTE 130-7 2015: Digital Program Insertion—Advertising Systems Interfaces Part 7—Message Transport
[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
[W3C-XSD]	XML Schema Part 1: Structures Second Edition

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

All normative references found in [SCTE130-7] are included and apply to this document. See [SCTE130-7] for additional information.

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: Digital Program Insertion—Advertising Systems Interfaces Part 1—Overview

3.0 COMPLIANCE NOTATION

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

4.0 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.

Advanced Query: The "Advanced Query" interface defined by SCTE 130 Part 8, this document, permits the consumer of a logical service implementation derived from SCTE 130 Part 8 to use an advanced query language to formulate queries against a logical service's data model.

AdvancedQueryLanguage: As used in this document, the term "Advanced Query Language" refers to any language used to formulate queries against a logical service's data model using the advanced query interface. XPath and XQuery are each examples of an advanced query language.

AdvancedQueryFilter: An "Advanced Query Filter" is a collection of free form data items that individually represent complete query terms for a given query language. The individual terms are additively applied (ANDed) together during a query operation against a specific service data model, which results in the identification of a collection of objects contained within the data store.

Basic Query: To obtain information from a logical service implementation derived from SCTE 130 Part 8, this document, a logical service consumer issues a "query" against the data. The "Basic Query" interface is based on an exchange of name/value pairs, referred to as qualifiers, and requires no specialized knowledge of advanced query languages such as XQuery.

BasicQueryFilter: A “Basic Query Filter” is a collection of name and value pairs additively applied (ANDed) together during a basic query operation against a specific service data model, which results in the identification of a collection of objects contained within the data store.

Cursor: A temporary construct containing static data. Consumers of logical services implementing the GIS interface *may* create and access cursor information using the standard query mechanisms described in this document.

Data Model: A Data Model is a formal view of the data items contained in an information store to which an information service implementing this standard will provide access and is specified for purposes of formulating and executing queries against the information store's data. This standard specifies one data model that *may* be used for querying a logical service's information store with this standard's "Basic Query" interface. More complex data models *may* be specified independently of this standard and *may* be queried with this standard's "Advanced Query" interface. In this latter case, the mechanisms by which a consumer incorporates a data model specification so that meaningful queries *may* be issued against it are outside the scope of this standard.

Default Data Model: The data model used when no specific data model is provided. The default data model typically occupies the first location in a data model sequence (i.e., a list of data models).

Qualifier: A “Qualifier” is a name/value pair used to describe one characteristic of an object in a logical service's basic query data model. For instance, <Qualifier name=”Age” value=”30to40”/> is an example of a qualifier where "Age" is the characteristic’s name and "30to40" is the characteristic’s value.

QualifierSet: A “QualifierSet” is a complete set of Qualifier elements that describe an object in a logical service's basic query data model.

UniqueQualifier: A “Unique Qualifer” is a set of one or more Qualifier elements that – taken together – uniquely identify an object in a logical service's basic query data model. To "uniquely identify" an object means that no other object in the data store has the same UniqueQualifer. However, an object *may* have more than one UniqueQualifier.

UniqueQualifierDeclaration: The “Unique Qualifier Declaration” defines the set of Qualifier element characteristic name identifiers comprising a unique qualifier. When each specified Qualifier element’s named characteristic identifier is paired with a value (i.e., a name/value pair which is a Qualifier element), the result is a service data model UniqueQualifier.

5.0 ABBREVIATIONS

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

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

6.0 GENERAL INFORMATION SERVICE OVERVIEW

Several logical services defined by SCTE 130 provide registration, query and notification services to service consumers.

Since the semantic elements of a logical service interface providing these functions are common to more than one logical service, the specification of these common semantic elements has been factored out into this separate General Information Service (GIS) specification.

For example, the Placement Opportunity Information Service (POIS) and the Subscriber Information Service (SIS) specifications *may* incorporate the registration, query and notification components specified herein.

Figure 1 provides an abstract representation of the GIS interface that *may* be implemented by an SCTE 130 logical service.

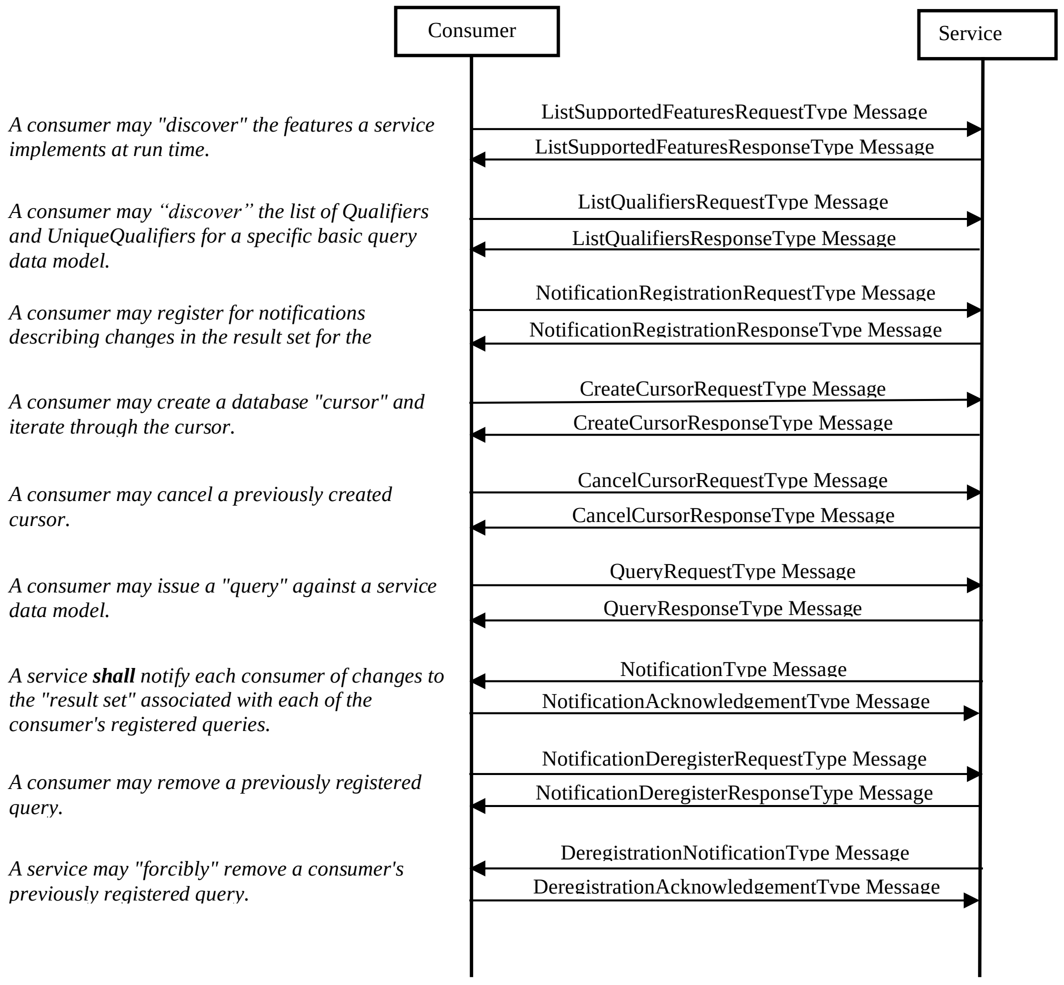

Consumer
Service
A consumer may "discover" the features a service implements at run time.
ListSupportedFeaturesRequestType Message
ListSupportedFeaturesResponseType Message
A consumer may "discover" the list of Qualifiers and UniqueQualifiers for a specific basic query data model.
ListQualifiersRequestType Message
ListQualifiersResponseType Message
A consumer may register for notifications describing changes in the result set for the
NotificationRegistrationRequestType Message
NotificationRegistrationResponseType Message
A consumer may create a database "cursor" and iterate through the cursor.
CreateCursorRequestType Message
CreateCursorResponseType Message
A consumer may cancel a previously created cursor.
CancelCursorRequestType Message
CancelCursorResponseType Message
A consumer may issue a "query" against a service data model.
QueryRequestType Message
QueryResponseType Message
A service shall notify each consumer of changes to the "result set" associated with each of the consumer's registered queries.
NotificationType Message
NotificationAcknowledgementType Message
A consumer may remove a previously registered query.
NotificationDeregisterRequestType Message
NotificationDeregisterResponseType Message
A service may "forcibly" remove a consumer's previously registered query.
DeregistrationNotificationType Message
DeregistrationAcknowledgementType Message

Figure 1. General Information Service Interface

The GIS interface *may* be implemented by a logical service that provides data to other logical services. This specification describes the messaging that *may* be used by a consumer to retrieve data by querying one or more data models supported by a logical service.

Since a logical service's data model *may* be an abstract representation of a large data store, a consumer *may* optionally need to locally cache data for performance optimization. The GIS specification provides support for consumer local data caching by allowing the consumer to "register" notification queries with a logical service. In this case, the result set for each query ***shall*** be returned to the consumer via "notification" messages reflecting differences (add, delete, or modification). Changes in the logical service's underlying data store while the registered query is "active" ***shall*** result in notification messages whose content reflects the changes.

Two kinds of query interfaces *may* be used to discover information about a logical services' data – a "basic query" and an "advanced query".

Basic queries leverage a limited name/value regular expression grammar. No specialized knowledge of advanced query languages is needed when formulating a basic query. An advanced query is formulated using XQuery [XQuery] or XPath [XPath] or any other query language.

A logical service *may* choose to implement either the basic query interface or the advanced query interface or both. See Section 12.3 for additional information on advanced query language support and Section 12.0 for additional information on basic and advanced query mechanisms.

Logical services implementing the interface described herein *may* support more than one service data model and each service data model *may* be queried with either the basic query mechanism or the advanced query mechanism or both. When more than one service data model is supported, the first service data model listed in the ListSupportedFeaturesResponse message type is referred to as the default data model (i.e., the first ServiceDataModelProfile element is distinguished as the default service data model). The default data model ***shall*** be used when no service data model is specified.

A service data model which *may* be queried using the basic query mechanism ***shall*** be constrained to describing a set of objects decorated with one or more name/value pairs – called “Qualifiers”. An additional constraint on the basic query data model is each object in the data model ***shall*** be uniquely identifiable by at least one subset of its name/value pairs – called the “UniqueQualifer”. The unique qualifier composition is described using the UniqueQualifierDeclaration element which provides the characteristic name identifier of the name/value pair using a sequence of QualifierDeclaration elements. See Section 12.27 and Section 12.19 for additional information on the UniqueQualifierDeclaration and QualifierDeclaration elements respectively.

As a concrete example of a service data model that could be easily queried using the basic query interface, consider a Subscriber Information Service (SIS) providing information to other logical services about subscribers. In this case, data representing each subscriber might be comprised of name/value pairs representing demographic information (such as age, income and preferences), location information (such as zip code, service group and street address), and equipment information (such as set-top type and MACAddress). The UniqueQualifier element for a subscriber might be comprised of one Qualifier element – the subscriber’s MACAddress.

A service data model *may* have more than unique qualifier (i.e., more than one UniqueQualifierDeclaration element). When a service data model has more than one UniqueQualifierDeclaration element, each UniqueQualifierDeclaration element ***shall*** have a @uniqueQualifierName attribute and no two UniqueQualifiersDeclarations ***shall*** use the same string for the @uniqueQualifierName attribute. Furthermore, the first UniqueQualifierDeclaration element listed in the BasicQueryDataModelDescription element ***shall*** be referred to as the default unique qualifier declaration (or default unique qualifier). The default unique qualifier declaration ***shall*** be used when no specific unique qualifier declaration is referenced. See Section 12.27 and Section 12.19 for additional information on the UniqueQualifierDeclaration and QualifierDeclaration elements respectively.

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 12.11 for additional information on cursors.

The GIS specification includes a notification model with associated notification management functions such as registration, deregistration and listing of active registrations.

The GIS specification does not restrict the number of data stores a logical service *may* use.

7.0 NOTATIONAL CONVENTIONS

7.1 Normative XML Schema

See [SCTE130-2] for information.

7.2 Document Conventions

SCTE 130 Part 8 employs the same document conventions as [SCTE130-2]. Refer to [SCTE130-2] for an explanation of document conventions. For example, the XML schema illustration is explained there.

This specification utilizes XML substitution groups for additional extensibility. XML substitution groups designate elements as substitutes for other element declarations without changing the original schema documents. Within this document, substitutable elements are graphically identified using the following illustrative technique.

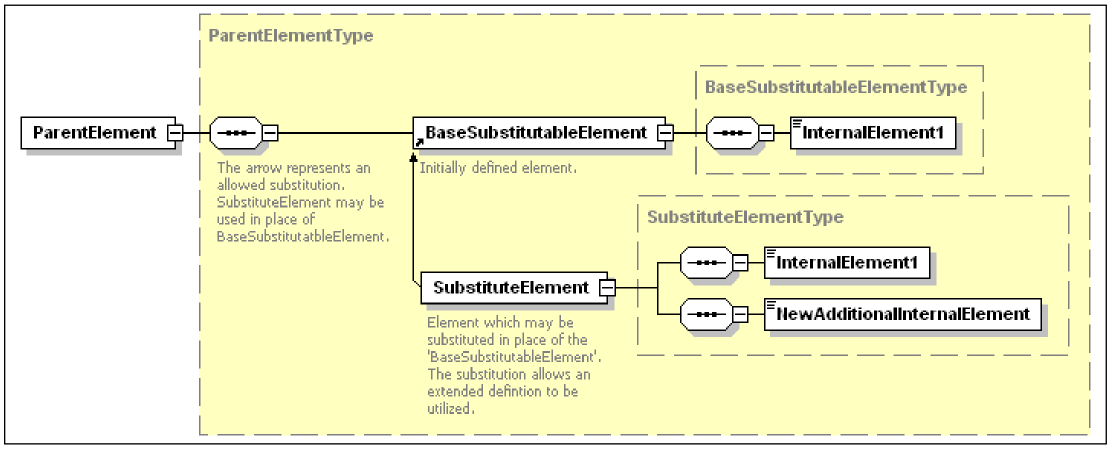

ParentElementType
ParentElement
BaseSubstitutableElementType
BaseSubstitutableElement
InternalElement1
Initially defined element.
The arrow represents an allowed substitution. SubstituteElement may be used in place of BaseSubstitutatbleElement.
SubstituteElementType
SubstituteElement
InternalElement1
NewAdditionalInternalElement
Element which may be substituted in place of the 'BaseSubstitutableElement'. The substitution allows an extended defintion to be utilized.

Figure 2. Substitution Group Schema Convention

In Figure 2, the element referred to as "SubstituteElement" *may* be used in place of the element named "BaseSubstitutableElement" provided the XML namespace declarations are included in the document as per [W3C-XSD]. The diagram's illustrative arrow signals the reader of the possible element substitution.

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 'gis' 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.

Standard	**XML Schema Prefix**	**XML Schema Elements**	**Value**
2020 (latest)	core (SCTE 130-2)	Schema namespace	http://www.scte.org/schemas/130-2/2008a/core[1]
		Schema version attribute	20200321
		Schema filename	SCTE_130-2_core_20200321.xsd
	gis (This doc.)	Schema namespace	http://www.scte.org/schemas/130-8/2011/gis[2]
		Schema version attribute	20200325

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

[2] While this specification has a ratified year of 2020, the XML schema/XSD namespace has a year of 2011, which is the year the XSD and this specification's syntax was initially ratified. All subsequent changes have been backwards compatible and thus, the namespace has not changed.

		Schema filename	SCTE_130-8_gis_20200325.xsd
	mut (This doc. Appendix F	Schema namespace	http://www.scte.org/schemas/130-8/2011/gis/mutable[3]
		Schema version attribute	20200325
		Schema filename	SCTE_130-8_mutable_20200325.xsd
XML foundation. See [W3C-XSD].	xsd	Schema namespace	http://www.w3.org/2001/XMLSchema

Table 1. XML Namespace Declarations

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

10.0 GIS MESSAGE TYPES

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 GIS message interface ***shall*** include the messages defined in [SCTE130-2] and logical services that incorporate the GIS message interface ***shall*** conform to the messaging transport mechanism described in [SCTE130-7].

Table 2 identifies additional SCTE 130 Part 8 (GIS) specific message types.

[3] While this specification has a ratified year of 2020, the XML schema/XSD namespace has a year of 2011, which is the year the XSD and this specification's syntax was initially ratified. All subsequent changes have been backwards compatible and thus, the namespace has not changed.

Message	Description
ListSupportedFeaturesRequestType	Request to retrieve a list of a logical service's supported features
ListSupportedFeaturesResponseType	Response to ListSupportedFeaturesRequest
ListQualifiersRequestType	Request to retrieve a list of names that *may* be used to construct basic queries using name/value pairs
ListQualifiersResponseType	Response to ListQualifiersRequest
ListNotificationRegistrationRequestType	Request to list existing registrations
ListNotificationRegistrationResponseType	Response to ListNotificationRegistrationRequest
NotificationRegistrationRequestType	Registration request for notification
NotificationRegistrationResponseType	Response to NotificationRegistrationRequest
NotificationType	Notification message indicating a change to the result set of a registered query
NotificationAcknowledgementType	Response to Notification
CreateCursorRequestType	Request to create a cursor
CreateCursorResponseType	Response to CreateCursorRequest
CancelCursorRequestType	Request to cancel an existing cursor
CancelCursorResponseType	Response to CancelCursorRequest
QueryRequestType	Request to acquire records from the GIS
QueryResponseType	Response to QueryRequest
NotificationDeregisterRequestType	Request to de-register a previously accepted registration
NotificationDeregisterResponseType	Response to NotificationDeregisterRequest
DeregistrationNotificationType	Deregistration notification
DeregistrationAcknowledgementType	Deregistration notification acknowledgement

Table 2. GIS Specific Message Types

Because this specification defines an abstract messaging interface used by other logical service implementations, this specification defines the message types rather than message elements. Every logical service implementing the interface specified by SCTE 130 Part 8 ***shall*** define a set of message elements semantically equivalent to the message types defined in SCTE 130 Part 8 by sub-classing the message types listed in Table 2 and by assigning a new message element name to each message formed by prefixing a string representing the logical service type to the message type name.

For example, the Subscriber Information Service (SIS) schema definition might contain this definition for the ListSupportedFeaturesRequest message.

```
<xsd:element name="SISListSupportedFeaturesRequest" type="gis:ListSupportedFeaturesRequestType"/>
```

Example 1. SISListSupportedFeaturesRequest Message Definition

The following subsections provide specific information on SCTE 130 Part 8's use of attributes and message types defined in [SCTE130-2]. For detailed information on these attributes and message types see [SCTE130-2].

10.1 @version Attribute

SCTE 130 Part 8 specifies message types with the intent that other logical services *may* create service specific concrete message definitions using the SCTE 130 Part 8 message types. Thus, SCTE 130 Part 8 ***shall not*** specify a value for the @version attribute. Logical services that derive messages from the SCTE 130 Part 8 message types ***shall*** set the @version attribute to a value that reflects the logical service's revision.

10.2 Request Base Message Type

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

10.3 Response Base Message Type

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

10.4 Notification Base Message Type

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

10.5 Acknowledgement Base Message Type

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

10.6 Messages Requiring Notification Registration

Consumers are required to register with any logical service implementing the GIS interface in order to receive Notification messages. The logical service ***shall not*** send a NotificationDeregisterRequest message type to any consumer when the consumer has no notification requests registered with the logical service. All other GIS message types do not require registration and *may* be sent at any time.

10.7 Default Logical Service Channel EndPoint

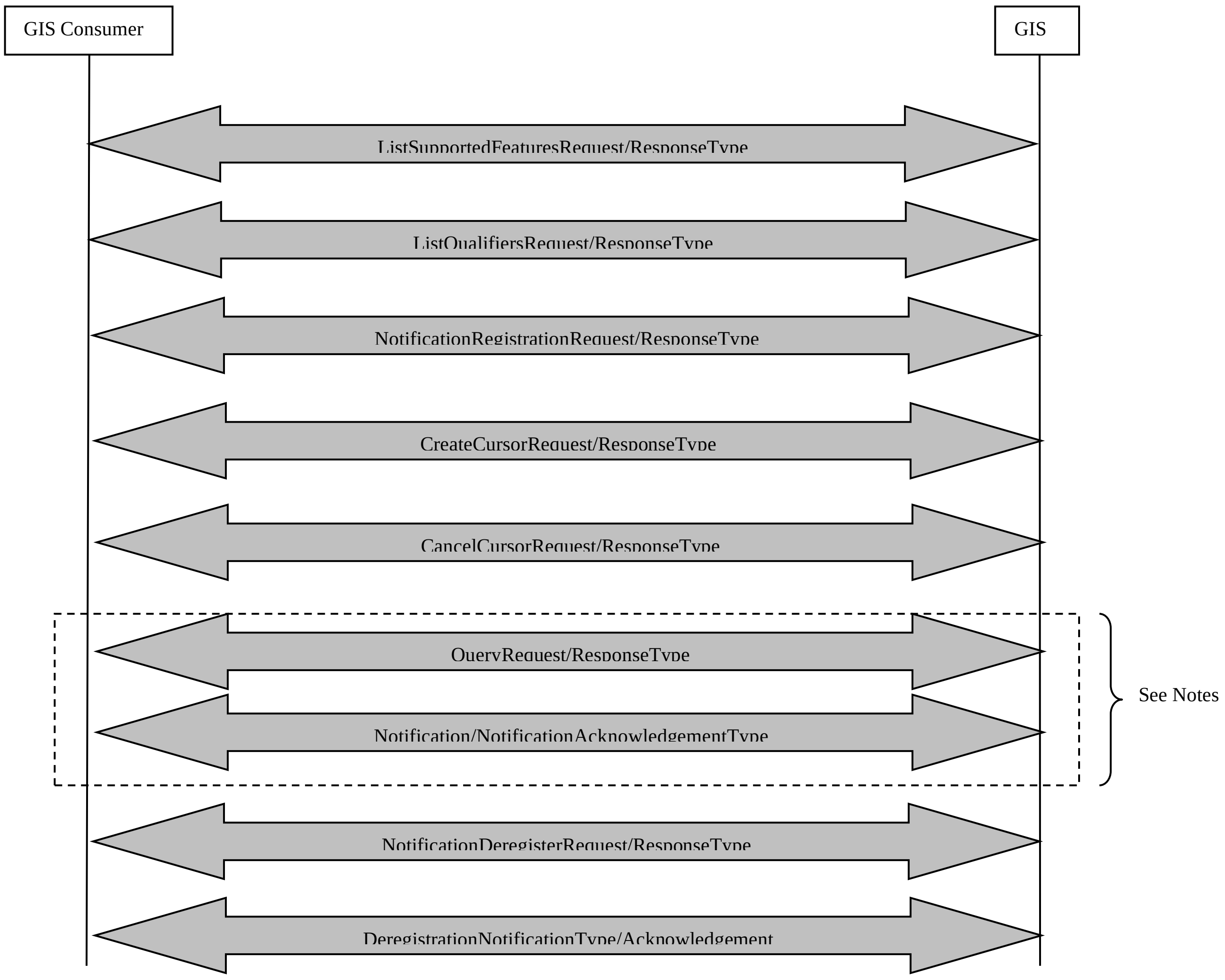
A logical service incorporating the GIS interface ***shall*** support the receipt of the ListSupportedFeaturesRequestType message type. Typically, this message type is received on a default (or well known) logical service channel endpoint address. Other GIS messages types *may* also be offered on the same logical service channel endpoint or they *may* be optionally offered on additional logical service channel endpoints. In order to discover all GIS provided logical service channel endpoints, consumers *should* complete a ListSupportedFeaturesRequestType/ListSupportedFeatureResponseType message type exchange.

See Section 11.2 for additional information on the ListSupportedFeaturesResponseType message type.

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

11.0 GIS MESSAGE EXCHANGE

The following diagram illustrates a typical message exchange between a GIS consumer and the GIS.


GIS Consumer
GIS
ListSupportedFeaturesRequest/ResponseType
ListQualifiersRequest/ResponseType
NotificationRegistrationRequest/ResponseType
CreateCursorRequest/ResponseType
CancelCursorRequest/ResponseType
QueryRequest/ResponseType
Notification/NotificationAcknowledgementType
See Notes
NotificationDeregisterRequest/ResponseType
DeregistrationNotificationType/Acknowledgement

Notes: *The Query and Notification exchange may be called or may occur repeatedly between logical service channel setup and tear down*.

Figure 3. GIS Message Exchange

Figure 3 illustrates the message exchanges specific to the GIS. The service check and service status message exchanges defined in [SCTE130-2] are not depicted in this illustration.

11.1 ListSupportedFeaturesRequest and ListSupportedFeaturesResponse Message Types

The ListSupportedFeaturesRequest and ListSupportedFeaturesResponse message types allow consumers to inquire about the service data models and advanced query languages supported by a logical service.

11.1.1 ListSupportedFeaturesRequest Message Type

The ListSupportedFeaturesRequest message type allows the consumer of a logical service to inquire about the service data models and advanced query languages supported by the service.

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

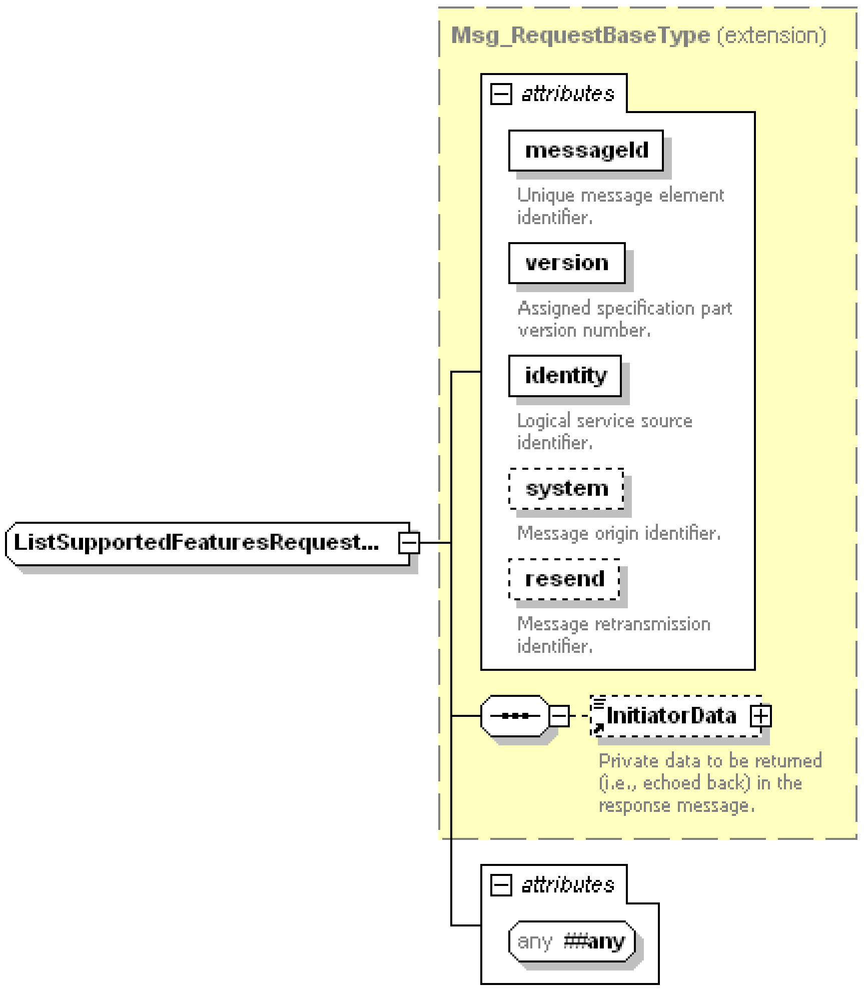

Msg_RequestBaseType (extension)
attributes
messageId
Unique message element identifier.
version
Assigned specification part version number.
identity
Logical service source identifier.
system
Message origin identifier.
resend
Message retransmission identifier.
ListSupportedFeaturesRequest...
InitiatorData
Private data to be returned (i.e., echoed back) in the response message.
attributes
any ##any

Figure 4. ListSupportedFeaturesRequestType XML Schema

The ListSupportedFeaturesRequest message type defines no elements in addition to those that are already defined by core:Msg_RequestBaseType.

@##any [Optional] — The @##any attribute enables a logical service incorporating the SCTE 130 Part 8 interface to optionally extend concrete messages derived from SCTE 130 Part 8 message types with attributes not specified by the schema.

11.1.2 ListSupportedFeaturesResponse Message Type

If a ListSupportedFeaturesRequest type message is successful, the matching ListSupportedFeaturesResponse message type ***shall*** contain, at a minimum, a single core:Callout element containing one or more core:Address element(s).

If a ListSupportedFeaturesRequest type message is not successful, the matching ListSupportedFeaturesResponse message type ***shall not*** contain a core:Callout element.

See [SCTE130-2] for additional information on the core:Callout element. The XML schema definition for this message is shown in Figure 5.

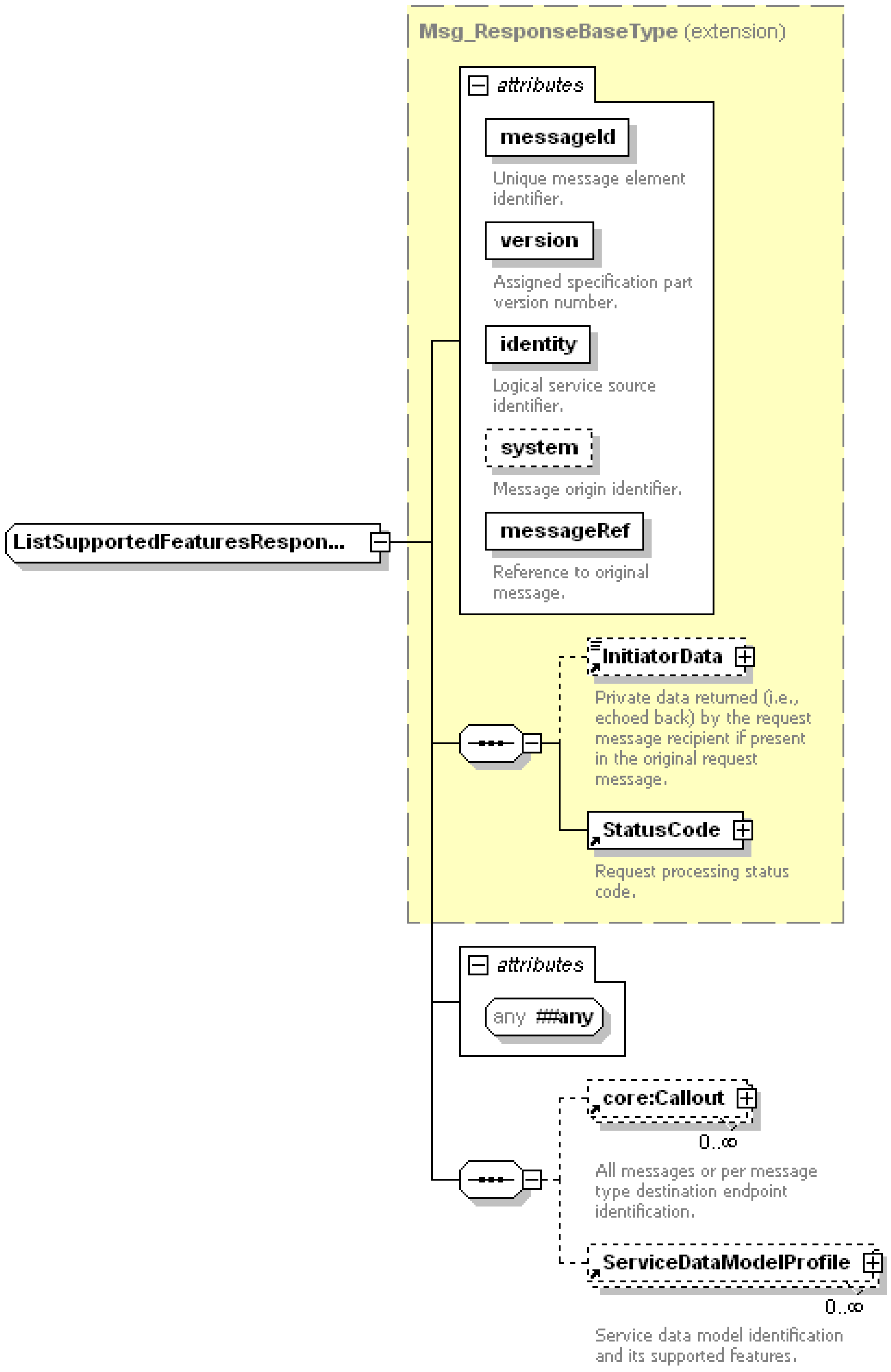

Msg_ResponseBaseType (extension)
attributes
messageId
Unique message element identifier.
version
Assigned specification part version number.
identity
Logical service source identifier.
system
Message origin identifier.
messageRef
Reference to original message.
ListSupportedFeaturesRespon...
InitiatorData
Private data returned (i.e., echoed back) by the request message recipient if present in the original request message.
StatusCode
Request processing status code.
attributes
any ##any
core:Callout
0..∞
All messages or per message type destination endpoint identification.
ServiceDataModelProfile
0..∞
Service data model identification and its supported features.

Figure 5. ListSupportedFeaturesResponseType XML Schema

The ListSupportedFeaturesResponse message type is derived from the core namespace base type core:Msg_ResponseBaseType and defines the following attributes and elements in addition to those already defined by core:Msg_ResponseBaseType.

@##any [Optional] — The @##any attribute enables a logical service incorporating the SCTE 130 Part 8 interface to optionally extend concrete messages derived from SCTE 130 Part 8 message types with attributes not specified by the schema.

core:Callout [Optional] — Zero or more core:Callout elements specifying the service channel message endpoint(s). See [SCTE130-2] for additional information on the core:Callout element.

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 where "xxx" is a string representing the logical service as shown in Example 1.

@message Attribute Value	Description
xxxNotificationRegistrationRequest	Destination endpoint for message of type NotificationRegistrationRequestType
xxxNotificationDeregisterRequest	Destination endpoint for message of type NotificationDeregisterRequestType
xxxListNotificationRegistrationRequest	Destination endpoint for message of type ListNotificationRegistrationRequestType
xxxListQualifiersRequest	Destination endpoint for message of type ListQualifiersRequestType
xxxCreateCursorRequest	Destination endpoint for message of type CreateCursorRequestType
xxxCancelCursorRequest	Destination endpoint for message of type CancelCursorRequestType
xxxQueryRequest	Destination endpoint for message of type QueryRequestType
ServiceStatusNotification	Destination endpoint for message of type core:ServiceStatusNotificationType
…	User defined address endpoint outside of the scope of this specification. The string ***shall*** be prefixed with the text “private:”

Table 3. ListSupportedFeaturesResponseType core:Callout @message values

All message values listed in Table 3 and not present in the ListSupportedFeaturesResponseType message’s core:Callout XML element sequence ***shall*** be available through the default endpoint if present. (The default endpoint is identified by a core:Callout element not having the @message attribute.) See [SCTE130-2] for additional information.

ServiceDataModelProfile[Optional] — A list of zero or more ServiceDataModelProfile elements – one for each service data model supported by the logical service. The first ServiceDataModelProfile element is referred to as the default service data model and ***shall*** be used when no service data model is specified. For more information on the ServiceDataModelProfile element see Section 12.25.

11.2 ListQualifiersRequest and ListQualifiersResponse Message Types

The ListQualifiersRequest and ListQualifiersResponse message types allow consumers to discover the qualifiers associated with service data models that *may* be queried using the basic query interface. If a specific service data model does not support the basic query interface, the ListQualifiersResponse message type ***shall*** return an error (specified later in this section).

11.2.1 ListQualifiersRequest Message Type

The ListQualifiersRequest message type allows the consumer of a logical service that implements the GIS interface to inquire about the unique qualifiers, the qualifier names and the qualifier descriptions used by a logical service's data model.

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

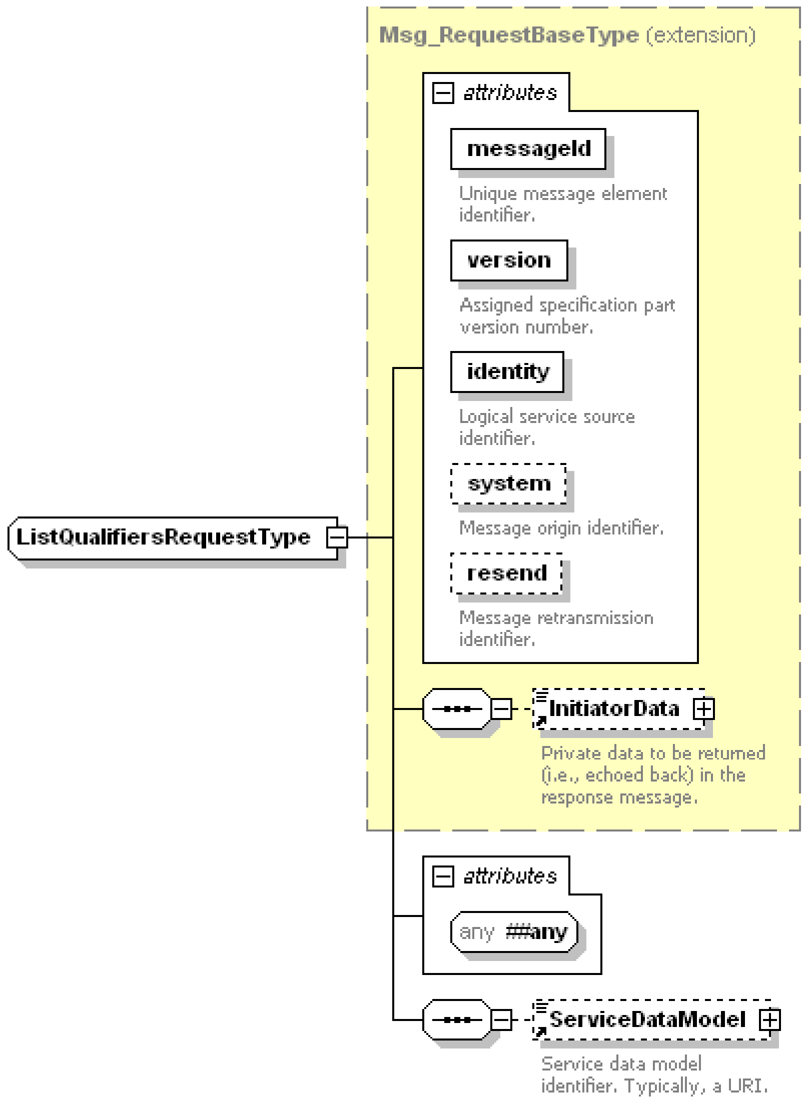

Msg_RequestBaseType (extension)
attributes
messageId
Unique message element identifier.
version
Assigned specification part version number.
identity
Logical service source identifier.
system
Message origin identifier.
resend
Message retransmission identifier.
InitiatorData
Private data to be returned (i.e., echoed back) in the response message.
ListQualifiersRequestType
attributes
any ##any
ServiceDataModel
Service data model identifier. Typically, a URI.

Figure 6. ListQualifiersRequestType XML Schema

The ListQualifiersRequest message type defines the following attributes and elements in addition to those that are already defined on the abstract base message core:Msg_RequestBaseType.

@##any [Optional] — The @##any attribute enables a logical service incorporating the SCTE 130 Part 8 interface to optionally extend concrete messages derived from SCTE 130 Part 8 message types with attributes not specified by the schema.

ServiceDataModel [Optional] — The ServiceDataModel element contains a unique identifier (usually a URI) for a service data model that *may* be accessed using the basic query interface. If the ServiceDataModel element is not present, the default service data model ***shall*** be selected. For additional information on the ServiceDataModel element, see Section 12.24

11.2.2 ListQualifiersResponse Message Type

If the service data model specified by the ServiceDataModel element in the ListQualifiersRequest type message is known to the logical service and is a basic query data model, the matching ListQualifiersResponse message type ***shall*** contain a single BasicQueryDataModelDescription element describing the service data model. If the service data model is known but does not support a basic query interface (i.e., it only supports advanced queries), the ListQualifiersResponse message type ***shall*** return an error where the core:StatusCode element ***shall*** contain a @detailCode attribute set to the value core:NotSupported.

If the service data model specified by the ServiceDataModel element in the ListQualifiersRequest message is not known to the logical service, the matching ListQualifiersResponse message type ***shall not*** contain a BasicQueryDataModelDescription element and the core:StatusCode element ***shall*** contain a @detailCode attribute set to the value core:ResourceNotFound.

The XML schema definition for this message is illustrated in Figure 7.

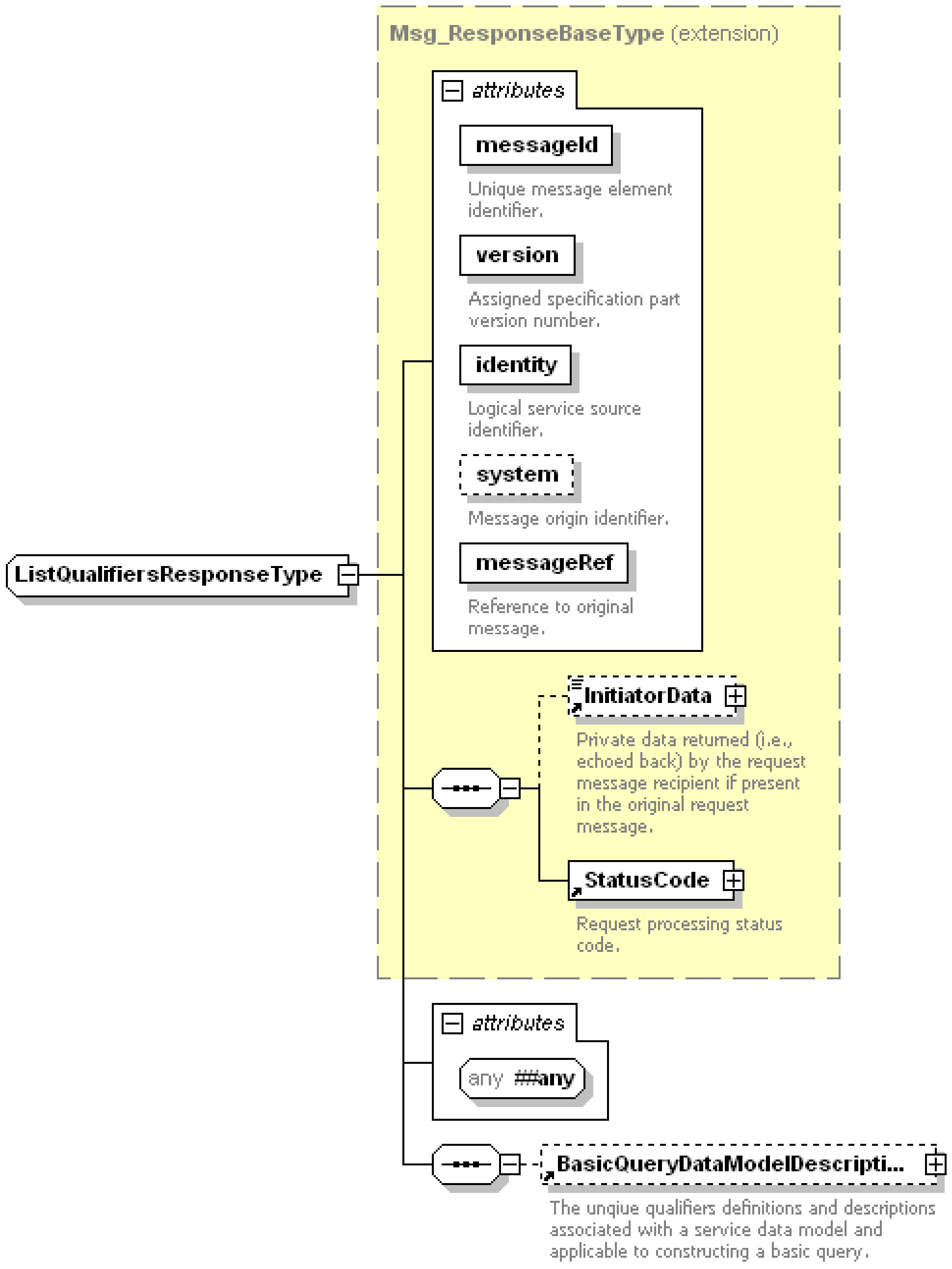

Msg_ResponseBaseType (extension)
attributes
messageId
Unique message element identifier.
version
Assigned specification part version number.
identity
Logical service source identifier.
system
Message origin identifier.
messageRef
Reference to original message.
ListQualifiersResponseType
InitiatorData
Private data returned (i.e., echoed back) by the request message recipient if present in the original request message.
StatusCode
Request processing status code.
attributes
any ##any
BasicQueryDataModelDescripti...
The unqiue qualifiers definitions and descriptions associated with a service data model and applicable to constructing a basic query.

Figure 7. ListQualifiersResponseType XML Schema

The ListQualifiersResponse message type is derived from the core namespace base type core:Msg_ResponseBaseType and defines the following attributes and elements in addition to those that are already defined on the abstract base message core:Msg_ResponseBaseType.

@##any [Optional] — The @##any attribute enables a logical service incorporating the SCTE 130 Part 8 interface to optionally extend concrete messages derived from SCTE 130 Part 8 message types with attributes not specified by the schema.

BasicQueryDataModelDescription [Optional] — The BasicQueryDataModelDescription element defines the service data model's unique qualifiers and provides descriptions for a group of qualifiers. For more information on the BasicQueryDataModelDescription element see Section 12.7.

11.3 ListNotificationRegistrationRequest and ListNotificationRegistrationResponse Message Types

A GIS consumer *may* inquire about current notification registrations by using the ListNotificationRegistrationRequest message type. The logical service ***shall*** respond to the ListNotificationRegistrationRequest type message with a ListNotificationRegistrationResponse type message. This message exchange allows a GIS consumer to discover the active queries previously installed by one or more NotificationRegistrationRequest type messages.

There are two list registration granularity levels:

- Per logical service (i.e., via the @identity attribute)
- Per registration message

A logical service *may* restrict access to registration information by returning a status code equating to "not authorized" for the list registration request on a per logical service basis. The per registration message granularity level ***shall*** be supported by all GIS compliant logical services. The per logical service message granularity level *may* be supported by any GIS compliant logical service.

11.3.1 ListNotificationRegistrationRequest Message Types

The ListNotificationRegistrationRequest message *may* be issued to a logical service to retrieve information about active notification registrations.

The XML schema definition for this message is illustrated in Figure 8.

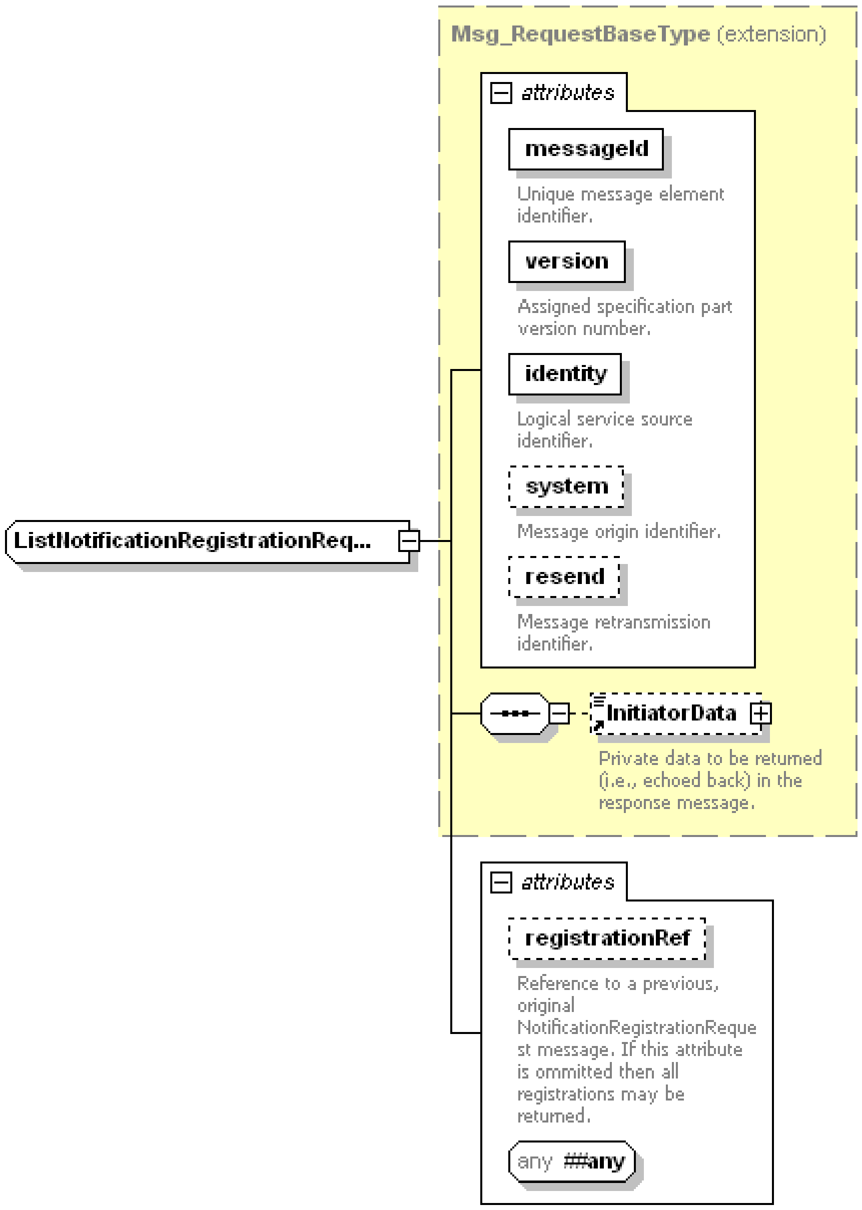

Msg_RequestBaseType (extension)
attributes
messageId
Unique message element identifier.
version
Assigned specification part version number.
identity
Logical service source identifier.
system
Message origin identifier.
resend
Message retransmission identifier.
ListNotificationRegistrationReq...
InitiatorData
Private data to be returned (i.e., echoed back) in the response message.
attributes
registrationRef
Reference to a previous, original NotificationRegistrationRequest message. If this attribute is ommitted then all registrations may be returned.
any ##any

Figure 8. ListNotificationRegistrationRequestType XML Schema

The ListNotificationRegistrationRequest message returns all active registrations associated with a specific logical service source as identified by the @identity attribute when the @registrationRef attribute is omitted. If only a specific registration is to be returned, both the @identity and the @registrationRef attributes ***shall*** be included. The following table defines the inclusion relationships.

@identity	@registrationRef	Description	Access Restrictions Permitted
Included	Omitted	All registrations matching the supplied @identity value.	Yes
Included	Included	A specific registration matching the @identity and @registrationRef values.	No

Table 4. List ADS Registration Inquiry Granularity Control

The access restrictions permitted column indicates whether the status code equivalent to core:NotAuthorized *may* optionally be returned.

The ListNotificationRegistrationRequest message type is derived from the core namespace base type core:Msg_RequestBaseType and defines the following attributes and elements in addition to those that are already defined on the abstract base message core:Msg_RequestBaseType.

@registrationRef [Optional, core:registrationRefAttrType] — A reference to an original NotificationRegistrationRequest type message. This attribute instructs the logical service to list only the notification registration request identified by this reference. The value ***shall*** be the NotificationRegistrationRequest message’s original @messageId attribute value. If this attribute is not present, then the matching ListNotificationRegistrationResponse type message ***shall*** contain information about all notifications registered for the consumer as identified by the @identity attribute.

@##any [Optional] — The @##any attribute enables a logical service incorporating the SCTE 130 Part 8 interface to optionally extend concrete messages derived from SCTE 130 Part 8 message types with attributes not specified by the schema.

11.3.2 ListNotificationRegistrationResponse Message Type

The ListNotificationRegistrationResponse message type is the response pair to the previously defined ListNotificationRegistrationRequest message type.

The XML schema definition for this message is shown in Figure 9.

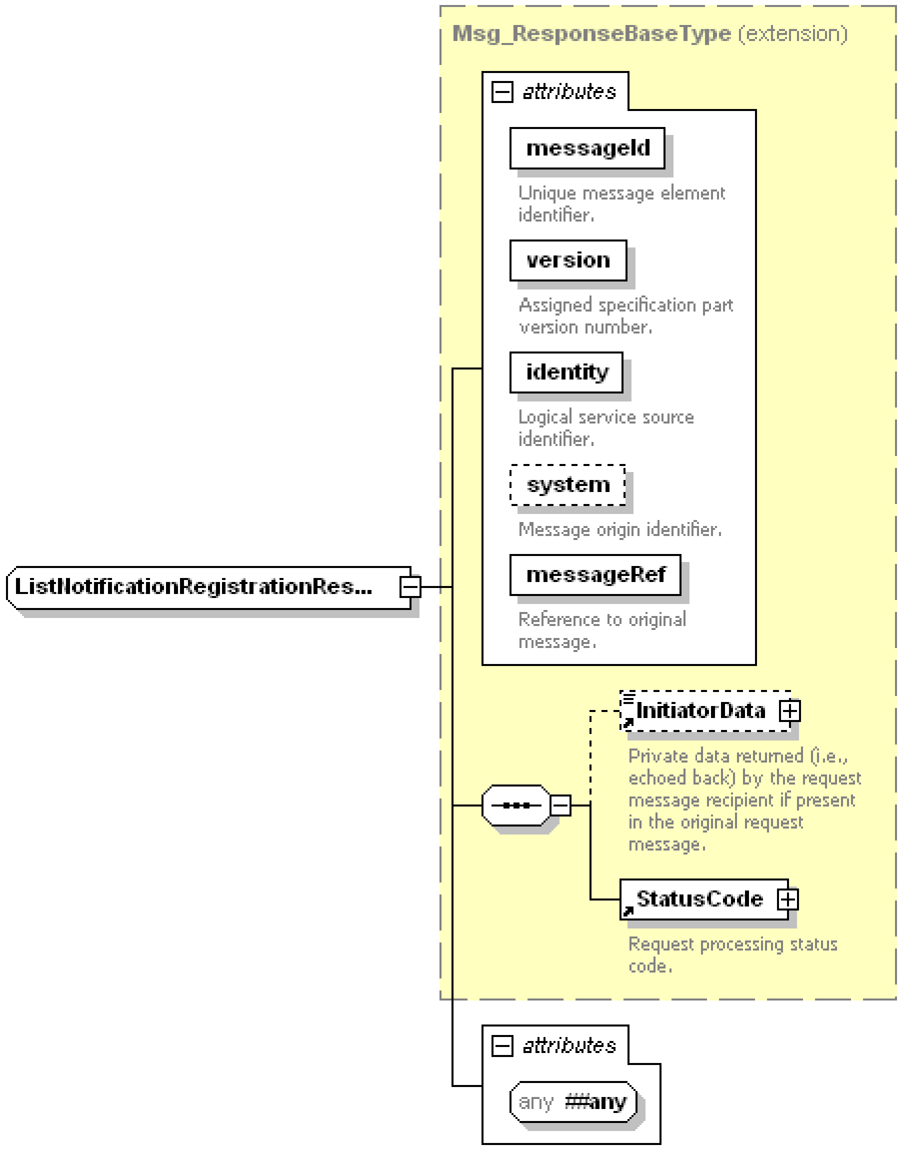

Msg_ResponseBaseType (extension)
attributes
messageId
Unique message element identifier.
version
Assigned specification part version number.
identity
Logical service source identifier.
system
Message origin identifier.
messageRef
Reference to original message.
ListNotificationRegistrationRes...
InitiatorData
Private data returned (i.e., echoed back) by the request message recipient if present in the original request message.
StatusCode
Request processing status code.
attributes
any ##any

Figure 9. ListNotificationRegistrationResponseType XML Schema

The ListNotificationRegistrationResponse message type is derived from the core namespace base type core:Msg_ResponseBaseType and defines the following attributes and elements in addition to those that are already defined on the abstract base message core:Msg_ResponseBaseType.

@##any [Optional] — The @##any attribute enables a logical service incorporating the SCTE 130 Part 8 interface to optionally extend concrete messages derived from SCTE 130 Part 8 message types with attributes not specified by the schema.

A logical service incorporating the GIS NotificationRegistrationRequest message type ***shall*** also include the GIS ListNotificationRegistrationRequest and GIS ListNotificationRegistrationResponse message types (typically defined as the xxxListNotificationRegistrationRequest and xxxListNotificationRegistrationResponse messages by the logical service). The xxxListNotificationRegistrationResponse message, which ***shall*** extend the GIS ListNotificationRegistrationResponse message type, ***shall*** include the logical service's xxxNotificationRegistrationRequest message as part of the element type definition and the inclusion *should* be specified as a sequence of zero or more instances of the xxxNotificationRegistrationRequest message.

The xxxListNotificationRegistrationResponse message type's xxxNotificationRegistrationRequest ***shall*** be a recoded 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 [SCTE130-2] for compliance requirements.

If the sub-classed ListNotificationRegistrationResponse message type is unable to locate the specific registration as specified by the ListNotificationRegistrationRequest message type's @registrationRef attribute or by the @identity attribute, the response message type's core:StatusCode element ***shall*** contain the value core:ResourceNotFound (i.e., a response of zero located registrations ***shall*** return an error).

11.4 NotificationRegistrationRequest and NotificationRegistrationResponse Message Types

A logical service that implements the GIS interface ***shall*** support registration for notification message delivery. The NotificationRegistrationRequest message type allows a consumer to specify notification interests relative to a basic or an advanced query.

On receipt of an update, addition or deletion event from its underlying data store affecting the result set of a registered query, the logical service ***shall*** send a Notification type message to each matching registered consumer.

11.4.1 NotificationRegistrationRequest Message Type

The NotificationRegistrationRequest message type allows a consumer to specify a set of notification interests by registering a query against a logical service's data model. These registered queries ***shall*** be examined by the logical service relative to changes in any data relevant to the query. If any change to the data causes a change to the query result for a registered query, a notification containing the new result ***shall*** be sent to the consumer in the form of a Notification message type.

The XML schema representation of the NotificationRegistrationRequest message type is illustrated in Figure 10.

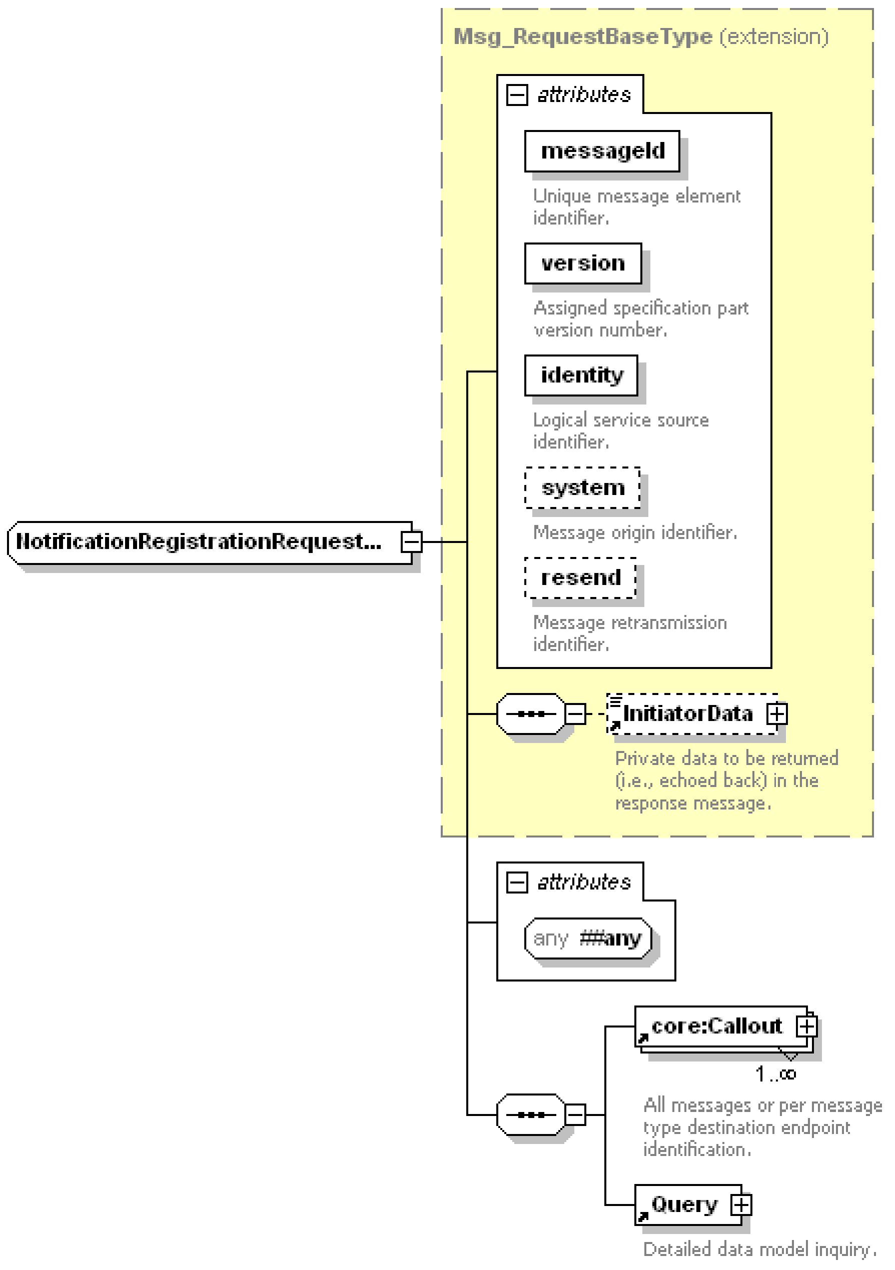

Msg_RequestBaseType (extension)
attributes
messageId
Unique message element identifier.
version
Assigned specification part version number.
identity
Logical service source identifier.
system
Message origin identifier.
resend
Message retransmission identifier.
NotificationRegistrationRequest...
InitiatorData
Private data to be returned (i.e., echoed back) in the response message.
attributes
any ##any
core:Callout
1..∞
All messages or per message type destination endpoint identification.
Query
Detailed data model inquiry.

Figure 10. NotificationRegistrationRequestType XML Schema

The NotificationRegistrationRequest message type is derived from the core namespace base type core:Msg_RequestBaseType and defines the following attributes and elements in addition to those already defined on the abstract base message core:Msg_RequestBaseType.

@##any [Optional] — The @##any attribute enables a logical service incorporating the SCTE 130 Part 8 interface to optionally extend concrete messages derived from SCTE 130 Part 8 message types with attributes not specified by the schema.

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

Before generating a NotificationRegistrationResponse type message, the logical service *may* send a core:ServiceCheckRequest message to a core:Address, in order to verify the validity of the callback endpoint.

An logical service implementing the GIS interface ***shall*** recognize the values listed in Table 5 as values for the core:Callout @message attribute. Values for the @message attribute *should* be used exactly as defined in this table where "xxx" is a string representing the logical service as shown in Example 1.

@message Attribute Value	Description
xxxNotification	Value associated with the address endpoint where Notification type messages ***shall*** be sent.
ServiceStatusNotification	Value associated with the address endpoint where core:ServiceStatusNotification messages ***shall*** be sent.
xxxDeregistrationNotification	Value associated with the address endpoint where DeregistrationNotification type 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 5. NotificationRegistrationRequest core:Callout @message Values

All message values listed in Table 5 and not present in the NotificationRegistrationRequestType message’s core:Callout XML element sequence ***shall*** be available through the default endpoint if present. (The default endpoint is identified by a core:Callout element not having the @message attribute.) See [SCTE130-2] for additional information. Either the xxxNotification or the default endpoint ***shall*** be present in the NotificationRegistrationRequest core:Callout element sequence.

Query [Required] — The Query element contains elements defining an inquiry for notification purposes. See Section 12.22 for additional details on the Query element.

11.4.2 NotificationRegistrationResponse Message Type

Upon completion of processing a NotificationRegistrationRequest type message, the logical service ***shall*** respond with a NotificationRegistrationResponse type message.

The XML schema diagram for the NotificationRegistrationResponse type message is shown in Figure 11.

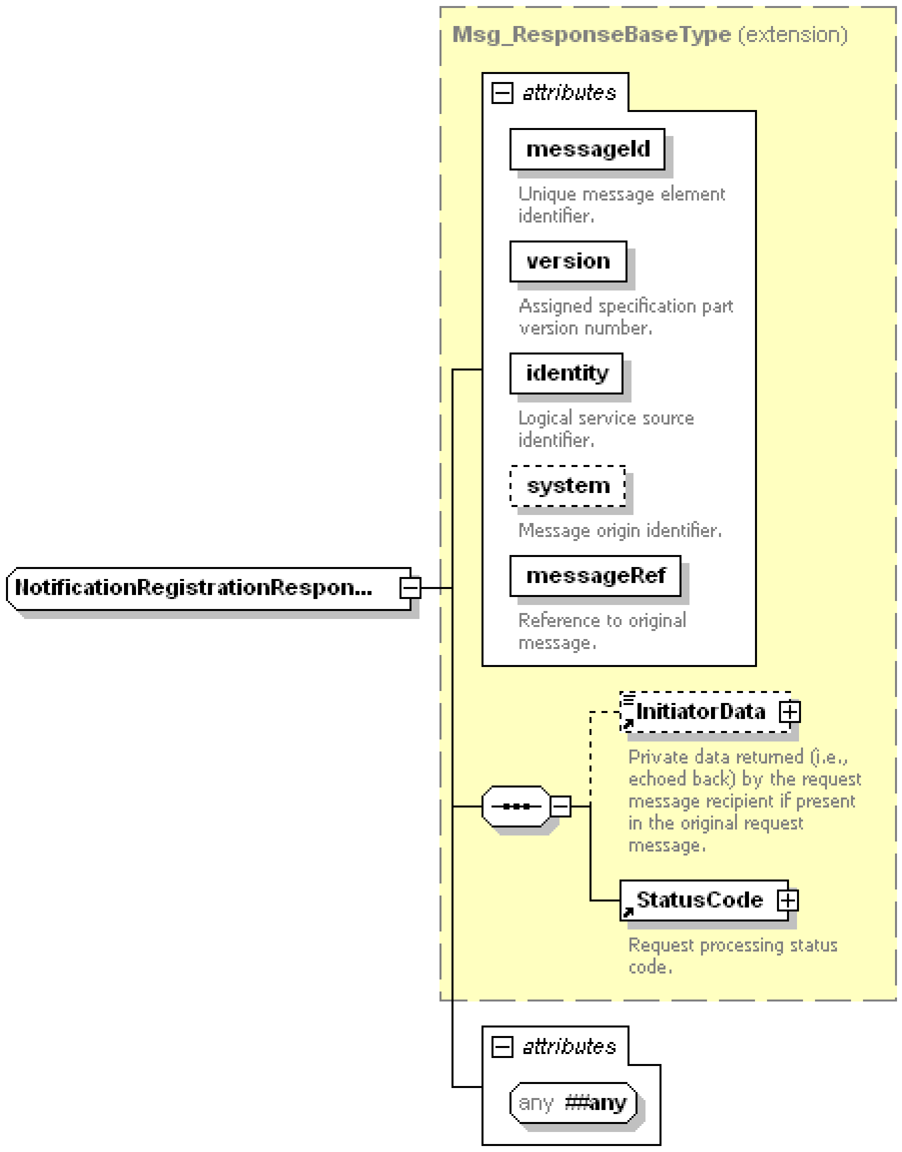

Msg_ResponseBaseType (extension)
attributes
messageId
Unique message element identifier.
version
Assigned specification part version number.
identity
Logical service source identifier.
system
Message origin identifier.
messageRef
Reference to original message.
NotificationRegistrationRespon...
InitiatorData
Private data returned (i.e., echoed back) by the request message recipient if present in the original request message.
StatusCode
Request processing status code.
attributes
any ##any

Figure 11. NotificationRegistrationResponseType XML Schema

The NotificationRegistrationResponse message type is derived from the core namespace base type core:Msg_ResponseBaseType and defines no elements in addition to those already defined on the abstract base message core:Msg_ResponseBaseType.

@##any [Optional] — The @##any attribute enables a logical service incorporating the SCTE 130 Part 8 interface to optionally extend concrete messages derived from SCTE 130 Part 8 message types with attributes not specified by the schema.

11.5 Notification and NotificationAcknowledgement Message Types

A logical service that implements the GIS interface ***shall*** support the exchange of Notification and NotificationAcknowledgement type messages with registered consumers for the purpose of notifying the consumer of changes in data relevant to the consumer's registered queries.

11.5.1 Notification Message Type

Upon detection of a change in the result set returned for one or more queries registered with the service, the logical service ***shall*** send a Notification type message to any consumers with registered queries that are affected. A change to the result set of a registered query includes the set of definitions described in Table 6. The values ***shall*** appear in the @noticeType attribute exactly as they appear in Table 6 (i.e. all in lower case).

@noticeType Attribute Value	**Description**
update	A service’s data store has been changed and the update has changed the result set of a registered query.
new	Data has been added to a service’s data store and the additions have changed the result set of a registered query.
delete	Data has been deleted from a service’s data store and the deletion(s) have changed the result set of a registered query.
…	User defined Notification types outside of the scope of this specification. The string ***shall*** be prefixed with the text “private:”.

Table 6. Notification @noticeType Values

The XML schema for the Notification type message is illustrated in Figure 12.

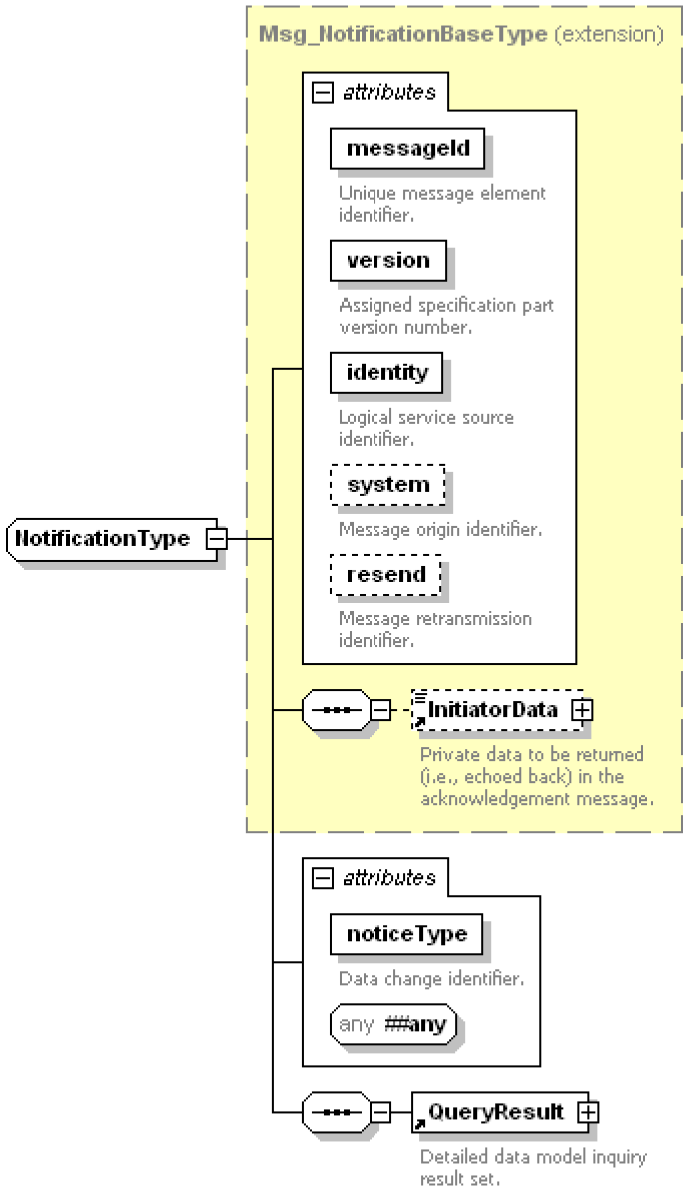

Msg_NotificationBaseType (extension)
attributes
messageId
Unique message element identifier.
version
Assigned specification part version number.
identity
Logical service source identifier.
system
Message origin identifier.
resend
Message retransmission identifier.
NotificationType
InitiatorData
Private data to be returned (i.e., echoed back) in the acknowledgement message.
attributes
noticeType
Data change identifier.
any ##any
QueryResult
Detailed data model inquiry result set.

Figure 12. NotificationType XML Schema

The Notification message type is derived from the core namespace base type core:Msg_NotificationBaseType and defines the following attributes and elements in addition to those defined by the core:Msg_NotificationBaseType.

@noticeType [Required, NotificationTypeEnumeration] — The @noticeType attribute is an enumeration that ***shall*** contain one of the values listed above in Table 6. See Section 13.9 for additional information.

@##any [Optional] — The @##any attribute enables a logical service incorporating the SCTE 130 Part 8 interface to optionally extend concrete messages derived from SCTE 130 Part 8 message types with attributes not specified by the schema.

QueryResult [Required] — This element contains the list of service data model components subject to the change in the data and is returned in response to the consumer's registered queries. See Section 12.23 for additional information on the QueryResult element.

11.5.2 NotificationAcknowledgement Message Type

Upon the receipt of a Notification message type, a GIS consumer ***shall*** respond with a NotificationAcknowledgement message type.

The XML schema for the NotificationAcknowledgement message type is shown in Figure 13.

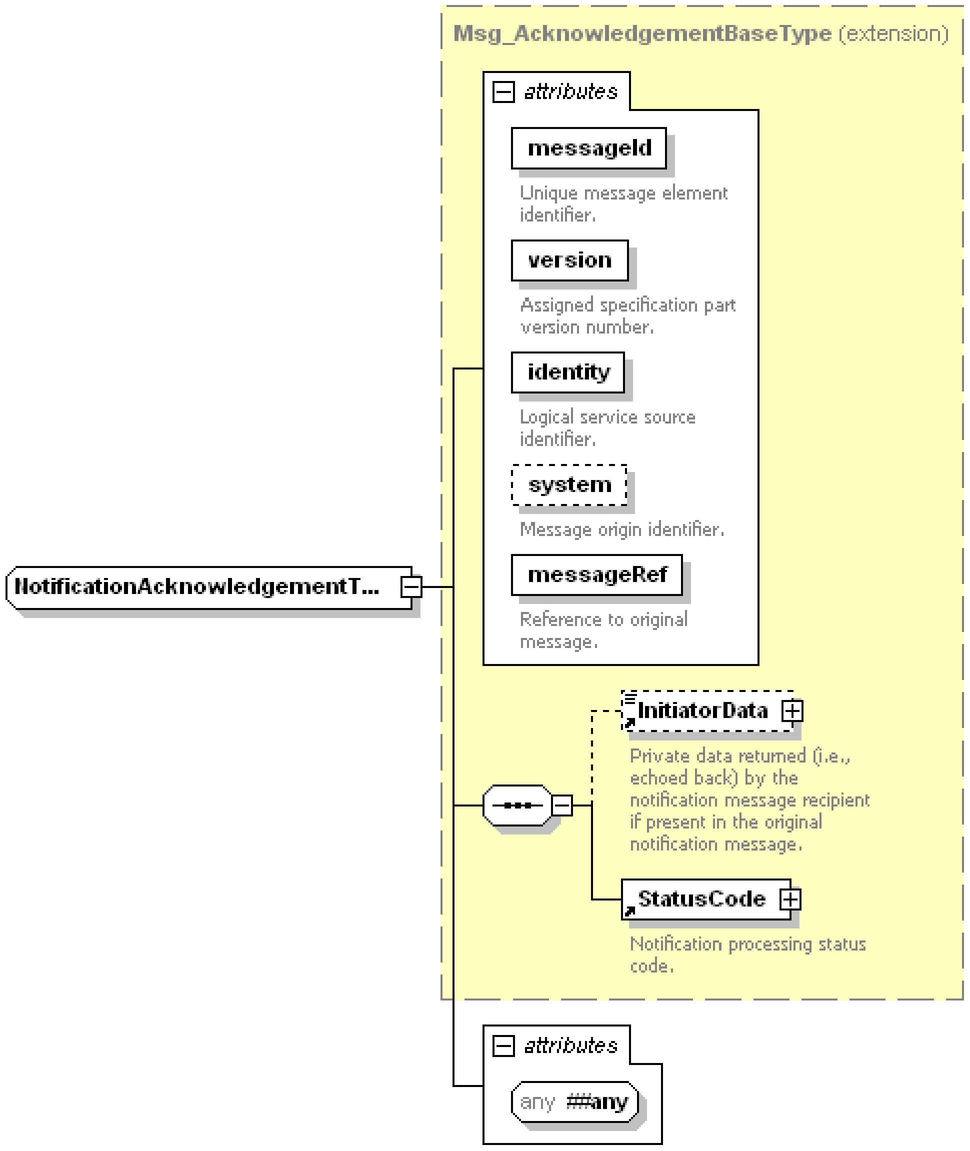

Msg_AcknowledgementBaseType (extension)
attributes
messageId
Unique message element identifier.
version
Assigned specification part version number.
identity
Logical service source identifier.
system
Message origin identifier.
messageRef
Reference to original message.
NotificationAcknowledgementT...
InitiatorData
Private data returned (i.e., echoed back) by the notification message recipient if present in the original notification message.
StatusCode
Notification processing status code.
attributes
any ##any

Figure 13. NotificationAcknowledgementType XML Schema

The NotificationAcknowledgement message type is derived from the core namespace base type core:Msg_AcknowledgementBaseType and defines no elements in addition to those defined by the core:Msg_AcknowledgementBaseType.

@##any [Optional] — The @##any attribute enables a logical service incorporating the SCTE 130 Part 8 interface to optionally extend concrete messages derived from SCTE 130 Part 8 message types with attributes not specified by the schema.

11.6 CreateCursorRequest and CreateCursorResponse Message Types

A logical service that implements the GIS interface ***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 using the GIS interface, the data information in the cursor ***shall*** remain static relative to the referenced data store.

Cursors have a limited life span, which is first requested by the consumer, but *may* be overridden by a logical service. As part of the cursor request message, the consumer ***shall*** specify a @cursorExpires attribute. This date and time is a request to a logical service for a specific end date and time for the cursor identified by the @cursorId attribute. A logical service, in order to maintain overall system health, *may* choose to override a requested cursor expires end date and time and substitute a different, implementation specific, cursor expires end date and time. See Section 12.11 for additional information on cursors.

11.6.1 CreateCursorRequest Message Type

The CreateCursorRequest message type is used to create an instance of a static cursor on a logical service that implements the GIS interface.

The XML schema for the CreateCursorRequest message type is listed in Figure 14.

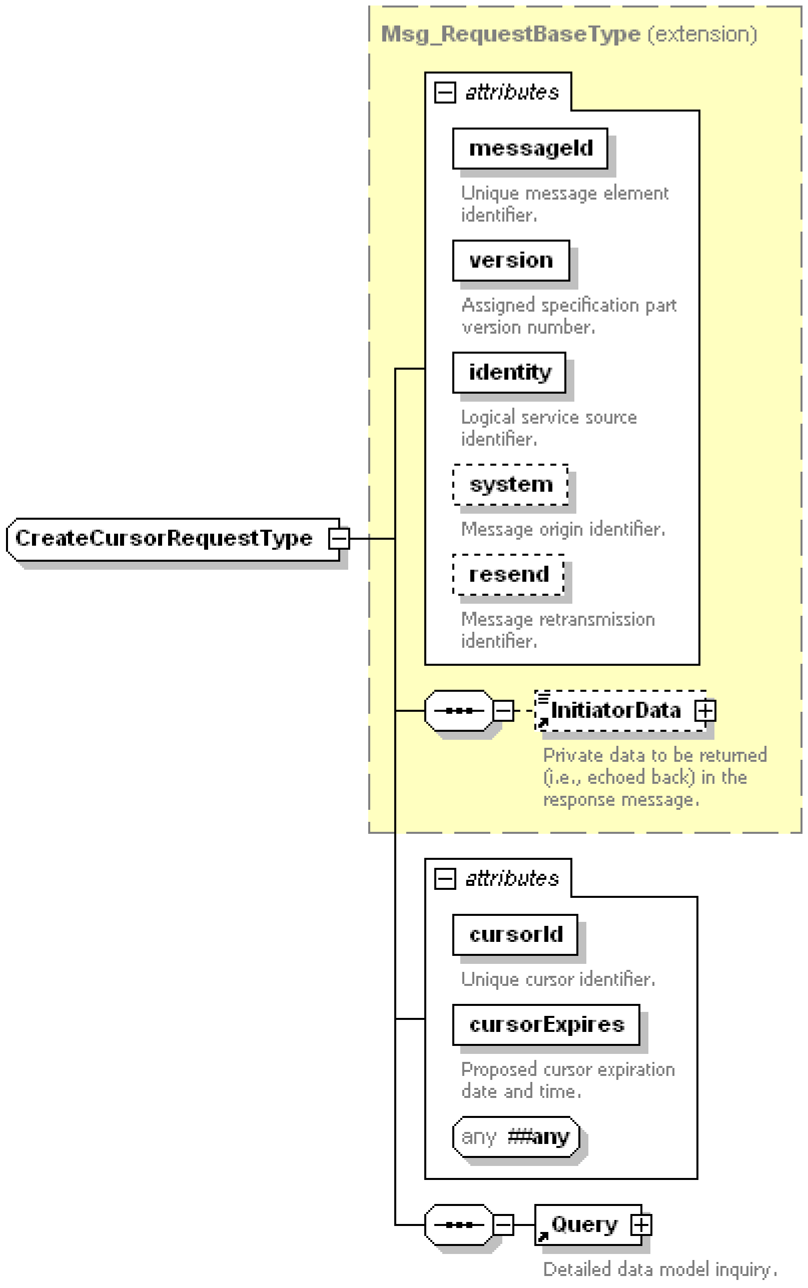

Msg_RequestBaseType (extension)
attributes
messageId
Unique message element identifier.
version
Assigned specification part version number.
identity
Logical service source identifier.
system
Message origin identifier.
resend
Message retransmission identifier.
CreateCursorRequestType
InitiatorData
Private data to be returned (i.e., echoed back) in the response message.
attributes
cursorId
Unique cursor identifier.
cursorExpires
Proposed cursor expiration date and time.
any ##any
Query
Detailed data model inquiry.

Figure 14. CreateCursorRequestType XML Schema

The CreateCursorRequest message type is derived from the core namespace base type core:Msg_RequestBaseType and defines the following attributes and elements in addition to those defined in the core:Msg_RequestBaseType.

@cursorId [Required, cursorIdAttrType] — The @cursorId attribute is a consumer generated identifier which ***shall*** be a service channel unique value. See Section 13.1 for additional information on the cursorIdAttrType type.

@cursorExpires [Required, core:dateTimeTimezoneType] — The @cursorExpires attribute is a consumer request for a cursor expiration date and time. A logical service ***shall not*** be required to create the cursor with the requested end date and time. A logical service *may* override the requested end date and time by returning an implementation specific end date and time that better fits within the implementation's specific design constraints. See [SCTE130-2] for information on the core:dateTimeTimezoneType.

@##any [Optional] — The @##any attribute enables a logical service incorporating the SCTE 130 Part 8 interface to optionally extend concrete messages derived from SCTE 130 Part 8 message types with attributes not specified by the schema.

Query [Required] — The query element contains the necessary attributes and elements for a logical service to execute one or more inquiries against its data store. Inquiry selected items ***shall*** be added to a static cursor construct identified by the supplied @cursorId attribute. See Section 12.22 for additional information on the Query element.

11.6.2 CreateCursorResponse Message Type

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

The XML schema for the CreateCursorResponse message type is listed in Figure 15

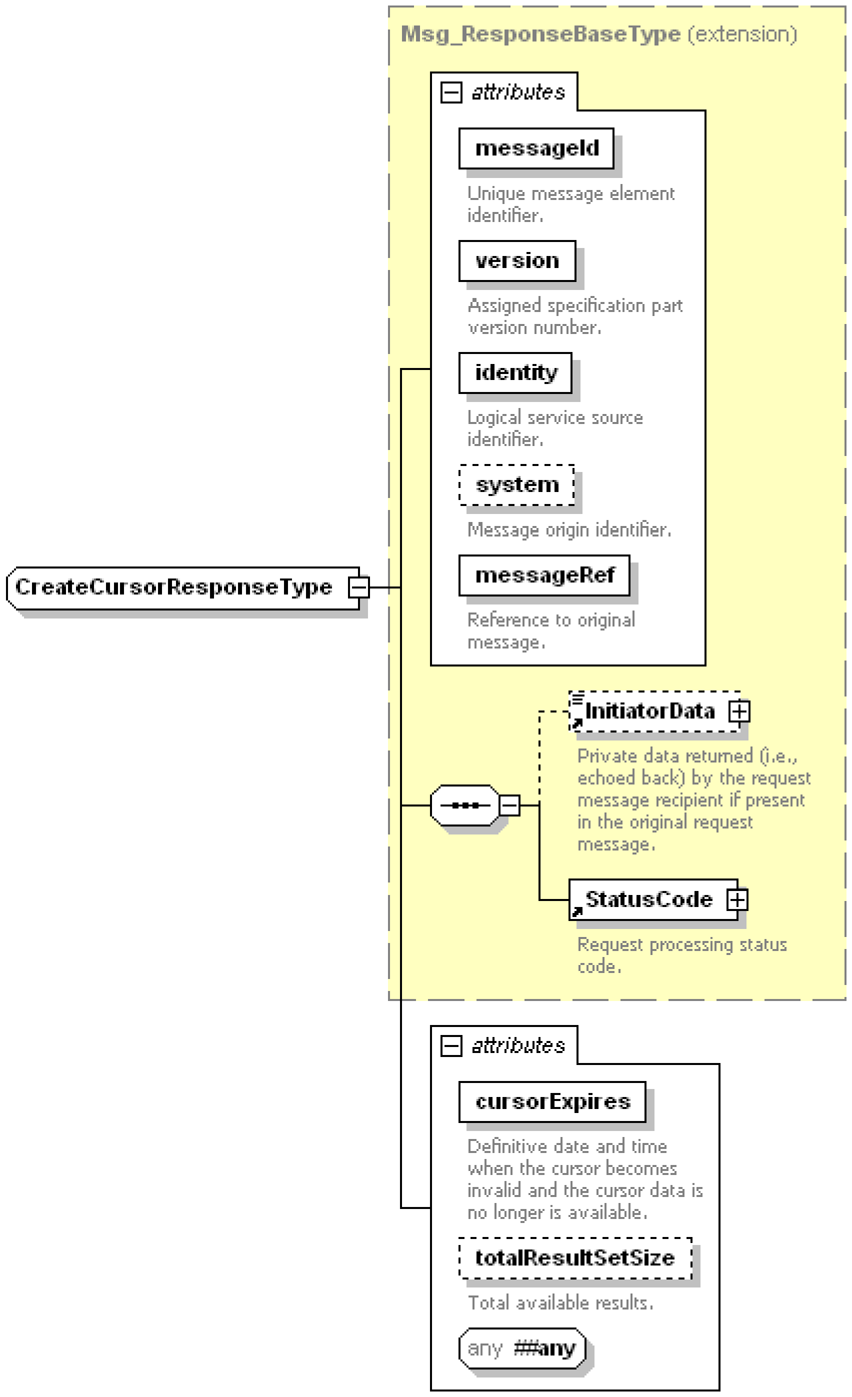

Msg_ResponseBaseType (extension)
attributes
messageId
Unique message element identifier.
version
Assigned specification part version number.
identity
Logical service source identifier.
system
Message origin identifier.
messageRef
Reference to original message.
CreateCursorResponseType
InitiatorData
Private data returned (i.e., echoed back) by the request message recipient if present in the original request message.
StatusCode
Request processing status code.
attributes
cursorExpires
Definitive date and time when the cursor becomes invalid and the cursor data is no longer is available.
totalResultSetSize
Total available results.
any ##any

Figure 15. CreateCursorResponseType XML Schema

The CreateCursorResponse message type is derived from the core namespace base type core:Msg_ResponseBaseType and defines the following attributes and elements in addition to those already defined in the core:Msg_ResponseBaseType.

@cursorExpires [Required, core:dateTimeTimezoneType] — The @cursorExpires attribute contains the logical service determined cursor expiration date and time. The value *may* be either the user requested end date and time from the CreateCursorRequest message, or a logical service specified expiration date and time. See [SCTE130-2] for additional information on the core:dateTimeTimezoneType.

@totalResultSetSize [Optional, totalResultSetSizeAttrType] — The @totalResultSetSize attribute specifies the result count contained in the cursor and the value is dependent upon the Query element's inquiry composition. This attribute ***shall*** be present if the Query element contained a basic query (i.e., the Query element contained a UniqueQualifier or BasicQueryFilter element). This attribute *may* be present if the Query element contained an advanced query (i.e., the Query element included an AdvancedQueryFilter element). If the advanced query set the Query element's @expandOutput attribute to "false", this attribute ***shall*** be present. Otherwise, this attribute's presence is optional and its value is implementation dependent. See Section 13.17 for additional information on the totalResultSetSizeAttrType type and its contents relative to the original inquiry composition.

@##any [Optional] — The @##any attribute enables a logical service incorporating the SCTE 130 Part 8 interface to optionally extend concrete messages derived from SCTE 130 Part 8 message types with attributes not specified by the schema.

11.7 CancelCursorRequest and CancelCursorResponse Message Types

A logical service ***shall*** allow a consumer to cancel an existing cursor before the expiration time has been passed.

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

Any additional communications from the consumer that reference the canceled or expired cursor ***shall*** result in an error with the core:StatusCode @detail attribute set to the value 8001 (Cursor Undefined), as described in Table 14.

11.7.1 CancelCursorRequest Message Type

This message type allows a logical service consumer to terminate a cursor before the cursor's expiration time.

The XML schema for the cancel cursor request message type is illustrated in Figure 16.

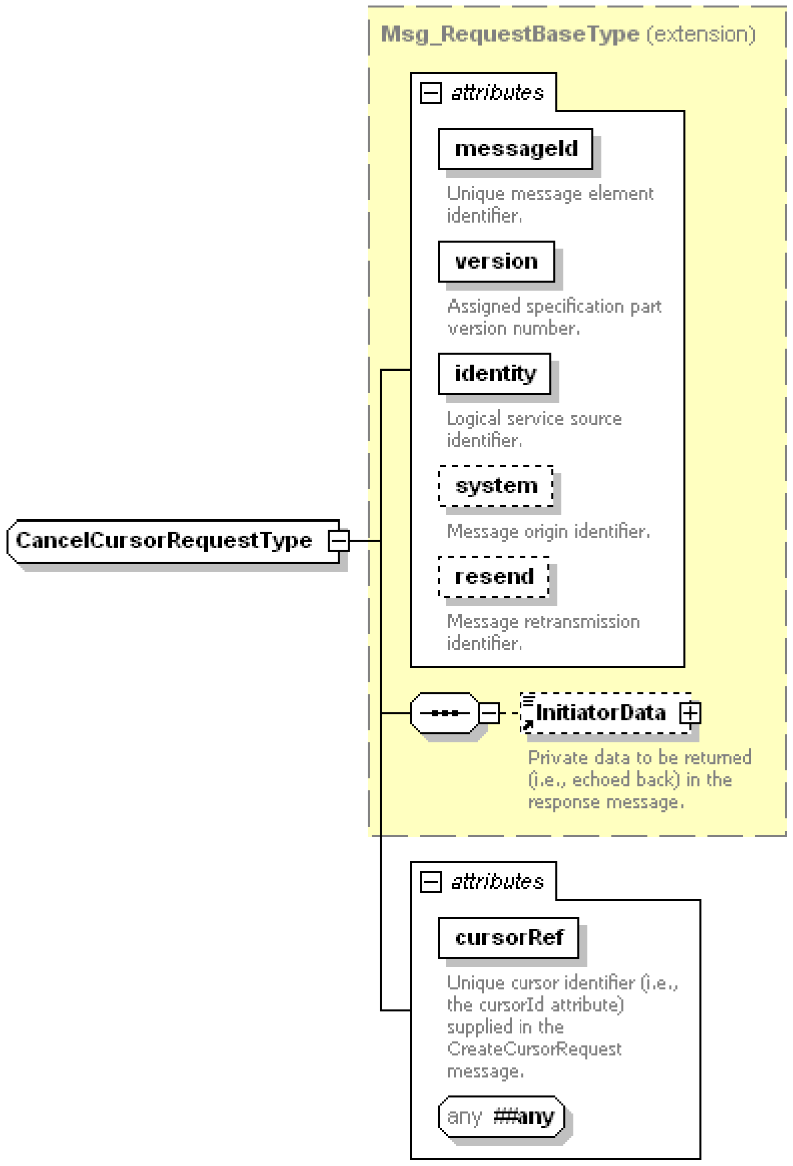

Msg_RequestBaseType (extension)
attributes
messageId
Unique message element identifier.
version
Assigned specification part version number.
identity
Logical service source identifier.
system
Message origin identifier.
resend
Message retransmission identifier.
InitiatorData
Private data to be returned (i.e., echoed back) in the response message.
CancelCursorRequestType
attributes
cursorRef
Unique cursor identifier (i.e., the cursorId attribute) supplied in the CreateCursorRequest message.
any ##any

Figure 16. CancelCursorRequestType XML Schema

The CancelCursorRequest message type is derived from the core namespace base type core:Msg_RequestBaseType and defines the following attributes and elements in addition to those already defined in the core:Msg_RequestBaseType.

@cursorRef [Required, cursorIdRefAttrType] — The value contained in @cursorRef ***shall*** be the same as the value of the @cursorId attribute in the CreateCursorRequest message used to create the cursor. The @cursorRef attribute value is used for referencing an existing cursor to the CreateCursorRequest message that created it.

For example, if a CreateCursorRequest message type used a @cursorId value of ‘123’ for a new cursor, at cancellation time for cursorId ‘123’, the CancelCursorRequest message type ***shall*** include a @cursorRef with the value ‘123’. See Section 13.2 for additional information on cursorIdRefAttrType.

@##any [Optional] — The @##any attribute enables a logical service incorporating the SCTE 130 Part 8 interface to optionally extend concrete messages derived from SCTE 130 Part 8 message types with attributes not specified by the schema.

11.7.2 CancelCursorResponse Message Type

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

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

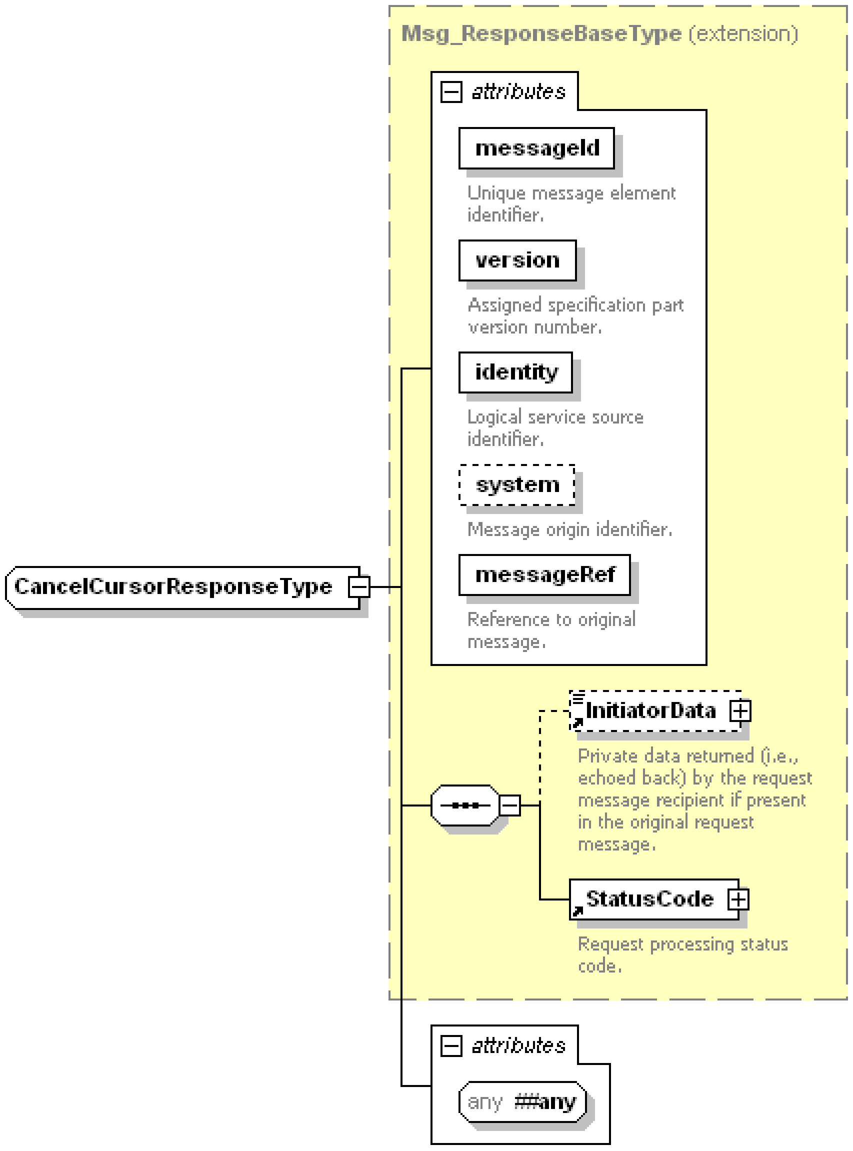

Msg_ResponseBaseType (extension)
attributes
messageId
Unique message element identifier.
version
Assigned specification part version number.
identity
Logical service source identifier.
system
Message origin identifier.
messageRef
Reference to original message.
CancelCursorResponseType
InitiatorData
Private data returned (i.e., echoed back) by the request message recipient if present in the original request message.
StatusCode
Request processing status code.
attributes
any ##any

Figure 17. CancelCursorResponseType XML Schema

The CancelCursorResponse message type is derived from the core namespace base type core:Msg_ResponseBaseType and defines no elements in addition to those already defined in core:Msg_ResponseBaseType.

@##any [Optional] — The @##any attribute enables a logical service incorporating the SCTE 130 Part 8 interface to optionally extend concrete messages derived from SCTE 130 Part 8 message types with attributes not specified by the schema.

11.8 QueryRequest and QueryResponse Message Types

The QueryRequest and QueryResponse type messages are used by consumers to initiate queries against a service data model implemented by a logical service.

These message types support both basic and advanced query mechanisms and references to existing static cursor information.

11.8.1 QueryRequest Message Type

The QueryRequest message type is the primary mechanism for a consumer to execute a query on a logical service data model. This message type contains either a Query element or a reference to a previously established Cursor element.

A QueryRequest message type containing a Query element ***shall*** have the query executed against all of the data in the referenced service data model. The query result set ***shall*** be returned in the QueryResponse message type.

A QueryRequest message containing a Cursor element ***shall*** return in a QueryResponse message all of the data components inclusive between the @startIndex and @count value within the static cursor data structure.

The QueryRequest message type XML schema definition is illustrated in Figure 18.

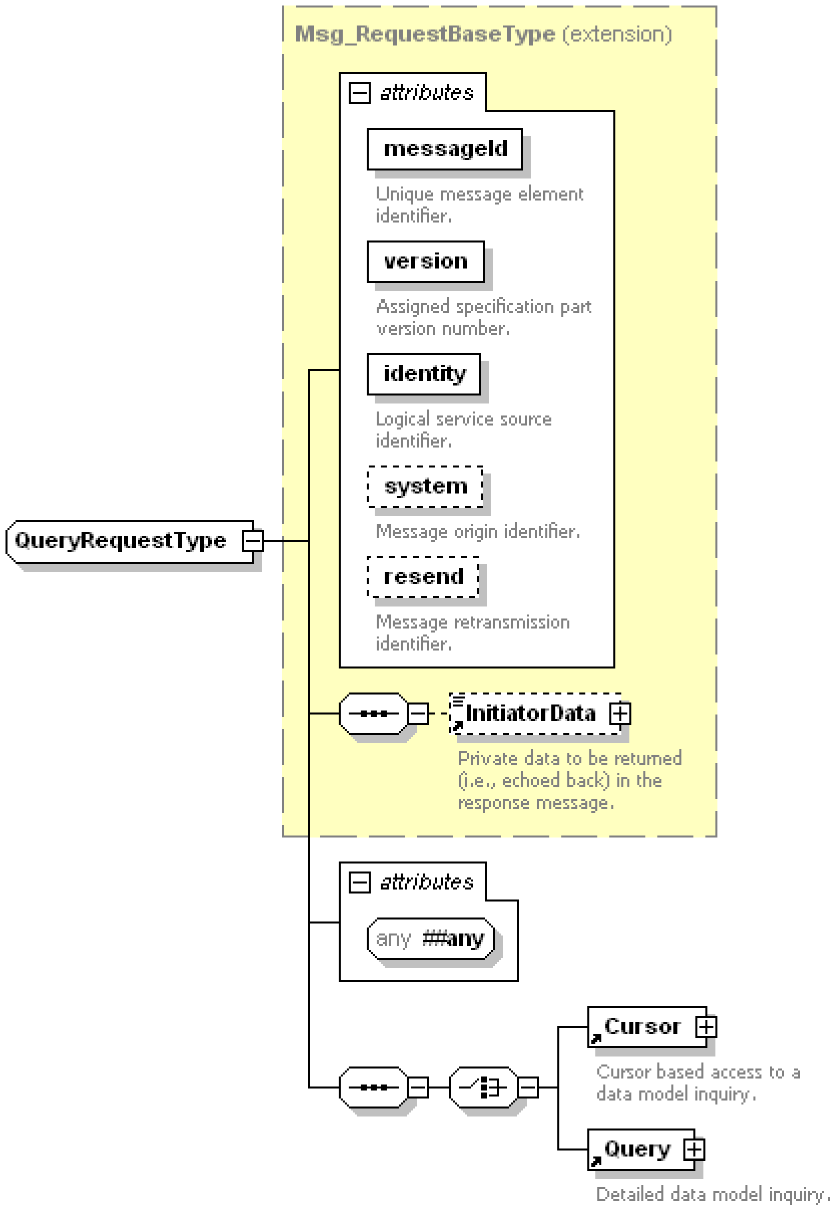

Msg_RequestBaseType (extension)
attributes
messageId
Unique message element identifier.
version
Assigned specification part version number.
identity
Logical service source identifier.
system
Message origin identifier.
resend
Message retransmission identifier.
QueryRequestType
InitiatorData
Private data to be returned (i.e., echoed back) in the response message.
attributes
any ##any
Cursor
Cursor based access to a data model inquiry.
Query
Detailed data model inquiry.

Figure 18. QueryRequestType XML Schema

The Query Request message type is derived from the core namespace base type core:Msg_RequestBaseType and defines the following attributes and elements in addition to those already defined in core:Msg_RequestBaseType.

@##any [Optional] — The @##any attribute enables a logical service incorporating the SCTE 130 Part 8 interface to optionally extend concrete messages derived from SCTE 130 Part 8 message types with attributes not specified by the schema.

Cursor [Required on choice] — The Cursor element contains a @cursorRef attribute identifying a cursor previously established using the

CreateCursorRequest message type. See Section 12.11 for details on the Cursor element.

Query [Required on choice] — The Query element contains all elements and attributes required for an asset information query. The entire result set of the query is returned in the QueryResponse message type. See Section 12.22 for additional information on the Query element.

11.8.2 QueryResponse Message Type

Upon receipt of a QueryRequest message type, a logical service that implements the GIS interface ***shall*** respond with a QueryResponse message type. This message type contains the query results (advanced, basic or cursor) in the QueryResult element.

The XML schema definition for this message is illustrated in Figure 19.

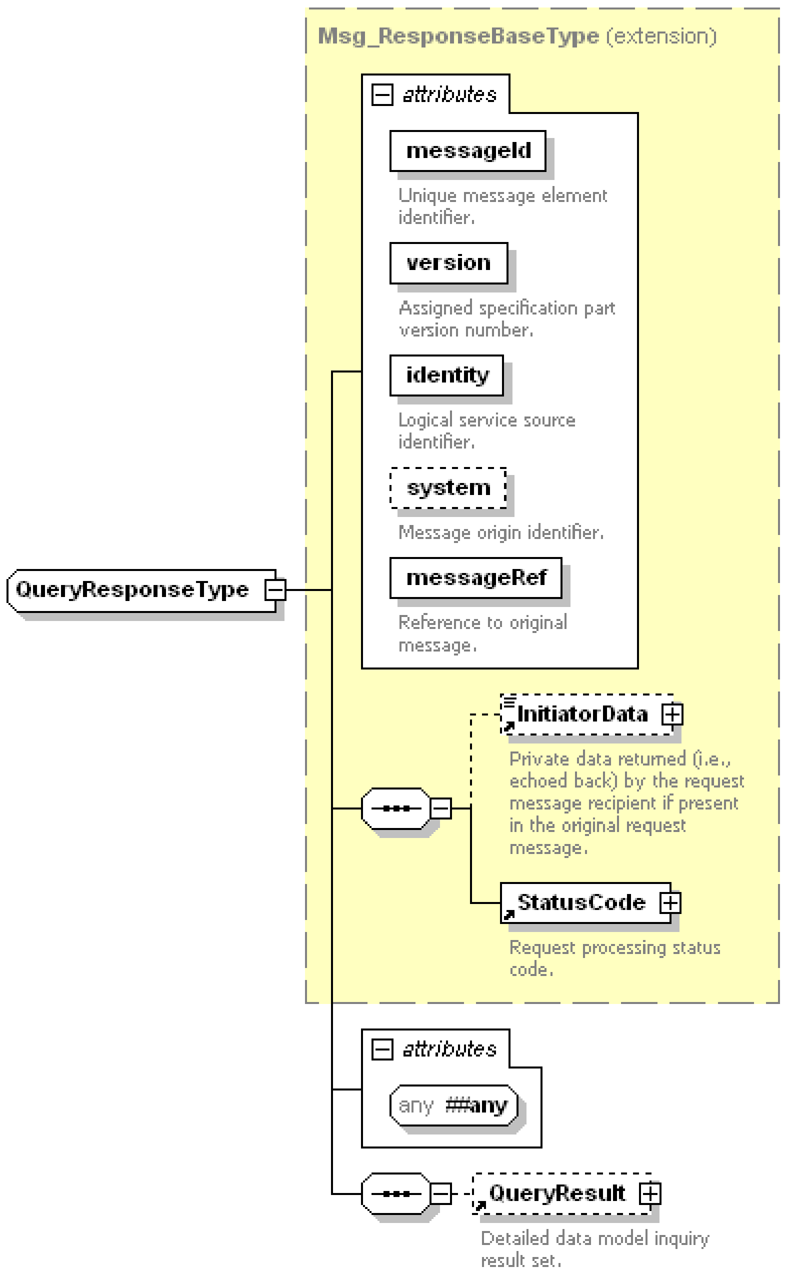

Msg_ResponseBaseType (extension)
attributes
messageId
Unique message element identifier.
version
Assigned specification part version number.
identity
Logical service source identifier.
system
Message origin identifier.
messageRef
Reference to original message.
QueryResponseType
InitiatorData
Private data returned (i.e., echoed back) by the request message recipient if present in the original request message.
StatusCode
Request processing status code.
attributes
any ##any
QueryResult
Detailed data model inquiry result set.

Figure 19. QueryResponseType XML Schema

The QueryResponse message type is derived from the core namespace base type core:Msg_ResponseBaseType and defines the following attributes and elements in addition to those already defined in core:Msg_ResponseBaseType.

@##any [Optional] — The @##any attribute enables a logical service incorporating the SCTE 130 Part 8 interface to optionally extend concrete messages derived from SCTE 130 Part 8 message types with attributes not specified by the schema.

QueryResult [Optional] — The QueryResult element contains the inquiry execution result for the query supplied in the QueryRequest element or the list of data components referenced in the Cursor element. The QueryResult element ***shall not*** be returned if the core:StatusCode element indicates an error. See Section 12.23 for additional information on the QueryResult element.

11.9 NotificationDeregisterRequest and NotificationDeregisterResponse Message Types

A logical service implementing the GIS interface ***shall*** allow a consumer to de-register a previously registered NotificationRegistrationRequest message type. This message exchange allows a logical service consumer to dynamically modify registration notifications using individual register and de-register commands.

11.9.1 NotificationDeregisterRequest Message Type

The NotificationDeregisterRequest message type removes an existing content notification registration from the logical service.

The XML schema for the NotificationDeregisterRequest type message is illustrated in Figure 20.

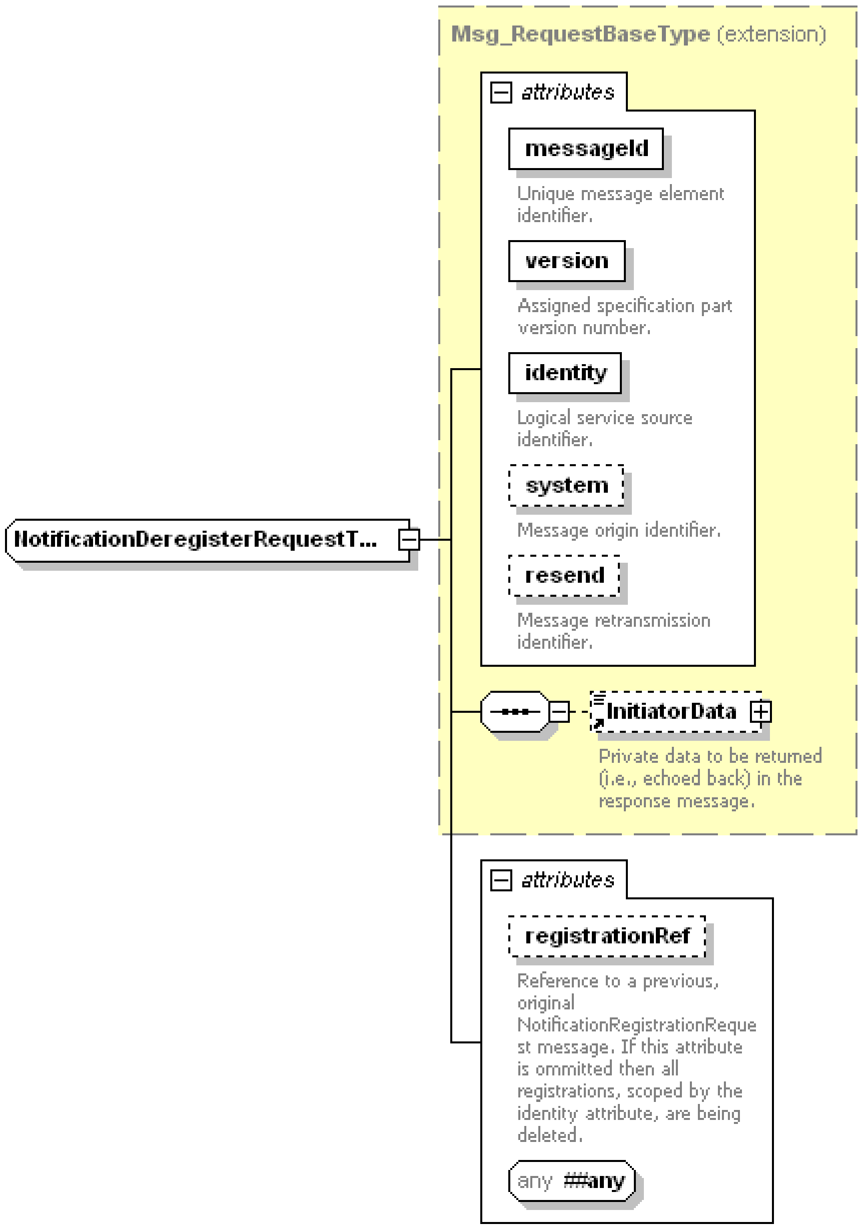

Msg_RequestBaseType (extension)
attributes
messageId
Unique message element identifier.
version
Assigned specification part version number.
identity
Logical service source identifier.
system
Message origin identifier.
resend
Message retransmission identifier.
NotificationDeregisterRequestT...
InitiatorData
Private data to be returned (i.e., echoed back) in the response message.
attributes
registrationRef
Reference to a previous, original NotificationRegistrationRequest message. If this attribute is ommitted then all registrations, scoped by the identity attribute, are being deleted.
any ##any

Figure 20. NotificationDeregisterRequestType XML Schema

The NotificationDeregisterRequest message type is derived from the core namespace base type core:Msg_RequestBaseType and defines the following attributes and elements in addition to those already defined in core:Msg_RequestBaseType.

@registrationRef [Optional, core:registrationRefAttrType] — The @registrationRef identifies the original NotificationRegistrationRequest message being deregistered. The value ***shall*** be the NotificationRegistrationRequest message's @messageId attribute value. If the

@registrationRef attribute is omitted from the message, a logical service that implements the GIS interface ***shall*** remove all notification registration request items scoped to the @identity attribute.

@##any [Optional] — The @##any attribute enables a logical service incorporating the SCTE 130 Part 8 interface to optionally extend concrete messages derived from SCTE 130 Part 8 message types with attributes not specified by the schema.

11.9.2 NotificationDeregisterResponse Message Type

Upon receipt of a NotificationDeregisterRequest message type from a consumer, a logical service that implements the GIS interface ***shall*** respond with a NotificationDeregisterResponse message type.

The XML schema for the NotificationDeregisterResponse message type is shown in Figure 21.

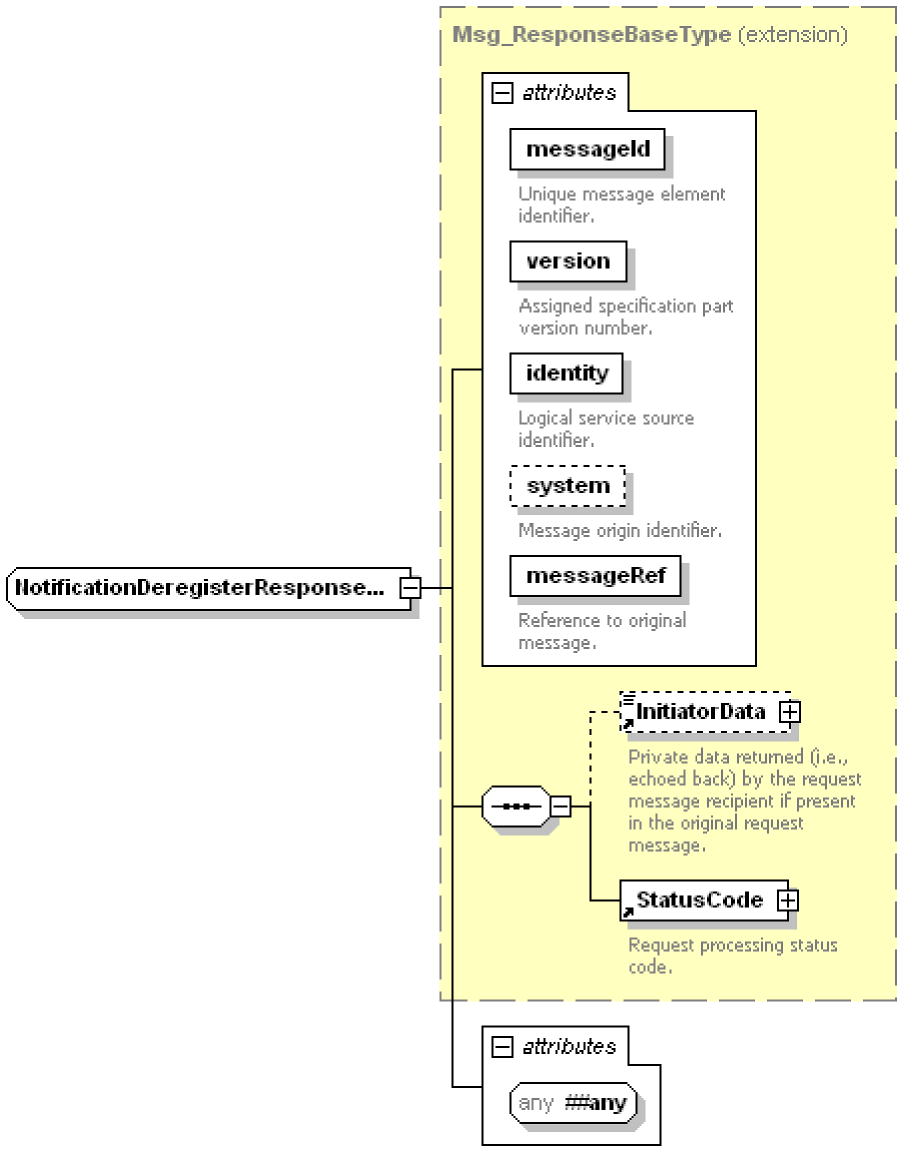

Msg_ResponseBaseType (extension)
attributes
messageId
Unique message element identifier.
version
Assigned specification part version number.
identity
Logical service source identifier.
system
Message origin identifier.
messageRef
Reference to original message.
NotificationDeregisterResponse...
InitiatorData
Private data returned (i.e., echoed back) by the request message recipient if present in the original request message.
StatusCode
Request processing status code.
attributes
any ##any

Figure 21. NotificationDeregisterResponseType XML Schema

The NotificationDeregisterResponse message type is derived from the core namespace base type core:Msg_ResponseBaseType and defines no elements other than those already defined in the core:Msg_ResponseBaseType.

@##any [Optional] — The @##any attribute enables a logical service incorporating the SCTE 130 Part 8 interface to optionally extend concrete messages derived from SCTE 130 Part 8 message types with attributes not specified by the schema.

If the sub-classed NotificationDeregisterRequest message type is unable to locate the specific registration as specified by the NotificationDeregisterRequest message type's @registrationRef attribute or by the @identity attribute, the response message type's core:StatusCode element ***shall*** contain the value core:ResourceNotFound (i.e., no matching registrations were found to deregister).

11.10 DeregistrationNotification and DeregistrationAcknowledgement Message Types

A logical service that implements the GIS interface ***shall*** have the ability to deregister consumers. Deregistration removes consumer registrations from the logical service and stops any content notification traffic from being sent to the deregistered consumer.

Upon receipt of a DeregistrationNotification message type, a GIS consumer ***shall*** reply with a DeregistrationAcknowledgement message type.

11.10.1DeregistrationNotification Message Type

At any time, the logical service *may* issue one or more DeregistrationNotification message types to registered GIS consumers. This message type informs the consumer that one or all of their active registrations (i.e., NotificationRegistrationRequest message types) have been terminated and no further notifications ***shall*** be expected related to those registrations.

The XML schema for the DeregistrationNotification message type is illustrated in Figure 22.

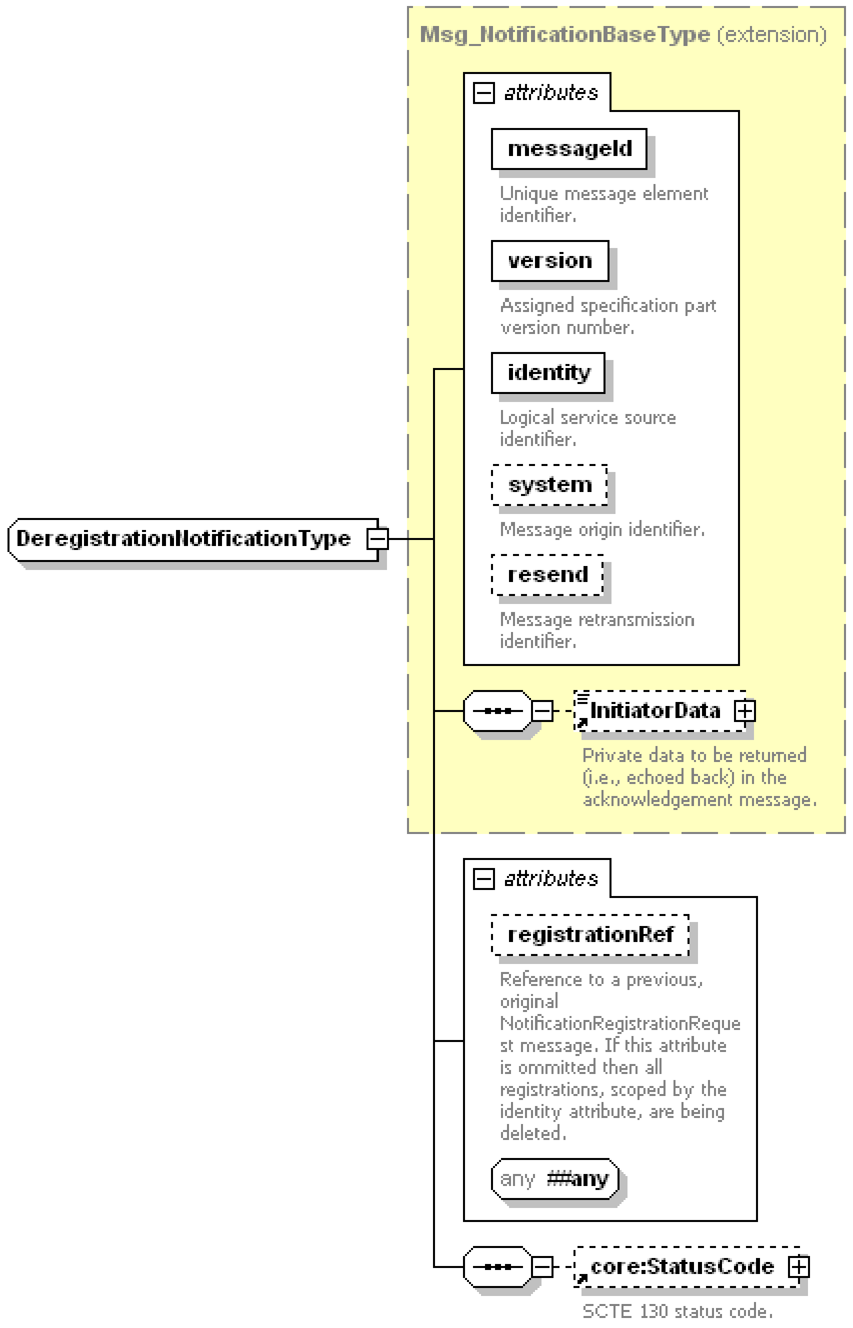

Msg_NotificationBaseType (extension)
attributes
messageId
Unique message element identifier.
version
Assigned specification part version number.
identity
Logical service source identifier.
system
Message origin identifier.
resend
Message retransmission identifier.
DeregistrationNotificationType
InitiatorData
Private data to be returned (i.e., echoed back) in the acknowledgement message.
attributes
registrationRef
Reference to a previous, original NotificationRegistrationRequest message. If this attribute is ommitted then all registrations, scoped by the identity attribute, are being deleted.
any ##any
core:StatusCode
SCTE 130 status code.

Figure 22. DeregistrationNotificationType XML Schema

The DeregistrationNotification message type is derived from the core namespace base type core:Msg_NotificationBaseType and defines the following attributes and elements in addition to those already defined in core:Msg_NotificationBaseType.

@registrationRef [Optional, core:registrationRefAttrType] — When present, this attribute identifies the original NotificationRegistrationRequest message type that ***shall*** be deregistered. The value ***shall*** be the NotificationRegistrationRequest message type's original @messageId attribute value. Issuing a DeregistrationNotification message type with this attribute informs the consumer the logical service has cleared only this registration information associated with the specific NotificationRegistrationRequest message type. If the @registrationRef attribute is absent, all registrations associated with the specified consumer identity have been deregistered.

@##any [Optional] — The @##any attribute enables a logical service incorporating the SCTE 130 Part 8 interface to optionally extend concrete messages derived from SCTE 130 Part 8 message types with attributes not specified by the schema.

11.10.2 DeregistrationAcknowledgement Message Type

Upon receipt of a DeregistrationNotification message type, a logical service consumer ***shall*** respond with a DeregistrationAcknowledgement message type. This message informs the logical service that the Notification message type was received by the intended consumer and processed.

The XML schema for the DeregistrationAcknowledgement message type is illustrated in Figure 23.

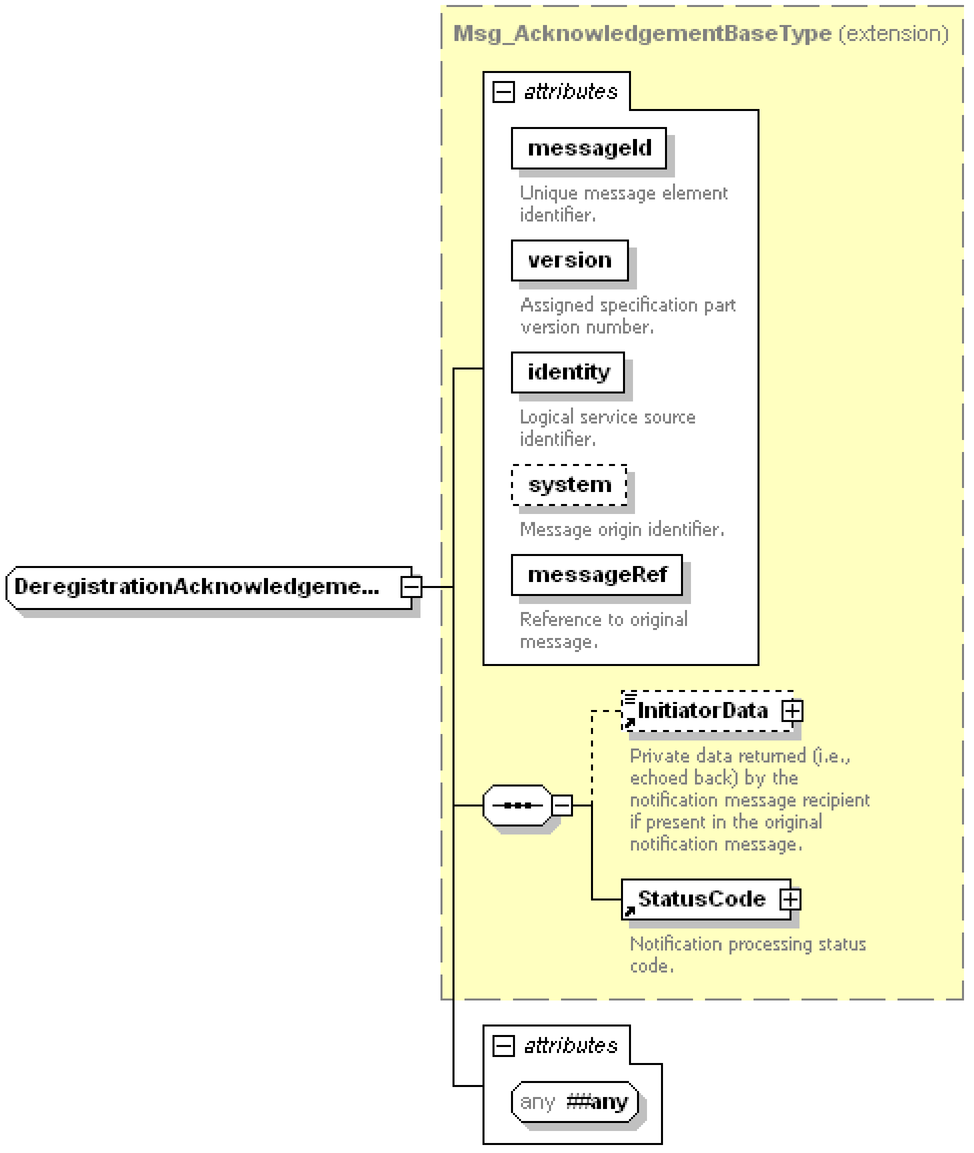

Msg_AcknowledgementBaseType (extension)
attributes
messageId
Unique message element identifier.
version
Assigned specification part version number.
identity
Logical service source identifier.
system
Message origin identifier.
messageRef
Reference to original message.
DeregistrationAcknowledgeme...
InitiatorData
Private data returned (i.e., echoed back) by the notification message recipient if present in the original notification message.
StatusCode
Notification processing status code.
attributes
any ##any

Figure 23. DeregistrationAcknowledgementType XML Schema

The DeregistrationAcknowledgement message type is derived from the core namespace base type core:Msg_AcknowledgementBaseType and defines no elements in addition to those defined in core:Msg_AcknowledgementBaseType.

@##any [Optional] — The @##any attribute enables a logical service incorporating the SCTE 130 Part 8 interface to optionally extend concrete messages derived from SCTE 130 Part 8 message types with attributes not specified by the schema.

11.11 Service Check Messages

Every logical service implementing the GIS interface ***shall*** support the ServiceCheck message exchange, which includes the core:ServiceCheckRequest and core:ServiceCheckResponse messages as defined by SCTE130-2].

11.12 Service Status Messages

Every logical service implementing the GIS interface ***shall*** support the service status message exchange, which includes the core:ServiceStatusNotification and core:ServiceStatusAcknowledgement messages as defined by [SCTE130-2].

12.0 GIS ELEMENT DETAILS

GIS elements are those that are used within the GIS top level message elements. Each of the GIS elementary messages defined in the GIS namespace are listed in Table 7 and are described in detail in subsequent document sections.

Element	**Description**
AdvancedFilterElement	Advanced query element
AdvancedQueryFilter	A sequence of AdvancedFilterElement elements forming an advanced inquiry
AdvancedQueryLanguage	Advanced query language identification
AdvancedQueryResult	Result container for advanced queries
AdvancedQueryResultData	Advance query result data return container
BasicFilterElement	Individual filter item for a basic query
BasicQueryDataModelDescription	Completely describes a logical service’s basic query data model
BasicQueryFilter	Container for a sequence of BasicFilterElement elements
BasicQueryResult	Basic query result data return container
BasicQueryResultAbstract	Abstract base type for a basic query result.
Cursor	Cursor definition
EnumerationValue	An element whose value denotes one of the allowed values that *may* be used with Qualifiers of type "enumeration"

MaxFloat	Maximum value for a qualifier with a value of type xsd:float
MaxInteger	Maximum value for a qualifier with a value of type xsd:integer
MaxLength	Maximum length value for a qualifier value
MinFloat	Minimum value for a qualifier with a value of type xsd:float
MinInteger	Minimum value for a qualifier with a value of type xsd:integer
Qualifier	A name/value pair describing one characteristic of an object
QualifierDeclaration	Identification of a qualifier's name identifier
QualifierDescription	Completely describes a qualifier e.g. name, type, lower bound, upper bound, or length
QualifierSet	A complete list of Qualifier elements for a single, uniquely identified data store object
Query	A container for elements that completely specify either a basic or advanced query
QueryResult	Result container for query
ServiceDataModel	Uniquely identifies a service data model
ServiceDataModelProfile	Contains one ServiceDataModel element and an optional sequence of supported advanced query languages
UniqueQualifier	The set of qualifiers that uniquely identifies a single entity in the service data model
UniqueQualiferDeclaration	Container specifying the QualifierDeclaration element sequence forming the basis for a unique qualifier

Table 7. GIS Elementary Element Details

A logical service implementing the GIS interface *may* support basic query processing and *may* support advanced query language processing or it *may* support both basic and advanced query processing.

If a GIS implementation supports advanced query language processing against an XML service data model, the implementation ***shall*** support one or both of the [SCTE130-7] or [W3C–XQuery] query languages listed in Table 8. Any query language value reference to query languages listed in Table 8 ***shall*** appear exactly as shown in Table 8 (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 8. Advanced Query Languages

12.1 AdvancedFilterElement

The AdvancedFilterElement contains a query identifier, a query language identifier and either a free form query string, typically encapsulated in an XML CDATA section, or an XML element sequence associated with a private query language, which are executed by the specified query language processor.

The XML schema diagram for this message is as follows in Figure 24.

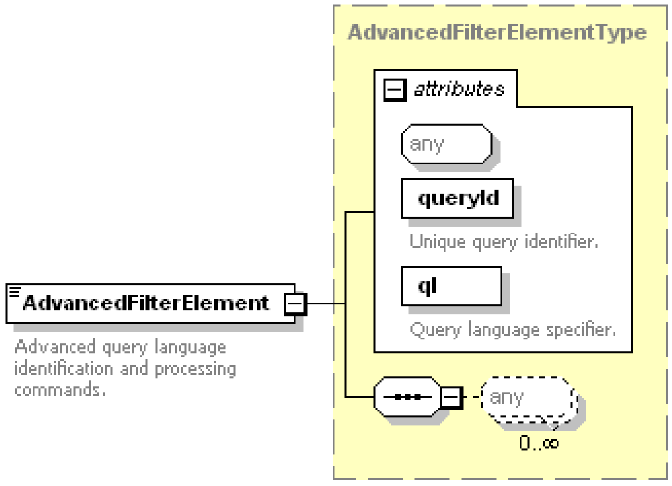

AdvancedFilterElementType
attributes
any
queryId
Unique query identifier.
ql
Query language specifier.
AdvancedFilterElement
Advanced query language identification and processing commands.
any
0..∞

Figure 24. AdvancedFilterElement Element XML Schema

The AdvancedFilterElement contains the following attributes:

@queryId [Required, 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. See Section 13.13 for additional information.

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.

@ql [Required, queryLanguageAttrType] — The @ql attribute identifies the specific query language engine that ***shall*** be used to process the query contained within the AdvancedFilterElement. See Section 13.15 for additional information on queryLanguageAttrType. See Table 8 for a list of allowed values for the @ql attribute.

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

The AdvancedFilterElement element's value ***shall*** contain CDATA encapsulated query language specific data or any type of markup content including an XML element sequence (allowed by the xsd:any element sequence). When either XPath or XQuery is the specified query language, the element's data ***shall*** be CDATA encapsulated. The AdvanceFilterElement element's value ***shall not*** be empty.

12.2 AdvancedQueryFilter

The AdvancedQueryFilter element is a container for AdvancedFilterElement elements. The AdvancedFilterElement items within a single AdvancedQueryFilter define a complete query that ***shall*** be applied to all assets referenced by a logical service implementing the GIS interface.

Multiple AdvancedFilterElement elements within a single AdvancedQueryFilter 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 AdvancedQueryFilter element is shown in Figure 25.

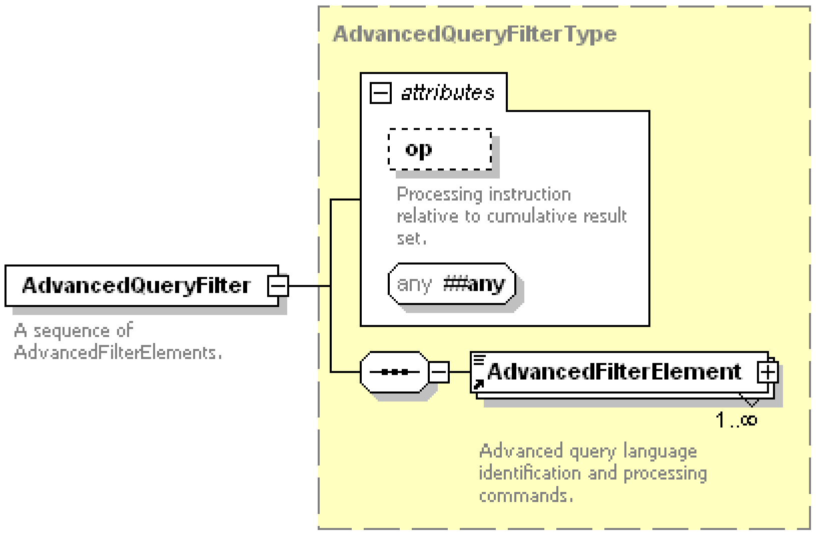

AdvancedQueryFilterType
attributes
op
Processing instruction relative to cumulative result set.
any ##any
AdvancedQueryFilter
A sequence of AdvancedFilterElements.
AdvancedFilterElement
1..∞
Advanced query language identification and processing commands.

Figure 25. AdvancedQueryFilter Element XML Schema

The AdvancedQueryFilter element has the following attributes:

@op [Optional, QueryFilterOpTypeEnumeration] — The @op attribute instructs the logical service how to process this part of the overall query with respect to the cumulative result sets from other AdvancedFilterElement elements within the same parent element. The accepted values for the @op attribute are described in Table 11. When not present, the default value for the @op attribute is 'include'. See Section 13.12 for additional information.

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

The AdvancedQueryFilter ***shall*** contain one or more of the following elements:

AdvancedFilterElement [Required] — The AdvancedFilterElement contains a detailed query language expression that ***shall*** be executed against the service 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 data element referenced by the logical service's data model and, depending on the value of the @op attribute, the results are added to or subtracted from the net result set. See Section 12.1 for additional information.

12.3 AdvancedQueryLanguage

The AdvancedQueryLanguage element's value is defined as type core:nonEmptyStringType. This element returns to the caller the types of advanced query languages supported by a logical service that incorporates the GIS interface. See Table 8 for additional information on supported advanced query languages.

The XML schema definition for the AdvancedQueryLanguage element is shown in Figure 26.

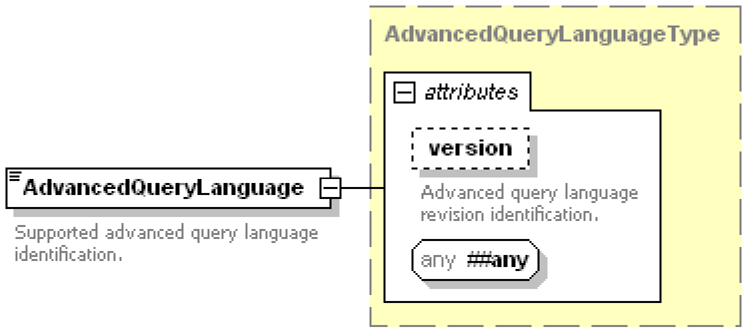

AdvancedQueryLanguageType
attributes
version
Advanced query language revision identification.
any ##any
AdvancedQueryLanguage
Supported advanced query language identification.

Figure 26. AdvancedQueryLanguage Element XML Schema

@version [Optional, core:nonEmptyStringType] — The @version attributed contains version information for the language type specified within the AdvancedQueryLanguage element.

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

The AdvancedQueryLanguage element's value ***shall*** be a value from Table 8.

12.4 AdvancedQueryResult

The XML schema definition for this element is shown in Figure 27.

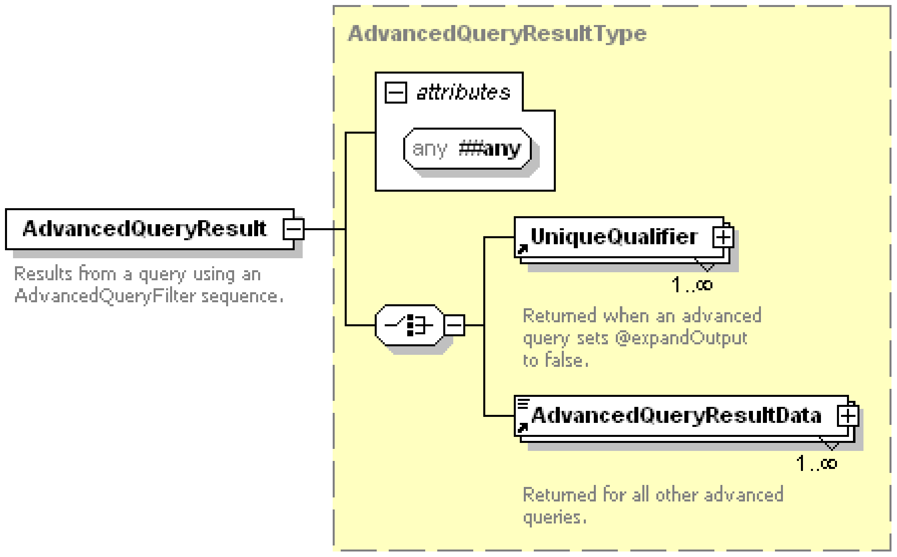

AdvancedQueryResultType
attributes
any ##any
AdvancedQueryResult
Results from a query using an AdvancedQueryFilter sequence.
UniqueQualifier
1..∞
Returned when an advanced query sets @expandOutput to false.
AdvancedQueryResultData
1..∞
Returned for all other advanced queries.

Figure 27. AdvancedQueryResult Element XML Schema

When the Query element's @expandOutput attribute is set to "true", the results from an advanced query, as specified in an AdvancedFilterElement element, are returned to the consumer using one or more AdvancedQueryResultData elements. The results ***shall*** be returned without intermediate formatting (i.e., as is).

When the @expandOutput attribute of the Query message is set to "false", the results from an advanced query, as specified in an AdvancedFilterElement element, are returned to the consumer using a UniqueQualifier element sequence. If a service data model supports the advanced query interface and also supports the basic query interface, the service data model's schema and UniqueQualifier set *may* be discovered using the ListQualifiersRequest type message. If a service data model supports the advanced query interface but does not support the basic query interface, discovery of the service data model's schema is outside the scope of this specification.

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

The AdvancedQueryResult element contains the following attributes and elements.

UniqueQualifer [Required Under Choice] — A sequence of one or more UniqueQualifier elements returned when the Query element's @expandOutput attribute is set to the value "false". For more information on the UniqueQualifier element, see Section 12.24.

AdvancedQueryResultData [Required Under Choice] — A sequence of one or more AdvancedQueryResultData elements providing the query language specific inquiry results and are returned when the Query element's @expandOutput attribute is set to the value "true.". For more information on the AdvanceQueryResutData element see Section 12.5.

12.5 AdvancedQueryResultData

The AdvancedQueryResultData element contains either CDATA encapsulated advanced query results or a private query resultant XML element sequence.

The XML schema definition for the AdvancedQueryResultData element is shown in Figure 28.

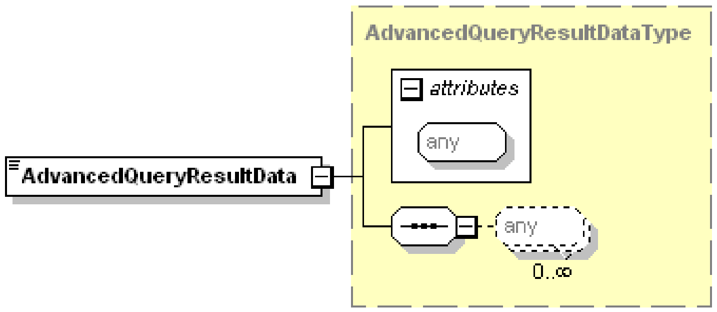


Figure 28. AdvancedQueryResultData Element XML Schema

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

The AdvancedQueryResultData element's value ***shall*** contain either CDATA encapsulated advanced query results or an XML element sequence which is specific to a private query language. The AdvancedQueryResultData element's value ***shall not*** be empty. The CDATA encapsulate results ***shall*** be returned without intermediate formatting (i.e., as is).

12.6 BasicFilterElement

Each BasicFilterElement contains a name and value attribute forming a portion of a basic query. The @name attributes refers to a particular QualifierDeclaration @name attribute located in the logical service's supported data model. The @value attribute refers to the value of the named Qualifier. The @value attribute ***shall*** support regular expression processing as defined in Section 14.1.

The XML schema diagram for the BasicFilterElement is shown in Figure 29.

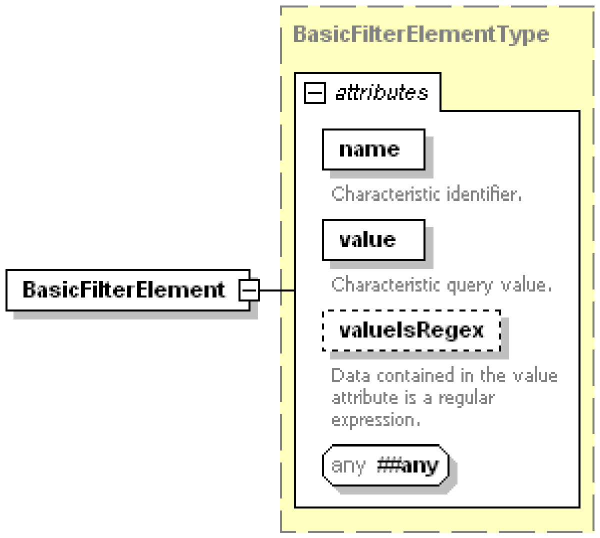

BasicFilterElementType
attributes
name
Characteristic identifier.
value
Characteristic query value.
valueIsRegex
Data contained in the value attribute is a regular expression.
any ##any
BasicFilterElement

Figure 29. BasicFilterElement Element XML Schema

The BasicFilterElement contains the following attributes:

@name [Required, filterElementNameAttrType] — The @name attribute contains the service data model qualifier name for a specific object characteristic from the specified service data model and ***shall not*** be empty. See Section 13.4 for additional information.

@value [Required, filterElementValueAttrType] — The @value attribute contains the qualifier value that ***shall*** be matched against the service data model named object characteristic specified by the @name attribute. The @value attribute ***shall*** support regular expression processing as defined in Section 14.1. See Section 13.5 for additional information on the filterElementValueAttrType type.

@valueIsRegex [Optional, xsd:boolean] — The @valueIsRegex attribute indicates whether the contents of the @value attribute *should* be treated as a regular expression. A value of "true" indicates the @value attribute contents *should* be treated as a regular expression. A value of "false" indicates the @value attribute contents *should not* be treated as a regular expression. If the optional @valueIsRegex attribute is omitted, the default value ***shall*** be "false".

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

The BasicFilterElement element's value ***shall not*** be used.

12.7 BasicQueryDataModelDescription

The BasicQueryDataModelDescription element describes a service data model supported by a logical service that implements the GIS interface and that *may* be queried using the basic query mechanism. The service data model description includes the service data model identifier, the declarations of all unique qualifiers for the service data model, and individual qualifier descriptions for all qualifiers contained in the service data model. The XML schema for the BasicQueryDataModelDescription element is shown in Figure 30.

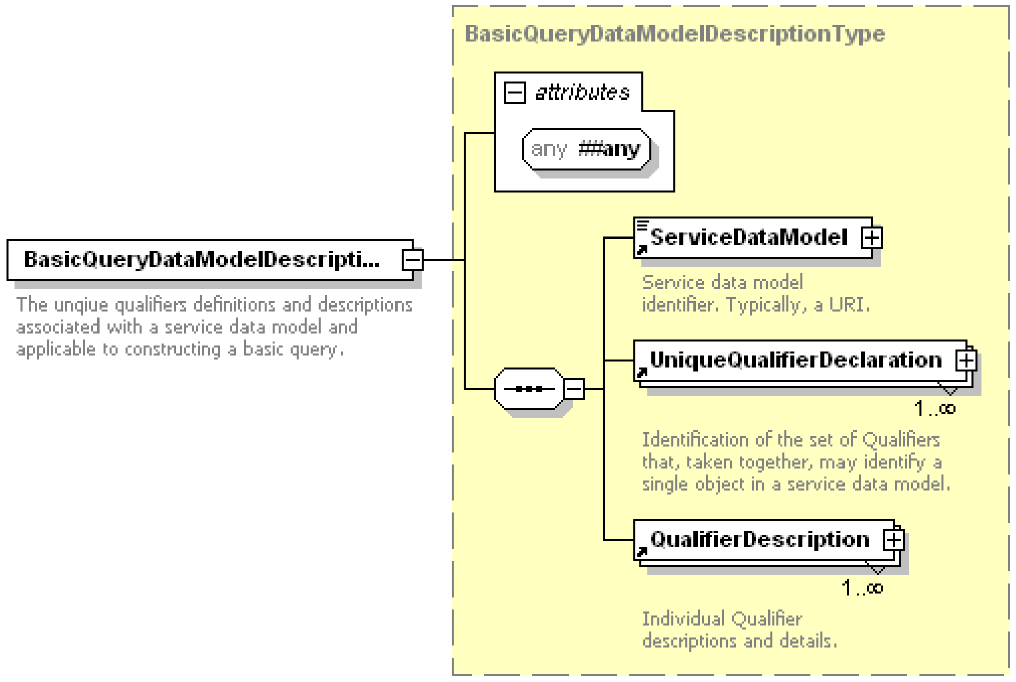

BasicQueryDataModelDescriptionType
attributes
any ##any
BasicQueryDataModelDescripti...
The unqiue qualifiers definitions and descriptions associated with a service data model and applicable to constructing a basic query.
ServiceDataModel
Service data model identifier. Typically, a URI.
UniqueQualifierDeclaration
1..∞
Identification of the set of Qualifiers that, taken together, may identify a single object in a service data model.
QualifierDescription
1..∞
Individual Qualifier descriptions and details.

Figure 30. BasicQueryDataModelDescription Element XML Schema

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

ServiceDataModel [Required] — Contains a non-empty string (usually a URI) that uniquely identifies the service data model. For more information on the ServiceDataModel element, see Section 12.24.

UniqueQualiferDeclaration [Required] — One or more UniqueQualifierDeclaration elements enumerating the qualifier declarations comprising a unique qualifier. If a service data model supports more than unique qualifier declaration, each UniqueQualiferDeclaration element enumerates the qualifiers specific to a single unique qualifier declaration. For each UniqueQualifierDeclaration element and all QualifierDeclaration elements appearing within the UniqueQualifierDeclaration element, each QualifierDeclaration element ***shall*** have a matching QualifierDescription element in the QualifierDescription sequence. Furthermore, the first UniqueQualifierDeclaration listed is referred to as the default unique qualifier declaration and ***shall*** be used when no specific UniqueQualifierDeclaration reference is provided. For more information on the UniqueQualifierDeclaration element see Section 12.27.

QualifierDescription [Required] — A sequence of one or more QualifierDescription elements each describing the characteristics of one qualifier in the service data model. At a minimum, the QualifierDescription element sequence ***shall*** contain descriptions for each QualifierDeclaration element in the UniqueQualifierDeclaration element and the QualifierDescription sequence *may* contain more than this mandatory minimum set. For more information on the QualifierDescription element see Section12.20.

12.8 BasicQueryFilter

The BasicQueryFilter element is a container for BasicFilterElement elements. The BasicFilterElement items within a single BasicQueryFilter element define a complete query that ***shall*** be applied to all assets referenced by a logical service that implements the GIS interface on query execution.

Multiple BasicFilterElement elements within a single BasicQueryFilter 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 BasicQueryFilter element is shown in Figure 31.

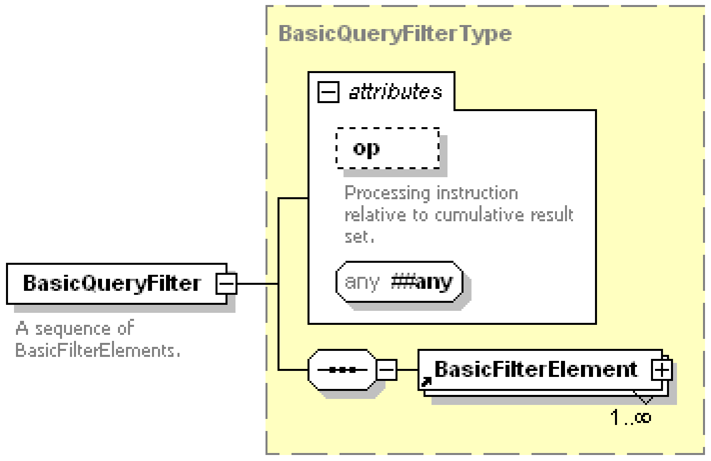

BasicQueryFilterType
attributes
op
Processing instruction relative to cumulative result set.
any ##any
BasicQueryFilter
A sequence of BasicFilterElements.
BasicFilterElement
1..∞

Figure 31. BasicQueryFilter Element XML Schema

The BasicQueryFilter element has the following attributes:

@op [Optional, QueryFilterOpTypeEnumeration] — The @op attribute instructs the logical service how to process this part of the overall query with respect to the cumulative result sets from other BasicFilterElement elements within the same parent element. The accepted values for the @op attribute are described in Table 11. The default value for the @op attribute is 'include' when the @op attribute is omitted.

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

The BasicQueryFilter element ***shall*** contain one or more of the following elements:

BasicFilterElement [Required] — The BasicFilterElement contains name/value pair attributes that indicate to the logical service which metadata names and values for an asset *should* be evaluated in the query. The BasicQueryFilter element *may* contain one or more BasicFilterElement elements.

When more than one BasicFilterElement is included within a BasicQueryFilter, a logical service ***shall*** combine (AND together) the BasicFilterElement elements into a single query statement. Only records qualifying against all of the BasicFilterElement elements contained within a single BasicQueryFilter ***shall*** be part of the 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 BasicQueryFilter element.

12.9 BasicQueryResult

The BasicQueryResult element is a substitution element for the BasicQueryResultAbstract abstract element returned as a response to a Query element containing either a UniqueQualifier element or one or more BasicQueryFilter elements. A BasicQueryResult element ***shall*** contain either a QualifierSet or a UniqueQualifier element sequence. This element's usage as a substitute for the BasicQueryResultAbstract element is at the discretion of the including advertising service specification.

The XML schema for the BasicQueryResult is shown in Figure 32.

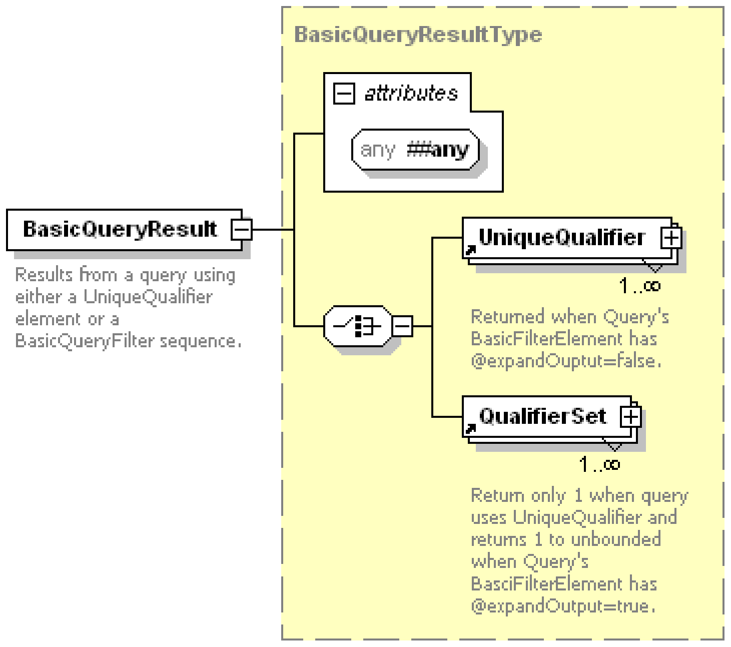

BasicQueryResultType
attributes
any ##any
BasicQueryResult
Results from a query using either a UniqueQualifier element or a BasicQueryFilter sequence.
UniqueQualifier
1..∞
Returned when Query's BasicFilterElement has @expandOuptut=false.
QualifierSet
1..∞
Return only 1 when query uses UniqueQualifier and returns 1 to unbounded when Query's BasciFilterElement has @expandOutput=true.

Figure 32. BasicQueryResult Element XML Schema

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

UniqueQualifier [Required by Choice] — One or more UniqueQualifier elements are returned as a response to a Query element containing one or more BasicFilterElement elements and the Query element has the @expandOutput attribute set to the value "false".

QualifierSet [Required by Choice] — One or more QualifierSet elements are returned as a response to a basic query when the originating Query element contains either a UniqueQualifer element or the combination of a BasicFilterElement element sequence in conjunction with Query element's @expandOutput attribute being set to the value "true".

When the initiating Query element contains a UniqueQualifer element, only one QualiferSet element ***shall*** be returned. If the inquiry result count evaluated to more than one, the query *should* be failed and a core:StatusCode element ***shall*** indicate the error condition using the value 8003 (Malformed Unique Qualifier). Note: By definition, a unique qualifier ***shall*** resolve to one and only one object in the service data model. Thus, more than one result is considered an error.

One or more QualiferSet elements ***shall*** be returned when the Query element contains a BasicFilterElement and the Query element's @expandOutput attribute is set to the value "true".

For additional information on the QualiferSet element see Section 12.21. For additional details regarding the inquiry result return see the QueryResult elelment in Section 12.23.

12.10 BasicQueryResultAbstract

The BasicQueryResultAbstract element is an abstract element mandating an XML substitution be defined. The substitution element ***shall*** be an XML extension of the BasicQueryResultAbstractType and ***shall*** be declared as a XML substitution group by including the XML attribute xsi:substitutionGroup="BasicQueryResultAbstract". The BasicQueryResult element, defined within this specification, is a concrete element instantiation conforming to these requirements. Other specifications *may* extend or define new, strongly typed elements to return their unique data model specific query results.

The XML schema for the BasicQueryResultAbstract is shown in Figure 33.

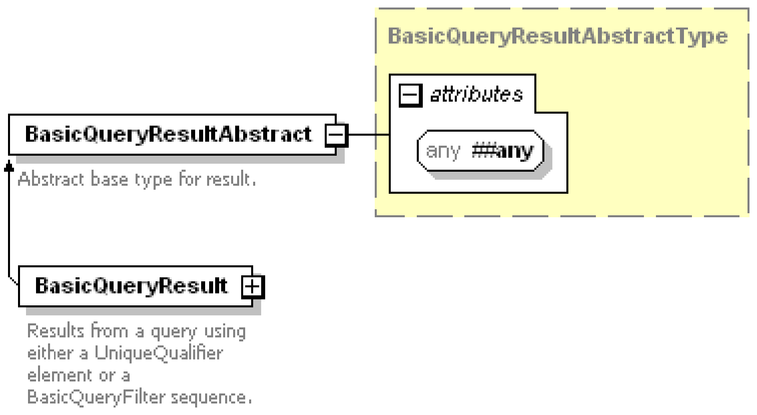

BasicQueryResultAbstractType
attributes
any ##any
BasicQueryResultAbstract
Abstract base type for result.
BasicQueryResult
Results from a query using either a UniqueQualifier element or a BasicQueryFilter sequence.

Figure 33. BasicQueryResultAbstract Element XML Schema

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

12.11 Cursor

A logical service implementing the GIS interface ***shall*** support the ability for consumer to create and reference static, cursor-based data for both basic and advanced queries. Cursors are lists of static data elements data with a limited accessibility lifetime. Once established, the data in a cursor ***shall not*** change. A cursor's accessibility end date and time is initially specified by the consumer using the CreateCursorRequest message type's @cursorExpires attribute, and finally established by the logical service via the @cursorExpires attribute in the CreateCursorResponse message type. The logical service *may* choose to destroy a cursor and reclaim system resources at any time in order to ensure overall system health.

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

Upon receipt of a CreateCursorRequest message type, a logical service ***shall*** create a new static result set and commence a countdown on the expiration duration for the cursor. If an existing cursor with the same @cursorId value already exists, the logical service ***shall*** generate an error and return the error 8002 (Cursor Already Exists) in the core:StatusCode element of the CreateCursorResponse message type. See Appendix A, for additional details.

The XML schema for the Cursor element is as follows in Figure 34.

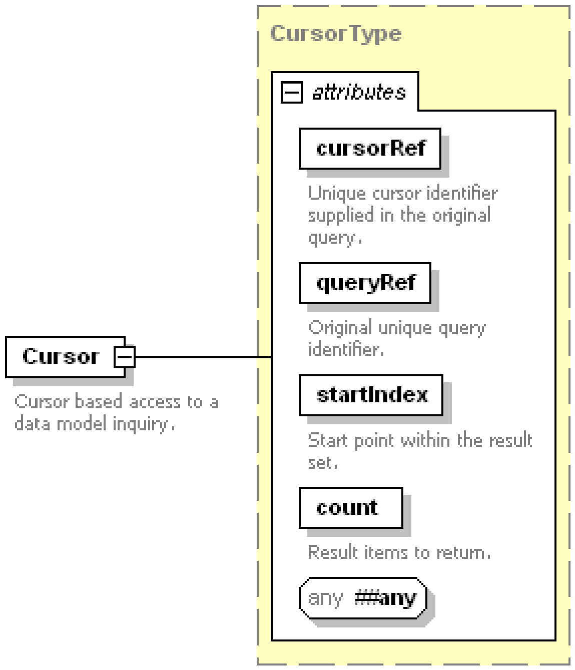

CursorType
attributes
cursorRef
Unique cursor identifier supplied in the original query.
queryRef
Original unique query identifier.
startIndex
Start point within the result set.
count
Result items to return.
any ##any
Cursor
Cursor based access to a data model inquiry.

Figure 34. Cursor Element XML Schema

The Cursor element defines the following attributes:

@cursorRef [Required, cursorIdRefAttrType] — The value contained in @cursorRef ***shall*** be the same as the value of the @cursorId attribute in the CreateCursorRequest message type used to create the cursor. The @cursorRef attribute value is used for referencing an existing cursor to the CreateCursorRequest message type that created it.

For example, if a CreateCursorRequest message type used a @cursorId value of '123' for a new cursor, at cancellation time for cursorId '123', the CancelCursorRequest message type ***shall*** include a @cursorRef with the value '123'. See Section 13.2 for additional information on cursorIdRefAttrType type.

@queryRef [Required, queryIdRefAttrType] — The value of the @queryRef attribute ***shall*** be the same as the value of the @queryId attribute used in the QueryRequest message type that initiated the query. The @queryRef attribute is used to reference a QueryResult element to the Query element that initiated the query. See Section 13.13 for further information on the queryIdRefAttrType type.

@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.

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

The Cursor element's value ***shall not*** be used.

A logical service ***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 logical service ***shall*** successfully return the remaining records in the static data list.

12.12 EnumerationValue

The EnumerationValue element returns one of the allowed values for a Qualifier element declared to be of type "enumeration".

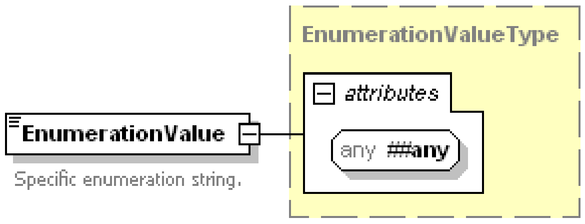


Figure 35. EnumerationValue Element XML Schema

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

The EnumerationValue element's value is of type core:nonEmptyStringType and the value ***shall*** contain a non-empty string that *may* be used as a value in the @value attribute of a Qualifier element of type "enumeration".

12.13 MaxFloat

The MaxFloat element specifies the maximum value for a qualifier declared to be of type "float".

Figure 36 illustrates the MaxFloat element's schema.

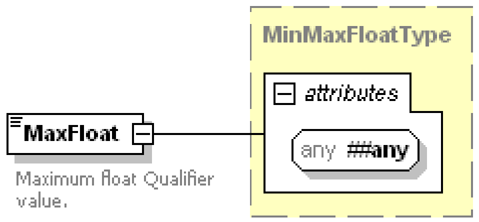


Figure 36. MaxFloat Element XML Schema

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

The MaxFloat element's value is of type MinMaxFloatType. See Section 13.5 for additional information.

12.14 MaxInteger

The MaxInteger element specifies the maximum value for a qualifier declared to be of type "integer".

Figure 37 illustrates the MaxInteger element's schema.

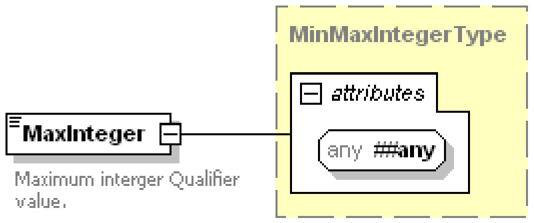


Figure 37. MaxInteger Element Xml Schema

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

The MaxInteger element's value is of type MinMaxIntegerType . See Section 13.7 for additional information.

12.15 MaxLength

The MaxLength element specifies the maximum length value for a qualifier.

Figure 38 illustrates the MaxLength element's schema.

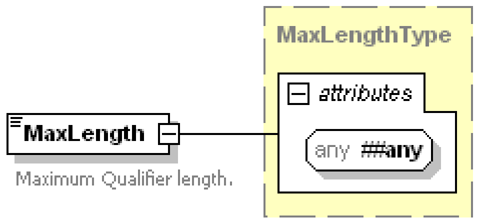


Figure 38. MaxLength Element XML Schema

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

The MaxLength element's value is of type MaxLengthType. See Section 13.8 for additional information.

12.16 MinFloat

The MinFloat element specifies the minimum value for a qualifier declared to be of type "float".

Figure 39 illustrates the MinFloat element's schema.

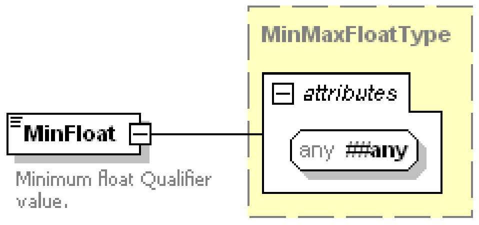


Figure 39. MinFloat Element XML Schema

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

The MinFloat element's value is of type MinMaxFloatType. See Section 13.5 for additional information.

12.17 MinInteger

The MinInteger element specifies the minimum value for a qualifier declared to be of type "integer".

Figure 40 illustrates the MinInteger element's schema.

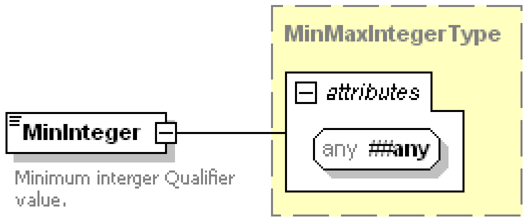


Figure 40. MinInteger Element XML Schema

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

The MinInteger element's value is of type MinMaxIntegerType. See Section 13.7 for additional information.

12.18 Qualifier

The Qualifier element is a name/value pair used in the context of a logical service's data model. Qualifiers *may* be used to query a service data model or *may* be returned as the result of an inquiry. Each name/value pair identifies one object characteristic and a collection of Qualifier elements *may* describe multiple unrelated object characteristics. A sequence of Qualifier elements which uniquely identify a single object in the service data model's data store is referred to as a unique qualifier. See Section 12.24 for additional information on the UniqueQualifier element.

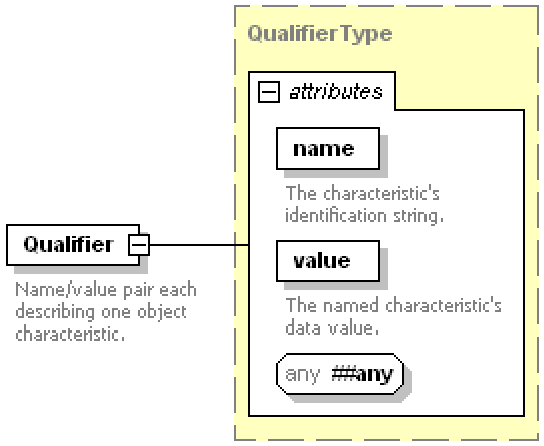

QualifierType
attributes
name
The characteristic's identification string.
Qualifier
value
The named characteristic's data value.
Name/value pair each describing one object characteristic.
any ##any

Figure 41. Qualifier Element XML Schema

@name [Required, qualifierNameAttrType] — A non-empty string uniquely identifying an object characteristic in the service data model. Values for the @name attribute are typically obtained using the list qualifiers message type exchange. See Section 11.2 for additional information on the list qualifiers message type exchange and see Section 13.10 for details on the qualifierNameAttrType type.

@value [Optional, core:nonEmptyStringType] — A value associated with the service data model object characteristic identified by the @name attribute. The @value attribute contents *should* conform to the associated QualifierDescription element obtained using the list qualifiers message type exchange. See Section 11.2 for additional information on the list qualifiers message type exchange and see Section 12.20 for information regarding the QualifierDescrption element.

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

The Qualifier element's value ***shall not*** be used.

12.19 QualifierDeclaration

The QualifierDeclaration element contains an object characteristic identification string used for example in a Qualifier element.

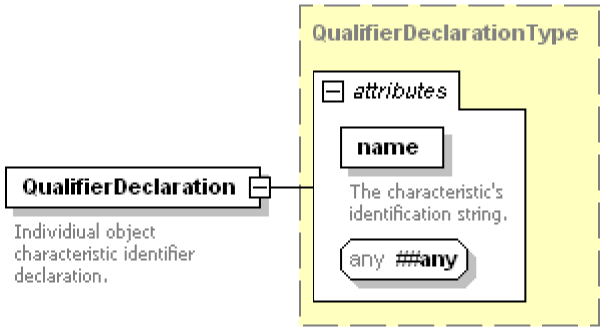

QualifierDeclarationType
attributes
name
The characteristic's identification string.
any ##any
QualifierDeclaration
Individiual object characteristic identifier declaration.

Figure 42. QualifierDeclaration Element XML Schema

@name [Required, qualifierNameAttrType] — A non-empty string declaring the existence of a unique identifier for an object characteristic in the service data model. See Section 13.10 for additional details.

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

The QualifierDeclaration element's value ***shall not*** be used.

12.20 QualifierDescription

The QualifierDescription element describes the value type and characteristic constraints for a given qualifier. The element includes the qualifier name identifier (the @name attribute), the qualifier type (the @valueType attribute), and constraints. The constraints are typically the min and max values for integer and float value types, the max length value for string types, or a list of allowed values for enumeration types. The XML schema for the QualifierDescription element is shown in Figure 43.

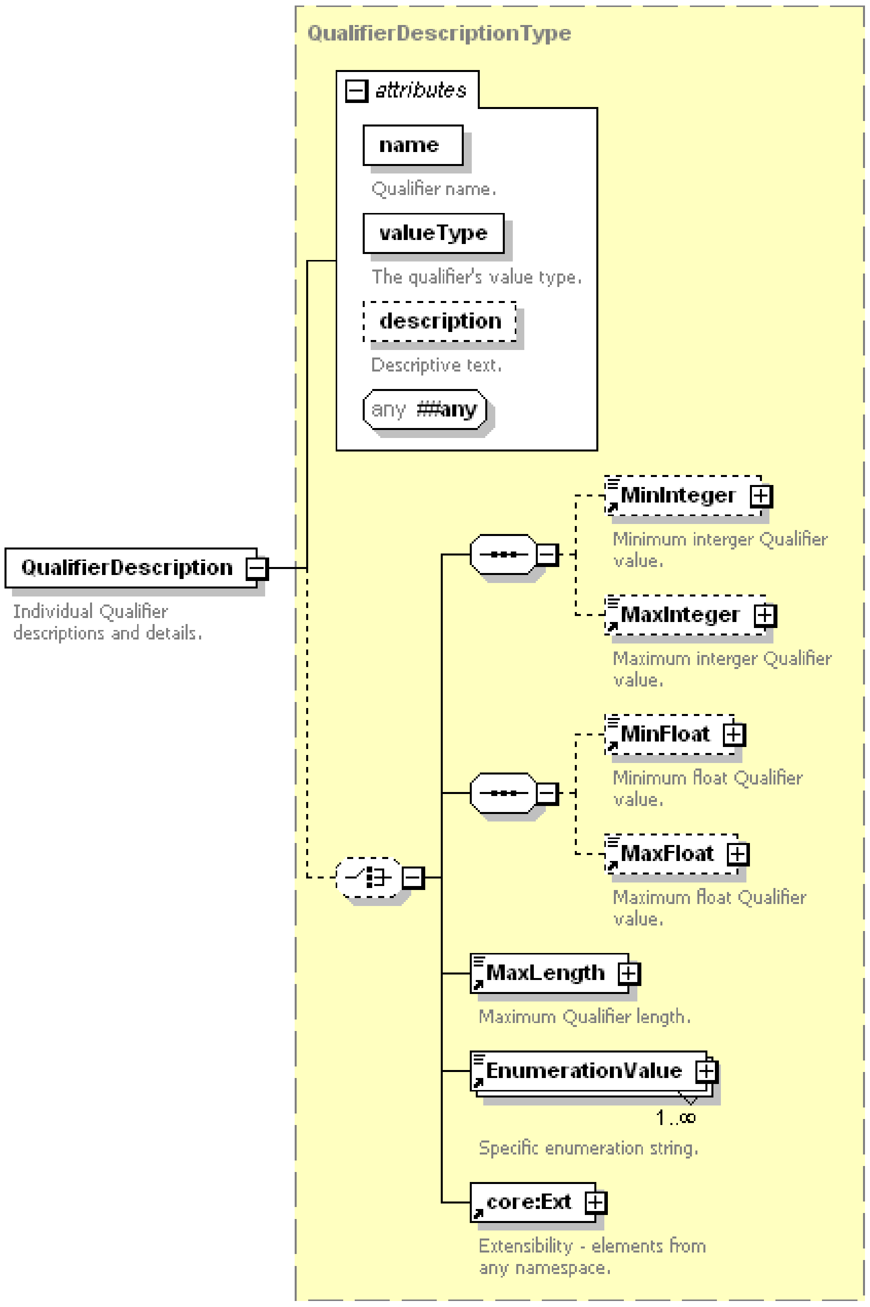

QualifierDescriptionType
attributes
name
Qualifier name.
valueType
The qualifier's value type.
description
Descriptive text.
any ##any
QualifierDescription
Individual Qualifier descriptions and details.
MinInteger
Minimum interger Qualifier value.
MaxInteger
Maximum interger Qualifier value.
MinFloat
Minimum float Qualifier value.
MaxFloat
Maximum float Qualifier value.
MaxLength
Maximum Qualifier length.
EnumerationValue
1..∞
Specific enumeration string.
core:Ext
Extensibility - elements from any namespace.

Figure 43. QualifierDescription Element XML Schema

@name [Required, qualifierNameAttrType] — The qualifier name. See Section 13.10 for additional information.

@valueType [Required, QualifierValueTypeEnumerationType] — Qualifier values have a type identified by the @valueType attribute. The allowed value types are shown in Table 9. See Section 13.11 for additional information

@valueType Attribute Value	**Description**
integer	xsd:integer
float	xsd:float
string	xsd:string
enumeration	value chosen from an enumerated list of xsd:strings.
boolean	xsd:boolean
dateTime	xsd:dateTime
…	User defined type outside of the scope of this specification. The string ***shall*** be prefixed with the text “private:”.

Table 9. Allowed QualifierDescription Element @valueType Values

@description [**Optional, core:nonEmptyStringType**] — Optional descriptive human readable text.

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

MinInteger [Optional — under Choice] — The MinInteger element contains an xsd:integer value indicating the minimum value that ***shall*** be used for a qualifier value of type "integer". See Section 12.17 for additional information.

MaxInteger [Optional — under Choice] — The MaxInteger element contains an xsd:integer value indicating the maximum value that ***shall*** be used for a qualifier value of type "integer". See Section 12.14 for additional information.

MinFloat [Optional — under Choice] — The MinFloat element contains an xsd:float value indicating the minimum value that ***shall*** be used for a qualifier value of type "float". See Section 12.16 for additional information.

MaxFloat [Optional — under Choice] — The MaxFloat element contains an xsd:float value indicating the maximum value that ***shall*** be used for a qualifier value of type "float". See Section 12.13 for additional information.

MaxLength [Optional — under Choice] — The MaxLength element contains an xsd:nonNegativeInteger value indicating the maximum character count that ***shall*** be used for a qualifier value of type "string". See Section 12.15 for additional information.

EnumerationValue [Optional — under Choice] — One or more EnumerationValue elements whose values denote the allowed values for a qualifier value of type "enumeration".

core:Ext [Optional] — A container for any additional elements from any namespace. See [SCTE130-2] for additional information.

12.21 QualifierSet

A QualifierSet is the complete list of Qualifiers assigned to one specific object in a logical service's basic query data model. The XML schema for the QualifierSet element is shown in Figure 44.

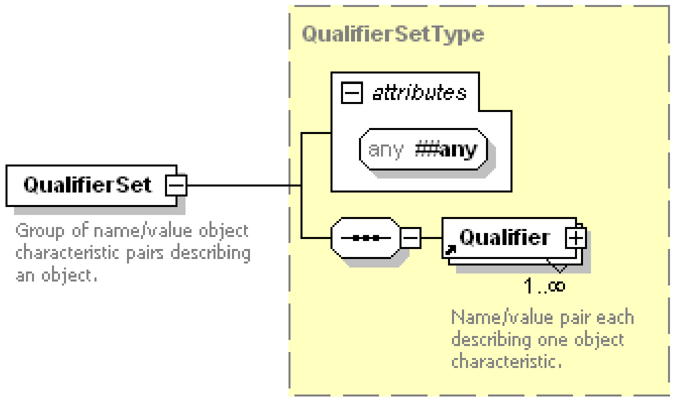

QualifierSetType
attributes
any ##any
QualifierSet
Group of name/value object characteristic pairs describing an object.
Qualifier
1..∞
Name/value pair each describing one object characteristic.

Figure 44. QualiferSet Element XML Schema

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

Qualifier [Required] — One or more Qualifier elements each containing a name/value pair describing one object characteristic. For more information on the Qualifier element see Section 12.18.

12.22 Query

The Query element contains elements specifying the semantics for a single GIS query match.

The XML schema definition for this element is provided in Figure 45.

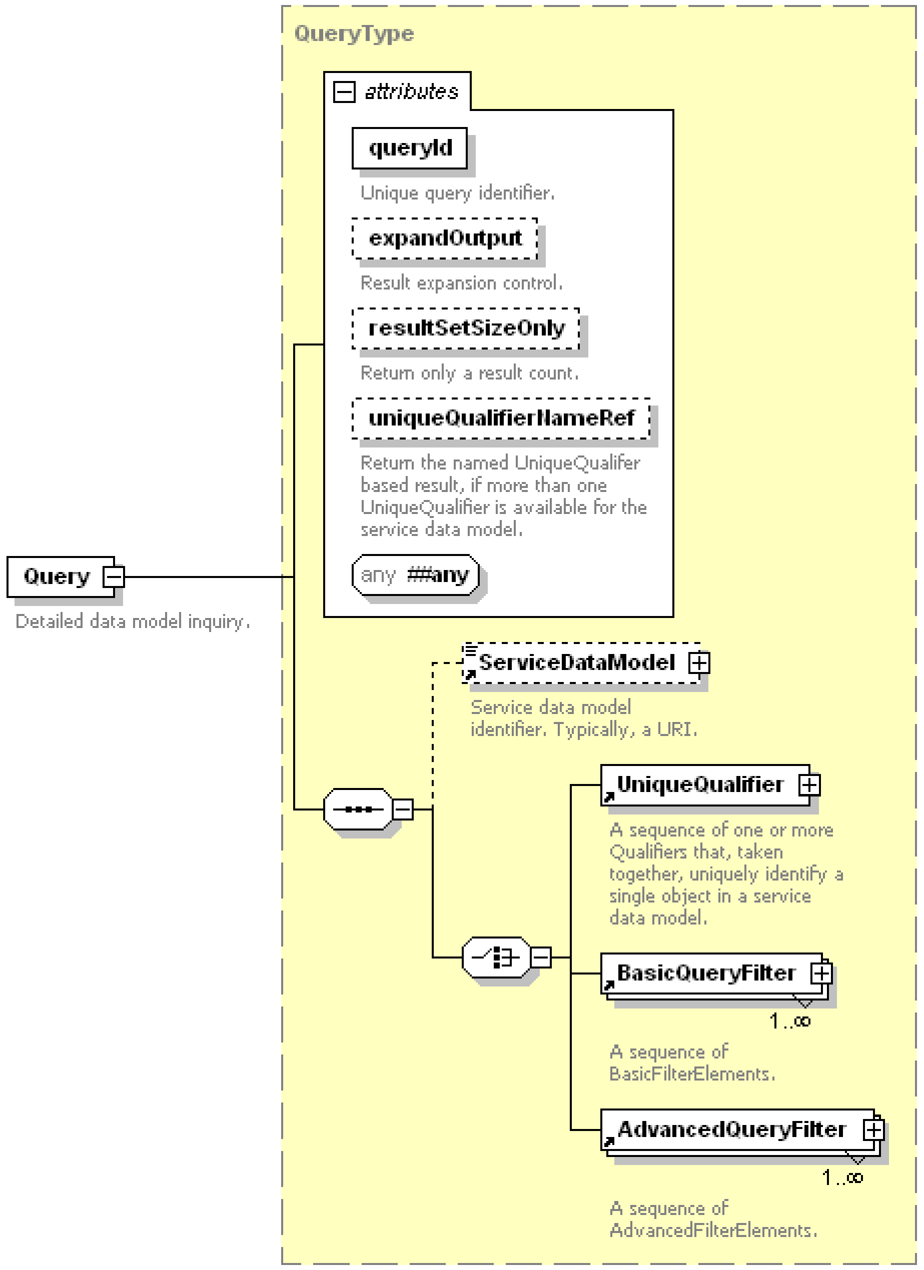

QueryType
attributes
queryId
Unique query identifier.
expandOutput
Result expansion control.
resultSetSizeOnly
Return only a result count.
uniqueQualifierNameRef
Return the named UniqueQualifer based result, if more than one UniqueQualifier is available for the service data model.
any ##any
Query
Detailed data model inquiry.
ServiceDataModel
Service data model identifier. Typically, a URI.
UniqueQualifier
A sequence of one or more Qualifiers that, taken together, uniquely identify a single object in a service data model.
BasicQueryFilter
1..∞
A sequence of BasicFilterElements.
AdvancedQueryFilter
1..∞
A sequence of AdvancedFilterElements.

Figure 45. Query Element XML Schema

The Query element contains the following attributes and elements:

@queryId [Required, queryIdAttrType] — The @queryId attribute is a unique identifier for the inquiry. This identifier ***shall*** be unique within the scope of the enclosing parent element's @identity attribute and ***shall not*** be empty. See Section 13.13 for further details on the queryIdAttrType.

@expandOutput [Optional, expandOutputAttrType] — The @expandOutput attribute controls the results output with the exact output behavior being dependent on the input elements. See the QueryResult element in Section 12.23 for precise output details. If the @expandout Ouput attribute is omitted, the default value is "false." See section 13.3 for further details on the expandOutputAttrType type.

@resultSetSizeOnly [Optional, resultSetSizeOnlyAttrType] — The @resultSetSizeOnly attribute restricts the QueryResult element to only return the inquiry total result count and no additional data ***shall*** be returned. If the @resultSetSizeOnly attribute is omitted, the default value is "false." The @resultsSetSizeOnly attribute ***shall*** only be present when the Query element is used within a QueryRequest message type element. The attribute ***shall not*** be present (i.e., used) when the Query element is located in the NotificationRegistrationRequest or CreateCursorRequest message types. See Section 13.16 for more information on the resultSetSizeOnlyAttrType type.

@uniqueQualifierNameRef [Optional, uniqueQualifierNameAttr] — The @uniqueQualifierNameRef attribute contains the @uniqueQualifierName attribute's value which was returned in the UniqueQualifierDeclaration element when the service data model defines more than UniqueQualifierDeclaration element. This attribute enables the caller to specify which UniqueQualifier element result that ***shall*** be returned in the QueryResult element. If omitted, the service data model's default unique qualifier is returned as applicable. See the QueryResult element in Section 12.23 for applicability details. See Section 12.27 for information on the UniqueQualifierDeclaration element and see Section 13.18 for more information on the uniqueQualifierNameAttr type.

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

ServiceDataModel [Optional] — The ServiceDataModel element contains a reference to the service data model that ***shall*** be used to satisfy the inquiry. When the ServiceDataModel element is not present, the logical service ***shall*** use the default service data model to satisfy the query. See Section 12.24 for further information on the ServiceDataModel element.

UniqueQualifer [Required — under Choice] — The UniqueQualifier element is a container element for Qualifiers that – taken together – uniquely identify an object in a basic query data model. See Section 12.24 for additional information.

BasicQueryFilter [Required — under Choice] — The BasicQueryFilter element is a container element for BasicFilterElement elements. Execution of the individual BasicFilterElement elements ***shall*** occur in document order. See 12.8 for further information on the BasicQueryFilter element.

AdvancedQueryFilter [Required — under Choice] — The AdvancedQueryFilter element is a container element for AdvancedFilterElement elements. Execution of the individual AdvancedFilterElement elements ***shall*** occur in document order. See section 12.2 for further information on the AdvancedQueryFilter.

12.23 QueryResult

Query results are returned to a calling consumer encapsulated within a QueryResult element. This element contains the 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 the QueryResult element is shown in Figure 46.

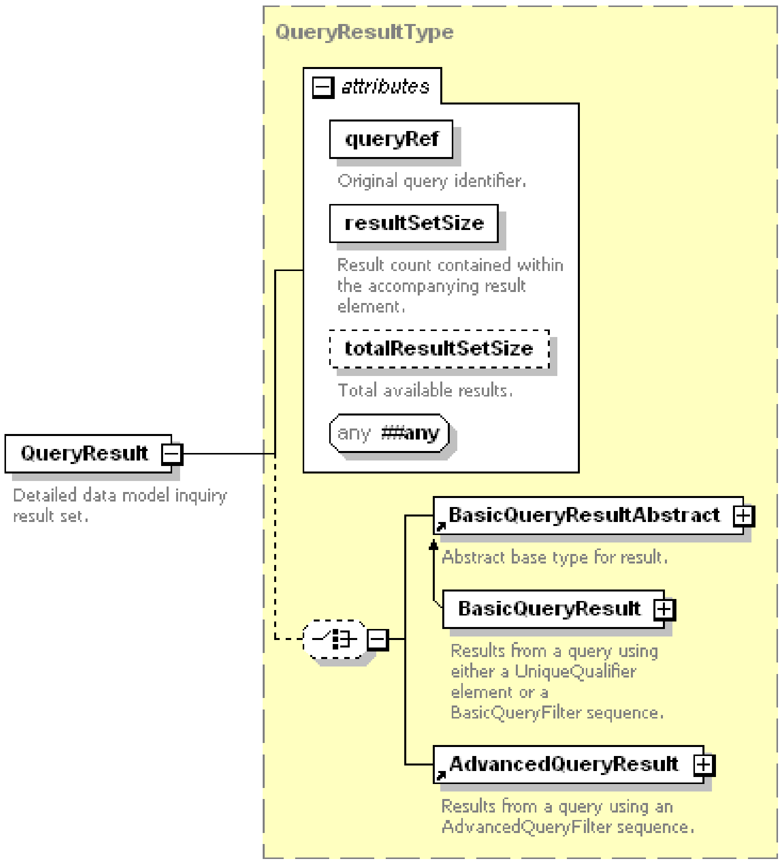

QueryResultType
attributes
queryRef
Original query identifier.
resultSetSize
Result count contained within the accompanying result element.
totalResultSetSize
Total available results.
any ##any
QueryResult
Detailed data model inquiry result set.
BasicQueryResultAbstract
Abstract base type for result.
BasicQueryResult
Results from a query using either a UniqueQualifier element or a BasicQueryFilter sequence.
AdvancedQueryResult
Results from a query using an AdvancedQueryFilter sequence.

Figure 46. QueryResult Element XML Schema

If the originating Query element was comprised of a UniqueQualifier element and the inquiry result count exceeds the value one, the core:StatusCode element of the encapsulating message type ***shall*** have an error value of 8003 (Malformed Unique Qualifier).

If the Query element contained one or more AdvancedQueryFilter elements and the Query element's @expandOutput attribute had the value "false" and the service data model does not support the return of a unique qualifier, the core:StatusCode element of the encapsulation message type ***shall*** have an error value of core:NotSupported.

The QueryResult element contains the following attributes and elements.

@queryRef [Required, queryIdRefAttrType] — The @queryRef attribute's value ***shall*** be the @queryId attribute's value used in the Query element that initiated the inquiry. See Section 13.13 for further information on the queryIdRefAttrType.

@resultSetSize [Required, xsd:nonNegativeInteger] — The @resultSetSize attribute's value ***shall*** contain the total number of individual XML elements present within either the BasicQueryResult element or the AdvancedQueryResult element (i.e., the value is the count of UniqueQualifier or QualifierSet elements within the BasicQueryResult element or the number of UniqueQualifier or AdvancedQueryResultData elements inside the AdvancedQueryResult element). If the attribute's value is zero, neither the BasicQueryResult element nor the AdvanceQueryResult element ***shall*** be present in the QueryResult element. See Table 10 for additional normative details regarding the attribute's value.

@totalResultSetSize [Optional, totalResultSetSizeAttrType] — The @totalResultSetSize attribute ***shall*** only be present with the Query element's @resultSetSizeOnly attribute is set to the value "true." Otherwise, this attribute ***shall not*** be present. When the @totalResultSetSize attribute is present, the attributes presence and value ***shall*** be dependent on the original Query element's inquiry composition.

If the Query element contained a UniqueQualifier element, the @totalResultSetSize attribute ***shall*** be present and the attribute's value ***shall*** be the total number QualifierSet elements in result set. Since a UniqueQualifier *may* only match one object by definition, the attribute's value ***shall*** only be the value zero or the value one.

If the Query element contained one or more BasicQueryFilter elements and the Query element's @expandOutput attribute had the value "false", the @totalResultSetSize attribute ***shall*** be present and attribute's value ***shall*** specify the total number of UniqueQualifier elements in the result.

If the Query element contained one or more BasicQueryFilter elements and the Query element's @expandOutput attribute had the value "true", the @totalResultSetSize attribute ***shall*** be present and attribute's value ***shall*** specify the total number of QualiferSet elements in the result. If the Query element's @uniqueQualifierNameRef attribute was present, the attribute's total is specific to the @uniqueQualiferNameRef attribute's named unique qualifier declaration. If the @uniqueQualifierNameRef attribute is omitted, the default unique qualifier declaration ***shall*** be applied and the @totalResultSetSize is the number QualifierStet elements returned for this result set.

If the Query element contained one or more AdvancedQueryFilter elements and the Query element's @expandOutput attribute had the value "false", the @totalResultSetSize attribute ***shall*** be present and attribute's value ***shall*** specify the total number of UniqueQualifier elements in the result.

If the Query element contained one or more AdvancedQueryFilter elements and the Query element's @expandOutput attribute had the value "true", the @totalResultSetSize attribute *may* be present and attribute's value ***shall*** specify the implementation specific matching record count where a record's definition is outside the scope of this specification.

See Table 10 for a detailed summary.

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

BasicQueryResultAbstract [Required under Choice] — The BasicQueryResultAbstract element is an abstract (or placeholder) element mandating an XML substitution. The BasicQueryResultAbstract element ***shall not*** be returned in a QueryResult element. Instead, a non-abstract (or concrete) form of the element, such as the BasicQueryResult element ***shall*** be returned and the advertising service and its data model define the returned element and its structure.

BasicQueryResult [Substitution under Choice] — The BasicQueryResult element is a substitution element for BasicQueryResultAbstract that ***shall*** contain result data originating from inquires using the UnqiueQualifier or BasicFilterElement elements. The BasicQueryResult element shal be present when specified by an advertising service's data model. See Table 10 below for details regarding the BasicQueryResults usage. See Section 12.9 for more information on the BasicQueryResult element.

AdvancedQueryResult [Required under Choice] — The AdvancedQueryResult element is a container element for the information returned from the successful execution of an advanced query. See Section 12.4 for details on the AdvancedQueryResult element.

Table 10 assumes the core:StatusCode element indicates a successful inquiry execution. An inquiry *may* return a result count of zero in which case no XML elements of any type ***shall*** appear within the QueryResult element. Attributes and element behavior not specifically covered within the table ***shall*** behave as previously defined.

When an advertising service's data model defines query results to be returned using the BasicQueryResult element and/or the AdvancedQueryResult element, Table 10 ***shall*** apply.

Query Element					QueryResult Element			
							Elements present within the QueryResult element when @resultsSetSize is non-zero	
Query Element Present	@expand Output	@result SetSize Only	@unique Qualifier NameRef		@total Result Set Size	@result SetSize	BQR or AQR	Elements present within BQR or AQR when @resultsSetSize is non-zero
Unique Qualifier	N/A	True	N/A		0 or 1	0	No elements ***shall*** be present.	
	True	False	N/A		NP	0 or 1	BQR	One QualifierSet element
	False	False	N/A		NP	0 or 1	BQR	One QualifierSet element
Basic Query Filter	True	True	N/A		0 to UB (Note 1)	0	No elements ***shall*** be present.	
	False	True	Not Present		0 to UB (Note 2)			
	False	True	Present		0 to UB (Note 2)			
	True	False	N/A		NP	0 to UB (Note 1)	BQR	One to unbounded QualifierSet elements
	False	False	Not Present		NP	0 to UB (Note 2)	BQR	One to unbounded UniqueQualifier elements using the default unique qualifier
	False	False	Present		NP	0 to UB (Note 2)	BQR	One to unbounded UniqueQualifier elements using the @uniqueQualifeirNameRef identified unique qualifier
Advanced Query Filter	True	True	N/A		0 to UB (Note 3)	0	No elements ***shall*** be present.	
	False	True	Not Present		0 to UB (Note 2)			
	True	True	Present		0 to UB (Note 2)			
	True	False	N/A		NP	0 to UB (Note 4)	AQR	1 to unbounded AdvancedQueryResultData elements
	False	False	Not Present		NP	0 to UB (Note 2)	AQR	1 to unbounded UniqueQualifier elements using the default unique qualifier
	False	False	Present		NP	0 to UB (Note 2)	AQR	1 to unbounded UniqueQualifier elements using the @uniqueQualifeirNameRef identified unique qualifier

Table 10. QueryResult Element Truth Table for an Advertsing Service Using the BasicQueryResult Element

N/A = Not Applicable (ignored if present)
NP = Not present (***shall not*** be returned)
UB = Unbounded
BQR = BasicQueryResult element
AQR = AdvancedQueryResult

Note 1: If non-zero, the mandatory value ***shall*** be the number of QualifierSet elements in the BasicQueryResponse element.
Note 2: If non-zero, the mandatory value ***shall*** be the number of UniqueQualifier elements in the BasicQueryResponse element.
Note 3: If non-zero, the optional value ***shall*** be the number of implementation specific records contained in the result where the definition of a record is outside the scope of this specification.
Note 4: If non-zero, the mandatory value ***shall*** be the number of AdvancedQueryResultData elements are contained in the AdvancedQueryResult element.

12.24 ServiceDataModel

The ServiceDataModel element contains a non-empty string uniquely identifying a service data model supported by a logical service implementing the GIS interface.

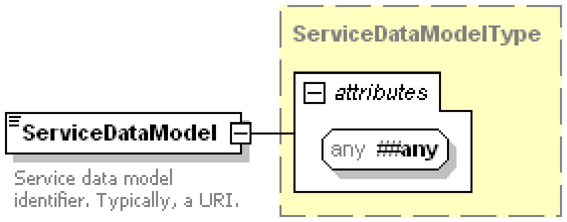


Figure 47. ServiceDataModel Element XML Schema

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

The ServiceDataModel element's value ***shall*** be a xsd:anyURI and ***shall not*** be empty.

12.25 ServiceDataModelProfile

The ServiceDataModelProfile element contains the information needed to uniquely identify a service data model and, if the service data model *may* be queried with an advanced query language, provides a list of one or more advanced query languages that *may* be used to query the service data model.

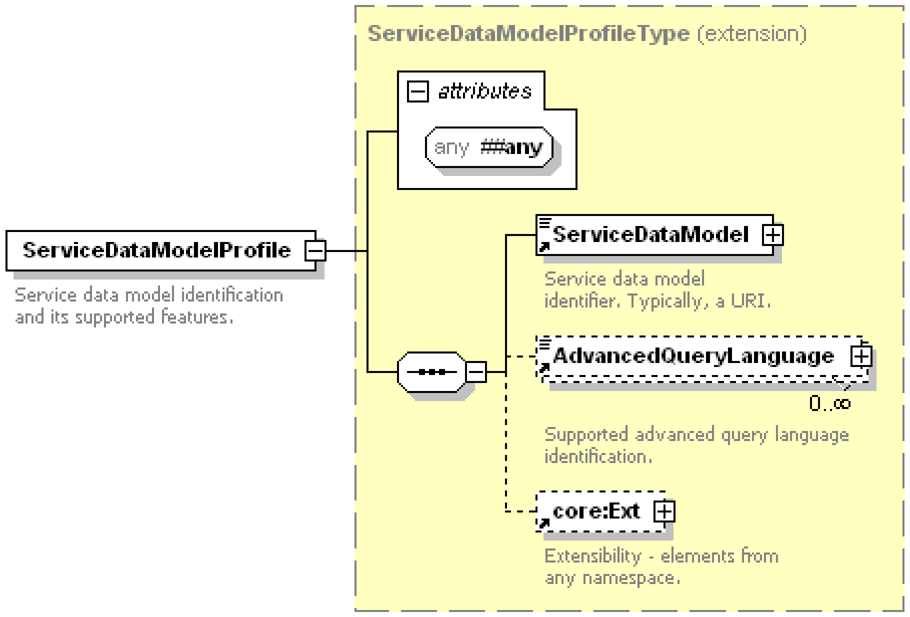

ServiceDataModelProfileType (extension)
attributes
any ##any
ServiceDataModelProfile
Service data model identification and its supported features.
ServiceDataModel
Service data model identifier. Typically, a URI.
AdvancedQueryLanguage
0..∞
Supported advanced query language identification.
core:Ext
Extensibility - elements from any namespace.

Figure 48. ServiceDataModelProfile Element XML Schema

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

ServiceDataModel [Required] — The service data model element uniquely identifies the service data model. For more information on the ServiceDataModel element see Section 12.24.

AdvancedQueryLanguage [Optional] — If present, the AdvancedQueryLanguage element sequence identifies one or more advanced query languages that *may* be used to query the service data model. For more information on the AdvancedQueryLanguage element see Section 12.3.

core:Ext [Optional] — Any additional elements from any namespace. For additional information on the core:Ext element see [SCTE130-2].

12.26 UniqueQualifier

A UniqueQualifier element contains one or more Qualifier elements which when processed as a basic query against a service data model reference a unique object within a logical service's data store. For example, if a logical service's data model describes VOD content, the UniqueQualifier might include two Qualifiers – ProviderId and ProviderAssetId. If a logical service's data model describes subscribers, the UniqueQualifier might include a set-top box DeviceId or MACAddress.

The XML schema for the UniqueQualifier element is shown in Figure 49.

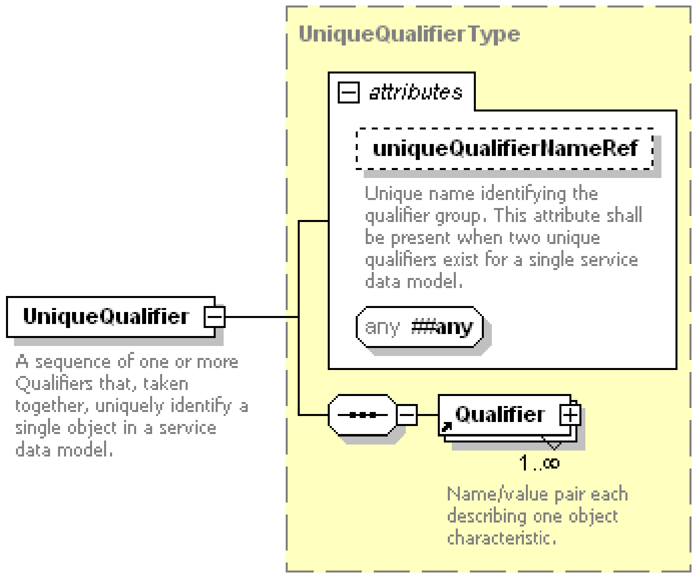

UniqueQualifierType
attributes
uniqueQualifierNameRef
Unique name identifying the qualifier group. This attribute shall be present when two unique qualifiers exist for a single service data model.
any ##any
UniqueQualifier
A sequence of one or more Qualifiers that, taken together, uniquely identify a single object in a service data model.
Qualifier
1..∞
Name/value pair each describing one object characteristic.

Figure 49. UniqueQualifer Element XML Schema

A basic query service data model ***shall*** have at least one unique qualifier and ***shall*** have at least one unique qualifier declaration. A service data model *may* have more than one unique qualifier declaration. When a service data model is described using more than one UniqueQualifierDelcaration, each UniqueQualifier ***shall*** have a @uniqueQualifierNameRef attribute to distinquish the unique qualifier being referenced and no two UniqueQualifiers ***shall*** use the same string for the @uniqueQualifierNameRef attribute.

@uniqueQualifierNameRef [Optional, uniqueQualifierNameAttrType] — A non-empty string used to identify different UniqueQualifierDeclaration elements in the case where a service data model has more than one UniqueQualifierDeclaration. See Section 13.18 for more information on the use of the @uniqueQualifierNameAttr type.

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

Qualifier [Required] — One or more Qualifier elements each containing a name/value pair describing one object characteristic. For more information on the Qualifier element see Section 12.18.

12.27 UniqueQualifierDeclaration

A UniqueQualifierDeclaration element contains one or more QualifierDeclaration elements. Each QualifierDeclaration element defines one qualifier identifier (i.e., the name component of the name/value pair) and the entire QualifierDeclaration element sequence defines the qualifier group forming the basis for a unique qualifier. All qualifiers declared in the QualfierDeclaration element sequence ***shall*** be included in a UniqueQualifier element as the entire set is required to uniquely identify an object.

A basic query service data model must have at least one UniqueQualifierDeclaration (and therefore must be able to uniquely identify objects via one unique qualifier group) and a service data model *may* have more than one unique qualifier (and thus, more than one UniqueQualifierDeclaration). When a service data model has more than one UniqueQualifierDeclaration, each UniqueQualifierDeclaration ***shall*** have a @uniqueQualifierName attribute and no two UniqueQualifiersDeclarations ***shall*** use the same string for the @uniqueQualifierName attribute.

The XML schema for the UniqueQualifier element is shown in Figure 50.

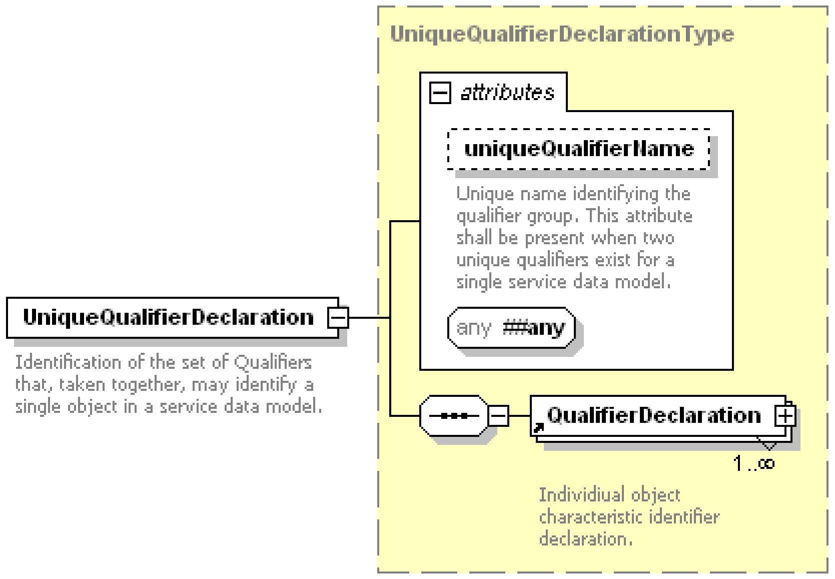

UniqueQualifierDeclarationType
attributes
uniqueQualifierName
Unique name identifying the qualifier group. This attribute shall be present when two unique qualifiers exist for a single service data model.
any ##any
UniqueQualifierDeclaration
Identification of the set of Qualifiers that, taken together, may identify a single object in a service data model.
QualifierDeclaration
1..∞
Individiual object characteristic identifier declaration.

Figure 50. UniqueQualifer Element XML Schema

@uniqueQualiferName [Optional, uniqueQualiferNameAttrType] — A service data model assigned string used to identify different UniqueQualifier elements in the case where a service data model has more than one UniqueQualifier. See Section 12.22 for more information on the use of the @uniqueQualiferName attribute.

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

QualifierDeclaration [Required] — One or more Qualifier elements each containing a name/value pair describing one object characteristic. For more information on the QualifierDeclaration element see Section 12.19.

13.0 SCTE 130 PART 8 GIS ATTRIBUTE TYPES AND COMMON TYPES

The following sections define the GIS attribute types and common complex types used throughout this document.

13.1 cursorIdAttrType Attribute Type

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

13.2 cursorIdRefAttrType Attribute Type

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

13.3 expandOutputAttrType Attribute Type

expandOutputAttrType [xsd:boolean] — This attribute type, typically referred to as the @expandOutput attribute, is a Boolean indication to the GIS to expand query results to include the full qualifier set for a service data model basic query or as an query specific representation for the selected service data model for an advanced query. See Section 12.22 and Section 12.23 for additional information.

13.4 filterElementNameType Attribute Type

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

13.5 filterElementValueAttrType Attribute Type

filterElementValueAttrType [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 GIS is a subset of the set of regular expressions typically supported by most regular expression processing facilities. See Section 14.1 for additional information.

13.6 MinMaxFloatType Complex Type

MinMaxFloatType [xsd:float] — This xsd:complexType, typically used as an element's value type, requires an xsd:float value to be present. The value defines either the minimum or maximum inclusive value an entity *may* contain.

13.7 MinMaxIntegerType Complex Type

MinMaxIntegerType [xsd:integer] — This xsd:complexType, typically used as an element's value type, requires an xsd:integer value to be present. The value defines either the minimum or maximum inclusive value an entity *may* contain.

13.8 MaxLengthType Complex Type

MaxLengthType [xsd:nonNegativeInteger] — This xsd:complexType, typically used as an element's value type, requires a xsd:nonNegativeInteger value to be present. The value defines the maximum length of a string or other similar character sequence.

13.9 NotificationTypeEnumeration Attribute Type

NotificationTypeEnumeration [core:nonEmpytStringType] — This attribute type, typically referred to as the @noticeType attribute, contains one of the values from Table 6 in Section 11.5.1. The attribute specifies the data store change that the Notification message type is providing.

13.10 qualifierNameAttrType Attribute Type

qualifierNameAttrType [core:nonEmptyStringType] — This attribute type, typically referred to as the @name attribute in the Qualifier and QualifierDeclaration elements, functions as a object characteristic identifier. This attribute ***shall not*** be empty.

13.11 QualifierValueTypeEnumerationType Attribute Type

QualifierValueTypeEnumerationType [xsd:enumeration] — This attribute type, typically referred to as the @valueType attribute, is an enumeration of the allowed types for the @value attribute of a Qualifier element.

The qualifierValueTypeEnumerationType includes the types shown in Table 9.

13.12 QueryFilterOpTypeEnumeration Attribute Type

QueryFilterOpTypeEnumeration [core:nonEmptyStringType] — This attribute type, typically referred to as the @op attribute, represents the operation 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 11 and ***shall*** appear exactly as they are in this table.

Enumeration Value	Description
include	The include value indicates to a logical service that implements the GIS interface that the results from the execution of this query ***shall*** be uniquely appended to the final result set.
exclude	The exclude value indicates to a logical service that implements the GIS interface that the results of this query ***shall*** be subtracted from the final result set.

Table 11. QueryFilterOpTypeEnumeration Values

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

13.13 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 consumer system and ***shall not*** be empty. The queryIdAttrType ***shall*** be unique within the scope of the consumer's @identity attribute.

13.14 queryIdRefAttrType Attribute Type

queryIdRefAttrType [queryIdAttrType] — This attribute type, used as the @QueryRef attribute, is a reference to an original @QueryId attribute and ***shall not*** be empty.

13.15 queryLanguageAttrType Attribute Type

queryLanguageAttrType [core:nonEmptyStringType] — This attribute type, typically referred to as the @ql attribute, is a string identifying the specific query language engine that ***shall*** be used to process a query and ***shall not*** be empty. The GIS ***shall*** support the advanced query languages listed in Table 8 if advanced query language support is supported by a logical service that incorporates the SCTE 130 Part 8 interface and if advanced queries *may* be executed against a service data model.

13.16 resultSetSizeOnlyAttrType Attribute Type

resultSetSizeOnlyAttrType [xsd:boolean] — This attribute type, used as the @resultSizeOnly attribute, is typically an optional attribute and when not present its value is assumed to be "false". When the @resultSetSizeOnly attribute is supplied with a value of "true", the QueryResult element ***shall not*** include a result set but ***shall*** instead return the result set size in the @totalResultSetSize attribute's value.

13.17 totalResultSetSizeAttrType Attribute Type

totalResultSetSizeAttrType [xsd:nonNegativeInteger] — This attribute type, typically used as the @totalResultSetSize attribute, specifies the total available result count. The attributes value interpretation is dependent on the inquiry composition. See Section 12.23 for additional information specific to the value's interpretation relative to the inquiry composition.

13.18 uniqueQualifierNameAttrType Attribute Type

uniqueQualifierNameAttrType [core:nonEmptyStringType] — This attribute type, used as the @uniqueQualifierName or uniqueQualiferNameRef attribute, is required when a basic query is made against a service data model that has more than one UniqueQualifier. When a basic query is formulated to return a UniqueQualifier sequence, all UniqueQualifier elements returned ***shall***

have a composition that matches the value of the @uniqueQualifierName attribute.

14.0 BASIC QUERIES AND REGULAR EXPRESSIONS

The BasicQueryFilter element *may* contain one or more BasicFilterElement elements. These BasicFilterElement elements contain name/value pairs that ***shall*** be applied to a logical service's data model. As an example, a BasicQueryFilter element *may* contain a BasicFilterElement containing a @name attribute and a complete or wild-carded @value attribute. For instance:

```
<BasicQueryFilter op=”include”>
  <BasicFilterElement name=”Age” value=”Under20” valueIsRegex=”false”/>
</BasicQueryFilter>
```

Example 2. BasicQueryFilter Element

In Example 2 above, the @name attribute of the BasicFilterElement contains the string “Under20”, which maps to the Age qualifier attribute located in the data model.

The @value attribute contains the expected value for the @name attribute that satisfies the query. In this example the @value attribute contains the string “Under20”. Any records referenced by the logical services data model that contain the attribute Age=”Under20”, will be returned in the result set. The @valueIsRegex attribute indicates that the value of the @value attribute *should not* be treated as a regular expression when processing this BasicFilterElement.

14.1 Regular Expressions and Wildcards

Wildcarding and regular expressions allow the consumer of a logical service that implements the GIS interface to select a targeted set of assets using a single query. For instance:

```
<BasicQueryFilter op=”include”>
  <BasicFilterElement name=”Income” value=”^[23]0Kto[34]0K” valueIsRegex=”true”/>
</BasicQueryFilter>
```

Example 3. BasicQueryFilter Element

In Example 3 above, the @value attribute of the BasicFilterElement contains the regular expression “^[23]0Kto[34]0K”. When executed, this query ***shall*** resolve to all records that contain Income qualifiers with values “20Kto30K” and “30Kto40K”.

Regular expressions are only allowed in the @value attribute of the BasicFilterElement and are only considered regular expression if the @valueIsRegex attribute is set to “true”.

The set of regular expressions that ***shall*** be supported by a logical service that implements the GIS interface is a subset of the regular expressions supported by most regular expression processing engines. Table 12 lists each regular expression and a short description of what each 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 of 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 12. GIS Regular Expressions

Table 13 contains some example expressions and results based on the regular expressions defined in Table 12.

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 start 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** z,y,z
(ab)+$	Match the group (ab) 1 or more times at the end of the line

Table 13. Regular Expression Examples

15.0 SCHEMA CHANGES

The Mutable GIS introduces a new schema with new elements and types. A logical service implementing the Mutable GIS ***shall*** include the new schema along with the GIS schema.

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

Table 14 contains SCTE 130 Part 8 applicable status codes for the @detail attribute.

@Detail Value	Name	Description
8001	Cursor Undefined	The requested cursor does not exist within the GIS.
8002	Cursor Already Exists	The requested cursor already exists within the GIS.
8003	Malformed Unique Qualifier	A unique qualifier ***shall*** match one and only one object in a data store. If more than one object matches an inquiry using a UniqueQualifier element, the query is malformed and the unique qualifier concept is violated.

Table 14. StatusCode Details

Table 15 contains descriptive references to the SCTE130-2 @detail values that *may* be returned in GIS top level messages.

NotificationRegistrationResponse = NRR
QueryResponse = QR
CreateCursorResponse = CR
CancelCursorResponse = CCR
NotificationDeregisterResponse = NDR
ListNotificationRegistrationResponse = LNRR
DeregistrationNotification = DN
Notification = N
ListSupportedFeaturesResponse = LSFR
ListQualifiersResponse=LQR

Description	NRR	QR	CR	CCR	NDR	LNRR	DN	N	LSFR	LQR
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	✓	✓	✓		✓	✓			✓	✓
Cursor Undefined		✓		✓						
Cursor Already Exists			✓							
Malformed Unique Qualifier		✓								

Table 15. Part 2 and Part 8 StatusCode Detail Usage

APPENDIX B. COMPLEX QUERIES & EXPANDED OUTPUT EXAMPLES (INFORMATIVE)

B.1 Multiple Filter Elements

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

```
<BasicQueryFilter>
            <BasicFilterElement name="Age" value="Under20"/>
            <BasicFilterElement name="Income" value="20Kto30K"/>
  </BasicQueryFilter>
```

Example 4. BasicQueryFilter Element

In Example 4 above, the result of the query is a specific asset identified by a particular Age and Income. The operator combining two or more BasicFilterElement elements within a BasicQueryFilter is the AND operator. Both BasicFilterElement elements in Example 4 are combined together to form a single query.

In Example 4 above, in order for a single record to satisfy this query, the age must be “Under20” and the income must be “20Kto30K”, otherwise the record will be rejected. The @valueIsRegex attribute of the both BasicFilterElement elements is set to “false” in Example 4 by the attribute’s omission. Setting the @valueIsRregex attribute to false indicates regular expression processing will not be executed on the contents of the @value attributes and this is the default behavior if the @valueIsRegex attribute is omitted from the BasicFilterElement altogether.

The BasicQueryFilter element *may* contain one or more BasicFilterElement elements. The @op attribute indicates to the logical service how the individual result sets from query *should* be applied to the overall result set.

```
<BasicQueryFilter>
        <BasicFilterElement name="Age" value="Under20"/>
        <BasicFilterElement name="Income" value="20Kto30K"/>
</BasicQueryFilter>
<BasicQueryFilter op="include">
        <BasicFilterElement name="Age" value="Under20"/>
        <BasicFilterElement name="Income" value="30Kto40K"/>
</BasicQueryFilter>
```

Example 5. BasicQueryFilter Element

The results of this query *may* include one UniqueQualifier for each record that matches the query.

In Example 6, an operator has been added to the second BasicQueryFilter to act as a negating agent in order to limit the output:

```
<BasicQueryFilter>
        <BasicFilterElement name="Age" value="Under20"/>
        <BasicFilterElement name="Income" value="^[23]0KTo[45]K" valueIsRegex="true"/>
</BasicQueryFilter>
<BasicQueryFilter op="exclude">
        <BasicFilterElement name="Age" value="Under20"/>
        <BasicFilterElement name="MaritalStatus" value="Single"/>
</BasicQueryFilter>
```

Example 6. BasicQueryFilter Element

In Example 6 above, the first BasicQueryFilter matches all records with age qualifier values of "Under20" and income qualifier values of 20K to 50K. The second BasicQueryFilter excludes records with a marital status qualifier value of "Single".

B.2 Expanded Output

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

```
<Query queryId="1" expandOutput="true">
        <ServiceDataModel>http://SuperDemographics.com</ServiceDataModel>
        <BasicQueryFilter>
                <BasicFilterElement name="MACAddress" value="00-1e-c2-01-d3-2d"/>
        </BasicQueryFilter>
</Query>
```

Example 7. Query Element

The query in Example 7 will result in the selection of a single record on the logical service's data model. The query results ***shall*** be returned to the caller in the form of the full QualifierSet for that record in the data model.

The resulting output from the query provided in Example 7 follows in Example 8:

```
<QueryResult queryRef="1" totalResultSetSize="1" resultSetSize="1">
        <BasicQueryResult>
                <QualifierSet>
                        <Qualifier name="MACAddress" value="00-1e-c2-01-d3-2d"/>
                        <Qualifier name="Age" value="Under20"/>
                        <Qualifier name="Income" value="30Kto40K"/>
                        <Qualifier name="MaritalStatus" value="Single"/>
                        <Qualifier name="Children" value="No"/>
                </QualifierSet>
        </BasicQueryResult>
</QueryResult>
```

Example 8. QueryResult Element

Note that if the data model queried in Example 7 had a UniqueQualifier comprised of one Qualifer with name, "MACAddress", the result shown in Example 8 could have been obtained with a query using the UniqueQualifer rather than the BasicQueryFilter. This query is shown in Example 9.

```
<Query queryId="1">
  <ServiceDataModel>http://SuperDemographics.com</ServiceDataModel>
  <UniqueQualifier>
    <Qualifier name="MACAddress" value="00-1e-c2-01-d3-2d"/>
  </UniqueQualifier>
</Query>
```

Example 9. Query Element using UniqueQualifer

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 records available from the data model.

```
  <Query queryId="1" expandOutput="false">
        <ServiceDataModel>http://SuperDemographics.com</ServiceDataModel>
        <BasicQueryFilter>
              <BasicFilterElement name="Age" value="Under20"/>
              <BasicFilterElement name="Income" value="^[234]0Kto[345]0K"
valueIsRegex="true"/>
        </BasicQueryFilter>
  </Query>
```

Example 10. Query Element

The results of the query from Example 10 are illustrated below in Example 11.

```
  <QueryResult queryRef="1" totalResultSetSize="3" resultSetSize="3">
        <BasicQueryResult>
              <UniqueQualifier>
                    <Qualifier name="MACAddress" value="00-1e-c2-01-d3-2d"/>
              </UniqueQualifier>
              <UniqueQualifier>
                    <Qualifier name="MACAddress" value="00-1e-52-74-2e-6a"/>
              </UniqueQualifier>
              <UniqueQualifier>
                    <Qualifier name="MACAddress" value="00-50-56-c0-00-01"/>
              </UniqueQualifier>
        </BasicQueryResult>
  </QueryResult>
```

Example 11. QueryResult Element

Note that in Example 11, three UniqueQualifiers are returned - each indicating the MACAddress of equipment owned by subscribers that are under 20 years old and have an income in the 20K to 50K range.

APPENDIX C. ADVANCED QUERIES (INFORMATIVE)

C.1 Advanced Query Examples

A logical service implementing the GIS interface *may* support advanced query mechanisms. In order to support advanced queries, the AdvancedQueryFilter element accepts the inclusion of one or more AdvancedFilterElement elements. The AdvancedFilterElement element contains a string schema type within a CDATA section containing the raw statement of the advanced query.

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

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

Example 12. AdvancedQueryFilter Element

In Example 12 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 AdvancedQueryFilter element 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 13:

```
  <QueryResult queryRef="1" totalResultSetSize="1" resultSetSize="1">
          <AdvancedQueryResult>
                  <AdvancedQueryResultData><![CDATA[
         <ADI>
          <Metadata>
            <AMS . . . />
          </Metadata>
         </ADI>
       ]]></AdvancedQueryResultData>
          </AdvancedQueryResult>
  </QueryResult>
```

Example 13. QueryResult Element

APPENDIX D. CURSOR EXAMPLES (INFORMATIVE)

D.1 Creating Cursors

Cursors are created by using the CreateCursorRequest message. An example of this request is shown in Example 14. The "xxx" preceding each message name is done assuming a logical service named "xxx" has implemented the GIS interface and is described in Example 1.

```
  <xxxCreateCursorRequest messageId="req-1" version="1.0" system="GISClient"
cursorId="cursor-1" cursorExpires="2009-08-10T12:00:00Z" identity="40DA910E-01AF-5050-
C7EA-5D7B4A475311">
        <Query queryId="query-1" expandOutput="false">
                <ServiceDataModel>http://SuperDemographics.com</ServiceDataModel>
                <BasicQueryFilter>
                        <BasicFilterElement name="Age" value="Under20"
valueIsRegex="false"/>
                </BasicQueryFilter>
        </Query>
  </xxxCreateCursorRequest>
```

Example 14. CreateCursorRequest Message

In Example 14 above, the CreateCursorRequest message contains a @cursorId attribute and a @cursorExpires attribute. The logical service implementing the GIS interface will associate the @cursorId value with the physical cursor instance so that subsequent calls from the consumer *may* refer back to the same physical cursor instance.

The @cursorExpires attribute contains a date and time, which is a request to the logical service to give the new cursor construct a particular life span. The logical service *may* choose to ignore the @cursorExpires date and time 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 is returned to the caller in the create cursor response message.

The logical service, upon receipt of the CreateCursorRequest, ***shall*** create the cursor construct, accept or adjust the cursor duration and then execute the supplied query in order to fill the cursor. The data within the cursor ***shall*** remain static for the lifetime of the cursor.

Once the cursor has been constructed, the advertising service will respond to the caller with a create cursor response message. This message will contain a reference to the original create cursor request message, a result set size, and the final value for the cursor expiration duration.

Example 15 contains an illustration of the CreateCursorResponse message.

```
  <xxxCreateCursorResponse messageId="resp-1" version="1.0" system="GISServer"
messageRef="req-1" totalResultSetSize="100" cursorExpires="2009-08-10T12:00:00.0Z"
identity="40DA910E-01AF-5050-C7EA-5D7B4A475312">
          <core:StatusCode class="0"/>
  </xxxCreateCursorResponse>
```

Example 15. CreateCursorResponse Message

Example 15 above, indicates that the cursor was created successfully and that the cursor information, identified by @cursorId=cursor-1, from the CreateCursorRequest, will be available until the end date specified by the @cursorExpires attribute. In this case, the advertising service 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 @totalResultSetSize attribute. In this case, 100 data items are contained in the cursor.

D.2 Traversing a Cursor

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

```
  <xxxQueryRequest messageId="req-2" version="1.0" system="GISClient"
identity="40DA910E-01AF-5050-C7EA-5D7B4A475311">
          <Cursor startIndex="0" count="10" cursorRef="cursor-1" queryRef="query-1"/>
  </xxxQueryRequest>
```

Example 16. QueryRequest Message

In Example 16 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 logical service 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 queryId (queryRef) and cursorId from the original CreateCursorRequest. (See Section 11.8.1).

If the requested cursor has timed out (expired), the QueryResponse will contain a status code indicating a class "Error" and Code (Cursor Undefined). See Section Appendix A for additional details.

To continue to iterate over an existing cursor, the @startIndex attribute of the cursor must be modified. Example 17 below, illustrates a continuation of the cursor walk from Example 16.

```
  <xxxQueryRequest messageId="req-2b" version="1.0" system="GISClient"
identity="40DA910E-01AF-5050-C7EA-5D7B4A475311">
        <Cursor startIndex="10" count="10" cursorRef="cursor-1" queryRef="query-2"/>
  </xxxQueryRequest>
```

Example 17. QueryRequest Message

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 consumer has finished working with a cursor, the cursor *may* be canceled before the duration time of the cursor has expired. Example 18 illustrates a complete CancelCursorRequest message.

```
  <xxxCancelCursorRequest messageId="3" version="1.0" system="GISClient"
cursorRef="cursor-1" identity="40DA910E-01AF-5050-C7EA-5D7B4A475311"/>
```

Example 18. CancelCursorRequest Message

In Example 19, the status code element indicates that the requested cursor was successfully removed from the advertising service.

Upon receipt of a CancelCursorRequest, the logical service ***shall*** remove the requested cursor for the specified system if the cursor exists. Example 19 illustrates the expected CancelCursorResponse message.

```
  <xxxCancelCursorResponse messageId="3b" version="1.0" system="GISServer"
messageRef="3" identity="40DA910E-01AF-5050-C7EA-5D7B4A475312">
        <core:StatusCode class="0"/>
  </xxxCancelCursorResponse>
```

Example 19. CancelCursorResponse Message

If the cursor identified in the CancelCursorRequest message has already expired before the cancel cursor request message arrives, the status code value in the CancelCursorResponse message *should not* indicate an error.

APPENDIX E. MESSAGE EXAMPLES (INFORMATIVE)

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

E.1 Listing Supported Features

The ListSupportedFeaturesRequest is the only service endpoint that is required to be available on the well-known address for a logical service that implements the GIS interface. All other service endpoints *may* also be available on the well-known 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.

Example 20 contains an example of a ListSupportedFeaturesRequest message.

```
  <xxxListSupportedFeaturesRequest messageId="acs-342" system="GISClient" version="1.0"
identity="40DA910E-01AF-5050-C7EA-5D7B4A475311"/>
```

Example 20. ListSupportedFeaturesRequest Message

Example 21 contains an example of a ListSupportedFeaturesResponse message. The single core:Callout element does not include an @message attribute. This indicates that all logical service channel endpoints are available through this well-known GIS address endpoint.

```
  <xxxListSupportedFeaturesResponse messageId="sca-343" system="GISServer" version="1.0"
identity="40DA910E-01AF-5050-C7EA-5D7B4A475312" messageRef="acs-342">
		<core:StatusCode class="0"/>
		<core:Callout>
			<core:Address type="SOAP
1.1">http://10.250.30.22/GISServer</core:Address>
		</core:Callout>
		<ServiceDataModelProfile>
			<ServiceDataModel>http://SuperDemographics.com</ServiceDataModel>
			<AdvancedQueryLanguage>XPath</AdvancedQueryLanguage>
			<AdvancedQueryLanguage>XQuery</AdvancedQueryLanguage>
		</ServiceDataModelProfile>
  </xxxListSupportedFeaturesResponse>
```

Example 21. ListSupportedFeaturesResponse Message

Example 22 contains a ListSupportedFeaturesResponse message that contains core:Callout elements for several specific logical service channel endpoints.

```
  <xxxListSupportedFeaturesResponse messageId="sca-343" system="GISServer" version="1.0"
identity="40DA910E-01AF-5050-C7EA-5D7B4A475312" messageRef="acs-342">
		<core:StatusCode class="0"/>
		<core:Callout>
			<core:Address type="SOAP
1.1">http://10.250.30.22/GISServer</core:Address>
		</core:Callout>
		<core:Callout message="NotificationRegistrationRequest">
			<core:Address type="SOAP
1.1">http://10.250.30.23/GISServer</core:Address>
		</core:Callout>
		<core:Callout message="NotificationDeregisterRequest">
			<core:Address type="SOAP
1.1">http://10.250.30.24/GISServer</core:Address>
		</core:Callout>
		<ServiceDataModelProfile>
			<ServiceDataModel>http://SuperDemographics.com</ServiceDataModel>
			<AdvancedQueryLanguage>XPath</AdvancedQueryLanguage>
			<AdvancedQueryLanguage>XQuery</AdvancedQueryLanguage>
		</ServiceDataModelProfile>
  </xxxListSupportedFeaturesResponse>
```

Example 22. ListSupportedFeaturesResponse Message

Example 23 contains three core:Callout elements. The first core:Callout element is the default core:Callout element. This element contains the default address for all logical service channel message endpoints. Two additional core:Callout elements in this example indicate that the service channel endpoints for the NotificationRegistrationRequest and NotificationDeregisterRequest messages are located on specific addresses, different from that of the default address(es).

```
  <xxxListSupportedFeaturesResponse messageId="sca-343" system="GISServer" version="1.0"
identity="40DA910E-01AF-5050-C7EA-5D7B4A475312" messageRef="acs-342">
		<core:StatusCode class="0"/>
		<core:Callout>
			<core:Address type="SOAP
1.1">http://10.250.30.22/GISServer</core:Address>
		</core:Callout>
		<core:Callout message="NotificationRegistrationRequest">
			<core:Address type="SOAP
1.1">http://10.250.30.23/GISServer</core:Address>
		</core:Callout>
		<core:Callout message="NotificationDeregisterRequest">
			<core:Address type="SOAP
1.1">http://10.250.30.24/GISServer</core:Address>
		</core:Callout>
		<ServiceDataModelProfile>
			<ServiceDataModel>http://SuperDemographics.com</ServiceDataModel>
			<AdvancedQueryLanguage>XPath</AdvancedQueryLanguage>
			<AdvancedQueryLanguage>XQuery</AdvancedQueryLanguage>
		</ServiceDataModelProfile>
  </xxxListSupportedFeaturesResponse>
```

Example 23. ListSupportedFeaturesResponse Message

E.2 Listing Qualifiers

The ListQualifiersRequest and ListQualifiersResponse messages allow a consumer of a logical service that implements the GIS interface to discover the list of qualifiers used in services basic query data model and to discover the UniqueQualifierSet for the data model.

Example 24 uses the ListQualifiersRequest message to discover the list of qualifiers and the UniqueQualifierSet for a logical service's data model.

```
  <xxxListQualifiersRequest messageId="acs-344" system="GISClient" version="1.0"
identity="40DA910E-01AF-5050-C7EA-5D7B4A475311">
		<ServiceDataModel>http://SuperDemographics.com</ServiceDataModel>
  </xxxListQualifiersRequest>
```

Example 24. ListQualifiersRequest Message

The result of the ListQualifiersRequest message is shown in Example 25.

```
<xxxListQualifiersResponse messageId="sca-345" system="GISServer" version="1.0" identity="40DA910E-
01AF-5050-C7EA-5D7B4A475312" messageRef="acs-344">
        <core:StatusCode class="0"/>
        <BasicQueryDataModelDescription>
                <ServiceDataModel>http://SuperDemographics.com</ServiceDataModel>
                <UniqueQualifierDeclaration>
                        <QualifierDeclaration name="MACAddress"/>
                </UniqueQualifierDeclaration>
                <QualifierDescription name="Age" valueType="enumeration">
                        <EnumerationValue>UnderTwenty</EnumerationValue>
                        <EnumerationValue>TwentyToForty</EnumerationValue>
                        <EnumerationValue>FortyToSixty</EnumerationValue>
                        <EnumerationValue>OverSixty</EnumerationValue>
                </QualifierDescription>
                <QualifierDescription name="Income" valueType="enumeration">
                        <EnumerationValue>Under50K</EnumerationValue>
                        <EnumerationValue>50Kto100K</EnumerationValue>
                        <EnumerationValue>100Kto200K</EnumerationValue>
                        <EnumerationValue>Over200K</EnumerationValue>
                </QualifierDescription>
                <QualifierDescription name="ZipPlusFour" valueType="string">
                        <MaxLength>10</MaxLength>
                </QualifierDescription>
                <QualifierDescription name="MACAddress" valueType="string">
                        <MaxLength>17</MaxLength>
                </QualifierDescription>
                <QualifierDescription name="CreditLimit" valueType="float">
                        <MinFloat>10.00</MinFloat>
                        <MaxFloat>5000000.00</MaxFloat>
                </QualifierDescription>
                <QualifierDescription name="SportsInterest" valueType="enumeration">
                        <EnumerationValue>Hockey</EnumerationValue>
                        <EnumerationValue>Football</EnumerationValue>
                        <EnumerationValue>Baseball</EnumerationValue>
                        <EnumerationValue>Basketball</EnumerationValue>
                        <EnumerationValue>Soccer</EnumerationValue>
                        <EnumerationValue>Tennis</EnumerationValue>
                        <EnumerationValue>Fishing</EnumerationValue>
                        <EnumerationValue>Hunting</EnumerationValue>
                        <EnumerationValue>None</EnumerationValue>
                </QualifierDescription>
        </BasicQueryDataModelDescription>
</xxxListQualifiersResponse>
```

Example 25. ListQualifiersResponse Message

The ListQualifiersResponse message returns a BasicQueryDataModelDescription element that contains all the information needed to construct a query request against a logical service's data model.

The ListQualifiersResponse shown in Example 25 shows the data model has one UniqueQualifier, the data model's UniqueQualifier has one Qualifier element, MACAddress, and five other Qualifiers — Age, Income, ZipPlusFour, CreditLimit, and SportsInterest.

The Age qualifier is of type "enumeration" and the allowed values are UnderTwenty, TwentyToForty, FortyToSixty, and OverSixty.

The ZipPlusFour qualifier is of type "string" with a maximum length of ten.

E.3 Query Examples

The QueryRequest is the workhorse of the GIS interface. This message provides consumers with many flexible query alternatives.

Example 26 uses the basic query mechanism to request a UniqueQualifierList for all records with an age Qualifier whose value is "Under20" and specifies the UniqueQualifier to be put in the UniqueQualifierList returned as the result of this request.

```
  <xxxQueryRequest messageId="sca-345" system="GISClient" version="1.0"
identity="40DA910E-01AF-5050-C7EA-5D7B4A475311">
        <Query queryId="234" expandOutput="false"
uniqueQualifierNameRef="MACAddressOnlyUQ">
                <ServiceDataModel>http://SuperDemographics.com</ServiceDataModel>
                <BasicQueryFilter>
                        <BasicFilterElement name="Age" value="Under20"
valueIsRegex="false"/>
                </BasicQueryFilter>
        </Query>
  </xxxQueryRequest>
```

Example 26. QueryRequest Message

The result of this query is shown in Example 27:

```
  <xxxQueryResponse messageId="csa-345" system="GISServer" version="1.0"
identity="40DA910E-01AF-5050-C7EA-5D7B4A475312" messageRef="sca-345">
		<core:StatusCode class="0"/>
		<QueryResult queryRef="234" totalResultSetSize="5" resultSetSize="5">
			<BasicQueryResult>
				<UniqueQualifier>
					<Qualifier name="MACAddress" value="00-1e-c2-01-d3-2d"/>
				</UniqueQualifier>
				<UniqueQualifier>
					<Qualifier name="MACAddress" value="00-1e-52-74-2e-6a"/>
				</UniqueQualifier>
				<UniqueQualifier>
					<Qualifier name="MACAddress" value="00-1e-52-74-23-6d"/>
				</UniqueQualifier>
				<UniqueQualifier>
					<Qualifier name="MACAddress" value="00-50-56-c0-00-01"/>
				</UniqueQualifier>
				<UniqueQualifier>
					<Qualifier name="MACAddress" value="00-50-56-c0-00-08"/>
				</UniqueQualifier>
			</BasicQueryResult>
		</QueryResult>
  </xxxQueryResponse>
```

Example 27. QueryResponse Message

Example 28 uses the basic query mechanism to request a QualifierSet element for the record that has a specific UniqueQualifier element.

```
  <xxxQueryRequest messageId="sca-345" system="GISClient" version="1.0"
identity="40DA910E-01AF-5050-C7EA-5D7B4A475311">
		<Query queryId="233" expandOutput="false">
			<ServiceDataModel>http://SuperDemographics.com</ServiceDataModel>
			<UniqueQualifier>
				<Qualifier name="MACAddress" value="00-1e-c2-01-d3-2d"/>
			</UniqueQualifier>
		</Query>
  </xxxQueryRequest>
```

Example 28. QueryRequest Message

The result of this query is shown in Example 29

```
  <xxxQueryResponse messageId="csa-345" system="GISServer" version="1.0" 
identity="40DA910E-01AF-5050-C7EA-5D7B4A475312" messageRef="sca-345">
		<core:StatusCode class="0"/>
		<QueryResult queryRef="233" totalResultSetSize="1" resultSetSize="1">
			<BasicQueryResult>
				<QualifierSet>
					<Qualifier name="MACAddress" value="00-1e-c2-01-d3-2d"/>
					<Qualifier name="Age" value="Under20"/>
					<Qualifier name="Income" value="Under50K"/>
					<Qualifier name="ZipPlusFour" value="01720"/>
					<Qualifier name="CreditLimit" value="100.00"/>
					<Qualifier name="SportsInterest" value="BasketBall"/>
				</QualifierSet>
			</BasicQueryResult>
		</QueryResult>
  </xxxQueryResponse>
```

Example 29. QueryResponse Message

Example 30 uses the basic query mechanism with the @expandOutput attribute set to "true" to request a QualifierSetList element for all records with an age Qualifier whose value is "Under20".

```
  <xxxQueryRequest messageId="sca-345" system="GISClient" version="1.0" 
identity="40DA910E-01AF-5050-C7EA-5D7B4A475311">
		<Query queryId="234" expandOutput="true">
			<ServiceDataModel>http://SuperDemographics.com</ServiceDataModel>
			<BasicQueryFilter>
				<BasicFilterElement name="Age" value="Under20" 
valueIsRegex="false"/>
			</BasicQueryFilter>
		</Query>
  </xxxQueryRequest>
```

Example 30. QueryRequest Message

The result of this query is shown in Example 31.

```
	<xxxQueryResponse messageId="csa-345" system="GISServer" version="1.0" identity="40DA910E-01AF-5050-
C7EA-5D7B4A475312" messageRef="sca-345">
		<core:StatusCode class="0"/>
		<QueryResult queryRef="234" totalResultSetSize="5" resultSetSize="5">
			<BasicQueryResult>
				<QualifierSet>
					<Qualifier name="MACAddress" value="00-1e-c2-01-d3-2d"/>
					<Qualifier name="Age" value="Under20"/>
					<Qualifier name="Income" value="Under50K"/>
					<Qualifier name="ZipPlusFour" value="01720"/>
					<Qualifier name="CreditLimit" value="100.00"/>
					<Qualifier name="SportsInterest" value="Basketball"/>
				</QualifierSet>
				<QualifierSet>
					<Qualifier name="MACAddress" value="00-1e-52-74-2e-6a"/>
					<Qualifier name="Age" value="Under20"/>
					<Qualifier name="Income" value="Under50K"/>
					<Qualifier name="ZipPlusFour" value="01847"/>
					<Qualifier name="CreditLimit" value="200.00"/>
					<Qualifier name="SportsInterest" value="Football"/>
				</QualifierSet>
				<QualifierSet>
					<Qualifier name="MACAddress" value="00-1e-52-74-2e-6d"/>
					<Qualifier name="Age" value="Under20"/>
					<Qualifier name="Income" value="Under50K"/>
					<Qualifier name="ZipPlusFour" value="01847"/>
					<Qualifier name="CreditLimit" value="100.00"/>
					<Qualifier name="SportsInterest" value="Tennis"/>
				</QualifierSet>
				<QualifierSet>
					<Qualifier name="MACAddress" value="00-50-56-c0-00-01"/>
					<Qualifier name="Age" value="Under20"/>
					<Qualifier name="Income" value="Under50K"/>
					<Qualifier name="ZipPlusFour" value="01720"/>
					<Qualifier name="CreditLimit" value="500.00"/>
					<Qualifier name="SportsInterest" value="Football"/>
				</QualifierSet>
				<QualifierSet>
					<Qualifier name="MACAddress" value="00-50-56-c0-00-08"/>
					<Qualifier name="Age" value="Under20"/>
					<Qualifier name="Income" value="20Kto30K"/>
					<Qualifier name="ZipPlusFour" value="01754"/>
					<Qualifier name="CreditLimit" value="50.00"/>
					<Qualifier name="SportsInterest" value="Fishing"/>
				</QualifierSet>
			</BasicQueryResult>
		</QueryResult>
	</xxxQueryResponse>
```

Example 31. QueryResponse Message

E.4 Notification Registration Examples

The following is a typical NotificationRegistrationRequest example. Note that this example includes the callout instruction for future Notification messages:

```
  <xxxNotificationRegistrationRequest messageId="acs-342" system="GISClient" version="1.0" 
identity="40DA910E-01AF-5050-C7EA-5D7B4A475311">
		<core:Callout message="Notification">
			<core:Address type="SOAP 
1.1">http://10.250.30.77/GISClient/Notification</core:Address>
		</core:Callout>
		<core:Callout>
			<core:Address type="SOAP 
1.1">http://10.250.30.77/GISClient/Default</core:Address>
		</core:Callout>
		<Query queryId="query-1">
			<ServiceDataModel>http://SuperDemographics.com</ServiceDataModel>
			<BasicQueryFilter>
				<BasicFilterElement name="Age" value="Under20" 
valueIsRegex="false"/>
			</BasicQueryFilter>
		</Query>
  </xxxNotificationRegistrationRequest>
```

Example 32. NotificationRegistrationRequest Message

Two things of interest in Example 32 are the core:Callout @message attribute and the core:Address @type attribute. The @message attribute indicates to the logical service implementing the GIS interface that messages of type Notification *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 consumer side. In this case, the address type is "SOAP1.1" 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 narrowed to include the income Qualifier element.

```
  <xxxNotificationRegistrationRequest messageId="acs-342" system="GISClient" version="1.0"
identity="40DA910E-01AF-5050-C7EA-5D7B4A475311">
		<core:Callout message="Notification">
			<core:Address type="SOAP
1.1">http://10.250.30.77/GISClient/Notification</core:Address>
		</core:Callout>
		<core:Callout>
			<core:Address type="SOAP
1.1">http://10.250.30.77/GISClient/Default</core:Address>
		</core:Callout>
		<Query queryId="query-1">
			<ServiceDataModel>http://SuperDemographics.com</ServiceDataModel>
			<BasicQueryFilter>
				<BasicFilterElement name="Age" value="Under20"/>
			</BasicQueryFilter>
			<BasicQueryFilter>
				<BasicFilterElement name="Income" value="20Kto30K"/>
			</BasicQueryFilter>
		</Query>
  </xxxNotificationRegistrationRequest>
```

Example 33. NotificationRegistrationRequest Message

In Example 33, the only two Qualifier elements that are of interest are “Age” and “Income”.

Note that the two BasicFilterElement elements in this example cannot be contained within the same BasicQueryFilter element. BasicFilterElement elements within a single query filter are logically (ANDed) together to form a single query.

E.5 Notification

Notification messages are sent to registered consumers when data in a service's data store has changed in a way that would change the result set of a previously registered query. Changes include the addition of new records, the deletion of old records or updates to existing records.

Changes to the data store are evaluated against consumer registration queries. Matches are packaged up into Notification messages and sent to the registered consumer. Consumers ***shall*** respond to Notification messages with NotificationAcknowledgement messages.

Example 34 contains a Notification message indicating a change to the data store that affects the result set of a previously registered query.

```
  <xxxNotification noticeType="new" messageId="sca-342" system="GISServer" version="1.0" 
identity="40DA910E-01AF-5050-C7EA-5D7B4A475312">
		<QueryResult totalResultSetSize="1" resultSetSize="1" queryRef="query-1">
			<BasicQueryResult>
					<UniqueQualifier>
						<Qualifier name="MACAddress" value="00-50-56-c0-00-
08"/>
					</UniqueQualifier>
			</BasicQueryResult>
		</QueryResult>99
  </xxxNotification>
```

Example 34. Notification Message

The @noticeType attribute of the Notification message is set to "new" indicating that the Qualifier element contained in the BasicQueryResultList element was recently added to the logical service's data store.

APPENDIX F. MUTABLE GIS INTERFACE (NORMATIVE)

F.1 Scope

This appendix defines the Mutable GIS interface extension consistent with the SCTE 130 standard including the Part 8 General Information Service (GIS) interface defined previously herein. The Mutable GIS, a.k.a., the writable GIS interface, provides XML types, elements and attributes enabling an information service Consumer/Client to modify a service data model's definition or its content.

The Mutable GIS is optional. A logical service including the GIS *may* define this extension as mandatory, optional, or excluded at its discretion. If a logical service includes the Mutable GIS extension as either optional or mandatory, this appendix ***shall*** be required and all its contents ***shall*** be considered normative.

F.2 Introduction

A logical service implements the Mutable GIS interface to allow altering of:

- A service data model by adding or removing exising qualifiers
- A service data model description by modifying an existing qualifier's description
- A service data model's qualifier value by changing the data through operations such as increment, decrement, assignment, etc.

An information service Consumer detects support for the Mutable GIS interface extension when the ListSupportedFeaturesResponse message type ***shall*** include the MutableServiceDataProfile element substituted for the gis:ServiceDataModelProfile element. The Mutable GIS provides new message types for mutating a service data model either in a single message update request or via a multi-message update request executed as a batch (i.e., as a single execution run). The message exchange model also supports the asynchronous completion of the mutation operation(s).

Figure 51 illustrates the basic message exchange for a single message mutable operation.

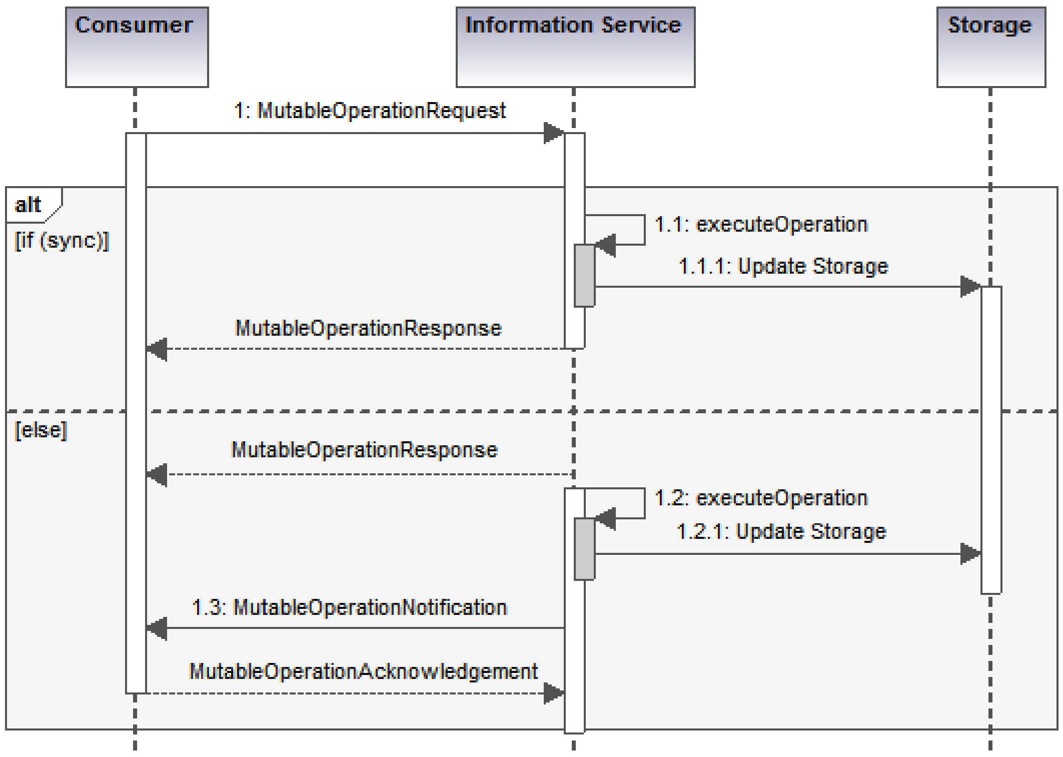


Figure 51. Mutable Operation Sequence Diagram

The information service Consumer initiates the process by sending a MutableOperationRequest message type to the Information Service. The Information Service executes the message body's specified operation(s) updating the service data model storage as appropriate. If the MutableOperationRequest message type indicated synchronous completion, the Information Services sends the response message type when the operation completes. Otherwise, the MutableOperationResponse message type is returned immediately, indicating success via the core:StatusCode element's @class attribute. See [SCTE130-2] for additional information regarding the StatusCode element and @class attribute values. At some future time, the Information Service ***shall*** asynchronously send a MutableOperationNotification message type to the Consumer including the completion status. The Consumer responds with a MutableOperationAcknowledgement message type completing the message exchange sequence.

Clients requesting asynchronous completion notification ***shall*** be prepared for callback prior to the return of the initiating call.

The batch mode message exchange enables a client to supply multiple execution instructions asynchronously and for these commands to be grouped together as a single execution unit identified by an assigned batch identifier, i.e., the @batchId attribute. The information service assigns the batch identifier upon receipt of the batch create request and communicates the value to the Consumer via the @batchId attribute from the BatchCreateResponse message. All instructions ***shall*** be present before commencing command processing via the BatchOperationRequest message type execute signal. This is enforced by the information service by rejecting any requests referencing a batch identifier for an already started batch. Figure 52 illustrates the basic batch operation message type sequence.

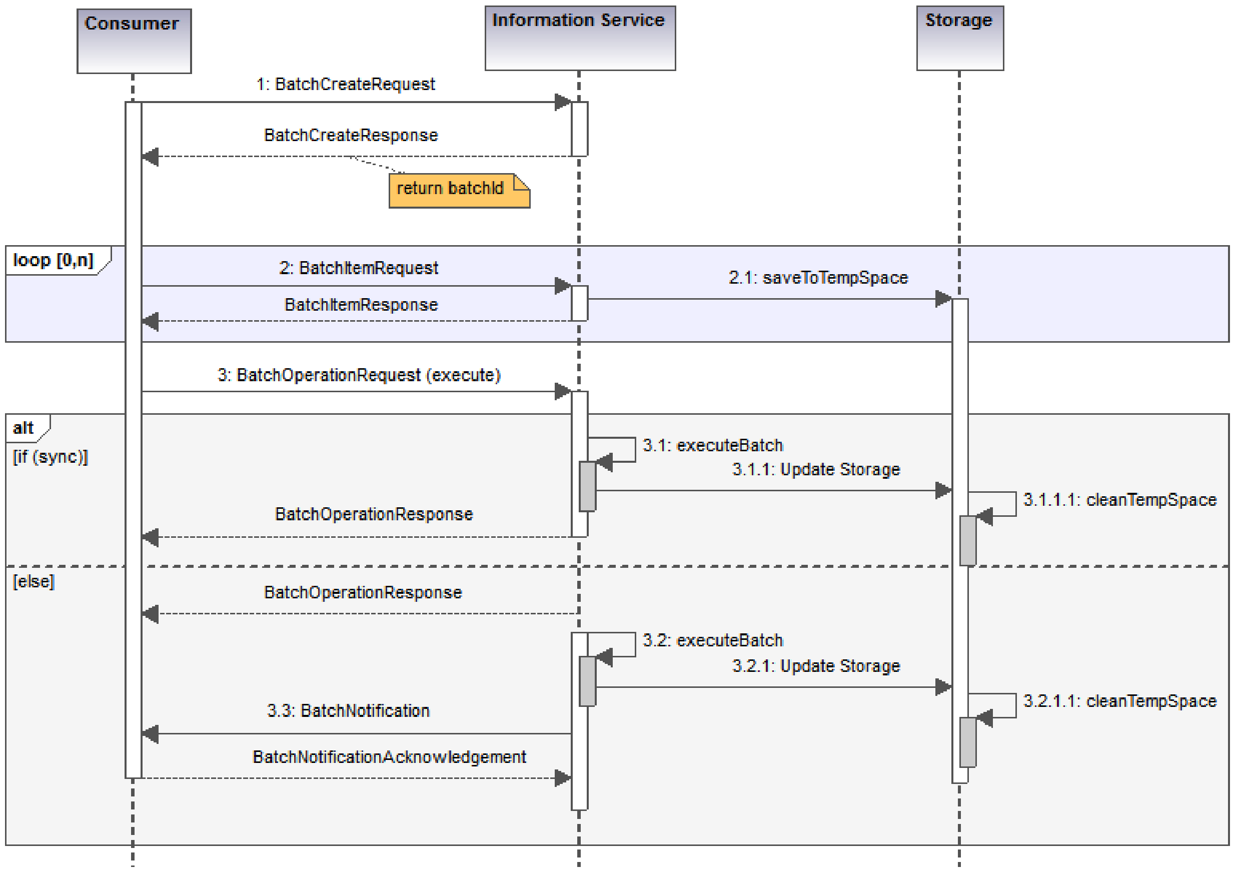


Figure 52. Batch Operation Sequence Diagram

An information service Consumer initiates the message flow by sending the BatchCreateRequest message type to the Information Service. The Information Service responds using the BatchCreateResponse message type. On successful batch creation, the response message type incorporates the Batch element including the mandatory @batchId attribute, which is referenced in all subsequent message type exchanges. The batch identifier ***shall*** be unique, and once assigned it cannot be reused for another batch.

The Consumer sends zero or more BatchItemRequest message types to the Information Service containing execution instructions which are stored pending the “start signal”. The start signal ***shall*** be supplied in the BatchOperationRequest message type using the ExecuteBatch element. The BatchOperationRequest message type *may* alternatively contain a “cancel signal” using the CancelBatch element which immediately terminates the message sequence and allows the Information Service to free all associated resources. The cancel signal *may* occur before any BatchItemRequest message type has been sent by the Consumer. Thus, the zero-message type count possibility.

The batch identifier is obsolete once the start or cancel signal has been received and accepted by the information service, and any subsequent messages from the Consumer referencing it ***shall*** be responded with an error message. The batch identifier is also obsoleted once the information service decides to cancel a batch and notifies the Consumer via the BatchNotification message.

All Consumer message referencing an already started, cancelled or unknown batch ***shall*** result in an error condition.

Similar to the MutableOperationRequest message type in semantics and mechanics, the BatchOperationRequest message type *may* be synchronous or asynchronously executed. When executed asynchronously, the completion indication occurs via the BatchNotification message type.

For all mutable command operations, execution order ***shall*** be determined by the Information Service implementation and is outside the scope of this standard.

F.3 Data Integrity

Mutable operations, either single or multiple message, *may* contain changes for multiple entities. During the execution of the mutable operation, an implementation of the Mutable GIS interface *may* encounter errors that prevents it from completing the operation successfully.

In order to ensure data integrity, a Consumer specifies the transaction level in the mutable operation. A transaction represents a series of operations that act as an atomic unit, meaning that either all occur, or nothing occurs. The transaction unit can either be successful, in which case the changes within that transaction are guaranteed to be completed, or it can fail, in which case the changes within the transaction are rolled back. When a transaction is rolled back, none of the changes within that transaction are completed, and the data in the service is unchanged.

In order to maintain data integrity, mutable operation messages allow a client to specify how the transactions are to be handled via the @transactionLevel attribute. Using this attribute, a Mutable GIS Consumer *may* request that all entities in the mutable operation (or batch) be considered as a single transaction and therefore, if one of the entity instructions fail all of the changes requested in the mutable operation ***shall*** be rolled back. The Consumer *may* also choose to set the transaction handling at the entity level. With transaction level set at the entity level, only the entities that fail to execute ***shall*** be rolled back, successful changes ***shall*** be committed.

F.4 Error Handling

Upon completion of a mutable operation, the Mutable GIS implementation ***shall*** notify the Consumer on the success status of the operation via the core:StatusCode element’s @class attribute. The @class attribute of the core:StatusCode element provides the status of the operation. See [SCTE130-2] for additional information regarding the StatusCode element and @class attribute values.

Depending on the type of operation, the status is provided as part of different messages and elements as detailed in Table 16**.**

Mutable Operation Type	**Status Notification Message Element Type**
Synchronous Single Message	MutableOperationResponseType

Asynchronous Single message	MutableOperationNotificationType
Synchronous Multiple Message (ie. Batch)	BatchOperationResponseType
Asynchronous Multiple Message (ie. Batch)	BatchNotificationType

Table 16**. Message Element Types By Mutable Operation Type**

When the @transactionLevel attribute is set to "entity", it is possible that within the mutable operation or batch, some entities *may* fail to update. In this situation, the operation ***shall*** be considered partially successful and all the failed entities ***shall*** have a corresponding UpdateError element in the notification message sent to the Consumer. The status code for the partially successful operation is defined in this standard as "5" and it represents an extension of the status code list defined in [SCTE130-2].

When the @transactionLevel attribute is set to "message", and within the mutable operation (or batch), some entities fail to update, the operation ***shall*** be considered failed and the information service ***shall*** set the @class attribute value to "1". If the operation (or batch) contains both data definition changes as well as data instance changes, it is possible that the data definition changes are successful but one or more of the the data instance changes fail. Handling this situation is defined by the implementation as the transactional behavior of the data definition changes is outside the scope of this standard.

The list of values of the @class attribute are defined in [SCTE130-2] and extended herein as detailed in Table 17.

@class Attribute Value	**Description**	**Defined By**
0	Success	SCTE130-2
1	Error	SCTE130-2
2	Warning	SCTE130-2
3	Information	SCTE130-2
4	Deferred to the ExternalStatusCode element and there ***shall*** be an ExternalStatusCode element present in this StatusCode element	SCTE130-2
5	Partial Success	Mutable API
6	Batch Cancelled	Mutable API
…	User defined and outside the scope of this standard.	

Table 17**. core:StatusCode Element @class Attribute Values**

F.5 XML Namespaces

Beyond the GIS namespaces defined in [SCTE130-8], this appendix uses the 'mut' prefix, as described in Table 3 for the interface associated with the specific XML namespace URI that ***shall*** be used by all implementations. Unless otherwise stated, all references to XML elements illustrated in this appendix are from the 'mut' namespace. Elements from other namespaces ***shall*** be prefixed with the name of the external namespace, e.g. <core:XXX>.

F.6 Enhanced GIS Message Types

Table 18 identifies message types defined by SCTE 130 Part 8 that are normatively modified via substitution by the Mutable GIS. A service implementing the mutable GIS ***shall*** substitute the Mutable GIS enhanced XML elements in place of the native GIS elements.

Message	Description
gis:ListFeaturesResponseType	Adds a description of mutable specific features via the mandatory inclusion of MutableServiceDataModelProfile element
gis:ListQualifiersResponseType	Adds descriptions of mutable qualifiers via the mandatory inclusion of MutableQualifierDescription element

Table 18**. GIS Message Types Altered by the Mutable GIS**

The following sections detail the GIS defined message types enhanced by mandatory XML element substitution.

F.6.1 gis:ListSupportedFeaturesRequest Message Type

The gis:ListSupportedFeaturesRequest and gis:ListSupportedFeaturesResponse message types allow consumers to inquire about the service data models and advanced query languages supported by a logical service. The Mutable GIS informs an information service Consumer if a service data model is mutable by substituting the gis:ServiceDataModelProfile element with the MutableServiceDataModelProfile element in the gis:ListSupportedFeaturesResponse message type. The MutableServiceDataModelProfile element describes the mutable service data model including its mutable characterstics. See Section F.9.12 for more information on MutableServiceDataModelProfile element.

Additionally, a Mutable GIS ***shall*** supply the additional core:Callout elements associated with the Mutable GIS message type endpoints. If an information service is using a default message endpoint supporting the Mutable GIS endpoints, no additional core:Callout elements are required. Otherwise, the information service specific extension values derived from Table 3 ***shall*** additionally be included in the core:Callout element sequence. NOTE: These values are not the exact values that ***shall*** appear. The exact message names are individual standard dependent.

Section F.7 introduces and provides message type details.

F.6.2 gis:ListQualifiers Messages

The gis:ListQualifiersRequest and gis:ListQualifiersResponse message types allow Consumers to discover the qualifiers associated with service data models that are queryable using the basic query interface. A logical service supporting the Mutable GIS ***shall*** indicate in the gis:ListQualifiersResponse message type which service data model qualifiers are mutable. This indication ***shall*** be accomplished utilizing the MutableQualifierDescription element substituted for the gis:QualifierDescription element. See also F.9.11 for more information on MutableQualifierDescription element.

If a specific service data model does not support the basic query interface, the gis:ListQualifiersResponse message type ***shall*** return an error.

F.7 GIS Mutable Message Types

Table 19 identifies the message types specific to Mutable GIS.

Message	Description
MutableOperationRequestType	A self-contained request to modify a service data model definition and/or service data model qualifier values
MutableOperationResponseType	Response to a MutableOperationRequest
BatchCreateRequestType	Creates a multi-message mutable operations construct (i.e., a batch).
BatchCreateResponseType	Response to a BatchCreateRequest. Returns the batch identifier.
BatchItemRequestType	Request to add mutable operations to a batch
BatchItemResponseType	Response to a BatchItemRequest
BatchOperationRequestType	Request to start or cancel the batch of mutable operations
BatchOperationResponseType	Response to a BatchOperationRequest
ListBatchRequestType	Request for information about one or more created mutable operation batches
ListBatchResponseType	Response to a ListBatchRequest
MutableOperationNotificationType	Asynchronous notification message providing completion status relative to a self-contained MutableOperationRequest
MutableOperationAcknowledgementType	Response to a MutableOperationNotification
BatchNotificationType	Asynchronous notification message providing completion status relative to a multi-message batch operation created via a BatchCreateRequest
BatchAcknowledgementType	Response to a BatchNotification

Table 19**. Mutable GIS Message Types**

F.7.1 MutableOperation Message Types

The MutableOperation message exchange initiates an execution of a mutable operation sequence defined within the request message body. The operational sequence *may* provide completion status either synchronously or asynchronously as indicated within the request message. An asynchronously completing operation sequence provides completion status using the MutableOperationNotification message type. See Section F.7.6 for more information on the MutableOperationNotification message exchange.

F.7.1.1 MutableOperationRequest Message Type

The MutableOperationRequest message type requests modifications to a service data model and/or service data model's qualifier values. The message type includes transaction control information.

The XML schema definition for this message is illustrated in Figure 53.

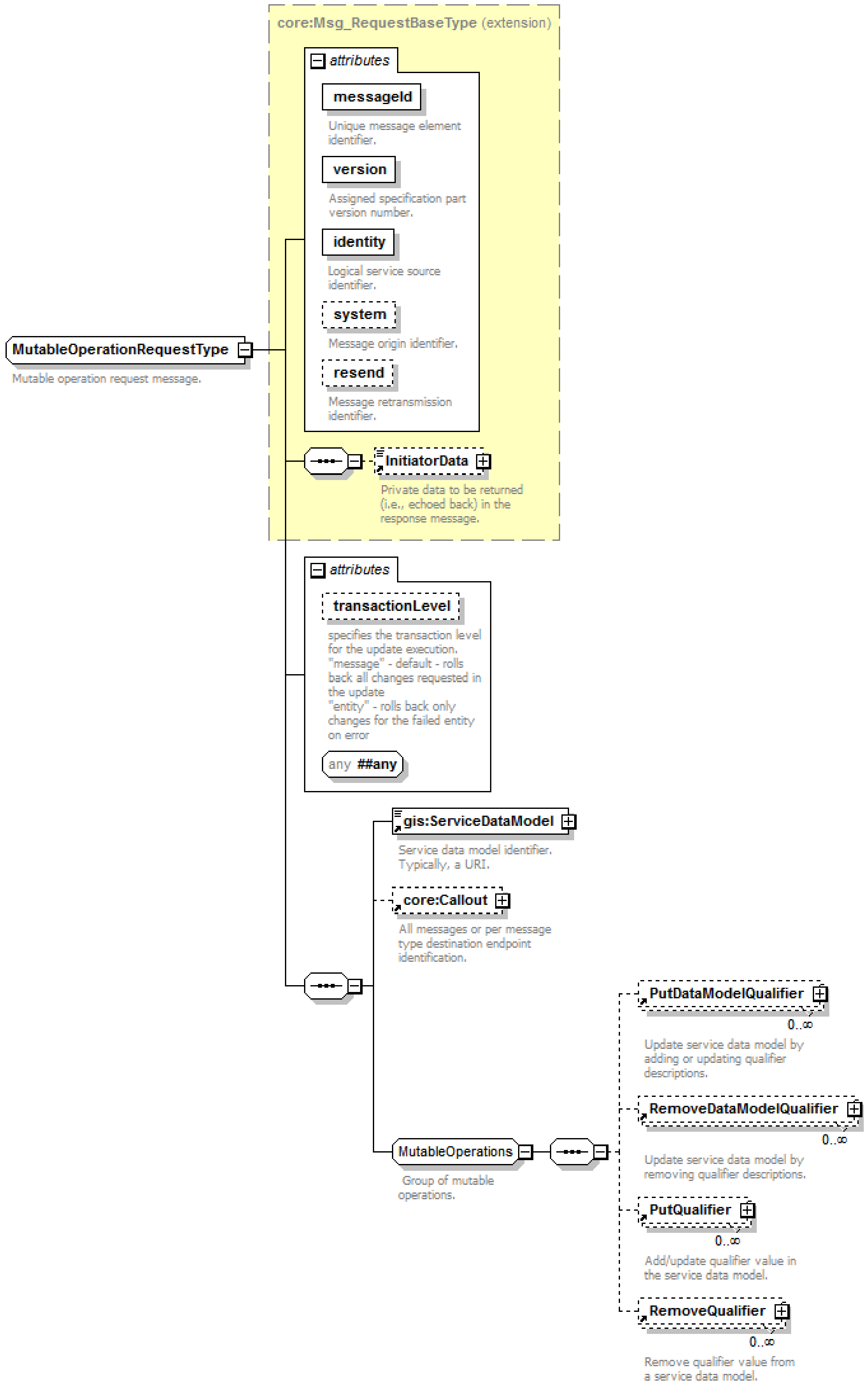

core:Msg_RequestBaseType (extension)
attributes
messageId
Unique message element identifier.
version
Assigned specification part version number.
identity
Logical service source identifier.
system
Message origin identifier.
resend
Message retransmission identifier.
MutableOperationRequestType
Mutable operation request message.
InitiatorData
Private data to be returned (i.e., echoed back) in the response message.
attributes
transactionLevel
specifies the transaction level for the update execution. "message" - default - rolls back all changes requested in the update "entity" - rolls back only changes for the failed entity on error
any ##any
gis:ServiceDataModel
Service data model identifier. Typically, a URI.
core:Callout
All messages or per message type destination endpoint identification.
MutableOperations
Group of mutable operations.
PutDataModelQualifier
0..∞
Update service data model by adding or updating qualifier descriptions.
RemoveDataModelQualifier
0..∞
Update service data model by removing qualifier descriptions.
PutQualifier
0..∞
Add/update qualifier value in the service data model.
RemoveQualifier
0..∞
Remove qualifier value from a service data model.

Figure 53. MutableOperationRequestType XML Schema

The MutableOperationRequest message type is derived from core:Msg_RequestBaseType and defines the following attributes and elements in addition to those already defined in core:Msg_RequestBaseType.

@transactionLevel [Optional, TransactionLevelTypeEnumeration] — The @transactionLevel attribute allows a Consumer to control how the information service ***shall*** handle the mutable transaction. See Table 6 for more information on this attribute's possible values and how they affect the update process behavior. See F.8.3 for more information on TransactionLevelTypeEnumeration.

@##any [Optional] — The @##any attribute enables a logical service incorporating the SCTE 130 Part 8 interface to optionally extend concrete messages derived from SCTE 130 Part 8 message types with attributes not specified by the schema.

gis:ServiceDataModel [Required] — The gis:ServiceDataModel element contains a service data model reference that ***shall*** be used to satisfy the update. For additional information on the gis:ServiceDataModel element see this document, SCTE 130 Part 8.

core:Callout [Optional] — The core:Callout element provides a Consumer callback message and address information if the Consumer desires asyncronous completion status notification. If a Consumer provides the core:Callout element, the operation and its completion status ***shall*** be performed asynchronously. Thus, the information service ***shall*** respond to the Consumer that this message has been received and accepted, and upon completion of the requested mutable operation(s), the information service ***shall*** notify the Consumer of the completion status(es) at the address specified in the core:Callout element. See [SCTE130-2] for a complete description of the core:Callout element. The core:Callout element's @message attribute ***shall*** either be empty or ***shall*** contain a derivative from the MutableOperationNotification message type (i.e., xxxMutableOperationNotification and e.g., SISMutableOperationNotification).

PutDataModelQualifier [Optional] — The PutDataModelQualifier element ***shall*** add or update a qualifier description in the specified service data model. See F.9.13 for more information on PutDataModelQualifier element.

RemoveDataModelQualifier [Optional] — The RemoveDataModelQualifier element ***shall*** remove a qualifier description from the specified service data model. See F.9.19 for more information on RemoveDataModelQualifier element.

PutQualifier [Optional] — The PutQualifier element ***shall*** add or update a qualifier instance value in the specified service data model. See F.9.14 for more information on PutQualifier element.

RemoveQualifier [Optional] — The RemoveQualifier element ***shall*** remove a qualifier instance value from the specified service data model. See F.9.20 for more information on RemoveQualifier element.

F.7.1.2 MutableOperationResponse Message Type

Upon receipt of a MutableOperationRequest message type, the information service ***shall*** respond with a MutableOperationResponse message type.

The core:StatusCode element meaning depends on the type of completion type notification:

- If asynchronous (indicated via the presence of the core:Callout element) the core:StatusCode ***shall*** indicate if the request message was successfully accepted. The operation completion status ***shall*** be provided via a MutableOperationNotification message type sent by the information service at a later time.

- If synchronous (default, the core:Callout element is not present) the core:StatusCode ***shall*** indicate the operation completion status.

The XML schema definition for this message is illustrated in Figure 54.

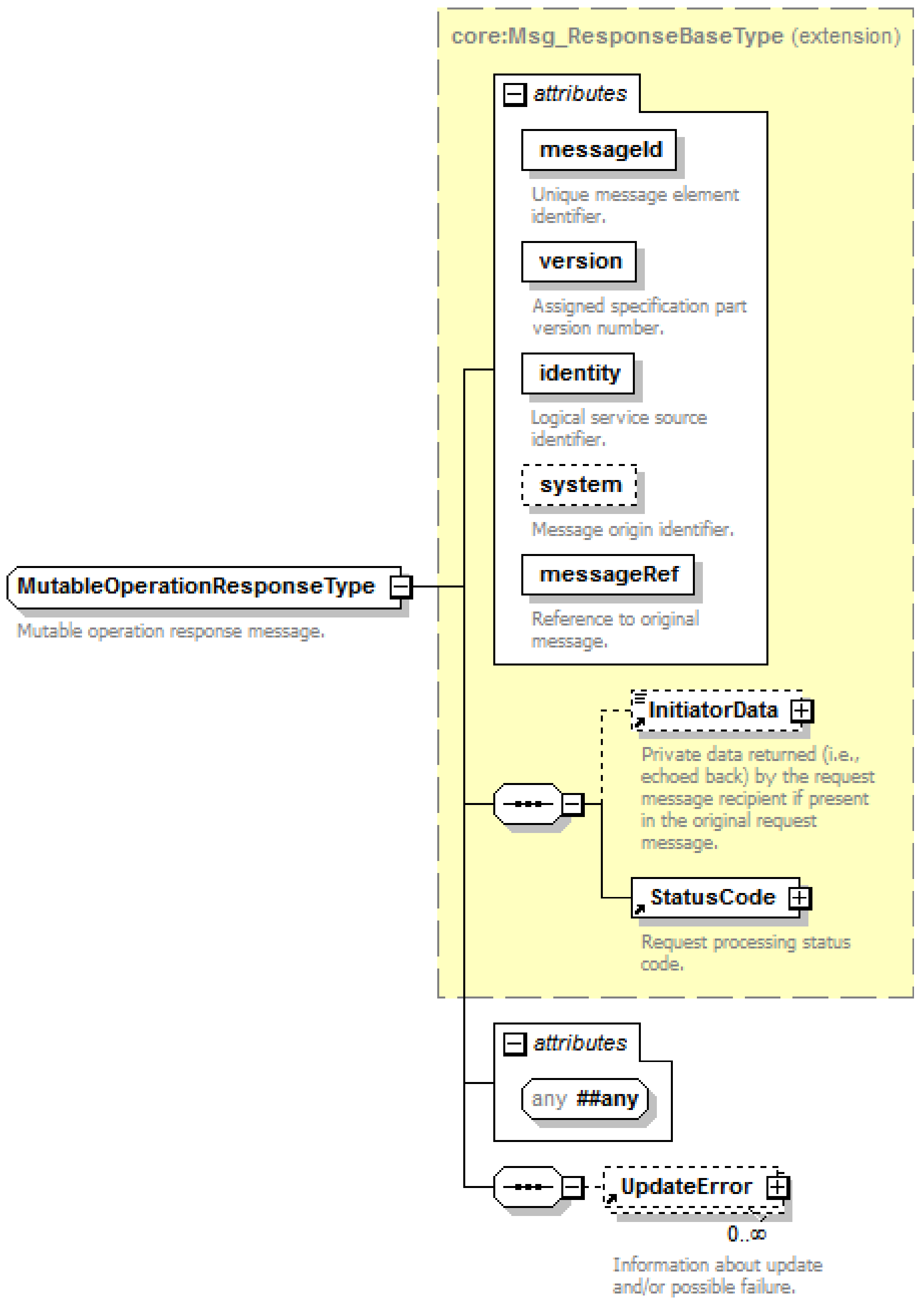

core:Msg_ResponseBaseType (extension)
attributes
messageId
Unique message element identifier.
version
Assigned specification part version number.
identity
Logical service source identifier.
system
Message origin identifier.
messageRef
Reference to original message.
MutableOperationResponseType
Mutable operation response message.
InitiatorData
Private data returned (i.e., echoed back) by the request message recipient if present in the original request message.
StatusCode
Request processing status code.
attributes
any ##any
UpdateError
0..∞
Information about update and/or possible failure.

Figure 54. MutableOperationResponseType XML Schema

The MutableOperationResponse message type is derived from core:Msg_ResponseBaseType and defines the following attributes and elements in addition to those that are already defined by the abstract base message core:Msg_ResponseBaseType.

@##any [Optional] — The @##any attribute enables a logical service incorporating the SCTE 130 Part 8 interface to optionally extend concrete messages derived from SCTE 130 Part 8 message types with attributes not specified by the schema.

UpdateError [Optional] — An UpdateError element ***shall*** be present if the logical service failed to complete a mutable operation successfully. Each UpdateError element references a specific request message type operation via the required @updateIdRef attribute. The element provides additional information regarding the specific failed operation. See F.9.21 for further information on the UpdateError element.

F.7.2 BatchCreate Messages

The BatchCreate message exchange is the first message pair exchanged in a multi-message sequence to define a multi-operation mutable transaction. The BatchCreate message exchange initiates the mutable operation. Upon successful creation of the batch, the logical service provides an identifier for the batch known as @batchId. This identifier ***shall*** be referenced in all subsequent message exchanges about the batch.

F.7.2.1 BatchCreateRequest Message Type

The BatchCreateRequest message type allows a logical service Consumer to setup a batch mutable operation consisting of modifications to a service data model and/or service data model’s qualifier values. The message type includes a reference to the service data model updated by the batch mutable operation.

The XML schema definition for this message is illustrated in Figure 55.

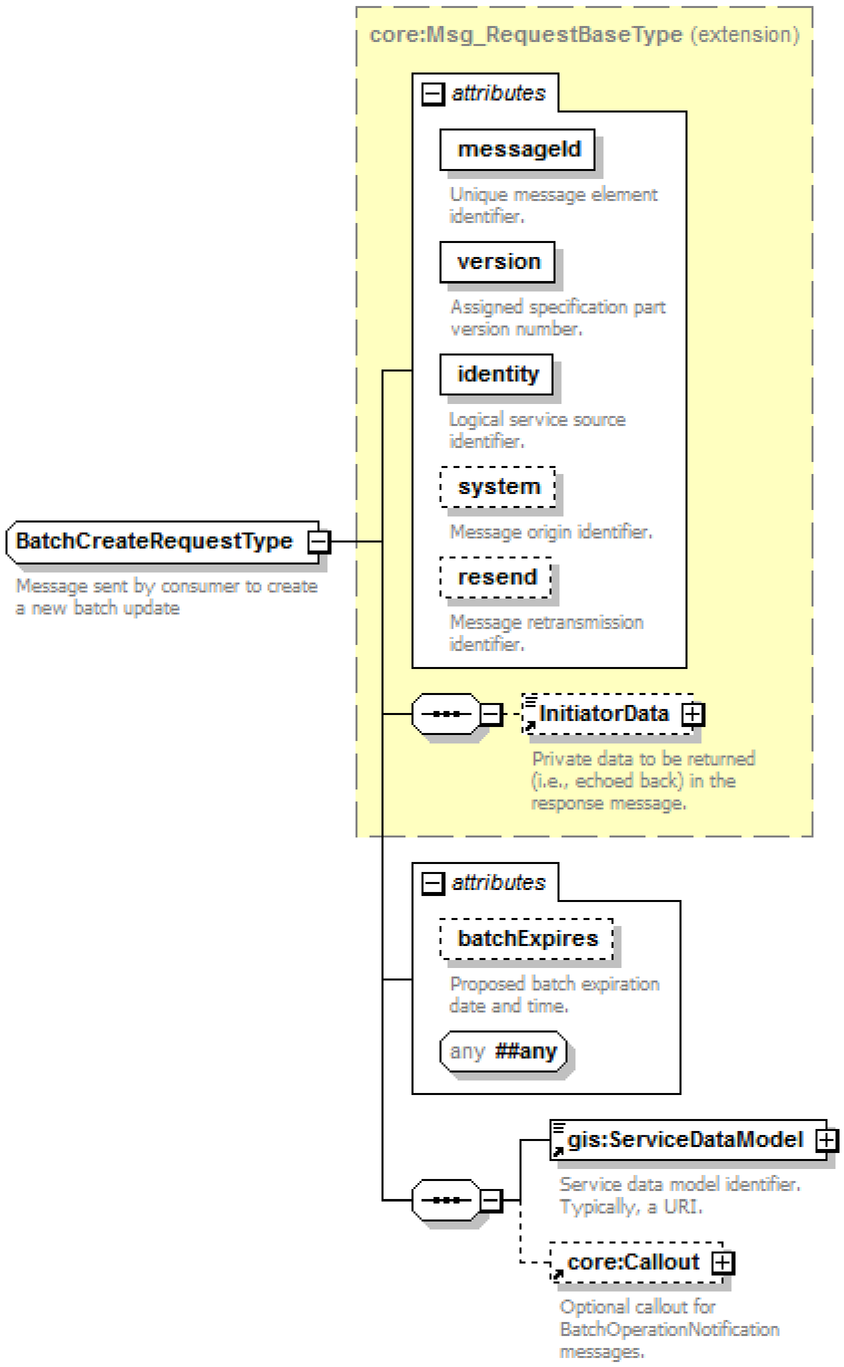

core:Msg_RequestBaseType (extension)
attributes
messageId
Unique message element identifier.
version
Assigned specification part version number.
identity
Logical service source identifier.
system
Message origin identifier.
resend
Message retransmission identifier.
BatchCreateRequestType
Message sent by consumer to create a new batch update
InitiatorData
Private data to be returned (i.e., echoed back) in the response message.
attributes
batchExpires
Proposed batch expiration date and time.
any ##any
gis:ServiceDataModel
Service data model identifier. Typically, a URI.
core:Callout
Optional callout for BatchOperationNotification messages.

Figure 55. BatchCreateRequestType XML Schema

The MutableOperationRequest message type is derived from the type core:Msg_RequestBaseType and defines the following attributes and elements in addition to those already defined in core:Msg_RequestBaseType.

@batchExpires [Optional, core:dateTimeTimezoneType] — The batch transaction *may* contain multiple message exchanges until the operation is actually started. These message exchanges *may* take a substantial amount of time and the information service *may* choose to store the batch information locally until the batch is started. The @batchExpires attribute enables the information service Consumer to suggest a time when the batch ***shall*** be considered expired. This information assists the information service with managing its resources. The value of the @batchExpires represents a date when a batch is considered expired relative to the receipt of the start or cancel signal for the batch operation. Once started, a batch can no longer expire regardless of the processing duration.

@##any [Optional] — The @##any attribute enables a logical service incorporating the SCTE 130 Part 8 interface to optionally extend concrete messages derived from SCTE 130 Part 8 message types with attributes not specified by the schema.

gis:ServiceDataModel [Required] — The gis:ServiceDataModel element contains a reference to a service data model that ***shall*** be used to satisfy the update. For additional information on the gis:ServiceDataModel element see this document, SCTE 130 Part 8.

core:Callout [Optional] - The core:Callout element provides callback message and address information to the logical service. If the core:Callout is provided, the information service ***shall*** use this information to notify the Consumer in case it needs to cancel the batch. See [SCTE130-2] for a complete description of the core:Callout element.

F.7.2.2 BatchCreateResponseMessageType

Upon receipt of a BatchCreateRequest message type, a logical service ***shall*** respond with a BatchCreateResponse message type. The response provides the Consumer with a batch identifier that ***shall*** be used in all subsequent message exchanges referencing the batch.

The XML schema definition for this message is illustrated in Figure 56.

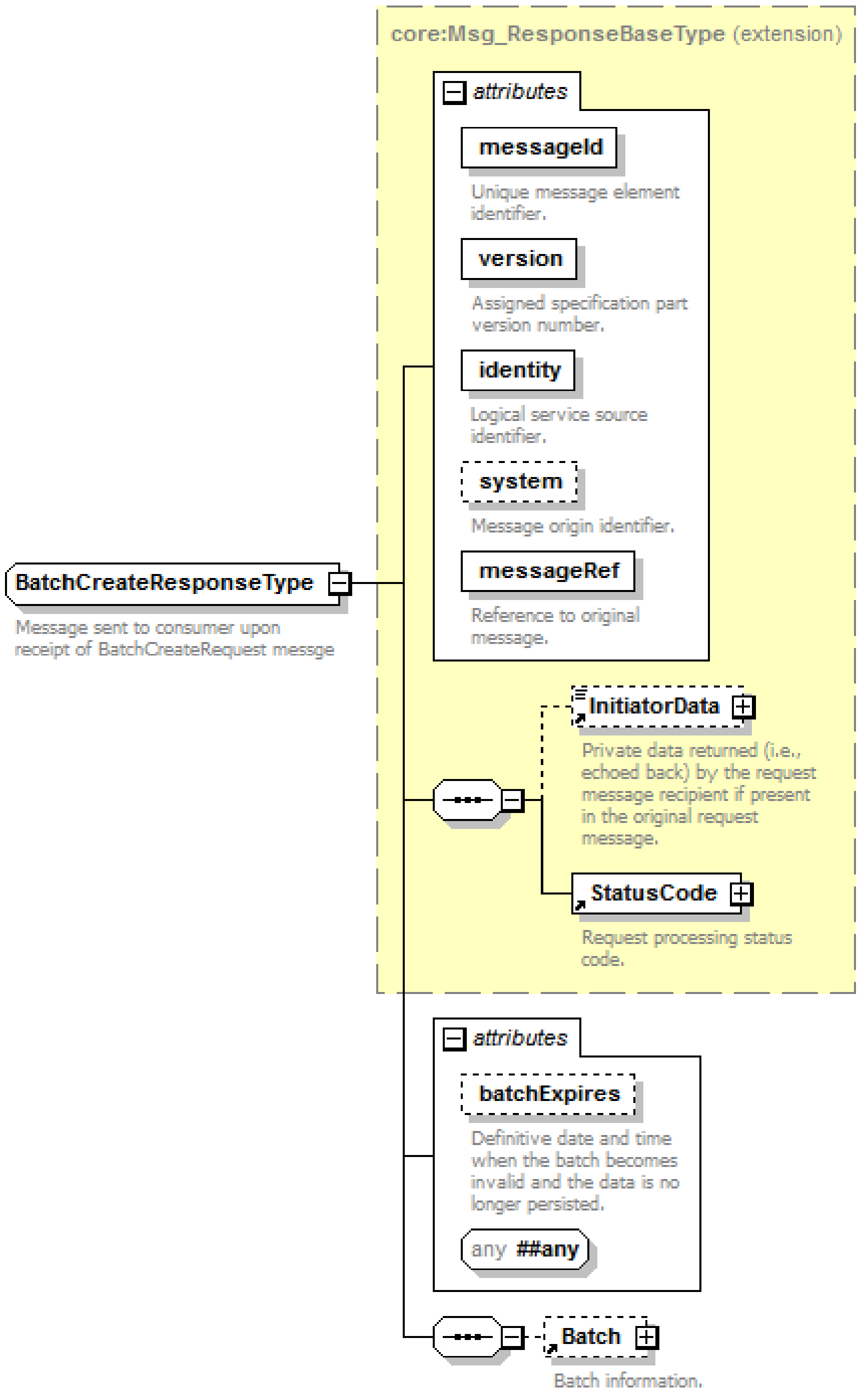

core:Msg_ResponseBaseType (extension)
attributes
messageId
Unique message element identifier.
version
Assigned specification part version number.
identity
Logical service source identifier.
system
Message origin identifier.
messageRef
Reference to original message.
BatchCreateResponseType
Message sent to consumer upon receipt of BatchCreateRequest messge
InitiatorData
Private data returned (i.e., echoed back) by the request message recipient if present in the original request message.
StatusCode
Request processing status code.
attributes
batchExpires
Definitive date and time when the batch becomes invalid and the data is no longer persisted.
any ##any
Batch
Batch information.

Figure 56. BatchCreateResponseType XML Schema

The BatchCreateResponse message type is derived from the core:Msg_ResponseBaseType and defines the following attributes and elements in addition to those that are already defined on the abstract base message core:Msg_ResponseBaseType.

@batchExpires [Optional, core:dateTimeTimezoneType] —The @batchExpires attribute, if present, ***shall*** inform the Consumer when the batch ***shall*** be considered expired. If the Consumer does not start the batch before the date and time indicated, the batch ***shall*** be cancelled and all batch data ***shall*** no longer be persisted. References to cancelled/expired batches ***shall*** result in an error condition. Cancelled batches ***shall not*** be included in the ListBatchResponse message type.

@##any [Optional] — The @##any attribute enables a logical service incorporating the SCTE 130 Part 8 interface to optionally extend concrete messages derived from SCTE 130 Part 8 message types with attributes not specified by the schema.

Batch [Optional] — If the batch was successfully created, the logical service ***shall*** include the Batch element that includes the @batchId attribute. See F.9.1 for more information on the Batch element.

F.7.3 BatchItem Messages

After creating a batch, a Consumer sends zero or more BatchItemRequest messages that add service data model changes to the mutable operation. Each BatchItemRequest is processed synchronously by the information service that ***shall*** accept the changes without actually performing the requested updates to the service data model. Instead, the changes are cached locally by the information service pending a BatchOperation message exchange.

F.7.3.1 BatchItemRequest Message Type

The Consumer references the batch in the request via @batchIdRef. The logical service ***shall*** add the supplied mutable operations to the referenced batch and this information ***shall not*** be applied to the service data model until a batch execute request is received. See F.7.4.1 for more information about batch execute request.

The XML schema definition for this message is illustrated in Figure 57.

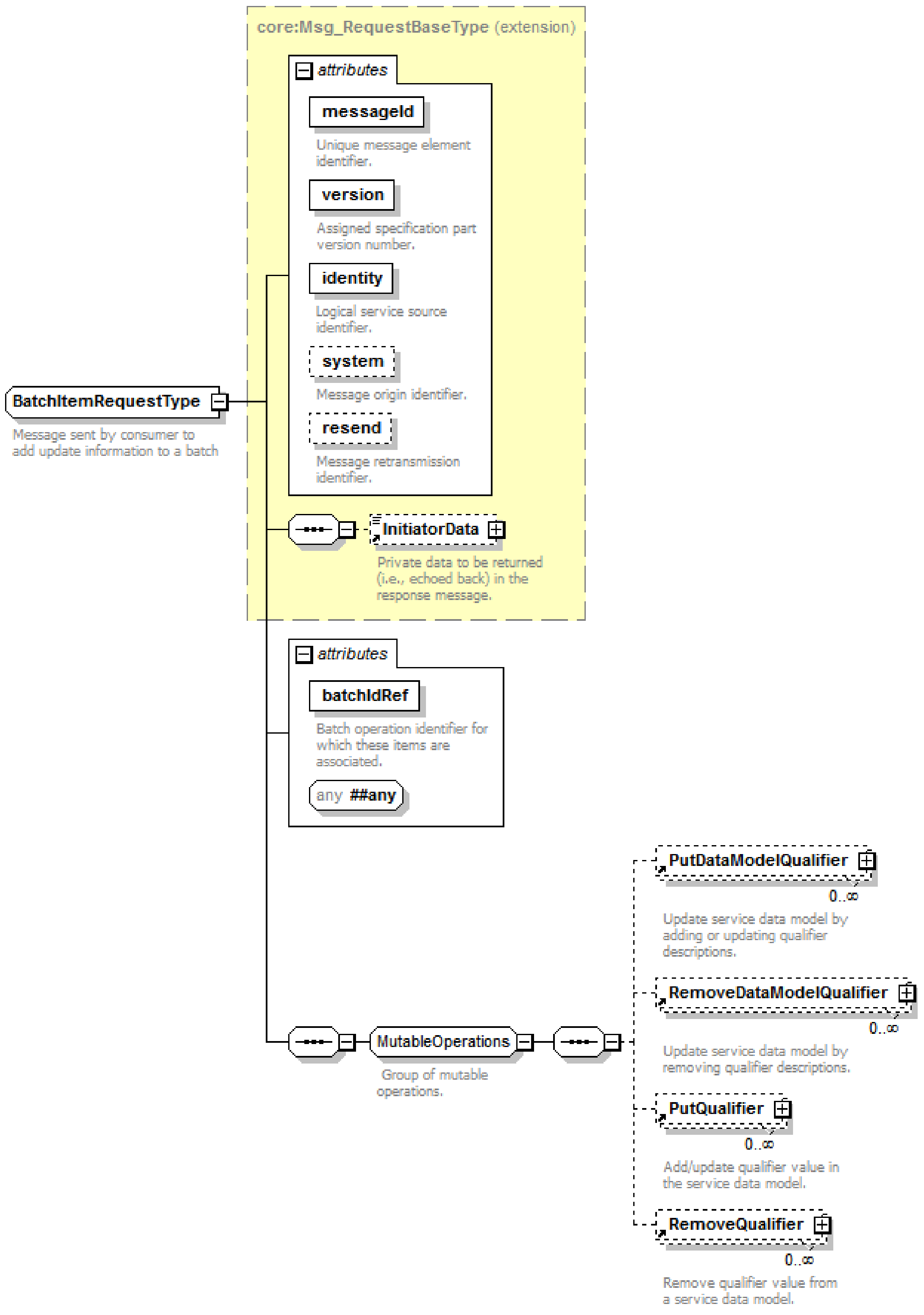

core:Msg_RequestBaseType (extension)
attributes
messageId
Unique message element identifier.
version
Assigned specification part version number.
identity
Logical service source identifier.
system
Message origin identifier.
resend
Message retransmission identifier.
InitiatorData
Private data to be returned (i.e., echoed back) in the response message.
BatchItemRequestType
Message sent by consumer to add update information to a batch
attributes
batchIdRef
Batch operation identifier for which these items are associated.
any ##any
MutableOperations
Group of mutable operations.
PutDataModelQualifier
0..∞
Update service data model by adding or updating qualifier descriptions.
RemoveDataModelQualifier
0..∞
Update service data model by removing qualifier descriptions.
PutQualifier
0..∞
Add/update qualifier value in the service data model.
RemoveQualifier
0..∞
Remove qualifier value from a service data model.

Figure 57. BatchItemRequestType XML Schema

The BatchItemRequest message type is derived from the core:Msg_RequestBaseType and defines the following attributes and elements in addition to those already defined in core:Msg_RequestBaseType.

@batchIdRef [Required, batchIdRefAttrType] — The @batchIdRef attribute ***shall*** contain a reference to the batch to which the mutable operations are being added. See F.8.2 for more information on batchIdRefAttrType.

@##any [Optional] — The @##any attribute enables a logical service incorporating the SCTE 130 Part 8 interface to optionally extend concrete messages derived from SCTE 130 Part 8 message types with attributes not specified by the schema.

PutDataModelQualifier [Optional] — The PutDataModelQualifier element ***shall*** add or update a qualifier description in the previously prescribed service data model. See F.9.13 for more information on PutDataModelQualifier element.

RemoveDataModelQualifier [Optional] — The RemoveDataModelQualifier element ***shall*** remove a qualifier description from the previously prescribed service data model. See F.9.19 for more information on RemoveDataModelQualifier element.

PutQualifier [Optional] — The PutQualifier element ***shall*** add or update a qualifier instance value in the previously prescribed service data model. See F.9.14 for more information on PutQualifier element.

RemoveQualifier [Optional] — The RemoveQualifier element ***shall*** remove a qualifier instance value from the previously prescribed service data model. See F.9.20 for more information on RemoveQualifier element.

F.7.3.2 BatchItemResponse Message Type

Upon receipt of the BatchItemRequest message type, a logical service ***shall*** respond with the BatchItemResponse message type. The core:StatusCode element indicates if the request message type's mutable operations were accepted. It ***shall not*** provide any form of mutable operation completion status.

The XML schema definition for this message is illustrated in Figure 58.

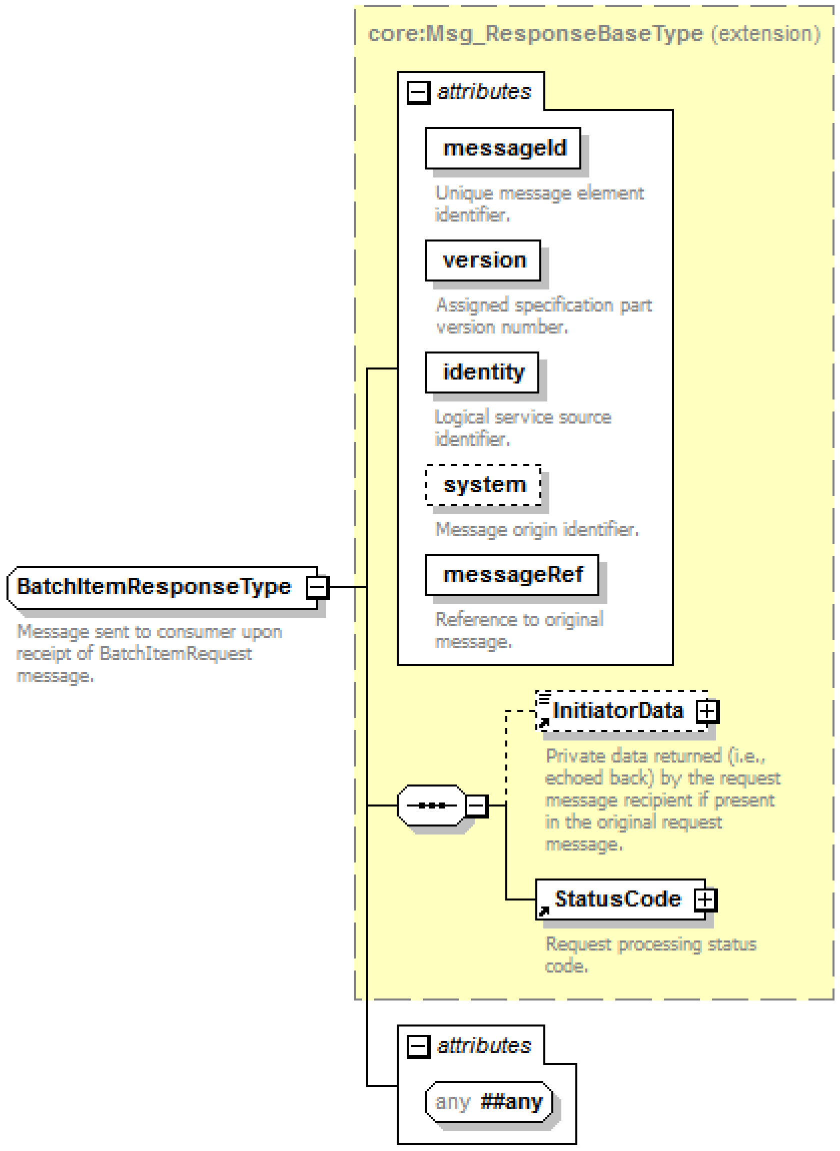

core:Msg_ResponseBaseType (extension)
attributes
messageId
Unique message element identifier.
version
Assigned specification part version number.
identity
Logical service source identifier.
system
Message origin identifier.
messageRef
Reference to original message.
BatchItemResponseType
Message sent to consumer upon receipt of BatchItemRequest message.
InitiatorData
Private data returned (i.e., echoed back) by the request message recipient if present in the original request message.
StatusCode
Request processing status code.
attributes
any ##any

Figure 58. BatchItemResponseType XML Schema

The BatchItemResponse message type is derived from the core:Msg_ResponseBaseType and defines the following attributes and elements in addition to those that are already defined on the abstract base message core:Msg_ResponseBaseType.

@##any [Optional] — The @##any attribute enables a logical service incorporating the SCTE 130 Part 8 interface to optionally extend concrete messages derived from SCTE 130 Part 8 message types with attributes not specified by the schema.

F.7.4 BatchOperation Messages

The BatchOperation message exchange is the third and final step from the batch message sequence. A logical service Consumer uses the BatchOperation message exchange to start or cancel the execution of a batch mutable operation.

F.7.4.1 BatchOperationRequest Message Type

The operational sequence *may* provide completion status either synchronously or asynchronously as indicated within the request message. An asynchronously completing operation sequence provides completion status using the BatchNotification message type.

The BatchOperationRequest message ***shall*** commence (start) or cancel a specified batch of mutable operations previously delivered via zero or more BatchItemRequest message types. The start and cancel operations are mutally exclusive. If a batch has been started, it can no longer be cancelled by the Consumer. Also, once a batch has been started, a Consumer can not modify its instructions.

The XML schema definition for this message is illustrated in Figure 59.

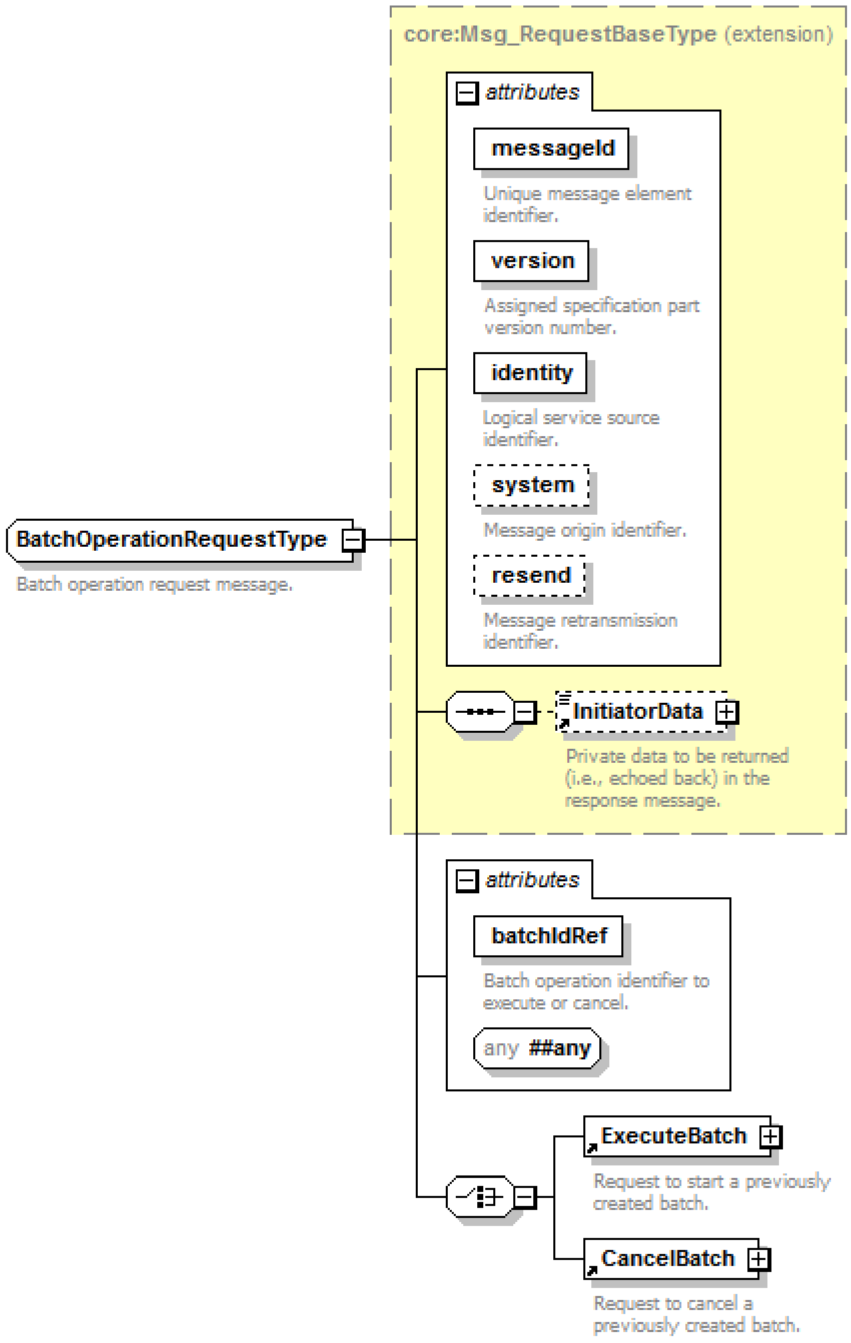

core:Msg_RequestBaseType (extension)
attributes
messageId
Unique message element identifier.
version
Assigned specification part version number.
identity
Logical service source identifier.
system
Message origin identifier.
resend
Message retransmission identifier.
BatchOperationRequestType
Batch operation request message.
InitiatorData
Private data to be returned (i.e., echoed back) in the response message.
attributes
batchIdRef
Batch operation identifier to execute or cancel.
any ##any
ExecuteBatch
Request to start a previously created batch.
CancelBatch
Request to cancel a previously created batch.

Figure 59. BatchOperationRequestType XML Schema

The MutableOperationRequest message type is derived from the type core:Msg_RequestBaseType and defines the following attributes and elements in addition to those already defined in core:Msg_RequestBaseType.

@batchIdRef [Required, batchIdRefAttrType] — The @batchIdRef attribute ***shall*** contain a reference to the mutable operations batch to which the contained command applies. See F.8.2 for more information on batchIdRefAttrType.

@##any [Optional] — The @##any attribute enables a logical service incorporating the SCTE 130 Part 8 interface to optionally extend concrete messages derived from SCTE 130 Part 8 message types with attributes not specified by the schema.

ExecuteBatch [Required under Choice] — The ExecuteBatch element ***shall*** start a batch previously created via the BatchCreateRequest message type and populated with one or more mutable operations via the BatchItemRequest message type. See F.9.6 for more information on ExecuteBatch element.

CancelBatch [Required under Choice] — The CancelBatch element ***shall*** cancel a batch previously created via the BatchCreateRequest message type. Once a batch has been cancelled, the batch mutable operations are no longer persisted by the information service storage. Cancelled batches ***shall not*** be referenced in future requests and cancelled batches ***shall not*** be included in the ListBatchResponse message type. Referencing cancelled batch operations ***shall*** result in an error condition. See F.9.5 for more information on CancelBatch element.

F.7.4.2 BatchOperationResponse Message Type

Upon receipt of the BatchOperationRequest message, a logical service implementing the mutable GIS ***shall*** respond with the BatchOperationResponse message.

The XML schema definition for this message is illustrated in Figure 60.

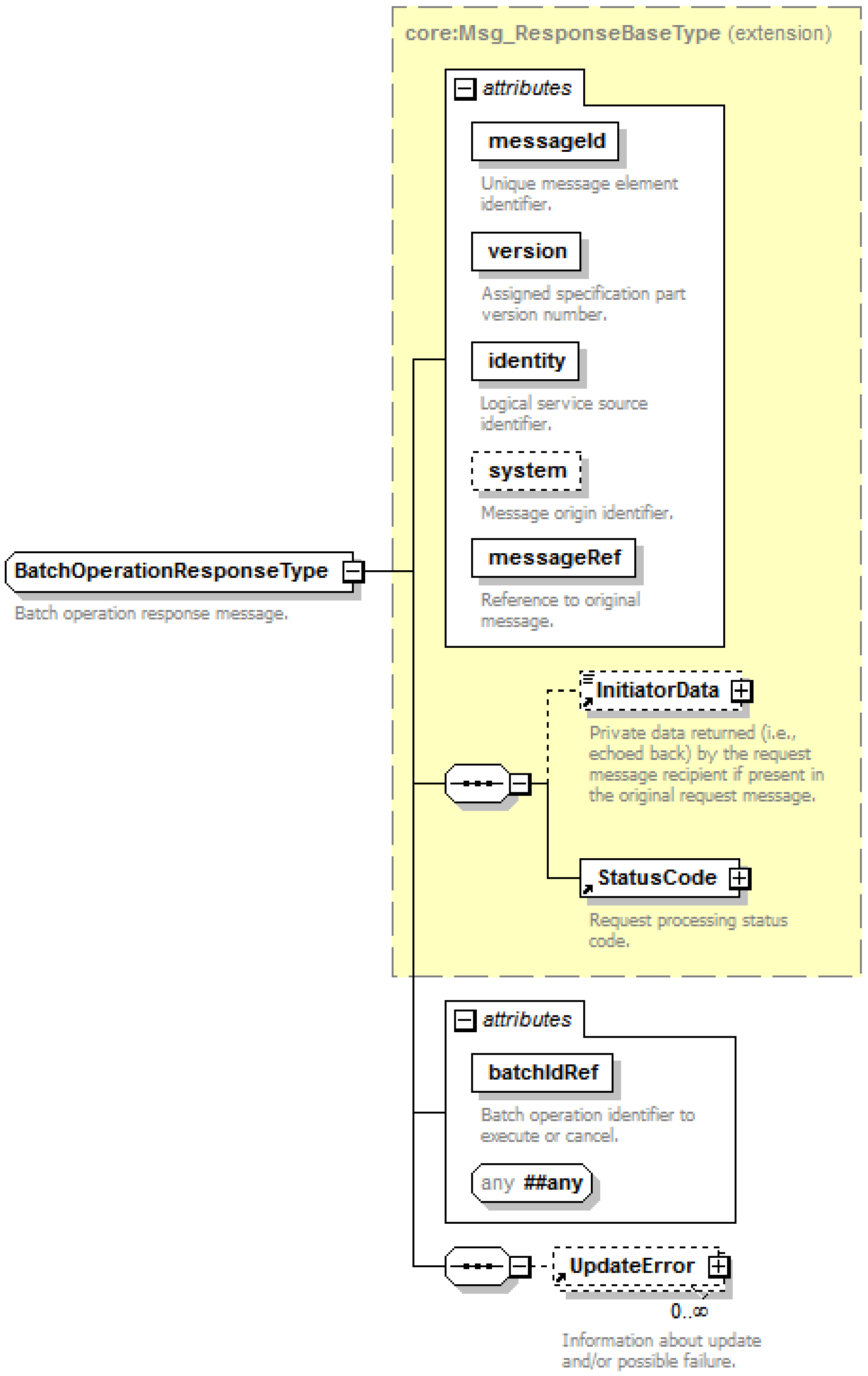

core:Msg_ResponseBaseType (extension)
attributes
messageId
Unique message element identifier.
version
Assigned specification part version number.
identity
Logical service source identifier.
system
Message origin identifier.
messageRef
Reference to original message.
BatchOperationResponseType
Batch operation response message.
InitiatorData
Private data returned (i.e., echoed back) by the request message recipient if present in the original request message.
StatusCode
Request processing status code.
attributes
batchIdRef
Batch operation identifier to execute or cancel.
any ##any
UpdateError
0..∞
Information about update and/or possible failure.

Figure 60. BatchOperationResponseType XML Schema

The BatchOperationResponse message type is derived from the type core:Msg_ResponseBaseType and defines the following attributes and elements in addition to those that are already defined on the abstract base message core:Msg_ResponseBaseType.

@batchIdRef [Required, batchIdRefAttrType] — The @batchIdRef attribute ***shall*** reference a batch identifier to which this response applies. See F.8.2 for more information on batchIdRefAttrType.

@##any [Optional] — The @##any attribute enables a logical service incorporating the SCTE 130 Part 8 interface to optionally extend concrete messages derived from SCTE 130 Part 8 message types with attributes not specified by the schema.

UpdateError [Optional] — An UpdateError element ***shall*** be present if the logical service failed to complete a mutable operation successfully. Each UpdateError element references a mutable operation element via the required @updateIdRef attribute. The element provides additional information regarding a specific failed operation. See F.9.21 for further information on the UpdateError element.

F.7.5 Batch Notification Messages

A logical service implementing the mutable GIS ***shall*** support the exchange of BatchNotification and BatchNotificationAcknowledgement type messages with Consumers.

F.7.5.1 BatchNotification Message Type

The BatchNotification message is used by the information service to report the batch execution status for a batch created by the Consumer and executed in asynchronous mode. The information service ***shall*** include UpdateError elements in the notification message if any error occurred during the batch execution.

The BatchNotification message is also used by the information service to report the cancellation of a batch in case the information service decides to cancel it before receiving the execution request from the Consumer. The information service *may* cancel a batch at any time and for any reason provided that it informs the Consumer using the BatchNotification message. The information service notifies the Consumer that a batch is cancelled by setting the value of the @class attribute of the core:StatusCode element to “6”.

The XML schema definition for this message is illustrated in Figure 61.

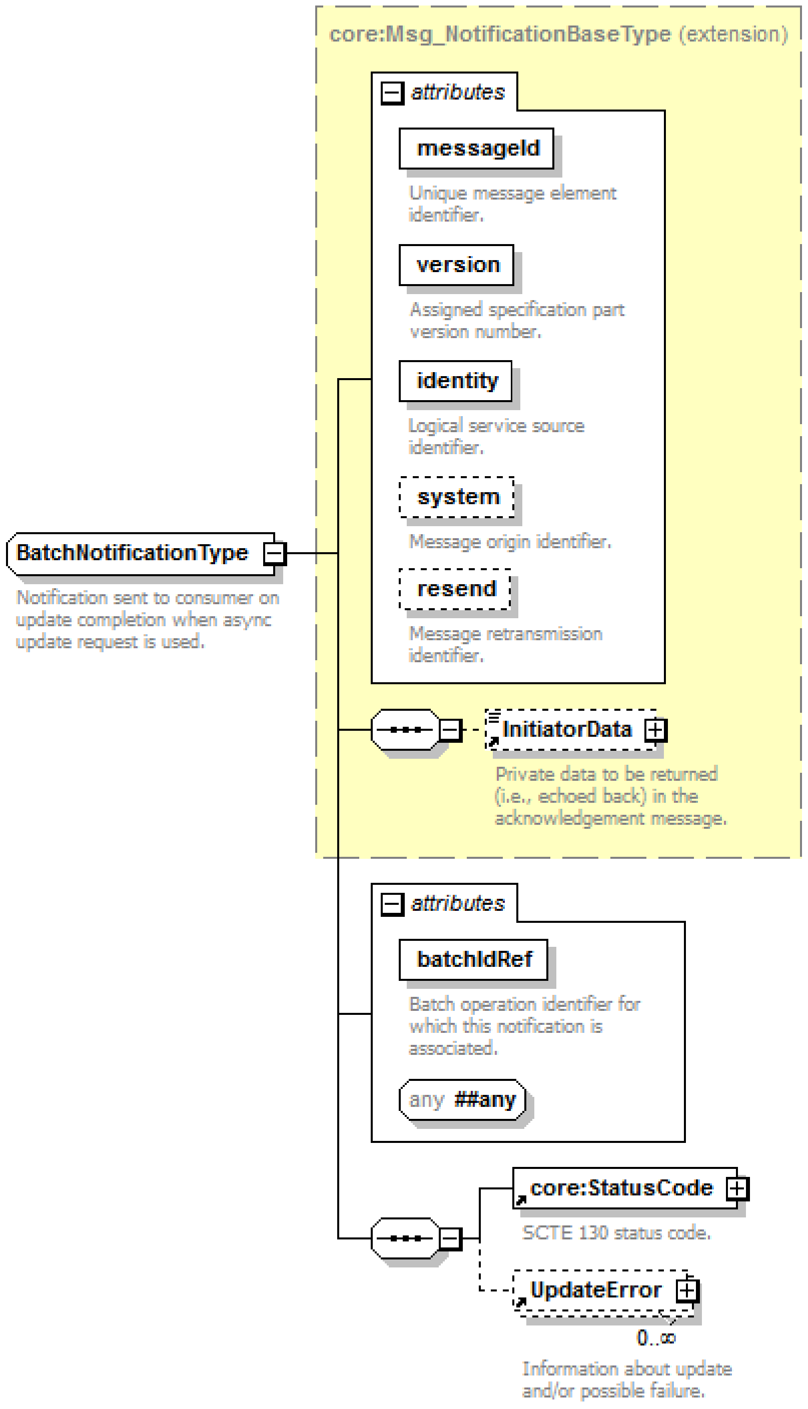

core:Msg_NotificationBaseType (extension)
attributes
messageId
Unique message element identifier.
version
Assigned specification part version number.
identity
Logical service source identifier.
system
Message origin identifier.
resend
Message retransmission identifier.
BatchNotificationType
Notification sent to consumer on update completion when async update request is used.
InitiatorData
Private data to be returned (i.e., echoed back) in the acknowledgement message.
attributes
batchIdRef
Batch operation identifier for which this notification is associated.
any ##any
core:StatusCode
SCTE 130 status code.
UpdateError
0..∞
Information about update and/or possible failure.

Figure 61. BatchNotificationType XML Schema

The BatchNotification message type is derived from the type core:Msg_NotificationBaseType and defines the following attributes and elements in addition to those defined by the core:Msg_NotificationBaseType.

@batchIdRef [Required, batchIdRefAttrType] — The @batchIdRef attribute is a reference to the identifier of the batch to which the operation applies. See F.8.2 for more information on batchIdRefAttrType.

@##any [Optional] — The @##any attribute enables a logical service incorporating the SCTE 130 Part 8 interface to optionally extend concrete messages derived from SCTE 130 Part 8 message types with attributes not specified by the schema.

core:StatusCode [Required] — The core:StatusCode element contains the operation processing status. The @class attribute of the core:StatusCode element provides the status of the batch. The @class attribute is also utilized to describe a cancelled batch by providing the value "6". A batch can be cancelled for different reasons. Regardless of the reason, the value of the @class attribute ***shall*** be "6" for a cancelled batch and the information service ***shall*** provide additional information via the @detail attribute and/or the Note child element. See F.4 for more information on core:StatusCode.

UpdateError [Optional] — The UpdateError element is added to the response if the logical service failed to complete the mutable operation successfully. Each UpdateError element references a specific operation from the request via required @updateIdRef attribute and provides more information regarding that specific failed operation. See F.9.21 for further information on UpdateError element.

F.7.5.2 BatchNotificationAcknowledgement Message Type

Upon the receipt of a BatchNotification message type, a GIS Consumer ***shall*** respond with a BatchNotificationAcknowledgement message type.

The XML schema for the BatchNotificationAcknowledgement message type is shown in Figure 62.

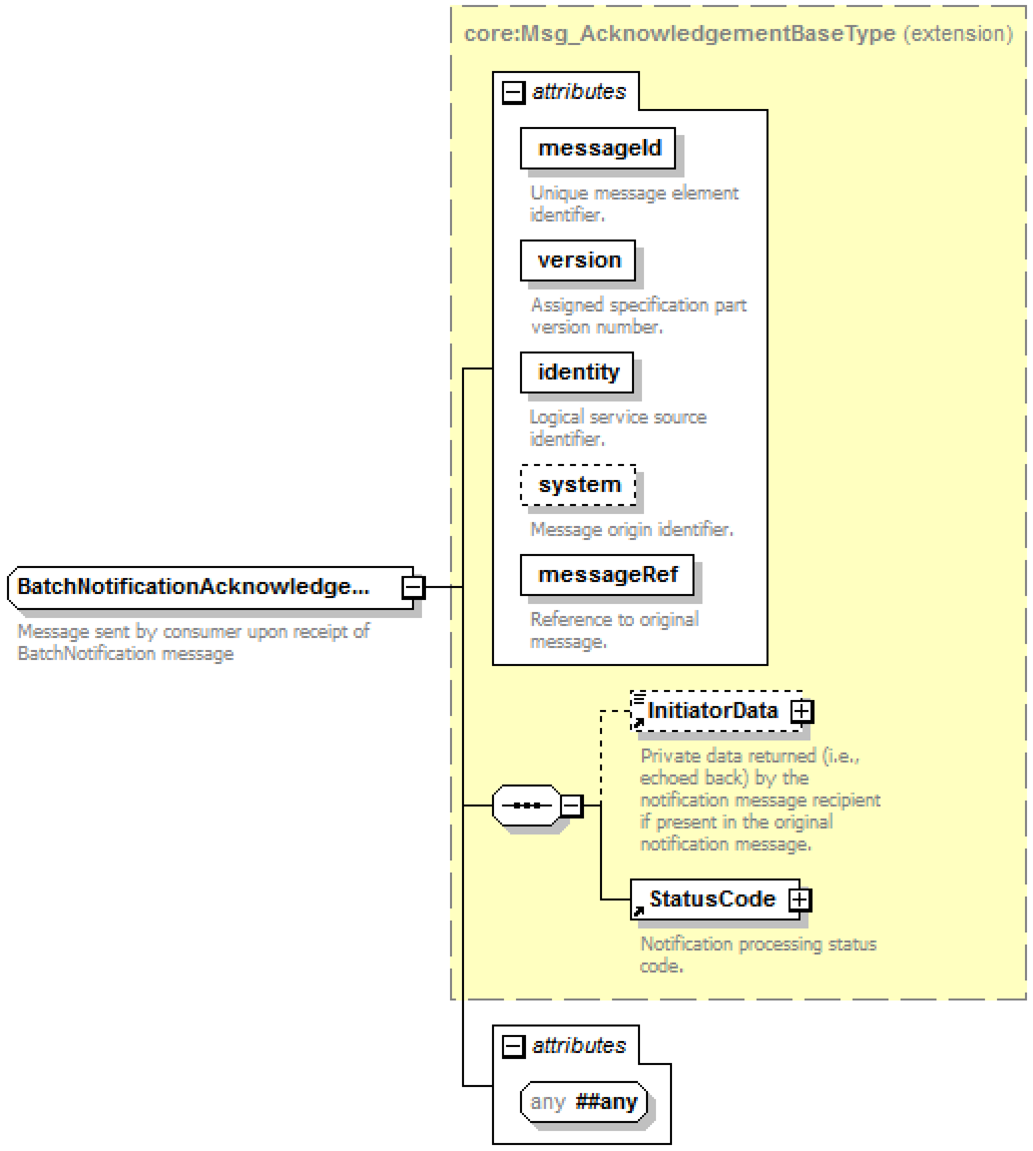

core:Msg_AcknowledgementBaseType (extension)
attributes
messageId
Unique message element identifier.
version
Assigned specification part version number.
identity
Logical service source identifier.
system
Message origin identifier.
messageRef
Reference to original message.
BatchNotificationAcknowledge...
Message sent by consumer upon receipt of BatchNotification message
InitiatorData
Private data returned (i.e., echoed back) by the notification message recipient if present in the original notification message.
StatusCode
Notification processing status code.
attributes
any ##any

Figure 62. BatchNotificationAcknowledgementType XML Schema

The BatchNotificationAcknowledgement message type is derived from the core namespace base type core:Msg_AcknowledgementBaseType and defines no elements in addition to those defined by the core:Msg_AcknowledgementBaseType.

@##any [Optional] — The @##any attribute enables a logical service incorporating the SCTE 130 Part 8 interface to optionally extend concrete messages derived from SCTE 130 Part 8 message types with attributes not specified by the schema.

F.7.6 Mutable Operation Notification Messages

A logical service implementing the mutable GIS ***shall*** support the mutable operation notification message exchange.

F.7.6.1 MutableOperationNotification Message Type

When a single mutable operation is requested to be executed in asynchronous mode, the information service ***shall*** return the response to the Consumer immediately and send a notification using the MutableOperationNotification message upon operation completion.

The XML schema for the MutableOperationNotification message type is shown invFigure 63.

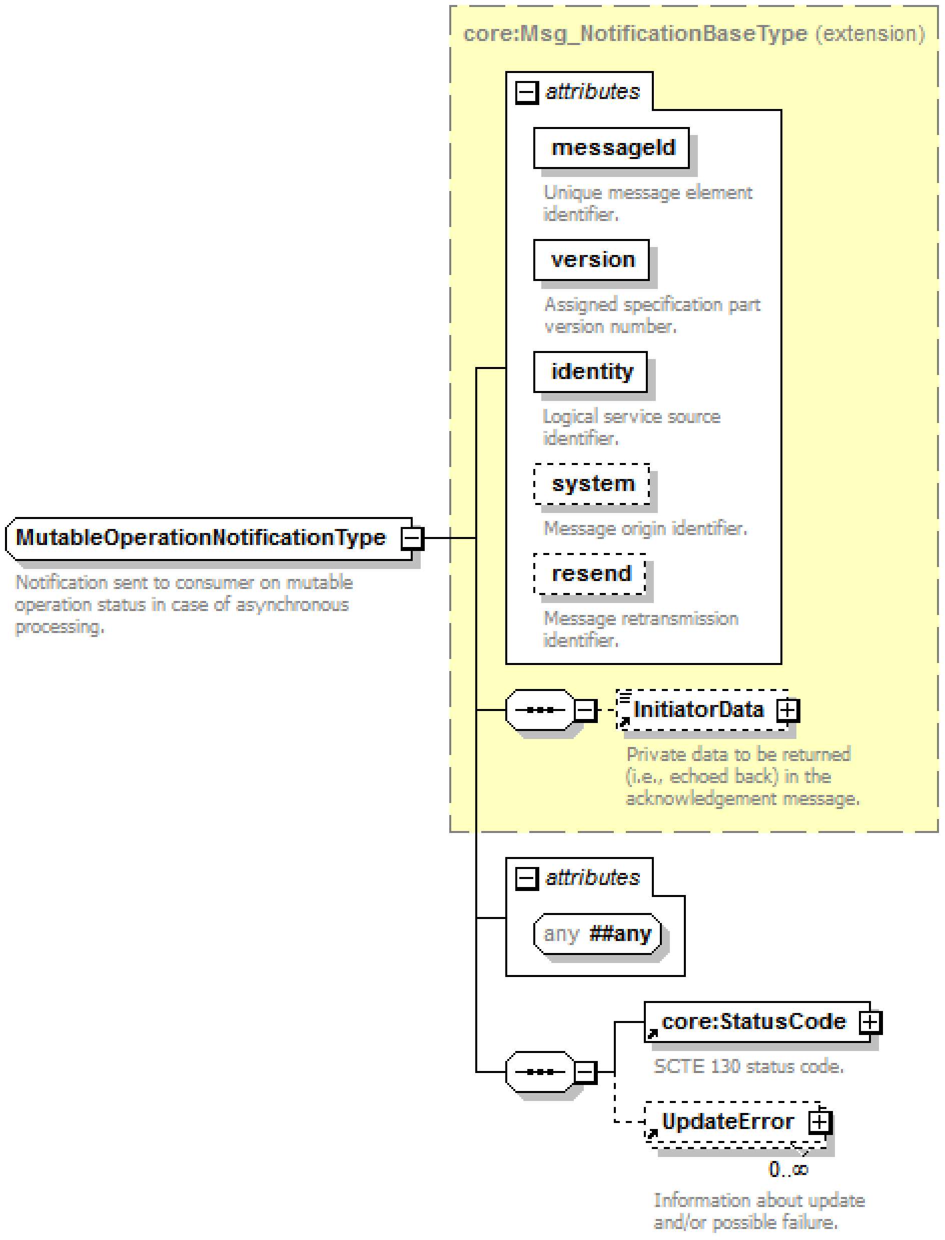

core:Msg_NotificationBaseType (extension)
attributes
messageId
Unique message element identifier.
version
Assigned specification part version number.
identity
Logical service source identifier.
system
Message origin identifier.
resend
Message retransmission identifier.
MutableOperationNotificationType
Notification sent to consumer on mutable operation status in case of asynchronous processing.
InitiatorData
Private data to be returned (i.e., echoed back) in the acknowledgement message.
attributes
any ##any
core:StatusCode
SCTE 130 status code.
UpdateError
0..∞
Information about update and/or possible failure.

Figure 63. MutableOperationNotificationType XML Schema

The MutableOperationNotification message type is derived from the core namespace base type core:Msg_NotificationBaseType and defines the following attributes and elements in addition to those defined by the core:Msg_NotificationBaseType.

@##any [Optional] — The @##any attribute enables a logical service incorporating the SCTE 130 Part 8 interface to optionally extend concrete messages derived from SCTE 130 Part 8 message types with attributes not specified by the schema.

core:StatusCode [Required] — The core:StatusCode element contains the operation processing status. The @class attribute of the core:StatusCode element provides the status of the operation. See F.4 for more information on core:StatusCode.

UpdateError [Optional] — The UpdateError element is added to the response if the logical service failed to complete the mutable operation successfully. Each UpdateError element references a specific operation from the request via required @updateIdRef attribute and provides more information regarding that specific failed operation. See F.9.21 for further information on UpdateError element.

F.7.6.2 MutableOperationAcknowledgement Message Type

Upon receipt of a MutableOperationNotification message, a Consumer ***shall*** respond with a MutableOperationAcknowledgement message.

The XML schema for the MutableOperationAcknowledgement message type is shown in Figure 64.

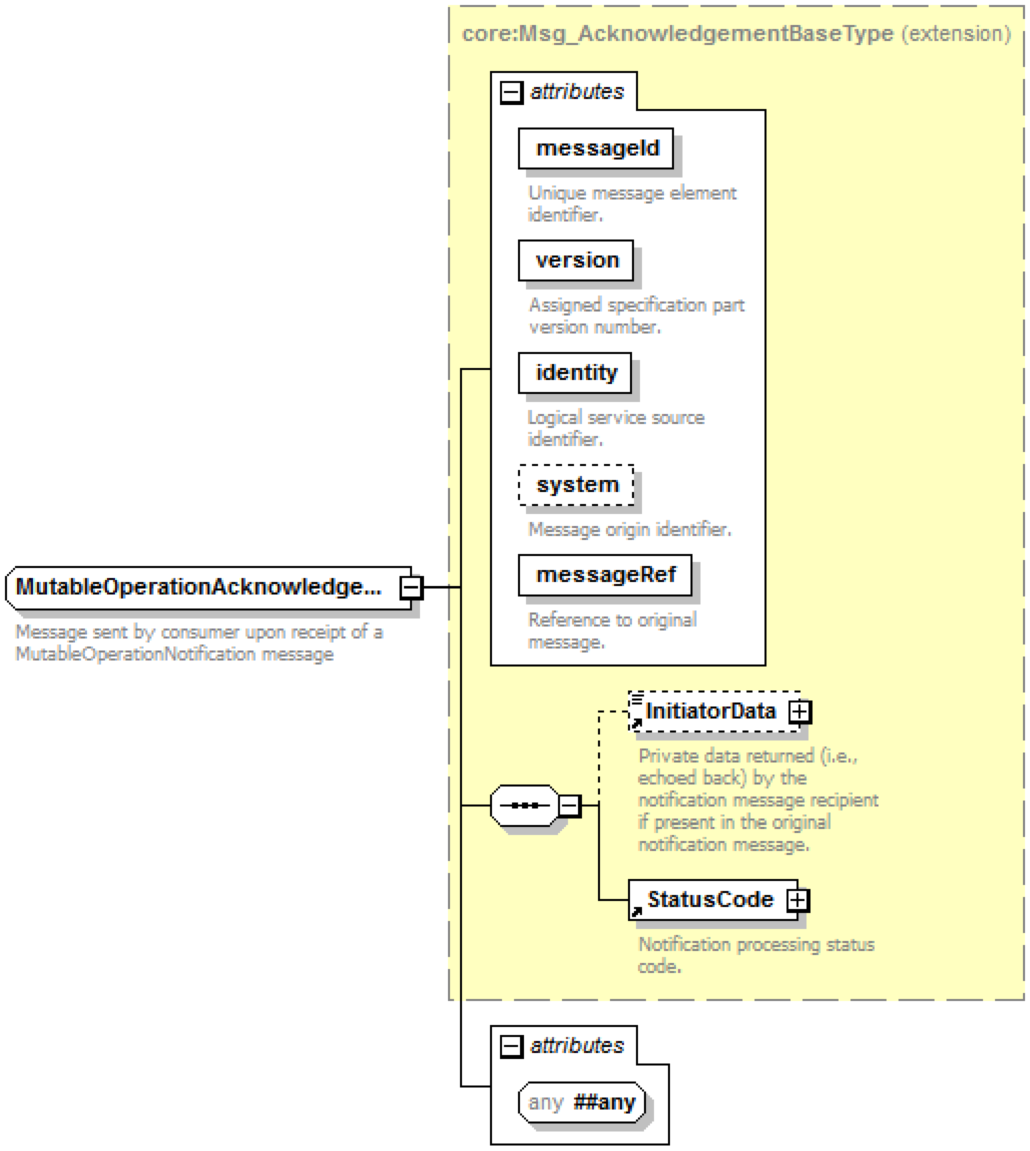

core:Msg_AcknowledgementBaseType (extension)
attributes
messageId
Unique message element identifier.
version
Assigned specification part version number.
identity
Logical service source identifier.
system
Message origin identifier.
messageRef
Reference to original message.
MutableOperationAcknowledge...
Message sent by consumer upon receipt of a MutableOperationNotification message
InitiatorData
Private data returned (i.e., echoed back) by the notification message recipient if present in the original notification message.
StatusCode
Notification processing status code.
attributes
any ##any

Figure 64. MutableOperationAcknowledgementType XML Schema

The MutableOperationNotificationAcknowledgement message type is derived from the core namespace base type core:Msg_AcknowledgementBaseType and defines no elements in addition to those defined by the core:Msg_AcknowledgementBaseType.

@##any [Optional] — The @##any attribute enables a logical service incorporating the SCTE 130 Part 8 interface to optionally extend concrete messages derived from SCTE 130 Part 8 message types with attributes not specified by the schema.

F.7.7 List Batch Messages

A logical service implementing the mutable GIS ***shall*** support the list batch message exchange

F.7.7.1 ListBatchRequest Message Type

A logical service Consumer can use the ListBatchRequest message to get status information about one or more mutable operation batches created or in progress. Completed batches, whether successful or not ***shall not*** appear in the list.

The XML schema definition for this message is illustrated in Figure 65.

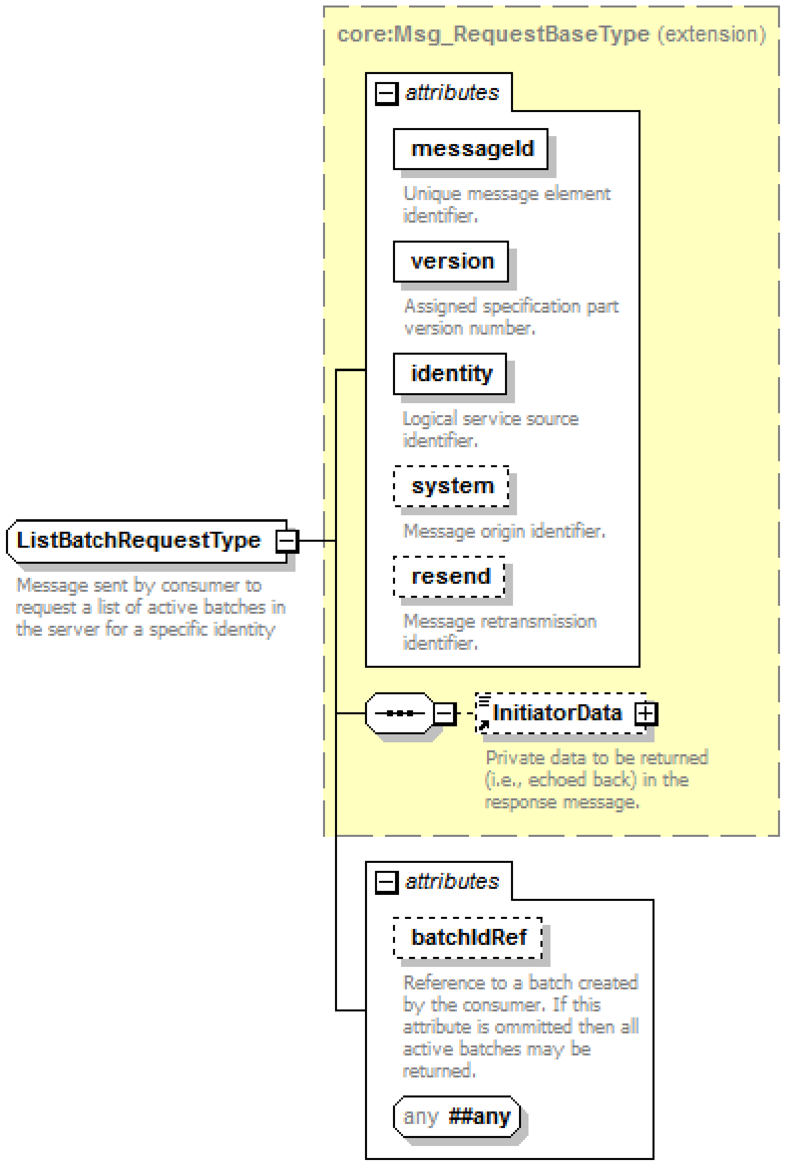

core:Msg_RequestBaseType (extension)
attributes
messageId
Unique message element identifier.
version
Assigned specification part version number.
identity
Logical service source identifier.
system
Message origin identifier.
resend
Message retransmission identifier.
ListBatchRequestType
Message sent by consumer to request a list of active batches in the server for a specific identity
InitiatorData
Private data to be returned (i.e., echoed back) in the response message.
attributes
batchIdRef
Reference to a batch created by the consumer. If this attribute is ommitted then all active batches may be returned.
any ##any

Figure 65. ListBatchRequestType XML Schema

The ListBatchRequest message type is derived from the core namespace base type core:Msg_RequestBaseType and defines the following attributes and elements in addition to those already defined in core:Msg_RequestBaseType.

@batchIdRef [Optional, batchIdRefAttrType] — The @batchIdRef attribute is a reference to the identifier of the batch to which the operation applies. If the @batchIdRef attribute is provided in the request, the information service ***shall*** return information about the referenced batch only. See F.8.2 for more information on batchIdRefAttrType.

@##any [Optional] — The @##any attribute enables a logical service incorporating the SCTE 130 Part 8 interface to optionally extend concrete messages derived from SCTE 130 Part 8 message types with attributes not specified by the schema.

F.7.7.2 ListBatchResponse Message Type

Upon receipt of a ListBatchRequest message a logical service ***shall*** respond with a ListBatchResponse message. The response contains information about one or more mutable operation batches according to the request.

The information service ***shall*** include in the list only batches that are active. An active batch is created but not started, or in progress (started). Cancelled or completed batches ***shall not*** be included in the list.

The XML schema definition for this message is illustrated in Figure 66.

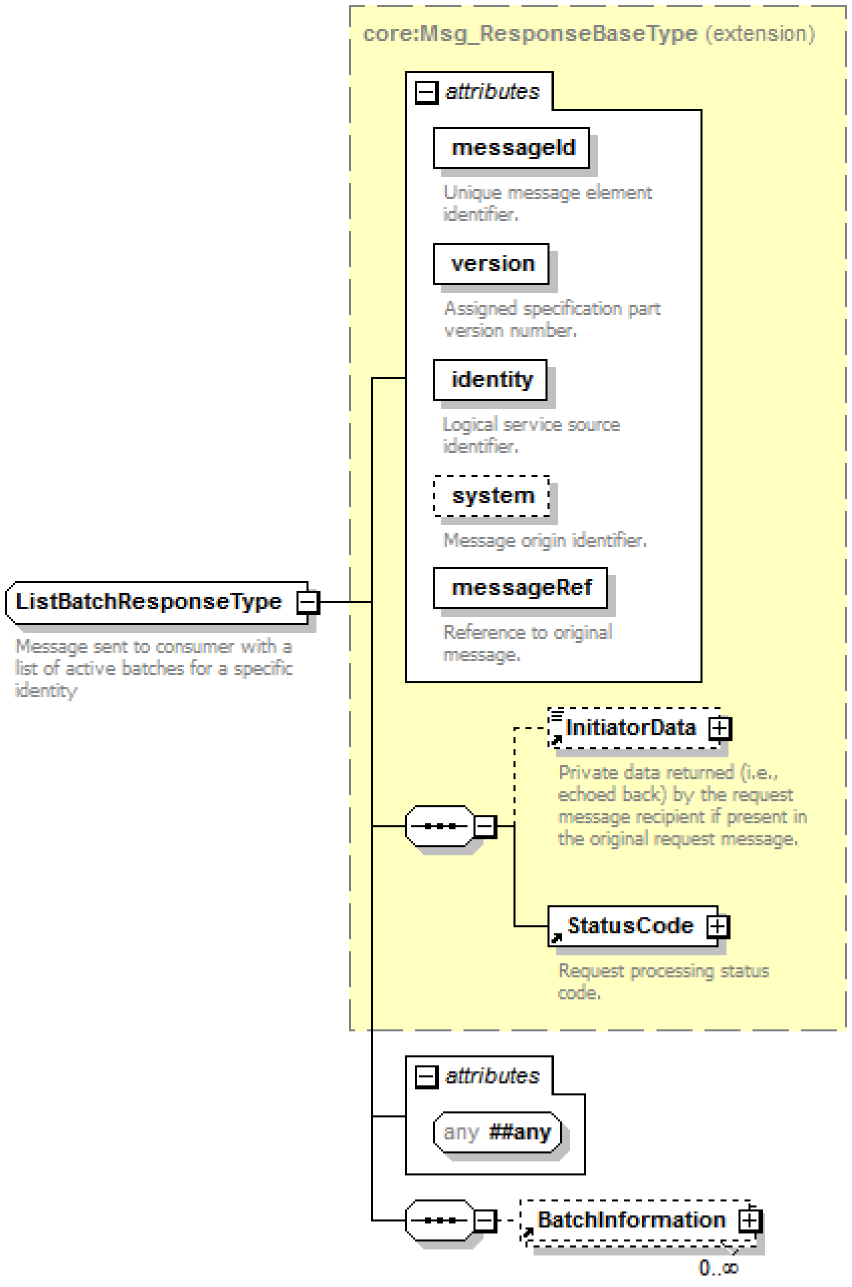

core:Msg_ResponseBaseType (extension)
attributes
messageId
Unique message element identifier.
version
Assigned specification part version number.
identity
Logical service source identifier.
system
Message origin identifier.
messageRef
Reference to original message.
ListBatchResponseType
Message sent to consumer with a list of active batches for a specific identity
InitiatorData
Private data returned (i.e., echoed back) by the request message recipient if present in the original request message.
StatusCode
Request processing status code.
attributes
any ##any
BatchInformation
0..∞

Figure 66. ListBatchResponseType XML Schema

The ListBatchResponse message type is derived from the core namespace base type core:Msg_ResponseBaseType and defines the following attributes and elements in addition to those that are already defined on the abstract base message core:Msg_ResponseBaseType.

@##any [Optional] — The @##any attribute enables a logical service incorporating the SCTE 130 Part 8 interface to optionally extend concrete messages derived from SCTE 130 Part 8 message types with attributes not specified by the schema.

BatchInformation [Optional] — The BatchInformation element contains information about a mutable operation batch. See F.9.3 for more information about BatchInformation element.

F.8 Mutable GIS Attribute Types and Common Types

The following sections define the Mutble GIS attribute types and common complex types used within the Mutable GIS extension.

F.8.1 batchIdAttrType Attribute Type

batchIdAttrType [core:nonEmptyStringType] — This attribute type, typically used as the @batchId attribute, identifies a batch execution group. The batch identifier is generated by a logical service typcially as a result of a BatchCreateRequest message type.

F.8.2 batchIdRefAttrType Attribute Type

batchIdRefAttrType [batchIdAttrType] — This attribute type, typically used as the @batchIdRef attribute, references a created batch using a previously returned batch identifier.

F.8.3 TransactionLevelTypeEnumeration Attribute Type

TransactionLevelTypeEnumeration [core:nonEmptyStringType] — This attribute type, typically referred to as the @transactionLevel attribute, specifies how the mutable operation transaction *should* be handled by the information service on updates. The @transactionLevel attribute ***shall not*** be empty.

The TransactionLevelTypeEnumeration values ***shall*** appear exactly as they specified in this Table 20**.**

Enumeration Value	Description
message	The “message” value indicates to a logical service that implements the Mutable GIS interface that in case of a failure, all the changes requested in the message will be rolled back and the mutable operation is considered failed.
entity	The “entity” value indicates to a logical service that implements the Mutable GIS interface that in case of a failure only the changes pertaining to the failed entity will be rolled back and the other successful entity changes will be persisted. The mutable operation will be considered partially successful and the error ***shall*** be reported to Consumer in the response message or via notification message.
…	User defined transaction isolation outside the scope of this specification. The string ***shall*** be prefixed with the text: “private:”.

Table 20. **TransactionLevelTypeEnumeration Values**

The default value for the @transactionLevel attribute is “message”.

F.8.4 updateIdAttrType Attribute Type

updateIdAttrType [core:nonEmptyStringType] — This attribute type, typically referred to as the @updateId attribute, assigns a unique identifier to an update operation. An information service uses the attribute’s value to reference the update operation when reporting back to the service Consumer. This attribute is required when an update request is made against a service data model.

F.8.5 updateIdRefAttrType Attribute Type

updateIdRefAttrType [updateIdAttrType] — This attribute type, typically referred to as the @updateIdRef attribute, is required in the UpdateError element provided in the response to the Consumer, when a mutable operation request made against a service data model failed to complete successfully. The @updateIdRef attribute references the incoming @updateId attribute for the failed update.

F.9 Mutable GIS Element Details

Mutable GIS elements *may* appear in the GIS or Mutable GIS top level message types or within any other elements including Mutable GIS elements themselves. Table 21 lists the Mutable GIS elements defined within the Mutable GIS XML namespace and described subsequently in detail.

Element	Description
Batch	Mutable operation batch description.
BatchCreateRequestAbstract	A copy of the batch create request.
BatchInformation	Mutable operation batch complete information.
BatchOperationRequestAbstract	A copy of the batch operation request.
CancelBatch	Cancel batch operation.
DataModelDescriptionQuery	Data model description inquiry.
DataModelDescriptionQueryResult	Data model description inquiry result set.
EvaluationArgument	A value included in the expression evaluation context.
ExecuteBatch	Request to start a previously created batch.
Expression	Description of an expression to be evaluated by the logical service.
MutableQualifierDescription	Completely describes a mutable qualifier.
MutableServiceDataModelProfile	Completely describes a mutable service data model.
PutDataModelQualifier	Add or update qualifier descriptions in the service data model.
PutQualifier	Add or update qualifier values in the service data model.
QualifierValue	The value of the qualifier.
QualifierValueDecrement	Decrement operation on the qualifier value.
QualifierValueExpression	An expression to be evaluated by the logical service to obtain the value of the qualifier.
QualifierValueIncrement	Increment operation on the qualifier value.
RemoveDataModelQualifier	Remove qualifier descriptions from the service data model.
RemoveQualifier	Remove qualifier values from the service data model.
UpdateError	Information about a mutable operation failure.
UpdateExpressionLanguage	Description of an expression language supported for mutable operations.

Table 21. **Mutable GIS Elementary Element Details**

F.9.1 Batch

The Batch element provides the unique identifier assigned to a batch operation and is typcially returned in response to a batch create operation.

The XML schema for the Batch element is shown in Figure 67.

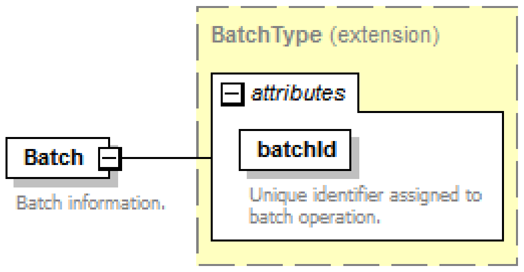


Figure 67. Batch XML Schema

@batchId [Required, batchIdAttrType] — A unique identifier, typically a UUID, assigned to identify a created batch. The @batchId identifier is supplied in subsequent message exchanges to reference the batch. See F.8.1for more information on batchIdAttrType.

F.9.2 BatchCreateRequestAbstract

The BatchCreateRequestAbstract is a copy of the BatchCreateRequest message sent by the Consumer to the information service. It is used to return batch information to a Consumer as part of the ListBatchResponse message.

The XML schema for the BatchCreateRequestAbstract element is illustrated in Figure 68.

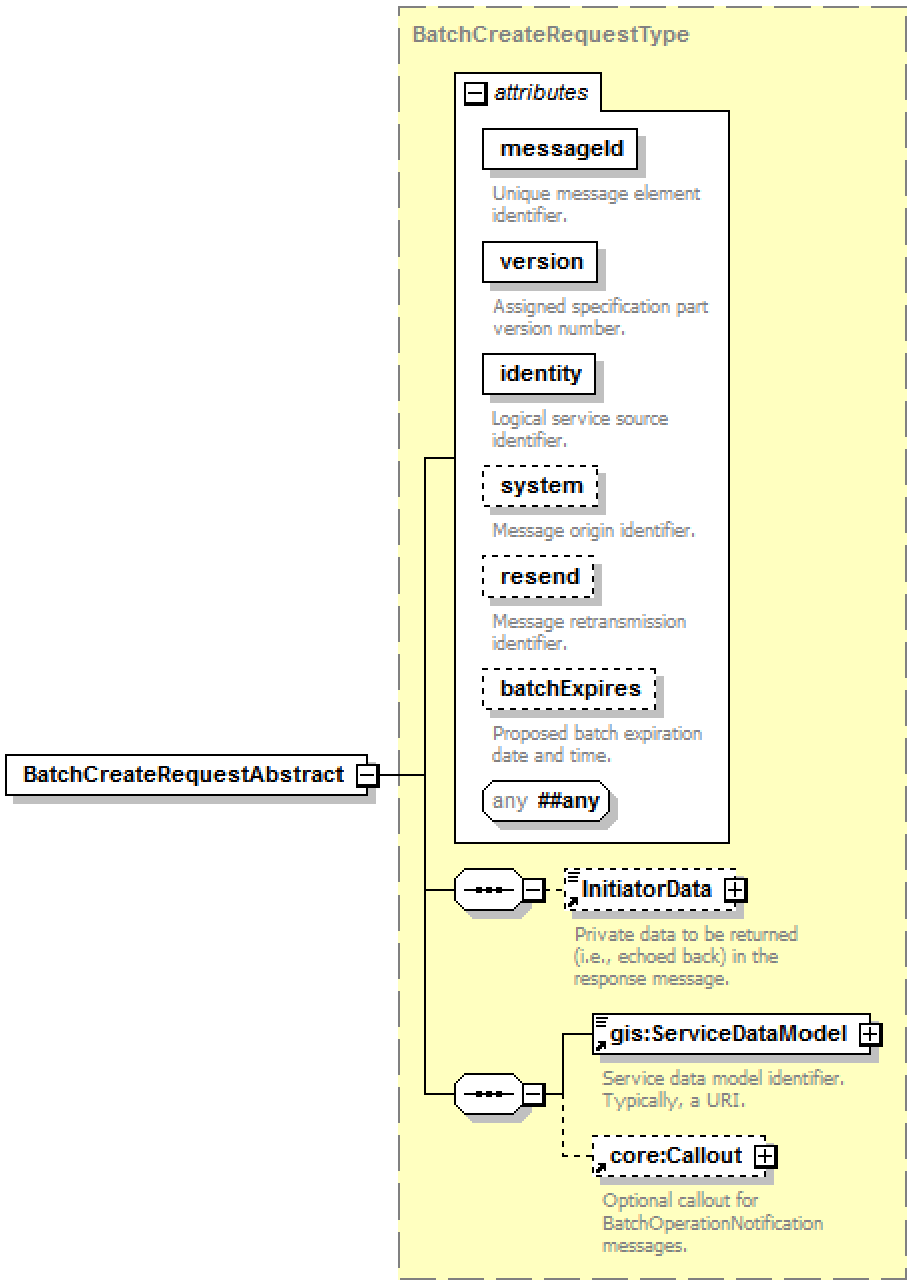

BatchCreateRequestType
attributes
messageId
Unique message element identifier.
version
Assigned specification part version number.
identity
Logical service source identifier.
system
Message origin identifier.
resend
Message retransmission identifier.
batchExpires
Proposed batch expiration date and time.
any ##any
BatchCreateRequestAbstract
InitiatorData
Private data to be returned (i.e., echoed back) in the response message.
gis:ServiceDataModel
Service data model identifier. Typically, a URI.
core:Callout
Optional callout for BatchOperationNotification messages.

Figure 68. BatchCreateRequestAbstract XML Schema

The BatchCreateRequestAbstract element is of type BatchCreateRequestType. See F.7.2.1 for more information on BatchCreateRequestType.

F.9.3 BatchInformation

The BatchInformation element is returned to a Consumer as a result of a ListBatchRequest message and contains information about a batch that has been created in the information service.

The XML schema for the BatchInformation is illustrated in Figure 69.

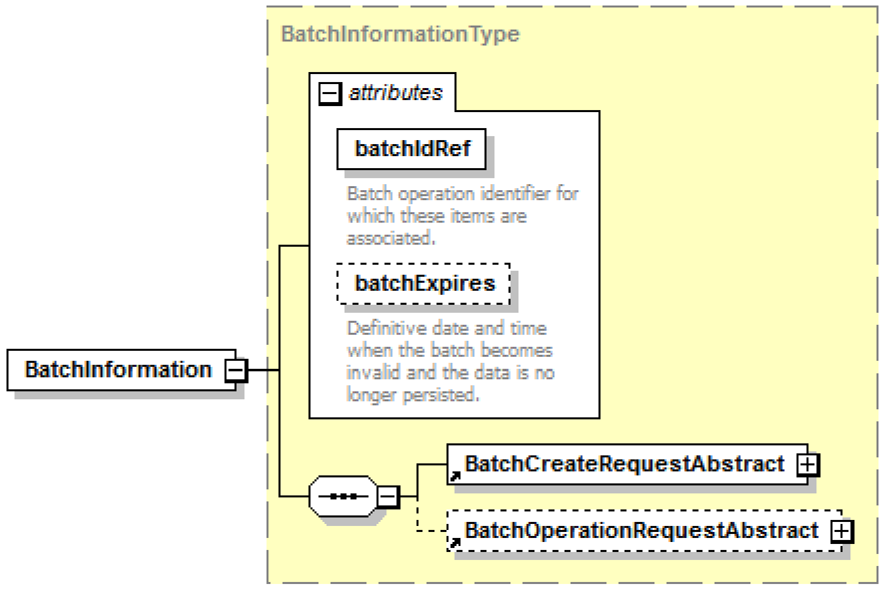

BatchInformationType
attributes
batchIdRef
Batch operation identifier for which these items are associated.
batchExpires
Definitive date and time when the batch becomes invalid and the data is no longer persisted.
BatchInformation
BatchCreateRequestAbstract
BatchOperationRequestAbstract

Figure 69. BatchInformation XML Schema

@batchIdRef [Required, batchIdRefAttrType] — The @batchIdRef attribute is a reference to the identifier of the batch for which the information is provided. See F.8.2 for more information on batchIdRefAttrType.

@batchExpires [Optional, core:dateTimeTimezoneType] —The @batchExpires attribute is used by the information service to inform the Consumer when the batch will be considered expired.

BatchCreateRequestAbstract [Required] — The BatchCreateRequestAbstract is a copy of the BatchCreateRequest message sent by Consumer to the information service. See F.9.2 for more information on BatchCreateRequestAbstract.

BatchOperationRequestAbstract [Optional] — The BatchOperationRequestAbstract is a copy of the BatchOperationRequest message sent by Consumer to the information service. See F.9.4 for more information on BatchOperationRequestAbstract.

F.9.4 BatchOperationRequestAbstract

The BatchOperationRequestAbstract is a copy of the BatchOperationRequest message sent by Consumer to the information service. It is used to return batch information to a Consumer as part of the ListBatchResponse message.

The XML schema for the BatchOperationRequestAbstract element is illustrated in Figure 70.

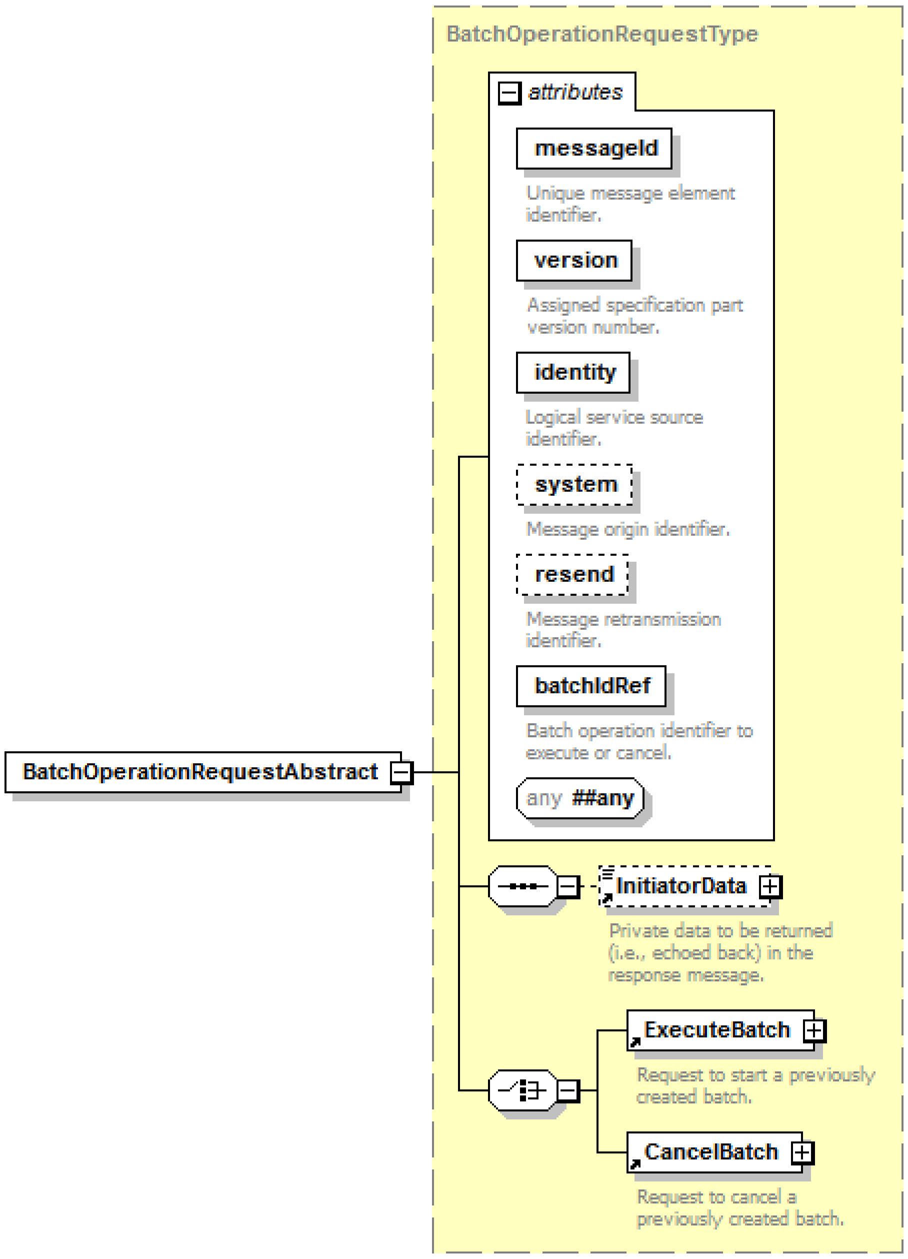

BatchOperationRequestType
attributes
messageId
Unique message element identifier.
version
Assigned specification part version number.
identity
Logical service source identifier.
system
Message origin identifier.
resend
Message retransmission identifier.
batchIdRef
Batch operation identifier to execute or cancel.
any ##any
BatchOperationRequestAbstract
InitiatorData
Private data to be returned (i.e., echoed back) in the response message.
ExecuteBatch
Request to start a previously created batch.
CancelBatch
Request to cancel a previously created batch.

Figure 70. BatchOperationRequestAbstract XML Schema

The BatchOperationRequestAbstract element is of type BatchOperationRequestType. See F.7.4.1 for more information on BatchOperationRequestType.

F.9.5 CancelBatch

The CancelBatch element is used by a Consumer in a BatchOperationRequest message to cancel a previously created batch mutable operation. Batches that have been started cannot be cancelled and such request ***shall*** result in an error message from the information service.

The XML schema for the CancelBatch element is illustrated in Figure 71.

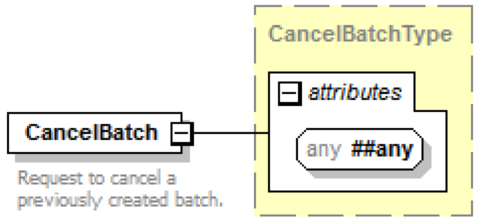


Figure 71. CancelBatch XML Schema

@##any [Optional] — The @##any attribute enables a logical service incorporating the SCTE 130 Part 8 interface to optionally extend concrete messages derived from SCTE 130 Part 8 message types with attributes not specified by the schema.

F.9.6 DataModelDescriptionQuery

The DataModelDescriptionQuery element contains elements specifying the semantics for data model description change inquiry. The DataModelDescriptionQuery element extends the QueryAbstract element, allowing it to be substituted for the QueryAbstract element.

The DataModelDescriptionQuery element is typically used by a Consumer to register for notifications on data model description changes. See [SCTE130-8] for more information on the notification messages. Upon successful registration for service data model description changes, the Consumer ***shall*** be notified whenever the data model description is modified. The notification ***shall not*** contain the details of the service data model description change as that information can be retrieved by the Consumer via the ListQualifiers message exchange. See F.6.2 for more information on ListQualifiers message exchange.

The DataModelDescriptionQuery element *may* also be included by a Consumer in a message of type gis:QueryRequestType, in which case the Mutable GIS ***shall*** always return a message indicating no changes.

The XML schema for the DataModelDescriptionQuery element is illustrated in Figure 72.

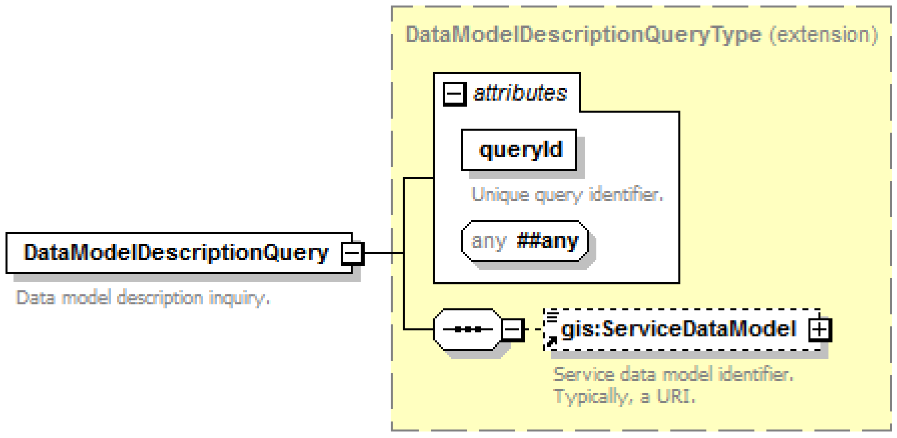

DataModelDescriptionQueryType (extension)
attributes
queryId
Unique query identifier.
any ##any
DataModelDescriptionQuery
Data model description inquiry.
gis:ServiceDataModel
Service data model identifier. Typically, a URI.

Figure 72. DataModelDescriptionQuery XML Schema

@queryId [Required, queryIdAttrType] — The @queryId attribute is a unique identifier for the inquiry. This identifier ***shall*** be unique within the scope of the enclosing parent element's @identity attribute and ***shall not*** be empty. See [SCTE130-8] for more information on queryIdAttrType.

@##any [Optional] — The @##any attribute enables a logical service incorporating the SCTE 130 Part 8 interface to optionally extend concrete messages derived from SCTE 130 Part 8 message types with attributes not specified by the schema.

gis:ServiceDataModel [Optional] — The gis:ServiceDataModel element contains a service data model reference that ***shall*** be used to satisfy the update. If the ServiceDataModel element is not present, the default service data model ***shall*** be selected. For additional information on the gis:ServiceDataModel element see this document, SCTE 130 Part 8.

F.9.7 DataModelDescriptionQueryResult

The DataModelDescriptionQueryResult element is used by the Mutable GIS to inform a Consumer that a data model description has changed. This element is provided as part of the gis:QueryResult element. See [SCTE130-8] for more information on gis:QueryResult.

The XML schema for the DataModelDescriptionQueryResult element is illustrated in Figure 73.

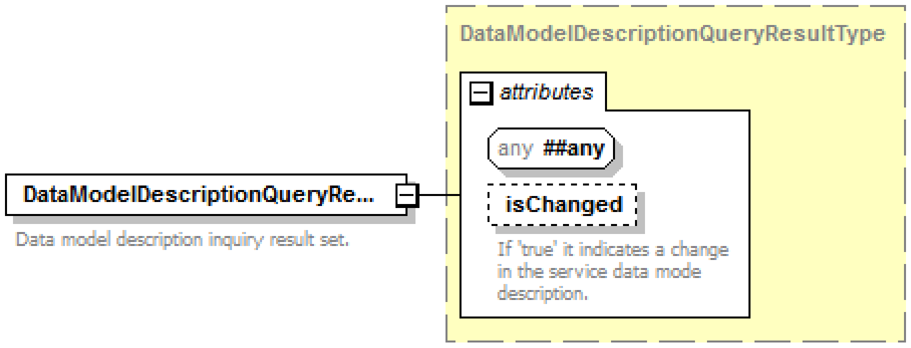

DataModelDescriptionQueryResultType
attributes
any ##any
isChanged
If 'true' it indicates a change in the service data mode description.
DataModelDescriptionQueryRe...
Data model description inquiry result set.

Figure 73. DataModelDescriptionQueryResult XML Schema

@isChanged [Optional, xsd:boolean] — The @isChanged attribute indicates whether the service data model description has changed. If the attribute is missing it is equivalent to having its value set to 'false' and indicates no changes in the service data model.

F.9.8 ExecuteBatch

The ExecuteBatch element is included by a Consumer in a BatchOperationRequest message to start execution of a previously created mutable operation batch.

The XML schema for the ExecuteBatch element is illustrated in Figure 74.

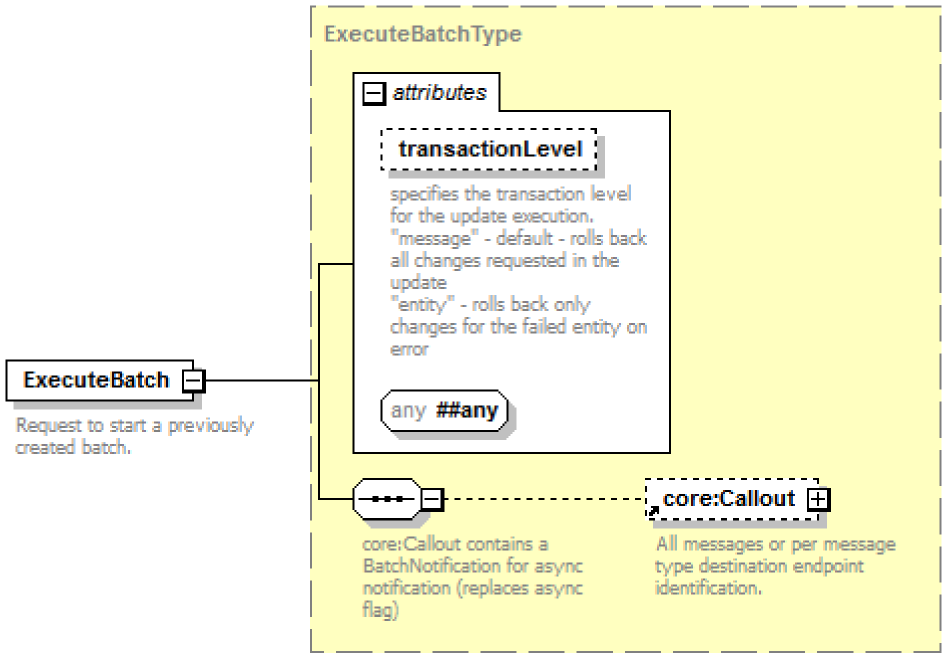

ExecuteBatchType
attributes
transactionLevel
specifies the transaction level for the update execution. "message" - default - rolls back all changes requested in the update "entity" - rolls back only changes for the failed entity on error
any ##any
ExecuteBatch
Request to start a previously created batch.
core:Callout contains a BatchNotification for async notification (replaces async flag)
core:Callout
All messages or per message type destination endpoint identification.

Figure 74. ExecuteBatch XML Schema

@##any [Optional] — The @##any attribute enables a logical service incorporating the SCTE 130 Part 8 interface to optionally extend concrete messages derived from SCTE 130 Part 8 message types with attributes not specified by the schema.

@transactionLevel [Optional, TransactionLevelTypeEnumeration] — the @transactionLevel attribute allows a Consumer to control how the information service will handle the mutable transaction. See F.8.3 for more information on the possible values of this attribute and how they affect the behavior of the update process.

core:Callout [Optional] — The core:Callout element provides callback message and address information to the logical service. If a Consumer provides the core:Callout element in the ExecuteBatch element, the operation ***shall*** be performed asynchronously. Thus, the information service ***shall*** respond to the Consumer that the message has been received, and upon completion of the requested mutable operation, the information service ***shall*** notify the Consumer at the address specified in the core:Callout element. See [SCTE130-2] for a complete description of the core:Callout element.

F.9.9 Expression

The Expression element is used to describe an expression to be evaluated by the information service. The content of the Expression element is usually a string that can be evaluated using the language specified by the @language attribute. Consumers can use the ListFeaturesRequest message to discover the expression languages supported by the logical service and then reference a supported language in the Expression element.

The XML schema for the Expression element is illustrated in Figure 75.

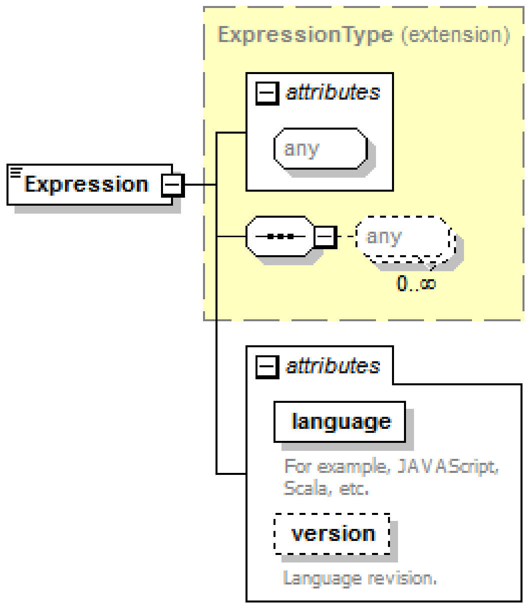


Figure 75. Expression XML Schema

@language [Required, core:nonEmptyStringType] — The @language attribute identifies the language used to interpret and evaluate the expression.

@version [Optional, core:nonEmptyStringType] — The @version attribute identifies a specific version of the language used by the expression.

F.9.10 EvaluationArgument

The EvaluationArgument element is used to pass input arguments to the expression evalution context.

The XML schema for the EvaluationArgument element is illustrated in Figure 76.

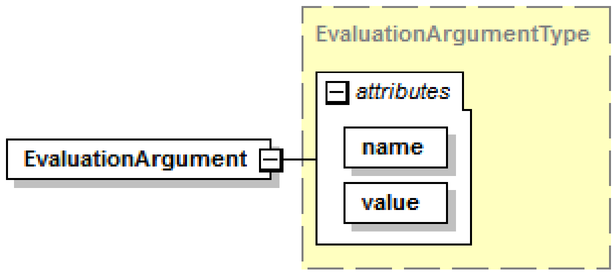


Figure 76. EvaluationArgument XML Schema

@name [Required, core:nonEmptyStringType] — The @name attribute identifies an argument in the expression context and ***shall not*** be empty.

@value [Required, core:nonEmptyStringType] — The @value attribute contains the value of an argument in the expression context and ***shall not*** be empty.

F.9.11 MutableQualifierDescription

The MutableQualifierDescription element defines a mutable qualifier in a service data model.

The XML schema for the MutableQualifierDescription element is illustrated in Figure 77.

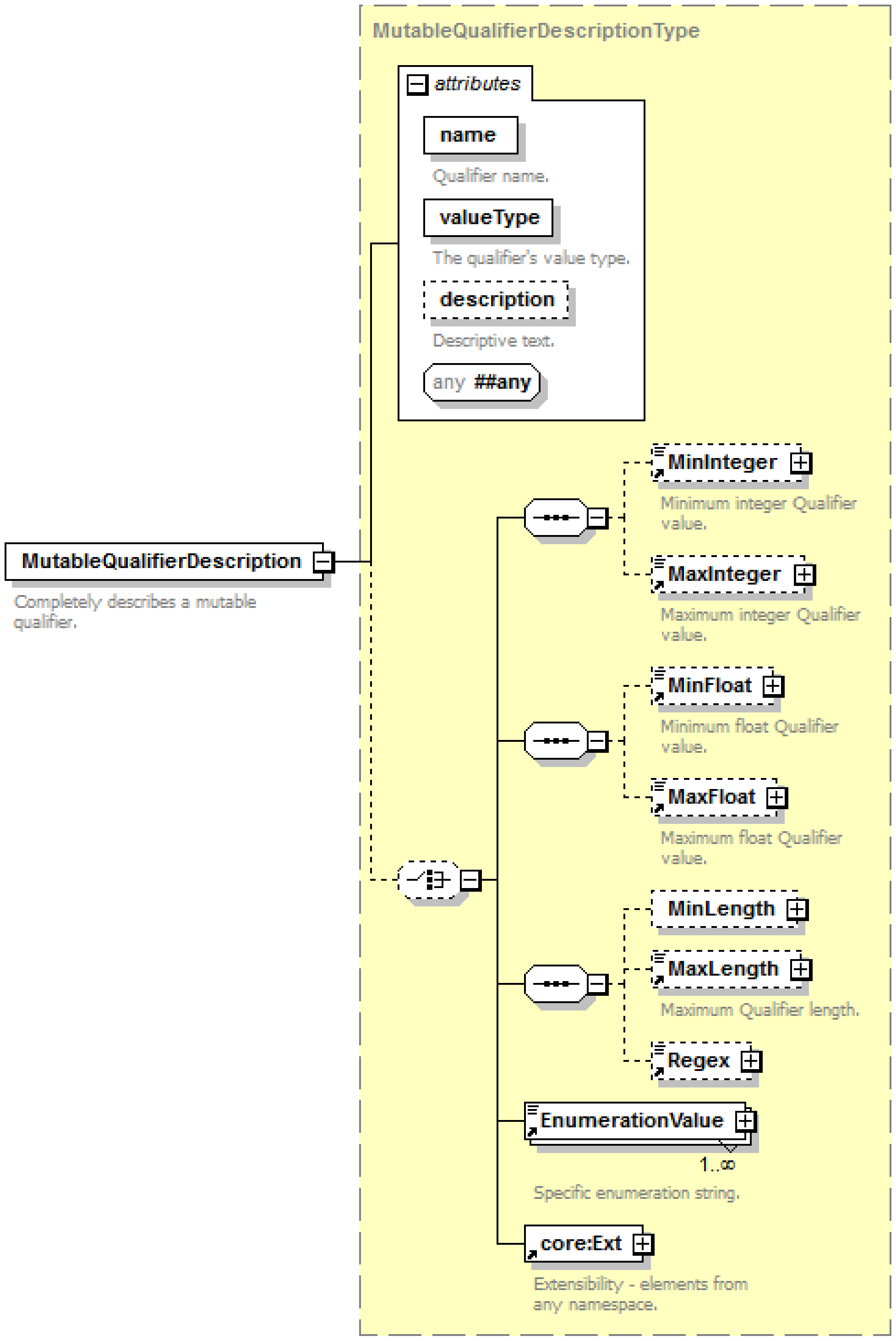

MutableQualifierDescriptionType
attributes
name
Qualifier name.
valueType
The qualifier's value type.
description
Descriptive text.
any ##any
MinInteger
Minimum integer Qualifier value.
MaxInteger
Maximum integer Qualifier value.
MinFloat
Minimum float Qualifier value.
MaxFloat
Maximum float Qualifier value.
MinLength
MaxLength
Maximum Qualifier length.
Regex
EnumerationValue
1..∞
Specific enumeration string.
core:Ext
Extensibility - elements from any namespace.
MutableQualifierDescription
Completely describes a mutable qualifier.

Figure 77. MutableQualifierDescription XML Schema

The MutableQualifierDescription element is defined by MutableQualifierDescription type that is an extension of the gis:QualifierDescriptionType defined in SCTE 130 Part 8. There are no new elements or attributes added.

F.9.12 MutableServiceDataModelProfile

The MutableServiceDataModelProfile element completely describes a mutable service data model.

The XML schema for the MutableServiceDataModelProfile element is illustrated in Figure 78.

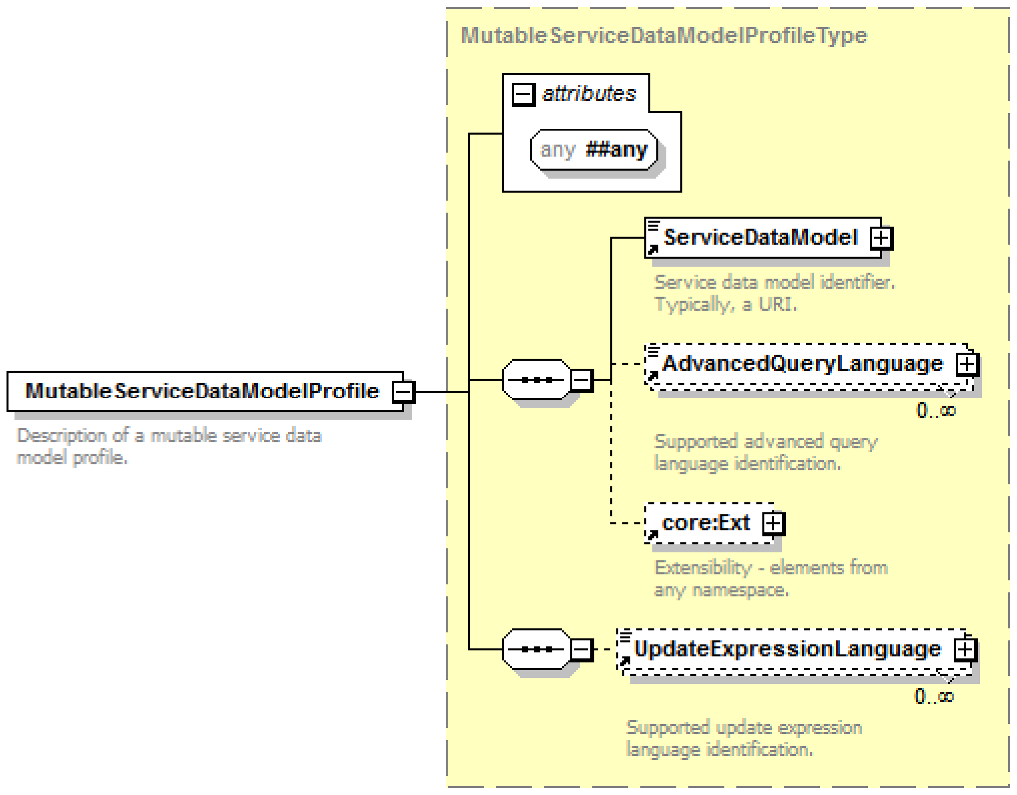

MutableServiceDataModelProfileType
attributes
any ##any
ServiceDataModel
Service data model identifier. Typically, a URI.
AdvancedQueryLanguage
0..∞
Supported advanced query language identification.
core:Ext
Extensibility - elements from any namespace.
UpdateExpressionLanguage
0..∞
Supported update expression language identification.
MutableServiceDataModelProfile
Description of a mutable service data model profile.

Figure 78. MutableServiceDataModelProfile XML Schema

The MutableServiceDataModel element is derived from the GIS namespace base gis:ServiceDataModelProfile element and defines the following attributes and elements in addition to those already defined in gis:ServiceDataModelProfile:

UpdateExpressionLanguage [Optional] — The UpdateExpressionLanguage element describes an expression language supported by the logical service for mutable operations. The expression language ***shall*** be referenced in the QualifierValueExpression element from a mutable operation. See F.9.17 for more information on QualifierValueExpression, and F.9.22 for more information on UpdateExpressionLanguage.

See this document, SCTE 130 Part 8 for more information on gis:ServiceDataModelProfile.

F.9.13 PutDataModelQualifier

The PutDataModelQualifier is used by a Consumer to add or update a qualifier definition in a service data model as part of a mutable operation.

The XML schema for the PutDataModelQualifier element is illustrated in Figure 79.

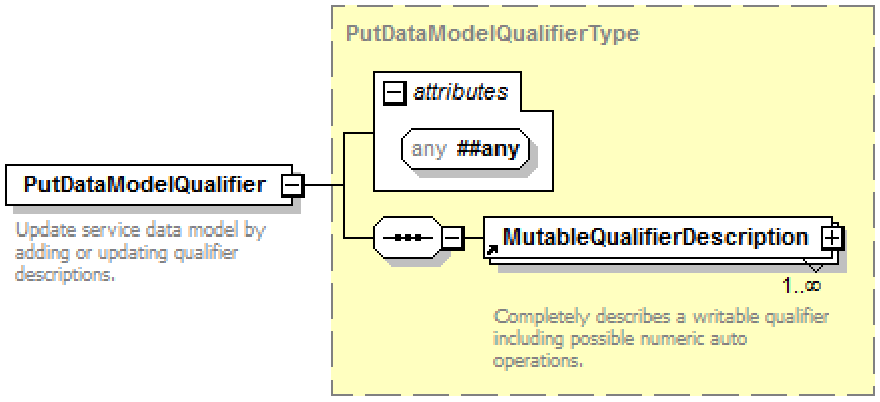

PutDataModelQualifierType
attributes
any ##any
PutDataModelQualifier
Update service data model by adding or updating qualifier descriptions.
MutableQualifierDescription
1..∞
Completely describes a writable qualifier including possible numeric auto operations.

Figure 79. PutDataModelQualifier XML Schema

@##any [Optional] — The @##any attribute enables a logical service incorporating the SCTE 130 Part 8 interface to optionally extend concrete messages derived from SCTE 130 Part 8 message types with attributes not specified by the schema.

MutableQualifierDescription [Required] — The MutableQualifierDescription element is used by the Consumer to add or update in the service data model the definition of a qualifier that is mutable. See F.9.11 for more information on MutableQualifierDescription element.

F.9.14 PutQualifier

The PutQualifier element is used by a Consumer to add or update a qualifier value in a service data model as part of a mutable operation. The value of a qualifier can be set explicitly with QualifierValue element or it can be modified using operations like QualifierValueDecrement, QualifierValueIncrement and QualifierValueExpression. Additional operations *may* be added by a logical service implementing the mutable API.

The XML schema for the PutQualifier element is illustrated in Figure 80.

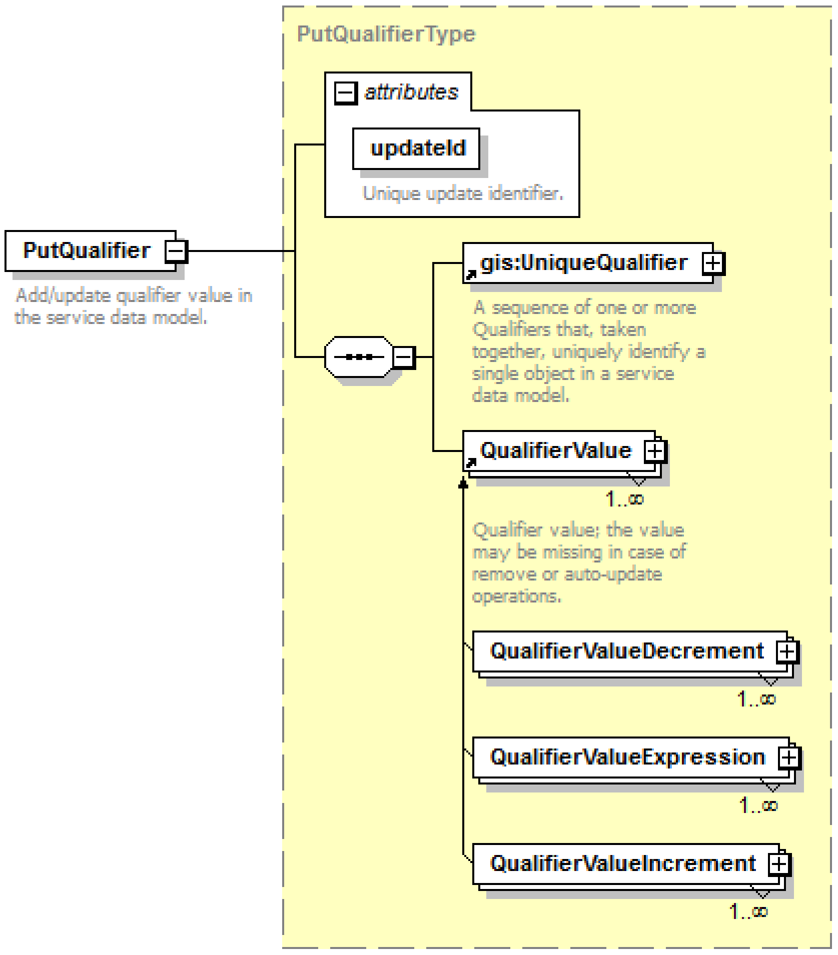

PutQualifierType
attributes
updateId
Unique update identifier.
PutQualifier
Add/update qualifier value in the service data model.
gis:UniqueQualifier
A sequence of one or more Qualifiers that, taken together, uniquely identify a single object in a service data model.
QualifierValue
1..∞
Qualifier value; the value may be missing in case of remove or auto-update operations.
QualifierValueDecrement
1..∞
QualifierValueExpression
1..∞
QualifierValueIncrement
1..∞

Figure 80. PutQualifier XML Schema

@updateId [Required, updateIdAttrType] — The @updateId attribute is used by a Consumer to identify a transaction unit inside a mutable operation. A mutable operation *may* modify multiple entities. Each modification is identified with a unique @updateId. Upon operation execution, the information service ***shall*** send a message to the Consumer with all the errors that occurred during execution. These errors ***shall*** reference the @updateId to identify the failed operation. See F.8.4 for more information on updateIdAttrType.

gis:UniqueQualifier [Required] — The gis:UniqueQualifier element is used by a Consumer to identify the entity in the service data model that is being modified. For more information on gis:UniqueQualifier element see this document, SCTE 130 Part 8.

QualifierValue [Required] — The QualifierValue element is used to set the value of a qualifier in service data model. See F.9.15 for more information on QualifierValue element.

QualifierValueDecrement [Required] — The QualifierValueDecrement is used by a Consumer to decrement the value of a numeric qualifier. This element can be used in a mutable operation only if the qualifier is described as being an integer. For more details on MutableQualifierDescription element see F.9.11. See F.9.16 for more information on QualfierValueDecrement element.

QualifierValueExpression [Required] — The QualifierValueExpression is used by a Consumer to set the value of a qualifier based on the evaluation result of an expression. See F.9.17 for more information on QualifierValueExpression element.

QualifierValueIncrement [Required] — The QualifierValueIncrement is used by a Consumer to increment the value of a numeric qualifier. This element can be used in a mutable operation only if the qualifier is described as being an integer. For more details on MutableQualifierDescription element see F.9.11. See F.9.18 for more information on QualifierValueIncrement element.

F.9.15 QualifierValue

The QualifierValue element is used in a mutable operation to modify the value of a qualifier in a service data model. This element is used add, update or remove a qualifier value. In case of remove operation, the @value attribute of the element *should not* be provided as it will be ignored by the information service when executing the operation.

The XML schema for the QualifierValue element is illustrated in Figure 81.

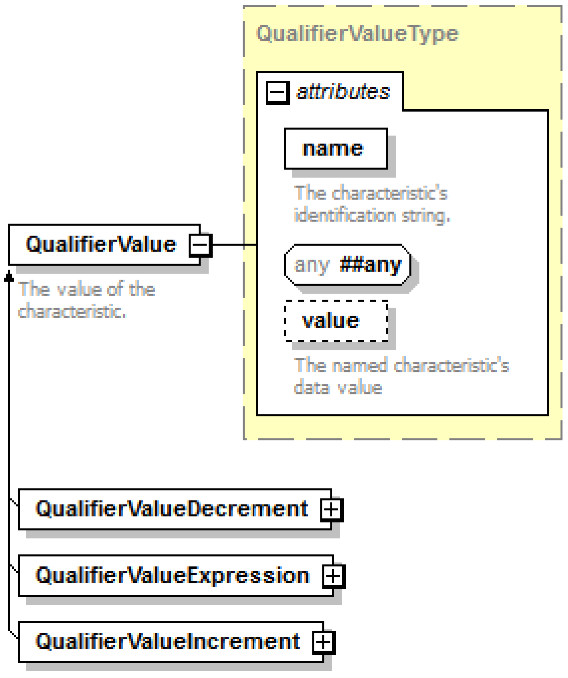


Figure 81. QualifierValue XML Schema

@name [Required, gis:qualifierNameAttrType] — The @name attribute identifies a qualifier in a service data model. For more information on gis:qualifierNameAttrType see this document, SCTE 130 Part 8.

@##any [Optional] — The @##any attribute enables a logical service incorporating the SCTE 130 Part 8 interface to optionally extend concrete messages derived from SCTE 130 Part 8 message types with attributes not specified by the schema.

@value [Optional, core:NonEmptyStringType] — The @value attribute is optional. If present it contains the value of the qualifier and it ***shall not*** be empty.

F.9.16 QualifierValueDecrement

The QualifierValueDecrement element is used by a Consumer in a mutable operation to decrement the value of an integer type qualifier.

The XML schema for the QualifierValueDecrement element is illustrated in Figure 82.

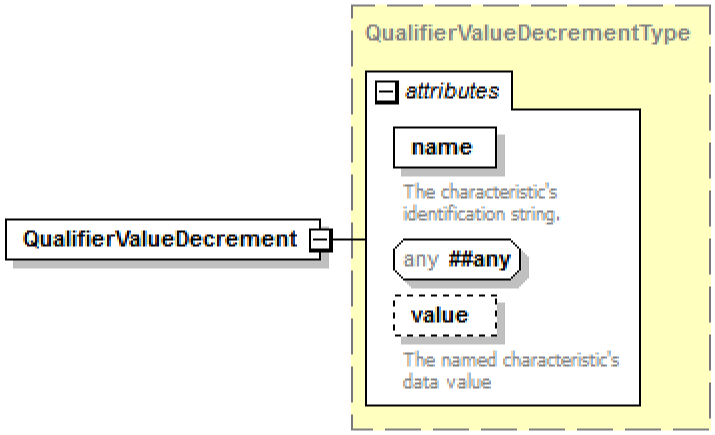

QualifierValueDecrementType
attributes
name
The characteristic's identification string.
any ##any
value
The named characteristic's data value
QualifierValueDecrement

Figure 82. QualifierValueDecrement XML Schema

@name [Required, gis:qualifierNameAttrType] — The @name attribute identifies a qualifier in a service data model. For more information on gis:qualifierNameAttrType see this document, SCTE 130 Part 8.

@##any [Optional] — The @##any attribute enables a logical service incorporating the SCTE 130 Part 8 interface to optionally extend concrete messages derived from SCTE 130 Part 8 message types with attributes not specified by the schema.

@value [Optional, core:NonEmptyStringType] — The @value attribute is optional. If present, it contains an integer representing the value to be subtracted from the existing qualifier value. If not present it ***shall*** default to “1”.

F.9.17 QualifierValueExpression

The QualifierValueExpression element is used by a Consumer in a mutable operation to set the value of qualifier in a service data model based on the evaluation result of an expression.

The XML schema for the QualifierValueExpression element is illustrated in Figure 83.

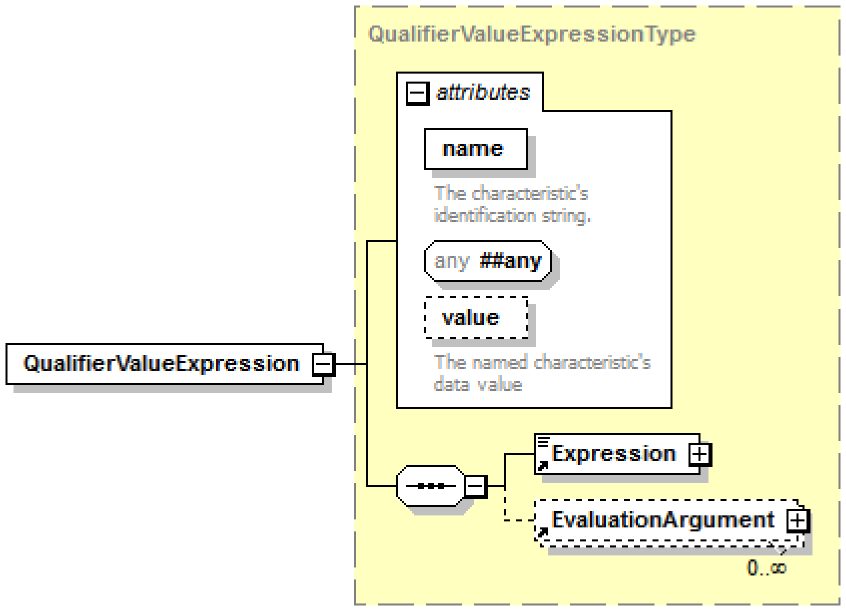

QualifierValueExpressionType
attributes
name
The characteristic's identification string.
any ##any
value
The named characteristic's data value
QualifierValueExpression
Expression
EvaluationArgument
0..∞

Figure 83. QualifierValueExpression XML Schema

@name [Required, gis:qualifierNameAttrType] — The @name attribute identifies a qualifier in a service data model. For more information on gis:qualifierNameAttrType see [SCTE130-8].

@##any [Optional] — The @##any attribute enables a logical service incorporating the SCTE 130 Part 8 interface to optionally extend concrete messages derived from SCTE 130 Part 8 message types with attributes not specified by the schema.

@value [Optional, core:NonEmptyStringType] — The @value attribute is optional and *should not* be present in the element. If present, the information service ***shall*** ignore it.

Expression [Required] — The Expression element contains the expression to be evaluated by the information service. See F.9.9 for more information on the Expression element.

EvaluationArgument [Optional] — The EvaluationArgument element is a name value pair used to pass input values to the expression evaluation context. See F.9.10 for more information on the EvaluationArgument element.

F.9.18 QualifierValueIncrement

The QualifierValueIncrement element is used by a Consumer in a mutable operation to increment the value of an integer type qualifier.

The XML schema for the QualifierValueIncrement element is illustrated in Figure 84.

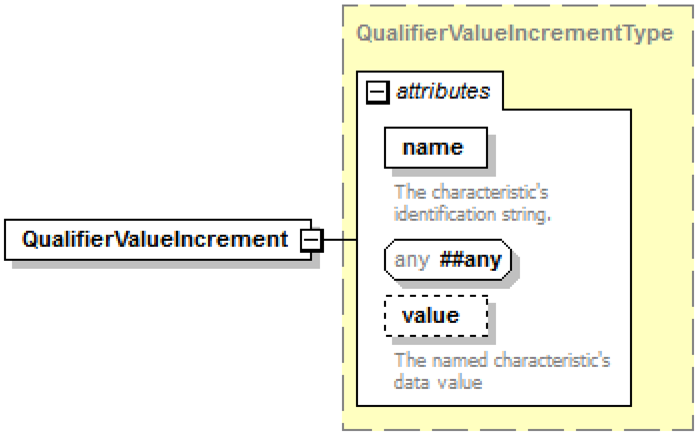

QualifierValueIncrementType
attributes
name
The characteristic's identification string.
any ##any
value
The named characteristic's data value
QualifierValueIncrement

Figure 84. QualifierValueIncrement XML Schema

@name [Required, gis:qualifierNameAttrType] — The @name attribute identifies a qualifier in a service data model. For more information on gis:qualifierNameAttrType see [SCTE130-8].

@##any [Optional] — The @##any attribute enables a logical service incorporating the SCTE 130 Part 8 interface to optionally extend concrete messages derived from SCTE 130 Part 8 message types with attributes not specified by the schema.

@value [Optional, core:NonEmptyStringType] — The @value attribute is optional. If present, it contains an integer which represents the value to be added to the existing qualifier value. If not present, it ***shall*** default to "1".

F.9.19 RemoveDataModelQualifier

The RemoveDataModelQualifier element is used by a Consumer to remove a qualifier definition from a service data model as part of a mutable operation.

The XML schema for the RemoveDataModelQualifier element is illustrated in Figure 85.

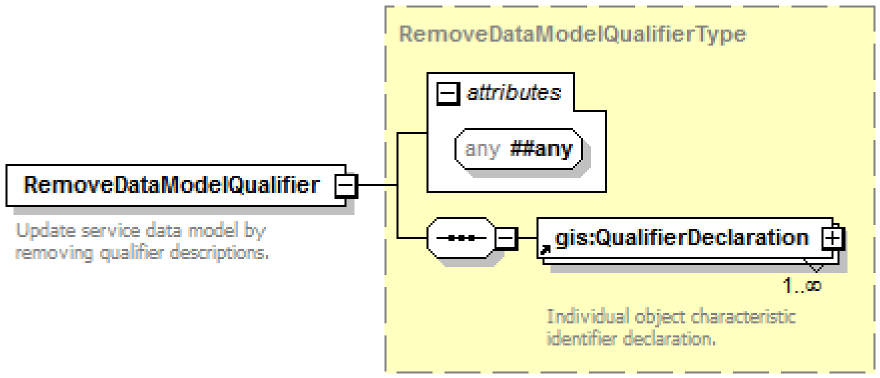

RemoveDataModelQualifierType
attributes
any ##any
RemoveDataModelQualifier
Update service data model by removing qualifier descriptions.
gis:QualifierDeclaration
1..∞
Individual object characteristic identifier declaration.

Figure 85. RemoveDataModelQualifier XML Schema

@##any [Optional] — The @##any attribute enables a logical service incorporating the SCTE 130 Part 8 interface to optionally extend concrete messages derived from SCTE 130 Part 8 message types with attributes not specified by the schema.

gis:QualifierDeclaration [Required] — The QualifierDeclaration element is used to indicate the qualifier definitions to be removed by the information service when executing the mutable operation. For more information on QualifierDeclaration see this document, SCTE 130 Part 8.

F.9.20 RemoveQualifier

The RemoveQualifier element is used by a Consumer to remove a qualifier value from a service data model as part of a mutable operation.

The XML schema for the RemoveQualifier element is illustrated in Figure 86.

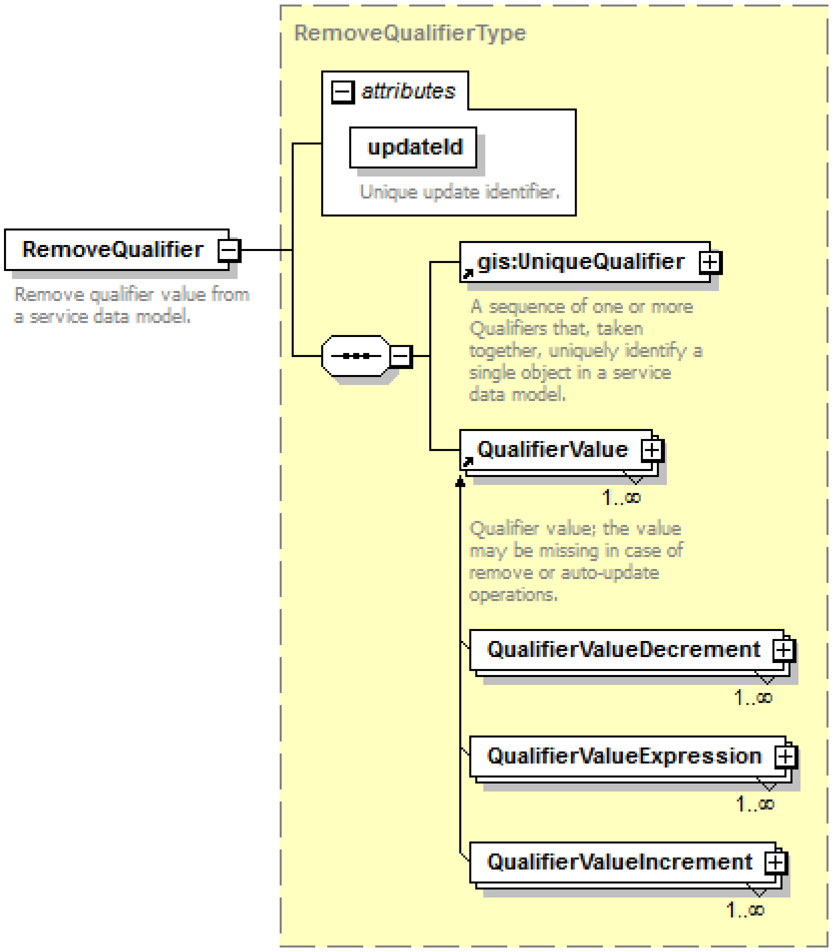

RemoveQualifierType
attributes
updateId
Unique update identifier.
RemoveQualifier
Remove qualifier value from a service data model.
gis:UniqueQualifier
A sequence of one or more Qualifiers that, taken together, uniquely identify a single object in a service data model.
QualifierValue
1..∞
Qualifier value; the value may be missing in case of remove or auto-update operations.
QualifierValueDecrement
1..∞
QualifierValueExpression
1..∞
QualifierValueIncrement
1..∞

Figure 86. RemoveQualifier XML Schema

@updateId [Required, updateIdAttrType] — The @updateId attribute is used by a Consumer to identify a transaction unit inside a mutable operation. A mutable operation *may* modify multiple entities. Each modification is identified with a unique @updateId. Upon operation execution, the information service ***shall*** send a message to the Consumer with all the errors that occurred during execution. These errors ***shall*** reference the @updateId to identify the failed operation. See F.8.4 for more information on updateIdAttrType.

gis:UniqueQualifier [Required] — The gis:UniqueQualifier element is used by a Consumer to identify the entity in the service data model that is being modified. For more information on gis:UniqueQualifier element see this document, SCTE 130 Part 8.

QualifierValue [Required] — The QualifierValue element is used to identify the qualifier value to be removed from the service data model. When used as part of the RemoveQualifier element, this element *should* only contain the @name attribute representing the qualifier name. The @value attribute, if included in the request ***shall*** be ignored. See F.9.15 for more information on QualifierValue element.

The QualifierValueIncrement, QualifierValueDecrement and QualifierValueExpression elements are not valid within the RemoveQualifier element. The information service ***shall*** report an error on the operation if any of these elements are present in the request.

F.9.21 UpdateError

The UpdateError element is added in the response message if the information service failed to complete the requested mutable operation. Using the UpdateError element, an implementation of the information service is able to provide the Consumer more information on a specific request failure. Each UpdateError element in the response references a mutable operation in the request via the required @updateIdRef attribute. See F.8.5 for more information on @updateIdRef attribute.

The XML schema for the UpdateError element is shown in Figure 87.

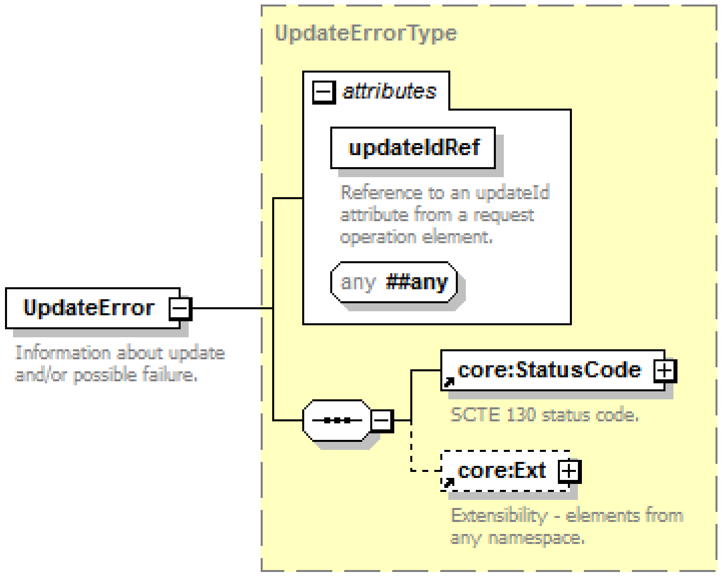

UpdateErrorType
attributes
updateIdRef
Reference to an updateId attribute from a request operation element.
any ##any
UpdateError
Information about update and/or possible failure.
core:StatusCode
SCTE 130 status code.
core:Ext
Extensibility - elements from any namespace.

Figure 87. UpdateError Element XML Schema

@updateIdRef [Required, updateIdRefAttrType] — The @updateIdRef attribute is a reference to the @updateId attribute from the PutQualifier/RemoveQualifier elements received in the mutable operation request. See F.8.5 for more information on updateIdRefAttrType. See F.9.14 and F.9.20 for more information on PutQualifier and RemoveQualifier respectively.

@##any [Optional] — The @##any attribute enables a logical service incorporating the SCTE 130 Part 8 interface to optionally extend concrete messages derived from SCTE 130 Part 8 message types with attributes not specified by the schema.

core:StatusCode [Required] — The core:StatusCode element contains the operation processing status. The @class attribute of the core:StatusCode element provides the status of the operation. See F.4 for more information on core:StatusCode.

core:Ext [Optional] — Any additional elements from any namespace. For additional information on the core:Ext element see [SCTE130-2].

F.9.22 UpdateExpressionLanguage

The UpdateExpressionLanguage describes an expression language supported by the logical service for mutable operations.

The XML schema for the UpdateExpressionLanguage element is shown in Figure 88.

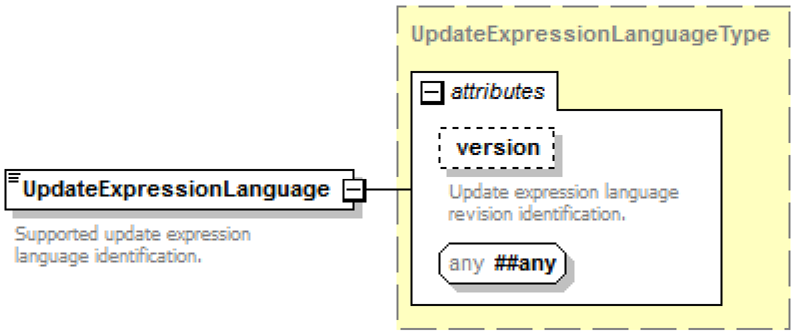

UpdateExpressionLanguageType
attributes
version
Update expression language revision identification.
any ##any
UpdateExpressionLanguage
Supported update expression language identification.

Figure 88. UpdateExpressionLanguage Element XML Schema

@version [Optional] — The @version attribute specifies which version of the language is supported. It is required when describing a language that has multiple versions. (for instance, JavaScript language version could be “1.5”, “1.6”, etc.)

@##any [Optional] — The @##any attribute enables a logical service incorporating the SCTE 130 Part 8 interface to optionally extend concrete messages derived from SCTE 130 Part 8 message types with attributes not specified by the schema.

APPENDIX G. MUTABLE GIS EXAMPLES (INFORMATIVE)

The following sections contain a selection of Mutable GIS top level example messages. All examples assume the following namespace prefix associations:

- xmlns="http://www.scte.org/schemas/130-8/2011/gis/mutable"
- xmlns:gis="http://www.scte.org/schemas/130-8/2011/gis"
- xmlns:core=http://www.scte.org/schemas/130-2/2008a/core

G.1 Listing Supported Features

Example 35 contains an example of ListSupportedFeatureRequest message.

```
  <gis:xxxListSupportedFeaturesRequest messageId="acs-342" system="GISClient"
version="1.0" identity="40DA910E-01AF-5050-C7EA-5D7B4A475311"/>
```

Example 35. ListSupportedFeaturesRequest Message

Example 36 contains a ListSupportedFeatureResponse message example containing a mutable service data model description.

```
  <gis:xxxListSupportedFeaturesResponse messageId="sca-343" system="GISServer" version="1.0"
identity="40DA910E-01AF-5050-C7EA-5D7B4A475312" messageRef="acs-342">
		<core:StatusCode class="0"/>
		<core:Callout>
			<core:Address type="SOAP 1.1">http://10.250.30.22/GISServer</core:Address>
		</core:Callout>
		<MutableServiceDataModelProfile>
			<gis:ServiceDataModel>http://SuperDemographics.com</gis:ServiceDataModel>
			<gis:AdvancedQueryLanguage>XPath</gis:AdvancedQueryLanguage>
			<gis:AdvancedQueryLanguage>XQuery</gis:AdvancedQueryLanguage>
			<UpdateExpressionLanguage version="1.7">JavaScript</UpdateExpressionLanguage>
		</MutableServiceDataModelProfile>
  </gis:xxxListSupportedFeaturesResponse>
```

Example 36. ListSupportedFeaturesResponse Message

G.2 Listing Qualifiers

Example 37 contains an example of ListQualifiersRequest message.

```
  <gis:xxxListQualifiersRequest messageId="acs-344" system="GISClient" version="1.0"
identity="40DA910E-01AF-5050-C7EA-5D7B4A475311">
        <gis:ServiceDataModel>http://SuperDemographics.com</gis:ServiceDataModel>
  </gis:xxxListQualifiersRequest>
```

Example 37. ListQualifiers Request Message

Example 38 contains an example of ListQualifiersResponse message for a mutable service data model.

```
  <gis;xxxListQualifiersResponse messageId="sca-345" system="GISServer" version="1.0" identity="40DA910E-
01AF-5050-C7EA-5D7B4A475312" messageRef="acs-344">
        <core:StatusCode class="0"/>
        <gis:BasicQueryDataModelDescription>
              <gis:ServiceDataModel>http://SuperDemographics.com</gis:ServiceDataModel>
              <gis:UniqueQualifierDeclaration>
                    <gis:QualifierDeclaration name="MACAddress"/>
              </gis:UniqueQualifierDeclaration>
              < gis:QualifierDescription name="Age" valueType="enumeration">
                    <gis:EnumerationValue>UnderTwenty</gis:EnumerationValue>
                    <gis:EnumerationValue>TwentyToForty</gis:EnumerationValue>
                    <gis:EnumerationValue>FortyToSixty</gis:EnumerationValue>
                    <gis:EnumerationValue>OverSixty</gis:EnumerationValue>
              </gis:QualifierDescription>
              <gis:QualifierDescription name="Income" valueType="enumeration">
                    <gis:EnumerationValue>Under50K</gis:EnumerationValue>
                    <gis:EnumerationValue>50Kto100K</gis:EnumerationValue>
                    <gis:EnumerationValue>100Kto200K</gis:EnumerationValue>
                    <gis:EnumerationValue>Over200K</gis:EnumerationValue>
              </gis:QualifierDescription>
              <MutableQualifierDescription name="mutable1" valueType="enumeration">
                    <gis:EnumerationValue>Hockey</gis:EnumerationValue>
                    <gis:EnumerationValue>Football</gis:EnumerationValue>
                    <gis:EnumerationValue>Baseball</gis:EnumerationValue>
                    <gis:EnumerationValue>Basketball</gis:EnumerationValue>
                    <gis:EnumerationValue>Soccer</gis:EnumerationValue>
                    <gis:EnumerationValue>Tennis</gis:EnumerationValue>
                    <gis:EnumerationValue>Fishing</gis:EnumerationValue>
                    <gis:EnumerationValue>Hunting</gis:EnumerationValue>
                    <gis:EnumerationValue>None</gis:EnumerationValue>
              </MutableQualifierDescription>
              <MutableQualifierDescription name="mutable2" valueType="integer">
                    <gis:MinInteger>0</gis:MinInteger>
              </MutableQualifierDescription>
        </gis:BasicQueryDataModelDescription>
  </gis:xxxListQualifiersResponse>
```

Example 38. ListQualifiersResponse Message

G.3 Mutable Operation

Example 39 contains an example of MutableOperationRequest message. The operation in this example is executed synchronously.

```
<xxxMutableOperationRequest messageId="acs-350" system="GISClient" version="1.0"
identity="40DA910E-01AF-5050-C7EA-5D7B4A475311">
  <gis:ServiceDataModel>http://SuperDemographics.com</gis:ServiceDataModel>
  <PutDataModelQualifier>
          <MutableQualifierDescription name="aqName1" valueType="string"/>
          <MutableQualifierDescription name="aqName2" valueType="string"/>
  </PutDataModelQualifier>
  <PutDataModelQualifier>
          <MutableQualifierDescription name="aqName3" valueType="string"/>
  </PutDataModelQualifier>
  <RemoveDataModelQualifier>
          <gis:QualifierDeclaration name="aqName0"/>
          <gis:QualifierDeclaration name="aqName01"/>
  </RemoveDataModelQualifier>
  <RemoveDataModelQualifier>
          <gis:QualifierDeclaration name="aqName02"/>
  </RemoveDataModelQualifier>
  <PutQualifier updateId="1">
          <gis:UniqueQualifier>
                  <gis:Qualifier name="stbId" value="1111111111111113"></gis:Qualifier>
          </gis:UniqueQualifier>
          <QualifierValueIncrement name="aqName03" value="1"/>
          <QualifierValueDecrement name="aqName05" value="2"/>
          <QualifierValueExpression name="aqName06">
                  <Expression language="JavaScript" version="1.5">a*value</Expression>
                  <!-- 'value' refers to the current value of the qualifier -->
                  <EvaluationArgument name="a" value="5"/>
          </QualifierValueExpression>
  </PutQualifier>
  <PutQualifier updateId="2">
          <gis:UniqueQualifier>
                  <gis:Qualifier name="stbId" value="1111111111111111"></gis:Qualifier>
          </gis:UniqueQualifier>
          <QualifierValue name="aqName03" value="123"/>
          <QualifierValue name="aqName04" value="aqValue234"/>
  </PutQualifier>
  <PutQualifier updateId="3">
          <gis:UniqueQualifier>
                  <gis:Qualifier name="stbId" value="1111111111111112"></gis:Qualifier>
          </gis:UniqueQualifier>
          <QualifierValue name="aqName03" value="456"/>
  </PutQualifier>
</xxxMutableOperationRequest>
```

Example 39. MutableOperationRequest Message (synchronous)

Example 40 contains an example of MutableOperationResponse message. The response in this example is for a successful operation.

```
  <xxxMutableOperationResponse messageId="sca-345" system="GISServer" version="1.0"
identity="40DA910E-01AF-5050-C7EA-5D7B4A475311" messageRef="acs-350" >
    <core:StatusCode class="0"/>
<xxxMutableOperationResponse>
```

Example 40. MutableOperationResponse Message

Example 41contains an example of MutableOperationRequest message. The operation in this example is executed asynchronously.

```
<xxxMutableOperationRequest messageId="acs-351" system="GISClient" version="1.0"
identity="40DA910E-01AF-5050-C7EA-5D7B4A475311">
  <gis:ServiceDataModel>http://SuperDemographics.com</gis:ServiceDataModel>
  <core:Callout>
        <core:Address type="SOAP 1.1">http://10.250.30.21/GISClient</core:Address>
  </core:Callout>
  <PutQualifier updateId="1">
        <gis:UniqueQualifier>
              <gis:Qualifier name="stbId" value="1111111111111111"></gis:Qualifier>
        </gis:UniqueQualifier>
        <QualifierValueIncrement name="aqName03" value="1"/>
        <QualifierValueDecrement name="aqName05" value="2"/>
  </PutQualifier>
  <PutQualifier updateId="2">
        <gis:UniqueQualifier>
              <gis:Qualifier name="stbId" value="1111111111111111"></gis:Qualifier>
        </gis:UniqueQualifier>
        <QualifierValue name="aqName03" value="100"/>
        <QualifierValue name="aqName04" value="aqValue234"/>
  </PutQualifier>
</xxxMutableOperationRequest>
```

Example 41. MutableOperationRequest Message (asyncronous)

Upon receipt of an asynchronous request a logical service replies with a response that in most cases will be a successful response and will look like Example 40. After the operation completes, the logical service will send a mutable operation notification message. Example 42 contains an example of MutableOperationNotification message:

```
<xxxMutableOperationNotification messageId="sca-346" system="GISServer" version="1.0"
identity="40DA910E-01AF-5050-C7EA-5D7B4A475311" messageRef="acs-351" >
  <core:StatusCode class="1"/>
  <UpdateError updateIdRef="2">
            <core:StatusCode class="1">
                      <core:Note>This update failed.</core:Note>
            </core:StatusCode>
  </UpdateError>
</xxxMutableOperationNotification>
```

Example 42. MutableOperationNotification Message

Upon receipt of the MutableOperationNotification message the Consumer will respond with a MutableOperationAcknowledgement message.

Example 43 contains an example of MutableOperationAcknowledgement message.

```
<xxxMutableOperationAcknowledgement messageId="acs-352" system="GISClient"
messageRef="sca-346" version="1.0" identity="40DA910E-01AF-5050-C7EA-
5D7B4A475311">
   <core:StatusCode class="0"/>
</xxxMutableOperationAcknowledgement>
```

Example 43. MutableOperationAcknowledgement Message

G.4 Batch Mutable Operation

Example 44 contains an example of BatchCreateRequest message.

```
<xxxBatchCreateRequest messageId="acs-360" system="GISClient" version="1.0"
identity="40DA910E-01AF-5050-C7EA-5D7B4A475311">
  <gis:ServiceDataModel>http://SuperDemographics.com</gis:ServiceDataModel>
  <core:Callout>
            <core:Address type="SOAP 1.1">http://10.250.30.21/GISClient</core:Address>
  </core:Callout>
</xxxBatchCreateRequest>
```

Example 44. BatchCreateRequest Message

Example 45 contains an example of BatchCreateResponse message.

```
<xxxBatchCreateResponse messageId="sac-348" system="GISServer" version="1.0"
messageRef="acs-360" identity="40DA910E-01AF-5050-C7EA-5D7B4A475311">
  <core:StatusCode class="0"/>
  <Batch batchId="1234"/>
</xxxBatchCreateResponse>
```

Example 45. BatchCreateResponse Message

Example 46 contains an example of BatchItemRequest message.

```
<xxxBatchItemRequest messageId="acs-361" system="GISClient" version="1.0"
batchIdRef="1234" identity="40DA910E-01AF-5050-C7EA-5D7B4A475311">
  <PutQualifier updateId="1">
		<gis:UniqueQualifier>
			<gis:Qualifier name="stbId" value="1111111111111111"></gis:Qualifier>
		</gis:UniqueQualifier>
		<QualifierValueIncrement name="aqName03" value="1"/>
		<QualifierValueDecrement name="aqName05" value="2"/>
  </PutQualifier>
 <PutQualifier updateId="2">
		<gis:UniqueQualifier>
			<gis:Qualifier name="stbId" value="1111111111111111"></gis:Qualifier>
		</gis:UniqueQualifier>
		<QualifierValue name="aqName03" value="100"/>
		<QualifierValue name="aqName04" value="aqValue234"/>
  </PutQualifier>
</xxxBatchItemRequest>
```

Example 46. BatchItemRequest Message

Example 47 contains an example of BatchItemResponse message.

```
<xxxBatchItemResponse messageId="sac-350" system="GISServer" version="1.0"
messageRef="acs-361" identity="40DA910E-01AF-5050-C7EA-5D7B4A475311">
  <core:StatusCode class="0"/>
</xxxBatchItemResponse>
```

Example 47. BatchItemResponse Message

Example 48 contains an example of BatchOperationRequest message.

```
<xxxBatchOperationRequest messageId="acs-362" system="GISClient" version="1.0"
batchIdRef="1234" identity="40DA910E-01AF-5050-C7EA-5D7B4A475311">
  <ExecuteBatch transactionLevel="message"/>
</xxxBatchOperationRequest>
```

Example 48. BatchOperationRequest Message

Example 49 contains an example of BatchOperationResponse message.

```
<xxxBatchOperationResponse messageId="sac-365" system="GISServer" version="1.0"
messageRef="acs-362" identity="40DA910E-01AF-5050-C7EA-5D7B4A475311"
batchIdRef="1234">
  <core:StatusCode class="5"/>
  <UpdateError updateIdRef="1">
        <core:StatusCode class="1"></core:StatusCode>
  </UpdateError>
</xxxBatchOperationResponse>
```

Example 49. BatchOperationResponse Message

Example 50 contains an example of BatchNotification message.

```
<xxxBatchNotification messageId="sac-360" system="GISServer" version="1.0"
batchIdRef="1234" identity="40DA910E-01AF-5050-C7EA-5D7B4A475311">
  <core:StatusCode class="5">
        <core:Note>Batch was cancelled</core:Note>
  </core:StatusCode>
</xxxBatchNotification>
```

Example 50. BatchNotification Message

Example 51 contains an example of BatchNotificationAcknowledgement message.

```
<xxxBatchNotificationAcknowledgement messageId="acs-370" system="GISClient"
messageRef="sac-360" version="1.0" identity="40DA910E-01AF-5050-C7EA-
5D7B4A475311">
  <core:StatusCode class="0"/>
</xxxBatchNotificationAcknowledgement>
```

Example 51. BatchNotificationAcknowledgement Message

G.5 Data Model Description Notification

In order to receive notifications on data model description changes, a Consumer need to register a notification containing the DataModelDescriptionQuery element. The following is an example of such a notification registration.

```
<gis:xxxNotificationRegistrationRequest messageId="acs-380" system="GISClient"
version="1.0"
identity="40DA910E-01AF-5050-C7EA-5D7B4A475311">
  <core:Callout>
            <core:Address type="SOAP 1.1">http://10.250.30.21/GISClient</core:Address>
  </core:Callout>
  <DataModelDescriptionQuery queryId="123">
            <gis:ServiceDataModel>http://SuperDemographics.com</gis:ServiceDataModel>
  </DataModelDescriptionQuery>
</gis:xxxNotificationRegistrationRequest>
```

Example 52. NotificationRegistrationRequest Message

The following is an example of Notification message.

```
<gis:xxxNotification messageId="sac-365" system="GISServer" version="1.0"
noticeType="update"
identity="40DA910E-01AF-5050-C7EA-5D7B4A475311">
  <gis:QueryResult resultSetSize="1" queryRef="123">
   <DataModelDescriptionQueryResult isChanged="true"/>
 </gis:QueryResult>
</gis:xxxNotification>
```

Example 53. Notification Message

####