International Telecommunication Union

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

J.380.8

(11/2011)

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

Digital transmission of television signals

Digital Program Insertion – Advertising systems interfaces – General information service

Recommendation ITU-T J.380.8

Recommendation ITU-T J.380.8

Digital program insertion – Advertising systems interfaces – General information service

Summary

Recommendation UIT-T J.380.8 describes the digital program insertion advertising systems interfaces' general information service (GIS) messaging and data type specification using XML, XML namespaces, and XML schema.

History

Edition	Recommendation	Approval	Study Group
1.0	ITU-T J.380.8	2011-11-13	9

Table of Contents

Page

List of Figures

List of Tables

Introduction

Several logical services defined by the Recommendation ITU-T J.380 series 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) Recommendation.

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 ITU-T J.380 logical service.

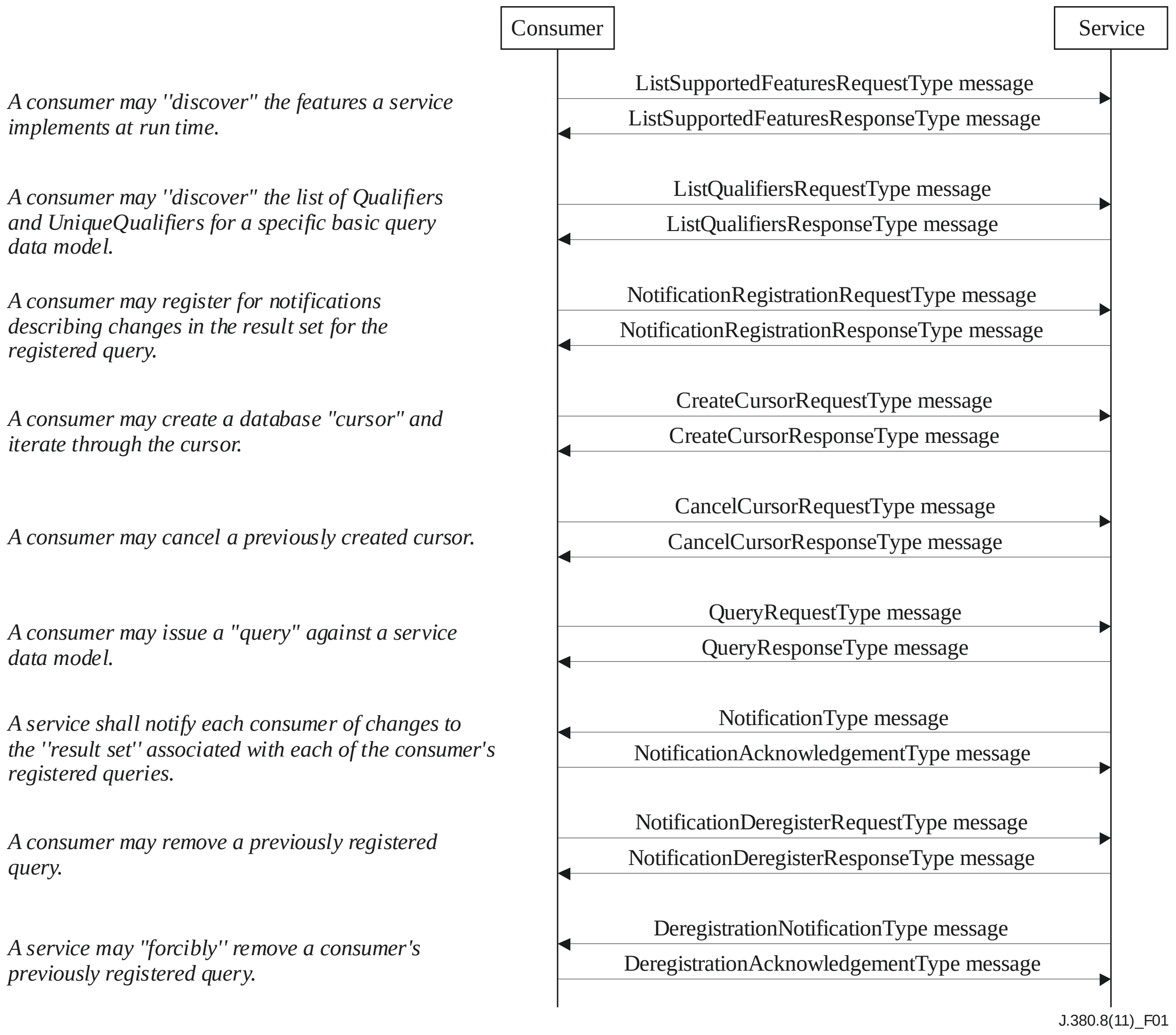

Consumer
Service
A consumer may "discover" the features a service implements at run time.
ListSupportedFeaturesRequestType message
ListSupportedFeaturesResponseType message
A consumer may "discover" the list of Qualifiers and UniqueQualifiers for a specific basic query data model.
ListQualifiersRequestType message
ListQualifiersResponseType message
A consumer may register for notifications describing changes in the result set for the registered query.
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
J.380.8(11)_F01

Figure 1 – General information service interface

The GIS interface may be implemented by a logical service that provides data to other logical services. This Recommendation 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 Recommendation 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 or 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 clause 8.3 for additional information on advanced query language support and clause 8 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 "UniqueQualifier". 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 clause 8.26 and clause 8.18 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 an @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 clauses 8.26 and 8.19 for additional information on the UniqueQualifierDeclaration and QualifierDeclaration elements respectively.

This Recommendation 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 clause 8.10 for additional information on cursors.

The GIS Recommendation includes a notification model with associated notification management functions such as registration, deregistration and listing of active registrations. The GIS Recommendation does not restrict the number of data stores that a logical service may use.

Recommendation ITU-T J.380.8

Digital Program Insertion – Advertising Systems Interfaces – General Information Service

1 Scope

This document 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 References

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

[ITU-T J.380.2]	Recommendation ITU-T J.380.2 (2011), *Digital program insertion – Advertising systems interfaces – Core data elements*.
[ITU-T J.380.7]	Recommendation ITU-T J.380.7 (2011), *Digital program insertion – Advertising systems interfaces – Message transport*.
[SCTE 130-8 Schema]	ANSI/SCTE 130-8 (2010), *Digital Program Insertion – Advertising Systems Interfaces Part 8 – General Information Service (GIS)*, schema file.
[W3C-XPath]	W3C Recommendation XPath (1999), *XML Path Language (XPath) Version 1.0*.
[W3C-XQuery]	W3C Recommendation XQuery (2007), *XQuery 1.0: An XML Query Language*.
[W3C-XSD]	W3C Recommendation (2004), *XML Schema Part 1: Structures Second Edition*. W3C Recommendation (2004), *XML Schema Part 2: Datatypes Second Edition*.

3 Definitions

Throughout this standard the terms below have specific meanings. As some of the terms are defined in other ITU-T J.380 documents and have 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 [ITU-T J.380-2] definitions are included herein. See [ITU-T J.380.2] for additional information.

3.1 Terms defined elsewhere

This Recommendation uses the following terms defined elsewhere:

None.

3.2 Terms defined in this Recommendation

This Recommendation defines the following terms:

3.2.1 advanced query: The "Advanced Query" interface defined by Recommendation ITU-T J.380.8 permits the consumer of a logical service implementation derived from J.380.8 to use an advanced query language to formulate queries against a logical service's data model.

3.2.2 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 both examples of an advanced query language.

3.2.3 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.

3.2.4 basic query: To obtain information from a logical service implementation derived from Recommendation ITU-T J.380.8, 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.

3.2.5 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.

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

3.2.7 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.

3.2.8 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.

3.2.9 default data model: The data model that is 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).

3.2.10 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.

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

3.2.12 UniqueQualifier: A "Unique Qualifier" 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 UniqueQualifier. However, an object may have more than one UniqueQualifier.

3.2.13 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.

4 Abbreviations and acronyms

All [ITU-T J.380.2] abbreviations are included herein. See [ITU-T J.380.2] for additional information.

This Recommendation uses the following abbreviations and acronyms:

CDATA	Character Data

5 Conventions

5.1 Notational conventions

5.1.1 Normative XML schema

This Recommendation employs the same notational conventions as [ITU-T J.380.2]. Refer to [ITU-T J.380.2] for an explanation of notational conventions.

5.1.2 Document conventions

This Recommendation employs the same document conventions as [ITU-T J.380.2]. Refer to [ITU-T J.380.2] for an explanation of document conventions. For example, the XML schema illustration is explained there.

This Recommendation 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.

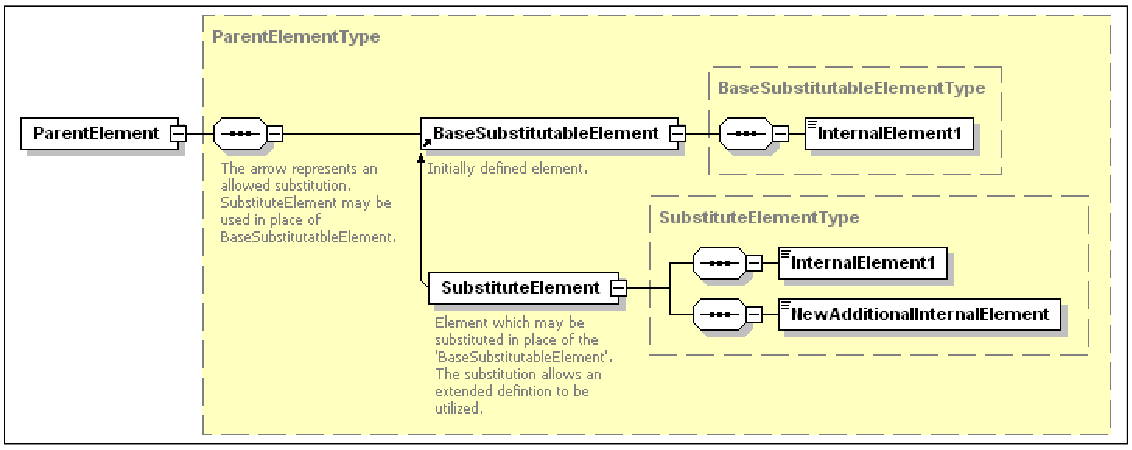

ParentElementType
BaseSubstitutableElementType
ParentElement
BaseSubstitutableElement
InternalElement1
The arrow represents an allowed substitution. SubstituteElement may be used in place of BaseSubstitutatbleElement.
Initially defined element.
SubstituteElementType
InternalElement1
SubstituteElement
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.

5.2 Processing conventions

Unknown/Unrecognized/Unsupported XML elements and attributes. See [ITU-T J.380.2] for information.

5.3 XML namespaces

This Recommendation 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.

Table 1 – XML namespace declarations

Prefix	Namespace	Description
core	http://www.scte.org/schemas/130-2/2008a/core	See [ITU-T J.380.2]
gis	http://www.scte.org/schemas/130-8/2010a/gis	ITU-T J.380.8 (this Recommendation)
xsd	http://www.w3.org/2001/XMLSchema	XML foundation. See W3C.

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>.

6 GIS Message types

The following topics are covered by [ITU-T J.380.2] and this Recommendation considers all aspects defined therein to be normative and applicable herein. See [ITU-T J.380.2] for additional information.

– Message format

– XML message carriage

– Transport mechanisms

– Message error handling

The GIS message interface shall include the messages defined in [ITU-T J.380.2] and logical services that incorporate the GIS message interface shall conform to the messaging transport mechanism described in [ITU-T J.380.7].

Table 2 identifies additional J.380.8 (GIS) specific message types.

Table 2 – GIS specific message types

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

Table 2 – GIS specific message types

Message	Description
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

Because this Recommendation defines an abstract messaging interface used by other logical service implementations, this Recommendation defines the message types rather than message elements. Every logical service implementing the interface specified by Recommendation ITU-T J.380.8 shall define a set of message elements semantically equivalent to the message types defined in Recommendation ITU-T J.380.8 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"/>
```

Figure 3 – Example 1 – SISListSupportedFeaturesRequest message definition

The following clauses provide specific information on ITU-T J.380.8's use of attributes and message types defined in [ITU-T J.380.2]. For detailed information on these attributes and message types see [ITU-T J.380.2].

6.1 @version attribute

Recommendation ITU-T J.380.8 specifies message types with the intent that other logical services may create service specific concrete message definitions using the ITU-T J.380.8 message types. Thus, Recommendation ITU-T J.380.8 shall not specify a value for the @version attribute. Logical services that derive messages from the ITU-T J.380.8 message types shall set the @version attribute to a value that reflects the logical service's revision.

6.2 Request base message type

All GIS top level ***request*** message types are derived from the core:Msg_RequestBaseType abstract base message type. See [ITU-T J.380.2] for details on the attributes and elements contained in this base message.

6.3 Response base message type

All GIS top level ***response*** message types are derived from the core:Msg_ResponseBaseType abstract base message type. See [ITU-T J.380.2] for details on the attributes and elements contained in this base message.

6.4 Notification base message type

All GIS top level ***notification*** message types are derived from the core:Msg_NotificationBaseType abstract base message type. See [ITU-T J.380.2] for details on the attributes and elements contained in this base message.

6.5 Acknowledgement base message type

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

6.6 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.

6.7 Default logical service channel end-point

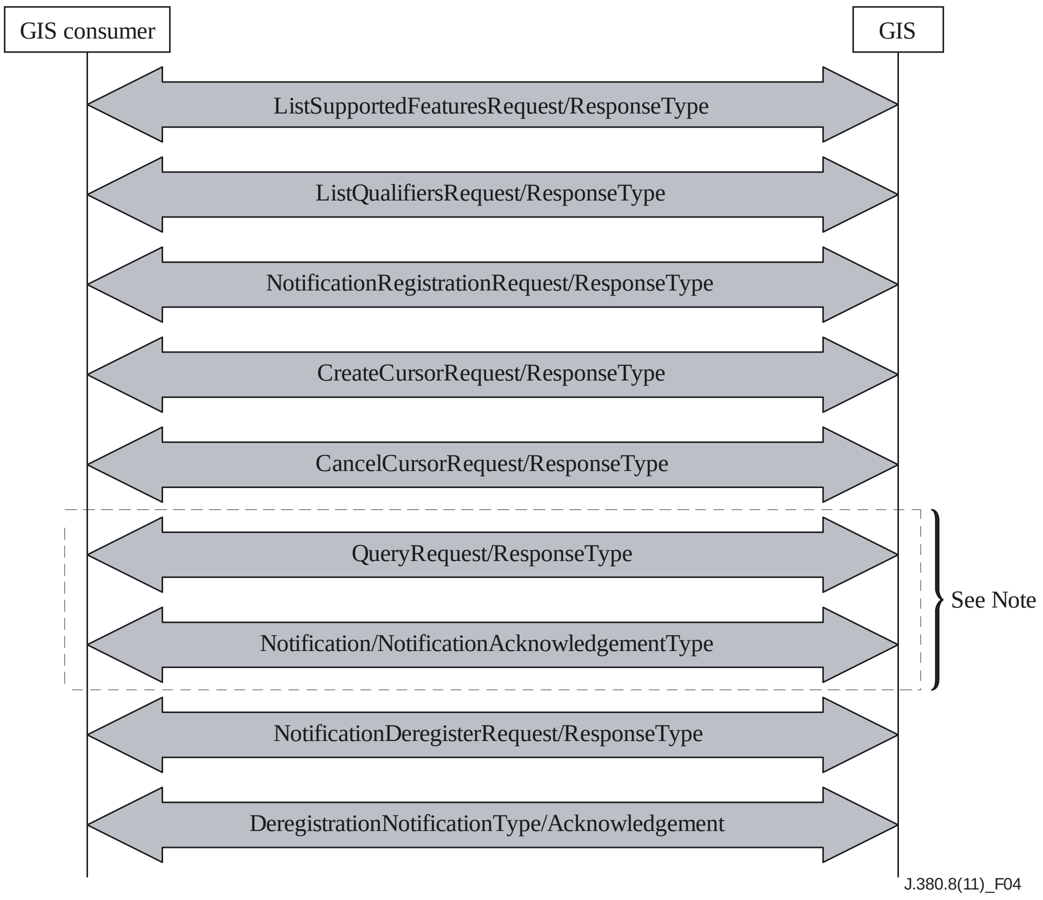
A logical service incorporating the GIS interface shall support the receipt of the ListSupportedFeaturesRequestType message type. Typically, this message type is received on a default (or well known) logical service channel end-point address. Other GIS messages types may also be offered on the same logical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

See clause 7.1.2 for additional information on the ListSupportedFeaturesResponseType message type.

See [ITU-T J.380.2] for a complete description of the terms (message, end-point and logical service channel).

7 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
NotificationDeregisterRequest/ResponseType
DeregistrationNotificationType/Acknowledgement
See Note
J.380.8(11)_F04

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

Figure 4 – GIS message exchange

Figure 4 illustrates all of the message exchanges that are specific to the GIS. The service check and service status message exchanges defined in [ITU-T J.380.2] are not depicted in this illustration.

7.1 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.

7.1.1 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 5.

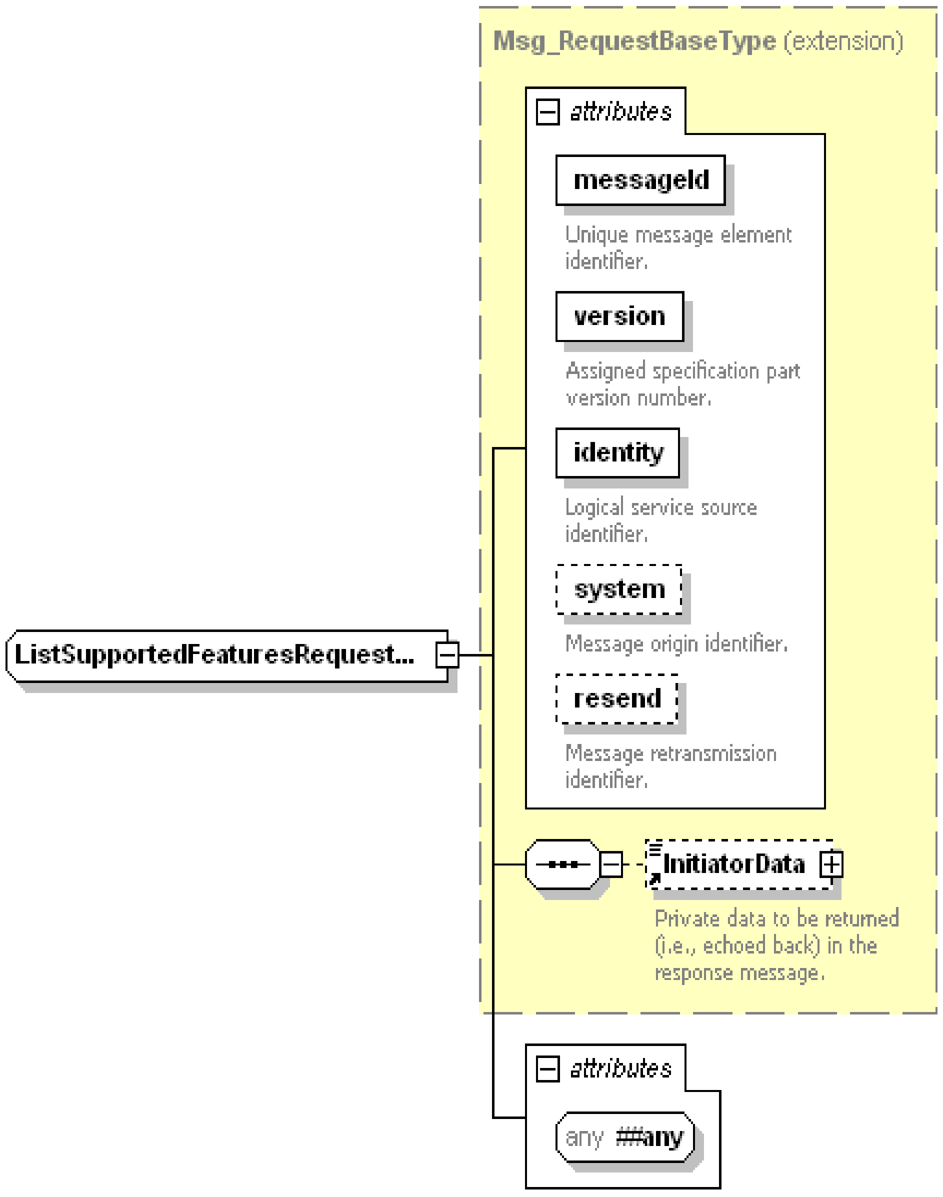

Msg_RequestBaseType (extension)
attributes
messageId
Unique message element identifier.
version
Assigned specification part version number.
identity
Logical service source identifier.
system
Message origin identifier.
resend
Message retransmission identifier.
ListSupportedFeaturesRequest...
InitiatorData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

Figure 5 – 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 ITU-T J.380.8 interface to optionally extend concrete messages derived from ITU-T J.380.8 message types with attributes not specified by the schema.

7.1.2 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 [ITU-T J.380.2] for additional information on the core:Callout element. The XML schema definition for this message is shown in Figure 6.

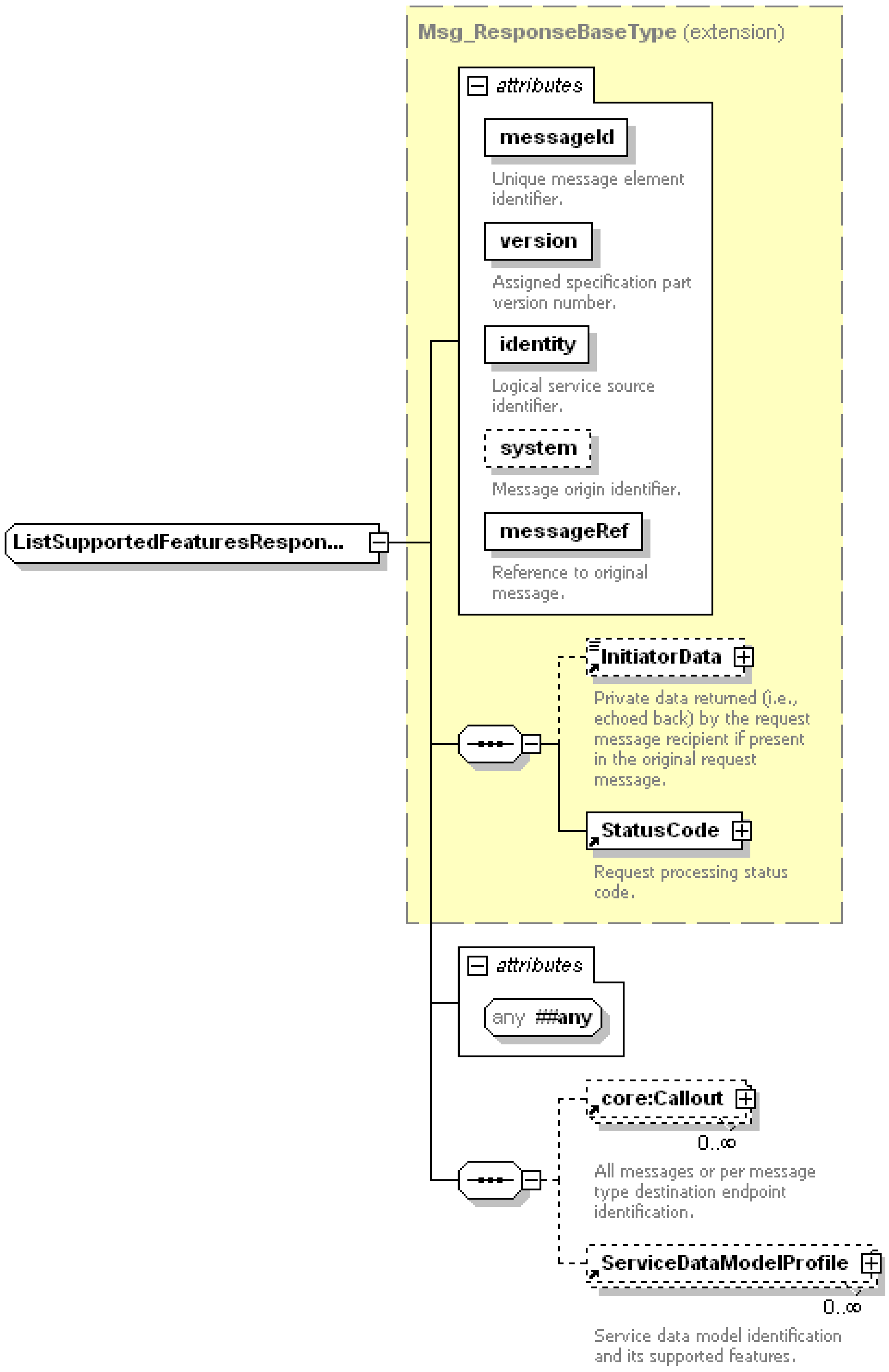

Msg_ResponseBaseType (extension)
attributes
messageId
Unique message element identifier.
version
Assigned specification part version number.
identity
Logical service source identifier.
system
Message origin identifier.
messageRef
Reference to original message.
ListSupportedFeaturesRespon...
InitiatorData
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 6 – 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 ITU-T J.380.8 interface to optionally extend concrete messages derived from ITU-T J.380.8 message types with attributes not specified by the schema.

core:Callout [Optional] – Zero or more core:Callout elements specifying the service channel message end-point(s). See [ITU-T J.380.2] 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.

Table 3 – ListSupportedFeaturesResponseType core:Callout @message values

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

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 [ITU-T J.380.2] 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 clause 8.24.

7.2 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 clause).

7.2.1 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 7.

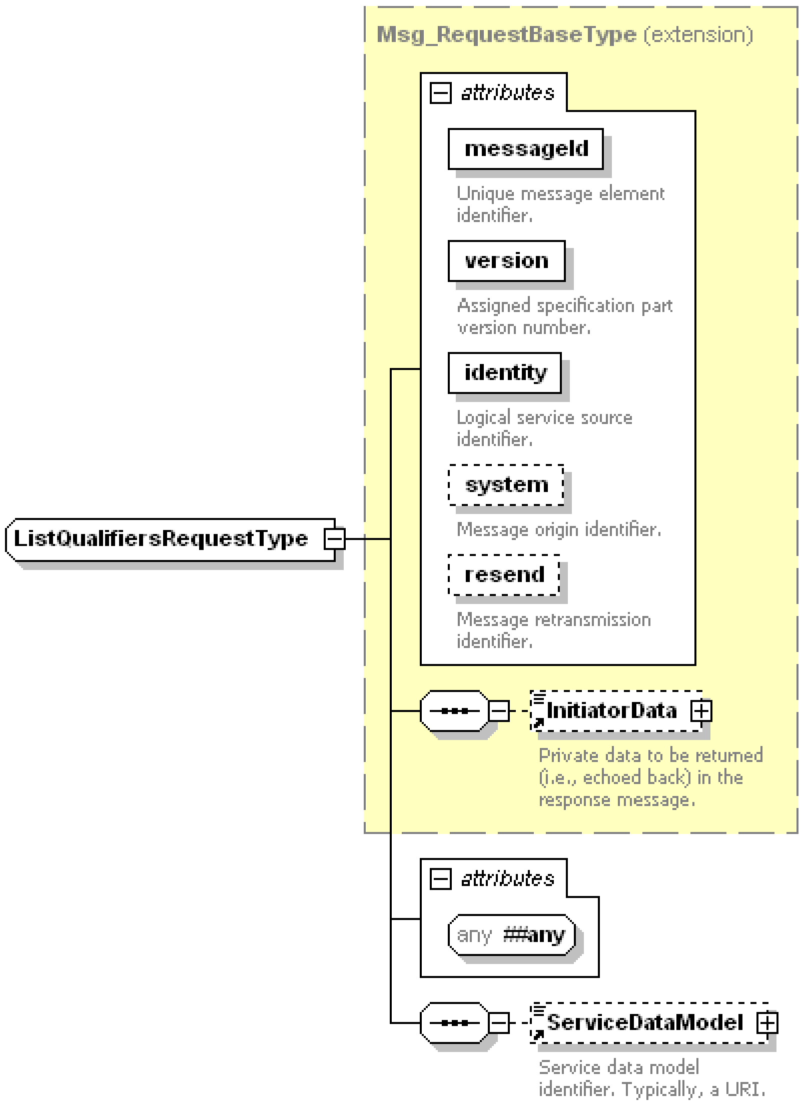

Msg_RequestBaseType (extension)
attributes
messageId
Unique message element identifier.
version
Assigned specification part version number.
identity
Logical service source identifier.
system
Message origin identifier.
resend
Message retransmission identifier.
ListQualifiersRequestType
InitiatorData
Private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

Figure 7 – 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 ITU-T J.380.8 interface to optionally extend concrete messages derived from ITU-T J.380.8 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 clause 8.23.

7.2.2 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 an @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 an @detailCode attribute set to the value core:ResourceNotFound.

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

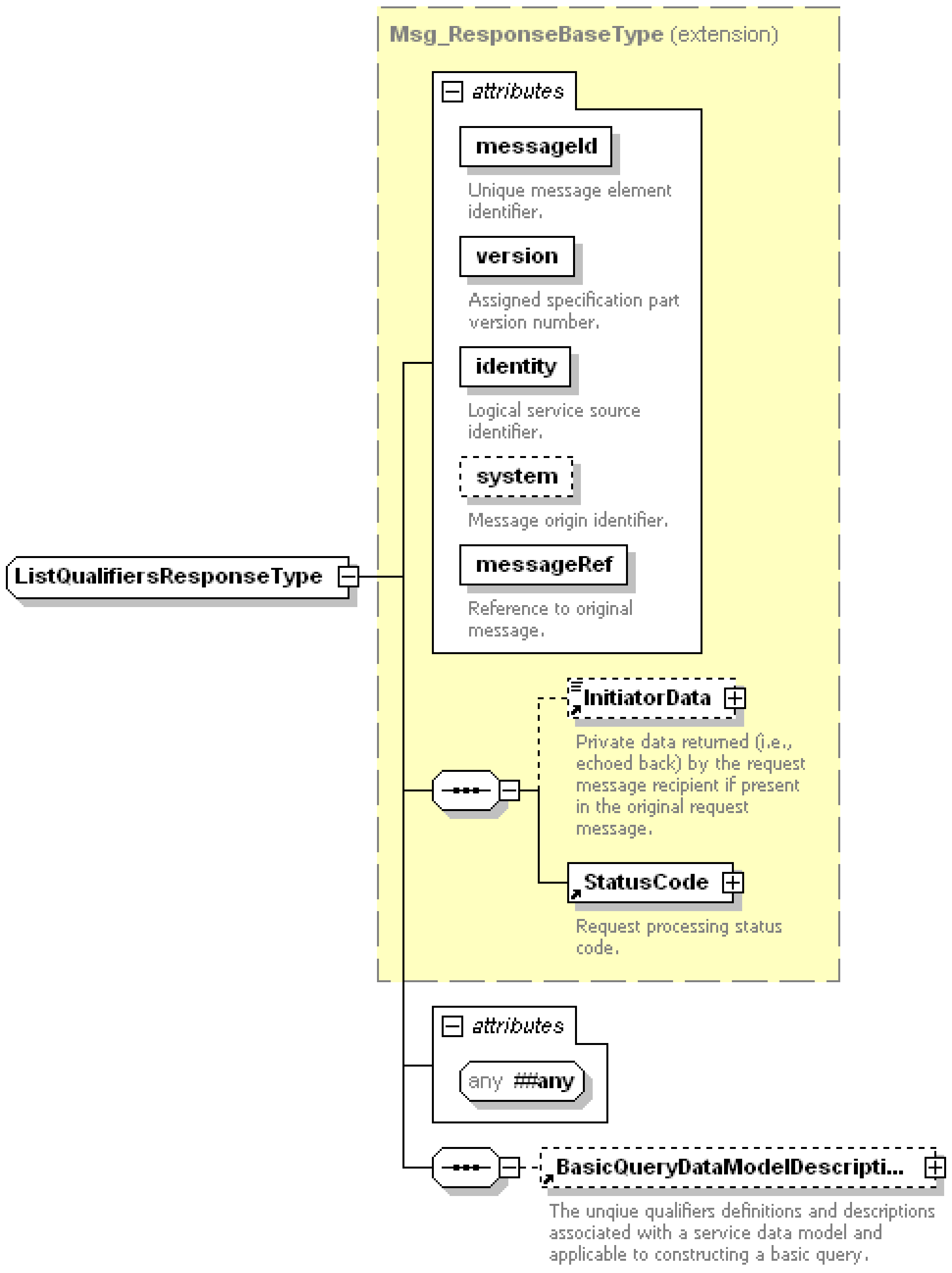

Msg_ResponseBaseType (extension)
attributes
messageId
Unique message element identifier.
version
Assigned specification part version number.
identity
Logical service source identifier.
system
Message origin identifier.
messageRef
Reference to original message.
ListQualifiersResponseType
InitiatorData
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 8 – 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 ITU-T J.380.8 interface to optionally extend concrete messages derived from ITU-T J.380.8 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 clause 8.7.

7.3 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)

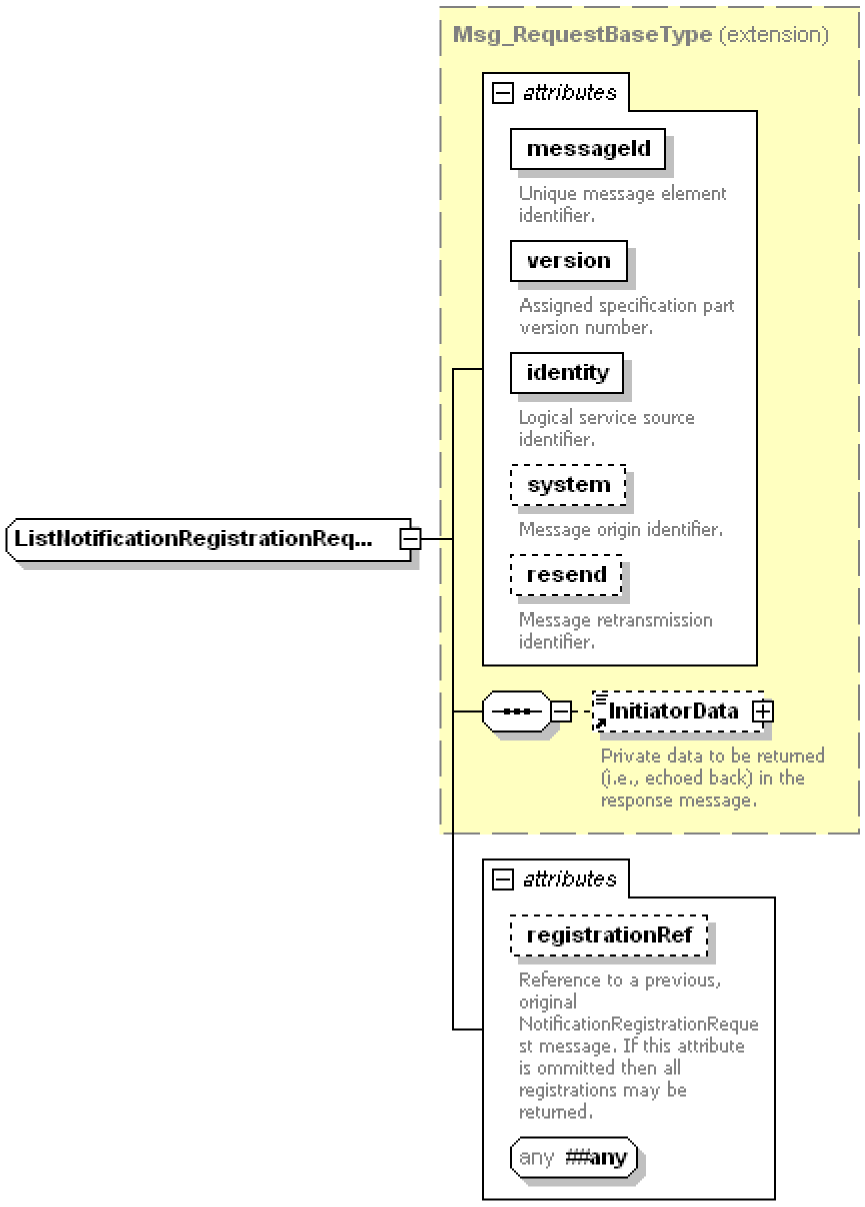
– Per registration message

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.

7.3.1 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 9.



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 9 – 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. Table 4 defines the inclusion relationships.

Table 4 – List ADS registration inquiry granularity control

@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

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 ITU-T J.380.8 interface to optionally extend concrete messages derived from ITU-T J.380.8 message types with attributes not specified by the schema.

7.3.2 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 10.

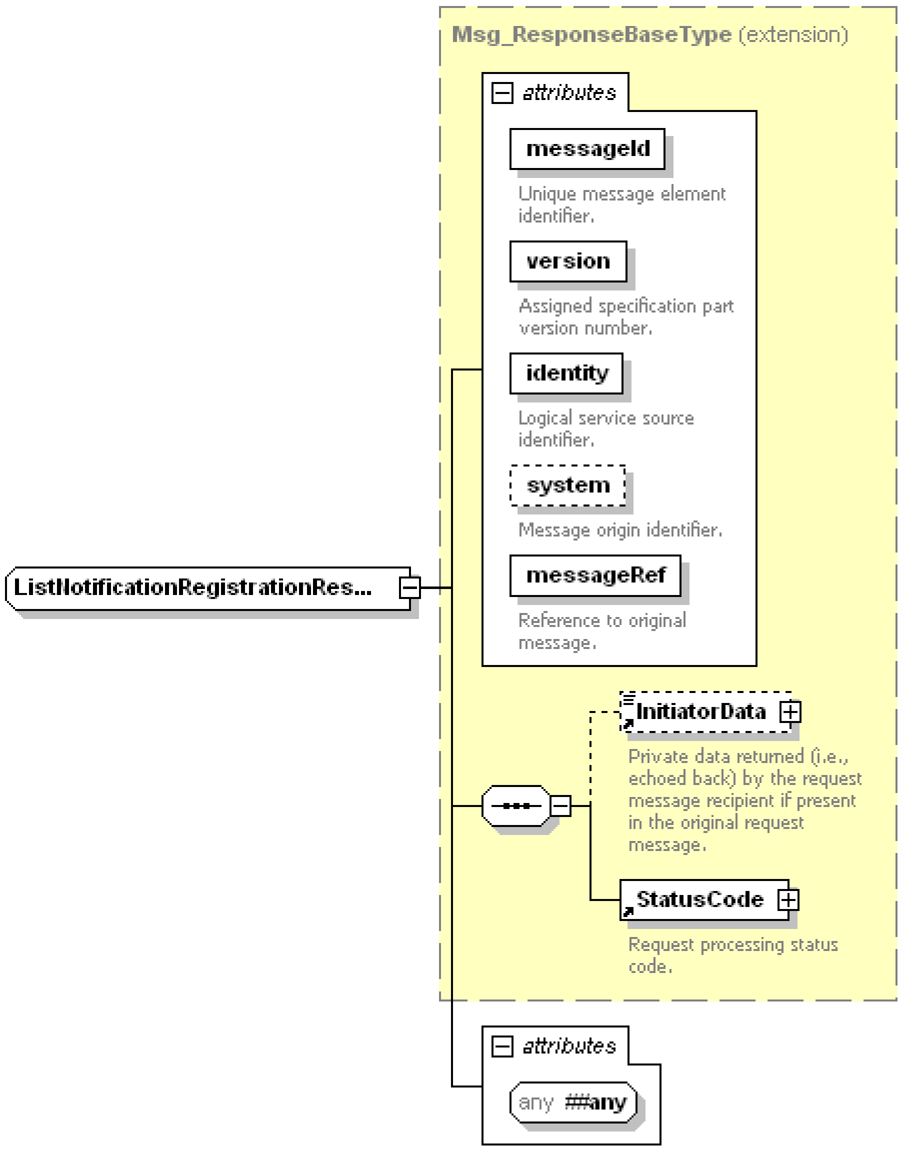

Msg_ResponseBaseType (extension)
attributes
messageId
Unique message element identifier.
version
Assigned specification part version number.
identity
Logical service source identifier.
system
Message origin identifier.
messageRef
Reference to original message.
ListNotificationRegistrationRes...
InitiatorData
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 10 – 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 ITU-T J.380.8 interface to optionally extend concrete messages derived from ITU-T J.380.8 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 an 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 [ITU-T J.380.2] 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).

7.4 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.

7.4.1 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 11.

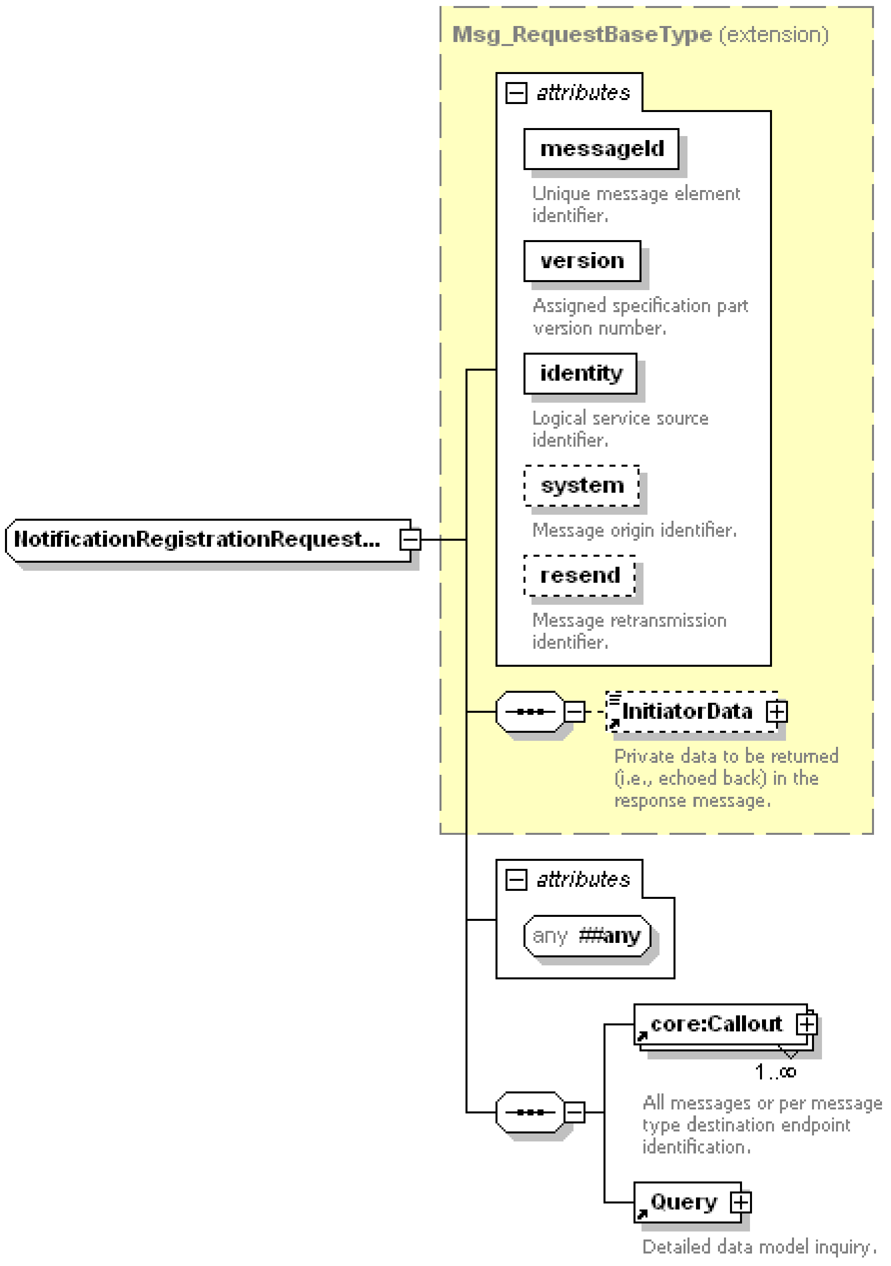

Msg_RequestBaseType (extension)
attributes
messageId
Unique message element identifier.
version
Assigned specification part version number.
identity
Logical service source identifier.
system
Message origin identifier.
resend
Message retransmission identifier.
InitiatorData
Private data to be returned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

Figure 11 – 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 ITU-T J.380.8 interface to optionally extend concrete messages derived from ITU-T J.380.8 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 [ITU-T J.380.2] 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.

A 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.

Table 5 – NotificationRegistrationRequest core:Callout @message values

@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 Recommendation. The string shall be prefixed with the text "private:".

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 [ITU-T J.380.2] 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 clause 8.21 for additional details on the Query element.

7.4.2 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 12.

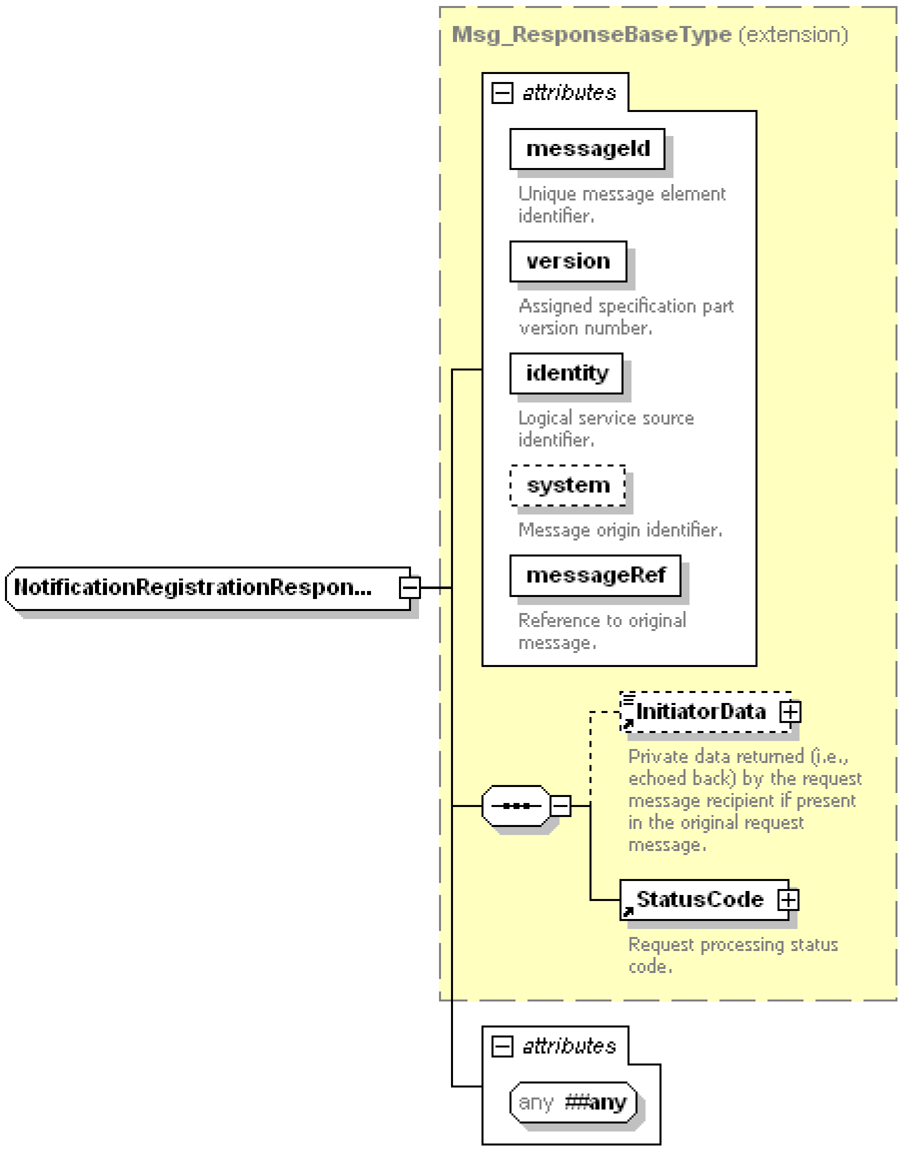

Msg_ResponseBaseType (extension)
attributes
messageId
Unique message element identifier.
version
Assigned specification part version number.
identity
Logical service source identifier.
system
Message origin identifier.
messageRef
Reference to original message.
NotificationRegistrationRespon...
InitiatorData
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 12 – 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 ITU-T J.380.8 interface to optionally extend concrete messages derived from ITU-T J.380.8 message types with attributes not specified by the schema.

7.5 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.

7.5.1 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).

Table 6 – Notification @noticeType values

@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 Recommendation. The string shall be prefixed with the text "private:".

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

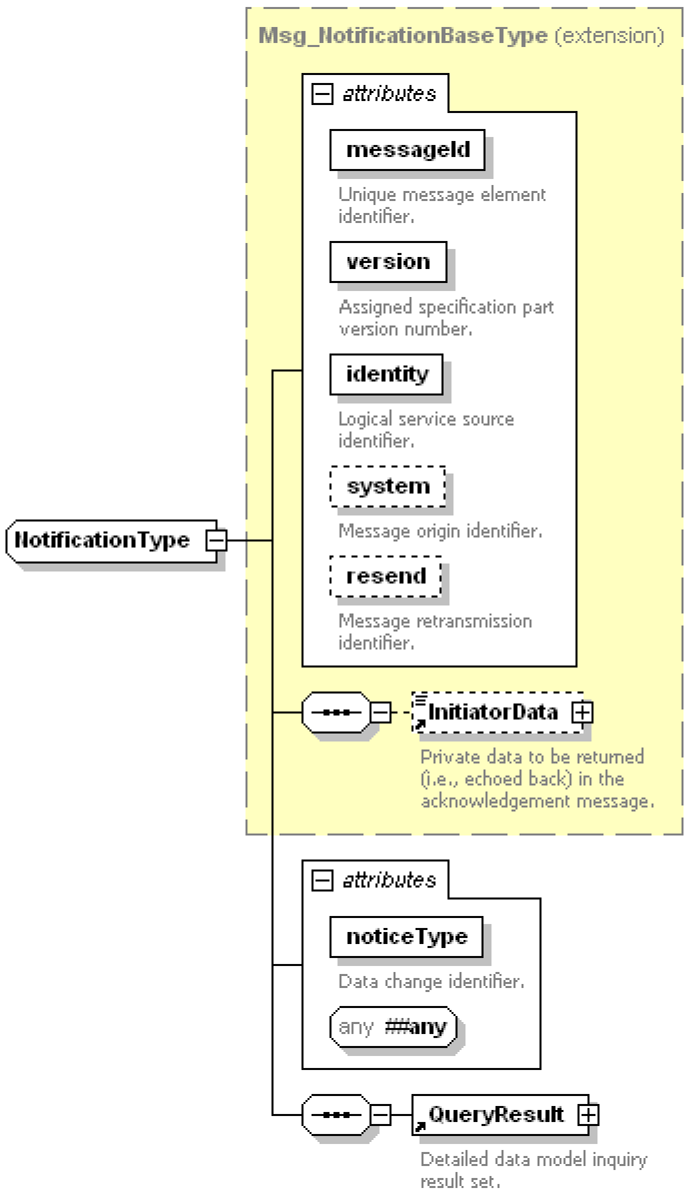

Msg_NotificationBaseType (extension)
attributes
messageId
Unique message element identifier.
version
Assigned specification part version number.
identity
Logical service source identifier.
system
Message origin identifier.
resend
Message retransmission identifier.
NotificationType
InitiatorData
Private 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 13 – 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 clause 9.9 for additional information.

@##any[Optional] – The @##any attribute enables a logical service incorporating the ITU-T J.380.8 interface to optionally extend concrete messages derived from ITU-T J.380.8 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 clause 8.22 for additional information on the QueryResult element.

7.5.2 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 14.

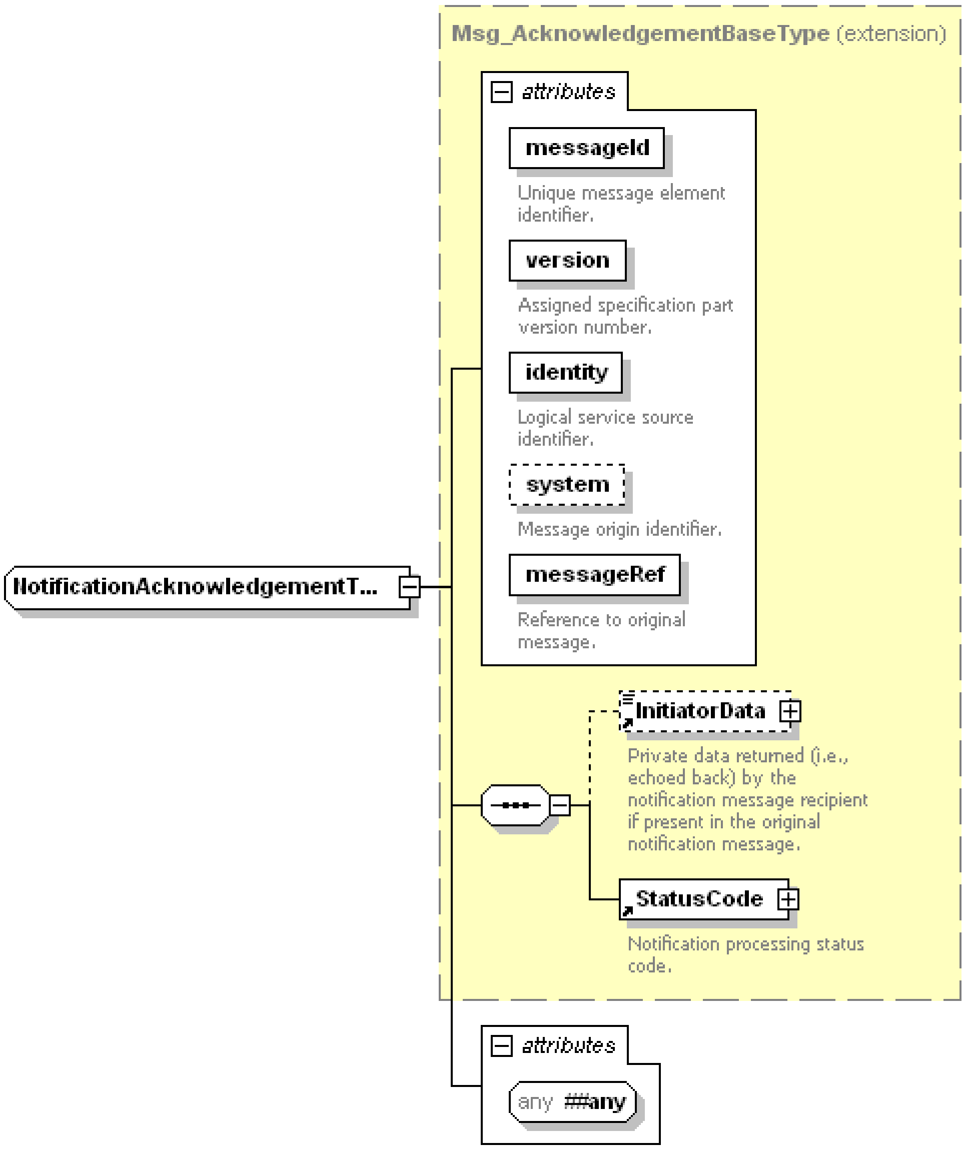

Msg_AcknowledgementBaseType (extension)
attributes
messageId
Unique message element identifier.
version
Assigned specification part version number.
identity
Logical service source identifier.
system
Message origin identifier.
messageRef
Reference to original message.
NotificationAcknowledgementT...
InitiatorData
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 14 – 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 ITU-T J.380.8 interface to optionally extend concrete messages derived from ITU-T J.380.8 message types with attributes not specified by the schema.

7.6 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 an @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 clause 8.10 for additional information on cursors.

7.6.1 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 15.

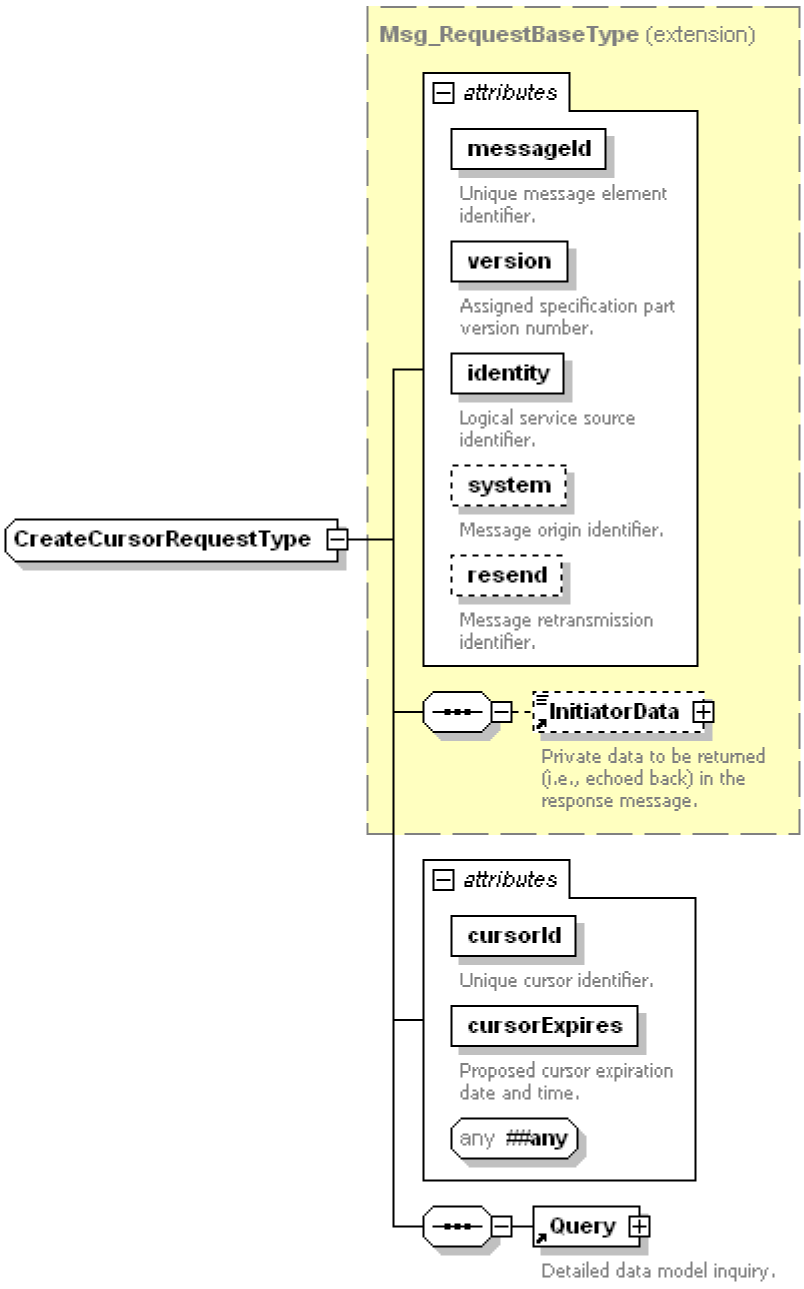

Msg_RequestBaseType (extension)
attributes
messageId
Unique message element identifier.
version
Assigned specification part version number.
identity
Logical service source identifier.
system
Message origin identifier.
resend
Message retransmission identifier.
CreateCursorRequestType
InitiatorData
Private 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 15 – 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 clause 9.1 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 [ITU-T J.380.2] for information on the core:dateTimeTimezoneType.

@##any[Optional] – The @##any attribute enables a logical service incorporating the ITU-T J.380.8 interface to optionally extend concrete messages derived from ITU-T J.380.8 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 clause 8.21 for additional information on the Query element.

7.6.2 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 16.

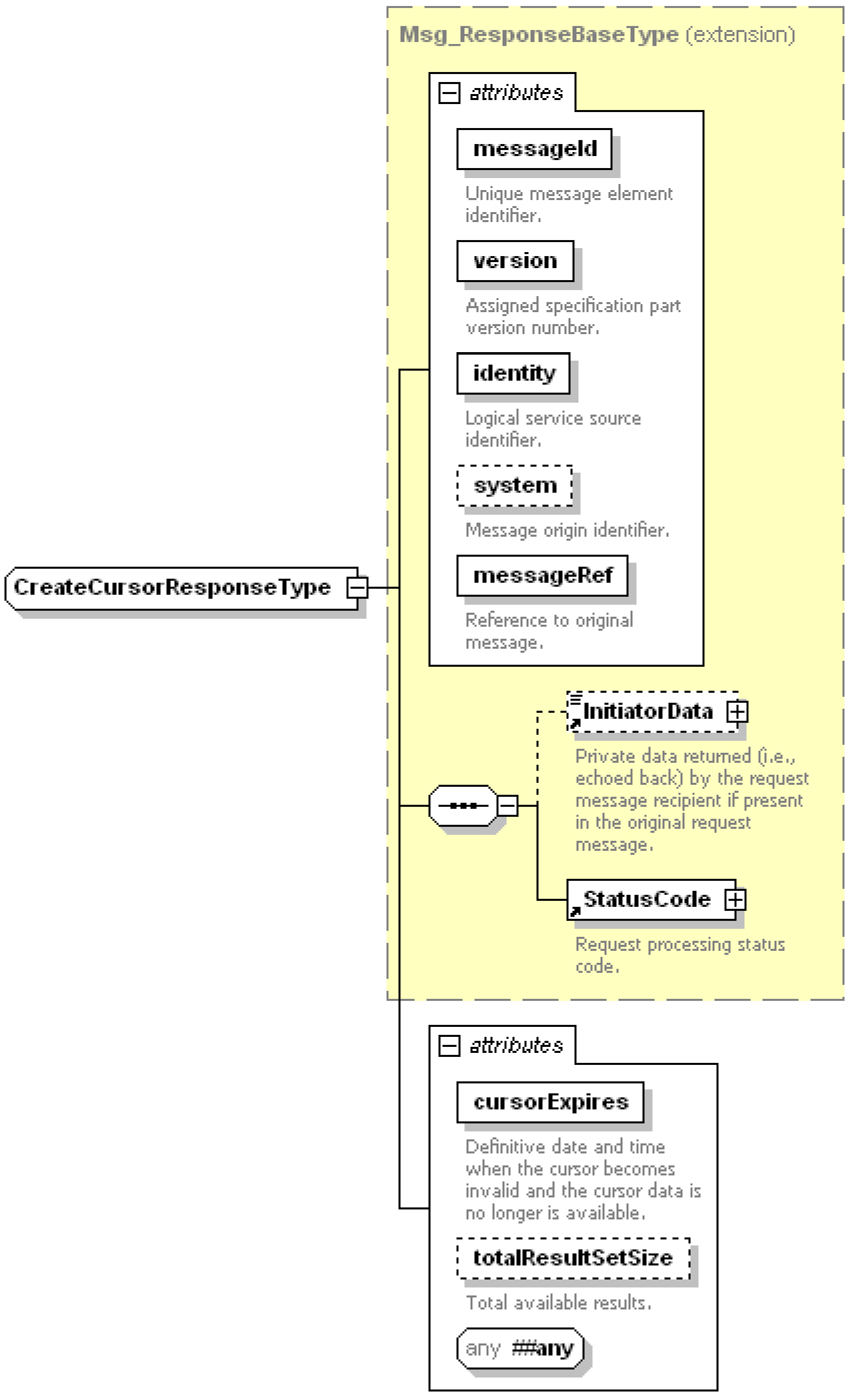

Msg_ResponseBaseType (extension)
attributes
messageId
Unique message element identifier.
version
Assigned specification part version number.
identity
Logical service source identifier.
system
Message origin identifier.
messageRef
Reference to original message.
CreateCursorResponseType
InitiatorData
Private 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 16 – 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 [ITU-T J.380.2] 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 clause 9.17 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 ITU-T J.380.8 interface to optionally extend concrete messages derived from ITU-T J.380.8 message types with attributes not specified by the schema.

7.7 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 cancelled 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.

7.7.1 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 17.

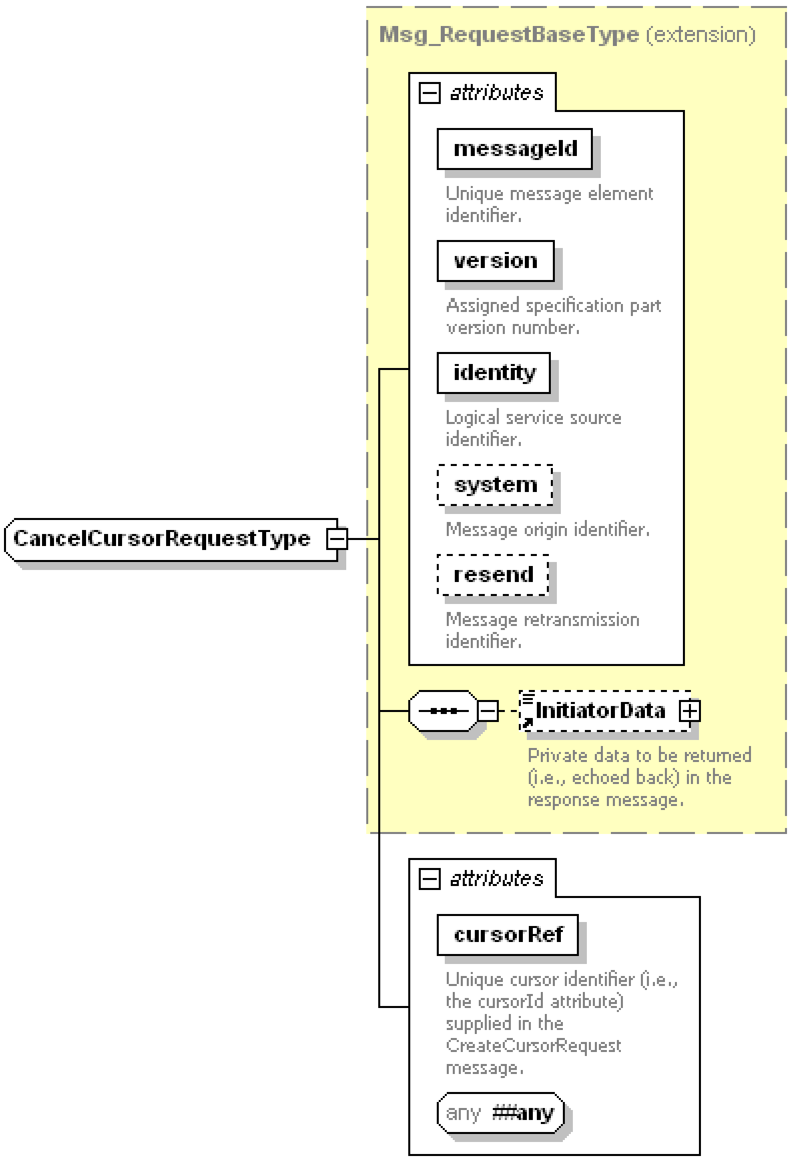

Msg_RequestBaseType (extension)
attributes
messageId
Unique message element identifier.
version
Assigned specification part version number.
identity
Logical service source identifier.
system
Message origin identifier.
resend
Message retransmission identifier.
CancelCursorRequestType
InitiatorData
Private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

Figure 17 – 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.

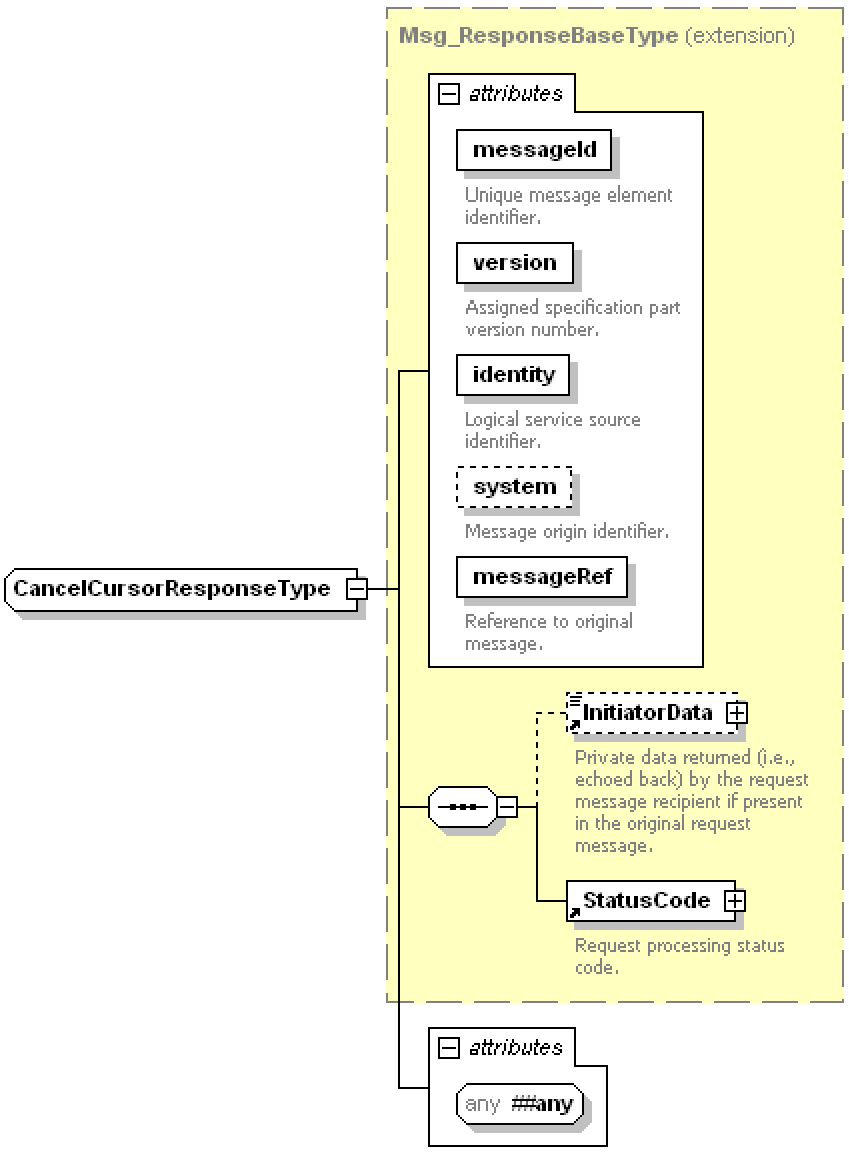
For example, if a CreateCursorRequest message type used an @cursorId value of '123' for a new cursor, at cancellation time for cursorId '123', the CancelCursorRequest message type shall include an @cursorRef with the value '123'. See clause 9.2 for additional information on cursorIdRefAttrType.

@##any[Optional] – The @##any attribute enables a logical service incorporating the ITU-T J.380.8 interface to optionally extend concrete messages derived from ITU-T J.380.8 message types with attributes not specified by the schema.

7.7.2 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 18.



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 18 – 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 ITU-T J.380.8 interface to optionally extend concrete messages derived from ITU-T J.380.8 message types with attributes not specified by the schema.

7.8 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.

7.8.1 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 19.

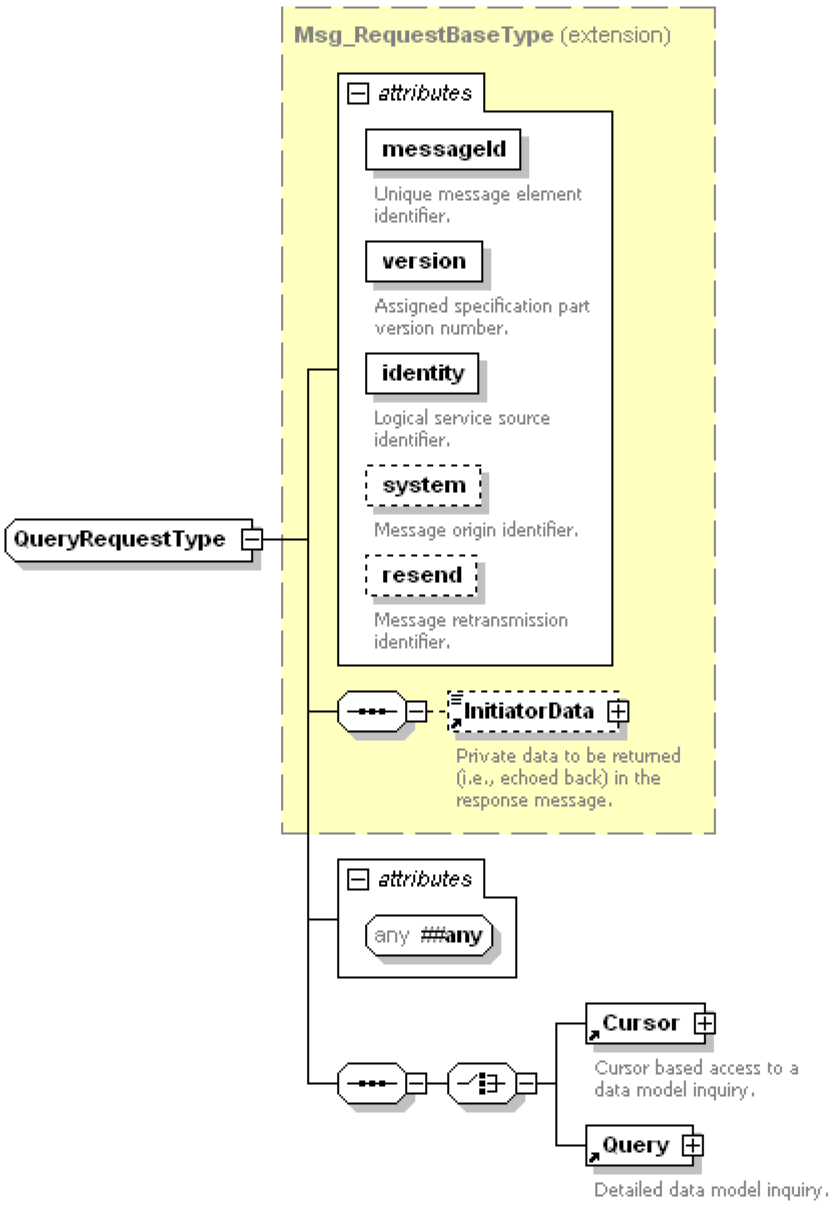

Msg_RequestBaseType (extension)
attributes
messageId
Unique message element identifier.
version
Assigned specification part version number.
identity
Logical service source identifier.
system
Message origin identifier.
resend
Message retransmission identifier.
QueryRequestType
InitiatorData
Private data 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 19 – 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 ITU-T J.380.8 interface to optionally extend concrete messages derived from ITU-T J.380.8 message types with attributes not specified by the schema.

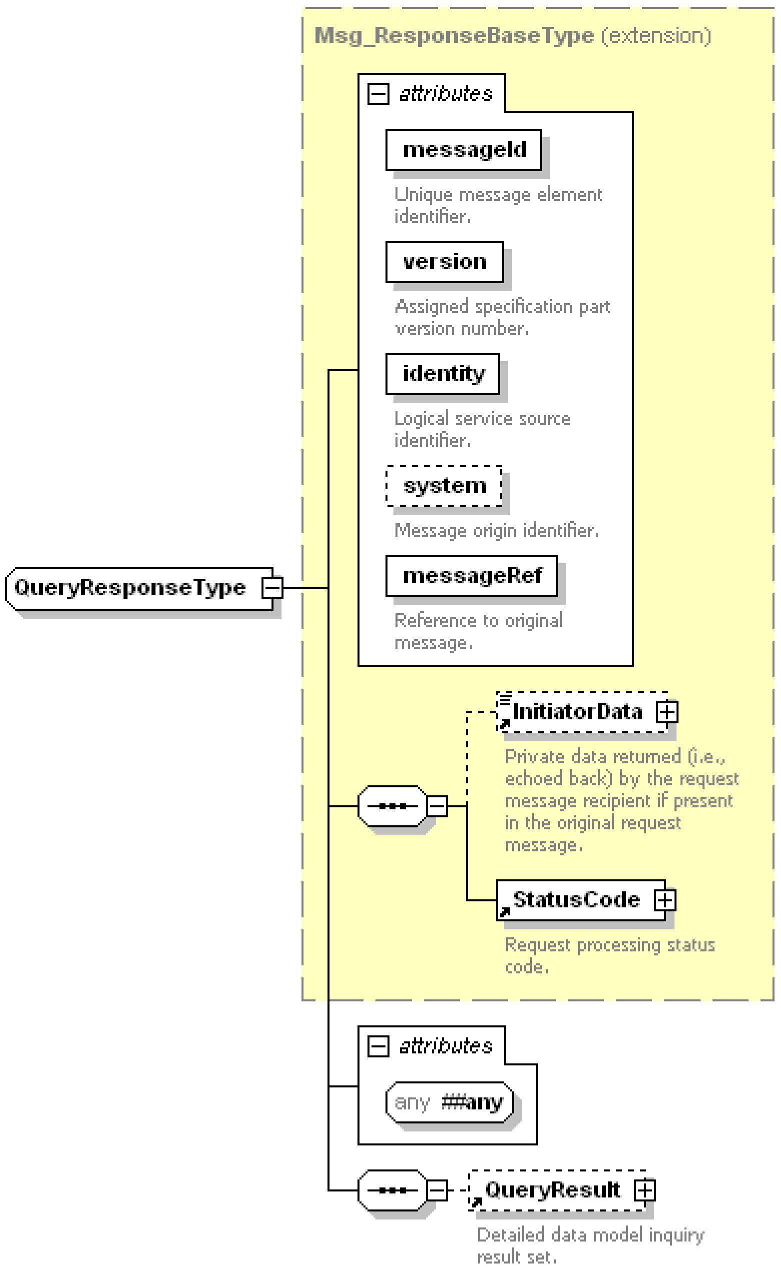
Cursor [Required on choice] – The Cursor element contains an @cursorRef attribute identifying a cursor previously established using the CreateCursorRequest message type. See clause 8.10 for details on the Cursor element.

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 clause 8.21 for additional information on the Query element.

7.8.2 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 20.



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 20 – 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 ITU-T J.380.8 interface to optionally extend concrete messages derived from ITU-T J.380.8 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 clause 8.22 for additional information on the QueryResult element.

7.9 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.

7.9.1 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 21.

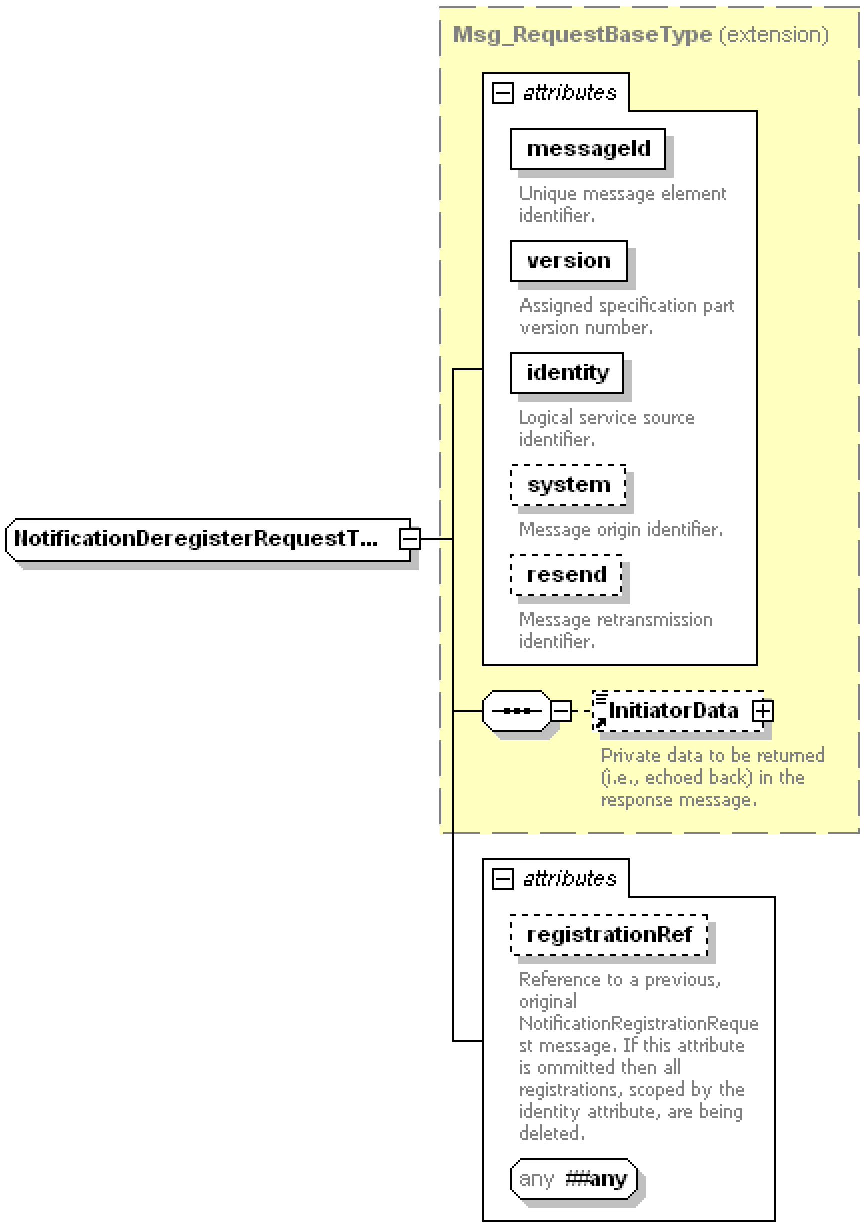

Msg_RequestBaseType (extension)
attributes
messageId
Unique message element identifier.
version
Assigned specification part version number.
identity
Logical service source identifier.
system
Message origin identifier.
resend
Message retransmission identifier.
NotificationDeregisterRequestT...
InitiatorData
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 21 – 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 ITU-T J.380.8 interface to optionally extend concrete messages derived from ITU-T J.380.8 message types with attributes not specified by the schema.

7.9.2 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 22.

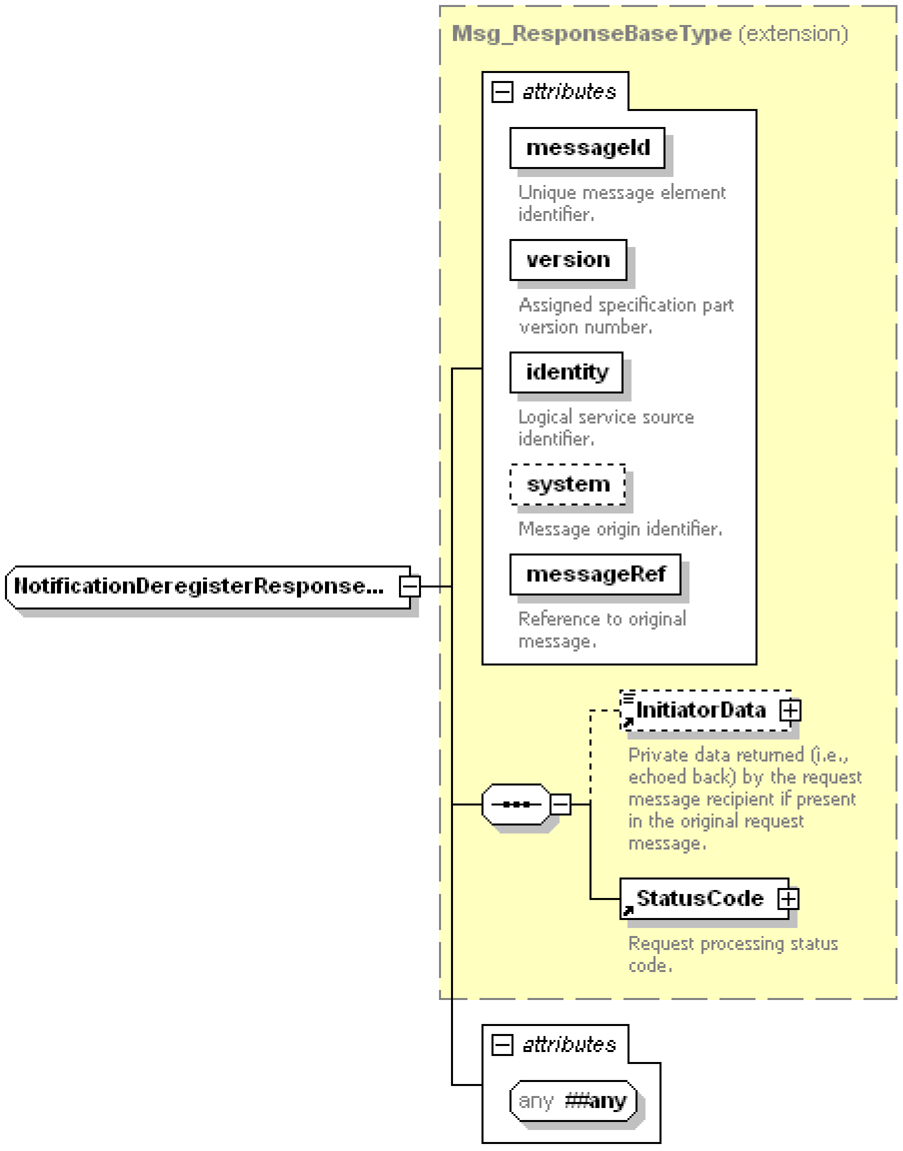

Msg_ResponseBaseType (extension)
attributes
messageId
Unique message element identifier.
version
Assigned specification part version number.
identity
Logical service source identifier.
system
Message origin identifier.
messageRef
Reference to original message.
NotificationDeregisterResponse...
InitiatorData
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 22 – 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 J.380.8 interface to optionally extend concrete messages derived from J.380.8 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).

7.10 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.

7.10.1 DeregistrationNotification 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 23.

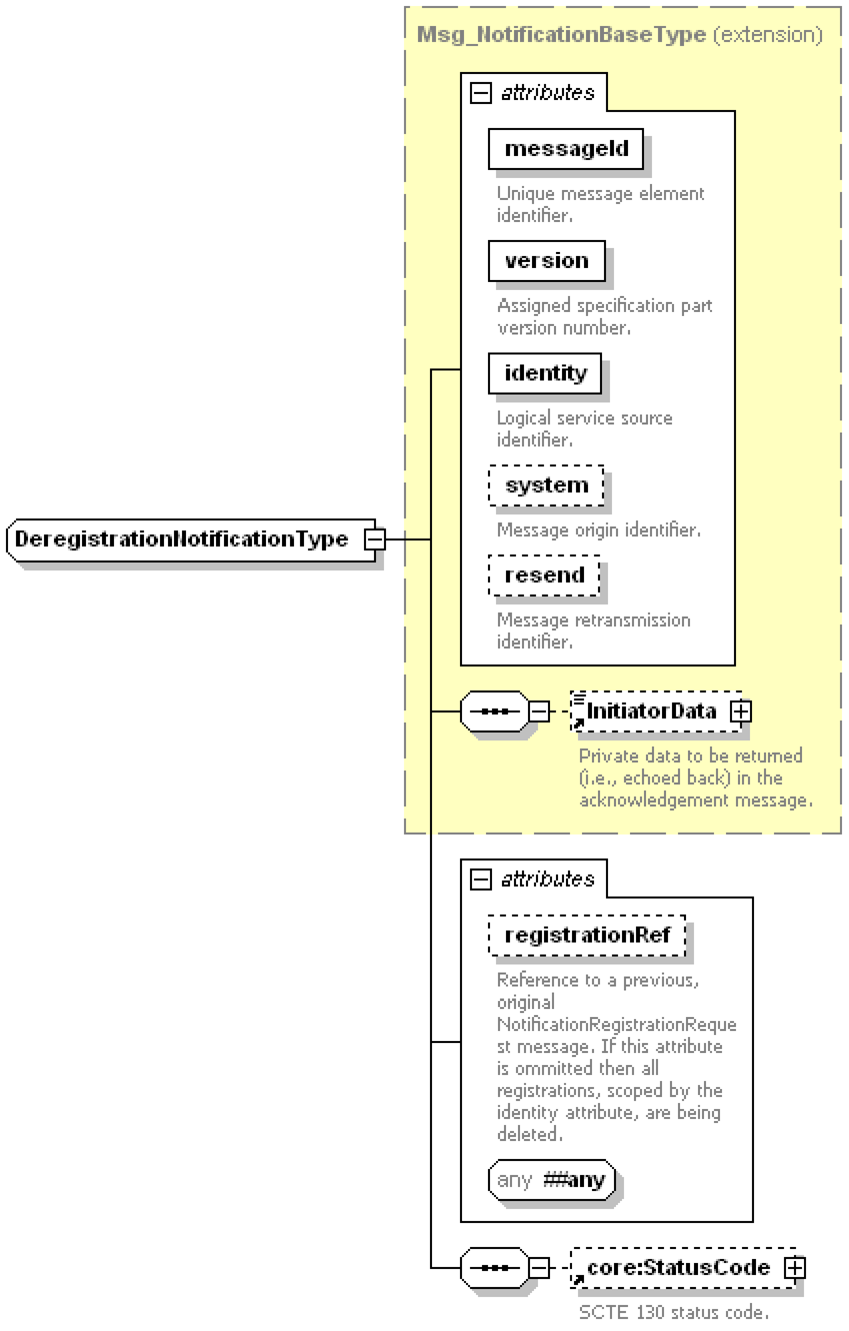

Msg_NotificationBaseType (extension)
attributes
messageId
Unique message element identifier.
version
Assigned specification part version number.
identity
Logical service source identifier.
system
Message origin identifier.
resend
Message retransmission identifier.
DeregistrationNotificationType
InitiatorData
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 23 – 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.

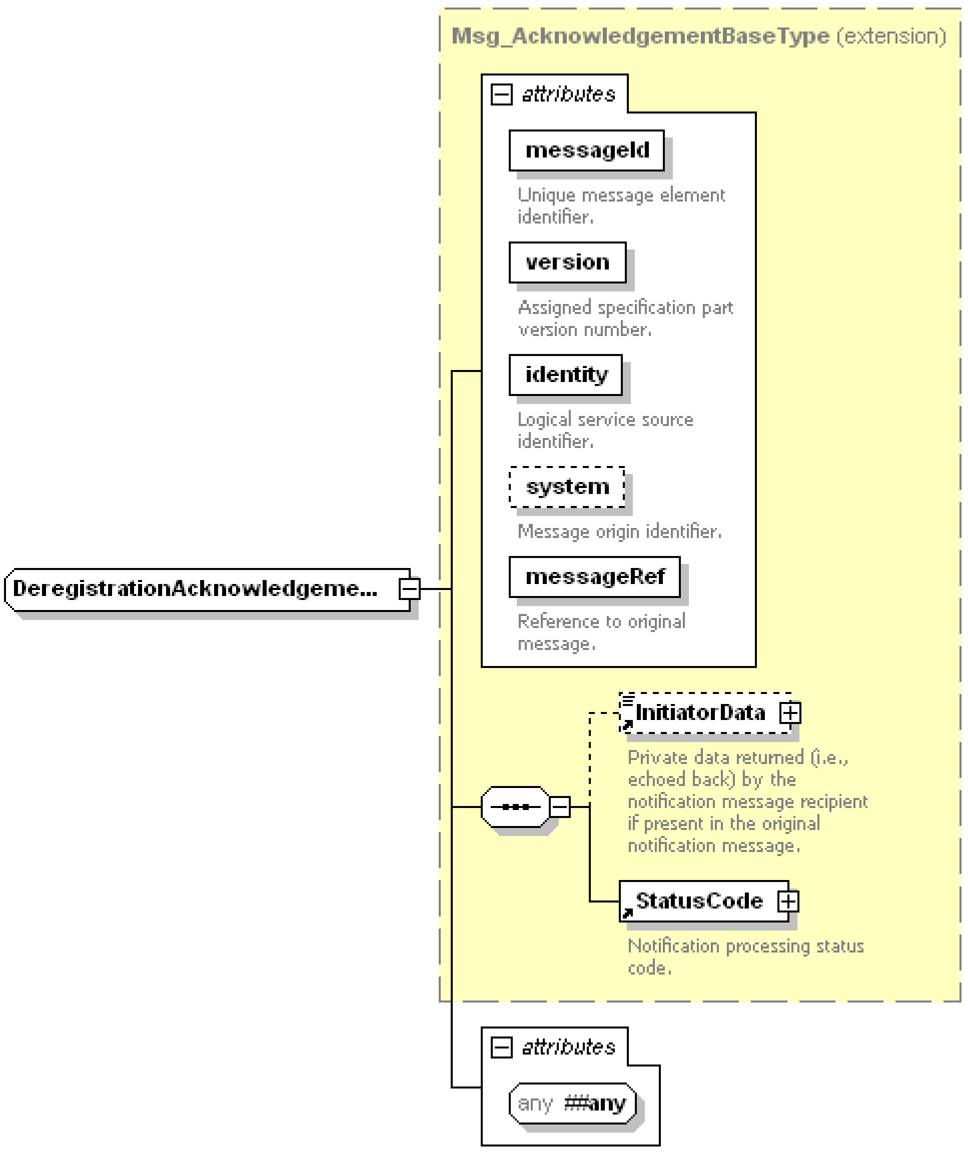
@registrationRef [Optional, core:registrationRefAttrType] – When present, this attribute identifies the original NotificationRegistrationRequest message type that shall be deregistered. The value shall be the NotificationRegistrationRequest message type's original @messageId attribute value. Issuing 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 ITU-T J.380.8 interface to optionally extend concrete messages derived from ITU-T J.380.8 message types with attributes not specified by the schema.

7.10.2 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 24.



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 24 – 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 ITU-T J.380.8 interface to optionally extend concrete messages derived from ITU-T J.380.8 message types with attributes not specified by the schema.

7.11 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 [ITU-T J.380.2].

7.12 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 [ITU-T J.380.2].

8 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 clauses.

Table 7 – GIS elementary element details

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
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 a 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

Table 7 – GIS elementary element details

Element	Description
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
UniqueQualifierDeclaration	Container specifying the QualifierDeclaration element sequence forming the basis for a unique qualifier

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 [W3C-XPath] 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).

Table 8 – Advanced query languages

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

8.1 AdvancedFilterElement

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

The XML schema diagram for this message is as follows in Figure 25.

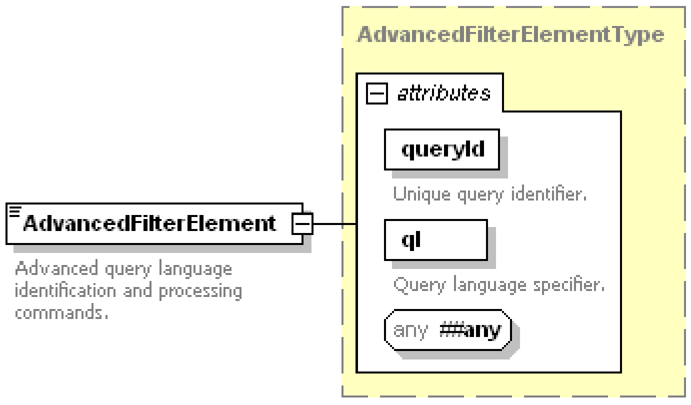

AdvancedFilterElementType
attributes
queryId
Unique query identifier.
ql
Query language specifier.
any ##any
AdvancedFilterElement
Advanced query language identification and processing commands.

Figure 25 – 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 clause 9.13 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 clause 9.15 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.

8.2 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 behaviour is controlled by the value of the @op attribute.

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

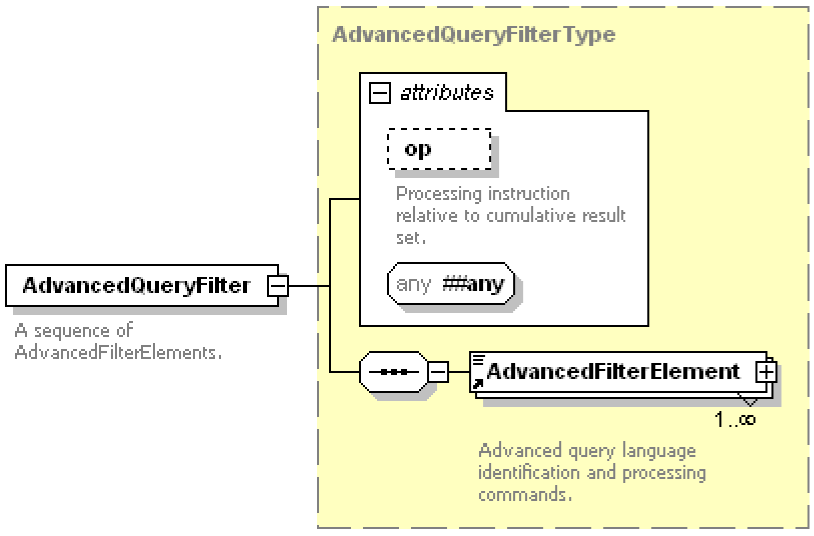

AdvancedQueryFilterType
attributes
op
Processing instruction relative to cumulative result set.
any ##any
AdvancedQueryFilter
A sequence of AdvancedFilterElements.
AdvancedFilterElement
1..∞
Advanced query language identification and processing commands.

Figure 26 – 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 clause 9.12 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 clause 8.1 for additional information.

8.3 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 27.

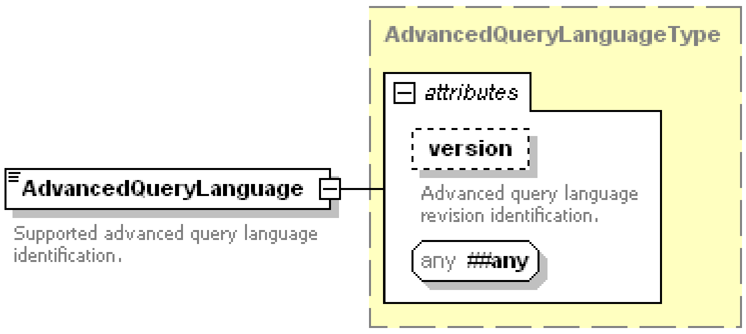

AdvancedQueryLanguageType
attributes
version
Advanced query language revision identification.
any ##any
AdvancedQueryLanguage
Supported advanced query language identification.

Figure 27 – 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.

8.4 AdvancedQueryResult

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

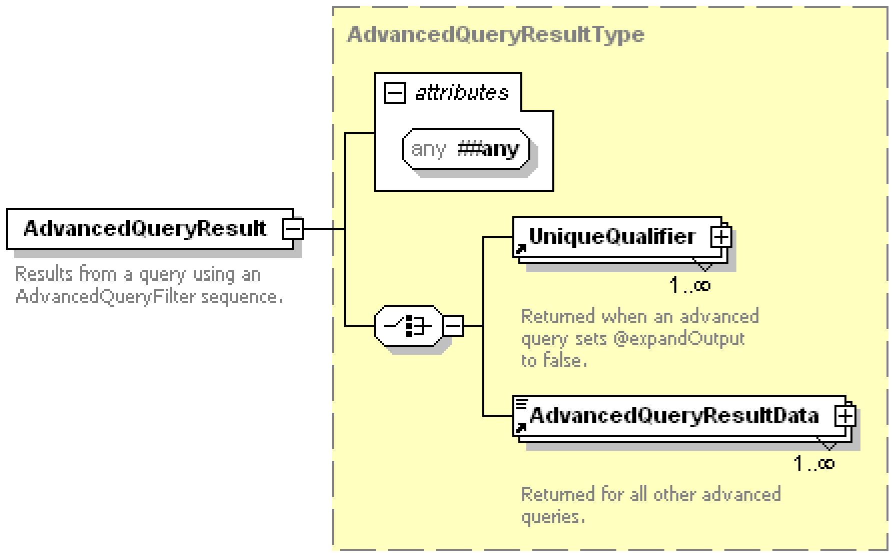

AdvancedQueryResultType
attributes
any ##any
AdvancedQueryResult
Results from a query using an AdvancedQueryFilter sequence.
UniqueQualifier
1..∞
Returned when an advanced query sets @expandOutput to false.
AdvancedQueryResultData
1..∞
Returned for all other advanced queries.

Figure 28 – 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). The AdvancedQueryResultData element shall contain a CDATA encapsulated result.

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 Recommendation.

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

The AdvancedQueryResult element contains the following attributes and elements.

UniqueQualifier [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 clause 8.25.

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 clause 8.5.

8.5 AdvancedQueryResultData

The AdvancedQueryResultData element contains CDATA encapsulated advanced query results.

The XML schema definition for the AdvancedQueryResultData element is shown in Figure 29.

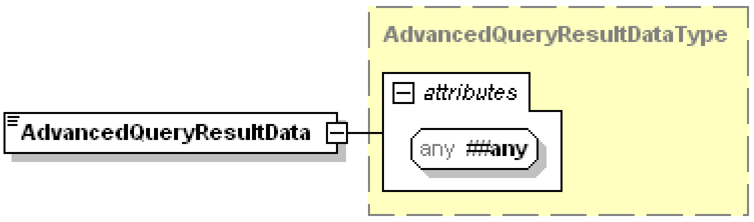


Figure 29 – AdvancedQueryResultData element XML schema

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

The AdvancedQueryResultData element's value shall contain CDATA encapsulated advanced query results.

8.6 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 clause 10.1.

The XML schema diagram for the BasicFilterElement is shown in Figure 30.

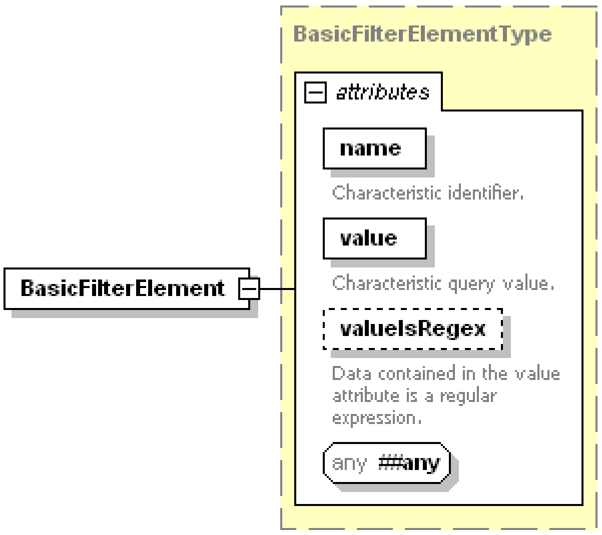

BasicFilterElementType
attributes
name
Characteristic identifier.
value
Characteristic query value.
valueIsRegex
Data contained in the value attribute is a regular expression.
any ##any
BasicFilterElement

Figure 30 – 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 clause 9.4 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 clause 10.1. See clause 9.5 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.

8.7 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 31.

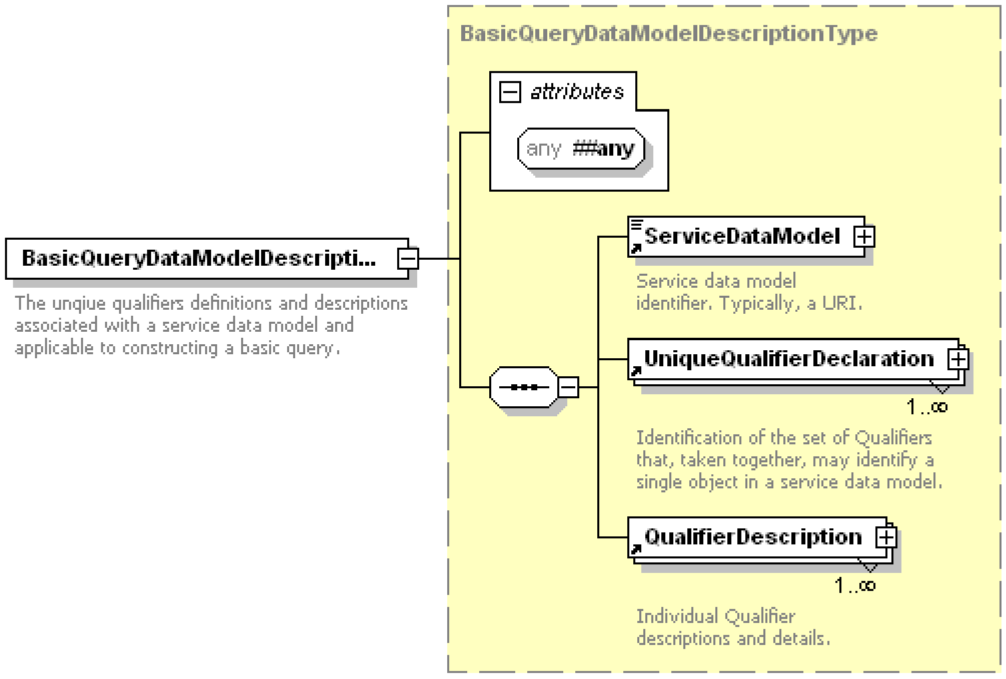

BasicQueryDataModelDescriptionType
attributes
any ##any
BasicQueryDataModelDescripti...
The unqiue qualifiers definitions and descriptions associated with a service data model and applicable to constructing a basic query.
ServiceDataModel
Service data model identifier. Typically, a URI.
UniqueQualifierDeclaration
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 31 – 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 clause 8.23.

UniqueQualifierDeclaration [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 UniqueQualifierDeclaration 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 clause 8.26.

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 clause 8.19.

8.8 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 behaviour is controlled by the value of the @op attribute.

The XML schema element for the BasicQueryFilter element is shown in Figure 32.

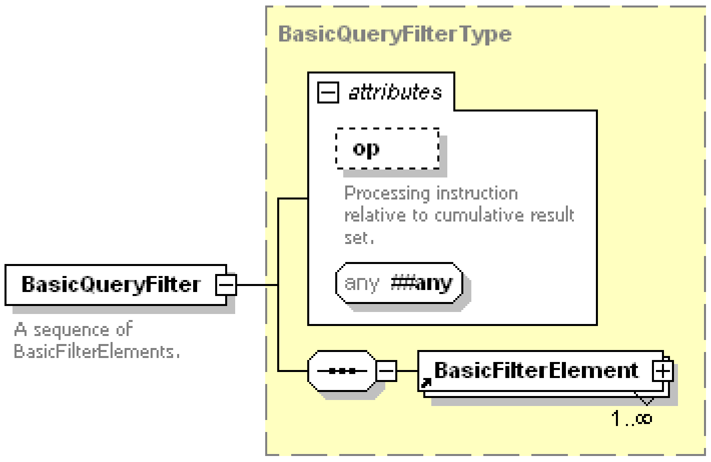

BasicQueryFilterType
attributes
op
Processing instruction relative to cumulative result set.
any ##any
BasicQueryFilter
A sequence of BasicFilterElements.
BasicFilterElement
1..∞

Figure 32 – 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.

8.9 BasicQueryResult

The BasicQueryResult element is 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. The XML schema for the BasicQueryResult is shown in Figure 33.

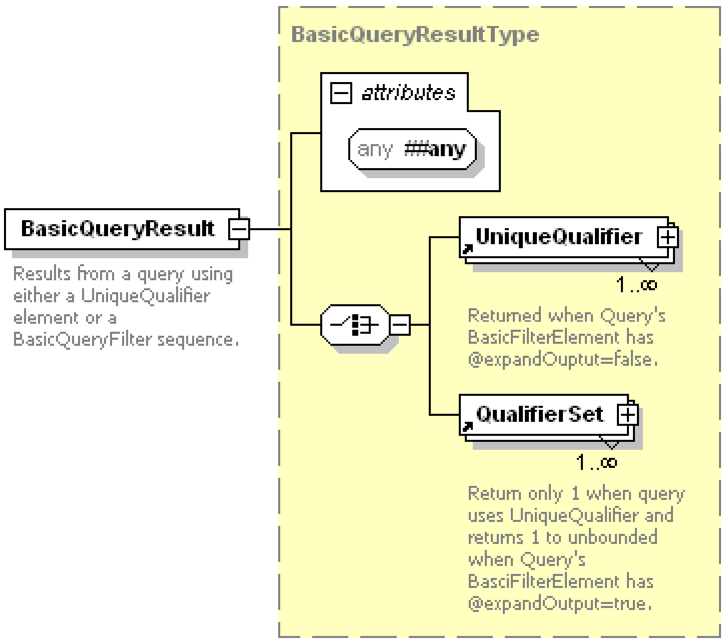

BasicQueryResultType
attributes
any ##any
BasicQueryResult
Results from a query using either a UniqueQualifier element or a BasicQueryFilter sequence.
UniqueQualifier
1..∞
Returned when Query's BasicFilterElement has @expandOuptut=false.
QualifierSet
1..∞
Return only 1 when query uses UniqueQualifier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

Figure 33 – 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 UniqueQualifier 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 UniqueQualifier element, only one QualifierSet 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 QualifierSet 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 QualifierSet element see clause 8.20. For additional details regarding the inquiry result return see the QueryResult element in clause 8.22.

8.10 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.

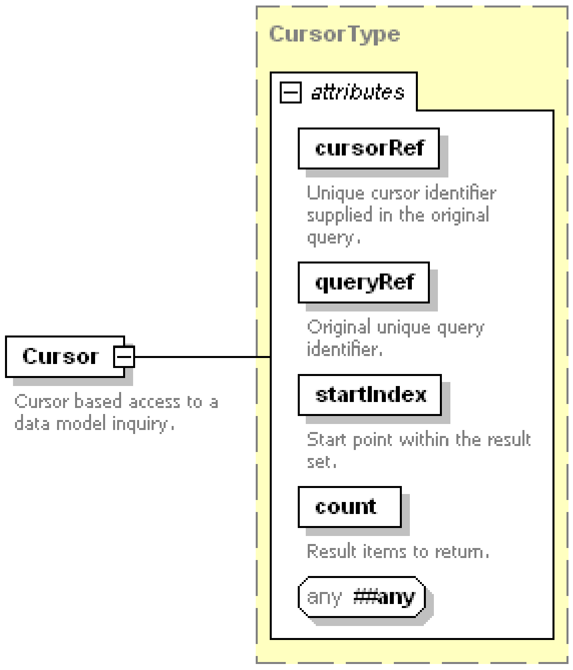

CursorType
attributes
cursorRef
Unique cursor identifier supplied in the original query.
queryRef
Original unique query identifier.
startIndex
Start point within the result set.
count
Result items to return.
any ##any
Cursor
Cursor based access to a data model inquiry.

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 an @cursorId value of '123' for a new cursor, at cancellation time for cursorId '123', the CancelCursorRequest message type shall include an @cursorRef with the value '123'. See clause 9.2 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 clause 9.13 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.

8.11 EnumerationValue

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

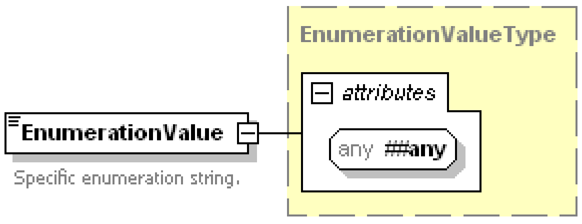


Figure 35 – EnumerationValue element XML schema

##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".

8.12 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.

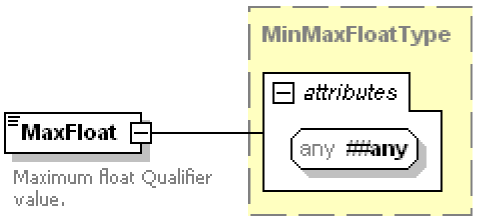


Figure 36 – MaxFloat element XML schema

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

The MaxFloat element's value is of type MinMaxFloatType. See clause 9.5 for additional information

8.13 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.

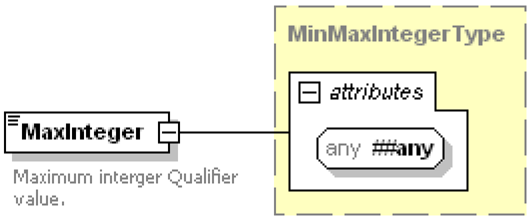


Figure 37 – MaxInteger element Xml schema

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

The MaxInteger element's value is of type MinMaxIntegerType. See clause 9.7 for additional information.

8.14 MaxLength

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

Figure 38 illustrates the MaxLength element's schema.

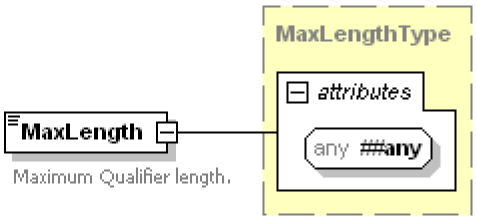


Figure 38 – MaxLength element XML schema

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

The MaxLength element's value is of type MaxLengthType. See clause 9.8 for additional information.

8.15 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.

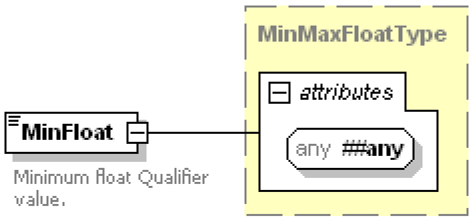


Figure 39 – MinFloat element XML schema

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

The MinFloat element's value is of type MinMaxFloatType. See clause 9.5 for additional information.

8.16 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.

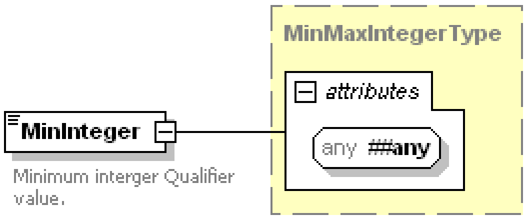


Figure 40 – MinInteger element XML schema

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

The MinInteger element's value is of type MinMaxIntegerType. See clause 9.7 for additional information.

8.17 Qualifier

The Qualifier element (Figure 41) 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 clause 8.25 for additional information on the UniqueQualifier element.

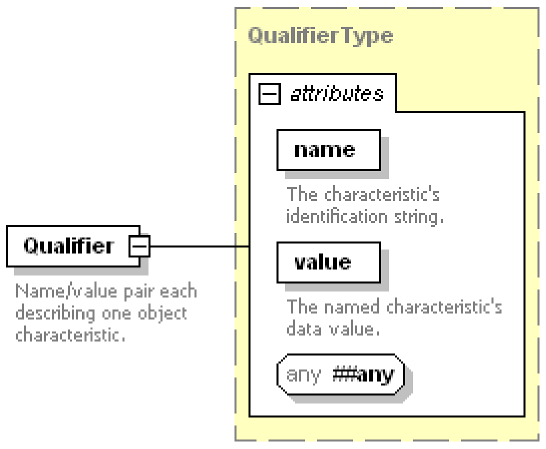

QualifierType
attributes
name
The characteristic's identification string.
value
The named characteristic's data value.
any ##any
Qualifier
Name/value pair each describing one object characteristic.

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 clause7.2 for additional information on the list qualifiers message type exchange and see clause 9.10 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 clause 7.2 for additional information on the list qualifiers message type exchange and see clause 8.19 for information regarding the QualifierDescription element.

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

The Qualifier element's value shall not be used.

8.18 QualifierDeclaration

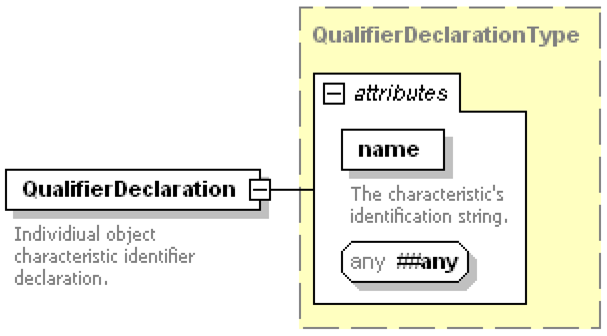

QualifierDeclarationType
attributes
name
The characteristic's identification string.
any ##any
QualifierDeclaration
Individiual object characteristic identifier declaration.

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 clause 9.10 for additional details.

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

The QualifierDeclaration element's value shall not be used (Figure 42).

8.19 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.

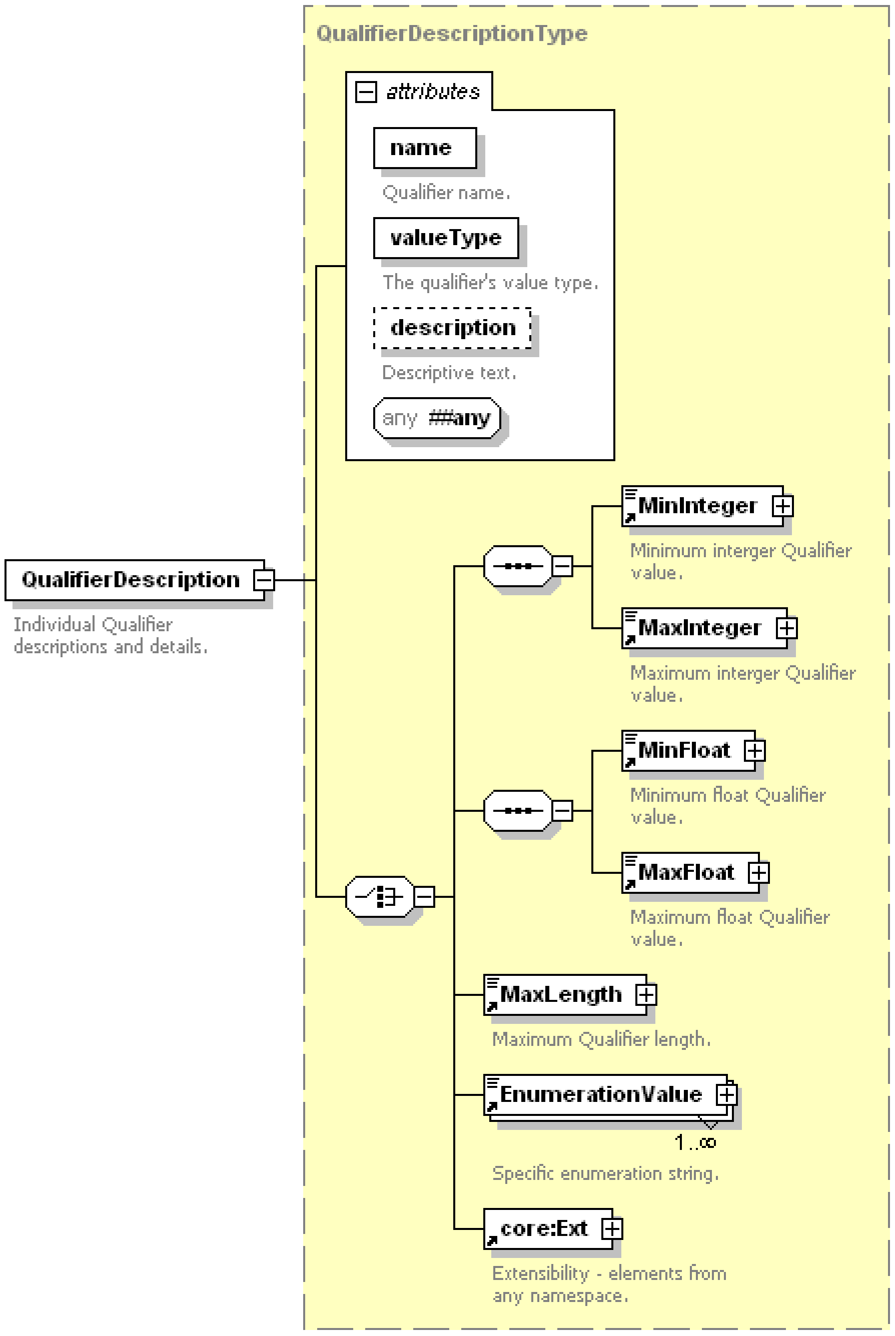

QualifierDescriptionType
attributes
name
Qualifier name.
valueType
The qualifier's value type.
description
Descriptive text.
any ##any
QualifierDescription
Individual Qualifier descriptions and details.
MinInteger
Minimum interger Qualifier value.
MaxInteger
Maximum interger Qualifier value.
MinFloat
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 clause 9.10 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 clause 9.11 for additional information.

Table 9 – Allowed QualifierDescription element @valueType values

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

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

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

MinInteger [Required – 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 clause 8.16 for additional information.

MaxInteger [Required – 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 clause 8.13 for additional information.

MinFloat [Required – 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 clause 8.15 for additional information.

MaxFloat [Required – 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 clause8.12 for additional information.

MaxLength [Required – 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 clause 8.14 for additional information.

EnumerationValue [Required – 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 [ITU-T J.380.2] for additional information.

8.20 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.

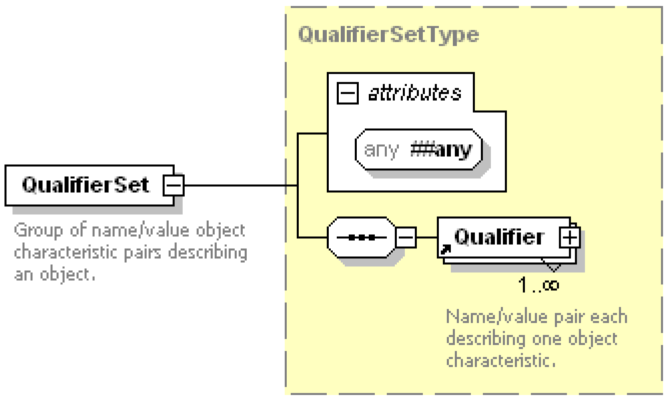

QualifierSetType
attributes
any ##any
QualifierSet
Group of name/value object characteristic pairs describing an object.
Qualifier
1..∞
Name/value pair each describing one object characteristic.

Figure 44 – QualifierSet 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 clause 8.17.

8.21 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.

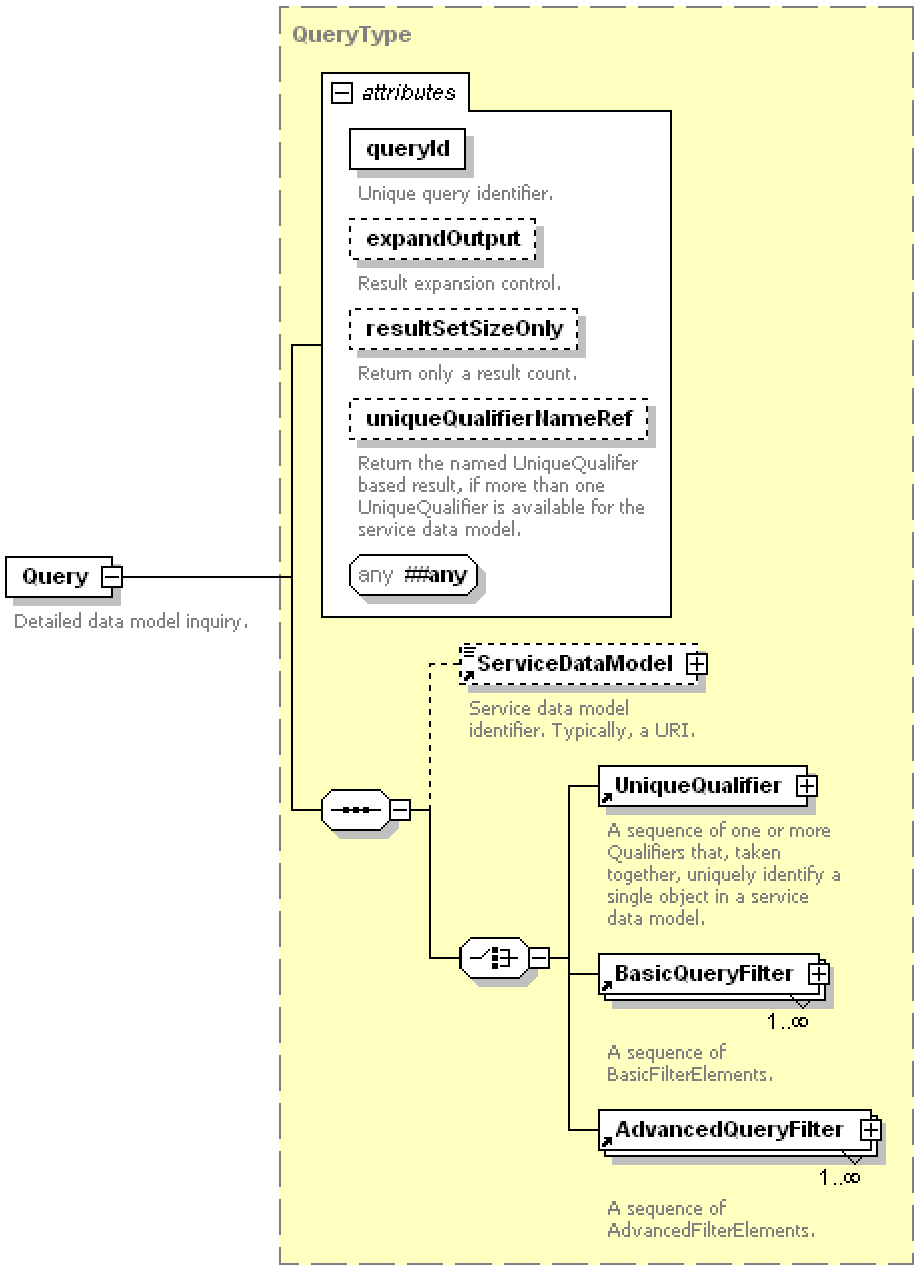

QueryType
attributes
queryId
Unique query identifier.
expandOutput
Result expansion control.
resultSetSizeOnly
Return only a result count.
uniqueQualifierNameRef
Return the named UniqueQualifer based result, if more than one UniqueQualifier is available for the service data model.
any ##any
Query
Detailed 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 clause 9.13 for further details on the queryIdAttrType.

@expandOutput [Optional, expandOutputAttrType] – The @expandOutput attribute controls the results output with the exact output behaviour being dependent on the input elements. See the QueryResult element in clause 8.22 for precise output details. If the @expandout Ouput attribute is omitted, the default value is "false." See clause 9.3 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 clause 9.16 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 clause 8.22 for applicability details. See clause 8.26 for information on the UniqueQualifierDeclaration element and see clause 9.18 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 clause 8.23 for further information on the ServiceDataModel element.

UniqueQualifier [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 clause 8.23 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 clause 8.8 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 clause 8.2 for further information on the AdvancedQueryFilter.

8.22 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.

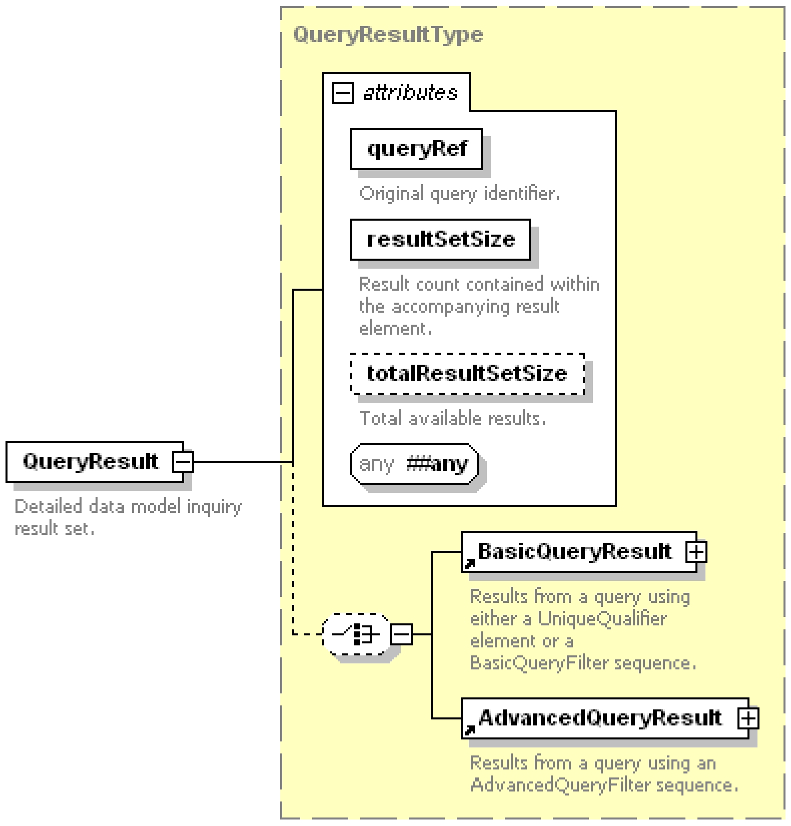

QueryResultType
attributes
queryRef
Original query identifier.
resultSetSize
Result count contained within the accompanying result element.
totalResultSetSize
Total available results.
any ##any
QueryResult
Detailed data model inquiry result set.
BasicQueryResult
Results from a query using either a UniqueQualifier 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 clause 9.13 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 QualifierSet elements in the result. If the Query element's @uniqueQualifierNameRef attribute was present, the attribute's total is specific to the @uniqueQualifierNameRef 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 QualifierSet 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 Recommendation.

See Table 10 for a detailed summary.

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

BasicQueryResult [Required under Choice] – The BasicQueryResult element is a container element for result data originating from inquires using the UniqueQualifier or BasicFilterElement elements. See Table 10 below for details regarding the BasicQueryResults usage. See clause 8.9 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 clause 8.4 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 behaviour not specifically covered within the table shall behave as previously defined.

Table 10 – QueryResult element truth table

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 @uniqueQualifierNameRef 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			

Table 10 – QueryResult element truth table

Query element					**QueryResult element**			
					(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 @uniqueQualifierNameRef identified unique qualifier

Legend:

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

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 Recommendation.

NOTE 4 – If non-zero, the mandatory value shall be the number of AdvancedQueryResultData elements are contained in the AdvancedQueryResult element.

8.23 ServiceDataModel

The ServiceDataModel element contains a non-empty "string" uniquely identifying a service data model supported by a logical service implementing the GIS interface. See Figure 47.

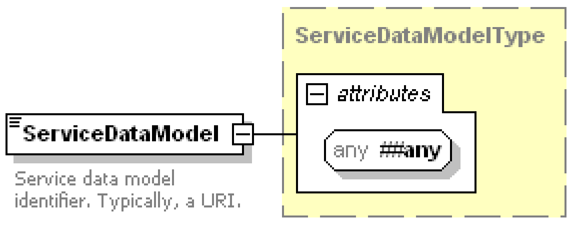


Figure 47 – ServiceDataModel element XML schema

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

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

8.24 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. See Figure 48.

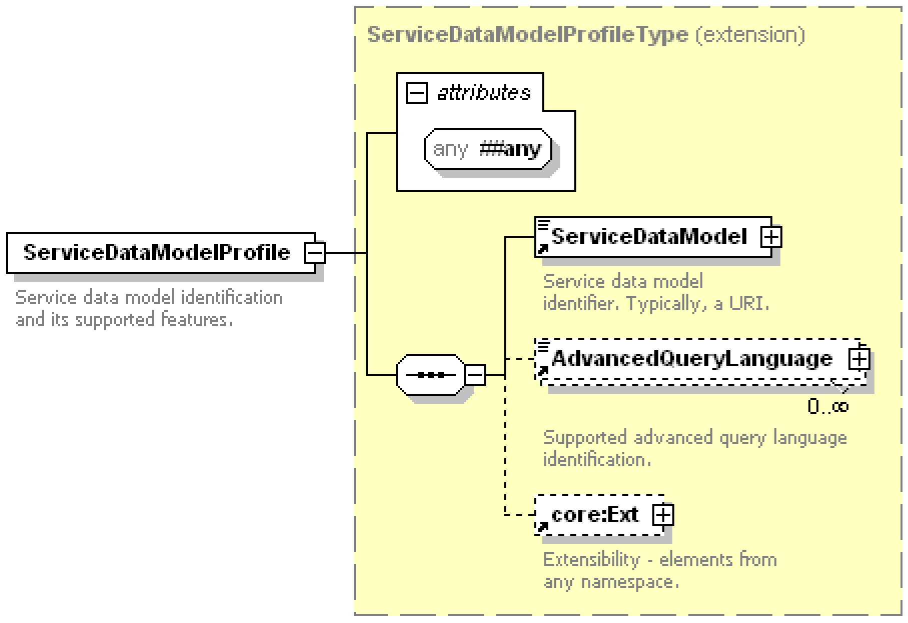

ServiceDataModelProfileType (extension)
attributes
any ##any
ServiceDataModelProfile
Service data model identification and its supported features.
ServiceDataModel
Service data model identifier. Typically, a URI.
AdvancedQueryLanguage
0..∞
Supported advanced query language identification.
core:Ext
Extensibility - 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 clause 8.23.

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 clause 8.3.

core:Ext [Optional] – Any additional elements from any namespace. For additional information on the core:Ext element see [ITU-T J.380.2].

8.25 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.

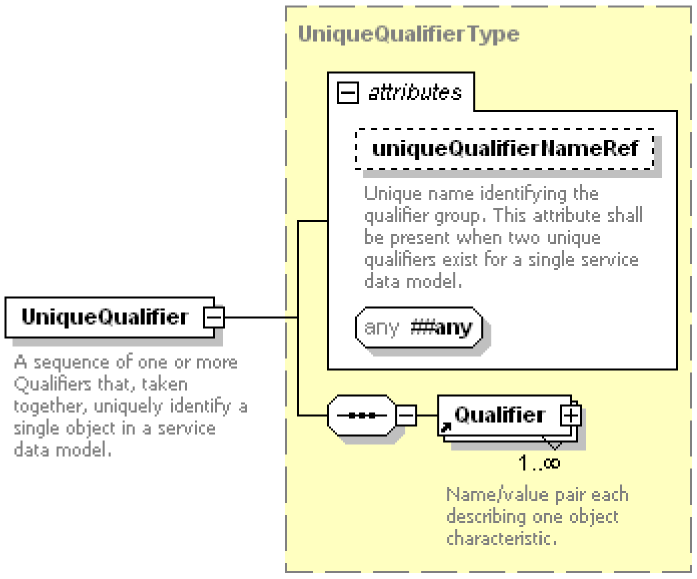

UniqueQualifierType
attributes
uniqueQualifierNameRef
Unique name identifying the qualifier group. This attribute shall be present when two unique qualifiers exist for a single service data model.
any ##any
UniqueQualifier
A sequence of one or more Qualifiers that, taken together, uniquely identify a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

Figure 49 – UniqueQualifier 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 UniqueQualifierDeclaration, each UniqueQualifier shall have an @uniqueQualifierNameRef attribute to distinguish 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 clause 9.18 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 clause 8.17.

8.26 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 QualifierDeclaration 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 an @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.

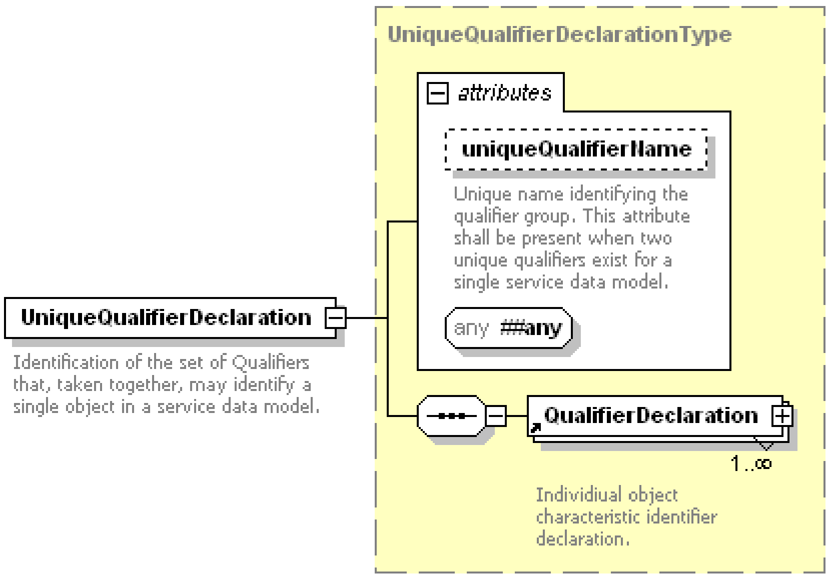

UniqueQualifierDeclarationType
attributes
uniqueQualifierName
Unique name identifying the qualifier group. This attribute shall be present when two unique qualifiers exist for a single service data model.
any ##any
UniqueQualifierDeclaration
Identification of the set of Qualifiers that, taken together,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

Figure 50 – UniqueQualifier element XML schema

@uniqueQualifierName [Optional, uniqueQualifierNameAttrType] – 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 clause 8.21 for more information on the use of the @uniqueQualifierName 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 clause 8.18.

9 ITU-T J.380.8 GIS attribute types

The following clauses define the GIS attribute types that are used throughout this document.

9.1 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.

9.2 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.

9.3 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 clause 8.21 and clause 8.22 for additional information.

9.4 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.

9.5 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 clause 10.1 for additional information.

9.6 MinMaxFloatType type

MinMaxFloatType [xsd:float] – This type, 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.

9.7 MinMaxIntegerType type

MinMaxIntegerType [xsd:integer] – This type, 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.

9.8 MaxLengthType type

MaxLengthType [xsd:nonNegativeInteger] – This type, 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.

9.9 NotificationTypeEnumeration type

NotificationTypeEnumeration [core:nonEmpytStringType] – This attribute type, typically referred to as the @noticeType attribute, contains one of the values from Table 6 in clause 7.5.1 The attribute specifies the data store change that the Notification message type is providing.

9.10 qualifierNameAttrType type

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

9.11 QualifierValueTypeEnumerationType

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.

9.12 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.

Table 11 – QueryFilterOpTypeEnumeration values

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.

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

9.13 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.

9.14 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.

9.15 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 ITU-T J.380.8 interface and if advanced queries may be executed against a service data model.

9.16 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.

9.17 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 clause 8.22 for additional information specific to the value's interpretation relative to the inquiry composition.

9.18 uniqueQualifierNameAttrType attribute type

uniqueQualifierNameAttrType [core:nonEmptyStringType] – This attribute type, used as the @uniqueQualifierName or uniqueQualifierNameRef 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.

10 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 an @name attribute and a complete or wild-carded @value attribute. For instance:

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

Figure 51 – Example 1-bis – BasicQueryFilter element

In Figure 51, 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.

10.1 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>
```

Figure 52 – Example 1-ter – BasicQueryFilter element

In Figure 52, 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.

Table 12 – GIS regular expressions

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

Table 13 – Regular expression examples

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

Annex A

Statuscode element @Detail attribute values

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

Table A.1 contains ITU-T J.380.8 applicable status codes for the @detail attribute.

Table A.1 – StatusCode details

@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 A.2 contains descriptive references to the ITU-T J.380.2 @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

Table A.2 – ITU-T J.380.2 and ITU-T J.380.8 StatusCode detail usage

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		✓								

Appendix I

Complex queries & expanded output examples

(This appendix does not form an integral part of this Recommendation.)

I.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>
```

Figure I.1 – Example 2 – BasicQueryFilter Element

In Figure I.1, 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 Figure I.1 are combined together to form a single query.

In Figure I.1, 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 Figure I.1 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 behaviour 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>
```

Figure I.2 – Example 3 – BasicQueryFilter element

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

In Figure I.3, 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>
```

Figure I.3 – Example 4 – BasicQueryFilter element

In Figure I.3 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".

I.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>
```

Figure I.4 – Example 5 – Query element

The query in Figure I.4 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 Figure I.4 follows in Figure I.5:

```
<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>
```

Figure I.5 – Example 6 – QueryResult element

Note that if the data model queried in Figure I.4 had a UniqueQualifier comprised of one Qualifier with name, "MACAddress", the result shown in Figure I.5 could have been obtained with a query using the UniqueQualifier rather than the BasicQueryFilter. This query is shown in Figure I.6.

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

Figure I.6 – Example 7 – Query element using UniqueQualifier

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>
```

Figure I.7 – Example 8 – Query element

The results of the query from Figure I.7 are illustrated below in Figure I.8.

```
  <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>
```

Figure I.8 – Example 9 – QueryResult element

Note that in Figure I.8, 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 II

Advanced queries

(This appendix does not form an integral part of this Recommendation.)

II.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 clause 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>
```

Figure II.1 – Example 10 – AdvancedQueryFilter element

In Figure II.1, the query language @ql is specified as XPath. The actual XPath expression is contained within the CDATA clause 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 Figure II.2:

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

Figure II.2 – Example 11 – QueryResult element

Appendix III

Cursor examples

(This appendix does not form an integral part of this Recommendation.)

III.1 Creating cursors

Cursors are created by using the CreateCursorRequest message. An example of this request is shown in Figure III.1. The "xxx" preceding each message name is done assuming a logical service named "xxx" has implemented the GIS interface and is described in Figure 3.

```
  <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>
```

Figure III.1 – Example 12 – CreateCursorRequest message

In FigureIII.1, the CreateCursorRequest message contains an @cursorId attribute and an @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 life time 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.

Figure III.2 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>
```

Figure III.2 – Example 13 – CreateCursorResponse message

Figure III.2 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.

III.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>
```

Figure III.3 – Example 14 – QueryRequest message

In Figure III.3 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 an @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 clause 7.8.1).

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

To continue to iterate over an existing cursor, the @startIndex attribute of the cursor must be modified. Figure III.4, illustrates a continuation of the cursor walk from Figure III.3.

```
  <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>
```

Figure III.4 – Example 15 – 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.

III.3 Cancelling existing cursors

Once a consumer has finished working with a cursor, the cursor may be cancelled before the duration time of the cursor has expired. Figure III.5 illustrates a complete CancelCursorRequest message.

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

Figure III.5 – CancelCursorRequest message

In Figure III.6, 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. Figure III.6 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>
```

Figure III.6 – Example 17 – 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 IV

Message examples

(This appendix does not form an integral part of this Recommendation.)

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

IV.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.

Figure IV.1 contains an example of a ListSupportedFeaturesRequest message.

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

Figure IV.1 – Example 18 – ListSupportedFeaturesRequest message

Figure IV.2 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>
```

Figure IV.2 – Example 19 – ListSupportedFeaturesResponse message

Figure IV.3 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>
```

Figure IV.3 – Example 20 – ListSupportedFeaturesResponse message

Figure IV.4 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>
```

Figure IV.4 – Example 21 – ListSupportedFeaturesResponse message

IV.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.

Figure IV.5 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>
```

Figure IV.5 – Example 22 – ListQualifiersRequest message

The result of the ListQualifiersRequest message is shown in Figure IV.6.

```
  <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>
```

Figure IV.6 – Example 23 – 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 Figure IV.6 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.

IV.3 Query examples

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

Figure IV.7 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>
```

Figure IV.7 – Example 24 – QueryRequest message

The result of this query is shown in Figure IV.8:

```
  <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>
```

Figure IV.8 – Example 25 – QueryResponse message

Figure IV.9 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>
```

Figure IV.9 – Example 26 – QueryRequest message

The result of this query is shown in Figure IV.10.

```
  <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>
```

Figure IV.10 – Example 27 – QueryResponse message

Figure IV.11 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>
```

Figure IV.11 – Example 28 – QueryRequest message

The result of this query is shown in Figure IV.12.

```
  <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>
```

Figure IV.12 – Example 29 – QueryResponse message

IV.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>
```

Figure IV.13 – Example 30 – NotificationRegistrationRequest message

Two things of interest in Figure IV.13 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 [ITU-T J.380.2] 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>
```

Figure IV.14 – Example 31 – NotificationRegistrationRequest message

In Figure IV.14, 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.

IV.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.

Figure IV.15 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>
```

Figure IV.15 – Example 32 – 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.

Bibliography

[b-ITU-T J.380.1] ITU-T J.380.1 (2011), *Digital program insertion – Advertising systems interfaces – Advertising systems overview*.

SERIES OF ITU-T RECOMMENDATIONS

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

Printed in Switzerland
Geneva, 2012