

BS EN ISO 19142:2010



BSI Standards Publication

Geographic information — Web Feature Service (ISO 19142:2010)

bsi.

...making excellence a habit.
AVA Documents™
Document Sharing Hub

National foreword

This British Standard is the UK implementation of EN ISO 19142:2010.

The UK participation in its preparation was entrusted to Technical Committee IST/36, Geographic information.

A list of organizations represented on this committee can be obtained on request to its secretary.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

© BSI 2011

ISBN 978 0 580 65191 5

ICS 35.240.70

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 31 January 2011.

Amendments issued since publication

Date	Text affected
------	---------------

English Version

Geographic information - Web Feature Service (ISO
19142:2010)

Information géographique - Services d'accès aux entités
géographiques par le web (ISO 19142:2010)

This European Standard was approved by CEN on 14 December 2010.

CEN members are bound to comply with the CEN/CENELEC Internal Regulations which stipulate the conditions for giving this European Standard the status of a national standard without any alteration. Up-to-date lists and bibliographical references concerning such national standards may be obtained on application to the CEN-CENELEC Management Centre or to any CEN member.

This European Standard exists in three official versions (English, French, German). A version in any other language made by translation under the responsibility of a CEN member into its own language and notified to the CEN-CENELEC Management Centre has the same status as the official versions.

CEN members are the national standards bodies of Austria, Belgium, Bulgaria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, Netherlands, Norway, Poland, Portugal, Romania, Slovakia, Slovenia, Spain, Sweden, Switzerland and United Kingdom.



EUROPEAN COMMITTEE FOR STANDARDIZATION
COMITÉ EUROPÉEN DE NORMALISATION
EUROPÄISCHES KOMITEE FÜR NORMUNG

Management Centre: Avenue Marnix 17, B-1000 Brussels

Foreword

This document (EN ISO 19142:2010) has been prepared by Technical Committee ISO/TC 211 “Geographic information/Geomatics” in collaboration with Technical Committee CEN/TC 287 “Geographic Information” the secretariat of which is held by BSI.

This European Standard shall be given the status of a national standard, either by publication of an identical text or by endorsement, at the latest by June 2011, and conflicting national standards shall be withdrawn at the latest by June 2011.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. CEN [and/or CENELEC] shall not be held responsible for identifying any or all such patent rights.

According to the CEN/CENELEC Internal Regulations, the national standards organizations of the following countries are bound to implement this European Standard: Austria, Belgium, Bulgaria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, Netherlands, Norway, Poland, Portugal, Romania, Slovakia, Slovenia, Spain, Sweden, Switzerland and the United Kingdom.

Endorsement notice

The text of ISO 19142:2010 has been approved by CEN as a EN ISO 19142:2010 without any modification.

Contents

Page

Foreword	xi
Introduction.....	xii
1 Scope	1
2 Conformance	2
3 Normative references	3
4 Terms and definitions	4
5 Conventions	8
5.1 Abbreviated terms	8
5.2 Use of examples	9
5.3 XML schemas	9
5.4 UML Notation	9
5.4.1 Class diagrams	9
5.4.2 State machine notation	10
6 Basic service elements	11
6.1 Introduction.....	11
6.2 Version numbering and negotiation	11
6.2.1 Version number form and value	11
6.2.2 Appearance in service metadata and in requests.....	11
6.2.3 Version number negotiation.....	11
6.2.4 Request encoding	11
6.2.5 KVP parameter encoding rules	12
6.3 Namespaces.....	13
6.4 Service bindings.....	13
7 Common elements	13
7.1 Encoding of features.....	13
7.2 Resource identifiers	13
7.2.1 Assigning resource identifiers.....	13
7.2.2 Encoding resource identifiers.....	14
7.2.3 Version identification	14
7.3 Property references.....	14
7.3.1 XPath subset.....	14
7.3.2 Accessor function	14
7.4 Predicate expression encoding	15
7.5 Exception reporting.....	15
7.6 Common request parameters	17
7.6.1 Introduction.....	17
7.6.2 Base request type.....	17
7.6.3 Standard presentation parameters.....	19
7.6.4 Standard resolve parameters	22
7.6.5 Standard input parameters.....	25
7.6.6 Additional common keywords for KVP-encoded requests.....	26
7.7 Standard response parameters	27
7.7.1 Parameter semantics	27
7.7.2 XML encoding.....	27
7.7.3 KVP encoding	27
7.7.4 Parameter discussion	27
7.8 Use of the schemaLocation attribute	30
7.9 Query expressions	30

7.9.1	Introduction	30
7.9.2	Ad hoc query expression	30
7.9.3	Stored query expression	40
8	GetCapabilities operation	42
8.1	Introduction	42
8.2	Request	43
8.2.1	Request semantics	43
8.2.2	XML encoding	43
8.2.3	KVP encoding	43
8.3	Response	43
8.3.1	Response semantics	43
8.3.2	XML encoding	44
8.3.3	Capabilities document	45
8.3.4	FeatureTypeList section	45
8.3.5	Parameters domains and constraints	48
8.4	Extension points	52
8.5	Exceptions	54
9	DescribeFeatureType operation	54
9.1	Introduction	54
9.2	Request	54
9.2.1	Request semantics	54
9.2.2	XML Encoding	54
9.2.3	KVP Encoding	55
9.2.4	Parameter discussion	55
9.3	Response	56
9.3.1	Introduction	56
9.3.2	Supporting multiple namespaces	56
9.4	Exceptions	57
10	GetPropertyValue operation	57
10.1	Introduction	57
10.2	Request	57
10.2.1	Request semantics	57
10.2.2	XML Encoding	58
10.2.3	KVP Encoding	58
10.2.4	Parameter discussion	58
10.3	Response	60
10.3.1	Response semantics	60
10.3.2	XML encoding	60
10.3.3	State parameter	61
10.3.4	Standard response parameters	61
10.4	Exceptions	61
11	GetFeature operation	62
11.1	Introduction	62
11.2	Request	62
11.2.1	Request semantics	62
11.2.2	XML encoding	63
11.2.3	KVP encoding	63
11.2.4	Parameter discussions	64
11.3	Response	64
11.3.1	Response semantics	64
11.3.2	XML encoding	65
11.3.3	Parameter discussions	66
11.3.4	Additional objects	69
11.3.5	GetFeatureById response	69
11.4	Exceptions	70
12	LockFeature operation	70
12.1	Introduction	70

12.2	Request.....	70
12.2.1	Request semantics.....	70
12.2.2	XML encoding.....	71
12.2.3	KVP encoding.....	71
12.2.4	Parameter discussions.....	72
12.2.5	State machine for WFS locking.....	73
12.3	Response.....	74
12.3.1	Response semantics.....	74
12.3.2	XML encoding.....	74
12.4	Exceptions.....	75
13	GetFeatureWithLock operation.....	75
13.1	Introduction.....	75
13.2	Request.....	75
13.2.1	Request semantics.....	75
13.2.2	XML encoding.....	75
13.2.3	KVP encoding.....	76
13.2.4	Parameter discussion.....	76
13.3	Response.....	77
13.3.1	Introduction.....	77
13.3.2	lockId parameter.....	77
13.4	Exceptions.....	77
14	Stored query management.....	77
14.1	Introduction.....	77
14.2	Defining stored queries.....	78
14.2.1	XML encoding.....	78
14.2.2	Parameter discussion.....	78
14.3	ListStoredQueries operation.....	81
14.3.1	Request semantics.....	81
14.3.2	XML encoding.....	82
14.3.3	KVP encoding.....	82
14.3.4	Response.....	82
14.3.5	Exceptions.....	83
14.4	DescribeStoredQueries operations.....	83
14.4.1	Request semantics.....	83
14.4.2	XML encoding.....	83
14.4.3	KVP encoding.....	84
14.4.4	Response.....	84
14.5	CreateStoredQuery operation.....	85
14.5.1	Request semantics.....	85
14.5.2	XML encoding.....	85
14.5.3	KVP encoding.....	85
14.5.4	Parameter discussions.....	85
14.5.5	Response.....	86
14.6	DropStoredQuery operations.....	86
14.6.1	Request semantics.....	86
14.6.2	XML encoding.....	87
14.6.3	KVP encoding.....	87
14.6.4	Response.....	87
14.7	Exceptions.....	87
15	Transaction operation.....	88
15.1	Introduction.....	88
15.2	Request.....	88
15.2.1	Request semantics.....	88
15.2.2	XML encoding.....	89
15.2.3	Parameter discussions.....	90
15.2.4	Insert action.....	91
15.2.5	Update action.....	92
15.2.6	Replace action.....	94

15.2.7	Delete action.....	94
15.2.8	Native action.....	95
15.3	Response.....	96
15.3.1	Response semantics	96
15.3.2	TransactionResponse element.....	96
15.3.3	TransactionSummary element	97
15.3.4	InsertResults element	97
15.3.5	UpdateResults element	98
15.3.6	ReplaceResults element	98
15.4	Exceptions.....	98
Annex A (normative) Conformance testing.....		99
A.1	Conformance classes.....	99
A.1.1	Simple WFS	99
A.1.2	Basic WFS	99
A.1.3	Transactional WFS	99
A.1.4	Locking WFS	100
A.1.5	HTTP GET	100
A.1.6	HTTP POST.....	100
A.1.7	SOAP.....	100
A.1.8	Inheritance.....	101
A.1.9	Remote resolve	101
A.1.10	Response paging.....	101
A.1.11	Standard joins	101
A.1.12	Spatial joins.....	101
A.1.13	Temporal joins	102
A.1.14	Feature versions	102
A.1.15	Manage stored queries.....	102
A.2	Basic tests	102
A.2.1	Version negotiation	102
A.2.2	Lists version number 2.0.0 as a supported request version number	103
A.2.3	Invalid version number	103
A.2.4	Version negotiation for the GetCapabilities request.....	103
A.2.5	Response to XML- and KVP-encoded. requests	103
A.2.6	Parameter ordering and case	104
A.2.7	Unrecognized parameters.....	104
A.2.8	Server operates on GML features	104
A.2.9	Feature identifiers.....	105
A.2.10	Invariant identifier.....	105
A.2.11	Versioning	105
A.2.12	XPath subset	106
A.2.13	Predicate encoding.....	106
A.2.14	Exception reporting.....	106
A.2.15	Common request parameters.....	107
A.2.16	Standard presentation parameters	108
A.2.17	Standard resolve parameters	109
A.2.18	Standard input parameters	112
A.2.19	Standard response parameters.....	113
A.2.20	Response paging.....	114
A.2.21	schemaLocation parameter	115
A.2.22	Query expressions	115
A.2.23	Declaring conformance.....	120
Annex B (informative) Examples		121
B.1	Exception report example.....	121
B.2	DescribeFeatureType examples.....	121
B.2.1	Example 1	121
B.2.2	Example 2	124

B.3	GetFeature examples	128
B.3.1	Introduction.....	128
B.3.2	Example 1.....	128
B.3.3	Example 2.....	128
B.3.4	Example 3.....	129
B.3.5	Example 4.....	129
B.3.6	Example 5.....	131
B.3.7	Example 6.....	131
B.3.8	Example 7.....	131
B.3.9	Example 8.....	132
B.3.10	Example 9.....	134
B.3.11	Example 10.....	136
B.3.12	Example 11.....	137
B.3.13	Example 12.....	138
B.3.14	Example 13.....	139
B.3.15	Example 14.....	142
B.3.16	Example 15.....	143
B.3.17	Example 16.....	143
B.3.18	Example 17.....	144
B.3.19	Example 18.....	145
B.3.20	Example 19.....	145
B.4	GetPropertyValue examples.....	146
B.4.1	Introduction.....	146
B.4.2	Example 1.....	147
B.4.3	Example 2.....	149
B.4.4	Example 3.....	149
B.4.5	Example 4.....	151
B.4.6	Example 5.....	152
B.4.7	Example 6.....	153
B.4.8	Example 7.....	154
B.4.9	Example 8.....	154
B.4.10	Example 9.....	155
B.5	LockFeature examples.....	156
B.5.1	Example 1.....	156
B.5.2	Example 2.....	157
B.5.3	Example 3.....	157
B.5.4	Example 4.....	158
B.6	Transaction examples.....	159
B.6.1	Insert example	159
B.6.2	Update examples	160
B.6.3	Delete examples	162
B.6.4	Mixed transaction example	163
B.6.5	Transaction response example.....	166
B.7	GetCapabilities example.....	167
B.8	KVP examples.....	182
B.8.1	Conventions.....	182
B.8.2	DescribeFeatureType examples	182
B.8.3	GetPropertyValue examples.....	183
B.8.4	GetFeature examples	185
B.8.5	LockFeature examples.....	190
Annex C (informative)	Consolidated XML schema.....	192
C.1	Introduction.....	192
C.2	wfs.xsd.....	192
Annex D (normative)	Service bindings.....	203
D.1	Introduction.....	203

D.2	HTTP GET and POST binding.....	203
D.3	HTTP status codes	203
D.4	SOAP binding.....	204
D.4.1	Introduction	204
D.4.2	SOAP Envelope.....	205
D.4.3	SOAP Header.....	205
D.4.4	SOAP Body.....	205
D.4.5	Encoding XML Schema in a SOAP Body	206
D.4.6	SOAP Fault	207
D.4.7	SOAP HTTP Binding.....	208
Annex E	(normative) Web Service Description Language (WSDL).....	209
E.1	Introduction	209
E.2	WFS Operations in WSDL	209
E.3	SOAP Binding	209
E.4	Binding style	210
E.5	Service	211
E.6	Service description using WSDL	211
E.6.1	Introduction	211
E.6.2	wfs-xml-interfaces.wsdl	211
E.6.3	wfs-kvp-interfaces.wsdl	213
E.6.4	wfs-responses.wsdl.....	215
E.6.5	wfs-http-bindings.wsdl.....	215
E.6.6	wfs-kvp-bindings.wsdl	218
E.6.7	wfs-soap-bindings.wsdl	219
E.6.8	Ancillary files.....	221
E.6.9	Examples (informative)	226
Annex F	(informative) Abstract model	229
F.1	Overview	229
F.2	Abstract Resource Model	229
F.2.1	Introduction	229
F.2.2	Basic Accessor Functions.....	229
F.3	Mapping of the General Feature Model (GFM) to the WFS Abstract Model.....	231
F.4	Identifiers.....	231
F.5	valueOf() function	231
F.6	WFS Operations	231
F.6.1	Introduction	231
F.6.2	featureTypeNameList() function.....	232
F.6.3	featureType() function.....	232
F.6.4	Query function	232
F.6.5	propertyValue() function	233
F.6.6	lock() function	234
F.6.7	transaction() function.....	234
F.6.8	Stored query operations	235
F.7	WFS Operations	236
F.8	Conceptual schema	236
Bibliography		238

Figures

Figure 1 — UML notation in class diagrams	9
Figure 2 — Summary of UML state diagram notations	10
Figure 3 — BaseRequest	17
Figure 4 — StandardPresentationParameters	19
Figure 5 — StandardResolveParameters	22
Figure 6 — StandardInputParameters	25
Figure 7 — StandardResponseParameters	27
Figure 8 — Ad hoc query expression	31
Figure 9 — Query projection clause	35
Figure 10 — Query sorting clause	39
Figure 11 — StoredQuery	41
Figure 12 — GetCapabilities request	43
Figure 13 — GetCapabilities response	44
Figure 14 — DescribeFeatureType request	54
Figure 15 — GetPropertyValue request	57
Figure 16 — GetPropertyValue response	60
Figure 17 — GetFeature request	63
Figure 18 — GetFeature response	65
Figure 19 — LockFeature request	70
Figure 20 — State diagram for a WFS lock	73
Figure 21 — LockFeature response	74
Figure 22 — GetFeatureWithLock request	75
Figure 23 — ListStoredQueries request	81
Figure 24 — ListStoredQueriesResponse	82
Figure 25 — DescribeStoredQueries request	83
Figure 26 — DescribeStoredQueriesResponse	84
Figure 27 — CreateStoredQuery request	85
Figure 28 — CreateStoredQuery response	86
Figure 29 — DropStoredQuery request	86

Figure 30 — Transaction request	89
Figure 31 — Transaction response	96
Figure F.1 — Web Feature Service interfaces overview	237

Tables

Table 1 — Conformance Classes	2
Table 2 — Operation request encoding	12
Table 3 — WFS exception codes	16
Table 4 — KVP encoding of the base request type	18
Table 5 — KVP encoding of standard presentation parameters	19
Table 6 — KVP encoding of standard resolve parameters	23
Table 7 — Additional common keywords for KVP-encoded. WFS requests	26
Table 8 — Keywords for Ad hoc query KVP encoding	32
Table 9 — KVP encoding of projection clause	35
Table 10 — Keywords for Stored query KVP encoding	41
Table 11 — Elements to describe feature types	47
Table 12 — Parameter domains for WFS operations	48
Table 13 — Service constraints	50
Table 14 — Operation Constraints	51
Table 15 — DescribeFeatureType KVP encoding	55
Table 16 — Keywords for GetPropertyValue KVP encoding	58
Table 17 — Keywords for GetFeature KVP encoding	64
Table 18 — Keywords for LockFeature KVP encoding	71
Table 19 — Additional keywords for GetFeatureWithLock KVP encoding	76
Table 20 — Keywords for ListStoredQueries KVP encoding	82
Table 21 — Keywords for DescribeStoredQueries KVP encoding	84
Table 22 — Keywords for DropStoredQuery KVP encoding	87
Table D.1 — Request encoding and transport methods	203
Table D.2 — Correlate OWS and WFS exception codes to HTTP status codes	204
Table F.1 — Mapping the WFS Abstract Model operations to WFS operations	236

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of technical committees is to prepare International Standards. Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

ISO 19142 was prepared by Technical Committee ISO/TC 211, *Geographic Information/Geomatics*, in collaboration with the Open Geospatial Consortium, Inc. (OGC). The Web Feature Service (WFS) was originated within OGC.

Introduction

The Web Feature Service (WFS) represents a change in the way geographic information is created, modified and exchanged on the Internet. Rather than sharing geographic information at the file level using File Transfer Protocol (FTP), for example, the WFS offers direct fine-grained access to geographic information at the feature and feature property level. Web feature services allow clients to only retrieve or modify the data they are seeking, rather than retrieving a file that contains the data they are seeking and possibly much more. That data can then be used for a wide variety of purposes, including purposes other than their producers' intended ones.

In the taxonomy of services defined in ISO 19119, the WFS is primarily a feature access service but also includes elements of a feature type service, a coordinate conversion/transformation service and a geographic format conversion service.

Geographic information — Web Feature Service

1 Scope

This International Standard specifies the behaviour of a web feature service that provides transactions on and access to geographic features in a manner independent of the underlying data store. It specifies discovery operations, query operations, locking operations, transaction operations and operations to manage stored parameterized query expressions.

Discovery operations allow the service to be interrogated to determine its capabilities and to retrieve the application schema that defines the feature types that the service offers.

Query operations allow features or values of feature properties to be retrieved from the underlying data store based upon constraints, defined by the client, on feature properties.

Locking operations allow exclusive access to features for the purpose of modifying or deleting features.

Transaction operations allow features to be created, changed, replaced and deleted from the underlying data store.

Stored query operations allow clients to create, drop, list and describe parameterized query expressions that are stored by the server and can be repeatedly invoked using different parameter values.

NOTE This International Standard does not address the access control issues.

This International Standard defines 11 operations:

- GetCapabilities (discovery operation);
- DescribeFeatureType (discovery operation);
- GetPropertyValue (query operation);
- GetFeature (query operation);
- LockFeature (locking operation);
- GetFeatureWithLock (query and locking operation);
- Transaction (transaction operation);
- CreateStoredQuery (stored query operation);
- DropStoredQuery (stored query operation);
- ListStoredQueries (stored query operation);
- DescribeStoredQueries (stored query operation).

2 Conformance

Table 1 specifies the conformance classes defined by this International Standard and the tests specified in Annex A that shall be satisfied in order to comply with each class.

Table 1 also lists the following.

- a) Which, if any, filter encoding (see ISO 19143:2010, Clause 2) conformance tests need to be satisfied with each WFS conformance class.
- b) Which, if any, GML (see ISO 19136:2007) conformance tests need to be satisfied with each WFS conformance class.

Table 1 — Conformance classes

Conformance class name	Operation or behaviour	WFS conformance test	FES conformance test(s)	GML conformance test(s)
Simple WFS	The server shall implement the following operations: GetCapabilities, DescribeFeatureType, ListStoredQueries, DescribeStoredQueries, GetFeature operation with at least the StoredQuery action. One stored query, that fetches a feature using its id, shall be available, but the server may also offer additional stored queries. Additionally, the server shall conform to at least one of the HTTP GET, HTTP POST or SOAP conformance classes.	A.1.1	ISO 19143:2010, A.1	ISO 19136:2007, A.1.1, A.1.4, A.1.5, A.1.7, B.3, B.5, B.2.3
Basic WFS	The server shall implement the Simple WFS conformance class and shall additionally implement the GetFeature operation with the Query action and the GetPropertyValue operation.	A.1.2	ISO 19143:2010, A.2, A.7, A.8, A.10, A.11, A.12, A.14	ISO 19136:2007, B.4
Transactional WFS	The server shall implement the Basic WFS conformance class and shall also implement the Transaction operation.	A.1.3		
Locking WFS	The server shall implement the Transactional WFS conformance class and shall implement at least one of the GetFeatureWithLock or LockFeature operations.	A.1.4		
HTTP GET	The server shall implement the Key-value pair encoding for the operations that the server offers.	A.1.5		
HTTP POST	The server shall implement the XML encoding for the operations that the server implements.	A.1.6		
SOAP	The server shall implement XML-encoded requests and results within SOAP Envelopes.	A.1.7		
Inheritance	The server shall implement the schema-element() function for XPath expressions.	A.1.8	ISO 19143:2010, A.15	

Table 1 (continued)

Conformance class name	Operation or behaviour	WFS conformance test	FES conformance test(s)	GML conformance test(s)
Remote resolve	The server shall implement the ability to resolve remote resource references.	A.1.9		ISO 19136:2007, B.2.1
Response paging	The server shall implement the ability to page through the set of response features or values.	A.1.10		ISO 19136:2007, B.3
Standard joins	The server shall implement join predicates using all Filter operators except the spatial and temporal operators.	A.1.11	ISO 19143:2010, A.8, A.10	
Spatial joins	The server shall implement join predicates using spatial operators.	A.1.12	ISO 19143:2010, A.11, A.12	
Temporal joins	The server shall implement join predicates using temporal operators.	A.1.13	ISO 19143:2010, A.9, A.10	
Feature versions	The server shall implement the ability to navigate feature versions.	A.1.14	ISO 19143:2010, A.11	
Manage stored queries	The server shall implement the CreateStoredQuery and the DropStoredQuery operations.	A.1.15	ISO 19143:2010, A.1	

3 Normative references

The following normative documents contain provisions, which, through reference in this text, constitute provisions of this document. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. For undated references, the latest edition of the normative document referred to applies.

ISO/TS 19103:2005, *Geographic information — Conceptual schema language*

ISO 19136:2007, *Geographic information — Geography Markup Language (GML)*

ISO 19143:2010, *Geographic information — Filter encoding*

IETF RFC 2616, *Hypertext Transfer Protocol — HTTP/1.1* (June 1999)

IETF RFC 4646, *Tags for Identifying Languages* (September 2006)

OGC 06-121r3, *OGC Web Services Common Specification*, OGC® Implementation Specification (9 February 2009)

OGC 07-092r3, *Definition identifier URNs in OGC namespace*, OGC® Best Practices (15 January 2009)

W3C SOAP, *Simple Object Access Protocol (SOAP) 1.2*, W3C Note (27 April 2007)

W3C WSDL, *Web Services Description Language (WSDL) 1.1*, W3C Note (15 March 2001)

W3C XML Namespaces, *Namespaces in XML*, W3C Recommendation (14 January 1999)

W3C XML Path Language, *XML Path Language (XPath) 2.0*, W3C Recommendation (23 January 2007)

W3C XML Schema Part 1, *XML Schema Part 1: Structures*, W3C Recommendation (2 May 2001)

4 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

4.1

attribute <XML>

name-value pair contained in an **element** (4.6)

[ISO 19136:2007, definition 4.1.3]

NOTE In this International Standard, an attribute is an XML attribute unless otherwise specified.

4.2

client

software component that can invoke an **operation** (4.17) from a **server** (4.28)

[ISO 19128:2005, definition 4.1]

4.3

coordinate

one of a sequence of n numbers designating the position of a point in n -dimensional space

[ISO 19111:2007, definition 4.5]

4.4

coordinate reference system

coordinate system (4.5) that is related to an object by a datum

[ISO 19111:2007, definition 4.8]

4.5

coordinate system

set of mathematical rules for specifying how **coordinates** (4.3) are to be assigned to points

[ISO 19111:2007, definition 4.10]

4.6

element <XML>

basic information item of an XML document containing child elements, **attributes** (4.1) and character data

[ISO 19136:2007, definition 4.1.23]

4.7

feature

abstraction of real world phenomena

[ISO 19101:2002, definition 4.11]

NOTE A feature can occur as a type or an instance. The term "feature type" or "feature instance" should be used when only one is meant.

4.8

feature identifier

identifier that uniquely designates a **feature** (4.7) instance

4.9

filter expression

predicate expression encoded using XML

[ISO 19143:2010, definition 4.11]

4.10

interface

named set of **operations** (4.17) that characterize the behaviour of an entity

[ISO 19119:2005, definition 4.2]

4.11

join predicate

filter expression (4.9) that includes one or more clauses that constrain properties from two different entity types

[ISO 19143:2010, definition 4.16]

NOTE In this International Standard, the entity types will be **feature** (4.7) types.

4.12

join tuple

set of two or more object instances that satisfy a filter that includes **join predicates** (4.11)

NOTE In this International Standard, the object instances will be **feature** (4.7) instances.

4.13

local resource

resource that is under the direct control of a system

NOTE In this International Standard, the system is a web feature service and the resource is held in a data store that is directly controlled by that service.

4.14

locator attribute

attribute (4.1) whose value is a reference to a **local resource** (4.13) or **remote resource** (4.20)

NOTE In XML, this attribute is commonly called an href and contains a URI reference to the remote resource (see W3C XLink).

4.15

Multipurpose Internet Mail Extensions (MIME) type

media type and subtype of data in the body of a message that designates the native representation (canonical form) of such data

[IETF RFC 2045:1996]

4.16

namespace

<XML>

collection of names, identified by a URI reference, which are used in XML documents as **element** (4.6) names and **attribute** (4.1) names

[W3C XML Namespaces:1999]

4.17

operation

specification of a transformation or query that an object may be called to execute

[ISO 19119:2005, definition 4.3]

4.18

property

facet or attribute of an object, referenced by a name

[ISO 19143:2010, definition 4.21]

4.19

resource

asset or means that fulfils a requirement

[ISO 19115:2003, definition 4.10]

NOTE In this International Standard, the resource is a **feature** (4.7), or any identifiable component of a feature (e.g. a property of a feature).

4.20

remote resource

resource that is not under the direct control of a system

NOTE In this International Standard, the system is a web feature service. The resource is not held in any data store that is directly controlled by that service and thus cannot be directly retrieved by the service.

4.21

request

invocation of an **operation** (4.17) by a **client** (4.2)

[ISO 19128:2005, definition 4.10]

4.22

relocate

<reference> update a reference to a resource that has been moved or copied to a new location

EXAMPLE A **server** (4.28) is generating a **response** (4.24) to a GetFeature **request** (4.21), it has to copy a referenced **feature** (4.7) into the response document and the server has to "relocate" the original link contained in the referencing feature to the copy placed in the response document.

4.23

resolve

retrieval of a referenced resource and its insertion into a server-generated response document

NOTE The insertion may be accomplished by either replacing the reference inline with a copy of the resource or by relocating the reference to point to a copy of the resource that has been placed in the response document.

4.24

response

result of an **operation** (4.17) returned from a **server** (4.28) to a **client** (4.2)

[ISO 19128:2005, definition 4.11]

4.25

response model

schema (4.26) defining the properties of each **feature** (4.7) type that can appear in the **response** (4.24) to a query **operation** (4.17)

NOTE This is the schema of feature types that a **client** (4.2) can obtain using the DescribeFeatureType operation (see Clause 9).

4.26

schema

formal description of a model

[ISO 19101:2002, definition 4.25]

NOTE In general, a schema is an abstract representation of an object's characteristics and relations to other objects. An XML schema represents the relationship between the **attributes** (4.1) and **elements** (4.6) of an XML object (for example, a document or a portion of a document).

4.27

schema <XML Schema>

collection of **schema** (4.26) components within the same target **namespace** (4.16)

[ISO 19136:2007, definition 4.1.54]

EXAMPLE Schema components of W3C XML Schema are types, **elements** (4.6), **attributes** (4.1), groups, etc.

4.28

server

particular instance of a **service** (4.29)

[ISO 19128:2005, definition 4.12]

4.29

service

distinct part of the functionality that is provided by an entity through **interfaces** (4.10)

[ISO 19119:2005, definition 4.1]

4.30

service metadata

metadata describing the **operations** (4.17) and geographic information available at a **server** (4.28)

[ISO 19128:2005, definition 4.14]

4.31

traversal

<XML>

using or following an XLink link for any purpose

[W3C XLink:2001]

4.32

tuple

ordered list of values

[ISO 19136:2007, definition 4.1.63]

NOTE In this International Standard, the ordered list will generally be a finite sequence of **features** (4.7), each of a specific feature type.

4.33

Uniform Resource Identifier

URI

unique identifier for a resource, structured in conformance with IETF RFC 2396

[ISO 19136:2007, definition 4.1.65]

NOTE The general syntax is <scheme>::<scheme-specified-part>. The hierarchical syntax with a **namespace** (4.16) is <scheme>://<authority><path>?<query>.

5 Conventions

5.1 Abbreviated terms

CGI	Common Gateway Interface
CRS	Coordinate Reference System
DCP	Distributed Computing Platform
EPSG	European Petroleum Survey Group
FES	Filter Encoding Specification
GML	Geography Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Secure Hypertext Transfer Protocol
IETF	Internet Engineering Task Force
KVP	Keyword-value pairs
MIME	Multipurpose Internet Mail Extensions
OGC	Open Geospatial Consortium
OWS	OGC Web Service
SQL	Structured Query Language
SOAP	Simple Object Access Protocol
UML	Unified Modelling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
URN	Uniform Resource Name
VSP	Vendor Specific Parameter
WFS	Web Feature Service
WSDL	Web Services Description Language
XML	Extensible Markup Language

5.2 Use of examples

This International Standard makes extensive use of XML examples. They are meant to illustrate the various aspects of a web feature service specified in this International Standard. The bulk of the examples can be found in Annex B with some examples embedded in the body of the specification. All examples reference fictitious servers and data. Thus, this International Standard does not assert that any XML or keyword-value pair encoded examples, copied from this International Standard, will necessarily execute correctly or validate using a particular XML validation tool.

5.3 XML schemas

Throughout this International Standard, XML Schema (see W3C XML Schema Part 1 and W3C XML Schema Part 2) fragments are used to define the XML encoding of WFS operations. These fragments are gathered into a single validated schema file in Annex C.

5.4 UML Notation

5.4.1 Class diagrams

Figure 1 describes the Unified Modelling Language (UML) notations used in this International Standard for UML class diagrams.

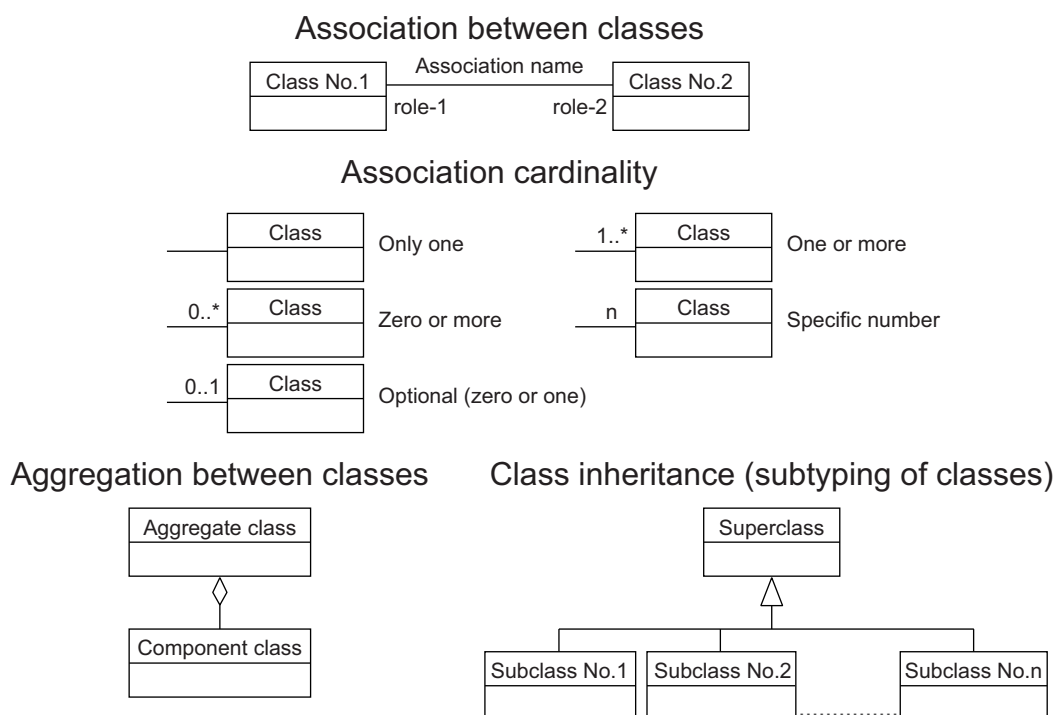


Figure 1 — UML notation in class diagrams

In these class diagrams, the following stereotypes of UML classes are used:

- a) `<<DataType>>` A descriptor of a set of values that lack identity (independent existence and the possibility of side effects). A DataType is a class with no operations, whose primary purpose is to hold the information.
- b) `<<Enumeration>>` A data type whose instances form a list of alternative literal values. Enumeration means a short list of well-understood potential values within a class.

- c) <<CodeList>> A flexible enumeration for expressing a long list of potential alternative values. If the list alternatives are completely known, an enumeration shall be used; if the only likely alternatives are known, a code list shall be used.
- d) <<Interface>> A definition of a set of operations that is supported by objects having this interface. An Interface class cannot contain any attributes.
- e) <<Type>> A stereotyped class used for specification of a domain of instances (objects), together with the operations applicable to the objects. A Type class may have attributes and associations.
- f) <<Union>> A list of alternate attributes where only one of those attributes may be present at any time.

See also ISO/TS 19103:2005, 6.8.2 and D.8.3.

In this International Standard, the following standard data types are used:

- 1) CharacterString – A sequence of characters;
- 2) LocalisedCharacterString – A CharacterString associated with a locale;
- 3) Boolean – A value specifying TRUE or FALSE;
- 4) URI – An identifier of a resource that provides more information;
- 5) Integer – An integer number.

5.4.2 State machine notation

The approach to dynamic modelling used is that described by the UML Reference Manual. The main technique is the state machine view. A summary of the UML notation for state diagrams is shown in Figure 2.

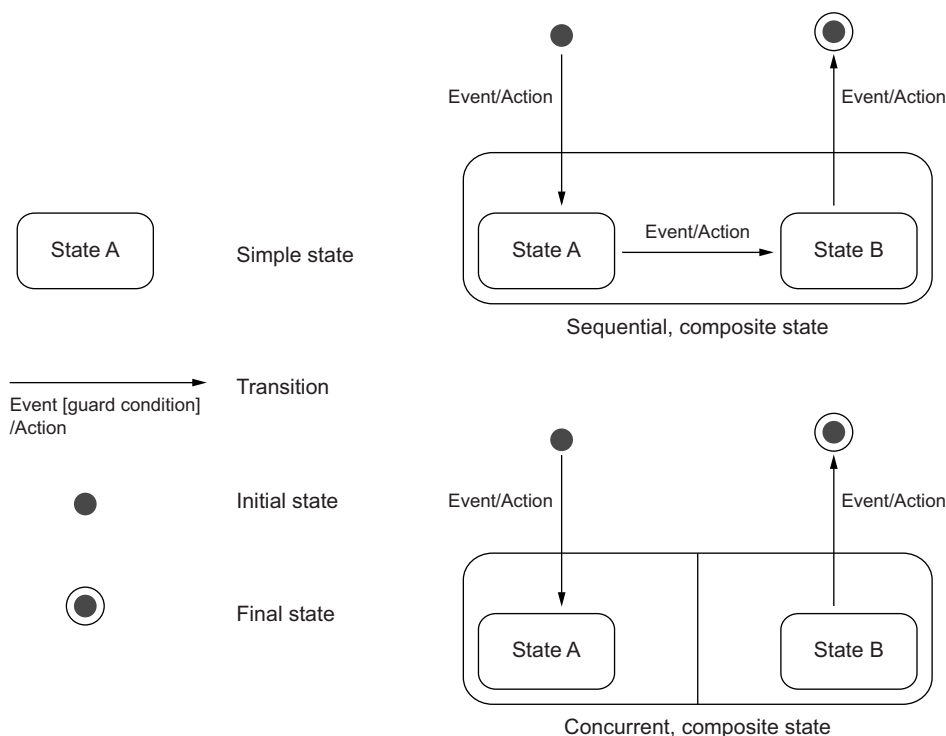


Figure 2 — Summary of UML state diagram notations

6 Basic service elements

6.1 Introduction

This clause describes aspects of a Web Feature Service's behaviour that are independent of particular operations or are common to several operations or interfaces.

6.2 Version numbering and negotiation

6.2.1 Version number form and value

Version numbers shall be encoded as described in OGC 06-121r3:2009, 7.3.1. Implementations of this International Standard shall use the value "2.0.0" as the protocol version number.

6.2.2 Appearance in service metadata and in requests

A web feature server that conforms to this International Standard shall list the version "2.0.0" in its service metadata (see 8.3). A server may support several versions whose values clients can discover according to a set of version negotiation rules (see OGC 06-121r3:2009, 7.3.2).

The version number used in requests to a server shall be equal to a version number that the server has declared it supports (except during negotiation as described in 6.2.3). If the server receives a request with a version number that it does not support, the server shall raise an `InvalidParameterValue` exception (see 7.5).

The version number 2.0.0 shall be specified for any request that conforms to this International Standard.

6.2.3 Version number negotiation

All WFS implementations shall support version negotiation according to the rules described in OGC 06-121r3:2009, 7.3.2.

6.2.4 Request encoding

This International Standard defines two methods of encoding WFS requests. The first uses XML as the encoding language. The second encoding uses keyword-value pairs (KVP) to encode the various parameters of a request.

EXAMPLE An example of a keyword-value pair is "REQUEST=GetCapabilities", where "REQUEST" is the keyword and "GetCapabilities" is the value.

The KVP encoding is a subset of the XML encoding since KVP encoding is not amenable for encoding certain WFS operations, such as the Transaction operation. In both cases, the response to a request or exception reporting shall be identical.

Table 2 correlates WFS operations and their encoding semantics as defined in this International Standard.

Table 2 — Operation request encoding

Operation	Request encoding
GetCapabilities	XML & KVP
DescribeFeatureType	XML & KVP
GetPropertyValue	XML & KVP
GetFeature	XML & KVP
GetFeatureWithLock	XML & KVP
LockFeature	XML & KVP
Transaction	XML
CreateStoredQuery	XML
DropStoredQuery	XML & KVP
ListStoredQueries	XML & KVP
DescribeStoredQueries	XML & KVP

6.2.5 KVP parameter encoding rules

6.2.5.1 Introduction

Aspects of KVP encoding WFS requests are described in 6.2.5.2 and 6.2.5.3.

6.2.5.2 Parameter ordering and case

Parameter names shall not be case-sensitive, but parameter values shall be case-sensitive. In this International Standard, parameter names used in KVP encoding are typically shown in uppercase for typographical clarity, not as a requirement.

Parameters in a request may be specified in any order.

A web feature service shall be prepared to encounter parameters that are not part of this International Standard. In terms of producing results according to this International Standard, a web feature service shall ignore such parameters.

6.2.5.3 Parameter lists

Parameters consisting of lists shall be encoded as described in ISO 19143:2010, 5.5.

EXAMPLE In the case where multiple join queries (see 7.9.2.5.3) are KVP-encoded, parentheses are used to isolate the various parameters so that their values align correctly. This is illustrated by the following example:

```
TYPENAMES=(ns1:F1,ns2:F2) (ns1:F1,ns1:F1) &ALIASES=(A,B) (C,D) &FILTER=(<Filter> ... for A,B ...
</Filter>) (<Filter>...for C,D...</Filter>)
```

This KVP-encoded fragment encodes two join queries. The first query joins the feature types "ns1:F1" and "ns2:F2" which have been aliased to "A" and "B". The second query joins the feature types "ns1:F1" and "ns1:F1" (a self-join) which have been aliased to "C" and "D". The FILTER parameter encodes two filters, one for each query, delimited with parentheses. All the parameter values align 1:1. This fragment is equivalent to the following two individual KVP-encoded fragments:

```
TYPENAMES=ns1:F1,ns2:F2&ALIASES=A,B&FILTER=<Filter>...for A,B...</Filter>
TYPENAMES=ns1:F1,ns1:F1&ALIASES=C,D&FILTER=<Filter>...for C,D...</Filter>
```

6.3 Namespaces

Namespaces (see W3C XML Namespaces) are used to discriminate XML vocabularies from one another. For the WFS, there are four normative namespace definitions, namely:

- (<http://www.opengis.net/wfs/2.0>) – for the WFS interface vocabulary;
- (<http://www.opengis.net/gml/3.2>) – for the GML vocabulary (see ISO 19136:2007);
- (<http://www.opengis.net/fes/2.0>) – for the OGC Filter vocabulary (see ISO 19143:2010, 5.4);
- (<http://www.opengis.net/ows/1.1>) – for the OWS Common vocabulary (see OGC 06-121r3).

In addition, WFS implementations may make use of one or more GML Application Schemas and these schemas will, in turn, use one or more application namespaces (e.g. <http://www.someserver.com/myns>). While many of the examples in this International Standard use a single namespace, multiple namespaces may be used, as shown in 11.3.3.

For XML-encoded requests, the namespaces used in the request shall be encoded using the notation "xmlns:prefix=namespace_uri" in the root element of the request (see W3C XML Namespaces).

For KVP-encoded requests, the NAMEPSPACES parameter (see 7.6.6) shall be used to declare any namespaces used in the request.

6.4 Service bindings

The main body of this International Standard defines the encoding of WFS request and response messages, independent of any particular communication protocol. However, implementations of this International Standard shall support one of HTTP GET, HTTP POST or SOAP over HTTP POST (see Clause 2). Annex D contains a detailed discussion of these service bindings.

7 Common elements

7.1 Encoding of features

Servers that conform to this International Standard shall operate upon features encoded using GML. The version of GML that shall be supported is ISO 19136:2007. However, the operations in this International Standard are defined in a manner that allows them to work with previous and future versions of GML. Therefore, servers may implement support for additional versions of GML, other than ISO 19136:2007. Servers shall advertise all supported versions of GML in their capabilities document, using the inputFormat and outputFormat parameter domains (see Table 12).

Servers may also support additional non-GML feature encodings that shall also be listed in the server's capabilities document (see Table 12). However, this International Standard does not describe how a server would operate upon such encodings.

7.2 Resource identifiers

7.2.1 Assigning resource identifiers

Each feature instance in a WFS shall be assigned a persistent unique resource identifier which is assigned by the server when the feature is created.

This identifier shall be invariant under all WFS operations, including delete, which means that a resource identifier cannot be reused once it has been assigned.

Resource identifiers are not intended to associate WFS resources with real-world objects and the values do not have to be meaningful outside the scope of a web feature service instance.

7.2.2 Encoding resource identifiers

For features encoded using GML, the resource identifier shall be encoded in the XML attribute `gml:id`. This International Standard does not describe how resource identifiers are encoded for other output formats.

Within filter expressions, specific feature instances can be identified using the `fes:Resourceid` element. If the server supports versioning, specific versions of a feature can be referenced using the `versionAction` attribute contained within an `fes:Resourceid` element (see ISO 19143:2010, 7.11.2).

7.2.3 Version identification

If the server supports versioning of features, then the server shall maintain version information about each feature instance. This International Standard makes no assumptions about how that version information is maintained. Functions defined in the Filter Encoding Standard (see ISO 19143:2010, 7.11.2) allow version navigation based on the resource identifier.

7.3 Property references

7.3.1 XPath subset

GML allows features to have complex or aggregate non-geometric properties. A problem thus arises about how components of the complex value of such properties are referenced in the various places where value references are required (e.g. query and filter expressions). When the feature content that a WFS offers is encoded using XML, a WFS shall use XPath (see W3C XML Path Language) expressions for referencing the properties and components of the value of properties of a feature. The minimum mandatory subset of XPath that servers shall support is described in ISO 19143:2010, 7.4.4.

Supporting the `schema-element()` XPath function is optional. However, if the server conforms to the Inheritance conformance class (see Table 1) then the server shall also support the `schema-element()` function (see ISO 19143:2010, 7.4.4).

7.3.2 Accessor function

In GML, a feature property may contain its value as content encoded inline or reference its value with a simple XLink (see ISO 19136:2007, 7.2.3). This means that, in the course of evaluating an XPath expression, a server may need to resolve a resource reference. To accommodate this requirement, all WFS implementations shall provide a concrete implementation of an XPath accessor function called `wfs:valueOf()`.

The argument to the function shall be the name of a property of a feature and the response shall be the value of the property. The value could simply be a text node or a list of element nodes that is the value of the named property.

The function shall resolve all locally referenced resources and, if the server advertises in its capabilities document that it can resolve remote references (see Table 13), the function shall resolve all remote resource references as well. In the event that the server only supports locally referenced resources, and it encounters a remotely referenced resource, the server shall raise an `OptionNotSupported` exception (see OGC 06-121r3:2009, Table 25).

When to use the `valueOf()` function in an XPath expression can be determined by inspecting the application schema of the server. If a property is declared so as to allow value references (see ISO 19136:2007, 7.2.3.3 and 7.2.3.7) then the `wfs:valueOf()` function should be specified in an XPath expression (see B.2.2 and B.4.5).

7.4 Predicate expression encoding

A number of operations defined in this International Standard contain predicate expressions that identify a subset of features to be operated upon. Such predicate expressions may enumerate a specific set of features to operate on, or a set of features may be defined by specifying constraints on the properties and/or components of the value of properties of a feature type.

XML-encoded predicate expressions shall be encoded using the `fes:Filter` element as described in ISO 19143:2010, 7.2.

KVP-encoded predicate expressions shall be encoded using the parameters describes in ISO 19143:2010, Table 2.

The specific set of predicates that a web feature service implements shall be advertised in the server's capabilities document using the filter capabilities section (see 8.3.3).

All implementations of this International Standard shall, at a minimum, implement the Query conformance class (see ISO 19143:2010, Table 1).

All implementations of this International Standard that implement the Basic WFS conformance class (see Table 1) shall also implement the Minimum Spatial Filter conformance class (see ISO 19143:2010, Table 1).

7.5 Exception reporting

In the event that a web feature service encounters an error while processing a request or receives an invalid request, it shall generate an XML document indicating that an error has occurred. The format of the XML error response is specified by, and shall validate against, the exception response schema defined in Clause 8 of the OWS Common Implementation Specification (see OGC 06-121r3:2009).

An `ows:ExceptionReport` element may contain one or more WFS processing exceptions specified using the `ows:Exception` element. The mandatory `version` attribute is used to indicate the version of the service exception report schema. This value shall be "2.0.0". The optional `language` attribute may be used to indicate the language used. The code list for the language parameter is defined in IETF RFC 4646.

Individual exception messages are contained within the `ows:ExceptionText` element. The mandatory `code` attribute shall be used to associate an exception code with the accompanying message.

The optional `locator` attribute may be used to indicate where an exception was encountered in the request that generated the error. Table 3 indicates what value the `locator` parameter should have for each exception code.

Table 3 — WFS exception codes

exceptionCode values	Meaning of code	"Locator" value	Conformance Class
CannotLockAllFeatures	A locking request with a lockAction of ALL failed to lock all the requested features.	If the operation includes the optional "handle" parameter, report its value as the value of the "location" parameter.	Locking WFS
DuplicateStoredQueryIdValue	The identifier specified for a stored query expression is a duplicate.	The "locator" parameter shall contain the value of the duplicate identifier.	Manage stored queries
DuplicateStoredQueryParameterName	This specified name for a stored query parameter is already being used within the same stored query definition.	The "locator" parameter shall list the name of the duplicate stored query parameter.	Manage stored queries
FeaturesNotLocked	For servers that do not support automatic data locking (see 15.2.3.1), this exception indicates that a transaction operation is modifying features that have not previously been locked using a LockFeature (see Clause 12) or GetFeatureWithLock (see Clause 13) operation.	If the operation includes the optional "handle" parameter, report its value as the value of the "location" parameter.	Locking WFS
InvalidLockId	The value of the lockId parameter on a Transaction operation is invalid because it was not generated by the server.	The "locator" parameter shall contain the value of the invalid lockId.	Locking WFS
InvalidValue	A Transaction (see Clause 15) has attempted to insert or change the value of a data component in a way that violates the schema of the feature.	The "locator" parameter shall contain the name of the property being incorrectly modified.	Transactional WFS
LockHasExpired	The specified lock identifier on a Transaction or LockFeature operation has expired and is no longer valid.	The "locator" parameter shall contain the value of the expired lock identifier.	Locking WFS
OperationParsingFailed	The request is badly formed and failed to be parsed by the server.	The "locator" parameter shall contain the value of the "handle" parameter if one is available. Otherwise, the "locator" parameter shall contain the name of the badly formed operation.	All (see Table 1)
OperationProcessingFailed	An error was encountered while processing the operation.	The "locator" parameter shall contain the value of the "handle" parameter if one is available. Otherwise, the "locator" parameter shall contain the name of the operation that failed.	All (see Table 1)
ResponseCacheExpired	The response cache used to support paging has expired and the results are no longer available.	If the operation includes the optional "handle" parameter, report its value as the value of the "locator" parameter.	Response paging

Servers shall implement the generic exception codes in Table 25 of OGC 06-121r3:2009 for all conformance classes defined in this International Standard.

Multiple exceptions may be reported in a single exception report, so server implementations should endeavour to report as many exceptions as necessary to clearly describe a problem.

EXAMPLE If parsing an operation fails because the value of a parameter is invalid, the server should report an `OperationParsingFailed` exception and an `InvalidParameterValue` exception.

Annex B contains examples of exception reports.

7.6 Common request parameters

7.6.1 Introduction

Request parameters common to all or several operations defined in this International Standard are described in 7.6.2 to 7.6.6.

7.6.2 Base request type

7.6.2.1 Request semantics

The base request type (see Figure 3) is an abstract type from which all WFS operations, except the `GetCapabilities` operation, are subtyped.

<i>BaseRequest</i>
+ service : CharacterString = "WFS" {frozen} + version : CharacterString = "2.0.0" {frozen} + handle [0..1] : CharacterString

Figure 3 — BaseRequest

7.6.2.2 XML encoding

The following XML Schema fragment specifies the XML encoding of the type `BaseRequest`:

```
<xsd:complexType name="BaseRequestType" abstract="true">
  <xsd:attribute name="service" type="xsd:string"
    use="required" fixed="WFS"/>
  <xsd:attribute name="version" type="xsd:string"
    use="required" fixed="2.0.0"/>
  <xsd:attribute name="handle" type="xsd:string"/>
</xsd:complexType>
```

7.6.2.3 KVP encoding

Table 4 defines the KVP encoding of the base request type.

The value of the mandatory `REQUEST` keyword shall indicate which service operation is being invoked.

NOTE XML-encoded requests do not have a `REQUEST` parameter because the name of the root element encodes the name of the service operation being invoked.

Table 4 — KVP encoding of the base request type

URLComponent	Operation	O/M ^a	Description
SERVICE	All operations.	M	See 7.6.2.4.
VERSION ^b (All operations)	All operations except GetCapabilities.	M	See 7.6.2.5.
^a O = Optional, M = Mandatory.			
^b VERSION is mandatory for all operations except the GetCapabilities operation.			

7.6.2.4 service parameter

In XML, this parameter shall be encoded using an attribute named service (see 7.6.2.2).

In the KVP encoding, this parameter shall be encoded using the SERVICE keyword (see 7.6.2.3).

The mandatory service parameter shall be used to indicate which of the available service types, at a particular server, is being invoked. When invoking a web feature service, the value of the service parameter shall be "WFS".

7.6.2.5 version parameter

In XML, this parameter shall be encoded using an attribute named version (see 7.6.2.2).

In KVP encoding, this parameter shall be encoded using the VERSION keyword (see 7.6.2.3).

All WFS requests (except the GetCapabilities operation) shall include a parameter called version.

The version parameter shall be used to indicate to which version of the WFS specification the request encoding conforms and is used in version negotiation as described in 6.2.3. When encoding a WFS request in accordance with this International Standard, the value of the version attributed shall be fixed to 2.0.0, which corresponds to the version of this International Standard.

7.6.2.6 handle parameter

In XML, this parameter shall be encoded using an attribute named handle (see 7.6.2.2).

This parameter is not defined for KVP encoding.

The handle parameter may optionally be specified on a request.

The purpose of the optional handle parameter is to allow a client application to associate a mnemonic name with a request for error handling purposes.

If a handle is specified for an operation and an exception is encountered (see 7.5) when processing that operation, a Web Feature Service shall assign the value of the handle attribute to the locator attribute in the ows:ExceptionText element (see OGC 06-121r3:2009, Clause 8) in order to identify the operation or action that generated the exception. If a handle is not specified, the server may omit the locator attribute in the ows:ExceptionText element or may use some other means, such as line numbers, to locate the exception within the operation. The handle attribute is particularly useful when used with the Transaction operation (see Clause 15) which can contain many actions. Specifying a handle for each action allows the server to exactly locate an exception within a Transaction operation.

7.6.3 Standard presentation parameters

7.6.3.1 Parameter semantics

Standard presentation parameters (see Figure 4) are used to control how query results are presented in a response document. These parameters may appear in the GetPropertyValue (see Clause 10), GetFeature (see Clause 11) and GetFeatureWithLock (see Clause 13) operations.

<i>StandardPresentationParameters</i>	<<Enumeration>> ResultType
+ resultType : ResultType = #results	+ results
+ outputFormat : CharacterString = "application/gml+xml; version=3.2"	+ hits
+ count[0..1] : Integer	
+ startIndex : Integer = 0	

Figure 4 — StandardPresentationParameters

7.6.3.2 XML encoding

The following fragment defines the XML encoding to the standard presentation parameters:

```
<xsd:attributeGroup name="StandardPresentationParameters">
  <xsd:attribute name="startIndex"
    type="xsd:nonNegativeInteger" default="0"/>
  <xsd:attribute name="count"
    type="xsd:nonNegativeInteger"/>
  <xsd:attribute name="resultType"
    type="wfs:ResultTypeType" default="results"/>
  <xsd:attribute name="outputFormat"
    type="xsd:string" default="application/gml+xml; version=3.2"
  </xsd:attributeGroup>
<xsd:simpleType name="ResultTypeType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="results"/>
    <xsd:enumeration value="hits"/>
  </xsd:restriction>
</xsd:simpleType>
```

7.6.3.3 KVP encoding

Table 5 defines the KVP encoding of the standard presentation parameters.

Table 5 — KVP encoding of standard presentation parameters

URLComponent	Operation	O/M ^a	Default	Description
STARTINDEX	GetPropertyValue, GetFeature, GetFeatureWithLock	O	1	See 7.6.3.4.
COUNT	GetPropertyValue, GetFeature, GetFeatureWithLock	O	1	See 7.6.3.5.
OUTPUTFORMAT	DescribeFeatureType, GetPropertyValue, GetFeature, GetFeatureWithLock	O	application/gml+xml; version=3.2	See 7.6.3.7.
RESULTTYPE	GetPropertyValue, GetFeature, GetFeatureWithLock	O	results	See 7.6.3.6.

^a O = Optional, M = Mandatory.

7.6.3.4 startIndex parameter

For XML-encoded requests, this parameter shall be encoded using an attribute named startIndex (see 7.6.3.2).

For KVP-encoded requests, this parameter shall be encoded using the STARTINDEX keyword (see 7.6.3.3).

The optional startIndex parameter indicates the index within the result set from which the server shall begin presenting results in the response document.

7.6.3.5 count parameter

For XML-encoded requests, this parameter shall be encoded using an attribute named count (see 7.6.3.2).

For KVP-encoded requests, this parameter shall be encoded using the COUNT keyword (see 7.6.3.3).

The optional count parameter limits the number of explicitly requested values (i.e. features or property values) that are presented in a response document.

Only values of the types explicitly requested as the value of the typeName parameter (see 7.9.2.4.1) shall be counted in the tally. Nested values contained within the explicitly requested value types shall not be counted.

EXAMPLE The tally of features in the following XML fragment is 4. The embedded reference to the feature with gml:id="2" and the feature with gml:id="5", which is that value abc:Prop4, are not included in the tally.

```
<?xml version="1.0"?>
<wfs:FeatureCollection
  timeStamp="2010-08-01T22:47:02"
  numberMatched="4" numberReturned="4"
  xmlns:myns="http://www.someserver.com/myns"
  xmlns:wfs="http://www.opengis.net/wfs/2.0"
  xmlns:gml="http://www.opengis.net/gml/3.2"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.someserver.com/myns ./myns.xsd
    http://www.opengis.net/wfs/2.0
    http://schemas.opengis.net/wfs/2.0.0/wfs.xsd
    http://www.opengis.net/gml/3.2
    http://schemas.opengis.net/gml/3.2.1/gml.xsd">

  <wfs:member>
    <myns:Feature gml:id="1">
      ...
    </myns:Feature>
  </wfs:member>
  <wfs:member>
    <myns:Feature gml:id="2">
      ...
    </myns:Feature>
  </wfs:member>
  <wfs:member>
    <myns:Feature gml:id="3">
      ...
    </myns:Feature>
  </wfs:member>
  <wfs:member>
    <myns:Feature gml:id="4">
      <myns:Property1> ... <myns:Property1>
      <myns:Property2> ... <myns:Property2>
      <myns:Property3 xlink:href="#2"/>
      <myns:Property4>
        <myns:Feature gml:id="5">
          ...
        </myns:Feature>
      </myns:Property4>
    </myns:Feature>
  </wfs:member>
</wfs:FeatureCollection>
```

In the case of a join query (see 7.9.2.5.3.1), each tuple of explicitly requested value types shall be counted as one in the tally. Nested value members contained within any of the explicitly requested value types of the tuple do not count.

The count value applies to the entire result set (i.e. the result set generated by processing one or more Query actions) and the constraint shall be applied to the values in the order in which they are presented. Once the count limit is reached, request processing may terminate and the response document, containing, at most, count values, shall be presented to the client.

There is no predefined default value defined for the count parameter and the absence of the parameter shall mean that all values in the result set shall be presented to the client, subject to any server, a configured limit. If the server does have a server configured count limit, that limit shall be advertised in the server's capabilities document using the CountDefault constraint (see Table 14).

7.6.3.6 resultType parameter

For XML-encoded requests, this parameter shall be encoded using an attribute named resultType (see 7.6.3.2).

For KVP-encoded requests, this parameter shall be encoded using the RESULTTYPE keyword (see 7.6.3.3).

A WFS can respond to a query operation in one of two ways (excluding an exception response). It may either generate a complete response document containing resources that satisfy the operation, or it may simply generate an empty response container that indicates the count of the total number of resources that the operation would return. Which of these two responses a WFS generates is determined by the value of the optional resultType parameter.

The possible values for this parameter are "results" and "hits".

If the value of the resultType parameter is set to "results", the server shall generate a complete response document containing resources that satisfy the operation. The root element of the response container shall include a count of the number of resources actually presented in the response document (see 7.7.4.3). The root element of the response container shall also include a count of the total number of resources that the operations actually found, which will always be equal to or greater than the number of resources presented in the response document (see 7.7.4.2).

If the value of the resultType attribute is set to "hits", the server shall generate an empty response document containing no resource instances and the root element of the response container shall contain the count of the total number of resources that the operation found (see 7.7.4.2). The value for the number of resources presented in the response document (see 7.7.4.3) shall be set to zero.

7.6.3.7 outputFormat parameter

For XML-encoded requests, this parameter shall be encoded using an attribute named outputFormat (see 7.6.3.2).

For KVP-encoded requests, this parameter shall be encoded using the OUTPUTFORMAT keyword (see 7.6.3.3).

The optional outputFormat parameter specifies the format used to encode resources in the response to a query operation. The default value is "application/gml+xml; version=3.2", indicating that resources in the response document shall be encoded using GML (see ISO 19136).

Every WFS that conforms to this International Standard shall support this default value.

A server may advertise additional values for the outputFormat parameter in its capabilities document (see 8.3.3), indicating that multiple output formats, including previous versions of GML, are supported. However, this International Standard does not assign any specific meaning to these additional values. In cases where additional outputFormat values are specified in the server's capabilities document, this International Standard recommends that a descriptive narrative be included for each value listed.

7.6.4 Standard resolve parameters

7.6.4.1 Parameter semantics

Servers that conform to this International Standard shall implement the ability to resolve local resource references.

Servers may optionally implement the ability to, upon request, resolve remote resource references and shall advertise this ability using the ImplementsRemoteResolve constraint in their capabilities document (see Table 13).

How resource references are handled by a server is controlled by the resolve, resolveDepth and resolveTimeout parameters as described in 7.6.4.2 to 7.6.4.7.

The standard resolve parameters (see Figure 5) may appear in the GetPropertyValue (see Clause 10), GetFeature (see Clause 11) and GetFeatureWithLock (see Clause 13) operations.

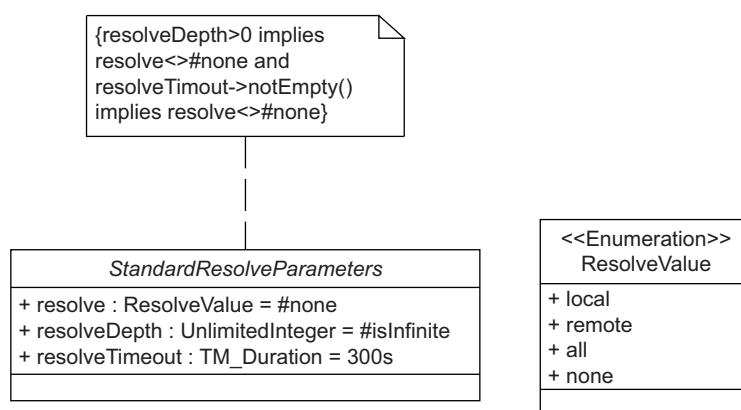


Figure 5 — StandardResolveParameters

7.6.4.2 XML encoding

The following fragment defines the XML encoding for the set of standard resolve parameters:

```
<xsd:attributeGroup name="StandardResolveParameters">
  <xsd:attribute name="resolve"
    type="wfs:ResolveValueType" default="none"/>
  <xsd:attribute name="resolveDepth"
    type="wfs:positiveIntegerWithStar" default="*"/>
  <xsd:attribute name="resolveTimeout"
    type="xsd:positiveInteger" default="300"/>
</xsd:attributeGroup>
<xsd:simpleType name="ResolveValueType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="local"/>
    <xsd:enumeration value="remote"/>
    <xsd:enumeration value="all"/>
    <xsd:enumeration value="none"/>
  </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="positiveIntegerWithStar">
  <xsd:union memberTypes="xsd:positiveInteger wfs:StarStringType"/>
</xsd:simpleType>
<xsd:simpleType name="StarStringType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="*"/>
  </xsd:restriction>
</xsd:simpleType>
```

7.6.4.3 KVP encoding

Table 6 defines the KVP encoding of the standard resolve parameters.

Table 6 — KVP encoding of standard resolve parameters

URLComponent	Operation	O/M ^a	Default	Description
RESOLVE	GetPropertyValue, GetFeature, GetFeatureWithLock	O	None	See 7.6.4.4.
RESOLVEDEPTH	GetPropertyValue, GetFeature, GetFeatureWithLock	O	*	See 7.6.4.5. RESOLVE parameter shall have a value other than "none".
RESOLVETIMEOUT	GetPropertyValue, GetFeature, GetFeatureWithLock	O	Server Specific (see ResolveTimeoutDefault, Table 14)	See 7.6.4.6. RESOLVE parameter shall have a value other than "none".

^a O = Optional, M = Mandatory.

7.6.4.4 resolve parameter

For XML-encoded requests, this parameter shall be encoded using an attribute named resolve (see 7.6.4.2).

For KVP-encoded requests, this parameter shall be encoded using the RESOLVE keyword (see 7.6.4.3).

The optional resolve parameter controls whether and which (i.e. local or remote) resource references are resolved by an operation.

The value domain of the resolve parameter is: "local", "remote", "all" or "none". Depending on which capabilities a server implements, it may support some or all of this value domain. Which resolve parameter values a server supports shall be advertised in the server's capabilities document (see 8.3.3).

A resolve parameter value of "local" means that an operation shall only resolve local references.

A resolve parameter value of "remote" means that an operation shall only resolve remote resource references.

A resolve parameter value of "all" means that an operation shall resolve all resource references.

A resolve parameter value of "none" means that an operation shall not resolve any resource references. This is also the default value if the resolve parameter is not specified.

7.6.4.5 resolveDepth parameter

For XML-encoded requests, this parameter shall be encoded using an attribute named resolveDepth (see 7.6.4.2).

For KVP-encoded requests, this parameter shall be encoded using the RESOLVEDEPTH keyword (see 7.6.4.3).

The optional resolveDepth parameter indicates the depth to which nested resource references shall be resolved in a response document.

The range of valid values for this parameter consists of non-negative integers plus "**".

A value specified for the `resolveDepth` parameter shall only be used if the `resolve` parameter (see 7.6.4.4) is specified on the operation or action where the `resolveDepth` parameter appears and the value of the `resolve` parameter is not set to "none".

If a value is not specified for the `resolve` parameter or if the value of the `resolve` parameter is set to "none", the server shall ignore any value specified for the `resolveDepth` parameter.

If the value of the `resolveDepth` parameter is specified as "0", a server shall not resolve any resource references.

If the value of the `resolveDepth` parameter is specified as "1", a server shall resolve the immediate resource references and include their value in the response document. However, if those resolved resources contain any nested resource references, those nested references shall not be resolved.

If the value of the `resolveDepth` parameter is specified as "*", a server shall resolve all immediate resource references and all nested resource references.

Having resolved references to the specified `resolveDepth`, a server shall arrange that any deeper nested references are relocated to point to their intended resources.

EXAMPLE 1 This may require that references to local resources within the data store of a WFS be converted to remote reference back to the same server in a response document.

In the event that a circular reference is detected by the server before reaching the specified `resolveDepth`, the server shall abort further nested resource resolution and arrange for the last reference in the chain to reference the appropriate, already resolved, resource.

EXAMPLE 2 Consider the following chain of resource references: A -> B -> C -> D -> E -> B. If the `resolveDepth` value is set to 8, a server would resolve all resources to E (i.e. 4 levels) and then arrange for resource E to point to the already resolved resource B. At that point, the server shall cease resource resolution since the remaining resources, to a depth of 8 levels, have already been resolved due to the circular reference.

7.6.4.6 `resolveTimeout` parameter

For XML-encoded requests, the `resolveTimeout` parameter shall be encoded using an attribute named `resolveTimeout` (see 7.6.4.2).

For KVP-encoded requests, the `resolveTimeout` parameter shall be encoded using the `RESOLVETIMEOUT` keyword (see 7.6.4.3).

The optional `resolveTimeout` parameter controls how long a server shall wait to receive a response when resolving resource references.

The `resolveTimeout` parameter is of type `xsd:positiveInteger` and specifies the expiry time, in seconds. If the `resolveTimeout` parameter is not specified, the server wait time is implementation-dependent and shall be advertised in the server's capabilities document using the `ResolveTimeoutDefault` constraint (see Table 14).

A value specified for the `resolveTimeout` parameter shall only be used if the `resolve` parameter is specified on the request element that includes the `resolveTimeout` parameter and the value of the `resolve` parameter is not set to "none".

If the value of the `resolve` parameter is set to "none", the server shall ignore any value specified for the `resolveTimeout` parameter.

7.6.4.7 Unresolvable references

In the event that a server cannot resolve a resource reference, the server shall simply report the original unresolved URI in the response. This is not considered an exception.

7.6.5 Standard input parameters

7.6.5.1 Parameter semantics

Standard input parameters (see Figure 6) are a set of parameters used to assert the encoding of resources upon input and the CRS of any geometric values those resources might contain.

These parameters can be specified on the Insert (see 15.2.4), Update (see 15.2.5) and Replace (see 15.2.6) actions of the Transaction operation (see Clause 15).

<i>StandardInputParameters</i>
+ srsName [0..1] : SC_CRS
+ inputFormat : CharacterString = "application/gml+xml; version=3.2"

Figure 6 — StandardInputParameters

7.6.5.2 XML encoding

The following fragment defines the XML encoding for the standard input parameters.

```

<xsd:attributeGroup name="StandardInputParameters">
  <xsd:attribute name="inputFormat" type="xsd:string"
    default="application/gml+xml; version=3.2"/>
  <xsd:attribute name="srsName" type="xsd:anyURI"/>
</xsd:attributeGroup>

```

7.6.5.3 KVP encoding

The standard input parameters are not defined for KVP-encoded requests because a KVP encoding is not defined for the Transaction operation (see Clause 15).

7.6.5.4 inputFormat parameter

For XML-encoded requests, the inputFormat parameter shall be encoded using an attribute named inputFormat (see 7.6.5.2).

The inputFormat parameter is not defined for KVP-encoded requests.

The inputFormat parameter may be used to assert the feature encoding used to express features upon input using the Insert action of the Transaction operation (see 15.2.4.1) or when features are updated using the Update (see 15.2.5.1) or Replace (see 15.2.6.1) actions.

For servers that implement the Transactional WFS conformance class (see Table 1), the default value of the inputFormat attribute shall be "application/gml+xml; version=3.2", indicating that input features are encoded using GML (see ISO 19136).

A server may advertise additional values for the inputFormat parameter in its capabilities document (see 8.3.3), indicating that multiple input formats, including previous versions of GML, are supported. However, this International Standard does not assign any specific meaning to these additional values. In cases where additional inputFormat values are specified in the server's capabilities document, this International Standard recommends that a descriptive narrative be included for each value listed.

7.6.5.5 srsName parameter

For XML-encoded requests, the srsName parameter shall be encoded using an attribute named srsName.

The standard input parameter, srsName, is not defined for KVP-encoded requests.

The srsName parameter may be specified as a parameter of the Transaction operation (see Clause 15) or as a parameter on the transaction actions Insert (see 15.2.4.1), Update (see 15.2.5.1) or Replace (see 15.2.6.1).

If specified as a parameter of the Transaction operation, the value of the srsName attribute shall assert the default CRS used to encode feature geometries within the Transaction (see 15.2.2).

If an srsName value is specified as a parameter of the Transaction actions Insert, Update or Replace, its value shall supersede any CRS value specified using the srsName parameter on the Transaction operation.

The srsName (see ISO 19136:2007, 10.1.3.1) parameter may also be used to assert the CRS of an individual geometry and supersedes any previous CRS assertions.

If specified, the value of the srsName parameter shall be equivalent to the default CRS value specified using the wfs:DefaultCRS element in the capabilities document (see 8.3.3) or any of the wfs:OtherCRS values (see Table 11) for the relevant feature type.

If the specified CRS is not supported for the specified feature type, the WFS shall raise an InvalidParameterValue exception (see Table 3).

If the srsName parameter is not specified, the WFS shall interpret this to mean that geometries are encoded in the default CRS as specified using the wfs:DefaultCRS element in the capabilities document (see 8.3.3).

7.6.6 Additional common keywords for KVP-encoded requests

Table 7 defines additional keywords for KVP-encoded WFS requests.

Table 7 — Additional common keywords for KVP-encoded WFS requests

URLComponent	Operation	O/M ^a	Description
NAMESPACES	All operations	O	Used to specify namespaces and their prefixes. The format shall be <i>xmlns(prefix,escaped_url)</i> where <i>escaped_url</i> is defined in OGC 06-121r3:2009, 11.3. If the prefix is not specified then the default namespace shall be assumed. More than one namespace may be bound by specifying a comma-separated list of <i>xmlns()</i> values. This parameter is not defined for XML-encoded requests because XML has another mechanism for asserting namespaces (see 6.3).
VSPs		O	A server may implement additional KVP parameters that are not part of this International Standard. These are known as VSPs. VSPs allow vendors to specify additional parameters that will enhance the results of requests. A server shall produce valid results even if the VSPs are missing or malformed, or if VSPs are supplied that are not known to the server. Unknown VSPs shall be ignored. A server may choose not to advertise some or all of its VSPs. If VSPs are included in the Capabilities XML (see 8.3), the ows:ExtendedCapabilities element (see OGC 06-121r3:2009, 7.4.6) shall be extended accordingly. Additional schema documents may be imported containing the extension(s) of the ows:ExtendedCapabilities element. Any advertised VSP shall include or reference additional metadata describing its meaning (see 8.4). WFS implementers should choose VSP names with care, to avoid clashes with WFS parameters defined in this International Standard.
^a O = Optional, M = Mandatory.			

7.7 Standard response parameters

7.7.1 Parameter semantics

Standard response parameters (see Figure 7) are response parameters used in the definition of the response collection for the GetPropertyValue (see Clause 10), GetFeature (see Clause 11) and GetFeatureWithLock (see Clause 13) operations.

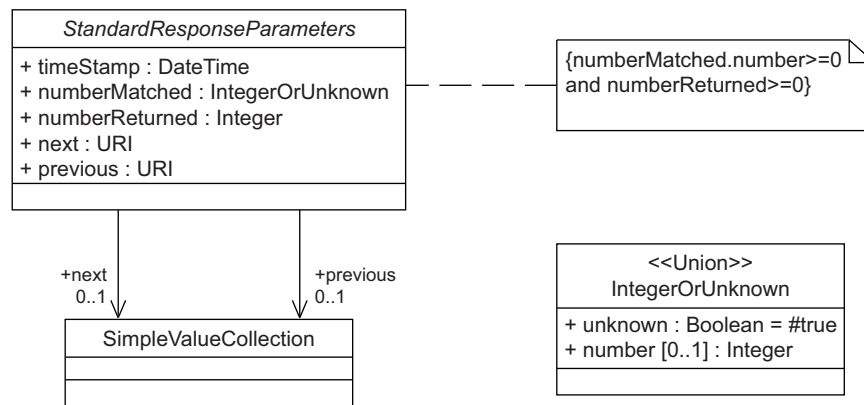


Figure 7 — StandardResponseParameters

7.7.2 XML encoding

The following XML Schema fragment defines the XML encoding of the standard response parameters:

```

<xsd:attributeGroup name="StandardResponseParameters">
  <xsd:attribute name="timeStamp" type="xsd:dateTime" use="required"/>
  <xsd:attribute name="numberMatched" type="xsd:nonNegativeIntegerOrUnknown" use="required"/>
  <xsd:attribute name="numberReturned" type="xsd:nonNegativeInteger" use="required"/>
  <xsd:attribute name="next" type="xsd:anyURI"/>
  <xsd:attribute name="previous" type="xsd:anyURI"/>
</xsd:attributeGroup>
<xsd:simpleType name="nonNegativeIntegerOrUnknown">
  <xsd:union>
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="unknown"/>
      </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType>
      <xsd:restriction base="xsd:nonNegativeInteger"/>
    </xsd:simpleType>
  </xsd:union>
</xsd:simpleType>

```

7.7.3 KVP encoding

Only an XML encoding is defined for WFS response messages.

7.7.4 Parameter discussion

7.7.4.1 timeStamp parameter

The timeStamp parameter shall be used by a WFS to indicate the time and date when a response collection was generated.

7.7.4.2 numberMatched parameter

The numberMatched parameter shall be used in a response document to report the total number of features or values, of the types requested in an operation, that are in the result set. This value does not necessarily have to match the number of features or values actually presented in the response document (see 7.7.4.3). If a server is unable to report the total number of matched features or values then it shall use the value "unknown" to indicate this.

By setting the value of the resultType parameter (see 7.6.3.6) to "hits", a count of the total number of features or values that an operation would return can be obtained without having to incur the cost of transmitting the result set. For servers that support response paging (see 7.7.4.4), the value of the next parameter shall be set to then fetch the first subset of response features or values.

7.7.4.3 numberReturned parameter

The numberReturned parameter shall be used to indicate a count of the number of features or values that are presented within a response document.

NOTE See 7.6.3.4 for value-counting rules.

If a value is specified for the count parameter (see 7.6.3.5), the value of the numberReturned parameter shall be equal to or less than the specified value of the count parameter.

If the value of the count parameter is not explicitly specified in the operation, a default value may be configured and advertised in the server's capabilities document (see 8.3.3) and shall have the same effect on numberReturned as if it were specified in the operation.

If the count parameter is not specified, either explicitly or through a server configuration, the value of the numberReturned response parameter shall be equal to the value of the numberMatched response parameter.

7.7.4.4 Response paging

7.7.4.4.1 Introduction

Response paging is the ability of a client to scroll through a set of response features or values, N features or values one at a time, much like one scrolls through the response from a search engine one page at a time.

Servers that support response paging shall advertise this fact in their capabilities document using the ImplementsResultPaging constraint (see Table 13).

Response paging is accomplished using the previous and next parameters defined on the response collections (see 10.3.1 and 11.3.1).

In order for paging to be triggered, either the count parameter (see 7.6.3.5) shall be set on the request or the server shall implement a default value for this parameter that shall be advertised in the server's capabilities document (see Table 14).

The value of the previous or next attribute shall be server-generated URIs that retrieve the corresponding set or results. The specific format of these URIs is implementation dependent, as are the details of how or if the server caches the results of an operation in order to be able to present them to the client one subset at a time.

Upon resolving the previous or next URIs, servers shall either generate a valid response collection containing the next or previous set of features or values (or join tuples in the case of a join), or an exception message indicating that the result set is no longer available (perhaps because the client waited too long before following the previous or next links and the results have, in the meantime, been purged from the server's cache). In this case, the server shall generate a ResponseCacheExpired exception (see Table 3).

Servers shall advertise how long result sets are cached for the purpose of response paging using the ResponseCacheTimeout constraint in their capabilities document (see Table 14).

EXAMPLE To illustrate the use of these parameters, consider the following GetFeature (see Clause 11) request:

```
<GetFeature service="WFS" version="2.0.0" count="100"
xmlns="http://www.opengis.net/wfs/2.0"
xmlns:cw="http://www.someserver.com/cw"
xmlns:fes="http://www.opengis.net/ogc/1.1"
xmlns:gml="http://www.opengis.net/gml/3.2">
  <Query typeName="cw:MyFeatureType"/>
</GetFeature>
```

The sequence of interactions with the server proceeds as follows.

- a) A client sends the request to a server that supports paging.
- b) The server responds with a wfs:FeatureCollection element containing the first 100 records in the result set. The *next* attribute is set so that the client can retrieve the next 100 features, but the *previous* attribute is not set since this is the first set of features in the response set.
- c) The client traverses the *next* URI.
- d) The server responds with another wfs:FeatureCollection element containing the next 100 features in the result set. In this case, the server sets both the *previous* and *next* attribute values, so that the client can retrieve the previous 100 features or the next 100 features in the result set of the GetFeature request originally posted in a).
- e) The client continues to traverse each *next* URI, until a <wfs:FeatureCollection> response is received without the *next* attribute set. This indicates that the last set of features, from the original request posted in a), has been retrieved.
- f) Similarly, a client can traverse the *previous* URI until a <wfs:FeatureCollection> response is received that does have the *previous* attribute set, indicating that the first set of 100 features has been retrieved.

7.7.4.4.2 Transactional consistency of response paging

7.7.4.4.2.1 Declaring transactional consistency

Servers, in their capabilities document, shall advertise whether they support transactional consistency for response paging using the PagingIsTransactionSafe constraint (see Table 14).

7.7.4.4.2.2 Response paging with transactional consistency

Servers that declare transactional consistency shall maintain transactional consistency between paging iterations. Thus, the view of the data that a client sees while paging through the response set shall be consistent with respect to the time that the originating request was executed.

7.7.4.4.2.3 Response paging without transactional consistency

Servers that do not declare transactional consistency shall not be required to maintain transactional consistency or state between paging iterations. Thus, it is possible for new features to be added, updated or removed from the complete result set between iterations. As a consequence, it is possible for a result set element to be skipped or duplicated between iterations.

7.8 Use of the schemaLocation attribute

Any GML document generated by a WFS shall reference an appropriate GML application schema document so that the output can be validated. This shall be accomplished using the schemaLocation attribute, as defined in the XML Schema specification (see W3C XML Schema Part 1).

EXAMPLE The following XML fragment shows the use of the schemaLocation attribute on the root element indicating the location of an XML Schema document that can be used for validation:

```
<?xml version="1.0" ?>
<wfs:FeatureCollection
  xmlns="http://www.someserver.com/myns"
  xmlns:myns="http://www.someserver.com/myns"
  xmlns:gml="http://www.opengis.net/gml/3.2"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.someserver.com/myns
http://www.someserver.com/wfs.cgi?request=DescribeFeatureType&typenames=TreesA_1M,RoadL_1M"> ...
```

In this example, the service uses a DescribeFeatureType operation (see Clause 9) to call back to itself to reference a schema document where the TreesA_1M and RoadL_1M elements are declared.

7.9 Query expressions

7.9.1 Introduction

A query expression is an action that directs a server to search its data store for resources that satisfy some filter expression encoded within the query. Query expressions search for resources of a single type or they can join two or more resource types and thus search for tuples of resources.

In this International Standard, the queryable resources are features (see 7.1 and ISO 19143:2010, 6.3.3.1.1).

A query expression resolves to a set of features or feature tuples that satisfy its filter expression. The result set is, in turn, operated upon by some operation defined in this International Standard.

EXAMPLE The GetFeature operation (see Clause 11) uses a query expression to identify a subset of features to present to a client in a response document. Similarly, the LockFeature operation (see Clause 12) uses a query expression to identify a subset of features to lock.

This International Standard defines two types of query expressions: ad hoc query expressions and stored query expressions.

Ad hoc query expressions are encoded in XML using the element wfs:Query. They are called "ad hoc" because the query is not stored by the server and is only known at runtime.

Stored query expressions are encoded in XML using the wfs:StoredQuery element. A stored query expression is a query that has been previously saved in the server's data store and can be executed at any time using the query's identifier.

Both query expression types are derived from the abstract query expression elements defined in Clause 6 of ISO 19143:2010.

7.9.2 Ad hoc query expression

7.9.2.1 Request semantics

An ad hoc query expression may be used in a GetPropertyValue (see Clause 10), GetFeature (see Clause 11), GetFeatureWithLock (see Clause 13) or LockFeature (see Clause 12) operation to identify the set of features to be operated upon.

As shown in Figure 8, an ad hoc query expression contains a typeName parameter, projection clause, a selection clause and a sorting clause.

The mandatory typeName parameter lists the name of one or more feature types to query.

The optional projection clause identifies a subset of optional feature properties that shall be presented in the result set. The projection clause for XML-encoded ad hoc query expressions may also be used to control, on a per-property basis, how resource references (see 7.6.4) are resolved in the response document.

NOTE How resource references are resolved in a response document can only be controlled at the operation level for KVP-encoded requests. This is accomplished using the RESOLVE, RESOLVEDEPTH and RESOLVETIMEOUT keywords (see 7.6.4.3). This is in contrast to XML-encoded requests where resource resolution can be controlled at the operation and property levels.

The optional selection clause specifies criteria that conditionally select features from a server's data store.

The optional sorting clause specifies how the features in the response document should be ordered.

An ad hoc query also allows a CRS to be asserted and used when presenting feature geometries in a response document.

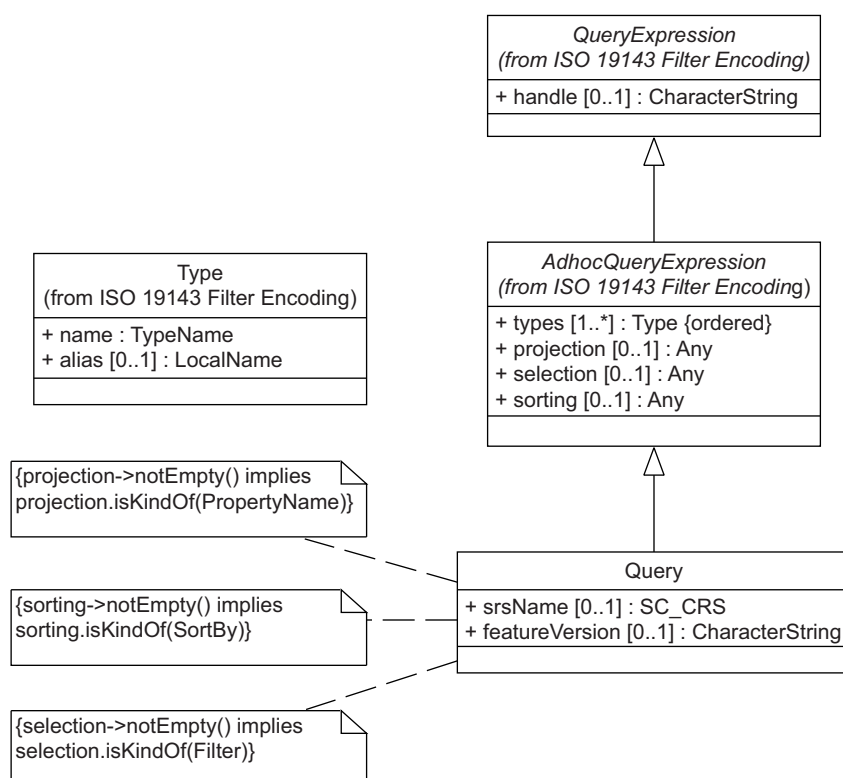


Figure 8 — Ad hoc query expression

7.9.2.2 XML encoding

The following XML Schema fragment defines the XML encoding for an ad hoc query expression:

```

<xsd:element name="Query" type="wfs:QueryType"
  substitutionGroup="fes:AbstractAdhocQueryExpression"/>
<xsd:complexType name="QueryType">
  <xsd:complexContent>
    <xsd:extension base="fes:AbstractAdhocQueryExpressionType">
      <xsd:attribute name="srsName" type="xsd:anyURI"/>
      <xsd:attribute name="featureVersion" type="xsd:string"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

```

The wfs:Query element includes the typeName parameter inherited from the fes:AbstractAdhocQueryExpressionType type (see ISO 19143:2010, 6.3.2).

7.9.2.3 KVP encoding

Table 8 defines the KVP encoding for ad hoc query expressions.

Table 8 includes parameters from 7.9.2.4 and ISO 19143:2010, Table 2, which are necessary to KVP-encode an ad hoc query expression.

Table 8 — Keywords for ad hoc query KVP encoding

URL component	O/M ^a	Description
TYPENAMES	M ^b	See 7.9.2.4.1.
ALIASES	O	See 7.9.2.4.3.
SRSNAME	O	See 7.9.2.4.4.
Projection clause	O	See Table 9.
FILTER	O	See ISO 19143:2010, 6.3.3.
FILTER_LANGUAGE	O	See ISO 19143:2010, 6.3.3.
RESOURCEID	O	See ISO 19143:2010, 6.3.3.
BBOX	O	See OGC 06-121r3.
SORTBY	O	See ISO 19143:2010, Clause 8. The SORTBY parameter is used to specify a list of property names whose values should be used to order (upon presentation) the set of feature instances that satisfy the query. The value of the SORTBY parameter shall have the form " <i>PropertyName [ASC DESC][,PropertyName [AASC DESC],...</i> " where the letters ASC are used to indicate an ascending sort and the letters DESC are used to indicate a descending sort. If neither ASC nor DESC are specified, the default sort order shall be ascending. An example value might be: " <i>SORTBY=Field1 DESC,Field2 DESC,Field3</i> ". In this case, the results are sorted by Field 1 descending, Field 2 descending and Field 3 ascending.
^a O = Optional, M = Mandatory.		
^b The TYPENAMES parameter is mandatory in all cases, except when the RESOURCEID parameter is specified (see 7.9.2.4.1).		

If multiple KVP-encoded query expressions appear in a WFS request, the parameter values for each query expression shall be isolated from one another using parentheses (see 6.2.5.3). If an optional parameter is specified for one query expression, then a value shall be specified for that parameter for all query expressions encoded in the request.

Only one of a set of mutually exclusive parameters shall be specified in a KVP-encoded request that encodes multiple query expressions; for the chosen parameter, a value shall be specified for each encoded query expression.

EXAMPLE If a KVP-encoded request contains multiple query expressions and one of those query expressions uses the FILTER keyword to encode the predicate, then all the query expressions in the request must use the FILTER keyword. You cannot, for example, have a KVP-encoded request with multiple query expressions where one query uses the FILTER keyword to encode the predicate and another query expression uses the RESOURCEID keyword.

If used in a request, the BBOX keyword shall only encode a single bounding box (see OGC 06-121r3:2009, 10.2.3). However, that bounding box predicate shall apply to all query expressions encoded in the request (see B.8.5.4). This is in contrast to the FILTER and RESOURCEID keywords which can encode multiple predicates corresponding to each query expression encoded in a KVP-encoded request (see B.8.4.15).

As is the case for XML-encoded requests, multiple KVP-encoded query expressions encoded in a single WFS operation shall be considered independent of each other.

7.9.2.4 Parameter discussion

7.9.2.4.1 typeNames parameter

For XML-encoded ad hoc query expressions, the typeNames parameter shall be encoded using the typeNames attribute on the wfs:Query element. This attribute is inherited from the abstract element:

```
fes:AbstractAdhocQueryExpressionType (see ISO 19143, 6.3.2).
```

For KVP-encoded ad hoc query expressions, the typeNames parameter shall be encoded using the TYPENAMES keyword (see Table 8). The TYPENAMES parameter is mandatory in all cases, except when the RESOURCEID parameter is specified. In this case, the TYPENAMES parameter may be omitted because each feature instance can be identified by its resource identifier alone (see 7.2). If both the TYPENAMES and RESOURCEID parameters are specified then all the feature instances identified by the RESOURCEID parameter shall be of the type specified by the TYPENAMES parameter; otherwise the server shall raise an InvalidParameterValue exception (see OGC 06-121r3:2009, Table 25) where the "locator" attribute (see OGC 06-121r3:2009, 8.4) value shall be set to "RESOURCEID".

The typeNames parameter (see ISO 19143:2010, 6.3.3.1.1) shall be used within an ad hoc query expression to encode the names of one or more correlated feature types to be queried. Individual feature type names shall be encoded as QName (see W3C XML Schema Part 2).

The value of each QName listed as a value of the typeNames parameter shall match one of the feature type names advertised in the server's capabilities document (see 8.3.3).

7.9.2.4.2 schema-element() function

If the list of values for the typeNames parameters contains a single name, then the schema-element() function can be used to trigger a sequence of queries on the specified feature type and any feature type whose object elements are in the substitution group of the specified feature type.

EXAMPLE typeNames="schema-element(ns1:Vehicle)" might, along with ns1:Vehicle, query the feature types ns1:Cars, ns1:Boats, etc.

7.9.2.4.3 aliases parameter

For XML-encoded ad hoc query expressions, the aliases parameter shall be encoded using the aliases attribute on the wfs:Query element. The aliases attribute is inherited from the abstract element:

```
fes:AbstractAdhocQueryExpressionType (see ISO 19143, 6.3.2).
```

For KVP-encoded ad hoc query expressions, the aliases parameter shall be encoded using the ALIASES keyword.

The optional aliases parameter may be used within a query expression to specify a list of alternate names for the feature type names specified as the value of the typeNames parameter. A feature type alias may be used anywhere; the feature type name may be used within the context of a query expression.

The number of list elements in the value of the aliases parameter shall match the number of corresponding feature type names in the value of the typeNames parameter and shall be correlated 1:1.

EXAMPLE 1

TYPENAMES=(ns1:FeatureType1,ns2:FeatureType2)(ns2:FeatureType2,ns2:FeatureType3)&ALIASES=(A,B)(C,D)

This KVP-encoded example encodes two ad hoc query expressions, each performing a join. The first expression is joining the feature types ns1:FeatureType1 and ns2:FeatureType2 which are aliased to A and B. The second expression is joining the feature type ns2:FeatureType2 and ns2:FeatureType3 which are aliased to C and D.

Each alias specified in the value of aliases parameter shall be unique within the context of a single query expression.

If the aliases parameter is used, an alias shall be specified for each feature type name listed as the value of typeNames parameter.

Aliases are typically used in query expressions that perform a join operation (see 7.9.2.5.3.1), to support self-join, that is a join of one feature type back to itself.

EXAMPLE 2

typeNames="myns:Feat1 myns:Feat1" aliases="a b"

This XML-encoded example shows the encoding of the typeName parameter. The first feature type, myns:Feat1, is aliased to the name "a" and the second feature type, myns:Feat1, is aliased to the name "b". Thus, properties from the first instance of myns:Feat1 can be referenced in a request as "/a/property_name", and properties from the second instance of myns:Feat1 can be referenced in a request as "/b/property_name", where the token "property_name" is used as a placeholder for the name of any property of the feature myns:Feat1.

7.9.2.4.4 srsName parameter

The optional srsName attribute may be used to assert a specific WFS-supported CRS transformation to be applied to the geometries of the features returned in a response document.

The value of the srsName parameter may be the wfs:DefaultCRS or any of the wfs:OtherCRS values listed for the feature type in a server's capabilities document (see 8.3.3). If no srsName value is supplied, then the feature geometries shall be encoded in the response document using the advertised wfs:DefaultCRS value.

This attribute has no meaning for feature types with no spatial properties and shall be ignored.

Servers that advertise more than one wfs:OtherCRS value in their capabilities document (see 8.3.3) shall be able to transform between the CRS used to store features and any CRS requested using the srsName attribute.

Servers that implement this International Standard shall be able to process srsName attribute values using the following format model:

urn:ogc:def:objectType:authority:version:<EPSG code> (see OGC 07-092r3)

In this format model, objectType shall have the value "crs", authority shall have the value "crs" and the value <EPSG Code> is a placeholder for the actual EPSG code value.

EXAMPLE srsName="urn:ogc:def:crs:EPSG::26986".

7.9.2.4.5 Projection clause

7.9.2.4.5.1 Request semantics

Every feature representation generated by a WFS shall include all the mandatory properties for the feature type according to the schema description (see Clause 9), and then may include a selection of the other properties, according to the schema description.

The projection clause enumerates which of the non-mandatory properties of a feature shall be included in the response to a query.

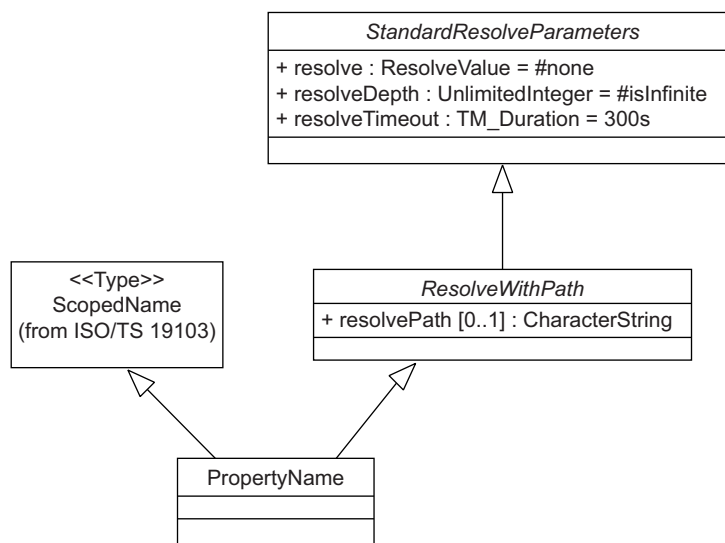


Figure 9 — Query projection clause

NOTE There is typically some flexibility in the structure of the description of a feature type of interest, especially concerning the optional or mandatory nature of each property. The W3C XML Schema (see W3C XML Schema Part 1) language, used to define GML application schemas, uses certain notations to indicate whether property elements are mandatory or optional or how many occurrences are valid. Thus, the GML representation of a feature may be schema-valid with only a subset of the possible properties present. A WFS client should be prepared to deal with a situation where it receives more property values than it requests using a projection clause.

7.9.2.4.5.2 XML encoding

The following XML-schema fragment defined the XML encoding for the wfs:PropertyName element:

```

<xsd:element name="PropertyName"
  substitutionGroup="fes:AbstractProjectionClause">
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:extension base="xsd:QName">
        <xsd:attributeGroup ref="wfs:StandardResolveParameters"/>
        <xsd:attribute name="resolvePath" type="xsd:string"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>

```

7.9.2.4.5.3 KVP encoding

Table 9 defines the KVP encoding of a projection clause.

Table 9 — KVP encoding of projection clause

URL component	O/M ^a	Description
PROPERTYNAME	O	A list of non-mandatory properties to include in the response. If more than one feature type name is specified as the value of the TYPENAMES keyword (in a non-join query), a corresponding list of parameter lists shall be specified (see 6.2.5.3). Each sublist shall correspond 1:1 with each feature type name listed as the value of the TYPENAMES parameter.
<i>StandardResolveParameters</i>		See Table 6.
^a O = Optional, M = Mandatory.		

7.9.2.4.6 Parameter discussion

7.9.2.4.6.1 PropertyName parameter

For XML-encoded requests, the projection clause shall be encoded using one or more `wfs:PropertyName` elements. The value of each `wfs:PropertyName` element is a QName whose value shall match the name of one of the property names of one of the feature types listed in the `typeNames` attribute of the parent `wfs:Query` element in the GML representation of the relevant feature.

For KVP-encoded requests, the projection clause shall be encoded using the keyword `PROPERTYNAME` (see Table 9). The value of the `PROPERTYNAME` keyword is a list or multiple lists of a QName whose values shall match the name of one of the property names of one of the feature types listed as the value of the `TYPENAMES` keyword. Multiple lists of property names corresponding to multiple query expressions shall be isolated from one another by enclosing them within parentheses (see 6.2.5.3).

7.9.2.4.6.2 Standard resolve parameters

For XML-encoded requests, the standard resolve parameters shall be encoded using the attributes named `resolve`, `resolveDepth` and `resolveTimeout` (see 7.6.4.2) on the `wfs:PropertyName` element. For XML-encoded requests, the standard resolve parameters on the projection clause control, on a per-property basis, how resource references are to be resolved within a response document. A value for these parameters specified on the `wfs:PropertyName` element shall supersede a value, if specified, on any enclosing parent element.

For KVP-encoded requests, the standard resolve parameters shall be encoded using the keywords `RESOLVE`, `RESOLVEDEPTH` and `RESOLVETIMEOUT` (see 7.6.4.3). For KVP-encoded requests, how resource references are resolved in a response document cannot be controlled on a pre-property basis and so no equivalent KVP encoding for the standard resolve parameters is specified at this level. Instead, the keywords `RESOLVE`, `RESOLVEDEPTH` and `RESOLVETIMEOUT` may be used in KVP-encoded requests to control how resource references are resolved in a response document for all feature properties

7.9.2.4.7 Resolution path

For XML-encoded requests, the `resolvePath` parameter shall be encoded as an attribute named `resolvePath`.

The `resolvePath` parameter is not defined for KVP-encoded requests.

The `resolvePath` parameter modifies the behaviour of the `resolve` parameter. The normal behaviour of the `resolve` parameter, when its value is set to `local`, `remote` or `all`, is to resolve all resource references to the depth specified by the `resolveDepth` parameter (see 7.6.4.5).

The `resolvePath` parameter, however, shall trigger resource resolution in the response document only along a certain property path.

A value specified for the `resolvePath` parameter shall only be used if the value of the nearest (in scope) `resolve` parameter is not set to `"none"`. If the value of the nearest `resolve` attribute is set to `"none"`, any value specified for the `resolvePath` parameter shall be ignored.

The value of the `resolvePath` parameter is a path expression but its specific encoding depends on how features are encoded.

For GML, which is the canonical feature encoding in this International Standard, the value of the `resolutionPath` parameter shall be an XPath (see W3C XML Path Language) expression that results in an object element. Value resolution shall stop at these instances and the value of the `resolveDepth` parameter shall be ignored.