Open Geospatial Consortium

Approval Date: 2011-12-14

Publication Date: 2012-02-09

External identifier of this OGC® document: http://www.opengis.net/doc/ows8-aixm-via-wfs2

Reference number of this document: OGC 11-073r2

Category: Engineering Report

Editors: Debbie Wilson, Ian Painter

OWS-8 Aviation: Guidance for Retrieving AIXM 5.1 data via an OGC WFS 2.0

Warning

This document is not an OGC Standard. This document is an OGC Public Engineering Report created as a deliverable in an OGC Interoperability Initiative and is not an official position of the OGC membership. This document is distributed for review and comment. It is subject to change without notice and may not be referred to as an OGC Standard. Further, any OGC Engineering Report should not be referenced as required or mandatory technology in procurements.

Document type : Public Engineering Report
Document subtype: NA
Document stage: Approved for public release
Document language: English

License Agreement

Preface

This report provides guidelines for implementing and configuring an OGC Web Feature Service (2.0) to retrieve and maintain aeronautical data encoded using the AIXM 5.1 application schema.

This report is aimed at system and client developers that shall use the OGC Web Feature Service 2.0 (WFS) interface for the exchange of aeronautical information.

This document is a deliverable for the OGC Web Services 8 (OWS-8) testbed activity. OWS testbeds are part of OGC's Interoperability Program, a global, hands-on and collaborative prototyping program designed to rapidly develop, test and deliver proven candidate standards or revisions to existing standards into OGC's Standards Program, where they are formalized for public release. In OGC's Interoperability Initiatives, international teams of technology providers work together to solve specific geoprocessing interoperability problems posed by the Initiative's sponsoring organizations. OGC Interoperability Initiatives include test beds, pilot projects, interoperability experiments and interoperability support services - all designed to encourage rapid development, testing, validation and adoption of OGC standards.

The OWS-8 sponsors are organizations seeking open standards for their interoperability requirements. After analyzing their requirements, the OGC Interoperability Team recommend to the sponsors that the content of the OWS-8 initiative be organized around the following threads:

* Observation Fusion

* Geosynchronization (Gsync)

* Cross-Community Interoperability (CCI)

* Aviation

More information about the OWS-8 testbed can be found at:

http://www.opengeospatial.org/standards/requests/74

OGC Document [11-139] “OWS-8 Summary Report” provides a summary of the OWS-8 testbed and is available for download:

https://portal.opengeospatial.org/files/?artifact_id=46176

Contents

Page

Figures

Page

Tables

Page

OGC® OWS-8 Aviation: Guidance for Retrieving AIXM 5.1 data via an OGC WFS 2.0OWS-8 Aviation: Guidance for Retrieving AIXM 5.1 data via an OGC WFS 2.0

1 Introduction

1.1 Overview

This guidance report builds upon the work sponsored by FAA and Eurocontrol within the OGC OWS Interoperability Experiment Aviation Threads (OWS-6 - 8) and the OGC FAA SAA Dissemination Pilot.

The need for a guidance report for implementing and configuring an OGC Web Feature Service 2.0 (WFS 2.0) providing access to AIXM 5.1 data for retrieval and maintenance emerged from these projects. Although it was successfully demonstrated that the WFS 2.0 is capable of supporting a wide range of flight planning and dispatch use case scenarios there were significant differences in how the WFS 2.0 specification was implemented by service providers, namely:

- Inconsistent implementation of the WFS 2.0 standard:
  - Not all mandatory operations supported
  - Different sets of query parameters supported
  - Different subsets of filter expressions supported
  - Need to align geometry and temporal data types and operands as defined in the Aviation GML profile with those supported by the WFS 2.0
- Different approaches in configuring how the data should be encoded in the response
  - Encoding individual time slices within an AIXM feature versus encoding multiple time slices (i.e. history) within a AIXM feature
  - Support for SNAPSHOT time slices
  - Support for the AIXM 5 Temporality Model

Some of these issues were expected, particularly in OWS-7 which tested the WFS 2.0 specification for the first time. Since then software implementations have matured through the FAA SAA Dissemination Pilot and OWS-8 and further experience has been gained better understanding the requirements for exchanging AIXM 5.1. To ensure that adoption and implementation of the WFS 2.0 specification by software vendors is consistent, enabling clients to easily integrate with multiple services guidance is required.

1.2 Scope

The scope of this guidelines report is to provide:

1. Overview of the OGC WFS 2.0 standard
2. Recommendations for a minimum set of operations and behaviours that should be supported to ensure consistency across software implementations.
3. Guidance for configuring the WFS 2.0 to retrieve AIXM 5.1 data
4. Summary of potential improvements to WFS/FE 2.0, GML and AIXM 5.1 specifications to better support aeronautical use cases

1.3 Document contributor contact points

All questions regarding this document should be directed to the editor or the contributors:

Name	Organization
Debbie Wilson (Editor)	Snowflake Software
Ian Painter (Editor)	Snowflake Software
Timo Thomas	Comsoft
Ulrich Berthold	Comsoft
David Burgraff	Galdos
Jeroen Dries	Luciad
Johannes Echterhoff	iGSI
Daniel Hardwick	Snowflake Software

1.4 Revision history

Date	Release	Editor	Primary clauses modified	Description
2011-03-31	0.0.1	D. Wilson		Defined initial document outline
2011-05-10	0.0.2	D. Wilson	All	First draft iteration consolidating contributions for internal review
2011-05-24	0.2.1	D. Wilson	7.1.1	Clarified comment
2011-07-29	0.3.0		All	integrated contributions and completed several sections
2011-08-28	0.5	D.Wilson	A,ll	Further contributions integrated and sections completed
2011-09-30	1.0	D.Wilson, I. Painter	All	Major review and revision based on comments received and presentation of outcomes at Aviation AWG at Sept 11 TC
2011-10-31	1.0.1	D.Wilson	6.2 7.4.1	Minor revisions based on comments received from Comsoft.
2011-11-04	1.0.2	D.Wilson	7.2 6.4.2	Re-formatting section numbering Added missing content.

1.5 Future work

The initial aim of this report was to define a normative guidance report, however it was recognized early on there were outstanding issues/questions that need to be resolved and the need to further evaluate more advanced behaviours of the WFS specification before this can be achieved. Therefore the recommendations defined throughout the report are informative and should be reviewed by the Aviation Domain Working Group (DWG) to ensure that they are in line with current and future operational requirements.

It is anticipated that the Aviation DWG shall take ownership of this report and build upon these recommendations to manage the development of a normative guidance report or Aviation WFS 2.0 Application Profile. NOTE: This should also consider the requirements for exchanging weather data.

To facilitate this work the following are proposed:

- Hold a workshop at the next OGC TC (Brussels, 2012) with all interested parties will to discuss:

- Implementing the AIXM 5 Temporal model for features within a WFS (subsetting timeslices, SNAPSHOTs with time period)
- Integrating the AIXM 5 Temporality model and OGC Dynamic Feature Model

- Recommendations for OWS 9 Aviation thread
  - Develop normative conformance document for developers, in line with the WFS 2.0 compliance test cases
  - Develop a WFS 2.0 compliance test using the OGC Team Engine. However, note that there is currently no WFS 2.0 compliance test to use as a starting block

1.6 Forward

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

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation

2 References

The following documents are referenced in 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.

EUROCONTROL and FAA, 2010 AIXM 5 Temporality Model

EUROCONTROL and FAA, 2011 AIXM 5: Feature Identification and Reference - use of xlink:href and UUID-

ISO 19136:2007 - Geographic information -- Geography Markup Language (GML)/ OGC 07-036 OpenGIS Geography Markup Language (GML) Encoding Standard

ISO 19142:2010 Geographic information – Web Feature Service/ OGC 09-025r1 OpenGIS Web Feature Service 2.0 Interface Standard

ISO 19143:2010 Geographic information - Filter encoding/OGC 09-026r1 OpenGIS Filter Encoding 2.0 Encoding Standard

OGC 06-121r3, OpenGIS® Web Services Common Standard

OGC 11-093r1 OWS-8 Aviation Architecture Engineering Report

3 Abbreviated terms

AIXM	Aeronautical Information Exchange Model
BBOX	Bounding Box
CRS	Coordinate Reference Systems
DWG	Domain Working Group
FE	Filter Encoding

GML	Geography Markup Language
HTTP	Hypertext Transfer Protocol
KVP	Key-Value Pair
OGC	Open Geospatial Consortium
UCUM	Unified Code for Units of Measure
UUID	Universal, Unique Identifier
WFS	Web Feature Service
XML	eXtensible Markup Language

4 Overview of OGC Web Feature Service 2.0 interface standard

4.1 Introduction

An OGC WFS 2.0 is an open, platform independent interface for retrieving and maintaining features contained within a remote data store. The WFS provides a comprehensive request/response interface for retrieving features or individual property values. This allows the WFS to support a wide range of use cases:

- **Data exchange:** download data on request or via synchronisation for use locally.
- **Decision-support:** directly query data remotely within client applications removing the need for local data stores.
- **Data maintenance:** direct, distributed data maintenance by multiple clients. Removing the need for multiple data maintenance flows, increasing efficiency and quality and reducing latency.

4.2 WFS 2.0 Operations

The WFS 2.0 standard defines a comprehensive set of operations for retrieving and maintaining data held in a remote data store (Figure 4-1 and Table 4-1):

- **Discovery operations:** allow the service to be interrogated to determine its capabilities and feature types served and access the application schema enabling the data to be validated
- **Query operations:** allow features or property values to be retrieved based on query criteria defined by the client. Queries can be either ad hoc or stored queries.
- **Manage stored query operations:** allow clients to interrogate, create and remove stored queries (re-usable, pre-defined queries)
- **Transaction and locking operations:** allow features to be created, modified, replaced and deleted and to enable exclusive retrieval of features for the purpose of update

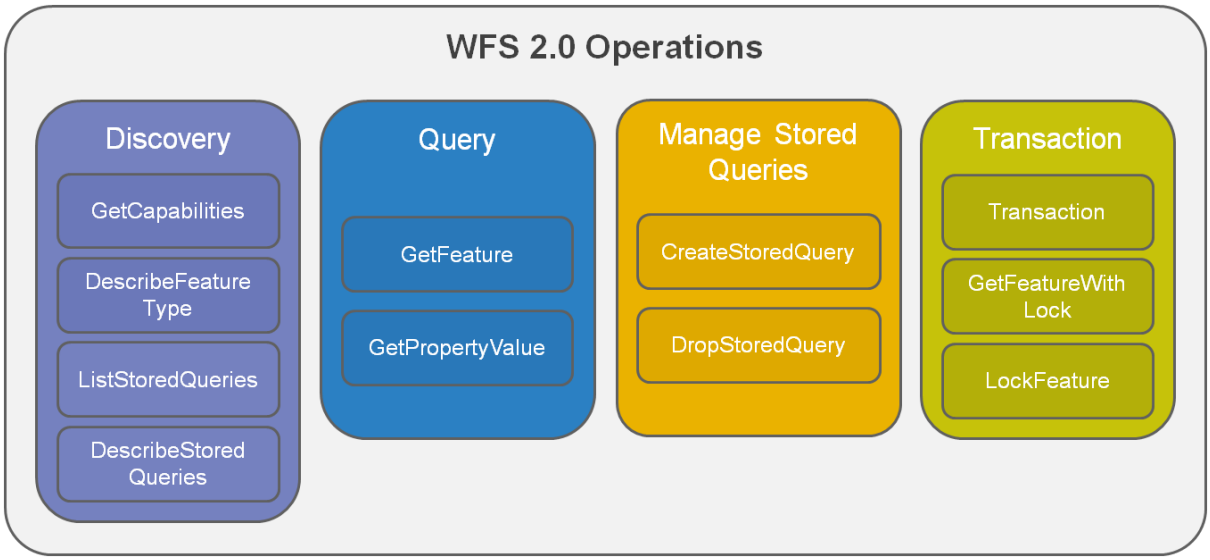


Figure 4-1. WFS 2.0 Operations

These operations shall be summarized in more detail in section 4.

Table 4-1. Summary of the WFS 2.0 Operations

Type of Operation	Operation	Description	Request Encoding
Discovery	GetCapabilities	Generates a service metadata document describing the feature types, coordinate references systems, output formats, operations and filter expressions supported by the service	XML & KVP
	DescribeFeatureType	Returns the application schema describing the feature types offered by the WFS	XML & KVP
Query	GetFeature	Returns a selection of feature instances that satisfy the query expression specified in the request	XML & KVP
	GetPropertyValue	Returns the value of a feature property or part of a complex property corresponding to the query criteria defined by the client	XML & KVP
Stored Query	ListStoredQueries	Returns a list of stored queries available on the server. All WFS shall return at least the getFeatureByID stored query	XML & KVP
	DescribeStoredQueries	Provides detailed metadata describing the stored query or queries requested	XML & KVP
	CreateStoredQuery	Allows clients to post a stored query expression on the service	XML
	DropStoredQuery	Allows stored queries to be deleted from the service	XML
Transaction and Locking	Transaction	Allows clients to modify the data using wfs:Insert, wfs:Update, wfs:Replace or wfs:Delete actions	XML
	GetFeatureWithLock	Similar to GetFeature but the services shall also lock the features in the response to enable them to be modified using a Transaction operation	XML & KVP
	LockFeature	Allows clients to lock one or more features in the data store to prevent another client from modifying the data. This is used to lock a feature before performing transactions.	XML & KVP

It is not required that a software implementing the WFS 2.0 specification support all operations as not all applications or use case scenarios require the use of all operations (see Section 4.3). Therefore, there is flexibility in which aspects of the WFS 2.0 specification can be implemented.

4.3 WFS 2.0 Conformance Classes

The WFS 2.0 specification defines a set of conformance classes that enables clients to understand which aspects of the WFS 2.0 specification have been implemented. Four core conformance classes are defined that can be used to categorize the WFS. These specify a minimum set of operations and behaviors that the WFS should support. In addition to these core conformance classes, the WFS may also implement one or more of the remaining 11 conformance classes (Figure 4-2).

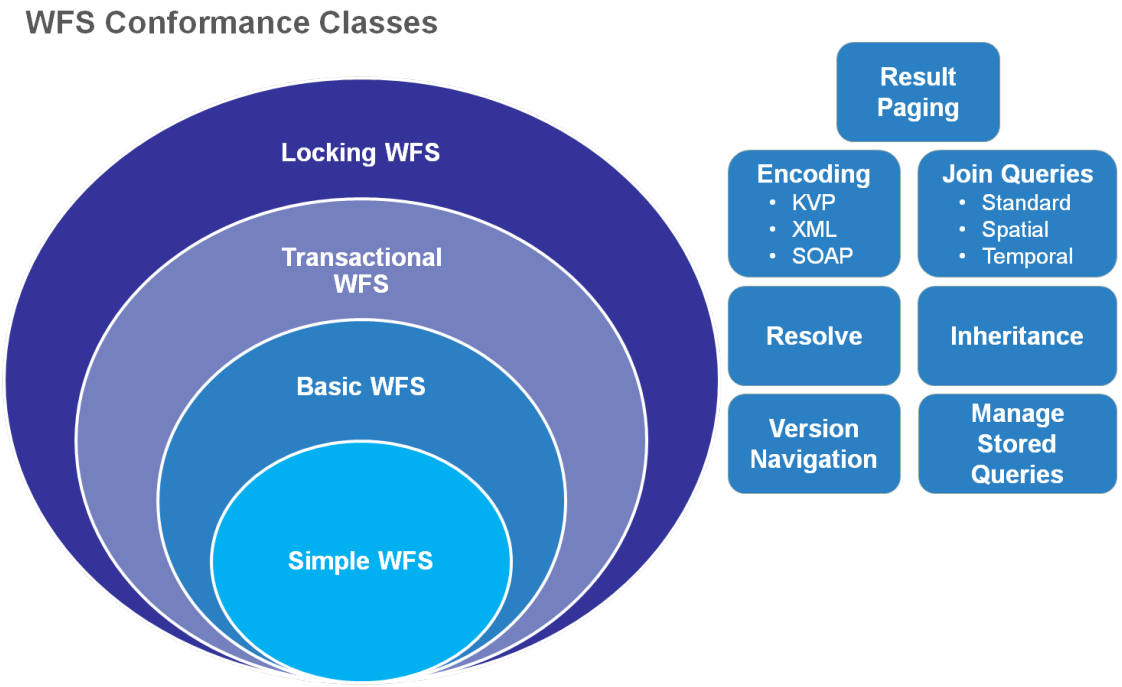


Figure 4-2 WFS 2.0 Conformance Classes

The following sections shall summarise the requirements of these conformance classes and this section shall conclude with a recommendation for the minimum set of conformance classes that should to be supported by a WFS 2.0 serving AIXM 5.1.

4.3.1 Simple WFS Conformance Class

All WFS implementations should conform to the simple WFS Conformance class as a minimum. The server shall implement the following operations:

- GetCapabilities
- DescribeFeatureType
- ListStoredQueries
- DescribeStoredQueries
- GetFeature (stored queries only)

All servers must implement one stored query that fetches a feature based on its resourceID: getFeatureByID.

The server shall conform to at least one of HTTP GET, HTTP POST or SOAP conformance class.

If a WFS only supports the simple WFS conformance class the server may be configured to offer additional stored queries. These stored queries may be specific to the application schema for the data being served or requirements of the end-user community.

A simple WFS offers several advantages and disadvantages.

Advantages:

- Service provider has control of the types of requests submitted to the server

- Service can be optimized to a pre-defined set of requirements
- Service interface can be kept simple making it easier thin and thick client applications to integrate with the server

Disadvantages:

- Client cannot define ad hoc queries to request data
- It is slower to support new user requirements as these must be formally defined to develop suitable stored queries

4.3.2 Basic WFS Conformance Class

The Basic WFS conformance class extends the Simple WFS conformance class. To conform to the Basic WFS Conformance class a server must implement the following operations:

- GetCapabilities
- DescribeFeatureType
- ListStoredQueries
- DescribeStoredQueries
- GetFeature
- GetPropertyValue

Both the GetFeature and GetPropertyValue operation must support both ad hoc and stored query actions (ISO 19143 conformance subclause A1.1 and A1.2).

To support ad hoc querying, the server shall implement the following filter encoding (ISO 19143:2010) conformance classes:

Conformance Class Name	**ISO 19143 Abstract Test Suite**	**Description**
Resource Identification	A.1.4	resourceID
Minimum Standard Filter	A.1.5	Comparison operators: PropertyIsEqualTo, PropertyIsNotEqualTo, PropertyIsLessThan, PropertyIsGreaterThan, PropertyIsLessThanOrEqualTo, PropertyIsGreaterThanOrEqualTo, Logical operators: And, Not, Or
Standard Filter	A.1.6	All the comparison and logical operators and may implement one or more additional functions
Minimum Spatial Filter	A.1.7	BBOX only
Sorting	A.1.12	Sorts the resources contained in the response based on values of one or more specified properties
Minimum XPath	A.1.14	Implements the minimum required set of XPath capabilities:

Conformance Class Name	ISO 19143 Abstract Test Suite	Description
		□ Abbreviated form of child/attribute axis specifier □ Context node shall be the resource element – except in a join query where it shall be the parent of the resource element □ Each step in the path may include an XPath predicate □ Minimum set of predicate expression shall be supported: □ Positive non-integer to indicate which child of the context node should be selected (i.e. index) □ Equality predicate for the form "=value" to indicate which child of the context node should be selected based on its value □ Equality tests of form"child=value" to indicate a specific object property by constraining the child elements. □ Last step of XPath Expression shall be a resource or subcomponent of a resource property

4.3.3 Transactional WFS Conformance Class

The Transactional WFS conformance class extends the Basic WFS conformance class and requires that the Transaction operation shall also be implemented.

4.3.4 Locking WFS Conformance Class

The Locking WFS conformance class extends the Transactional WFS conformation class requiring that the WFS shall implement at least one of the GetFeatureWithLock or LockFeature operations.

4.3.5 Additional WFS 2.0 Conformance Classes

The four core WFS conformance classes described above provide 4 high-level categories of WFS 2.0. Software implementing the WFS 2.0 specification may also choose to implement one or more of the 11 additional WFS 2.0 conformance classes:

Behaviour	Conformance Class Name	Description	WFS Conformance Test	FES Conformance Test	GML Conformance Test
Encoding (NB: a WFS must support at least one of the encoding conformanc	HTTP GET	Implements Key-value pair (KVP) encoding for operation requests.	A.1.5		
	HTTP POST	Implements XML encoding for operation	A.1.6		

Behaviour	Conformance Class Name	Description	WFS Conformance Test	FES Conformance Test	GML Conformance Test
e classes)		requests.			
	SOAP	Implements XML encoded requests and response within SOAP Envelopes.	A.1.7		
Inheritance	Inheritance	Implements schema-element() function in Xpath expressions.	A.1.8	A.1.15	
Resolve	Remote resolve	Implements ability to resolve remote resource references.	A.1.9		B.2.1
Response Paging	Response paging	The server shall implement the ability to page through the response features or values.	A.1.10		B.3
Join Queries	Standard joins	Implements join predicates using all comparison and logical filter operators	A.1.11	A.1.5, A.1.6	
	Spatial joins	Implements join predicates using spatial operators.	A.1.12	A.1.7, A.1.8	
	Temporal joins	Implements join predicates using temporal operators.	A.1.13	A.1.9, A.1.10	
Version Navigation	Feature versions	Implements the version parameters in the fes:resourceID filter to navigate feature versions.	A.1.14	A.1.11	
Manage Stored queries	Manage stored queries	Implements CreateStoredQuery and DropStoredQuery operations.	A.1.15	A.1.1	

4.3.6 Proposed Conformance Classes for an AIXM 5.1 WFS 2.0

It is proposed that the following conformance classes should by supported by a WFS 2.0 serving AIXM 5.1. Justification and further detail is summarized in the sections below:

Conformance Class Name	**Description**	**WFS Conformance Test**	**FES Conformance Test**	**GML Conformance Test**
Basic WFS	Implements the Basic WFS Conformance Tests	A.1.2	A.1.2, A.1.4, A.1.5, A.1.6, A.1.7, A.1.12, A.1.14	B.4
HTTP GET	Implements Key-value pair (KVP) encoding for operation requests.	A.1.5		
HTTP POST	Implements XML encoding for operation requests.	A.1.6		
HTTP SOAP	Implements XML encoded requests and response within SOAP Envelopes.	A.1.7		
Inheritance	Implements schema-element() function in Xpath expressions.	A.1.8	A.1.15	
Remote Resolve	Implements ability to resolve remote resource references.	A.1.9		B.2.1
Spatial Filter	Implements BBOX and one or more of the other spatial operators		A.1.8	
Temporal Filter	Implements During and one or more of the other temporal operators		A.1.10	

If a WFS 2.0 is to be used to maintain a dataset then it shall optionally support:

Conformance Class Name	**WFS Conformance Test**	**FES Conformance Test**	**GML Conformance Test**
Transactional WFS	A.1.3		

4.3.6.1 Encoding

It is proposed that any software implementing the WFS 2.0 should support all encodings for an operation request/response. Use cases have been identified when each type of request encoding is more appropriate than another:

- **HTTP GET:** allows encoding of concrete external references in feature associations and the DescribeFeatureType request can be used in the schemaLocation string to call back to the application schema used to configure the service to enable schema validation.
- **HTTP POST:** allows encoding of complex requests including filter and XPath expressions which may not be supported by HTTP GET
- **SOAP:** required to support security requirements

4.3.6.2 Inheritance

The Inheritance conformance class is extremely useful as it allows clients to specify the parent substitution type (e.g. gml:AbstractFeature or gml:AbstractGeometricPrimitive) in a request. This is particularly useful when requesting multiple features where properties may be named differently but are derived from a common type.

Example:

```
<wfs:GetFeature xmlns:aixm="http://www.aixm.aero/schema/5.1"
xmlns:fes="http://www.opengis.net/fes/2.0"
xmlns:gml="http://www.opengis.net/gml/3.2"
xmlns:wfs="http://www.opengis.net/wfs/2.0" service="WFS" version="2.0.0">
    <wfs:Query typeNames="aixm:PrecisionApproachRadar aixm:Navaid">
        <fes:Filter>
            <fes:And>
                <fes:DWithin>
                    <fes:ValueReference>//schema-
element(gml:AbstractGeometricPrimitive)</fes:ValueReference>
                    <gml:LineString gml:id="Line123"
srsName="urn:ogc:def:crs:OGC:1.3:CRS84">
                        <gml:pos>4.34 50.9</gml:pos> <!-- Bruessel -->
                        <gml:pos>11.04 49.27</gml:pos> <!-- Nuernberg -->
                        <gml:pos>8.54 47.38</gml:pos> <!-- Zuerich -->
                    </gml:LineString>
                    <fes:Distance uom="M">25000</fes:Distance>
                </fes:DWithin>
                <fes:PropertyIsEqualTo>
<fes:ValueReference>aixm:timeSlice/./aixm:interpretation</fes:ValueReference>
                    <fes:Literal>BASELINE</fes:Literal>
                </fes:PropertyIsEqualTo>
            </fes:And>
        </fes:Filter>
    </wfs:Query>
</wfs:GetFeature>
```

4.3.6.3 Remote Resolve

The AIXM 5.1 specification is a relational model and includes associations between related AIXM features encoded as references. The remote resolve query parameters in GetFeature and GetPropertyValue requests can be used to simplify retrieval of related features.

For example, a user may wish to retrieve features related to airports contained within an area of interest (i.e. spatial query). Not all AIXM features have geometry properties so may not be returned in the spatial query, it may only contain RunwayElements, TaxiwayElements and ApronElements. The client is then required to perform subsequent requests to retrieve the associated features that are referenced to in the returned features.

The resolve parameters in a GetFeature or GetPropertyValue request remove the need to perform these subsequent requests. If included in the request the query should traverse any property values containing references to retrieve the referenced feature and include it in the response.

So in the example above, a single query containing a spatial filter and resolve parameters may be performed against the RunwayElement, TaxiwayElement and ApronElement feature types. By including resolve parameters in the request, the related Runway, Taxiway and Apron features shall be returned. If the resolve depth is set to a level > 1, then the response may also include the AirportHeliport and related to the Runway, Taxiway and Apron features.

Supporting Remote Resolve

The WFS 2.0 specification states that a server shall support the ability to resolve local references. A WFS may optionally implement the ability to resolve concrete external (remote) references.

Supporting local resolve only is sufficient for aviation requirements as most WFS provide access to most referenced resources internally. However, remove resolve may be implemented if the WFS references to resources (e.g. metadata) contained in external services.

4.3.6.4 Spatial Filter

The WFS 2.0 Spatial Filter Conformance Class only requires that the BBOX and one or more of the other spatial operators are implemented. It is recommended that a minimum set of spatial filters and geometry operands are defined to ensure consistency across all WFS implementations.

The following minimum set of spatial operators and geometry operands are proposed based on existing implementations and typical requests defined for flight dispatch and planning scenarios in OWS-7/8 and FAA SAA Dissemination Pilot.

Minimum set of spatial filters:

Spatial operator	Definition
BBOX	This is a convenient and more compact way of encoding the BBOX constraint based on the gml:Envelope. The BBOX operator shall identify all features whose geometries spatially intersect the envelope.
Intersects	The Intersects operator shall test whether the geometric properties of a features intersect the geometric coordinates

	defined in the query
Within	The Within operator shall test whether the geometric properties of a features are contained within the geometric coordinates defined in the query.
DWithin*	The DWithin operator shall test whether the geometric properties of a feature are within a specified distance of a specified geometric literal value.

Minimum set of geometry operands:

Basic Geometry Operands	gml:Envelope gml:Point gml:LineString gml:Curve gml:Polygon gml:Surface gml:Arc
Additional Geometry Operands that support the Aviation GML Profile	gml:ArcString gml:GeodesicString gml:ArcByCenterPoint gml:CircleByCenterPoint

Recommendation: *DWithin Distance Parameter

A WFS should implement the Unified Code for Units of Measure (UCUM) units for use within the DWithin distance parameter to ensure that all servers can handle a consistent set of distance units.

NOTE: International, case insensitive symbols should be implemented.

Examples:

Length Unit	**UCUM Symbol (c/i)**
Meter	M
Mile	[MI_I]
Nautical Mile	[NM_I]
Foot	[FT_I]

4.3.6.5 Temporal Filter

The WFS 2.0 Temporal Filter Conformance Class only requires that the During and one or more of the other temporal operators are implemented. It is recommended that a minimum set of spatial filters and geometry operands are defined to ensure consistency across all WFS implementations.

The following minimum set of spatial filters and geometry operands are proposed based on existing implementations.

Minimum set of temporal filters:

It is proposed that a WFS shall implement all temporal filters as combinations of temporal filters are often required to filter features or property values based on temporal properties

Operator	**ISO 19108 temporal operator**
After	After
Before	Before
Begins	Begins
TContains	TContains
During	During
TEquals	TEquals
TOverlaps	TOverlaps
Meets	Meets
OverlappedBy	OverlappedBy
MetBy	MetBy
EndedBy	EndedBy
AnyInteracts[1]	N/A

[1] AnyInteracts was introduced in ISO 19143 as a convenient and more compact way of encoding temporal operators. It is semantically equivalent to NOT (Before OR Meets OR MetBy OR After).

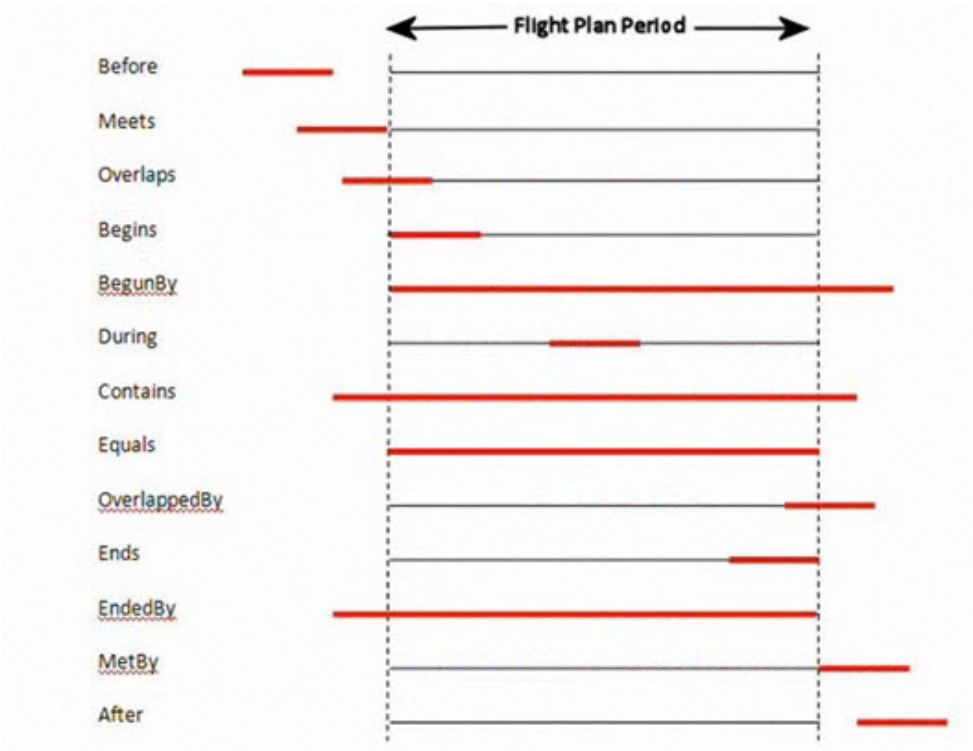


Figure 4-3. Summary of semantics of ISO 19108 temporal operators

Recommendation: Minimum set of temporal operators

To meet the requirements to request SNAPSHOT timeslices for a time instant and retrieving BASELINE, TEMPDELTA and PERMDELTA timeslices for a time period requires that both gml:TimePeriod, gml:TimeInstant temporal operators shall be supported.

4.3.6.6 Transactional WFS

If a WFS is to be used to maintain AIXM 5.1 data, the server shall support the Transaction operation.

5 Overview of WFS 2.0 Operations

5.1 Introduction

This section shall describe the key WFS 2.0 operations in more detail, describing what the operation does and the query parameters that can be included in a request. Recommendations may also be included to provide supporting guidance when implementing the WFS 2.0 specification.

5.2 Discovery Operations

5.2.1 GetCapabilities

The GetCapabilities operation generates a response containing a metadata document describing the service. The GetCapabilities response contains the following mandatory and optional sections:

- Service Identification
- Service Provider
- Operation Metadata
- WSDL (optional)
- Feature Type List
- Filter Capabilities

Example GetCapabilities request/response are available in the supporting zip in the GetCapabilities directory.

5.2.1.1 Service Identification Section

The service identification section contains general information describing the service to facilitate discovery and describe constraints that may affect access. The section shall contain the following mandatory and optional properties:

Service Identification Property	Description	Obligation (OWS 1.1 clause 7.4.4)
Title	Title of the resource. Typically used for display.	O
Abstract	Brief narrative description of the service for display.	O
Keywords	One or more keywords used to describe or classify the service to support discovery.	O
Service Type	Type of service. This shall be “WFS” by default.	M
ServiceTypeVersion	Unordered list of versions supported by the server. At least “2.0.0” should be declared. Note this information is insufficient for version negotiation the versions supported must be declared on the operations metadata using the parameter property.	M
Profile	Unordered list of identifiers of WFS 2.0	O

Service Identification Property	Description	Obligation (OWS 1.1 clause 7.4.4)
	Application Profiles implemented by the server.	
Fees	Summary of any fees that may be incurred to access and use the service. This may be a URL to an external resource which provides detailed information about any fees.	O
Access Constraints	Unordered list of access constraints applied to ensure the protection or privacy or intellectual property or other restrictions on retrieving or using the data accessible via the server.	O

5.2.1.2 Service Provider

The Service Provider section provides contact information for the organization providing the WFS. It contains the following mandatory and optional properties:

Service Provider Property	Description	Obligation (OWS 1.1 clause 7.4.5)
Provider Name	Unique identifier of the organization responsible for the server.	M
Provider Site	Reference to a relevant website of the service provider	O
Service Contact	Information describing how to contact the service provider. This section is comprised of the following: i) Individual Name; ii) Position Name: iii) Contact Info (address, phone, email) and iv) Role	O

5.2.1.3 Operation Metadata

This section is divided into 4 parts providing information describing individual supported WFS operations:

1. **Operation:** provides information about the supported connection points (DCP) for the specified operation and any default and allowable parameter values for use within a request. It may optionally contain information about any constraints and additional metadata describing the operation.
2. **Parameter:** the optional parameter part should be used to define the allowed version parameter values for all operations except GetCapabilities instead of declaring it individually on each operation. This is required to support version negotiation. NOTE: version="2.0.0" is mandatory for all operations.
3. **Constraints:** is used to declare any operation constraints that apply. This section is divided into conformance declaration and capacity constraints. NOTE: capacity

constraints may either be defined globally or in the constraints element for an individual operation if it only applies to that operation.

4. **Extended Capabilities:** description of any additional operation capabilities beyond those defined in the WFS 2.0 specification.

5.2.1.4 WSDL

This section allows a server to reference an optional WSDL document describing the operations that the service offers.

Example:

```
<wfs:WSDL
xlink:href="http://demo.snowflakesoftware.com:8081/AIXM51_WFS2/GOPublisherWFS?wsdl"/>
```

WSDL Response

```
<wsdl:definitions
targetNamespace="http://demo.snowflakesoftware.com:8081/AIXM51_WFS2/GOPublisherWFS">
        <wsdl:documentation>
                <dc:date>2010-01-01</dc:date>
                <dc:description>This WSDL document defines the service-specific properties of a
/AIXM51_WFS2/GOPublisherWFS WFS implementation;it specifies available endpoints and alternative
bindings.</dc:description>
        </wsdl:documentation>
        <wsdl:import namespace="http://www.opengis.net/wfs/soap/2.0" location="./wfs-soap-bindings.wsdl"/>
        <wsdl:import namespace="http://www.opengis.net/wfs/http/2.0" location="./wfs-http-bindings.wsdl"/>
        <wsdl:import namespace="http://www.opengis.net/wfs/http/kvp/2.0" location="./wfs-kvp-bindings.wsdl"/>
        <wsdl:service name="/AIXM51_WFS2/GOPublisherWFS">
                <wsdl:documentation>A WFS-2.0 implementation. Includes alternative SOAP bindings for the
WFS interfaces.</wsdl:documentation>
                <wsdl:port binding="wfs-soap:wfs-SOAP" name="wfs-SOAP-Port">
                        <soap:address
location="http://demo.snowflakesoftware.com:8081/AIXM51_WFS2/GOPublisherWFS"/>
                </wsdl:port>
                <wsdl:port binding="wfs-http:wfs-POST" name="wfs-POST-Port">
                        <http:address
location="http://demo.snowflakesoftware.com:8081/AIXM51_WFS2/GOPublisherWFS"/>
                </wsdl:port>
                <wsdl:port binding="wfs-http-kvp:wfs-GET" name="wfs-GET-Port">
                        <http:address
location="http://demo.snowflakesoftware.com:8081/AIXM51_WFS2/GOPublisherWFS?"/>
                </wsdl:port>
        </wsdl:service>
</wsdl:definitions>
```

Recommendation: A WFS should provide a WSDL document that can be harvested within a registry to facilitate service discovery and evaluation.

5.2.1.5 Feature Type List

The FeatureTypeList section defines a list of features accessible by the WFS. This section is used to discover services based on the features served. It is also useful to enable clients to understand the extent (spatial/temporal) of the features available, default and alternate coordinate reference systems (CRS) that can be transformed into on request.

The feature type list provides general information about each feature type offered by the WFS. It contains the following mandatory and optional properties:

Element Name	Description	Obligation (WFS 2.0 clause 8.3.4)
Name	Namespace-qualified name of the feature type. Example aixm:Airspace	M
Title	Unordered list of 0..* human-readable titles that can be used to identify the feature type in menus. If more than one wfs:Title element is listed each title shall have a different xml:lang attribute	O
Abstract	Unordered list of 0..* descriptive narrative for about the feature type.	O
Keywords	Short classification value of the feature type to support discovery	O
Default CRS	Defines the default coordinate reference system used to encode geometry properties of a feature types if the srsName parameter is not used in a GetFeature or GetPropertyValue request. Note: the default CRS may not be in the same as the CRS used in the underlying datastore if the WFS supports CRS transformation.	M
Other CRS	If the WFS supports CRS transformation one or more wfs:OtherCRS may be declared as being allowed values for the srsName parameter.	O
NoCRS	If a feature type has no spatial properties, the NoCRS element shall be used.	O
OutputFormats	List of MIME types indicating any alternate output formats that may be generated for a feature type. If this element is not specified, then the output format declared in the GetFeature operation shall be used.	O
WGS84BoundingBox	Lower left and upper right latitude and longitude coordinate values in WGS84 decimal degrees defined the approximate overall extent of the data available for the specified feature type.	O
MetadataURL	This is a XLink reference containing a URL to one or more metadata records providing more detailed description of the feature type.	O
ExtendedDescription	A service provider may wish to include further information describing the feature type. The extended description element provides a mechanism for adding further information without having to redefine the capabilities schema.	O

The following recommendations were identified to improve discovery and evaluation:

Recommendation: WGS84 Bounding Box Definition

- The approximate extent should be declared using a bounding box that is as small as possible. If the extent of a feature type is not global then the default global extent shall not be used.
- If the bounding box crosses the 180 meridian, then the value of the westBoundLongitude will be greater than the eastBoundLongitude value.

Discussion Point: Declaring available timeslice interpretations for each feature type

During OWS-8, a requirement was identified for client applications to have the ability to discover which kinds of timeslice are available via the WFS. Therefore, it was discussed that the WFS should advertise which timeslice interpretations are available for each feature in the GetCapabilities.

The wfs:ExtendedDescription element in the wfs:FeatureType element is an extension mechanism that could used to specify which types of timeslice interpretation are available for individual features. This would not require any extensions to the WFS specification, to implement this would only need an agreed domain definition.

Example:

```
<wfs:FeatureType>
<wfs:Name>aixm:AirportHeliport</wfs:Name>
<wfs:Title>AirportHeliport</wfs:Title>
<wfs:Abstract>GO Publisher mapping from database table AIRPORTHELIPORT</wfs:Abstract>
<ows:Keywords>
<ows:Keyword>AirportHeliport</ows:Keyword>
<ows:Keyword>AIRPORTHELIPORT</ows:Keyword>
</ows:Keywords>
<wfs:DefaultCRS>urn:ogc:def:crs:OGC:1.3:CRS84</wfs:DefaultCRS>
<wfs:OtherCRS>urn:ogc:def:crs:EPSG::4326</wfs:OtherCRS>
<ows:WGS84BoundingBox>
<ows:LowerCorner>-180 -90</ows:LowerCorner>
<ows:UpperCorner>180 90</ows:UpperCorner>
</ows:WGS84BoundingBox>
<wfs:ExtendedDescription>
<wfs:Element name="TimesliceInterpretation" type="xsd:string">
<ows:Metadata xlink:href="http://www.someserver.com/AboutTimesliceInterpretation.html"/>
<wfs:ValueList>
<wfs:Value>BASELINE</wfs:Value>
<wfs:Value>TEMPDELTA</wfs:Value>
<wfs:Value>PERMDELTA</wfs:Value>
<wfs:Value>SNAPSHOT</wfs:Value>
</wfs:ValueList>
</wfs:Element>
 </wfs:ExtendedDescription>
</wfs:FeatureType>
```

5.2.1.6 Filter Capabilities

The Filter Capabilities section is used to advertise the filter expressions and operands that are supported by the WFS that may be used to form query predicates. The Filter Capabilities section is comprised of the following mandatory and optional parts:

1. **Conformance (Mandatory):** this section is required to declare conformance to the FE 2.0 ISO 19143 Conformance classes.
2. **ID Capabilities:** this section is used to declare support for the filter operator: resourceID
3. **Scalar Capabilities:** this section is used to declare support for comparison and logical filter operators
4. **Spatial Capabilities:** this section is used to declare support for geometry operands and spatial filter operators
5. **Temporal Capabilities:** this section is used to declare support for temporal operands and temporal filter operators
6. **Functions:** this section is used to declare support for any function operators
7. **Extended Capabilities:** this section is used to declare support for any extended capabilities that the server may offer which are not defined in WFS/FE 2.0 specifications

Discussion Point: Declaration of support for retrieval of SNAPSHOT timeslices

In section 5.2.1.5, a recommendation is proposed to used the wfs:ExtendedDescription element to declare that SNAPSHOT timeslices can be retrieved for a feature type. However, it was discussed that this may not be the most appropriate place in the GetCapabilities document to declare support for SNAPSHOT timeslices.

The fes:ExtendedCapabilities section may be a more appropriate place to declare support for retrieving SNAPSHOT timeslices as these are intended to be generated on request by the WFS. It was argued that SNAPSHOT timeslices should not be mixed with BASELINE, PERMDELTA or TEMPDELTA timeslices which are persistent.

Declaring support for SNAPSHOTs in the fes:ExtendedCapabilities would enable the server to also declare the ability for query for SNAPSHOTs by time instant and/or by time period.

Before a formal recommendation can be made further discussions about how SNAPSHOTs are handled by the WFS are required as implementations of SNAPSHOT support within a WFS are not yet mature.

5.2.2 DescribeFeatureType

The DescribeFeatureType operation returns an application schema (.xsd) describing how feature types accessible via the WFS should be encoded in a response (GetFeature, GetPropertyValue) or on input via a transaction action (Insert).

The returned application schema must support the outputFormat value of ‘application/gml+xml; version=3.2’ as each version of the WFS specification is bound to a specific version of the GML specification.

If the data specification has also defined GML application schemas in conforming to previous versions of GML (e.g. 3.1.1) these may be requested if declared in the GetCapabilities.

Example DescribeFeatureType request/responses are available in the supporting zip in the DescribeFeatureType directory.

5.2.3 ListStoredQueries

The ListStoredQueries operation returns a response listing all of the stored queries available on a server. All WFS shall support at least one stored query: getFeatureByID.

The ListStoredQueries response provides the following information:

- **id**: unique, persistent identifier assigned to the stored query. This id is used to call the the stored query in a GetFeature or GetPropertyValue request
- **Title**: optional human readable title identifying the stored query
- **ReturnedFeatureType**: one or more feature types that the stored query expression operates on

Example ListStoredQueries request/responses are available in the supporting zip in the ListStoredQueries directory.

5.2.4 DescribeStoredQueries

The DescribeStoredQueries operation request can be used to generate a response providing a detailed description of either all stored query expressions or one or more specified stored query expressions that are available on the server.

The DescribeStoredQuery response contains the following parameters:

Property	**Description**	**Obligation (WFS 2.0 clause 14.4.4)**
Id	Unique, persistent identifier assigned to the stored query	M
Title	Title that describes and identifies the stored query	O
Abstract	Brief narrative describing the stored query	O
Parameter	Definition of the name, data type and narrative summary describing each parameter defined for the stored query	M
queryExpressionText	Contains the text of the stored query expression. The text is expressed in some implementation language (e.g. wfs:Query). NOTE: If the queryExpressionText was set to "private" in the CreateStoredQuery request then the wfs:Query shall be excluded from the DescribeStoredQueries response.	O

Example DescribeStoredQueries request/responses are available in the supporting zip in the DescribeStoredQueries directory.

5.3 Query Operations

The WFS 2.0 interface provides two operations for querying the data store to retrieve data:

1. **GetFeature:** returns a response containing a selection of zero or more features corresponding to the criteria defined in the request
2. **GetPropertyValue:** returns a response containing zero or more of the selected feature property values that corresponds to query criteria defined in the request

Both the GetFeature and GetPropertyValue operations support two types of query:

- **wfs:Query:** these are an ad hoc queries generated by a client to retrieve specified feature types or property values
- **wfs:StoredQuery:** this is a pre-defined parameterised query that has been stored on the server for re-use by clients

5.3.1 GetFeature

The GetFeature operation request is used to retrieve features using one or more ad hoc queries (wfs:Query) or stored queries (wfs:StoredQuery). The GetFeature request may also contain the following parameters:

- **Standard Presentation Parameters:** these can be used defining how the result set is presented in the response (Table 5-1)
- **Resolve Parameters:** if the server supports Remote Resolve, these parameters can be used to whether the server should resolve references and include any associated features in the response (Table 5-2)

Recommendation: Support for Standard Presentation Parameters

outputFormat and resultType have default values defined for them which will not be changed for AIXM so these parameters may not need to be used. To date no operational requirements have been identified to support resultType=hits, so this can remain an optional element.

If a WFS constrains the maximum number of features to be returned for a request, as advertised in the CountDefault capacity constraint in the GetCapabilties document, then the startIndex should be implemented. This is important as the startIndex request parameter enables the client to re-submit the request to retrieve any remaining features where the response does not contain all features corresponding to a request.

Table 5-1. GetFeature Operation Standard Presentation Parameters

Parameter	Description
count	This parameter is used to limit the number of features to be included in the response. There is no predefined default value and if this value is absent then the server shall return all features corresponding to the request query expression
startIndex	Indicates the index from which the server shall begin presenting the results in the response. The default value is 1if not specified. This is typically only required where a WFS constrains the maximum number of features contained in the response, as defined by the CountDefault in the capacity constraints in the GetCapabilities to enable the retrieval of all of the features that correspond to a query.
outputFormat	Specifies the format used to encode the response. The default value is "application/gml+xml; version 3.2" if not specified. This parameter should only be required where a server supports the ability to encode the result in multiple formats.
resultType	If the ResultType parameter is not expressed, then the response shall contain a wfs:FeatureCollection containing all features corresponding to the request query expression. If the resultType=hits then the response shall be a count of the number of features corresponding to the request query expression. This is useful to identify how many features may be returned before submitting the response.

Recommendation: Support for Standard Presentation Parameters

outputFormat and resultType have default values defined for them which will not be changed for AIXM so these parameters may not need to be used. To date no operational requirements have been identified to support resultType=hits, so this can remain an optional element.

If a WFS constrains the maximum number of features to be returned for a request, as advertised in the CountDefault capacity constraint in the GetCapabilties document, then the startIndex should be implemented. This is important as the startIndex request parameter enables the client to re-submit the request to retrieve any remaining features where the response does not contain all features corresponding to a request.

Table 5-2. GetFeature Operation Resolve Parameters

Parameter	Description
resolve	Controls whether and which resource references are to be resolved (local, remote, all or none). NOTE 1: If no resolve parameters are specified then the server shall not resolve any references by default NOTE 2: If resolve parameters are defined at the per-property level, these shall supercede any values defined per-operation
resolveDepth	resolveDepth defines the depth to which nested references shall be resolved. For example, if a feature contains a property that contains a reference to another feature, which contains a property that references another resource then this property specifies whether or not to resolve the properties in the nested referenced resources.
resolveTimeout	Controls how long a server shall wait to receive a response when resolving resource references. This is specified in seconds
resolvePath	Limits the resource resolution to the property specified for the resolvePath NOTE: only applicable when applied to a specific property and will supercede any resolve parameters defined at the feature level.

5.3.2 wfs:Query

The wfs:Query element is used to define an ad hoc query to request one or more feature types from the data store. A wfs:Query is comprised of the following parameters:

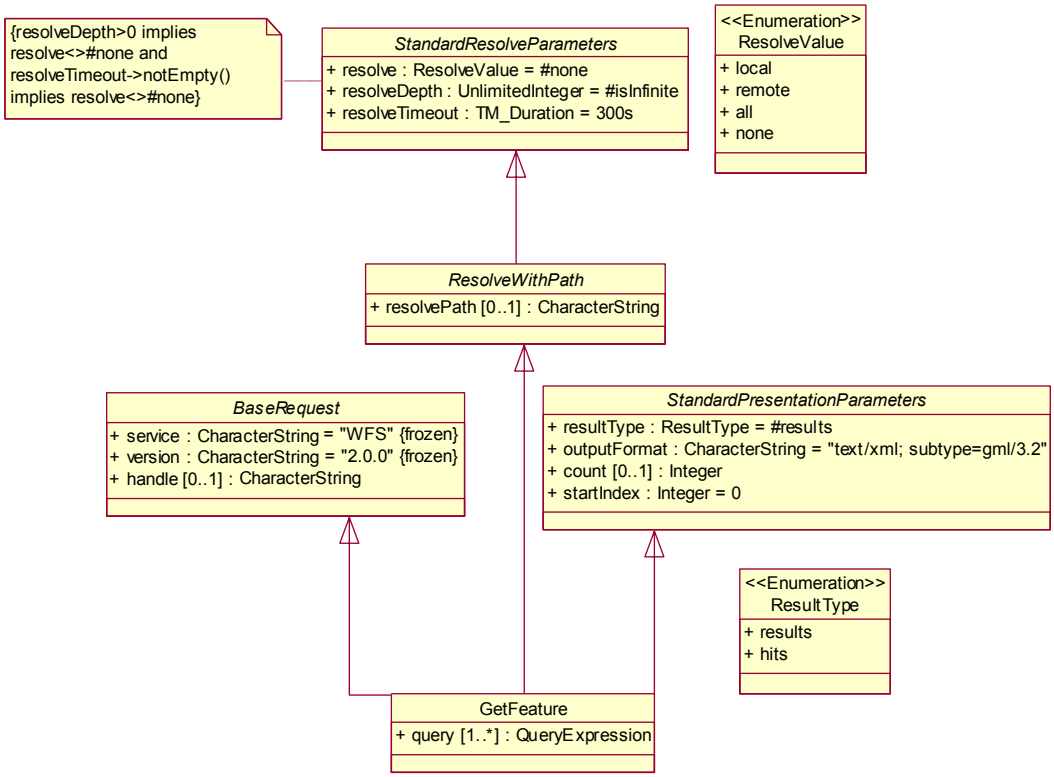


Figure 5-1 GetFeature request (from ISO 19142 (Figure 17))

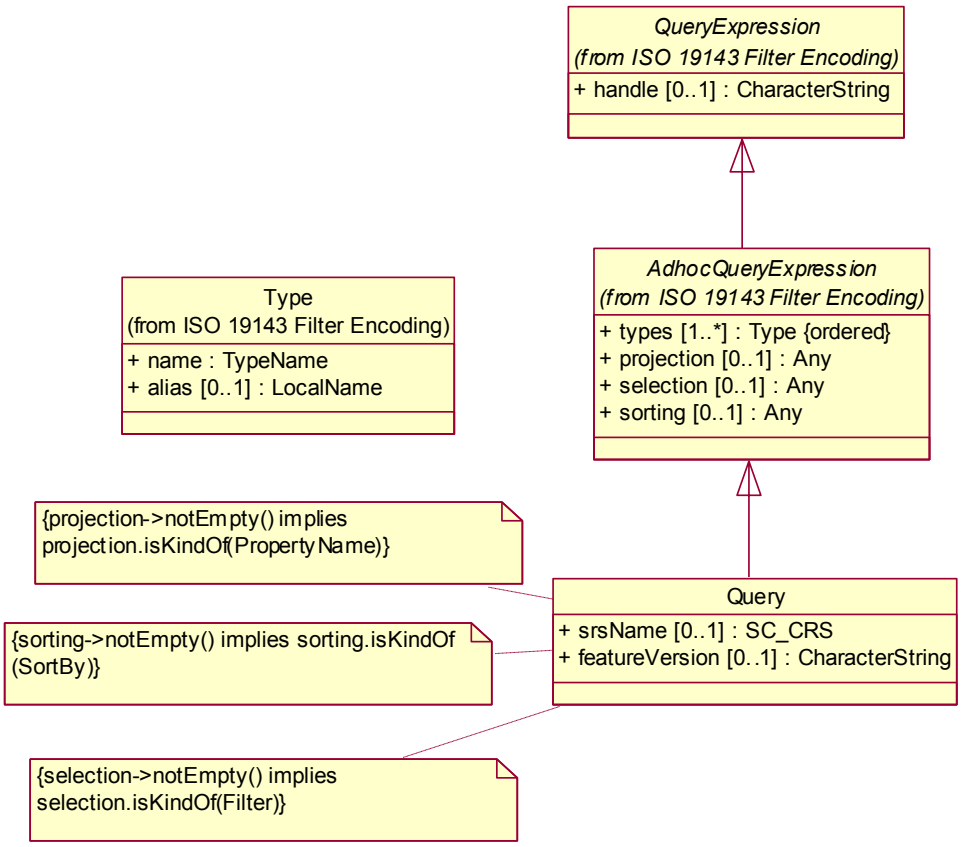


Figure 5-2. Ad-hoc query expression (from ISO19142 (Figure 8))

Query Parameter		**Description**
wfs:Query attributes	**typeNames**	TypeNames is the only mandatory parameter and is used to specify which feature types should be included in the response.
	aliases	The optional aliases parameter is used to specify a list of alternate names for the feature types defined in the TypeNames. A feature type alias may be used anywhere the feature type name may be used within the context of a query expression but is commonly used in Join Queries.
	featureVersion	If a server advertises that it implements feature versioning in its GetCapabilities, the featureVersion parameter can be used to define a specific version of a feature. NOTE: support for feature versioning is not required for any WFS serving AIXM 5.1 as the WFS does not support versioning in line with the AIXM Temporality Model.
	srsName	If a server advertises that the data can be returned in multiple coordinate reference systems (CRS) in the GetCapabilities, the srsName parameter can be used to specify the preferred CRS.

Query Parameter		Description
		If no srsName is defined then all data shall be returned in the CRS defined as the defaultCRS in the GetCapabilities. NOTE 1: this property is not supported by KVP encoded requests NOTE 2: the WFS shall support the processing of srsName attribute values encoded using the following format: urn:ogc:def:objectType:authority:version:<EPSG code> Example: srsName=”urn:ogc:def:crs:EPSG::4326”
Projection Clause	**wfs:PropertyName**	The projection clause is used to define a subset of optional feature properties to be included in the feature contained in the response. A list of one or more optional properties are defined using PropertyName parameter. The projection clause has been extended in the WFS specification to allow resolve parameters to be defined at the property level which supercede any resolve parameter values defined at the operation level,
Selection Clause	**fes:Filter**	The selection clause is used to select a subset of feature from the feature types specified in the typeNames parameter. In XML encoded requests, the fes:Filter element can contain one or more logical, comparison, spatial, or temporal filter expressions.
Sorting Clause	**fes:Sort**	The sort clause can be used to specify the order in which the features are presented the response. One or more parameters can be specified to order the features. The order can be specified as either ascending or descending. If the sort order is not specified then the default order shall be ascending. If the sort clause is not included in a request then the response can output the features in any order. NOTE: Sorting is only supported for single query expressions

Within OWS 7 and 8 and the FAA SAA Dissemination Pilot a wide range of example queries were defined based on typical flight dispatch and planning user cases. Example queries are provided in the Ad Hoc Queries directory in GetFeature directory in the supporting zip file.

5.3.3 wfs:StoredQuery

The wfs:StoredQuery element is used to select features using a parameterized pre-defined query. All WFS shall support at least one stored query: getFeatureByID. This stored query allows the user to retrieve a single feature based on its gml:id.

Example getFeatureByID queries are provided in the Stored Query directory in the GetFeature operation example directory in the supporting zip.

Additional Example stored queries are provided in the Stored Query directory for the following stored queries:

- getFeatureByIdentifier: which returns a single feature based on its gml:identifier
- getAirportHeliportByDesignator: which returns a single AirportHeliport feature based on its aixm:designator value.

5.3.4 GetPropertyValue

The GetPropertyValue operation allows the property value or part of property value for a complex property defined in the valueReference parameter to be retrieved for a set of features identified using a query expression (Figure 5-3).

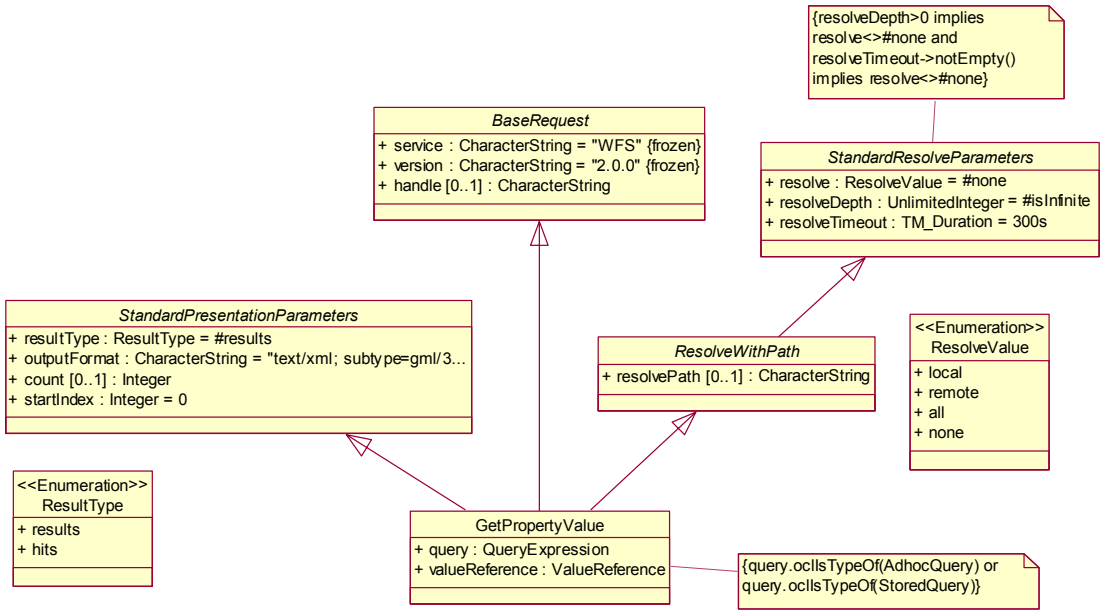


Figure 5-3. GetPropertyValue Operation (from ISO 19142 (Figure 15))

Example GetPropertyValue queries are provided in the GetPropertyValue directory in the supporting zip. These were developed within the FAA SAA Pilot and were designed to replicate example SAA web service operations.

5.4 Manage Stored Query Operations

Manage stored query operations enable a service provider to open up the WFS to allow users to create and drop stored queries on the server. Enabling users to define their own stored queries offers several benefits:

1. Allows client developers to define stored queries for common requests that may be easier to implement than complex ad hoc queries
2. Removes reliance on the service provider to define and implement specific stored queries for individual clients, enabling autonomy

3. If stored query expressions are defined using the wfs:Query language these can be shared across WFS

Although a WFS service provider may support manage stored query operations, they may not be openly accessible. A service provider may choose to restrict these operations to authorized users only to protect the service.

5.4.1 CreateStoredQuery

A stored query is created using the optional CreateStoredQuery operation. The CreateStoredQuery operation can be encoded in XML only.

The CreateStoredQuery request must contain a wfs:StoredQueryDefinition. The wfs:StoredQueryDefinition contains the same parameters as defined in the DescribeStoredQueries response (see section 5.2.2).

By default the stored query expression should be defined as a wfs:Query. However, the WFS 2.0 specification allows stored query expressions to be defined using other query languages such as SQL.

5.4.2 DropStoredQuery

The optional DropStoredQuery operation is a request that deletes a specified stored query from the server.

Example DropStoredQuery XML encoded request/response are available in the supporting zip in the DropStoredQuery directory.

Change Request: Keywords for DropStoredQuery KVP encoding (Request ID: 186.)

A change request has been submitted to correct the WFS 2.0 document to define the correct keywords for the DropStoredQuery which have been incorrectly defined as the DescribeStoredQueries Keywords in ISO 19142 clause 14.6.3:

Table 5-3 — Keywords for DropStoredQuery KVP encoding

URL Component	O/M[a]	Description
Common Keywords (REQUEST= DescribeStoredQueries)		See Table 7 (Only keywords for all operations or the DescribeStoredQueries operation.)
STOREDQUERY_ID	M	A comma-separated list of stored query identifiers to describe. If the keyword is not specified then all stored queries offered by a server shall be described.
[a] O = Optional, M = Mandatory		

5.5 Transaction Operations

The optional Transaction operation is used to maintain features in the underlying data store. The transaction operation supports the ability to Insert (create), Update (modify), Replace or Delete features.

When the transaction has been completed, a response document shall be returned to the client indicating the completion status of the operation.

Example Transaction operations for inserting pending and approved SAA Airspace Activation schedules are available in the supporting zip in the Transaction directory.

Recommendation: Support Transaction Insert Action only

The AIXM 5 Temporality Model has been defined so that an AIXM feature is never deleted, replaced or updated. If a change occurs to a feature a new feature containing either a TEMPDELTA or PERMDELTA timeslice is inserted into the database which defines:

- Properties that have changed
- Cancel, disapprove or correct erroneous property values
- Define end of a feature lifecycle

Consequently, a WFS 2.0 shall support the Insert transaction action, only.

6 Configuring a WFS 2.0 for retrieving AIXM 5.1

6.1 Introduction

How the WFS 2.0 is configured will impact how a user can retrieve features encoded in AIXM 5.1. It is important that the service provider has a good understanding of the end user requirements for the WFS instance as this will determine how the WFS should be considered. There are a number of key requirements that must be identified:

1. Which types of timeslice should be provided?
2. Does the WFS need to support SNAPSHOT timeslices?
3. Should users be able to query only the current history (valid now and future) or provide access to past history?

Once these high-level requirements are understood, the service provider must then also consider other issues such as how to handle resource identifiers, feature associations, support for non simple features geometry types and enabling schema validation. Recommendations and guidance for handling these issues shall be provided in the following sections.

6.2 Encoding AIXM 5.1 Features within an OGC WFS

The WFS 2.0 specification is designed to enable users to retrieve features using the GetFeature operation or individual property values of a feature using the GetPropertyValue operation. The WFS implements the ISO 19109 Generic Feature Model (GFM) where each instance (or version) of a feature in the WFS represents the state of a real-world object at a specific time instant or time period[1].

[1] NOTE: In some GML application schemas, temporal properties may not be defined so it is implied that the feature represents the "current" state of the real-world object.

This implementation model is does not correspond with the AIXM 5.1 Temporality model, where an aeronautical feature can be represented in different ways: static vs dynamic representation.

The AIXM 5.1 Temporality Model has been loosely built upon the GML Dynamic Feature model where dynamic, non-persistent properties may be contained within a timeslice property to represent the history of feature over a specific time period. The AIXM 5.1 Temporality Model has adopted this timeslice concept which has been applied to both static and dynamic representations of aeronautical features (Table 6-1).

Table 6-1. AIXM 5.1 timeslice interpretations

	Timeslice	Description
State	BASELINE	Contains a complete set of property values defining the state of a feature as a result of permanent change
	SNAPSHOT	Contains a complete set of property values defining the state of a feature that overlays any temporary change onto the baseline state
Event	PERMDELTA	Contains only the property values that have permanently changed
	TEMPDELTA	Contains only the property values that have changed due to a temporary event (e.g. runway closure due to snow, military airspace activation)

The AIXM 5.1 Temporality Model poses a challenge for configuring the WFS due to the conceptual differences between the GFM where each instance of a feature represents a real-world object at a specific time versus AIXM 5 where each aeronautical object is represented by an AIXM feature containing one or more timeslice properties.

6.2.1 Representing the temporality of aeronautical features in a WFS

The AIXM Temporality Model provides flexibility in how an AIXM feature can be encoded within a WFS. For example, an AIXM feature may contain:

1. A single timeslice containing properties representing either the state or event at a particular time instant or period
2. Multiple timeslices representing the history of the object for a particular time period
3. Multiple timeslices representing both the state and event at a particular time (e.g. Digital NOTAM containing a TEMPDELTA and SNAPSHOT)

This flexibility enables AIXM 5.1 support the wide range of use cases (i.e. charting, Digital NOTAM, e-AIP) therefore it is essential when configuring a WFS the end use application(s) are well understood.

Examples:

1. **Digital NOTAM:** may require AIXM features containing only TEMPDELTAs or TEMPDELTA and SNAPSHOT timeslices
2. **Charting:** may require AIXM features containing only BASELINE timeslices
3. **Flight Dispatch and Planning:** may require AIXM features containing either BASELINE, TEMPDELTA or SNAPSHOT timeslices

The WFS/FE 2.0 specification does not currently support the ability for a user to submit a request where the response contains a single AIXM features containing a subset of one or more timeslice properties that corresponds to the end user request criteria (see section 7.2 for proposed improvements). Therefore, configuration of the WFS is really important to ensure that users can effectively retrieve the information they need.

Example: Flight Dispatch and Planning Application

A user may submit a request for TEMPDELTA timeslices for a specified AIXM feature valid between t1 and t2 to identify whether any changes have occurred that may impact a flight.

To ensure that the user retrieves only the TEMPDELTA timeslices that intersect the user defined valid time period, each TEMPDELTA timeslice should be encoded in a separate instance of an AIXM feature within the WFS.

Recommendation: The WFS should be configure to enabled the user to retrieve only the timeslices that correspond to the query expression

To enable effective retrieval of individual or sets of timeslices, the WFS should be configured to ensure that the user only retrieves the data they request.

This may require each timeslice to be encoded within a separate instance of a feature.

NOTE: this introduces an issue with how to handle feature identifiers as each feature must be assigned a unique, persistent resource identifier (i.e. gml:id) within the WFS. This shall be discussed in section Handling resource identifiers

6.2.2 Handling history

A WFS can be configured to provide access to the full history of a feature. However, this is not a requirement in most aeronautical use case scenarios such as flight planning and dispatch. In this case, users will only want to know about the operational situation now and in the future. Therefore, the WFS does not need to provide access to any timeslices that are no longer valid.

6.2.3 Timeslice version handling

A WFS can be configured to provide access to all versions of a timeslice available in an underlying datastore. While this may be useful for auditing requirements, there is often no requirement for accessing previous versions of a timeslice by end users. Therefore, the WFS shall be constrained to provide access to the latest version of a timeslice.

Recommendation: Only the timeslice with the highest correctionNumber should be accessible via the WFS

The WFS must be configured to provide access to the timeslice with the highest correction number for TEMPDELTA, PERMDELTA and BASELINEs.

6.3 Handling resource identifiers

The recommendation to encode different types of timeslice interpretation within different instances of a feature and also to restrict each feature instance to only contain a single timeslice means that the handling of resource identifiers for AIXM features within a WFS must differ from the guidance provided in the AIXM 5 Feature Identification and Reference Guide.

In subclause 7.2 of the WFS 2.0 specification it states that:

"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."

"For features encoded using GML, the resource identifier shall be encoded in the XML attribute gml:id."

All types that extend from GML object (i.e. aixm:TimeSlice, gml:TimePeriod, gml:Point, aixm:SurfaceCharacteristics, aixm:AirspaceVolume, etc.) are also required to have a gml:id, which is intended as a local unique identifier, within the XML data set.

Lexical Rule for gml:id

The gml:id has to comply with the same rules as any other XML ID attribute. It has to be unique within the XML file and must start with a letter.

6.3.1 Managing feature identifiers

In the AIXM 5 Feature Identification and Reference Guide, it recommends that the feature identifier (i.e. gml:id) assigned to AIXM 5 features should re-use the UUID, prefixed with "uuid".

Example

```
<aixm:Airspace gml:id="uuid.a82b3fc9-4aa4-4e67-8def-aaea1ac595j">
<gml:identifier
codeSpace="urn:uuid:">a82b3fc9-4aa4-4e67-8defaaea1ac595j</
gml:identifier>
<aixm:timeSlice>
<aixm:AirspaceTimeSlice gml:id="ID00001">
...
</aixm:Airspace>
```

Unfortunately, within the WFS this approach will not satisfy the requirement for uniqueness, as there are multiple instances of the AIXM feature accessible via the WFS. It is there proposed that the gml:id should be defined in the following ways:

Recommendation: Assigning Feature identifiers to features containing BASELINE, PERMDELTA or TEMPDELTA timeslices

It is recommended that the gml:id assigned to features should consist of 5 parts:

["uuid"]. [UUID].[Timeslice Interpretation Code].[sequenceNumber].[correctionNumber]

Timeslice Interpretation Code

B = BASELINE
T = TEMPDELTA
P = PERMDELTA

Example

```
<aixm:Airspace gml:id="uuid.a82b3fc9-4aa4-4e67-8def-aaea1ac595j.B.2.3">
<gml:identifier
codeSpace="urn:uuid:">a82b3fc9-4aa4-4e67-8defaaea1ac595j</
gml:identifier>
<aixm:timeSlice>
<aixm:AirspaceTimeSlice gml:id="ID00001">
<gml:validTime>
<gml:TimePeriod gml:id="ID00002">
<gml:beginPosition>2010-06-29T17:31:00</gml:beginPosition>
<gml:endPosition>2010-06-29T19:00:00</gml:endPosition>
</gml:TimePeriod>
</gml:validTime>
<aixm:interpretation>BASELINE</aixm:interpretation>
<aixm:sequenceNumber>2</aixm:sequenceNumber>
<aixm:correctionNumber>3</aixm:correctionNumber
...
  </aixm:Airspace>
```

NOTE: if the WFS only provides access to a single version of the feature then the "uuid".UUID approach can be used.

Open Issue: Assigning Feature identifiers to features containing a SNAPSHOT timeslice

In the AIXM 5 Temporality Model it states that SNAPSHOT timeslices are generated dynamically on request. An open issue remains as to how the WFS should assigned unique feature identifiers to the feature containing a SNAPSHOT timeslice.

6.3.2 Managing object identifiers

No guidance has been provided for managing the object identifiers that must be assigned to feature properties (i.e. aixm:Timeslice, gml:TimePeriod).

There is no requirement for these object identifiers to be assigned a persistent unique identifier so these identifiers can be autogenerated on request by the WFS. To keep file sizes small, it is recommended that identifiers are kept succinct.

The only time that an object identifier may need to be assigned a persistent unique identifier are those assigned to geometry properties as these may be referred to in other geometries.

6.3.3 Handling feature identifier uniqueness constraints in WFS queries

The WFS 2.0 specification enables the user to submit multiple queries within one request. The response are multiple sets of features, one for each query. If a feature is part of more than

one result set, the uniqueness constraint of the gml:id of the feature cannot be fulfilled. Where a complete feature is returned multiple times in the response it has to be encoded first inline, then referenced in subsequent occurrences using XLink. However, this is only possible if all feature instances in the result are identical – which might not be true if different projection clauses are in place. An example (the most simple one showing the issue) is given in the following.

Given a WFS feature store containing the following feature with two optional properties propA and propB:

```
<Feature gml:id="id1">
  <propA>…</propA>
  <propB>…</propB>
</Feature>
```

Now consider a GetFeature request for retrieving this feature with different projection clauses, one for propA and one for propB:

```
<wfs:GetFeature>
   <wfs:Query>
     <wfs:PropertyName>propA</wfs:PropertyName>
     <fes:Filter>
       <fes:ResourceId rid="id1"/>
     </fes:Filter>
   </wfs:Query>
   <wfs:Query>
     <wfs:PropertyName>propB</wfs:PropertyName>
     <fes:Filter>
       <fes:ResourceId rid="id1"/>
     </fes:Filter>
   </wfs:Query>
</wfs:GetFeature>
```

It is not possible to generate a response where the first instance of the feature is encoded inline, while the second is referenced to the first as the user has requested different representations of the feature in each query. Therefore, the two features are not the same and the expected response would be the following:

```
<wfs:FeatureCollection>
   <wfs:member>
     <wfs:FeatureCollection>
       <wfs:member>
         <Feature gml:id="id1">
           <propA>…</propA>
         </Feature>
       </wfs:member>
     </wfs:FeatureCollection>
   </wfs:member>
   <wfs:member>
     <wfs:FeatureCollection>
       <wfs:member>
         <Feature gml:id="id1">
           <propB>…</propB>
         </Feature>
       </wfs:member>
     </wfs:FeatureCollection>
   </wfs:member>
</wfs:FeatureCollection>
```

However, this is not possible as it breaks the document-uniqueness constraint on the gml:id attribute.

Several solutions to this issue were identified but no approach has been agreed to resolve it so it remains an open issue.

Possible solutions proposed include:

1. Remove the requirement for the gml:id to be mandatory on objects[2]
2. Assign a suffix to the gml:id to ensure uniqueness

```
<wfs:FeatureCollection>
   <wfs:member>
     <wfs:FeatureCollection>
       <wfs:member>
         <Feature gml:id="id1-1">
           <propA>…</propA>
         </Feature>
       </wfs:member>
     </wfs:FeatureCollection>
   </wfs:member>
   <wfs:member>
     <wfs:FeatureCollection>
       <wfs:member>
         <Feature gml:id="id1-2">
           <propB>…</propB>
         </Feature>
       </wfs:member>
     </wfs:FeatureCollection>
  </wfs:member>
</wfs:FeatureCollection>
```

3. Do not allow users to submit multiple queries in a single request
4. Prohibit the use of the project clause in a wfs:Query

6.4 Handling Feature Associations

The AIXM 5.1 data specification defines inter-relationships between many AIXM features which are encoded as feature associations using XLink. The AIXM Feature Association and Reference Guide provides recommendations and guidance for encoding feature associations for AIXM 5.1, however, these are not completely applicable for use within the WFS.

6.4.1 Encoding feature associations

Feature associations can be encoded in two ways:

1. **Concrete local references:** the xlink value contains a reference to the local version of the feature available to the service
2. **Concrete external references:** the xlink value contains a resolvable Universal Resource Locator (URL) that the consumer of the message can follow directly to retrieve the feature

6.4.1.1 Concrete local references

Concrete local references can be implemented by including a reference to the gml:id of the associated feature.

[2] Please note that this issue might not be limited to gml:ids on features. If projection clauses are introduced that change the representation of feature properties, it is also present for any gml object that might occur in different representations in the same response document.

Example:

```
<aixm:clientAirspace xlink:href="uuid.a82b3fc9-4aa4-4e67-8def-
aaea1ac595j.B.2.3"/>
```

6.4.1.2 Concrete external references

In a WFS, the value of the XLink containing a concrete external reference can contain a HTTP GET request such as the GetFeatureByID stored query.

Example:

```
 <aixm:clientAirspace xlink:href="http://someAIXMserver.com/wfs?
service=WFS&version=2.0.0&request=GetFeature&STOREDQUERY_ID=urn:ogc:def:que
ry:OGC-WFS::GetFeatureById&ID= uuid.a82b3fc9-4aa4-4e67-8def-
aaea1ac595j.B.2.3"/>
```

Note: the XLink contains a reference to a specific instance of a feature timeslice type and version.

6.4.2 Supporting reverse associations

In AIXM 5.1, associations between features are implemented in a uni-directional way. Similar to the normalization in the theory of relational databases, this ensures smaller, well-structured relations in the model. However, these uni-directional relationships mean that requesting data based on associations within the WFS difficult.

In OWS-7, an extension schema was developed by Eurocontrol containing "reverse associations" enable bi-directional relationships between features to defined

In OWS-8, further investigation of this approach revealed some issues related to the server side handling of the redundant data:

- if an association changes, new time slices for the feature
  - loosing the association
  - and/or gaining the association

  must be generated and inserted into the WFS data store to ensure data consistency. It's not yet specified if the client or the server is responsible of providing these time slices. This also implies that there may exist delta[3] time slices solely for the purpose of updating reverse associations. Because the list of reverse associations is a complex property, all existing associations have to be repeated in that time slice. This requires extra merging effort.

- if multiple reverse associations change at the same time, they have to be merged into one time slice. This is because no two time slices with the same valid time and interpretation may exist for a feature. The merging may become difficult to synchronize for the server if the data is inserted from different sources.

[3] in most cases a PERMDELTA, as a TEMPDELTA is less likely to carry a change in associations

- the list of reverse associations of a feature is only distinct for a time instant. This implies that only SNAPSHOT time slices can carry reverse associations.

  Example:

  Given
  - a RadarSystem R
  - an OrganisationAuthority O1
  - an OrganisationAuthority O2

  and the following time slices:

```
BASELINE 1
  RadarSystem R
  validTime: May - August
  associations: office -> O1

BASELINE 2
  RadarSystem R
  validTime: August - December
  associations: office -> O2

BASELINE 3
  OrganisationAuthority O2
  validTime: January - December

BASELINE 4
  OrganisationAuthority O1
  validTime: January – June
```

  What are the reverse associations of the BASELINEs 3 and 4? There is no answer that is valid for the whole validTime of the requested BASELINE. This breaks the contract of the meaning of validTime.

The above issues show that the introduction of reverse associations increases the complexity of the AIXM model itself and especially an AIXM data store implementation. Further testing is needed including the mentioned use cases to test whether this does not outweigh the benefits for clients.

However, the mentioned issues only apply to the introduction of reverse association as a regular property of a feature. If instead only SNAPSHOTs may carry them, they could be generated on request by the server. The result is well defined.

Some features contain a lot of associations (e.g. an AirportHeliport is linked to more than 30 other feature types). To save extra processing time on the server, the client should be enabled to pass a parameter containing all reverse references required in the response of that specific request.

6.5 Handling non Simple Feature Geometry Types

The Aviation GML Profile defines the geometry types that should be supported by applications supporting AIXM 5.1. A number of non-simple feature geometry types are defined in this profile: arcByCentrePoint, circleByCentrePoint, geometries containing

references to remote geometry properties. These geometry types do not have a corresponding spatial data type that is supported by any relational databases. This causes an issue for WFS implementations that rely on querying spatial data types stored in the database when performing spatial queries.

Where this occurs, the service provider must derive a simple feature geometric representation of the geometry to support spatial filtering within the WFS. Two types of approach have been identified to derive a simple feature geometric representation:

1. **Densification:** where the original geometry type is transformed into linestring or surface geometry
2. **Bounding Box:** where an approximate extent is defined to represent the geometry

Both approaches were implemented by different WFS service providers in the OWS Interoperability Experiments and FAA SAA Dissemination Pilot. Differences in the level of accuracy of a response to a spatial query were observed. The densification approach being more accurate than the bounding box approach which introduced errors of commission in the response (i.e. more features were returned).

When implementing a solution for deriving the geometry, service providers need to balance the trade off between performance and the accuracy of the response. Performance of spatial operations is dependent on the complexity of the geometry. A spatial operation will perform faster on geometries containing fewer coordinates, therefore if the derived geometry is less complex the request can be executed quicker.

Recommendation: A minimum acceptable accuracy level should be defined for a queries performed on non-simple feature geometry types

Eurocontrol and FAA should define a minimum acceptable accuracy level for the response of spatial filters that are performed on derived geometries for non-simple feature geometry types to ensure consistency in the number of features returned in response to the same query by different service providers.

6.6 Enabling schema validation

The response of a GetFeature operation is a GML document, therefore, it shall reference the GML application schema that describes the data specification to enable the output to be validated. This reference shall be declared using the schemaLocation attribute. There are several possible approaches for declaring the application schema within the schemaLocation attribute.

1. Reference a definitive, online version of the application schema
2. Include a DescribeFeatureType request to call back to the WFS to retrieve the schema
3. Reference to a local version of the application schemas stored within the WFS or on the application server

It is important that regardless of which approach is implemented, that a client application can access the application schema to enable validation.

If the WFS has been configured to support AIXM plus extensions, a definitive online repository for the extension schemas may not exist so the WFS will need to reference a local

version of the application schema. In this instance, the schemaLocation attribute should not include relative paths that cannot be resolved. The service provider should make the schema accessible in a web accessible folder or include the DescribeFeatureType request in the schemaLocation string.

> **Recommendation: Reference to application schemas in the schemaLocation string must be resolvable**
>
> It is recommended that the schemaLocation attribute such as a WFS DescribeFeatureType request or reference to an online schema repository to ensure that the response can be validated by client applications.

7 Proposed improvements to support retrieval of AIXM 5.1 via an OGC WFS

7.1 Introduction

The aim of this guidance report was to provide recommendations for the implementation and configuration of the WFS 2.0 specification for retrieving AIXM 5.1 based on identified use cases. During the process of developing this report and establishing WFS 2.0 services within the Interoperability Experiments and FAA SAA Dissemination Pilot several improvements and change requests were identified for WFS/FE 2.0 specifications, GML 3.2.1 and AIXM 5.1. This section shall summarise the proposed improvements which require further investigation.

7.2 Improvements for WFS 2.0

7.2.1 Enable response to return a subset of timeslices within a feature

The WFS 2.0 specification should be extended to enable the client to construct subsets of time slices when querying features with a GetFeature request. In the existing feature model of WFS, which is non-aware of the AIXM temporality model, this implies the introduction of new projection clauses. The existing PropertyName projection clause is only capable of including non-mandatory properties of a feature in the response. In the following, projection clauses will be defined that enable the client to exclude time slices based on a FES 2.0 filter expression.

7.2.1.1 Filter Time Slices

A projection clause for filtering the time slices of a feature based on an FE 2.0 filter should exclude canceled or corrected time slices by default. An example:

Query: Give me all PERMDELTAs of a Runway that are valid (i.e. not corrected or canceled) and meet certain criterias

```
<wfs:Query typeNames="aixm:Runway">

  <!-- Projection clause(s) -->
  <wfs-aixm:filterTimeslices>
    <fes:And>
      <fes:PropertyIsEqual>
          <fes:ValueReference>aixm:interpretation</fes:ValueReference>
          <fes:Literal>PERMDELTA</fes:Literal>
      </fes:PropertyIsEqual>
```



```
      [... other criteria on time slice level ...]
    </fes:And>
  </wfs-aixm:filterTimeslices>

  <!-- Filter -->
  <fes:filter>
    <fes:PropertyIsEqual>
        <fes:ValueReference>gml:identifier</fes:ValueReference>
        <fes:Literal>[some UUId]</fes:Literal>
    </fes:PropertyIsEqual>
  </fes:filter>

</wfs:Query>
```

If the client wants to include canceled and corrected time slices, he could do so by adding attributes to the filterTimeslices element:

```
<wfs-aixm:filterTimeslices includeCanceled="true" includeCorrected="true">
      [...]
</wfs-aixm:filterTimeslices>
```

7.2.1.2 Create an Extract

In OGC 11-093r1 the concept of Extracts was introduced. Extracts are the subset of time slices which are relevant for a specific point in time or a time period. Again, a projection clause could be introduced to enable the client to create the extracts:

```
<wfs-aixm:extract encoding="RELEVANT_SET">
  [ <gml:TimePeriod> or <gml:TimeInstant> ]
</wfs-aixm:extract>
```

The reason for the attribute named “encoding” is given in the next section.

7.2.1.3 Creating SNAPSHOTs

In AIXM, SNAPSHOT time slices represent the state of a feature at a given point in time. Thus, SNAPSHOT time slices contain the accumulated information from BASELINE and TEMPDELTA time slices that are valid at given time instant. Because there is an unlimited number of time instants, the server cannot store SNAPSHOTs, instead he has to generate them on request. SNAPSHOT time slices can be seen as virtual contents of a feature, as they are not part of the persistent version of the feature.

The creation of SNAPSHOTs could be done with the projection clause defined above by replacing the encoding attribute:

```
<wfs-aixm:extract encoding="SNAPSHOT">
  [ <gml:TimePeriod> or <gml:TimeInstant> ]
</wfs-aixm:extract>
```

Please note that if a TimePeriod is given, the result is a list of SNAPSHOTs, each of them holding a validTime property containing a TimePeriod. This is an extension of the

SNAPSHOT specification in the AIXM Temporality Model, where SNAPSHOTs are only valid for a point in time. See OGC 11-093r1, section “Snapshot List” for details.

7.2.2 Use case oriented approach to time slice retrieval with WFS 2.0

The existing proposals to extend the WFS and FES specification to support the AIXM temporality model focused on the introduction of new projection clauses and filter functions. New projection clauses where introduced for filtering corrected time slices, cancelled time slices and creating “Extracts” of time slices (e.g. SNAPSHOTs). A new filter function was defined to allow the evaluation of a filter at a given point in time (see section 7.3.1). These extensions provided the necessary flexibility for dealing with the temporality model, however, they added some complexity to the creation of the query – even for common use cases. Several of the new elements have to be combined to achieve the desired result.

Here, an alternative approach is presented that solves the temporality problem on a higher level. This is done by taking a use case oriented approach to the problem and the introduction of a new query type: a temporal query. This simplifies the query language by incorporating common filters. For example, in a temporal query, the filtering of cancelled and corrected time slices is done implicitly.

7.2.2.1 Identifying Use Cases

Use Case 1: Retrieve the full history of the time slices

This is the normal operation of a WFS (non-temporal query) and follows the existing specification. All time slices of the feature(s) are returned. Filter criteria select features only. This use case is for backup or full replication purposes.

Use Case 2: retrieve time slices of specific features and fulfilling certain constraints

Query parameters:

- a list of feature types and ids (preferably UUIDs)
- filter criteria (FES 2.0) to be applied on the time slices (not features). This can include a constraint on the interpretation property, through which BASELINEs, PERMDELTAs and TEMPDELTAs may be selected (SNAPSHOTs should be handled slightly differently, see below)

Return value:

A list of features containing the time slices that match the filter constraints. Only the time slices with the highest correction number of a sequence are returned (lesser corrections are irrelevant). Cancelled sequences are not returned.

Use Case 3: Retrieve SNAPSHOT time slices

Query parameters:

- a list of feature types and ids (preferably UUIDs)
- a time instant

- filter criteria (FES 2.0) to be applied on the time slices. The constraints have to be fulfilled at the requested point of time.
- a boolean value if properties with schedules should be evaluated or not

Return value:

A list of features containing the SNAPSHOT time slices that match the filter constraints. Properties with schedules are evaluated on request by replacing the schedule with the current value at the given time. The filter must match the resulting value.

Use Case 4: Retrieve the time slices relevant for a point in time

Query parameters:

- a list of feature types and ids (preferably UUIDs)
- a time instant
- filter criteria (FES 2.0) to be applied on the time slices. The constraints have to be fulfilled at the requested point of time.

Return value:

A list of features containing all time slices that match the filter constraints and are relevant for the given point in time. This is at least one BASELINE and may additionally include PERMDELTAs and TEMPDELTAs. Only the time slices with the highest correction number of a sequence are returned (lesser corrections are irrelevant). Cancelled sequences are not returned.

7.2.2.2 Examples

The following XML fragments show a possible implementation of temporal queries in XML. An XML Schema definition for temporal queries does not yet exist.

Example for operation mode 2), time slice filtering: "*Retrieve all PERMDELTA time slices of Runway X whose validTimes intersect with time period Y"*

```
<wfs-ext:TemporalQuery mode="DEFAULT" typeNames="aixm:Runway">
    <!-- filter on feature level -->
  <fes:filter>
    <fes:PropertyIsEqual>
        <fes:ValueReference>gml:identifier</fes:ValueReference>
        <fes:Literal>[X = some UUId]</fes:Literal>
    </fes:PropertyIsEqual>
  </fes:filter>
  <!-- filter on time slice level, implicit filtering of cancelled and corrected
       time slices -->
  <fes:filter>
    <fes:And>
      <fes:PropertyIsEqualTo>
        <fes:ValueReference>interpretation</fes:ValueReference>
        <fes:Literal>PERMDELTA</fes:Literal>
      </fes:PropertyIsEqualTo>
      <fes:TIntersects>
        <fes:ValueReference>validTime</fes:ValueReference>
         <gml:TimePeriod>
          <gml:beginPosition>[... Y - begin ...]</gml:beginPosition>
          <gml:endPosition>[... Y - end ...]</gml:endPosition>
```



```
        </gml:TimePeriod>
      </fes:TIntersects>
    </fes:And>
  </fes:Filter>
</wfs-ext:TemporalQuery>
```

Example for operation mode 3), SNAPSHOT retrieval: *"For all airspaces that are active now, return SNAPSHOT time slices."*

```
<wfs-ext:TemporalQuery mode="EXTRACT"[4] encoding="SNAPSHOT"
typeNames="aixm:Airspace">

  <!-- filter on feature level -->
  <!-- here: unused -->

  <!-- time instant of SNAPSHOT -->
  <gml:TimeInstant>
    <gml:timePosition>[the date of today]</gml:timePosition>
  </gml:TimeInstant>
    <!-- Filter on time slice level, evaluated at the given point in time -->
  <fes:filter>
    <fes:PropertyIsEqualTo>
      <fes:ValueReference>activation/AirspaceActivation/status</fes:ValueReference>
      <fes:Literal>ACTIVE</fes:Literal>
    </fes:PropertyIsEqualTo>
  </fes:Filter>
</wfs-ext:TemporalQuery>
```

Example for operation mode 4), relevant time slices retrieval: *"For all airspaces that are active now, return the relevant time slices."*

The XML for this example is identical to the previous one for operation mode 3) except for the root element:

```
<wfs-ext:TemporalQuery mode="EXTRACT" encoding="RELEVANT_SET"
typeNames="aixm:Airspace">
  <!-- see example for operation mode 3) -->
</wfs-ext:TemporalQuery>
```

7.2.2.3 Discussion

The operation modes 2), 3) and 4) require a replacement of the wfs:Query element in the request to accommodate the extra parameters. This is different to the other proposals, where the wfs:Query element was untouched. This solution would be still a general one in terms of being not linked to aeronautical features. It is suitable for all features based on the temporality model.

Advantages

- a simpler, clearer interface oriented at use cases
- easier to implement, as the complexity is reduced

Disadvantages

[4] The concept of "Extracts" of time slices and their encoding is introduced in [Architecture ER - "Extract - Extending the Snapshot Concept"]

- the query language is less powerful
- the extension is less generic from a WFS perspective

7.3 Improvements for FES 2.0

7.3.1 Introduction of a new temporal filter or function: 'evaluateDuring'

The AIXM 5 temporality model requires very specific handling to support all of it features. This makes it difficult to use a generic temporal filter that answers simple queries like: 'return all airspaces that are currently active'. One of the reasons for this is that time slices can overlap and override each other. Another reason is that an extensive timesheet system is used to define recurring properties (e.g. an airspace be active between sunrise and sunset on work days).

One solution for this would be the declaration of a custom filter function. This was already done in the SAA pilot, where an SLD was defined for an airspace based on its time to activation. The complete filter for the 'evaluateDuring' function,given below, is called with two arguments. The first argument simply specifies a filter that needs to be applied on AIXM time slices, the second argument specifies the relevant time property. The function will return true if the first filter accepts a snapshot that can be generated during the time property specified in the second argument. The second argument can specify either a period, or an instant.

```
 <ogc:Filter>
        <ogc:PropertyIsEqualTo matchCase="true">
          <ogc:Function name="evaluateDuring">
            <ogc:Literal>
              <ogc:Filter>
                <ogc:PropertyIsEqualTo matchCase="true">
                  <ogc:PropertyName
xmlns:ns1="http://www.aixm.aero/schema/5.1">ns1:activation/ns1:AirspaceActi
vation/ns1:status</ogc:PropertyName>
                  <ogc:Literal>ACTIVE</ogc:Literal>
                </ogc:PropertyIsEqualTo>
              </ogc:Filter>
            </ogc:Literal>
            <ogc:Literal>
              <ns0:TimePeriod>
                <ns0:beginPosition>2011-03-
30T15:59:51.003+02:00</ns0:beginPosition>
                <ns0:endPosition>2011-03-
30T19:58:51.003+02:00</ns0:endPosition>
              </ns0:TimePeriod>
            </ogc:Literal>
          </ogc:Function>
          <ogc:Literal>true</ogc:Literal>
        </ogc:PropertyIsEqualTo>
    </ogc:Filter>
```

Advantages

- No extension schemas needed, the function only needs to be completely described to ensure that all implementations are the same.
- The result of this query will contain the original time slices
- The function is both very simple and generic

Disadvantages

- Custom functions are not covered by the WFS spec, so needs to be defined elsewhere, generic clients will not be able to use it.

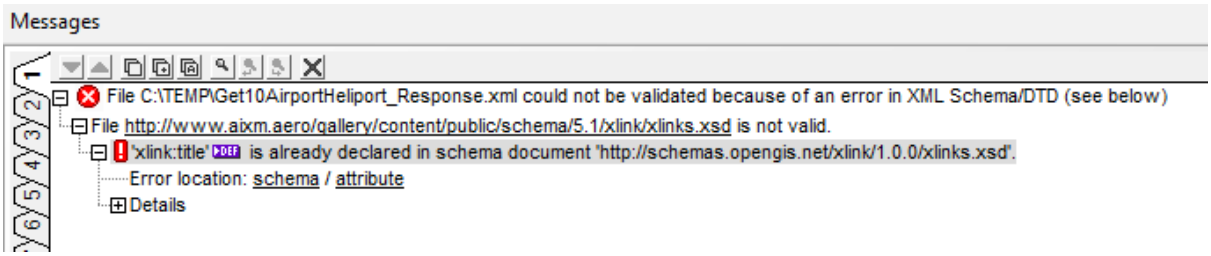
7.4 Improvements to AIXM 5.1

7.4.1 Change Request for imports of Foundation Schema

The AIXM 5.1 application schema includes import references to local online copies of the foundation schema (GML 3.2.1, ISO 19115, XLink 1.1):

- http://www.aixm.aero/gallery/content/public/schema/5.1/ISO_19136_Schemas/
- http://www.aixm.aero/gallery/content/public/schema/5.1/ISO_19139_Schemas/
- http://www.aixm.aero/gallery/content/public/schema/5.1/xlink/

The inclusion of references to local copies of the foundation schema, particularly GML 3.2.1 resulted in issues when tools attempted to perform schema validation of the GetFeature wfs:FeatureCollection response.



The wfs:FeatureCollection schemaLocation string refers to both the AIXM 5.1 and WFS 2.- schemas to enable validation. But the conflicting location of the same schemas causes issues for some XML tools (such as XMLSpy) to validate the wfs:FeatureCollection response.

This issue can resolved by editing the AIXM 5.1 schemas to reference the definitive online version of the foundation schemas rather than the online copy local to the AIXM 5.1 schema.

Proposed AIXM 5.1 Change Request: Amend import schemaLocation for Foundation Schema

Change the import schemaLocations for GML 3.2.1, XLink and ISO Metadata schemas from to import the definitive version of the schemas accessible via the OGC schema repository.

```
<!-- Component: AIXM: Basic Message -->
<schema xmlns:message="http://www.aixm.aero/schema/5.1/message" xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:gml="http://www.opengis.net/gml/3.2" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:aixm="http://www.aixm.aero/schema/5.1"  targetNamespace="http://www.aixm.aero/schema/5.1/message"
elementFormDefault="qualified" attributeFormDefault="unqualified" version="5.1">
<import namespace="http://www.opengis.net/gml/3.2" schemaLocation="http://schemas.opengis.net/gml/3.2.1/gml.xsd"/>
<import namespace="http://www.w3.org/1999/xlink" schemaLocation="http://schemas.opengis.net/xlink/1.0.0/xlinks.xsd"/>
<import namespace="http://www.aixm.aero/schema/5.1" schemaLocation="/AIXM_Features.xsd"/>
```

7.5 Improvements to GML

The Dynamic Feature Model within GML should be revised to bring it into line with the AIXM 5.1 Temporality Model which provides a more comprehensive mechanism for handling the temporality of real-world objects. The AIXM 5.1 Temporality Model has application within domains outside aviation as all real-world objects exist in both space and time. The Dynamic Feature Model should be replaced by a Temporality Model which should be developed with the aim for it to become part of the ISO TC 211 standard series.

Making the AIXM 5.1 Temporality Model the base of a generic temporality model will help with the development of generic solutions to better handle the retrieval of AIXM data within OGC web services.

8 Conclusion

The initial aim of this guidance report was to provide a normative document for implementing an AIXM 5.1 WFS 2.0. However, during the development of the report and flight dispatch and planning scenarios within OWS-8, several issues remain outstanding relating to temporality and SNAPSHOT handling that require further development and testing. Therefore this guidance report provides an overview of the operations supported by the WFS 2.0 specification and recommendations for a minimum set of operations and behaviours that an implementation of the WFS 2.0 should support. The recommendations that have been provided should be taken forward by the Aviation DWG to form the basis of a normative document that defines a minimum set of operations and behaviours that a WFS 2.0 serving AIXM 5.1 should support.

But before a normative report can be developed further work should be undertaken to discuss and test proposed solutions to improve support for temporality and SNAPSHOT handling, namely:

1. Handling SNAPSHOT timeslices within the WFS
2. Investigate the ability to subset feature timeslices properties in a response
3. Further develop and test the proposed temporal filter/function:EvaluateDuring
4. Align the GML Dynamic Feature Model and AIXM 5 Temporality Model

To facilitate this, the following next steps are proposed:

- Hold a workshop at the next OGC TC (Brussels, 2012) with all interested parties will to discuss:
  - Implementing the AIXM 5 Temporal model for features within a WFS (subsetting timeslices, retrieving SNAPSHOTs (time instant and time period))
  - Integrating the AIXM 5 Temporality model and OGC Dynamic Feature Model
- Recommendations for OWS 9 Aviation thread
  - Develop normative conformance document for developers, in line with the WFS 2.0 compliance test cases
  - Develop a WFS 2.0 compliance test using the OGC Team Engine. However, note that there is currently no WFS 2.0 compliance test to use as a starting block

Annex A: Aviation client use cases and WFS requirements

This section contains some relevant use cases for the use of AIXM 5 data. A WFS based client that wants to support these use cases subsequently has some requirements on the WFS. These are also listed. This should give WFS implementors an idea of what to support and why it should be supported.

Note that it has already been demonstrated that the number of time slices in a single feature can be very large, which is the underlying reason why most use cases require some sort of filtering on the number of returned time slices.

A.1. Thick client

A thick client typically runs on a regular desktop system, but has specific capabilities to handle AIXM 5 data:

- Decoding of AIXM 5
- Visualization of AIXM 5
- Creation of snapshots
- Creation of new AIXM 5 time slices

A.1.1 Flight planning

A flight dispatcher wants to retrieve data for features that are relevant for his flight plan during the time of flight. Updates during the flight are received as events, and are merged with the loaded data.

Client requirements on WFS:

- Spatial filtering using flight plan as input to a buffer query
- Temporal filtering on response to limit the number of returned time slices

An amateur pilot wants to plan a flight in a specific region, but wants to avoid any SAA airspaces that will be activated during his flight. The client uses time dependent styling to visualize when an airspace will become active.

Client requirements on WFS:

- Spatial filtering using the bounding box of his flying region
- Temporal filtering on response to limit the number of returned time slices
- Temporal filtering to retrieve only time slices that are active during his time of flight

A.1.2. Data authoring

An airport wants to schedule work on a runway while ensuring that the other runways will be available during the runway closure. The client allows simulating the result of a timeslice insertion, because it supports snapshot creation.

Client requirements on WFS:

- Filtering of runways for a given airport using wfs:valueOf
- Temporal filtering on response to limit the number of returned time slices

- Update transaction support to include a new time slice in the WFS

A military user wants to create an ad-hoc airspace. He wants to use the location of an airport and possibly some GeoBorders to do this.

Client requirements on WFS:

- Limit the response to include only baseline time slices

A1.2.3. Data auditor

A data auditor is carrying out an investigation to review digital notam processes.

Client requirements on WFS:

- Retrieve full feature history including corrected time slices

A.2. Thin client

A thin client typically has limited resources and capabilities. It can be for instance an internet application running in a browser. It does not necessarily want to visualize the data, but can simply extract values from it to show in a web page.

Requirements on WFS:

- Retrieve snapshot time slices for a time instant.

