Open Geospatial Consortium, Inc.

Date: 2010-08-18

Reference number of this document: OGC 10-131r1

Category: OGC Engineering Report

Editor: Debbie Wilson

OGC® OWS-7 Aviation – AIXM Assessment Report

.

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. It 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.

Preface

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

This public Engineering Report (ER) is a deliverable of the Open Geospatial Consortium (OGC) Interoperability Program Open Web Service (OWS) Testbed phase 7 (OWS-7). The report describes the outcomes of the evaluation and advancement of AIXM that focussed specifically on using and testing new AIXM 5.1 features in an OGC Web Services environment.

This report shall focus on evaluating the ability to:

- Serve, filter and update AIXM 5.1 data via the OGC WFS-T 2.0 interface
- Recommend guidelines or cross-walks for interpreting the new AIXM 5.1 schedules in conjunction with the Timeslice model in a web services environment

These new features were evaluated within a typical real-world aviation scenario relating to dispatch operations: flight planning, pre-flight briefing and en-route assistance.

The document shall summarize the key aspects investigated, highlight issues encountered with both the OGC Web Services and AIXM 5.1 specification and provide recommendations for improvements to enable the aviation domain to adopt OGC web services for access and use of aeronautical information within a distributed service oriented infrastructure.

Suggested additions, changes, and comments on this draft report are welcome and encouraged. Such suggestions may be submitted by email message or by making suggested changes in an edited copy of this document.

The changes made in this document version, relative to the previous version, are tracked by Microsoft Word, and can be viewed if desired. If you choose to submit suggested changes by editing this document, please first accept all the current changes, and then make your suggested changes with change tracking on.

Contents

Page

Figures ..... Page

Tables ..... Page

OGC® OWS-7 Aviation – AIXM Assessment Report

1 Introduction

1.1 Scope

The OWS-7 Aviation Thread shall focus on investigating and demonstrating the applicability of OGC Web Services for accessing and using AIXM 5.1 and WXXM 1.1 data within a typical real-world Flight Dispatch scenario supporting flight planning (including General Aviation) and preparation (MET and AIM); calculating weight and balance; estimating fuel requirements and in-flight emergency response.

The OWS-7 Aviation - AIXM Assessment Engineering Report shall summarize the key outcomes of the thread tasked with using and testing new AIXM 5.1 features in an OGC Web Services environment:

- Serving, filtering and updating AIXM 5.1 data via the OGC WFS-T interface,
- Recommending guidelines or cross-walks for interpreting the new AIXM 5.1 schedules in conjunction with the Timeslice model in a web services environment

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.

1.2 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 Ltd.
Daniel Hardwick	Snowflake Software Ltd.
Ian Painter	Snowflake Software Ltd.
Ulrich Berthold	Comsoft

1.3 Revision history

Date	Release	Editor	Primary clauses modified	Description
08.06.2010	0.1.0	DW	All	Definition of initial document outline for contribution from authors
11.06.2010	0.2.0	DW	All	Initial population of content received from contributing authors
16.06.2010	0.3.0	DW	All	Additional input of content and review
17.06.2010	0.4.0	DW	All	Final content received and review ready for release for formal review
30.06.2010	1.0	DW	All	Final updates following formal review

1.4 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), Aeronautical Information Exchange Model (AIXM) Temporality Model – Draft v0.6

EUROCONTROL (2010), xSNOWAM-TEC: AIXM Complex Topics. Version 0.1

ISO 19109:2005, *Geographic Information – Rules for Application Schema*

ISO 19115(all parts), *Geographic information - Metadata*

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

ISO/DIS 19142, *Geographic information — Web feature service*

ISO/DIS 19143, *Geographic information – Filter Encoding*

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

NOTE This OWS Common Specification contains a list of normative references that are also applicable to this Implementation Specification.

OGC 09-050r1, *OpenGIS® OWS-6-AIM Engineering Report*

OGC 10-079, *OWS-7 Aviation Architecture Engineering Report*

3 Terms and definitions

For the purposes of this report, the terms and definitions are defined in the OWS-6 Event Architecture Engineering Report (09-032) and the OWS-6 AIM Engineering Report (09-050r1).

4 Conventions

4.1 Abbreviated terms

AIM	Aeronautical information Management
AIXM	Aeronautical Information Exchange Model
ATS	Air Transport System
EFB	Electronic Flight Bag
ER	Engineering Report
FE	Filter Encoding
GML	Geography Markup Language
HTTP	HyperText Transport Protocol
ISO	International Standardization Organization
NOTAM	NOTice To AirMen
OGC	Open Geospatial consortium
OWS	OGC Web Services
OWS-7	OWS testbed phase 7
SNOWTAM	Snow Notice to Airmen
SWIM	System Wide Information Management
WFS	Web Feature Service
WFS-T	Web Feature Service -Transactional
WXXM	Weather Information Exchange Model
XML	Extensible Markup Language

5 AIXM Assessment Overview

5.1 Introduction

The Aviation Thread of OWS-7 builds on the Aeronautical Information Management (AIM) thread of OWS-6 and seeks to further develop and demonstrate the use of the Aeronautical Information Exchange Model (AIXM) and the Weather Information Exchange Model (WXXM) in an OGC Web Services environment.

The key objective is to demonstrate the ability of the Web Feature Service (WFS 2.0) and Filter Encoding specifications (FES 2.0) to enable aeronautical information encoded in AIXM 5.1 to be accessed by the wide variety of end-users through both pull (request/response) services and push (publish/subscribe) event services[1] to enable a common operating picture (Figure 1).

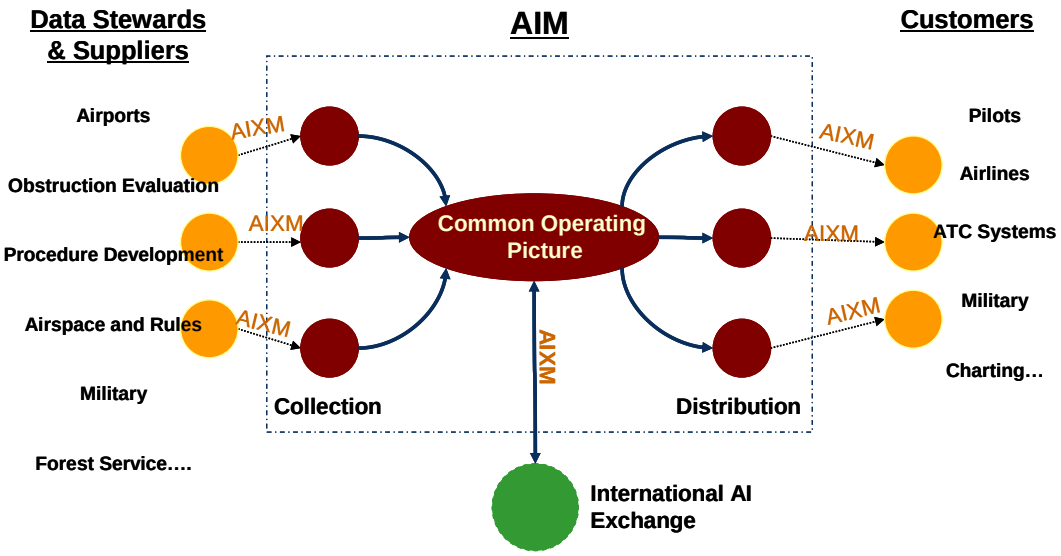


Figure 1. AIXM in Support of New AIM Paradigm

5.2 AIXM 5.1 overview

The US Federal Aviation Administration (FAA) and EUROCONTROL have developed AIXM as a global standard for the representation and exchange of aeronautical information. AIXM has been designed as a basis for enabling the transition to a net-centric, global interoperable Air Transport System (ATS).

[1] Note: Detailed discussion of the aviation event architecture is out of scope of this report as this is covered in 10-079 OWS-7 Aviation Architecture Engineering Report.

AIXM uses the OGC Geography Markup Language (GML 3.2) tailored to the specific requirements for the representation of aeronautical objects, including the temporality feature that allows for time dependent changes affecting AIXM features. FAA and EUROCONTROL are using AIXM as an integral part of their efforts to modernize their aeronautical information procedures and to transition to a net-centric, global aeronautical management capability.

AIXM 5.1 was officially released on February, 2nd 2010 and is currently available for download at www.aixm.aero.

5.3 Changes to AIXM 5.1 from AIXM 5.0

AIXM 5.1 contains many enhancements to the prior version of AIXM (5.0):

- Concept review of schedules and temporality: a new "PropertiesWithSchedule" concept is introduced in order to replace the Timetable/Timesheet classes inherited from earlier AIXM versions. This change puts in agreement the "schedules" modelling with the new Temporality Concept of AIXM 5. It also indicates clearly which feature attributes are actually concerned with regular schedules.
- Harmonisation of usage classes for Airport/Heliport, Airspace, Route, Procedure and AerialRefuelling features
- Update of the SurfaceContamination model to improve the modelling of the current SNOWTAM information
- Change of all AIXM enumerated lists of values into open codelists, to comply with ISO 19136
- Optimisation of the AIXM XSD Schemas: nilReason is now defined at data type level

The complete list of changes is available on the AIXM wiki: www.aixm.aero/wiki.

6 Serving AIXM 5.1 data via OGC Web Feature Service (WFS-T 2.0) Interface

Two service providers were commissioned to deploy OGC web services serving AIXM 5.1 to the Dispatch client and Electronic Flight Bag (EFB): i) Snowflake Software deployed its GO Publisher WFS and ii) Comsoft deployed its CADAS-AIM$_{DB}$

Both systems developed two data access mechanisms to support the various decision-support applications defined within the two scenarios:

1) ***Pull – request/response:*** clients define a filter request to return AIXM 5.1 data matching their request criteria via the WFS 2.0 interface

2) ***Push – publish/subscribe:*** on insert of a timeslice via WFS-T interface the service generates and pushes digital NOTAMs to an event service which then pushes the NOTAM to subscribed clients based on their subscription criteria

The following subsections shall summarize the server side implementation architecture established by Snowflake and Comsoft to serve AIXM 5.1 via the OGC WFS-T 2.0 interface and the data management processes required to establish the server side architecture. Ongoing maintenance of the AIXM 5.1 data is described in Section 10.

6.1 Snowflake GO Publisher WFS

Snowflake Software's GO Publisher commercial off-the shelf (COTS) product is comprised of flexible, scalable components that support the translation and data exchange requirements of aeronautical and weather information systems (Figure 2).

Aeronautical and weather data are published using GO Publisher via three components:

1. **WFS 2.0:** this provides read-only access to AIXM 5.1 and WXXM 1.1 data by the aviation decision support clients: Dispatch Client and Electronic Flight Bag (EFB)
2. **WFS-T 2.0:** this allows authorized aviation clients (create/edit) to update the datastore to post inserts of timeslices to the Snowflake AIXM 5.1 datastore
3. **Event Publisher:** creates events such as digital NOTAMs or WXXM events from the source database which are pushed to an event service
4. **Event Pusher:** is a java application that subscribes to the Event Subscription service via SOAP. It runs as a background process polling an FTP directory where new NOTAMs are published by the Event Publisher and pushed to the Event service within a SOAP container.

Based on experience gained from OWS-6 and the Eurocontrol Digital Snowtam trials, our aim was to provide a comprehensive implementation of the WFS 2.0 specification and to test the new operations and functions to improve retrieval of AIXM features and properties (Table 1).

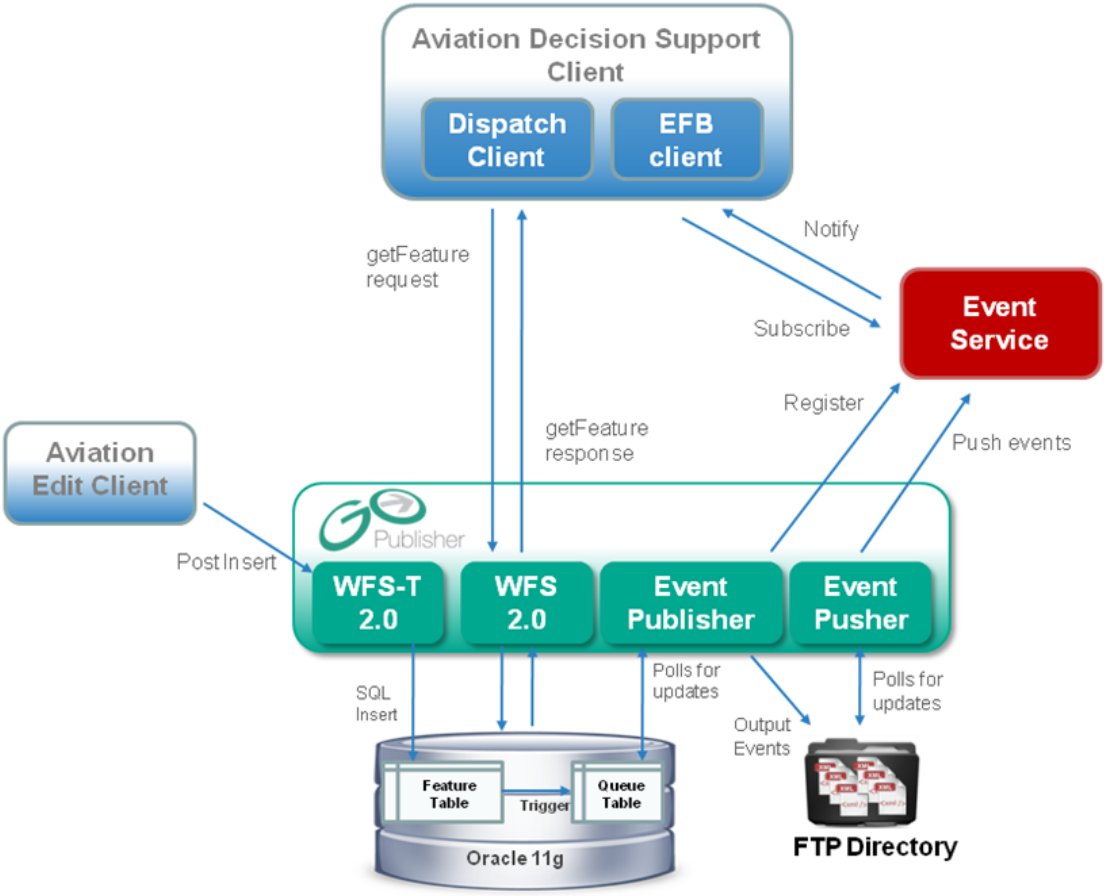


Figure 2. Overview of the Snowflake aviation component architecture

Table 1. GO Publisher WFS 2.0 Conformance

		WFS 2.0 Conformance Levels				
Feature		**Simple**	**Basic**	**Transactional**	**Locking**	**GO Publisher WFS 2.0**
Service Metadata	Get Capabilities	✓	✓	✓	✓	✓
	WSDL	·	·	·	·	✓
	Describe Feature Type	✓	✓	✓	✓	✓
Retrieval	Get Feature	✓	✓	✓	✓	✓
	Get Property Value	·	✓	✓	✓	✓
Stored Queries	Describe Stored Queries	✓	✓	✓	✓	✓
	List Stored Queries	✓	✓	✓	✓	✓
Manage Stored Queries	Create Stored Queries	·	·	·	·	✓
	Drop Stored Queries	·	·	·	·	·
Transactions	Transaction	·	·	✓	✓	✓
	Lock Feature	·	·	·	✓	·
	Get Feature With Lock	·	·	·	✓	·
Filter Encoding (ISO 19143)	Minimal FE Spatial Filter (BBOX)	·	✓	✓	✓	✓
	Spatial Filter	·	·	·	·	✓
	Temporal Filter	·	·	·	·	✓
	Standard Filter	·	·	·	·	✓
	Version Navigation	·	·	·	·	·
	Sorting	·	·	·	·	·
	Stored Query	·	·	·	·	✓
	Ad hoc Query	·	·	·	·	✓
Binding	HTTP GET	·	·	·	·	✓
	HTTP POST	·	·	·	·	✓
	SOAP	·	·	·	·	✓

6.2 Comsoft CADAS-AIM$_{DB}$

COMSOFT's datastore is a unique AIXM 5.1 based database (CADAS-AIMDB), which is able to handle static and dynamic data, so that users (such as pilots, flight dispatcher) will be supported by up to date aeronautical information. The data fusion of static and dynamic data simplify the operational procedures significantly. CADAS-AIMDB is fully compliant to the following standards and recommendations:

- ISO 19107 Spatial
- ISO 19108 Temporal
- ISO 19115 Metadata
- Universal Markup Language (UML)
- Extensible Markup Language (XML)
- ISO 19136 Geographic Markup Language (GML)

Air Traffic Controllers and pilots can directly benefit from the increased data quality and the CADAS AIMDB based system will considerably contribute to safety improvement at reduced operational expenses.

CADAS-AIMDB is a fully featured AIXM 5 database that implements the temporality model as defined in version 5.1 of AIXM. It is designed to serve as a base for integrating all kinds of AIM products and components such as electronic AIP, Charting, NOTAM Office or Briefing. The SOA principles apply for all parts of the database architecture thus providing an open interface to clients.

COMSOFT's contribution to the OWS-7 is to implement a WFS-T 2.0 interface in order to access all functions of the database through a standardized interface. The following standards are implemented:

- ISO 19142 (Web Feature Service 2.0)
- ISO 19143 (Filter-Encoding)

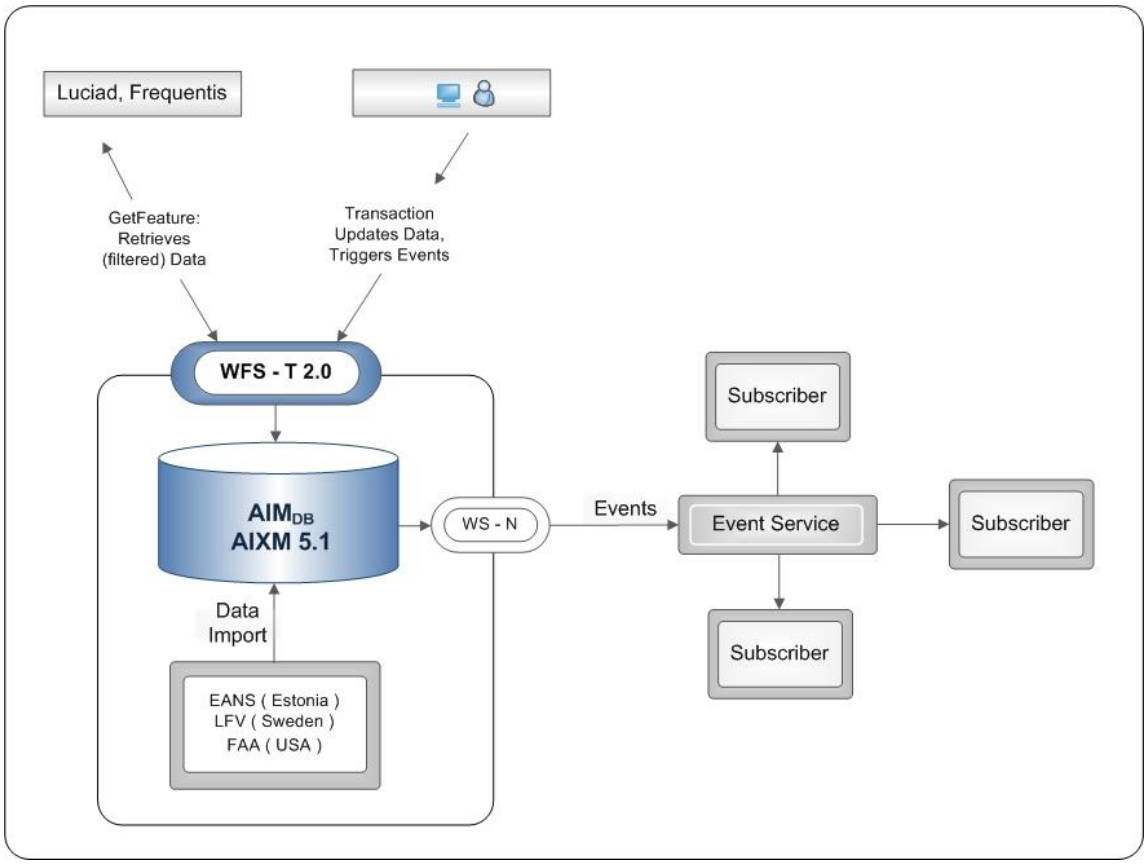


Figure 3. Comsoft AIXM 5.1 Datastore

6.3 Establishing OWS-7 AIXM 5.1 Datastore

Prior to setting up server side environments, data was sourced from numerous providers and supplied to Snowflake and Comsoft. However, few of the data provided were encoded as AIXM 5.1 consequently; data transformation processes were required to translate data from legacy data specifications (e.g. AMDB, DAIM, AIXM 5.0) into AIXM 5.1. Once data was successfully transformed into AIXM 5.1 both Comsoft and Snowflake shared their datasets to ensure that clients could access all AIXM data seamlessly.

6.3.1 Data Transformation and Load by Snowflake Software

Snowflake Software was supplied data from a wide range of providers and services and re-used AIXM 5.0 data from OWS-6. The supply of data in numerous legacy specifications posed a significant challenge to the project, however, Snowflake's generic schema translation products: GO Loader and GO Publisher were able to transform any dataset into AIXM 5.1 and WXXM 1.1 within days of receipt[2]. This enabled Snowflake

[2] The generic schema translation technologies within GO Loader and GO Publisher remove the need for bespoke software coding enabling systems to consume and publish data to new data specifications within days at minimal cost

to serve the data via the WFS to support the development of the dispatch clients and Electronic Flight Bags (EFB) early in the test bed to maximize client productivity.

To create an OWS-7 AIXM 5.1 data store, Snowflake transformed these disparate aeronautical datasets into AIXM 5.1 prior to loading into the Snowflake OWS-7 data store (Figure 4). Datasets were initially loaded into an Oracle 11g staging database using GO Loader. GO Loader automatically generated database schemas aligned to the source application schema and loaded the data into these staging tables. GO Publisher Desktop was then connected to these database tables and used to configure the schema translations required to transform the data into AIXM 5.1. Once the schema translations were configured GO Publisher Desktop published AIXM 5.1 files from the staging database.

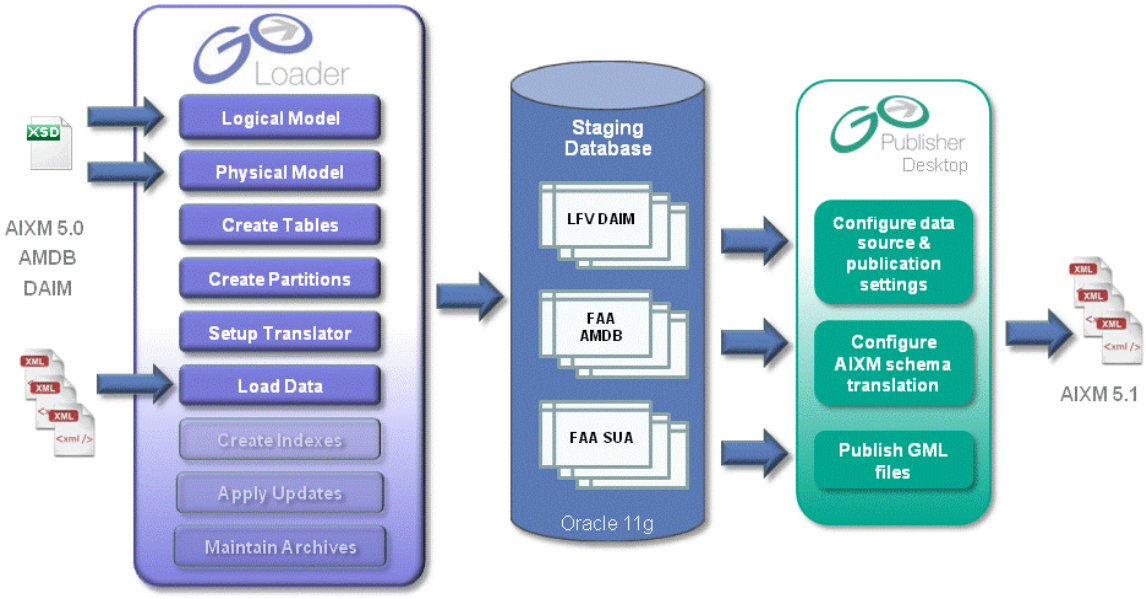


Figure 4 - Translating aviation data into AIXM 5.1: initial translation of non AIXM 5.1 data to publish into OWS-7 AIXM 5.1 database

Provider	Data Source	Schema	FeatureTypes extracted	Comments
FAA	AMDB WFS	AMDB GML 3.1.1.	• amdb:amdb_runway • amdb:amdbtaxiway • nasr:nasr_arp • nasr:nasrnavigation_aid • amdb:amdbnavaid • amdb:amdbcontrol_tower • amdb:amdbobstacle • geography:states • geography:rivers	AMDB sample data while similar did not exactly match the AIXM5.1 structure. For example a single amdb_runway feature type was used to create both the AIXM Runway and RunwayElement features types.
	Special Use Airspace (SUA)	AIMS 5.0	• Airspace • GeoBorder • AirTrafficControlService	FAA SUA AIXM5.0 data was manually modified to replace ArcbyCentrePoint, CirclebyCentrePoint and href geometries with geometries types supported by Oracle Spatial to enable the data to be loaded. Although we were provided with several SUA datasets we only loaded the datasets applicable to scenario 2 due to the overhead involved in manually transforming the data.
EUROCONTROL	Snowtam Trial (TempDelta)	AIXM 5.1	• aixm:AirportHeliport • aixm:Runway • aixm:Taxiway	
	ESMS Scheduled Availability (Baseline)	AIXM 5.1	• aixm:AirportHeliport	

Provider	Data Source	Schema	FeatureTypes extracted	Comments
LFV	DAIM Baseline	DAIM GML 2.1	• DAIM_BASELINE.AM_AERODROMEREFERENCEPOINT • DAIM_BASELINE.AM_RUNWAYELEMENT • DAIM_BASELINE.AM_TAXIWAYELEMENT	DAIM sample data while similar did not exactly match the AIXM5.1 structure. For example a single Runway feature type was used to create both the Runway and RunwayElement features types.
OWS-6	OWS-6 AIXM Oracle Database (Baseline)	AIXM 5.0	• aixm:Route • aixm:RouteSegment	OWS6 data was added to provide an example of how the proposed MTT information would be encoded.
EANS	Estonia	AIXM 5.1	All AIXM 5.1 features contained in data	Estonia data was provided from Comsoft for inclusion within our AIXM 5.1 data to ensure that the services had a consistent dataset to meet the requirements of the scenarios

In a second step, GO Loader generated an AIXM 5.1 database schema and loaded both the AIXM 5.1 generated from the translation process and data supplied in AIXM 5.1 from EUROCONTROL and Comsoft into the OWS-7 AIXM 5.1 data store (Figure 5). GO Publisher Desktop was again used to configure the translation required to publish AIXM 5.1 and digital NOTAMs and these were deployed within the WFS 2.0 and Event Publisher, respectively.

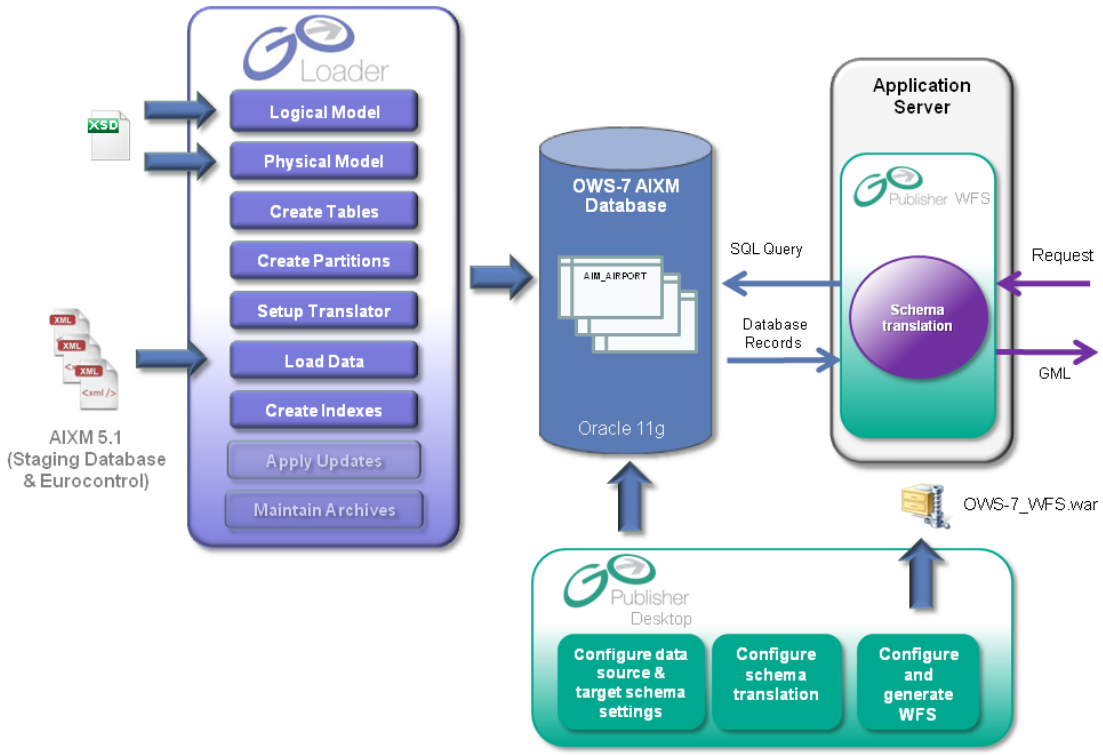


Figure 5 - Translating aviation data into AIXM 5.1: initial load of AIXM 5.1 data into OWS-7 AIXM databaseIssues with data loading

6.3.1.1 Data Load Issues

Issues were encountered loading datasets which contained complex geometry types: ArcByCentrePoint, CircleByCenterPoint and geometries encoded ByReference. For example, the Special Use Airspace data received from FAA contained features containing ArcbyCentrePoint, and CirclebyCentrePoint geometries. GO Loader was unable to load these geometries as these geometries are not included in the Simple Features for SQL specification and therefore not supported by Oracle Spatial or other mainstream databases. Consequently, manual transformation steps had to be performed as the data to convert the data into supported geometry types which were then loaded into the database.

6.3.2 Data Loaded by Comsoft

For the OWS-7 testbed COMSOFT used the AIXM 5.1 AIP Data from Estonian Air Navigation Services (EANS), this was subsequently supplied to Snowflake for inclusion

in their AIXM 5.1 datastore. Comsoft also loaded the AIXM 5.1 data provided by Snowflake, which after some conversion effort was also imported into CADAS-AIMDB.

6.4 AIXM 5.1 Temporality Model

The AIXM 5.1 Temporality Model is one of most the significant improvements that have been made to the AIXM specification as changes to the state of aeronautical features are of time, critical importance to aviation operational safety.

The AIXM 5.1 conceptual model states that a real-world aeronautical object should be represented by a single feature containing 1 or more timeslice objects representing the state and/or events that have occurred or will occur during its lifespan (Table 2). However, this conceptual representation poses a challenge for implementing systems using OGC WFS 2.0.

Table 2. AIXM 5.1 Timeslice classifications

AIXM Feature Type	TIMESLICE	Definition
State	BASELINE	Timeslice that describes the state of a feature as a result of a permanent change. The BASELINE timeslice contains all properties that are in effect during the featureLifetime
	SNAPSHOT	Timeslice that describes the "current" state of a feature at a particular time. This SNAPSHOT timeslice contains all properties that are in effect where any properties that are changed by temporary events replace the permanent property value contained within the BASELINE.
Event	PERMDELTA	Timeslice that contains only the new values for properties that have changed due to an event that permanent change. These values will then be contained in a new BASELINE timeslice object
	TEMPDELTA	Timeslice that contains only the values for properties that have changed due to an event that results in a temporary change. TEMPDELTA will not result in changes to a BASELINE timeslice object, but should supersede baseline properties in a SNAPSHOT timeslice

The OGC WFS 2.0 assumes that the features contained within the service conform to the ISO 19109 General Feature Model and is designed to retrieve complete features or values for a single specified property. There is no operation within the WFS to return a feature containing a subset of objects. Therefore, if an aeronautical object is represented as a single feature containing multiple timeslice objects then the service will return all the timeslices.

This is a significant issue as the AIXM 5.1 specification has been designed to reduce the amount of information that needs to be exchanged online due to bandwidth constraints. It also poses a burden on client applications as they would be required filter the data after receipt.

To enable data to be served via OGC WFS, the implementation of the AIXM Temporality Model requires that each real world aeronautical feature is represented by 1 or more AIXM Features each sharing the same gml:identifier. There are several options for representing an AIXM feature depending on how the data is being used:

- **History:** represents an AIXM Feature contains 1 or more event timeslices: PERMDELTA and TEMPDELTAs. The history AIXM feature may contain a set of timeslices describing all of the events that occurred during the features lifecycle or only the event timeslices for a specified time period.
  ***NOTE*:** a history feature is commonly used for data exchange to update data in distributed systems rather than request/response scenarios
- **State:** represents an AIXM Feature that contains a single timeslice: BASELINE or SNAPSHOT that represents the state of the feature during particular time period
- **Event:** represents an AIXM Feature that contains a single timeslice: PERMDELTA, TEMPDELTA that represents an event that changes the properties of a feature for a specified time

Representing each real-world aeronautical feature as one or more AIXM features enables the user to request only the state or events that apply during a specified time period or instant via the WFS.

6.4.1 Implementing the AIXM Temporality Model using GO Publisher WFS

GO Publisher WFS can be configured to represent AIXM features as either 'history', 'state' and 'event' feature types. However, the data retrieval requirements of the dispatch system did not need to support 'history' feature types as both the dispatch client and EFB require fine-grained retrieval of individual timeslices. Therefore Snowflake chose to represent aeronautical features as separate AIXM features, logically linked via the gml:identifier containing a single BASELINE or TEMPDELTA/PERMDELTA timeslice.

Snowflake did not initially implement representing the current state of an AIXM feature using the SNAPSHOT timeslice, as it was anticipated that client applications would generate these features (NOTE: Luciad provides this functionality). However, during the course of the testbed, Frequentis requested that SNAPSHOT event features should be provided as their client would not be capable of generating SNAPSHOT timeslices.

Snowflake also identified that service providers should provide AIXM features containing a SNAPSHOT timeslice that represents the current state as there are instances when a client can post a query to request data against AIXM features containing

BASELINE and TEMPDELTA timeslices which will return in inaccurate view of the current state. These issues will be discussed in section xxx.

Snowflake investigated the ability to generate AIXM features containing SNAPSHOT timeslices that represent the "current state" of an *AIXMFeature* for particular time periods (rather than time instances) across the whole feature lifecycle. However, this could not be achieved within OWS-7 timescales due to the complex rules for generating SNAPSHOT timeslices. Therefore, more work is required to better understand how to efficiently manage AIXM 5.1 data within the datastore to convert event features containing TEMPDELTA/PERMDELTA timeslices via the WFS-T interface into event features containingBASELINE/SNAPSHOT timeslices for access via the WFS 2.0 interface.

6.4.2 Implementing the AIXM Temporality Model using CADAS-AIMDB

The operations in the WFS 2.0 ISO draft specification are based on feature retrieval, modification and filtering. In AIXM 5, however, the main entities dealt with are time slices, not features. This is due to the time varying nature of all properties. This results in a conceptual conflict when using a WFS 2.0 for handling AIXM 5 data.

A physical real world object is represented by exactly one AIXM 5 feature that has a UUID (gml:identifier). The feature contains all associated timeslices and according to the AIXM 5 temporality model timeslices are never deleted but only added. Note that a timeslice in general can be uniquely identified by the following meta data:

- Feature Type
- UUID (gml:identifier)
- Interpretation (baseline, tempdelta, permdelta, snapshot)
- Sequence number
- Correction number

Therefore, whenever new timeslices are inserted, the feature contains the previously existing timeslices as well as the inserted timeslices (Figure 6).
However, using the WFS 2.0 specification it is not possible to directly support the concept required by AIXM 5 as discussed above. When using an implementation strictly compliant with the WFS 2-0 specification, there are two options, but none of them satisfies the typical user's needs.

1. The features offered by the WFS are interpreted as the features in AIXM.
2. The features offered by the WFS are interpreted as the time slices in AIXM.

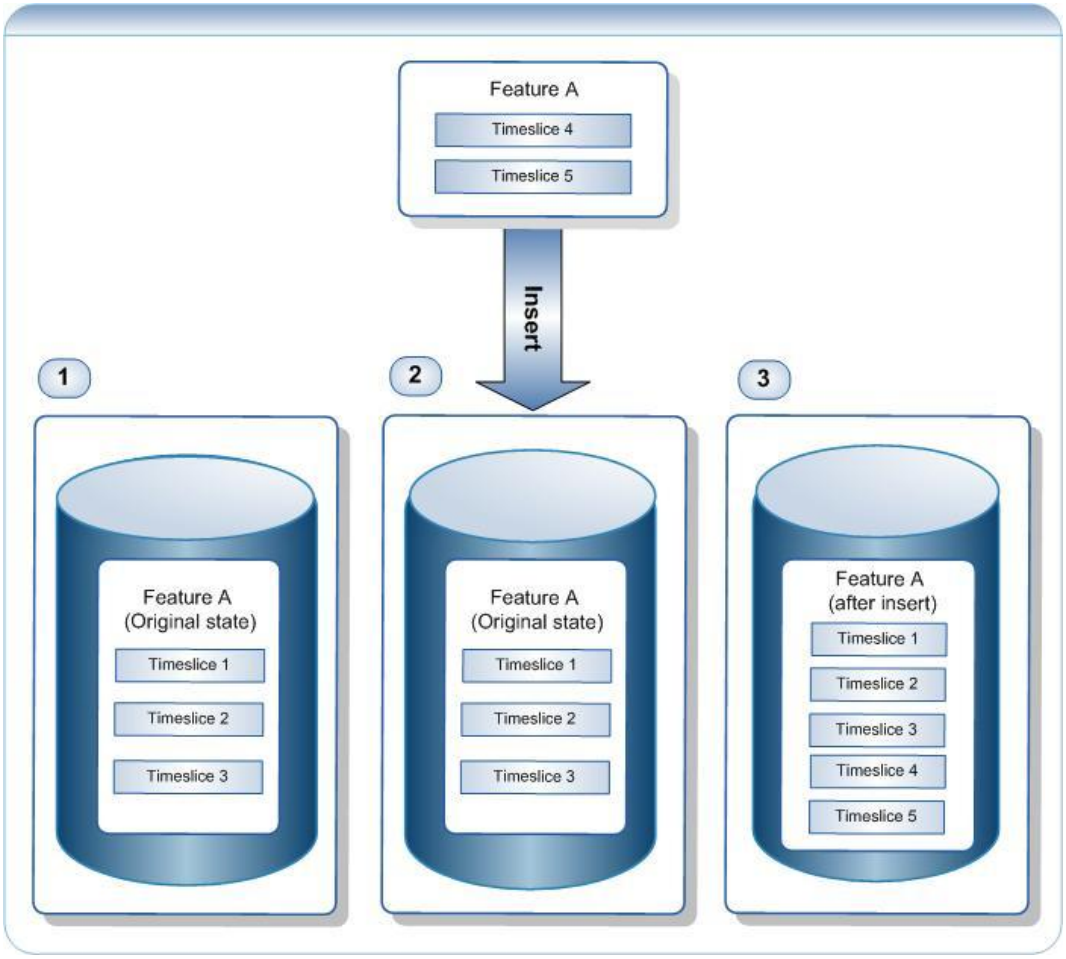


Figure 6. Features and timeslices in AIXM 5

The features offered by the WFS are interpreted as the features in AIXM.

This approach has two drawbacks. First, the insertion of new timeslices replaces the old timeslices, as only complete features can be updated. To maintain the complete set of timeslices the client has to take care of the existing timeslices and merge them before inserting them into the database (c.f. Figure 7).

Furthermore, because filtering in WFS only constrains the set of features, not their content, this approach doesn't allow the filtering of time slices. In a GetFeature request, the client can only limit the number of features retrieved. To be worse, every feature returned must contain all of its time slices, because from a XML perspective, an AIXM feature is a document and the complete list of time slices is part of its content.

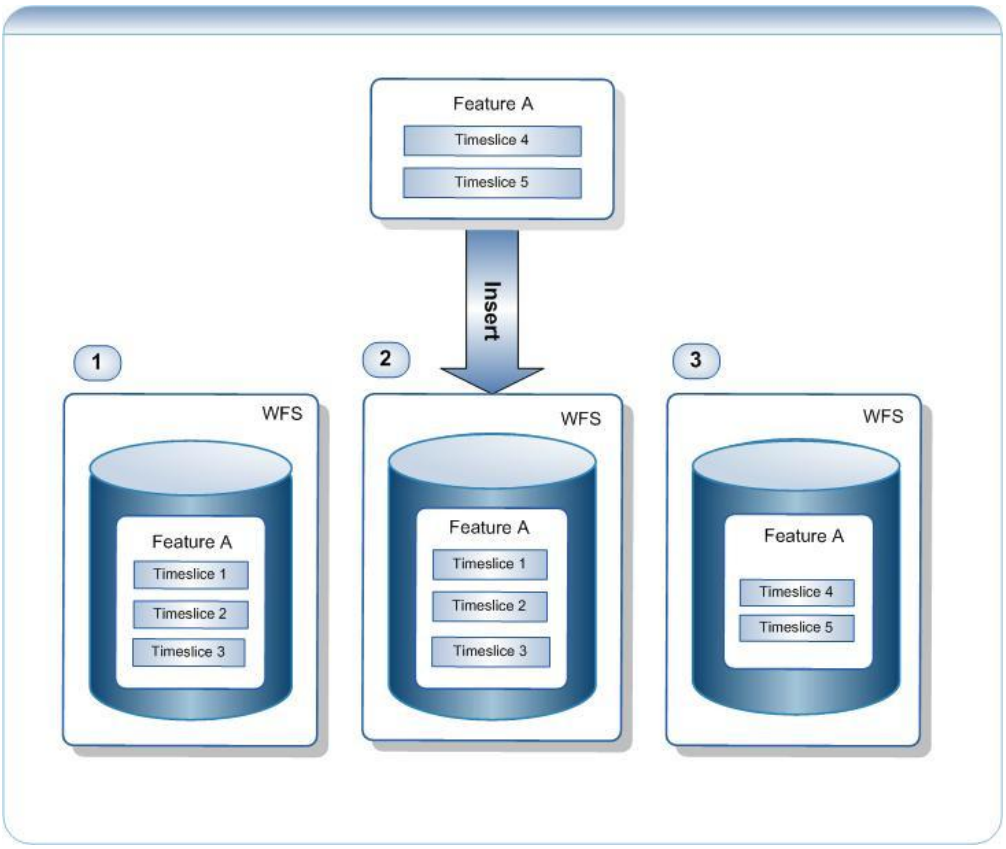


Figure 7. Features in WFS 2.0 as AIXM 5 features

The features offered by the WFS are the time slices in AIXM.

This approach allows the retrieval and filtering of time slices as required by the typical use cases (c.f. Figure 3). However, the filter constraints are limited on the time slice properties. Important properties of the feature like the gml:identifier element used for identification cannot be restricted.
Comsoft worked around this issue by implementing a mix of both approaches: the features offered by WFS are the features in the AIXM world, but the filtering is applied on the time slices (i.e. the returned entities are AIXM features containing a filtered set of time slices). This satisfies the user's needs but doesn't comply with the ISO 19109 General Feature model. Further issues with this compromise are:

- What should the response of a transaction operation count? The number of modified/deleted/inserted features or time slices? Comsoft chose the number of features to be consistent with the standard.
- What should be returned as the numberMatched and numberReturned values in a GetFeature response? Is it the number of features or the number of time slices? Comsoft chose the number of features to be consistent with the standard.
- What does response paging affect? Features or time slices? Comsoft chose the number of time slices, because they indirectly limit the amount of features returned as well and their size is more predictable.

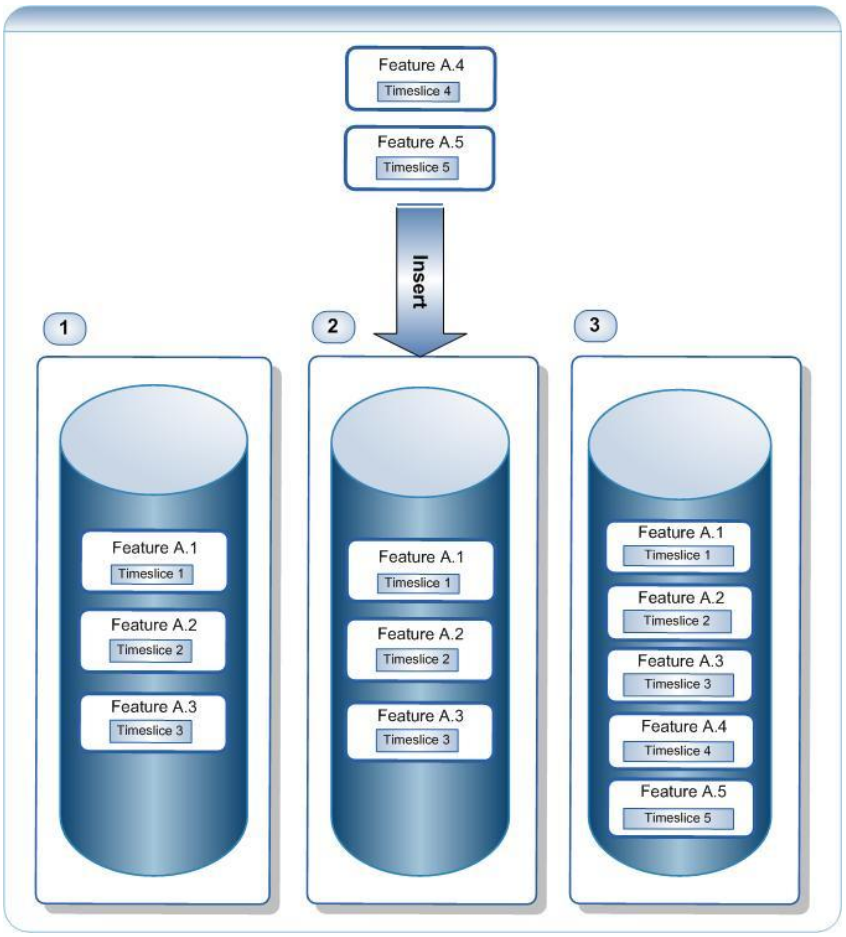


Figure 8. Features in WFS 2.0 as AIXM 5 timeslices

Using WFS 2.0 join queries to retrieve timeslices contained within an AIXM feature?

Join queries do not help here because it isn't possible to define a join clause that links a child property: time slice to its parent: *AIXMFeature*. In other words, there is no common property that a time slice and feature share.

7 Filtering AIXM 5.1 data via OGC WFS 2.0 – T Interface

The OGC WFS 2.0 interface provides two operations to retrieve and filter data:

1. **GetFeature:** allows features to be retrieved based on client specified query expressions
2. **GetPropertyValue:** allows values of selected properties to be retrieved based on client specified query expressions

Within OWS-7 two real world dispatch scenarios were developed to demonstrate how dispatch clients and EFBs will access and use AIXM 5.1 data via the GetFeature and GetPropertyValue operations of the WFS 2.0 interface. Data requests can either be performed using:

- **Simple Queries**: where the client posts a request using pre-defined parameterized filter that can be requested using HTTP GET requests:
    a. Get all instances of a specific feature type:
    &request=GetFeature&TYPENAME=aixm:AirportHeliport
    b. Return all aixm:AirportHeliport and aixm:RunwayElement features within a specified bounding box:
    &request=GetFeature&TYPENAME=aixm:AirportHeliport, aixm:RunwayElement&bbox=-32.5,52,-31.5,53
    c. Get feature by ID:
    &request=GetFeature&TYPENAME=aixm:AirportHeliport&featureid=EDDF.6692a2c0-6b4a-4d43-bf83-ce459397b8eb.

- **Complex Queries:** where the client posts a request which consists of a complex query expression that filters the response based on combinations of query parameters that are submitted via HTTP POST (See Annex I – Query 1 for example HTTP POST Query).
    Example: Dispatcher identifying alternate destination airports for a flight – select all airports located within US airspace and within 200 nautical miles of route that has:
        - A runway that is at least 7,000 ft and has a hard surface
        - Passenger facilities
        - Re-fueling facilities
        - Must not have a scheduled closing time between 15:00 and 23:00

To support the ability to post both simple and complex query requests to retrieve AIXM 5.1 features and properties, both the Comsoft and Snowflake WFS 2.0 servers implemented all of the ISO/DIS 19143 Geographic Information – Filter Encoding Specification (Table 3).

Table 3. Supported ISO/DIS 19143 Filter Encoding query expressions

Query Expression Category	Query Expression
Logical Expressions	And, Or, Not
Comparison Expressions	PropertyIsEqualTo, PropertyIsNotEqualTo, PropertyIsLessThan, PropertyIsGreaterThan, PropertyIsLessThanOrEqualTo, PropertyIsGreaterThanOrEqualTo, PropertyIsLike, PropertyIsNull and PropertyIsBetween
Spatial Filter Expressions	Equals, Disjoin, Touches, Within, Overlaps, Crosses, Intersects, Contains, DWithin, Beyond, BBOX
Temporal Filter Expressions	After, Before, Begins, BegunBy, TContains, During, TEquals, TOverlaps, Meets, OverlappedBy, MetBy, EndedBy

7.1 Filtering AIXM 5.1 using GO Publisher WFS-T 2.0

The WFS 2.0 specification provides two core query types to request AIXM features: i) Ad hoc queries and ii) Stored queries.

7.1.1 Ad hoc queries

Ad hoc queries are requests that are unknown by the server and are constructed of one of more well known parameters and ISO 19143 Filter Encoding expressions that are defined by the client to be posted to the WFS by either HTTP GET or HTTP Post requests.

Within OWS-7, a range of simple and complex queries were defined within the two scenarios. GO Publisher WFS 2.0 successfully demonstrated the ability to support all of the queries defined within both the Dispatch and EFB clients developed by Frequentis and Luciad[3].

However, during internal testing of additional complex queries we identified that there were situations where a client could post a query that aimed to return the 'current state' of an AIXM feature for a specified time instant or range which resulted in an inaccurate response. For example:

The AIXM datastore contained AIXM Features representing a BASELINE and TEMPDELTA timeslice for AirportHeliport: KMOT. The TEMPDELTA represented a

[3] For a comprehensive demonstration of the queries defined within the OWS-7 scenario see https://portal.opengeospatial.org/twiki/bin/view/OWS7/SnowflakeWFS20. NOTE: this is a secure twiki you will need to register as an observer to view the contents.

temporary change to the operational status of the airport, closing the airport between 13:00 and 18:00 on all days between Mon 10th May and Friday 15th May (Figure 9).

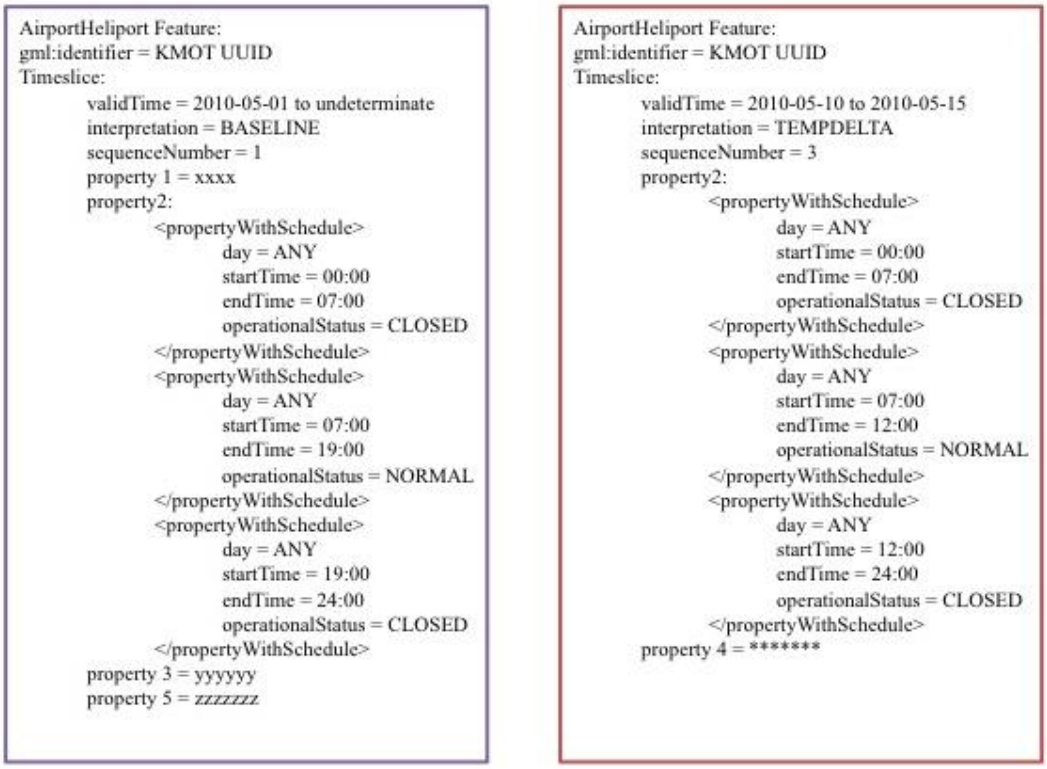


Figure 9. BASELINE and TEMPDELTA timeslice for KMOT AirportHeliport feature instances

During a typical dispatch scenario, a dispatcher may post a request to the WFS 2.0 to identify potential suitable diversion airports while in US Airspace that are operational when the flight is scheduled to be travelling through the airspace between 13:00 and 17:00pm (see Annex I for example HTTP POST request). In this scenario, the request returns the AIXM feature containing the BASELINE timeslice as it states KBOS open while the AIXM feature containing the TEMPDELTA stating it is closed is excluded from the response. This results in a false view of the current state of KMOT.

If the WFS 2.0 served AIXM feature containing a SNAPSHOT timeslice (Figure 10) which enables clients to search against the current effective status, then the query would return a response that excluded KMOT.

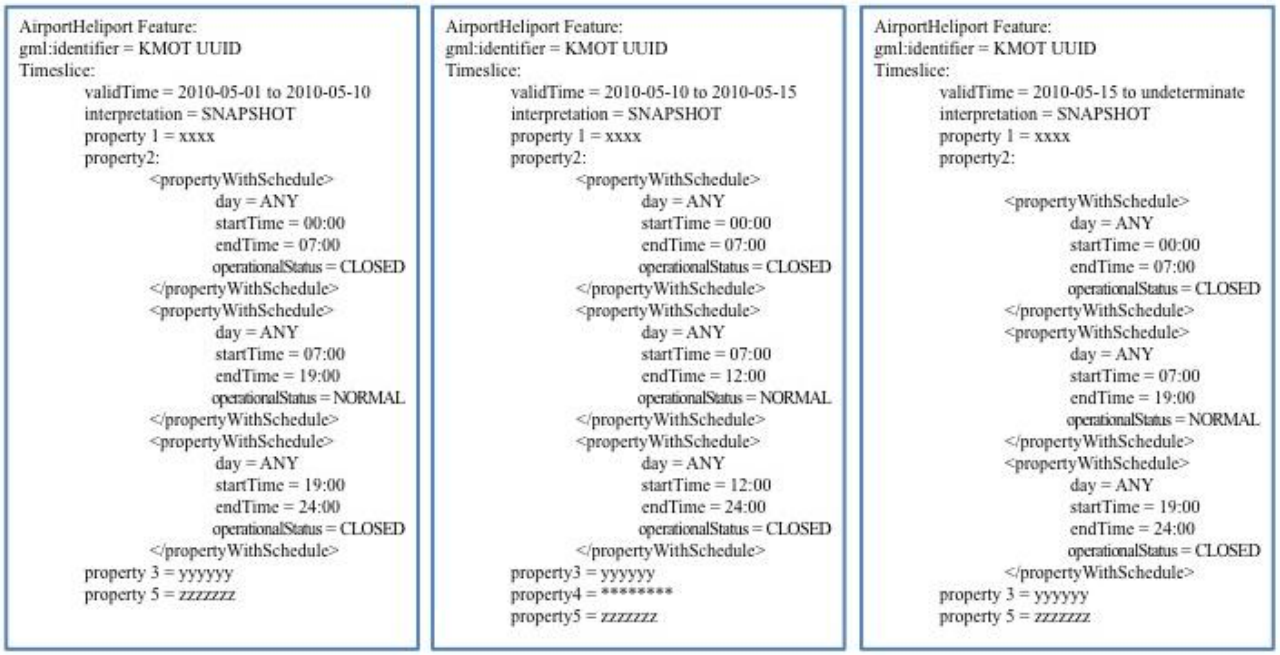


Figure 10. SNAPSHOT timeslices representing the current state for KMOT derived from BASELINE and TEMPDELTA timeslices

There are two options for generating snapshots:

OPTION 1: Client side creation of SNAPSHOT

The client performs a two-step query. In step 1 the client requests all of the BASELINE and TEMPDELTAs that are in effect during the time period requested. In step 2 the client filters through the data to identify if there are any TEMPDELTAs which supercede the BASELINE.

OPTION 2: Server side creation of SNAPSHOT

The service provider generates AIXM features containing SNAPSHOT timeslices representing the current state for a specific time period on-the-fly triggered by the insert of a TEMPDELTA or PERMDELTA via the WFS-T interface.

Of these options, option 2 is preferred as it removes the burden of generating SNAPSHOTs from the client, reduces the amount of data exchanged and ensures that clients can leverage the full query capability of the WFS 2.0.

7.1.2 Stored Queries

During OWS-6-AIM the ability to perform efficient temporal queries using the ISO/DIS 19143 *Geographic Information - Filter Encoding* specification was evaluated. The temporal query functionally of the Filter Encoding was proven to support the aviation

use-cases, however, using the current Filter Encoding Specification temporal queries were deemed unnecessarily complex and verbose. As a result, a change request (FES 2.0 CR 09-066) was submitted to the OGC which was to be subsequently submitted to the ISO TC/211 Committee proposing that an "anyInteracts" temporal filter expression be added. This change request was proposed to remove the requirement to combine with 4 or 8 separate temporal filters to select features based on temporal properties that are "valid" between a specified time period to a single temporal filter. Unfortunately, this change request has not been taken into account to date so clients still have to perform either 4 or 8 separate filters to perform an "anyInteracts" temporal filter.

In order to address the complexity issues identified in OWS6. Snowflake extended its GO Publisher WFS to support the new 'Stored Query' functionality available in WFS 2.0. Stored Queries' are pre-configured queries that are stored within the WFS and can be executed by simply using the query's identifier and passing the appropriate query parameters. The most significant benefits of stored queries is the simplification of complex queries and their ability to be called via HTTP GET as well as HTTP POST.

In OWS-7 GO Publisher WFS supported the following stored query operators:

- **ListStoredQueries**: list of available stored queries offered by the WFS
- **DescribeStoredQueries**: provides information describing the stored query. Including the embedded Filter query,
- **CreateStoredQueries**: insert new stored queries into the WFS

The remaining stored query operation DropStoredQueries was not implemented for the testbed however stored queries could dropped by the WFS administrator using an Administrator interface.

Snowflake configured four stored queries in addition to the mandatory GetFeatureByID stored query:

- **urn:ogc:def:query:OGC-WFS:GetFeatureById (Mandatory)**: the GetFeatureByID stored query enables the client to request features based on their gml:ID.
- **urn:snowflake:def:query:AirportHeliportTemporalBufferQuery**: This stored query accepts the parameters: "beginPosition","endPosition", "lineString" and "distance" to return AirportHeliports within a buffer distance of a linestring or point
- **urn:snowflake:def:query:OGC-WFS:AirportHeliportByDesignatorQuery**: this stored queries accepts the parameter "designator" to return all AirportHeliport features containing the requested designator (e.g. KJFK, KBOS)

- **urn:snowflake:def:query:gmlIdentifierTemporalQuery**: this stored query aims to simplify a long HTTP POST query requesting timeslices for a specified validTime range for specific AirportHeliports based on their gml:ID.

- **urn:snowflake:def:query:PropertyIsEqualToQuery**: this stored query will return AirportHeliport features based on their path length.

Example stored queries can be found in Annex II.

7.1.3 GetPropertyValue

The WFS 2.0 specification also has extended the mechanisms for retrieving data through the inclusion of an additional request operation: GetPropertyValue. GetPropertyValue enables the retrieval of specific property values rather than the whole feature which contains the property values. For example, where a dispatcher or pilot only wants to be presented with a list of suitable diversion airports.

Example GetPropertyValue operations can be found in Annex III.

7.2 Filtering AIXM 5.1 using Comsoft AIXM 5.1 WFS-T 2.0

7.2.1 Overview

One design principle of CADAS-AIM$_{DB}$ is the interoperability with other systems. As the database is the core of any integrated AIM solution an open interface that can be used independently from any platform and programming language is one of the key features. On the other side the interface has to fit the special use cases of aeronautical data without restricting the access to a limited set of pre-defined functions. For an optimal support of AIXM 5 CADAS-AIM$_{DB}$ provides the CAW-interface. However, after some modifications also the WFS 2.0-T standard can be used for accessing AIXM 5 data.

7.2.2 WFS Functionality

The *GetFeature* function is the central place to retrieve AIXM 5 compliant data in the form of Time Slices. Retrieving only certain properties of a feature is available. Thereby the correctness of the result against the AIXM schema is ensured. This means that all properties are placed in correct Time Slice data structures and that any object relations are fully expressed.

Selecting the data is done by specifying one or more filter criteria according to the AIXM constraints. All these filter criteria a expressed in the query syntax as defined by ISO 19143. All available filters can be differentiated into the following classes:

- Basic selection rules are provided by the AIXM Filter Criteria that restrict the queries to one or more AIXM feature types and that provides simple expressions to search for specific Time Slice types. This filter also provides the type Snapshot that creates an aggregation of multiple Time Slices representing a specific feature state at a

point in time or during a time period (see the AIXM specification for an exact definition of Snapshot Time Slices).

- The Temporal Filter Criteria address the core of all AIXM data which is the inherent temporality of all kinds of information. These criteria provide a bi-temporal selection using either single time stamps or time periods. The bi-temporality is expressed in a Functional Time ("when are the information valid") and a Technical Time ("since when do we know about that information ").

- The Property Filter Criteria provide a simple selection on the values by arbitrary feature properties. All AIXM data types together with the respective comparison operators are supported. Property filter can be applied on the properties on the Time Slice level as well as on the Object level.

- A specialized version of the Property Filter Criteria exist for geospatial information in form, the so called Geospatial Filter Criteria. With these criteria all AIXM 5 features can be searched that have a location assigned (in the form of a point or boundary). They provide the capabilities of the proprietary geospatial database extension as standardized WFS queries to all clients and users. Geospatial comparison operators depending on the actual data type as defined in ISO 18143 are supported.

- Join Filter Criteria are used to traverse the linking between multiple AIXM 5.1 features. Thereby only references that are defined in the AIXM specification can be used. This is ensured by specifying the association that creates the link as defined in the AICM. Only stating the feature types that should be joined is not sufficient for AIXM data and is thus not supported. When using a Join Filter Criteria always all affected features are returned by the GetFeature function.

8 Updating AIXM 5.1 data via the OGC WFS-T 2.0 interface

Within OWS6-AIM Events were triggered after the insertion of Digital NOTAMs into the datastore using offline data loading tools (GO Loader). OWS-7 aimed to demonstrate the ability to publish AIM Events as a result of WFS Transactions enabling clients to directly update a remote AIXM datastore via a standard based web service. Timeslices were created using WFS-T Inserts which in turn created Digital NOTAMs that were passed onto the Event service.

The AIXM 5.1 Temporality Model states that no instances of a timeslice shall be deleted or updated. The AIXM 5.1 Temporality Model requires that any change to an AIXM Feature will be performed through the insertion of a new AIXM feature instance that contains one or more PERMDELTA or TEMPDELTA timeslices.

The OWS-7 aviation thread was not designed to provide a comprehensive demonstration of the long-term maintenance of an AIXM 5.1 datastore using the WFS-T 2.0 interface due to the short timeframes and ambitious nature of the thread. However, both WFS providers were able to demonstrate the ability to insert new features containing TEMPDELTA timeslices which triggered the generation of NOTAMs which were published through the Event Service and retrieval of the inserted TEMPDELTAs via the WFS 2.0.

8.1 Snowflake GO Publisher WFS-T and Event Publisher

8.1.1 Inserting AIXM 5.1 timeslices via GO Publisher WFS-T 2.0

GO Publisher WFS-T combines the loading and schema translation and capability of GO Loader alongside the translation and publication capabilities of GO Publisher WFS. For the OWS-7-AIM only the WFS INSERT transaction was supported as the AIXM Temporality Model assumes that baseline data will not be modified or edited, only new perm or temp delta added. As a result when changes to the state of an aeronautical feature occur or correction are made these are inserted as a new gml Features with Timeslice.

An example of a WFS-T Insert can be found in Annex IV.

8.1.2 Integration of WFS-T with the Event Service

The publication of events re-used the architecture developed within OWS-6. The Event Source Publication system is based on GO Publisher Agent which is a command-line bulk data publishing system. This system was been extended by the development of an event pusher, which posts the events created by GO Publisher Agent to the Event service using SOAP messages.

To enable publication of events based on the insertion of a time slice into the database, a Change Detection Module is installed on the database side and uses APIs or triggers to identify inserts.

Triggers on the database tables populate a GO Publisher Agent queue database table with the id of the new Timeslice. GO Publisher Agent reads the queue and constructs a Digital NOTAM. For NOTAM updates for non-geographic features GO Publisher Agent extracts the geometries from related features and constructs a bounding box and these events are published to an FTP directory. The Event Pusher subscribes to the Event service and polls the FTP directory. Once events are published to FTP directory, the Event Pusher then pushes them to the event service which subsequently forwards them to clients.

The information flow diagram in Figure 11 shows the message exchange in the pre-flight and in-flight conditions and inter-operations between the Clients, WFS and the Event Service.

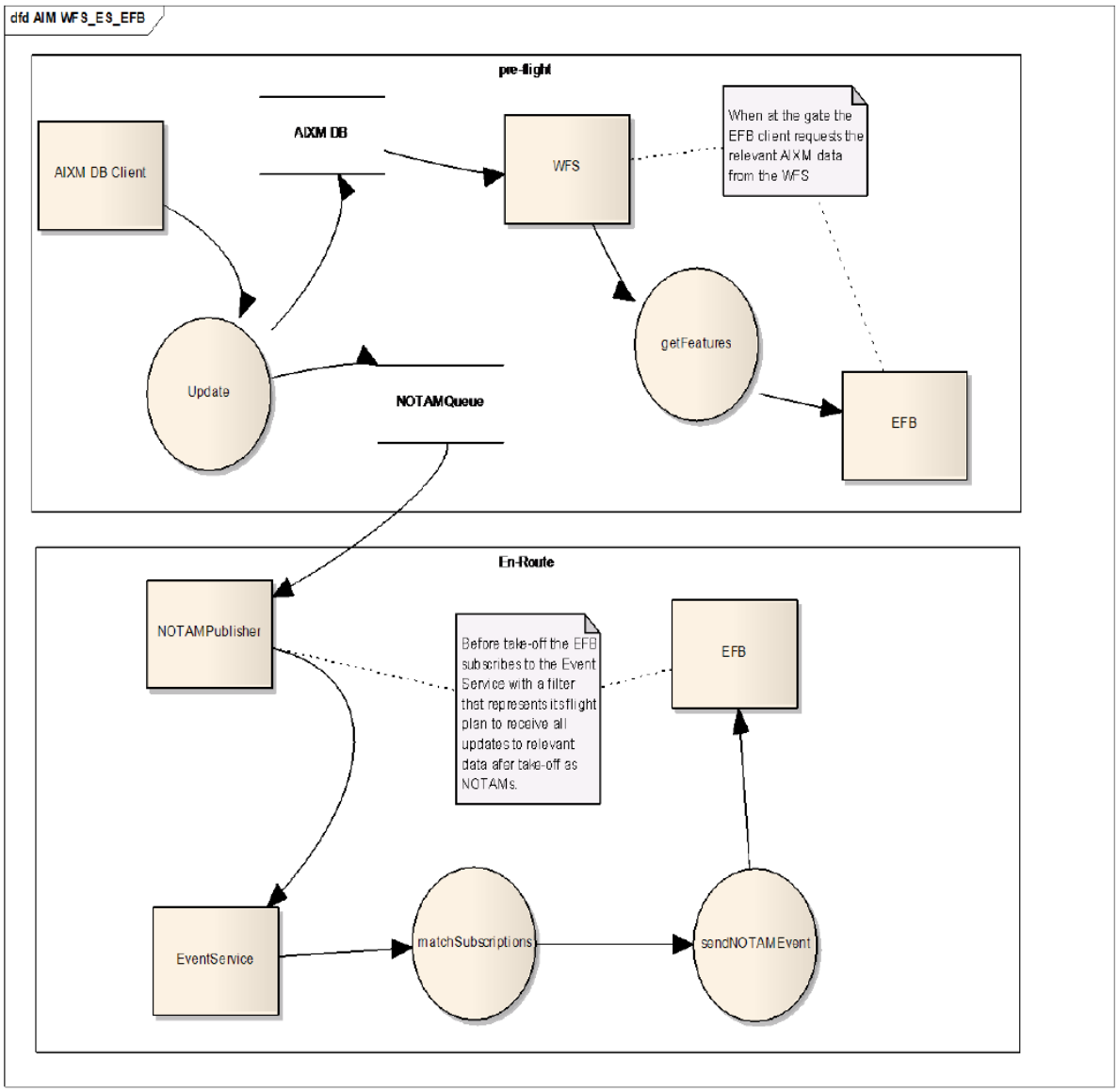


Figure 11 - Generation of events triggered by insertion of time slices

The updating of the AIXM database causes a trigger to create an update alert that is written to a Queue. The queue is monitored by the NOTAM Agent and a NOTAM event is populated via a WFS request providing the feature ID of the changed AIXM feature and then sent to the NOTAM queue from where it is fetched by a Publisher service that pushes the NOTAM event to the Event Service for subscription matching and propagation to matching subscribers as is shown in Figure 12.

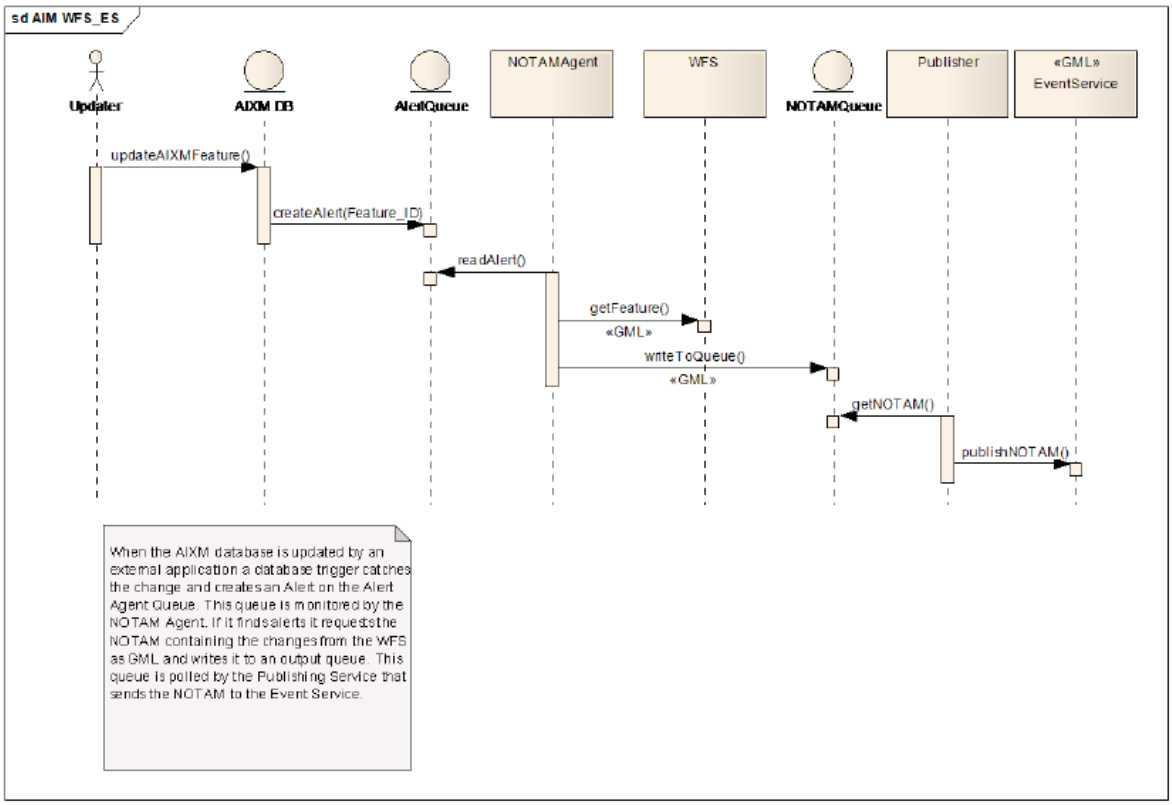


Figure 12 - Sequence Diagram summarizing the GO Publisher event source architecture

8.2 Comsoft WFS-T Data modification with the *Transaction* function

The Transaction method provides all operations for manipulating the contents of CADAS-AIMDB. Data manipulation is always done by means of AIXM data structures.

For inserting data into the database the Insert operation is provided. This operation is the standard way to insert new AIXM 5 Time Slices. With this operation Time Slices for both event types (permanent and temporary delta) can be inserted. Also corrections and cancellations can be issued. By inserting respective Time Slices, the commissioning and decommissioning of AIXM 5 features is handled too.

As AIXM 5 defines the Time Slice types Baseline and Snapshot as aggregations of multiple permanent or temporary deltas these Time Slices types cannot be inserted directly. Instead the respective deltas have to be inserted.

9 Lessons Learned

9.1 AIXM 5.1 Temporality Model

The AIXM 5.1 Temporality Model posed some challenges for the server providers to understand how to implement the model to ensure that data can be effectively retrieve via the pull mechanism of the WFS 2.0 interface. Criticisms were raised to the widely agreed approach to encode each instance of a timeslice as a separate feature sharing the same gml:identifier, thereby logically conforming to the model.

However, there are some minor amendments required to the AIXM 5.1 Temporality Model. The key amendment required to is change the SNAPSHOT timeslice from defining the current state of an AIXM feature at a particular time instant to either a time instant or time period. Both the servers and the clients, recognize that the ability to generate a SNAPSHOT that defines the current state over a specific time period is essential to flight planning and en-route assistance.

9.2 Complexity of the Filter Encoding Specification

Ad hoc queries using the Filter Encoding Specification have many advantages, as the server does not need to be configured to support the wide range of different requests that may be performed on the data. However, this does place a requirement on client providers to be able to understand and create complex filter encodings using intuitive user interfaces. Both the new client and server providers who were not involved in OWS-6 raised the issue about the over complexity of generating a temporal query to restrict the response to return timeslices that are valid during a specified timeperiod. This issues is compounded as most queries to be performed must specify a validTime to constrain the response to only return the features in effective during the time period of a flight and not all past and future features that may also correspond to the query. Such complexity removes the ability for a client to perform simple HTTP Get requests and must support HTTP Post requests.

It is therefore recommended that the OGC and OGC Aviation DWG follow up the change request submitted in OWS-6 (FES 2.0 CR 09-066) to develop a corrigendum or addendum to the ISO 19143 Filter Encoding specification to add an “anyInteracts” temporal filter expression.

9.3 Effective retrieval of current state using BASELINE and DELTA timeslices only

Within the OWS-7 scenario, several queries were identified that the most effective method for querying the data would require the generation of a SNAPSHOT timeslice by the service provide to represent the current state rather than allow the client to generate these on the fly.

The generation of a SNAPSHOT timeslice requires an understanding of complex rules to trigger the generation of a SNAPSHOT within the database on insert of a TEMPDELTA or PERMDELTA via the WFS-T interface. These rules have not been well developed

within the AIXM 5.1 Temporality Model so further work is required in conjunction with FAA and EUROCONTROL to define these for implementation within an OGC Web Services environment.

9.4 Exception Reporting within WFS-T

All of the transactional WFS operations (insert, update, replace, delete) share an all or nothing policy. Either all requested operations are completely successful or they fail at all. In some cases, e.g. the modification of features or time slices, it may be helpful to allow a partial success, too. In the response of a transaction operation the specification allows the server to return the total number of inserted, updated, replaced or deleted features to the client. However, if, for any reason, this number doesn't equal the number of features in the request, there is no place to report the reasons for the partial failure back to the client. The server is left with two options: either fail completely and report an exception or fail partially and let the client be unaware of the reasons of the failure and the affected features/time slices.

The specification could be improved by allowing a place for warnings or notes in the response that could be displayed or processed by the client. This should be an optional feature available on the client's request only, because the client may not be able to deal with partial successes.

9.5 Formal deficiencies of the XPath subset

The EBNF for the XPath subset is incomplete. The specification defines a mandatory "or" operator to be used in an "OrExpr", which is the only alternative to an "NumbericExpr" [sic] in an "Expr". The same is true for "AndExpr". The definitions should be altered in the following way (note the question marks at the end of line):

- OrExpr ::= AndExpr ("or" AndExpr)?
- AndExpr ::= ComparisonExpr ("and" ComparisonExpr)?

A FunctionCall should be able to have parameters other than "."

9.6 gml:ID

Resource Ids shouldn't be based on XML IDs like gml:id. According to the specification, every feature offered by a WFS service should be identifiable by a unique resource id.

> *"7.2.1 Assignment*
>
> *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. "*

Further on, as AIXM is an GML application, the resource id for AIXM features should be encoded in the gml:id attribute of the feature element.

> *"7.2.2 Encoding*
>
> > *For features encoded using GML, the resource identifier shall be encoded in the XML attribute gml:id."*

The gml:id is of the XML type ID which bears the constraint of being unique in a single XML document. Now, if the client is using a GetFeature request with multiple queries or a join queries, the response may contain the same feature multiple times. In this is case it is impossible to fulfill both document-wide gml:id uniqueness and service-wide resource id uniqueness at the same time.

How about XLinks? The use of XLinks does not help because it allows only the linking to whole features (in wfs:member elements), not individual time slices (i.e. this won't work if the response contains the same feature multiple times with different time slices).

The proposed solution is to change the specification and to drop the requirement to use gml:id as the resource id and use the gml:identifier (UUID) instead.

According to the specification, every feature offered by a WFS service should be identifiable by a unique resource id. Further on, for features based on gml, the resource id should be encoded in the gml:id attribute of the feature element. The gml:id is of the XML type ID which bears the constraint of being unique in a single XML document. Now, if the client is using a GetFeature request with multiple queries, the response may contain the same feature multiple times. To assure a valid xml document is returned, the server has to make use of the XLink standard to comply with the document-uniqueness constraint of gml:ids. This complicates server and client implementation, and, for the server, may result in extra processing time for identifying feature duplicates and replacing them with XLinks.

10 Accomplishments

The OWS-7 AIXM Assessment thread successfully achieved to demonstrate the ability to serve, filter and update AIXM 5.1 data using OGC Web Services within the tight timescales provided. Key accomplishments in OWS-7 AIXM Assessment thread included:

1. All aspects of AIXM 5.1 can be loaded, queried and updated over WFS 2.0;
2. Filter Encoding Specification 2.0 (including Temporal Operator) is fully capable of supporting all requirements of both scenarios;
3. WFS-T 2.0 Insert transactions can insert new temp and perm deltas timeslices;
4. WFS-T 2.0 Insert operations are capable of triggering the creation of Digital NOTAM;
5. Digital NOTAM generated from WFS-T 2.0 Insert can pushed to an Event Service via WS-Notification;
6. WFS 2.0 Stored Queries can be used to simplify complex Filter Encoding queries;
7. WFS 2.0 can be delivered over SOAP enabling deployment on the FUSE Enterprise Service Bus (See FUSE Integration ER for more details)
8. AIXM 5.1 Schedules can be loaded, served, queried and updated over WFS-T 2.0;
9. WFS responses can be validated against the AIXM Schematron Business Rules;

11 Next Steps

- Address the conflict between the AIXM Temporality model and OGC Feature Model. Develop best practice guidance for using AIXM over WFS;

- Develop more scenarios to test SNAPSHOT queries and how these can be deployed via WFS;

- Further test WFS Stored Queries, specifically testing client integration for creating, uploading and deleting Stored Queries and how Stored Queries can be orchestrated within a Service Orientated Architecture;

- Integrate the execution of Schematron business rules into the WFS response validation;

- Test the full spectrum of geometry types utilized in AIXM5.1, including: gml:Arc, gml:ArcString, gml:ArcStringByBulge, gml:ArcByBulge, gml:ArcByCenterPoint, gml:Circle, gml:CircleByCenterPoint gml:Geodesic, gml:GeodesicString, gml:BoundingShapeType. Investigate the use of these geometry types in WFS Filter queries;

- Investigate the use of WFS 2.0 Additional Objects as a means of simplifying xlink referencing and resolving;

- Increase data volumes in the underlying data stores and carry out performance testing. Investigate and benchmark the use of XML compression technologies such as Efficient XML (EXI) over WFS;

Annex I –Example GetFeature Request

Complex spatial-temporal query to retrieve only runways >2000m with hard surface within a 200 nautical miles of KBOS

```
<?xml version="1.0" encoding="UTF-8"?>
<wfs:GetFeature
…..
      <wfs:Query typeNames="aixm:AirportHeliport" handle="Q01">
            <fes:Filter>
                  <fes:And>
                        <fes:DWithin>
                              <fes:ValueReference>aixm:timeSlice//aixm:ElevatedPoint</fes:ValueReference>
                              <gml:LineString srsName="urn:ogc:def:crs:ogc:1.3:CRS84"
xmlns:gml="http://www.opengis.net/gml/3.2">
                                    <gml:pos>-97 32.5 -96 33</gml:pos>
                              </gml:LineString>
                              <fes:Distance units="nautical mile">20</fes:Distance>
                        </fes:DWithin>
                        <fes:Not>
                              <fes:Or>
                                    <fes:Before>
                                          <fes:ValueReference>aixm:timeSlice//gml:validTime</fes:ValueReference>
                                          <fes:Literal>
                                                <gml:TimePeriod gml:id="id1">
                                                      <gml:beginPosition>2009-10-30T00:00:00.000Z</gml:beginPosition>
                                                      <gml:endPosition>2009-10-31T00:00:00.000Z</gml:endPosition>
                                                </gml:TimePeriod>
                                          </fes:Literal>
                                    </fes:Before>
                                    <fes:After>
                                          <fes:ValueReference>aixm:timeSlice//gml:validTime</fes:ValueReference>
                                          <fes:Literal>
                                                <gml:TimePeriod gml:id="id2">
                                                      <gml:beginPosition>2009-10-30T00:00:00.000Z</gml:beginPosition>
                                                      <gml:endPosition>2009-10-31T00:00:00.000Z</gml:endPosition>
                                                </gml:TimePeriod>
                                          </fes:Literal>
                                    </fes:After>
                                    <fes:Meets>
                                          <fes:ValueReference>aixm:timeSlice//gml:validTime</fes:ValueReference>
                                          <fes:Literal>
                                                <gml:TimePeriod gml:id="id3">
                                                      <gml:beginPosition>2009-10-30T00:00:00.000Z</gml:beginPosition>
                                                      <gml:endPosition>2009-10-31T00:00:00.000Z</gml:endPosition>
                                                </gml:TimePeriod>
                                          </fes:Literal>
                                    </fes:Meets>
                                    <fes:MetBy>
                                          <fes:ValueReference>aixm:timeSlice//gml:validTime</fes:ValueReference>
                                          <fes:Literal>
                                                <gml:TimePeriod gml:id="id4">
                                                      <gml:beginPosition>2009-10-30T00:00:00.000Z</gml:beginPosition>
                                                      <gml:endPosition>2009-10-31T00:00:00.000Z</gml:endPosition>
                                                </gml:TimePeriod>
                                          </fes:Literal>
                                    </fes:MetBy>
                              </fes:Or>
                        </fes:Not>
```

```
                </fes:And>
            </fes:Filter>
        </wfs:Query>
</wfs:GetFeature>
```

Select a suitable diversion airport while flight passes through US Airspace.

To identify potential suitable diversion airports while in US Airspace a dispatcher is required to identify only those Airports that are operational when the flight is scheduled to be travelling through the airspace (i.e. between 13:00 and 17:00pm). So the dispatcher is required to perform a query to select only AirportHeliports that are scheduled to be operational on 10th May between 13:00 and 17:00 pm.

```
<?xml version="1.0" ?>
<wfs:GetFeature
service="WFS"
version="2.0.0"
outputFormat="application/gml+xml; version=3.2"
xmlns:wfs="http://www.opengis.net/wfs/2.0"
xmlns:aixm="http://www.aixm.aero/schema/5.1"
xmlns:ogc="http://www.opengis.net/ogc"
xmlns:gml="http://www.opengis.net/gml/3.2"
xmlns:fes="http://www.opengis.net/fes/2.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.opengis.net/wfs/2.0
                http://www.pvretano.com/schemas/wfs/2.0.0/wfs.xsd
                http://www.opengis.net/gml/3.2
                http://schemas.opengis.net/gml/3.2.1/gml.xsd">
<wfs:Query typeName="aixm:AirportHeliport" handle="Q01"> <fes:Filter> <fes:And>
            <fes:PropertyIsEqualTo>
                        <fes:ValueReference>gml:identifier</fes:ValueReference>
                        <fes:Literal>urn-x:ows7:snowflake:KMOT</fes:Literal>
            </fes:PropertyIsEqualTo>
            <fes:Or>
                <fes:PropertyIsEqualTo>
                    <fes:ValueReference>//aixm:interpretation</fes:ValueReference>
                    <fes:Literal>BASELINE</fes:Literal>
                </fes:PropertyIsEqualTo>
                <fes:PropertyIsEqualTo>
                    <fes:ValueReference>//aixm:interpretation</fes:ValueReference>
                    <fes:Literal>TEMPDELTA</fes:Literal>
                </fes:PropertyIsEqualTo>
            </fes:Or>
    <fes:Or>
            <fes:And>
                <fes:PropertyIsLessThanOrEqualTo>
                    <fes:ValueReference>//aixm:startTime</fes:ValueReference>
                    <fes:Literal>13:00</fes:Literal>
                </fes:PropertyIsLessThanOrEqualTo>
                <fes:PropertyIsGreaterThanOrEqualTo>
                    <fes:ValueReference>//aixm:endTime</fes:ValueReference>
                    <fes:Literal>17:00</fes:Literal>
                </fes:PropertyIsGreaterThanOrEqualTo>
```

```
            <fes:Or>
                <fes:PropertyIsEqualTo>
                            <fes:ValueReference>//aixm:day</fes:ValueReference>
                            <fes:Literal>MON</fes:Literal>
                </fes:PropertyIsEqualTo>
                        <fes:PropertyIsEqualTo>
                            <fes:ValueReference>//aixm:day</fes:ValueReference>
                            <fes:Literal>ANY</fes:Literal>
                </fes:PropertyIsEqualTo>
            </fes:Or>
        </fes:And>
        <fes:And>
            <fes:PropertyIsNull>
                <fes:ValueReference>//aixm:endTime</fes:ValueReference>
            </fes:PropertyIsNull>
            <fes:PropertyIsNull>
                <fes:ValueReference>//aixm:startTime</fes:ValueReference>
            </fes:PropertyIsNull>
        </fes:And>
    </fes:Or>
    <fes:Or>
            <fes:PropertyIsNotEqualTo>
                <fes:ValueReference>//aixm:operationalStatus</fes:ValueReference>
                <fes:Literal>CLOSE</fes:Literal>
            </fes:PropertyIsNotEqualTo>
            <fes:PropertyIsNull>
                <fes:ValueReference>//aixm:operationalStatus</fes:ValueReference>
            </fes:PropertyIsNull>
    </fes:Or>
    <fes:Not>
        <fes:Or>
            <fes:Before>
                <fes:ValueReference>aixm:timeSlice//gml:validTime</fes:ValueReference>
                <fes:Literal>
                    <gml:TimePeriod gml:id="id1">
                        <gml:beginPosition>2010-05-10T13:00:00.000Z</gml:beginPosition>
                        <gml:endPosition>2010-05-10T17:00:00.000Z</gml:endPosition>
                    </gml:TimePeriod>
                </fes:Literal>
            </fes:Before>
            <fes:After>
                <fes:ValueReference>aixm:timeSlice//gml:validTime</fes:ValueReference>
                <fes:Literal>
                    <gml:TimePeriod gml:id="id2">
                        <gml:beginPosition>2010-05-10T13:00:00.000Z</gml:beginPosition>
                        <gml:endPosition>2010-05-10T17:00:00.000Z</gml:endPosition>
                    </gml:TimePeriod>
                </fes:Literal>
            </fes:After>
            <fes:Meets>
                <fes:ValueReference>aixm:timeSlice//gml:validTime</fes:ValueReference>
                <fes:Literal>
                    <gml:TimePeriod gml:id="id3">
                        <gml:beginPosition>2010-05-10T13:00:00.000Z</gml:beginPosition>
```

```
                        <gml:endPosition>2010-05-10T17:00:00.000Z</gml:endPosition>
                    </gml:TimePeriod>
                </fes:Literal>
            </fes:Meets>
            <fes:MetBy>
                <fes:ValueReference>aixm:timeSlice//gml:validTime</fes:ValueReference>
                <fes:Literal>
                    <gml:TimePeriod gml:id="id4">
                        <gml:beginPosition>2010-05-10T13:00:00.000Z</gml:beginPosition>
                        <gml:endPosition>2010-05-10T17:00:00.000Z</gml:endPosition>
                    </gml:TimePeriod>
                </fes:Literal>
            </fes:MetBy>
        </fes:Or>
    </fes:Not>
</fes:And>
</fes:Filter>
</wfs:Query>
</wfs:GetFeature>
```

Annex II –Example Stored Queries

The complexity of ad hoc queries required for retrieving AIXM timeslice features has been highlighted as an issue in both OWS-6 and OWS-7, with particular reference to temporal querying. To understand the benefits that the new OGC WFS 2.0 Stored Queries operations can provide to simplify large complicated queries to parameterized queries that can be posed as GET queries, Snowflake created 4 custom stored queries based upon queries that would commonly be performed in flight planning scenarios:

- **urn:snowflake:def:query:AirportHeliportTemporalBufferQuery**: This stored query accepts the parameters: "beginPosition","endPosition", "lineString" and "distance" to return AirportHeliports within a buffer distance of a linestring or point

- **urn:snowflake:def:query:OGC-WFS:AirportHeliportByDesignatorQuery**: this stored queries accepts the parameter "designator" to return all AirportHeliport features containing the requested designator (e.g. KJFK, KBOS)

- **urn:snowflake:def:query:gmlIdentifierTemporalQuery**: this stored query aims to simplify a long HTTP POST query requesting timeslices for a specified validTime range for specific AirportHeliports based on their gml:ID.

- **urn:snowflake:def:query:PropertyIsEqualToQuery**: this stored query will return AirportHeliport features based on their path length.

Custom Stored queries are reusable FES 2.0 filter queries which are stored by the WFS 2.0. The stored queries are standard WFS FES 2.0 queries which have some parameters replaced by variables (${*variable*}). The client accesses a stored query stored in the WFS by requesting the filter by name and including values for the variable parameters. To understand what stored queries are available within a server, the WFS 2.0 can be queried for a list of available stored queries and a description including the FES2.0 filter and variable parameters used can be obtained using the ListStoredQueries and DescribeStoredQueries operations:

- **ListStoredQueries:** http://demo.snowflakesoftware.com:8080/OWS7_AIXM51_WFS20/GOPublisherWFS?service=WFS&version=2.0.0&request=ListStoredQueries
- **DescribeStoredQueries:** custom stored query for AirportHeliportTemporalBufferQuery http://demo.snowflakesoftware.com:8080/OWS7_AIXM51_WFS20/GOPublisherWFS?service=WFS&version=2.0.0&request=DescribeStoredQueries&STOREDQUERY_ID=urn:snowflake:def:query:AirportHeliportTemporalBufferQuery

The Stored queries within the WFS can be tailored to the particular application requirement and provides a simpler interface for clients requiring the reuse of large complicated filter requests. For example, the urn:snowflake:def:query:AirportHeliportTemporalBufferQuery simplifies a query to return all airportHeliports within a specified buffer along a flight path by defining the following query parameters:

- **beginPosition:** start time of validity period of interest
- **endPosition:** end time of the validity period of interest
- **distance:** buffer distance (in nautical miles) around the flight path within which to search
- **lineString:** geometry of the flight path (or portion of flight path)

The example below shows how the FES 2.0 filter expression is defined as stored within a WFS2.0 Stored Query

<![CDATA[<wfs:Query xmlns:wfs="http://www.opengis.net/wfs/2.0" xmlns:fes="http://www.opengis.net/fes/2.0" xmlns:gml="http://www.opengis.net/gml/3.2" xmlns:aixm="http://www.aixm.aero/schema/5.1" typeName="aixm:AirportHeliport" handle="Q01"><fes:Filter><fes:And><fes:DWithin><fes:ValueReference>aixm:timeSlice//aixm:ElevatedPoint</fes:ValueReference><gml:LineString srsName="urn:ogc:def:crs:ogc:1.3:CRS84" xmlns:gml="http://www.opengis.net/gml/3.2"><gml:pos>${lineString}</gml:pos></gml:LineString><fes:Distance units="nautical mile">${distance}/<fes:Distance></fes:DWithin><fes:Not><fes:Or><fes:Before><fes:ValueReference>aixm:timeSlice//gml:validTime</fes:ValueReference><fes:Literal><gml:TimePeriod gml:id="id1"><gml:beginPosition>${beginPosition}</gml:beginPosition><gml:endPosition>${endPosition}</gml:endPosition></gml:TimePeriod></fes:Literal></fes:Before><fes:After><fes:ValueReference>aixm:timeSlice//gml:validTime</fes:ValueReference><fes:Literal><gml:TimePeriod gml:id="id2"><gml:beginPosition>${beginPosition}</gml:beginPosition><gml:endPosition>${endPosition}</gml:endPosition></gml:TimePeriod></fes:Literal></fes:After><fes:Meets><fes:ValueReference>aixm:timeSlice//gml:validTime</fes:ValueReference><fes:Literal><gml:TimePeriod gml:id="id3"><gml:beginPosition>${beginPosition}</gml:beginPosition><gml:endPosition>${endPosition}</gml:endPosition></gml:TimePeriod></fes:Literal></fes:Meets><fes:MetBy><fes:ValueReference>aixm:timeSlice//gml:validTime</fes:ValueReference><fes:Literal><gml:TimePeriod gml:id="id4"><gml:beginPosition>${beginPosition}</gml:beginPosition><gml:endPosition>${endPosition}</gml:endPosition></gml:TimePeriod></fes:Literal></fes:MetBy></fes:Or></fes:Not></fes:And></fes:Filter></wfs:Query>]]>

Using these parameters the client can construct a GET WFS query which is far simpler and concise compared to a comparable a normal spatial and temporal POST query:

Resultant AirportHeliportTemporalBufferQuery Stored Query:

http://demo.snowflakesoftware.com:8080/OWS7_AIXM51_WFS20/GOPublisherWFS?service=WFS&version=2.0.0&request=GetFeature&STOREDQUERY_ID=urn:snowflake:def:query:AirportHeliportTemporalBufferQuery&beginPosition=2009-10-30T00:00:00.000Z&endPosition=2009-10-31T00:00:00.000Z&distance=20&lineString=-97%2032.5%20-96%2033

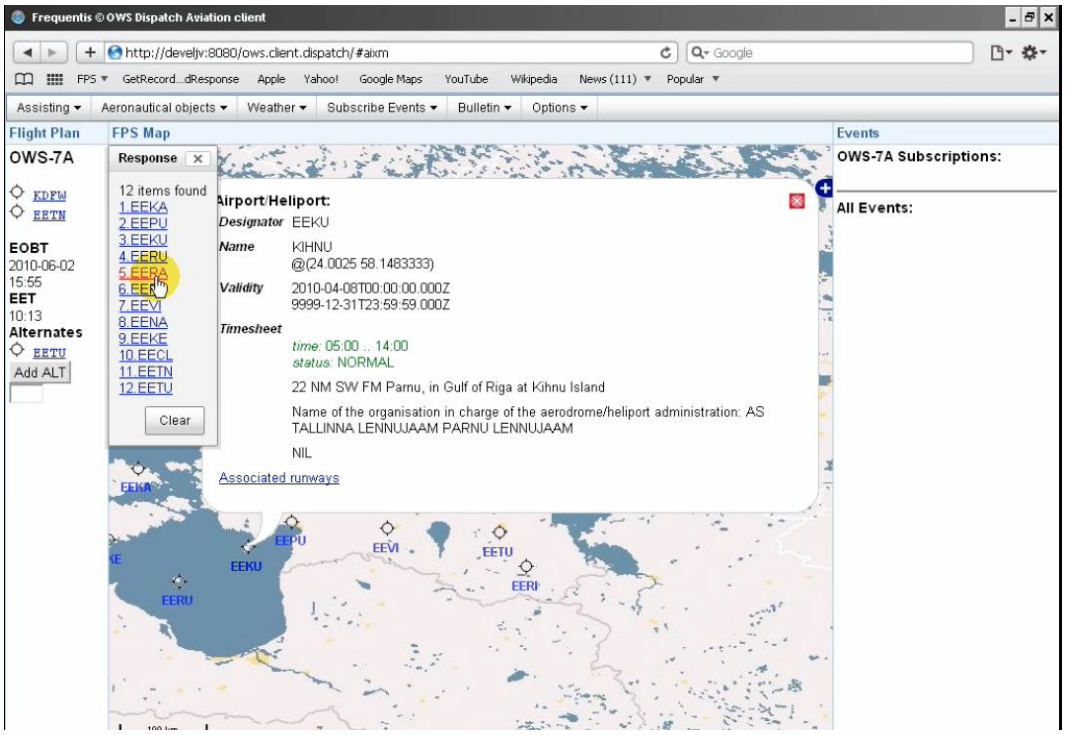
The initial investigation of the applicability of defining custom stored queries for more efficient retrieval of AIXM 5.1 within aviation clients has demonstrated that offer significant benefits:

- Simplified queries which should make it easier to develop client applications with complex query support
- Less likely for client developers to build invalid complex queries
- Provides a more RESTful interface

Annex III – GetProperty Requests

Retrieving AIXM 5.1 property values using GetPropertyValue queries

The GetPropertyValue operation is a new operation within the WFS 2.0 specification. It has been included as it was recognized that there are cases where the client does not need the complete feature returned, just the values for specified properties meeting the query criteria. In OWS-7-AIM GetPropertyValue was used by the Frequentis client to populate menus for user choices, as below:



Within OWS-7 scenarios no explicit query requirements were identified for using the GetPropertyValue operations, however, Snowflake demonstrated how the GetPropertyValue query works for two potential queries:

GetPropertyValue query 1: Return the centroid coordinates for an AirportHeliport

Request:

```
<?xml version="1.0" ?>
<wfs:GetPropertyValue
service="WFS"
version="2.0.0"
valueReference="//gml:validTime"
outputFormat="application/gml+xml; version=3.2"
xmlns:wfs="http://www.opengis.net/wfs/2.0"
```

```
xmlns:aixm="http://www.aixm.aero/schema/5.1"
xmlns:ogc="http://www.opengis.net/ogc"
xmlns:gml="http://www.opengis.net/gml/3.2"
xmlns:fes="http://www.opengis.net/fes/2.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.opengis.net/wfs/2.0
                  http://www.pvretano.com/schemas/wfs/2.0.0/wfs.xsd
                  http://www.opengis.net/gml/3.2
                  http://schemas.opengis.net/gml/3.2.1/gml.xsd">
<wfs:Query typeName="aixm:AirportHeliport" handle="Q01">
<fes:Filter>
<fes:PropertyIsEqualTo>
          <fes:ValueReference> timeSlice/AirportHeliportTimeSlice/designator</fes:ValueReference>
          <fes:Literal>KDFW</fes:Literal>
</fes:PropertyIsEqualTo>
</fes:Filter>
</wfs:Query>
</wfs:GetPropertyValue>
```

Response:

```
<wfs:ValueCollection xmlns:wfs="http://www.opengis.net/wfs/2.0" numberReturned="1">
<wfs:member>-97.0379958333333 32.8968280555556</wfs:member>
</wfs:ValueCollection>
```

GetPropertyValue query 2: Return the validTime properties for an AirportHeliport

Request:

```
<?xml version="1.0" ?>
<wfs:GetPropertyValue
service="WFS"
version="2.0.0"
valueReference="//gml:validTime"
outputFormat="application/gml+xml; version=3.2"
xmlns:wfs="http://www.opengis.net/wfs/2.0"
xmlns:aixm="http://www.aixm.aero/schema/5.1"
xmlns:ogc="http://www.opengis.net/ogc"
xmlns:gml="http://www.opengis.net/gml/3.2"
xmlns:fes="http://www.opengis.net/fes/2.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.opengis.net/wfs/2.0
                  http://www.pvretano.com/schemas/wfs/2.0.0/wfs.xsd
                  http://www.opengis.net/gml/3.2
                  http://schemas.opengis.net/gml/3.2.1/gml.xsd">
<wfs:Query typeName="aixm:AirportHeliport" handle="Q01">
<fes:Filter>
<fes:PropertyIsEqualTo>
          <fes:ValueReference> timeSlice/AirportHeliportTimeSlice/designator</fes:ValueReference>
          <fes:Literal>KDFW</fes:Literal>
</fes:PropertyIsEqualTo>
</fes:Filter>
</wfs:Query>
```

```
</wfs:GetPropertyValue>
```

Response:

```
<wfs:ValueCollection xmlns:wfs="http://www.opengis.net/wfs/2.0" numberReturned="1">
<wfs:member>
<gml:validTime xmlns:gml="http://www.opengis.net/gml/3.2">
<gml:TimePeriod gml:id="urn-x:ows7:snowflake:tp:nasr_arp.16189">
<gml:beginPosition>2000-01-01T00:00:00.000Z</gml:beginPosition>
<gml:endPosition>9999-12-31T23:59:59.000Z</gml:endPosition>
</gml:TimePeriod>
</gml:validTime>
 </wfs:member>
</wfs:ValueCollection>
```

Annex IV – WFS-T 2.0 Insert

WFS-T 2.0 insert for Runway TEMPDELTA

```
<?xml version="1.0" encoding="UTF-8"?>
<wfs:Transaction service="WFS" version="2.0.0"
xsi:schemaLocation="http://www.opengis.net/wfs/2.0 wfs/2.0.0/wfs.xsd http://www.aixm.aero/schema/5.1
AIXM_Features.xsd"
xmlns:wfs="http://www.opengis.net/wfs/2.0"
xmlns:aixm="http://www.aixm.aero/schema/5.1"
xmlns:gml="http://www.opengis.net/gml/3.2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<wfs:Insert>
<aixm:Runway gml:id="urn-x:ows7:snowflake:amdb_runway.fid-20bf8b95_12777030502_-70d4.a">
<gml:boundedBy>
<gml:Envelope srsName="urn:ogc:def:crs:OGC:1.3:CRS84">
<gml:lowerCorner>-86.62301239 39.1376874</gml:lowerCorner>
<gml:upperCorner>-86.6110716 39.14287482</gml:upperCorner>
</gml:Envelope>
</gml:boundedBy>
<aixm:timeSlice>
<aixm:RunwayTimeSlice gml:id="urn-x:ows7:snowflake:ts:amdb_runway.fid-20bf8b95_12777030502_-
70d4.a">
<gml:validTime>
<gml:TimePeriod gml:id="urn-x:ows7:snowflake:tp:amdb_runway.fid-20bf8b95_12777030502_-70d4.a">
<gml:beginPosition>2010-05-05T00:00:00.000Z</gml:beginPosition>
<gml:endPosition>2010-06-05T00:00:00.000Z</gml:endPosition>
</gml:TimePeriod>
</gml:validTime>
<aixm:interpretation>TEMPDELTA</aixm:interpretation>
<aixm:sequenceNumber>2</aixm:sequenceNumber>
<aixm:correctionNumber>1</aixm:correctionNumber>
<aixm:designator>06/24</aixm:designator>
<aixm:lengthStrip uom="M">1000</aixm:lengthStrip>
<aixm:widthStrip uom="M">30</aixm:widthStrip>
</aixm:RunwayTimeSlice>
</aixm:timeSlice>
</aixm:Runway>
</wfs:Insert>
</wfs:Transaction>
```

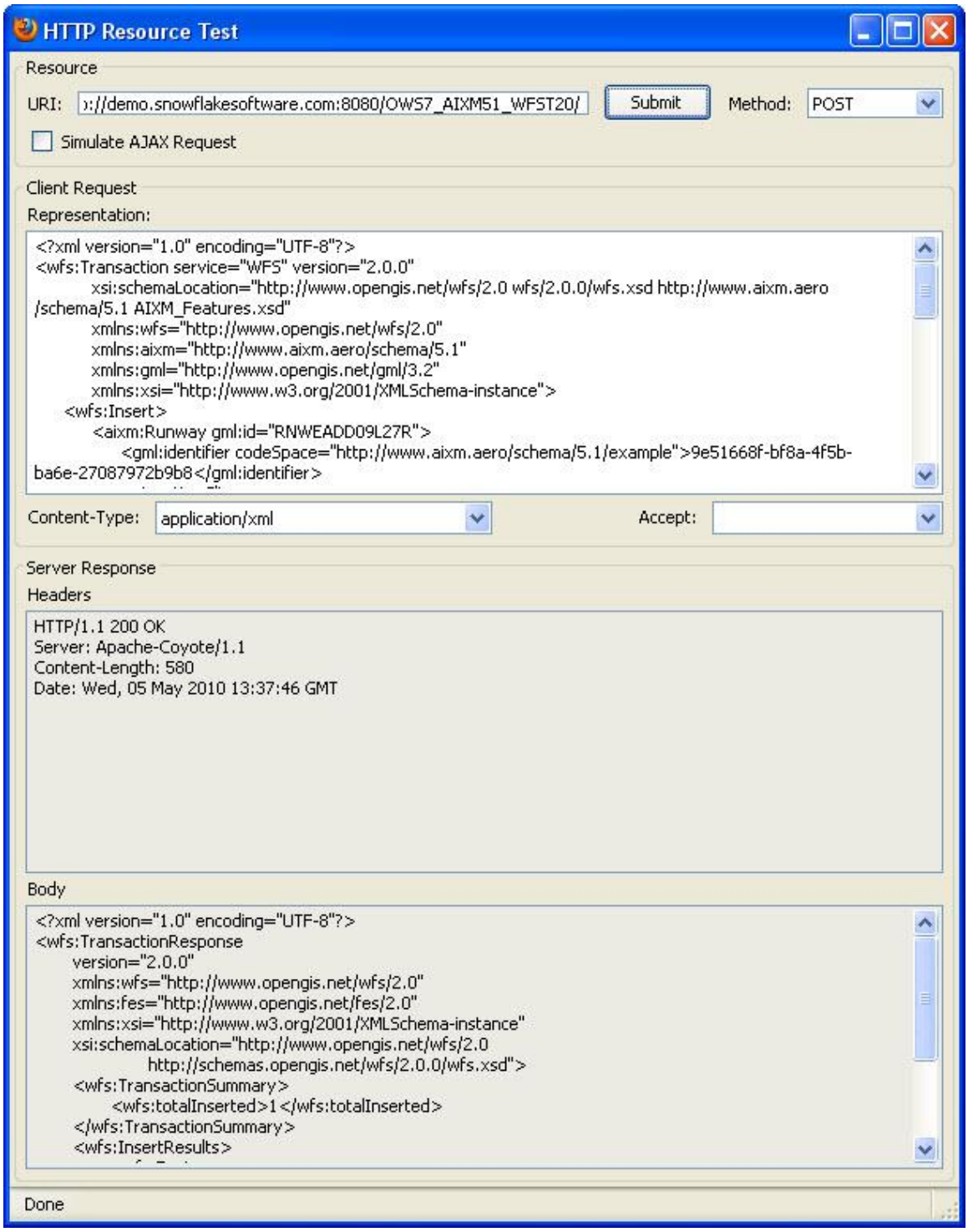


Figure 13. Submitting a Runway TEMPDELTA using GO Publisher WFS-T 2.0 interface

Once the WFS-T has inserted the timeslice into the AIXM 5.1 datastore, the server returns a response to the client to state that the insert completed successfully.

Example WFS-T 2.0 Response to insert of Runway TEMPDELTA timeslice:

HTTP/1.1 200 OK Server: Apache-Coyote/1.1 Content-Length: 632 Date: Wed, 05 May 2010 16:00:25 GMT <?xml version="1.0" encoding="UTF-8"?> <wfs:TransactionResponse version="2.0.0" xmlns:wfs="http://www.opengis.net/wfs/2.0" xmlns:fes="http://www.opengis.net/fes/2.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.opengis.net/wfs/2.0 http://schemas.opengis.net/wfs/2.0.0/wfs.xsd">
<wfs:TransactionSummary>
<wfs:totalInserted>1</wfs:totalInserted>
</wfs:TransactionSummary>
<wfs:InsertResults>
<wfs:Feature>
<fes:ResourceId rid="aixm:Runway.urn-x:ows7:snowflake:amdb_runway.fid-20bf8b95_12777030502_-70d4.a"/>
</wfs:Feature>
</wfs:InsertResults>
</wfs:TransactionResponse>

Once inserted into the database, it triggers the automatic generation of a digital NOTAM for publication to the Event Service. The event service will then push the NOTAM to any subscribed users.

Digital NOTAM triggered by the insertion of the Runway TEMPDELTA timeslice by GO Publisher WFS-T 2.0

<?xml version="1.0" encoding="UTF-8"?><!--Created by GO Publisher Agent 1.5-beta-1-SNAPSHOT Build 19477 from 2010-04-29 19:36--> <!--Snowflake Software Ltd. (http://www.snowflakesoftware.co.uk)--> <!--**THIS DATA SHOULD NOT BE USED FOR NAVIGATION**--> <dnotam:Event xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:dnotam="http://www.aixm.aero/schema/5.1/dnotam" xmlns:gml="http://www.opengis.net/gml/3.2" xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:gmd="http://www.isotc211.org/2005/gmd" xmlns:gco="http://www.isotc211.org/2005/gco" xmlns:gss="http://www.isotc211.org/2005/gss" xmlns:gts="http://www.isotc211.org/2005/gts" xmlns:gsr="http://www.isotc211.org/2005/gsr" xmlns:aixm="http://www.aixm.aero/schema/5.1" xsi:schemaLocation="http://www.aixm.aero/schema/5.1/dnotam Digital_NOTAM_Event.xsd" gml:id="gml_fc_1">
<dnotam:hasMember>
<aixm:Runway gml:id="urn-x:ows7:snowflake:amdb_runway.fid-20bf8b95_12777030502_-70d4.a">
<gml:boundedBy>
<gml:Envelope srsName="urn:ogc:def:crs:OGC:1.3:CRS84">
<gml:lowerCorner>-86.62301239 39.1376874</gml:lowerCorner>
<gml:upperCorner>-86.6110716 39.14287482</gml:upperCorner>
</gml:Envelope>
</gml:boundedBy>
<aixm:timeSlice>
<aixm:RunwayTimeSlice gml:id="urn-x:ows7:snowflake:ts:amdb_runway.fid-20bf8b95_12777030502_-70d4.a">
<gml:validTime>
<gml:TimePeriod gml:id="urn-x:ows7:snowflake:tp:amdb_runway.fid-20bf8b95_12777030502_-70d4.a">

```
 <gml:beginPosition>2010-05-05T00:00:00.000Z</gml:beginPosition>
 <gml:endPosition>2010-06-05T00:00:00.000Z</gml:endPosition>
 </gml:TimePeriod>
 </gml:validTime>
 <aixm:interpretation>TEMPDELTA</aixm:interpretation>
<aixm:sequenceNumber>2</aixm:sequenceNumber>
<aixm:correctionNumber>1</aixm:correctionNumber>
 <aixm:designator>06/24</aixm:designator>
 <aixm:lengthStrip uom="M">1000</aixm:lengthStrip>
 <aixm:widthStrip uom="M">30</aixm:widthStrip>
 </aixm:RunwayTimeSlice>
 </aixm:timeSlice>
 </aixm:Runway>
 </dnotam:hasMember>
 </dnotam:Event>
```

Also any client can also request the Runway TEMPDELTA timeslice via the WFS 2.0

http://demo.snowflakesoftware.com:8080/OWS7_AIXM51_WFST20/GOPublisherWFS?service=WFS&version=2.0.0&request=GetFeature&TYPENAME=aixm:Runway&RESOURCEID=urn-x:ows7:snowflake:amdb_runway.fid-20bf8b95_12777030502_-70d4.a