Open Geospatial Consortium

Publication Date: 2016-01-18

Approval Date: 2015-06-05

Posted Date: 2015-06-17

Reference number of this document: 15-027r1

External identifier of this OGC® document: http://www.opengis.net/doc/PER/t11-notam-validation

Category: Public Engineering Report

Editor: Aleksandar Balaban

OGC® Testbed 11 Digital Notice to Airmen (NOTAM) Validation and Enrichment Service Engineering Report

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

Contents

Page

Figures

Tables

Listings

OGC® Testbed 11 Digital Notice to Airmen (NOTAM) Validation and Enrichment Service Engineering Report

Abstract

This OGC Engineering Report (ER) is a deliverable of the OGC Testbed 11. This ER describes the Digital Notice to Airmen (NOTAM) enrichment and validation services in the Testbed 11 Aviation thread, including:

- A description of the architecture and architectural options.
- An overview of the implemented components and workflows followed by a short description of each component.
- Documentation of the issues, lessons learned as well as accomplishments and scenarios that were of general interest in the Aviation thread.

More detailed information on other specific aspects considered in OWS-11 Aviation may be found in the individual Aviation Engineering Reports.

Business Value

The described service enhances the value of Digital NOTAM content through improvement in data quality as a result of automation in validation and enrichment of the Digital NOTAM content.

Keywords

ogcdoc, ogc documents, testbed, t11, testbed 11, aviation, notam

1 Introduction

1.1 Scope

This OGC® document describes the approach and the architecture implemented in the Testbed 11 Digital NOTAM Validation and Enrichment Service.

- A description of architectural options and implementations for both enrichment and validation services, as well as the integration of these services for a role in Testbed scenario demonstrations.

- An overview of the implemented components and workflows followed by a short description of each component.

- A discussion about possible improvements regarding WFS-T 2.0, AIXM 5.1, as well as static and dynamic aeronautical data related validation rules.

- Documentation of the issues and lessons learned as well as accomplishments.

More detailed information on specific aspects considered in the Testbed 11 Aviation Thread may be found in other Aviation Engineering Reports.

1.2 Document contributor contact points

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

Name	Organization
Aleksandar Balaban	m-click.aero
Thomas Forbes	Snowflake
Volker Grabsch	m-click.aero
Matthias Pohl	m-click.aero
Markus Schneider	m-click.aero

1.3 Summary of Accomplishments

During the conception, design, and implementation activities of this Testbed 11 Thread several technologies, data formats, and COTS products suitable for enrichment and validation service implementations were identified, as follows.

1. Demonstration of a declarative[1] digital NOTAM validation based on Schematron for XSLT2 deployed in a standard Java web container and server hardware.
2. An OGC Web Processing Service (WPS) 1.0 based validation service endpoint was implemented. Specifically, the ability to load data from an external source given as an input parameter was advantageous. However, WPS is only a good match if the input data can be expected to be well-formed XML.
3. Geographic Metadata extensible markup language (GMD) was used for the validation results after some evaluation. The GMD was chosen in order to fulfill

[1] The phrase "declarative" addresses such a computational approach where a set of well-defined rules are specified and used by a rule engine in order to perform document validation without procedural programing.

the required "standard reusability policy" and to avoid special purpose data format creation.

4. Both the enrichment and the validation services were integrated using an aviation client application. While almost every client capable of invoking WFS and WPS services to visualize the aeronautical data could retrieve enriched documents and request the validation, the m-click aviation client with additional validation result visualization capabilities was used for demonstration purposes.
5. An extension of the event specification schema was made in order to allow reverse associations from events to affected aeronautical entities. Originally, events were not aware of static data (because AIXM 5.1 supports only the child-to-parent direction). Therefore an advanced enrichment was demonstrated performing full integration of both NOTAMs and associated aeronautical entities in a local aeronautical database. The navigation through the data sets along the dependencies between events and static data were implemented for purposes of advanced visualization and processing.

1.4 Future work

For future work, the tests and demonstrations should be performed on a more mature set of validation rules from both the Digital NOTAM event specification, as well as general AIXM 5.1 formatting rules. Preferably, the final rule set shall be considered, if available. Furthermore, attention should be given to the question whether the Schematron-based rule engine has needed capabilities to validate the (future) complete set of rules related to the Digital NOTAM documents.

1.5 Foreword

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 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.

OWS Engineering Reports:

1. OGC 11-092, OWS-8 Report on DNES (Digital NOTAM Event Specification) https://portal.opengeospatial.org/files/?artifact_id=46243

2. OGC 12-146, OWS-9 WFS-TE ER, Web Feature Service Temporality Extension Engineering Report, https://portal.opengeospatial.org/files/?artifact_id=51811

3. OGC 15-025, Testbed 11 Aviation Architecture Engineering Report

4. OGC 15-028, Testbed 11 Aviation Guidance Using SBVR Engineering Report

Other OGC Documents:

5. OGC 09-025r1 WFS 2.0, OpenGIS Web Feature Service 2.0 Interface Standard, https://portal.opengeospatial.org/files/?artifact_id=39967

6. OGC 09-026r1, Filter Encoding 2.0 Encoding Standard, https://portal.opengeospatial.org/files/?artifact_id=39968

7. OGC 05-007r7, WPS 1.0 Web Processing Service 1.0 http://portal.opengeospatial.org/files/?artifact_id=24151

8. OGC 12-027r3, WFS-TE Temporality Extension (TE) Discussion Paper http://www.opengeospatial.net/doc/DP/wfs-temporality-extension

9. Geographic MetaData extensible markup language (GMD) http://www.isotc211.org/schemas/2005/gmd/

Aviation Documents:

10. Aeronautical Information Exchange Model (AIXM) - AIXM Application Schema Generation, online at http://www.aixm.aero/public/standard_page/download.html

11. DNES 1.0, Digital NOTAM Event Specification, ed. 1.0 (Proposed Release), online at http://www.aixm.aero/public/standard_page/digital_notam_specifications.html

12. AIXM-TM 1.0 - Temporality Model v1.0, online at (AIXM TM)

Other Documents:

13. ISO/IEC 19757-3:2006 Document Schema Definition Language (DSDL) -- Part 3: Rule-based validation – Schematron, http://standards.iso.org/ittf/PubliclyAvailableStandards/c040833_ISO_IEC_19757-3_2006(E).zip

14. ISO/IEC 11404:2007 Information technology, General-Purpose Datatypes (GPD)

15. OMG Semantics Of Business Vocabulary And Business Rules (SBVR), V1.2, http://www.omg.org/spec/SBVR/1.2/PDF

3 Terms and definitions

For the purposes of this report, the definitions specified in Clause 4 of the OWS Common Implementation Standard [OGC 06-121r3] apply.

4 Conventions

4.1 Abbreviated terms

AIXM	Aeronautical Information Exchange Model
DNOTAM	Digital NOTAM
DNES	Digital NOTAM Event Specification
FES	Filter Encoding Specification
GML	Geography Markup Language
ICAO	International Civil Aviation Organization
NOTAM	Notice To Airmen
OASIS	Organization for the Advancement of Structured Information Standards
OGC	Open Geospatial Consortium
OWS	OGC Web Service
WCS	Web Coverage Service
WFS	Web Feature Service
WFS-T	WFS Transactional
WFS-TE	WFS Temporality Extension
WMS	Web Map Service
WPS	Web Processing Service

XML	Extensible Markup Language

4.2 UML Notation

Unified Modeling Language notations used in this document are based on UML 2.0 component and sequence diagrams. In order to keep things understandable, the diagrams presented in this document are not strictly compatible with corresponding UML notations. Their purpose is to provide an overview of basic building elements included in proposed solutions and to present the most important execution steps involved in both validation and enrichment processes

5 Digital NOTAM Validation and Enrichment Service Overview

The work on the concept of Digital NOTAM (DNOTAM) [DNES 1.0] was begun as part of an overall modernization effort for aeronautical services. International Civil Aviation Organization (ICAO) has noted that "one of the major operational constraints of today is that aeronautical information is usually provided on paper." Also, "the implementation of the DNOTAM will provide the means to create truly temporal electronic databases and form the foundation for the creation and maintenance of the universal operational airspace situation picture available on the ground and in the air." As part of that effort, the concept of DNOTAM aims to provide enhanced global civil aviation safety and efficiency improvements.

This OGC ER presents the prototyping results for two important technical services involved in the Digital NOTAM data provision and quality assurance. The ER starts with an overview of previous and related Testbed work [OWS-8 Report on DNES] and continues by explaining several common concepts regarding provision of aeronautical data and encoding of NOTAMs as defined in the Digital NOTAM Event Specification 1.0 document. The ER continues with a summary of the major tasks for both validation and enrichment depicting logical architecture in the form of architectural views and artifacts modeled in UML. The ER then summarizes the work completed, provides the results and lessons learned, and presents recommendations for future activities.

The evaluation of validation rules from DNES (AIXM 5.1 general encoding rules) was partially done in the OWS-8. As part of the OWS-8 Testbed aviation thread, the concept of digital NOTAM validation rules was analyzed through:

- review of the conceptual aspects of digital NOTAM design and AIXM 5.1 validation rules;
- encoding declarative, common purpose AIXM 5.1 validation rules in the machine-readable Schematron format; and
- providing examples for automatic NOTAM validation based on common formatting validation rules encoded in a machine-readable format.

The output of these validation efforts is documented in the corresponding OWS-8 ER. This ER continues the work on validation considering recommendations given in the future work section of the OWS-8 ER and additionally using several Digital NOTAM specific validation rules which fit to the Testbed 11 scenarios. In keeping with the Testbed scenario, these validation rules are related to runway and airport AIXM entities. The objective of this report is also to explore the augmentation of Digital NOTAM inputs to:

- validate Digital NOTAM applying the business rules stated in the current version of DNES (2014);
- provide full information about quality of Digital NOTAM;
- enrich aeronautical entities (provided as TEMPDELTA) shipped within a Digital NOTAM with corresponding TimeSlices, including BASELINE TimeSlices of related features; and
- help to visualize Digital NOTAM through enrichment.

Currently, only the general AIXM 5.1 validation rules are fully available. The Digital NOTAM-specific rules are under development. Still, several draft Digital NOTAM-specific rules are provided for validation and verification purposes within this Testbed.

6 Motivation and Use Cases

Digital NOTAM quality has been described as inconsistent across the globe. Although considerable work has been done to unify the data, services, and processes in the domain of aviation worldwide, different characterisitcs of regional abbreviations or extensions to the lowest denominator of rules given by the ICAO still remain. Automating the quality assessment and enrichment of Digital NOTAMs in a machine readable form has huge potential to increase aviation safety and to further minimize the risk that the wrong information is available at the wrong time. The common encoding and data validation rules, as well as the awareness regarding possible extensions and the ability to capture and interpret those rules, are essential for the successful implementation of the Digital NOTAM concept of operations.

The OGC Testbed 11 deals with three different demonstration scenarios (see chapter 11 of OGC 15-025 Aviation Architecture ER[2]) which include flights from Europe to the USA depicting two slightly different concepts in the NOTAM encoding (relevant at departure and arrival airports as part of pre-flight information bulleting) as well as the different approaches regarding the availability of static aeronautical data. Using a declarative validation rule approach and enrichment of basic data, the motivation was to

[2] https://portal.opengeospatial.org/files/?artifact_id=63793

demonstrate a possible generic way to overcome the problems caused by data heterogeneity automating the quality assessment and enrichment of Digital NOTAMs in a machine readable form.

6.1 Digital NOTAM Key Principles

Each category of relevant real-life events represented by a NOTAM needs to be identified in order to specify its format, as well as the rules that are specific for that kind of event. A Digital NOTAM Event Specification 1.0 [DNES 1.0] is being developed for this purpose.

The DNES defines how information that is currently published through Digital NOTAM messages shall be encoded in a harmonized way based upon AIXM 5.1. The logical data model of AIXM 5.1 defines features and properties that enable the provision of static and dynamic aeronautical information. Additionally, the current version of DNES includes definition for 23 different event types called scenarios, as well as the rules of their encodings. An event scenario is applied to an AXIM 5.1 feature and describes the Digital NOTAM condition or event that is being reported, as show in Table 1.

Table 1 – Digital NOTAM scenarios

Code	Explanation
SAA.ACT	Published special activity area - activation
ATSA.ACT	Published ATS airspace - activation or deactivation
SAA.NEW	Ad-hoc special activity area - creation
ATSA.NEW	Ad-hoc ATS airspace - creation
RTE.CLS	Route portion closure
RTE.OPN	Route portion opening
NAV.UNS	Navaid unserviceable
OBS.NEW	New obstacle
OBL.UNS	Obstacle lights unserviceable
AD.CLS	Airport/Heliport closure
RWY.CLS	Runway closure
RWE.CLS	Runway portion closure

TWY.CLS	Taxiway Closure
THR.CHG	Displaced Threshold
RWD.CHG	Runway Declared Distance Change
ALS.UNS	Approach Lights Unserviceable
ALS.LIM	Approach lights downgraded
VAS.UNS	Visual approach slope indicator unserviceable
RWL.UNS	Runway Lights Unserviceable
TWL.UNS	Taxiway Lights Unserviceable
SFC.CON	Airport surface contamination
VOLC.WRN	Volcano pre-eruption
OTHER	Other Events

Scenario names are encoded using simple syntax consisting of AIXM 5.1 entity type prefix following by an obligatory event type.

Every Digital NOTAM scenario type is related to a specific aeronautical entity type (encoded as a name prefix). The types are encoded in accordance with AIXM 5.1 document schema and represent typical use cases such as creation, activation, deactivation, closure, and serviceability. For each of the events listed above, the corresponding section of DNES provides detailed list of normative business rules which are automatically tested during the validation process.

In essence, a Digital NOTAM is encoded as an AIXM 5.1 XML document, which contains at least one Event child element (containing TimeSlices in accordance with the AIXM 5.1 schema) and at least one TEMPDELTA TimeSlice belonging to affected feature and pointing to the event element. Events frequently communicate temporal or permanent change in the capability or structure of aeronautical features (including the creation of a new entity), which is why static data TimeSlices have to be associated with the features. The association is directed from child to parent. That means a TEMPDELTA TimeSlice references to digital NOTAM event feature using the XLink mechanism (xlink:href attribute), as shown on the highlighted part of the event document segment given below in Listing 1. An airport TEMPDELTA TimeSlice points to an event and at the same time communicates airport closure (through the operational status property).

```xml
<aixm:timeSlice>
   <aixm:AirportHeliportTimeSlice gml:id="ID_662">
     <gml:validTime>
       <gml:TimePeriod gml:id="ID_663">
         <gml:beginPosition>2014-12-15T16:00:00Z</gml:beginPosition>
         <gml:endPosition>2014-12-15T17:00:00Z</gml:endPosition>
       </gml:TimePeriod>
     </gml:validTime>
     <aixm:interpretation>TEMPDELTA</aixm:interpretation>
     <aixm:sequenceNumber>1</aixm:sequenceNumber>
     <aixm:availability>
       <aixm:AirportHeliportAvailability gml:id="ID_664">
          <aixm:operationalStatus>CLOSED</aixm:operationalStatus>
       </aixm:AirportHeliportAvailability>
     </aixm:availability>
     <aixm:extension>
       <event:AirportHeliportExtension gml:id="ID_667">
         <event:theEvent xlink:href="#uuid.f50faabn" xsi:nil="true"/>
       </event:AirportHeliportExtension>
     </aixm:extension>
   </aixm:AirportHeliportTimeSlice>
 </aixm:timeSlice>
```

Listing 1 - An airport event TimeSlice points to an event

As stated in DNES, a prerequisite for any Digital NOTAM application is the availability of the static data in the form of AIXM 5.1 BASELINE TimeSlices. Here we must distinguish between AIXM 5.1 temporal concepts such as the BASELINE, PERMDELTA and TEMPDELTA. The BASELINE TimeSlice is required both for the originator and for NOTAM receivers. The static data availability might be achieved using different approaches, but optimal method is provision via an OGC WFS 2.0 service [WFS 2.0], which an enrichment service would invoke to fetch missing TimeSlices.

For the scope of this ER (enrichment and validation) it is important to summarize that:

- every event has a scenario to which it belongs. Valid Digital NOTAM always contain AIXM 5.1 TEMPDELTA TimeSlices because the NOTAMs always communicate temporalities[3];

- as part of Digital NOTAM scenarios, there are validation rules encoded in natural language and in the OMG Semantics of Business Vocabulary and Business Rules

[3] An event could also trigger permanent change or the commissioning of a new feature. In the DNES document it mentions the use of digital NOTAM for the exchange of ad-hoc special activity airspace (section 4.4). This use case describes the creation of an SAA for which a PERMDELTA and a BASELINE timeslice would be transmitted along with the Event feature. It is a permanent change in the sense that it is the creation of a new feature (a change from nothing to something).

(SVBR) format: the Schematron is the final machine-readable format of choice for automatic validation;

- AIXM entity TimeSlice information shipped within an event message is sometimes not sufficient to fully validate, visualize, and process Digital NOTAM messages because such content does not contain full information about affected aeronautical entities;
- for validation, we need to retrieve the BASELINE data and merge them with available TEMPDELTA (this is called Digital NOTAM enrichment); and
- Digital NOTAM enrichment is important for visualization as not all AIXM 5.1 feature types contain geometries.

6.2 Enrichment

In some cases, the normative statements of the Digital NOTAM Event Specification presume the availability of information which is not part of a Digital NOTAM (for example, checking against designators in BASELINE data). To process such rules, a validation tool must provide a method for retrieving such information from a referent static data repository. The outcome of the OWS-8 task regarding Digital NOTAM was that some of the observed issues presumably also apply for enrichment regarding automatic validation.

In the overall context of Digital NOTAM validation, the enrichment service is responsible for the retrieval of BASELINE TimeSlices that correspond to a TEMPDELTA TimeSlice, in an AIXM 5.1 event entity. Alternatively, a SNAPSHOT TimeSlice might also be retrieved as a carrier for natural key information when the use of artificial keys (UUIDs) is not possible. To enrich a Digital NOTAM with related features that contain geometries is a key requirement of this service due to the fact that AIXM 5.1 static data BASELINE will not usually be transmitted as part of event message due to data consistency problems and performance reasons. This sort of behavior is also stated in the DNES.

The availability of such BASELINE static data is crucial for enrichment service implementation in order to establish a “fully operational picture regarding entities affected by certain digital NOTAM.”

6.3 Validation

The process of validation encloses syntactic and semantic validation based on a predefined set of normative business rules from the DNES, as follows.

- Receiving a validation request using appropriate service end point and input message format.

- XML schema validation check (inclusive of ad-hoc XML syntactic validation which is not the subject of this document but has been mentioned here for sake of completeness).
- Enforcement of normative rules encoded in ISO Schematron for XSLT.
- Encoding and delivering the validation result to the requestor using appropriate data format.

The Schematron based validation approach relies on technologies like XPath 2.0 and XSLT.

The following table provides several examples for rules' assertions encoded in both natural language and machine friendly Schematron representation (listed for explanatory purposes, the real assertions might differ).

Table 2 - Rule Examples

AIXM 5.1 Rule	Description	Rule Definition (Schematron)	Status
1A33C6	Each Runway shall have assigned associated AirportHeliport value	aixm:RunwayTimeSlice/(not(./aixm:interpretation='BASELINE' or ./aixm:interpretation='SNAPSHOT') or ((./*:associatedAirportHeliport) and not(./*:associatedAirportHeliport[@xsi:nil='true']))))) else ((./*:associatedAirportHeliport) and not(./*:associatedAirportHeliport[@xsi:nil='true']))"	Error
1A33C7	Each Runway shall have assigned surfaceProperties value	aixm:RunwayTimeSlice/(not(./aixm:interpretation='BASELINE' or ./aixm:interpretation='SNAPSHOT') or ((./*:surfaceProperties) and not(./*:surfaceProperties[@xsi:nil='true']))))) else ((./*:surfaceProperties) and not(./*:surfaceProperties[@xsi:nil='true']))"	Error
1A33D3	Each Runway shall have assigned associated AirportHeliport value	aixm:RunwayTimeSlice/(not(./aixm:interpretation='BASELINE' or ./aixm:interpretation='SNAPSHOT') or ((./*:associatedAirportHeliport) and not(./*:associatedAirportHeliport[@xsi:nil='true']))))) else ((./*:associatedAirportHeliport) and not(./*:associatedAirportHeliport[@xsi:nil='true']))"	Error

The OWS-8 activity suggested splitting up the overall process of validation into two parts: (1) the XML schema validation and (2) semantic based evaluation of normative business rules using Schematron. The process of validation must also consider additional data provided by enrichment. An annotation (augmentation) of original NOTAM documents with validation results is also recommended.

6.4 Use Cases

The use cases considered in this ER are the three defined in the Testbed-11 scenarios. The digital NOTAM validation and enrichment requirements suggest they also describe the airport and runway entities (runway directions inclusive) affected by certain irregularities (flooding), which lead to changes in their operational state (airport or runway closure). Thus the departure and arrival airports are located in the regions with slightly different digital NOTAM validation rules (Europe, USA), such that both rule sets must be considered to provide a flexible, declarative solution.

7 Solution Architecture

7.1 Overview

For both validation and two enrichment services, several architectural options were considered. One choice is between synchronous or asynchronous processing (batch). Enrichment results could be stored in a repository (for future consumption) or forwarded directly to a requestor. The validation service might further provide results as a dedicated response document or insert them directly into the input NOTAM document. The following tables provide a summary regarding possible ways to implement and integrate the services.

Message exchange pattern, interface definitions:

Table 3 - Service Interface Options

ID	Option Description	Pros./Cons.
IOP-01	Results of enrichment and validation are actually the input documents additionally containing the BASELINE TimeSlices of static data in case of enrichment or the list of validation results with assertions in case of validation.	NOTAM documents become extended for validation meta-data and additional aeronautical entities. Low implementation effort and good integration into existing workflows.
IOP-02	Both enrichment and validation services implemented using standard OGC interface specifications (WFS, WPS).	High grade of interoperability through the reusability.
IOP-03	Dedicated HTTP/REST or HTTP/SOAP service endpoints might be designed for enrichment and validation purposes.	Optimized solution, but requiring additional effort and resulting in lower interoperability.

We implemented services utilizing the IOP-01 and IOP-02 options.

Components and integration into workflows:

Table 4 - Other architectural options

Option ID	Option Description	Pros./Cons.
AOP-01	Human actor invokes validation service using HMI controls. Validation service always loads NOTAMs provided by the enrichment service and performs the validation.	Both services use Request/Response MEP. Enrichment can still be used autonomously for visualization purposes or as preprocessing step for validation.
AOP-02	Human actor initiates manual enrichment using HMI control and receives enriched NOTAM. The same is true for validation. If validation needs enriched NOTAM, the human actor will invoke the validation service with enriched NOTAM as input parameter.	Both services use Request/Response MEP. Zero automation.
AOP-03	Human actor performs manual enrichment using HMI controls and stores enrichment in an internal repository. In the next step, the validation service will be invoked with a link to the enriched document as an input parameter.	In this case the human actor has the whole control and might decide about process steps. On the other side it is unlikely that such a low grade of automation make sense in production systems.
AOP-04	Every NOTAM imported in a local system is automatically enriched. The validation occurs manually on-demand and via a HMI command. Validation service uses a WFS endpoint provided by enrichment service to retrieve event documents for validation. Event document contain all static data.	NOTAM and aeronautical entities are threated separately in a database. Enrichment was performed automatically and human actor can decide to perform the validation.
AOP-05	Enrichment is performed automatically. Events and linked entities are stored in an association aware WFS database. Both client and validation service can collect needed data from the database (using WFS) and perform the validation. Validation result can be stored back in the database.	Might provide the best solution for a production system. With enhanced enrichment and integration of NOTAMs into local aeronautical data repository (with reverse associations) the WFS queries would be capable enough to retrieve both NOTAMs and associated aeronautical entities.

For Testbed 11, we demonstrated two cases. First, we implemented the process based on two major steps (enrichment and validation) and integration via an aviation client (AOP-04). Enriched NOTAM documents are stored in an internal enrichment's repository and used by the validation service (or by an aviation client). The retrieval is performed via the enrichment's WFS endpoint. The validation results will be delivered embedded in the enriched NOTAM document.

Additionally, an implementation of the AOP-05 was made using event schema specification extended for reverse associations. Enrichment was performed using tight integration with a local aeronautical database. (see Figure 13 – An aviation client depicts runway closure using reverse event associations).

The formal architectural description is provided using four architectural views containing artifacts encoded in UML 2.0:

1. Use Case View
2. Service View
3. Process View
4. Component View

7.2 DNOTAM Validation

The process of validation includes the use of machine-readable encoded rules enforced against a given input data.

7.2.1 Use Case View

Three stakeholders were identified:

- Data Originator
- NOTAM Office
- End User

In a NOTAM distribution scenario, the validation is likely to be performed at End User or at NOTAM Office. The office might use the results of validation to reject or accept a NOTAM proposal made by Data Originator.

7.2.2 Process View

Validation is the checking of input data against available validation rules. In order to perform that check, the following process execution steps occur:

1. fetch event message using a WFS instance;
2. perform XML schema validation;

3. fetch affected aeronautical entities using a WFS instance;
4. invoke business rule engine (Schematron, XSLT) and perform business rule validation;
5. annotate input event message with validation results;
6. (Optional: store the result in the local repository using WFS-T); and
7. return results of validation to the service requestor.

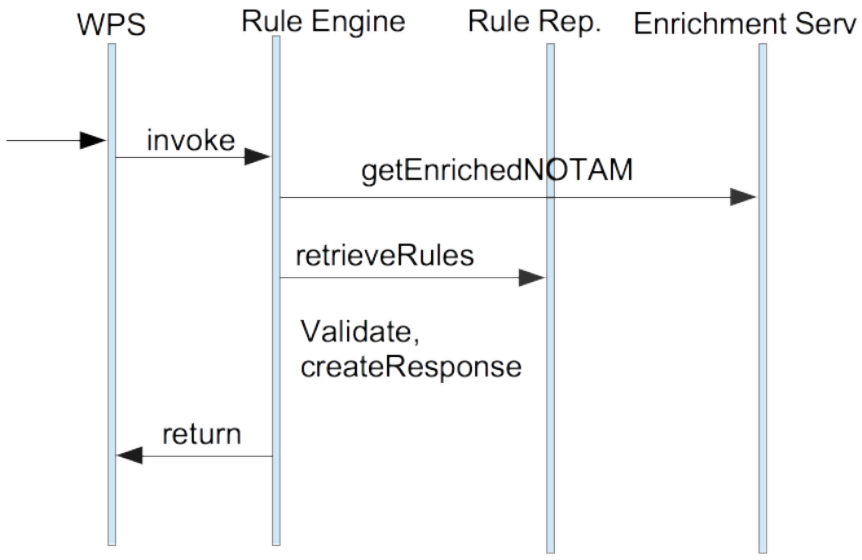


Figure 1 - Validation Activity Diagram

An external service (enrichment service) can optionally be invoked in order to fetch additional static aeronautical feature data needed for validation. The option we chose for Testbed 11 was to assume the enrichment was already done and that enriched NOTAMS are available via an enrichment service (WFS) endpoint.

7.2.3 Service View

A validation service can be implemented considering two standardized interfaces or some dedicated solution. Consideration was made to utilize either an OGC WFS 2.0 or an OGC WPS 1.0 interface [WPS 1.0]. Since the WFS 2.0 standard defines a common purpose CRUD (create, read, update, delete) interface for geographical data, it would be semantically confusing to use it for validation purposes (although it might be considered if the validation has to be performed in the background upon the arrival of new NOTAMs). Therefore, the OGC Web Processing Service (WPS 1.0) seems to be the best implementation choice.

Note on SOAP: The reason not to consider solutions based on the SOAP is complexity. Such relatively heavy service interfaces could be justified only in case of complex document processing chain when message-based security or other special functions such as the reliable messaging are required.

7.2.4 Component View

Following components are identified for Digital NOTAM validation.

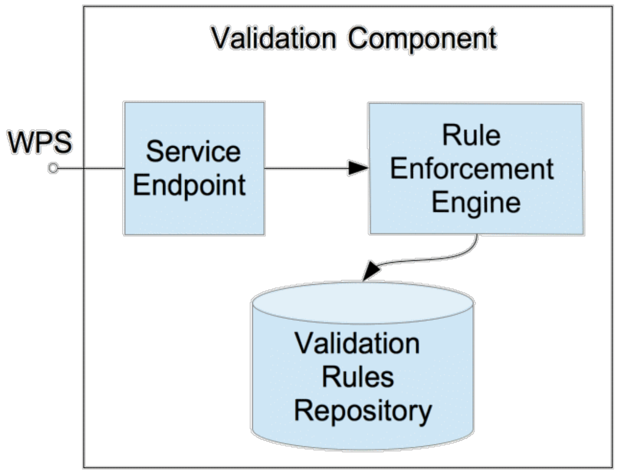


Figure 2 – Validation Component Diagram

Service Endpoint implements access to the service logic providing well-defined syntax and semantic for service requestors. It was implemented as a WPS 1.0 compatible service.

Validation Rule Repository is where the DNOTAM business rules are stored and managed. It contains the formal machine-readable sets of rules in accordance with DNES requirements. Concrete implementation used for scenario demonstrations was based on the ISO Schematron using XSLT. For this implementation, the rule repository was actually a large XML file containing XSLT assertions generated from original validation rules during the preparation phase (validating XSLT stylesheet).

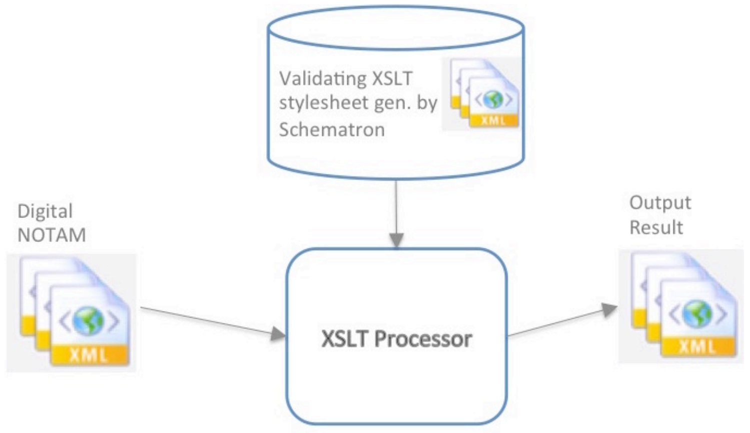


Figure 3 – Validation via Schematron for XSLT

Rule Enforcement Engine is the component responsible for execution of business rules. The implementation based on Schematron for XSLT uses a common purpose XSLT processor for rule enforcement (validation). The XSLT assertions used by the processor to evaluate NOTAM documents are created during the preparation (phase 1) and stored in the validation XSLT stylesheet document (rule repository). The process of validation is depicted as phase 2 on Figure 4 below.

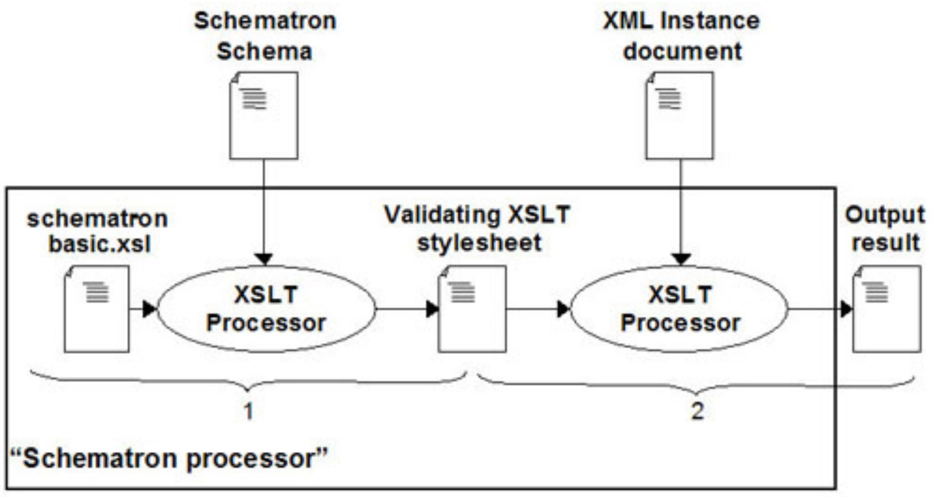


Figure 4 – Schematron based preparation and validation process

7.3 DNOTAM Enrichment

Two implementations of the Digital NOTAM enrichment service prototype were developed for Testbed 11. Both implementations follow steps as outlined in section 7.3.1.2. However differing executions workflows of these steps were undertaken; the two implementations are described below.

7.3.1 m-click

7.3.1.1 Use Case View

Three stakeholders were identified:

- Data Originator
- NOTAM Office
- End User

While enrichment might theoretically be the responsibility of both the data originator and the data user, we assume the enrichment is initiated by the data user. The enrichment at the data originator level would not be in accordance with the DNES specification which states that only AIXM 5.1 feature property changes will be transmitted as part of Digital NOTAM event while the baseline shall remain available (for retrieval). In other words, a NOTAM office will publish Digital NOTAMs and distribute them to subscribers and at the same time will maintain a service providing the access to static BASELINE data. We assume access via an OGC WFS 2.0 compatible Business to Business interface.

7.3.1.2 Process View

Enrichment is a service built around processing logic that fetches the related AIXM 5.1 BASELINE TimeSlices from a static data source, integrates/merges them with an original Digital NOTAM event, and stores the digital NOTAM document in a data repository making it available for provision, visualization, or further processing.

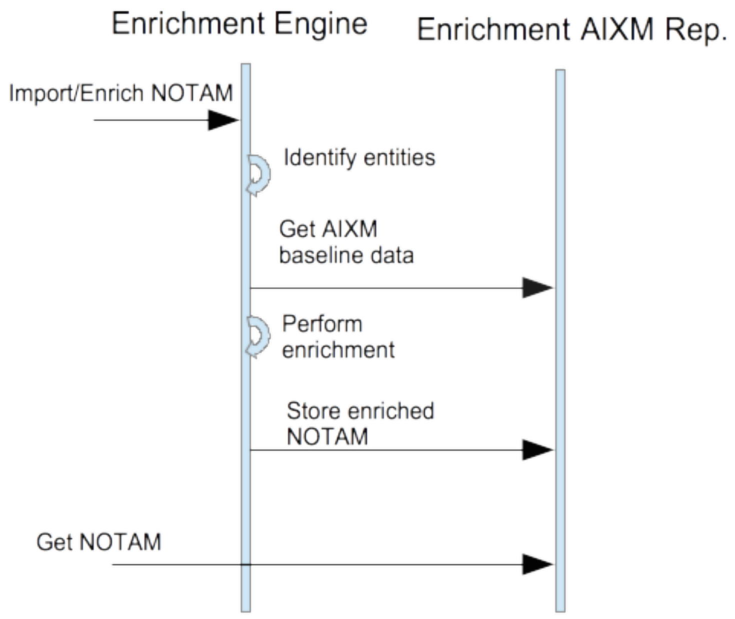


Figure 5 - Enrichment Sequence Diagram

The precondition for successful enrichment is a syntactically valid original Digital NOTAM event message and the availability of associated AIXM 5.1 baseline data. XML Schema validation might be executed as a first step in the process of enrichment if syntactic conformance has not already been ensured. The process of enrichment is triggered by insertion or loading of a new Digital NOTAM event into the enrichment service data store; this insertion or loading process could be undertaken using numerous methods, including a WFS-T interface or COTS software. Once the enrichment has been internally executed, enriched event documents will be internally stored and provided via a WFS for visualization or as input for the validation service.

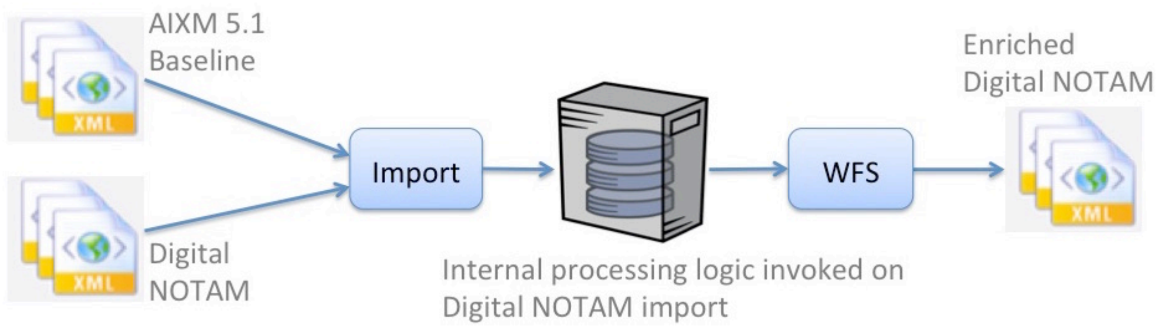


Figure 6 – Enrichment workflow

The following steps depict the general process of Digital NOTAM enrichment. The execution starts by the import of new Digital NOTAM event document into the enrichment service (intern database).

1. Scan Digital NOTAM event document and find AIXM 5.1 feature TEMPDELTA(s), which are linked with the event. Identify affected feature.
2. Perform queries in order to localize the entity's BASELINE TimeSlice.
3. Link/Merge Digital NOTAM document from Step 1 with BASELINE TimeSlices from Step 2.
4. Store the enriched Digital NOTAM in a local repository and make it available via a WFS 2.0 endpoint.

The enrichment is performed as transactional insertion of event documents into an aeronautical database (WFS-T). This is possible due to specific characteristics of the database used for event persistence. The database schemas, which define the structure of aeronautical entities and relations, are derived from a version of the standard AIXM 5.1 schema called MXIA, which allows reverse (bidirectional) associations between entities. Further, an extension to NOTAM event schema originally constructed for this Testbed allows events to reverse reference entities. The database is both static data and event entity aware and can therefore manage any kind of aeronautical data.

AIXM 5.1 baseline static data is pre-loaded to the database where additional associations between aeronautical entities have already been established. A Digital NOTAM document contains only an event feature and a TEMPDELTA of the affected entity, which is usually not contained in the database [Listing 1]. Using that TEMPDELTA, the entity's BASELINE can be identified. With transactional WFS-T calls, a Digital NOTAM can be stored in the database. The transaction includes storing of an event document with additional links to event related entities, as well as persisting of entity's TEMPDELTA. Upon successful transactional insertion, the database will contain new, updated event document, TEMPDELTA and BASELINE TimeSlices, as well as all associations needed to retrieve any kind of event or entity related information using a standard WFS endpoint. A WFS query for the NOTAM stored in the database can then retrieve an enriched NOTAM document.

The diagram below depicts the solution based on the concept of reversal associations. The black lines provide the links as originally specified in the AIXM 5.1 and in the Digital NOTAM extension schema. The event extension (as well as the MXIA) has been exclusively made for enrichment purpose and is depicted through blue colored association. With all the associations in place, a digital NOTAM retrieval becomes a reasonably simple task and the WFS queries performed on such structured data always return an enriched NOTAM document.

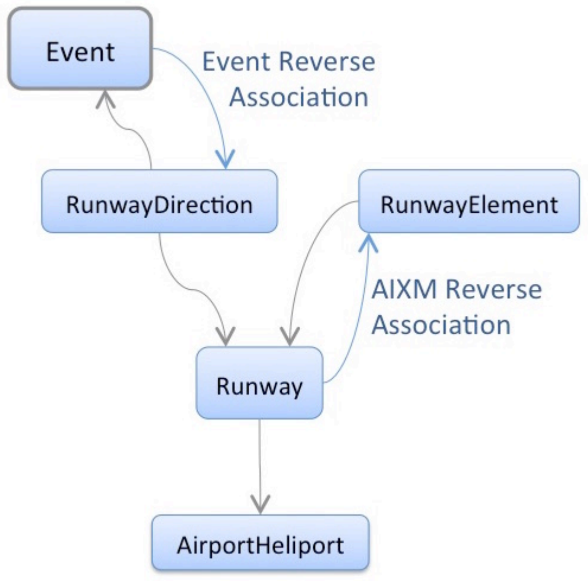


Figure 7 - Reverse associations between events and entities

The figure depicts a special case where an event communicates changes related to a runway direction entity. Additional entities allow enrichment provision, better data handling, and visualization.

7.3.1.3 Service View

The enrichment service is implemented as an asynchronous service using two OGC WFS-T 2.0 service endpoints. Digital NOTAM documents containing TEMPDELTA TimeSlices are stored in a database via a single WFS-T update request. That update performs actual enrichment. The update transaction persists the event document and associated entity's TimeSlices (BASELINE, PERMDELTA). Finally, an enriched event document becomes available via the second enrichment's WFS 2.0 endpoint.

7.3.1.4 Component View

The following logical components are identified for Digital NOTAM enrichment and could be identified in both enrichment service versions:

1. Import Endpoint (WFS-T) or COTS Software
2. Query Endpoint (WFS)
3. Enrichment Repository
4. Enrichment Engine

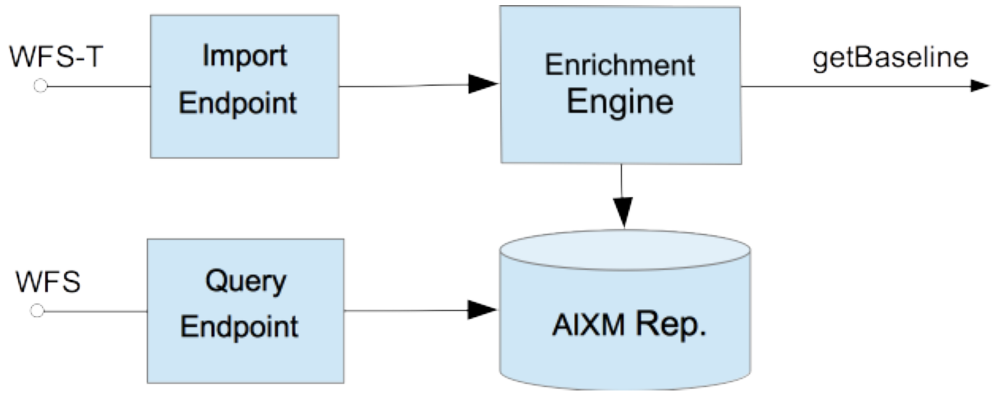


Figure 8 - Enrichment Service Component Diagram

The enrichment engine is a unique component, which contains the business logic for static data retrieval (BASELINE TimeSlices) and Digital NOTAM document structure manipulation (enrichment).

This enrichment service implementation includes the following solution stack for implementation:

1. deegree - an OGC compliant data store (enrichment data store)
2. WFS and WFS-T service endpoints provided by degree

3. Enrichment preparation and processing logic (enrichment engine)

7.3.2 Snowflake Software

7.3.2.1 Use Case View

See Section 7.3.1.1 for use case view description.

7.3.2.2 Process View

The enrichment is a service built around processing logic, which fetches the related AIXM BASELINE TimeSlices from an authoritative data source, merges them with the original digital NOTAM document, and finally stores enriched Digital NOTAM documents in a local repository making it available for provision, visualization or further processing. See- Enrichment Sequence Diagram (Figure 5).

A high-level overview of this implementation is provided in OGC 15-025 *Testbed 11 Aviation Architecture Engineering Report*. A detailed overview of the implementation is provided below.

AIXM baseline data is pre-loaded into a database using Commercial Off The Shelf (COTS) software. A Digital NOTAM document containing only the Event feature (BASELINE and PERMDELTA TimeSlices) and the affected AIXM feature (TEMPDELTA) is loaded into the database using COTS software. During the loading of the Digital NOTAM, the database triggers a series of internal functions and logic that retrieve the relevant AIXM baseline data to enrich the Digital NOTAM.

The loading of the TEMPDELTA TimeSlice initially triggers retrieval of the associated BASELINE TimeSlice. Once this BASELINE is retrieved, BASELINE TimeSlices of associated features are also retrieved. For example, a runway closure as observed within the Testbed 11 scenario retrieves the aixm:RunwayDirection BASELINE TimeSlices that are associated with the TEMPDELTA found within the digital NOTAM, as well as associated BASELINE TimeSlices for aixm:Runway, aixm:RunwayElement, and aixm:AirportHeliport features.

Where associations do not exist between features, internal logic must be used to retrieve the non-referenced feature BASELINE TimeSlices. For example, aixm:RunwayDirection references aixm:Runway, and aixm:Runway references aixm:AirportHeliport. However, from these features there are no references to the aixm:RunwayElement which contain the geometries required to visualize a runway closure. In this instance, Snowflake Software's implementation of the Digital NOTAM enrichment service contains logic that uses the scenario element of the Event feature to check that all relevant features are included in the enriched message. A 'RWY.CLS' scenario requires related aixm:RunwayElements to be included in the enriched Digital NOTAM, therefore the associated aixm:RunwayElement BASELINE TimeSlices are retrieved (this is based on the aixm:Runway BASELINE retrieved previously). As discussed below, the reverse associations AIXM extension would enable the internal logic and checking of Event scenarios to be simplified (see **Error! Reference source not found.**).

Following the retrieval of all relevant AIXM BASELINE TimeSlices for the affected feature and related features, the enriched digital NOTAM is available to end-users via a WFS 2.0 endpoint.

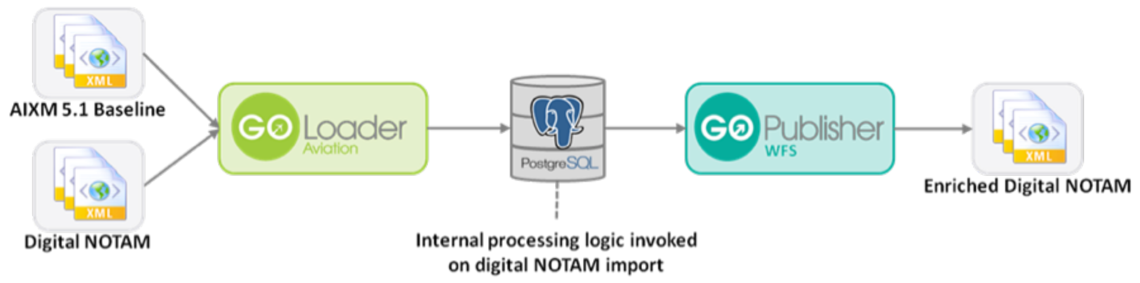


Figure 9 – Enrichment workflow

7.3.2.3 Service View

The enrichment service is implemented as an asynchronous service. Data loading, enrichment, and consumption are separated functions. While the data loading occurs using a COTS based solution, an OGC WFS 2.0 based endpoint is used to query enriched events. Event documents are stored in an internal database using a COTS load process. This activity triggers internal processing logic, which executes enrichment. During the enrichment, a digital NOTAM document is analyzed and BASELINE TimeSlice retrieved and merged with the original digital NOTAM document. For a detailed description please read the process view section.

7.3.2.4 Component View

Following logical components are identified for Digital NOTAM enrichment and could be identified in both enrichment service versions (see also **Error! Reference source not found.**).

5. Import Endpoint (WFS-T) or COTS Software
6. Query Endpoint (WFS)
7. Enrichment Repository
8. Enrichment Engine

The enrichment engine is a unique component that contains the business logic for static data retrieval (BASELINE TimeSlices) and DNOTAM document structure manipulation (enrichment).

Snowflake Software's implementation of the Digital NOTAM enrichment service includes the following logical components.

1. COTS Software (GO Loader Aviation) for data import
2. COTS Software (GO Publisher WFS) for query endpoint (WFS)
3. Enrichment Data Store
4. Enrichment Processing Logic

COTS software is used for loading both the AIXM baseline and the Digital NOTAM data to the enrichment data store. On loading the enrichment processing logic is integrated within the data store to begin the enrichment process. Enriched Digital NOTAM are then made available via a WFS endpoint created using COTS software.

8 Validation and Enrichment Accomplishments

The Testbed 11 aviation thread successfully demonstrated Digital NOTAM enrichment and validation using standard OGC Web Services. Key accomplishments concerning the Digital NOTAM are provided in the following sections.

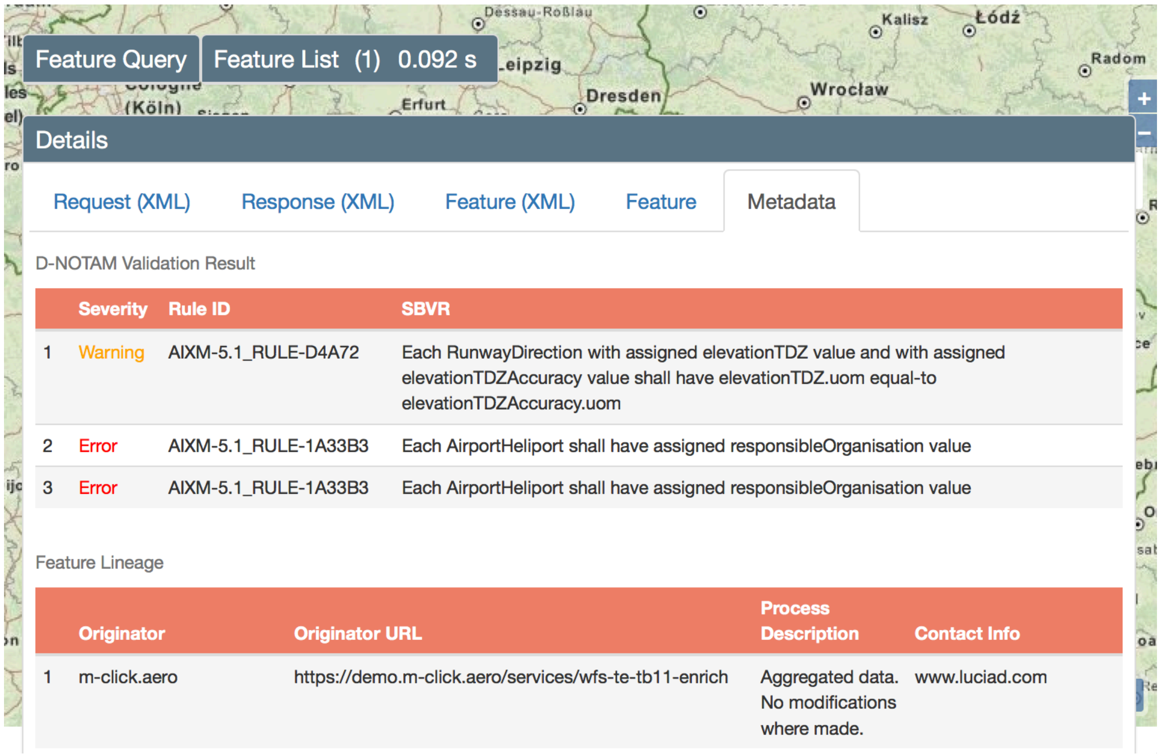
8.1 Validation service implementation using standard solution stack

A Digital NOTAM validation service based on the Schematron for XSLT2 was successfully developed and deployed using a solution stack based on a Java web container, standard database, and affordable server hardware.

The validation component uses the following standardized data types and service endpoints.

- OGC WPS 1.0
- AIXM 5.1
- Digital NOTAM 2.0 Event Specification
- Schematron rule-based validation (ISO/IEC 19757)
- ISO Geographic MetaData extensible markup language (GMD)

The public service endpoint implementation is based on OGC WPS 1.0 and AIXM 5.1, including the Digital NOTAM Event Specification 2.0. The response document contains meta-data (ISO GMD) elements used to encode final validation results. A typical result consists of content as depicted on Figure 10 below.



D-NOTAM Validation Result

	Severity	Rule ID	SBVR
1	Warning	AIXM-5.1_RULE-D4A72	Each RunwayDirection with assigned elevationTDZ value and with assigned elevationTDZAccuracy value shall have elevationTDZ.uom equal-to elevationTDZAccuracy.uom
2	Error	AIXM-5.1_RULE-1A33B3	Each AirportHeliport shall have assigned responsibleOrganisation value
3	Error	AIXM-5.1_RULE-1A33B3	Each AirportHeliport shall have assigned responsibleOrganisation value

Feature Lineage

	Originator	Originator URL	Process Description	Contact Info
1	m-click.aero	https://demo.m-click.aero/services/wfs-te-tb11-enrich	Aggregated data. No modifications where made.	www.luciad.com

Figure 10 - Validation results rendered in an aviation client

The following solution stack and COTS products were used for this implementation.

- Saxon XSLT Processor (rule engine)
- ISO Schematron for XSLT2
- deegree - an OGC compliant data store
- Apache Tomcat Web Container
- Nginx HTTP Server
- PostgreSQL & PostGIS

8.2 Enrichment service implementation using standard solution stack

The enrichment service was successfully deployed on a platform based on a Java web container, standard database, and affordable server hardware.

The enrichment component uses the following standardized data types and service endpoints.

- AIXM 5.1

- Digital NOTAM 2.0 Event Specification

The public service endpoint implementation is based on OGC WFS 2.0 and AIXM 5.1, including the Digital NOTAM Event Specification 2.0. The following solution stack and COTS products were used for the Testbed 11 implementation.

- deegree - an OGC compliant data store
- Apache Tomcat Web Container
- Nginx HTTP Server
- GO Loader (Snowflake Software COTS)
- GO Publisher (Snowflake Software COTS)
- PostgreSQL & PostGIS

8.3 Automatic validation and result provision

During the Testbed 11 activity, several NOTAM specific validation rules related to runways and airports were selected based on how well they fit the demonstration scenarios. In the exercises, the participants demonstrated enrichment and validation, retrieving additional AIXM BASELINE elements for enrichment, and running the validation engine on the enriched data. Some of the rules were available as SVBR and some rules were encoded using natural language. The rules were provided by both Eurocontrol and the FAA. The rules were analyzed and converted to machine friendly Schematron form (which was the part of an another, dedicated Testbed 11 task) and loaded into the validation service.

8.4 Fetching aeronautical data using enrichment service

For this Testbed, the enrichment service was designed and implemented as an WFS 2.0 service providing on-demand enriched documents. The actual enrichment process of importing, querying, and document merging logic was performed internally during the preparation phase. In the future the enrichment process may use additional logic to provide spatial coverage calculations in cases in which the spatial coverage is not provided by the NOTAM originator.

The WFS service augmented with the AIXM specific temporality extension (WFS-TE) might be very useful for enrichment logic in order to select only those entity time slices which are relevant for the enrichment. In our case, that would be the BASELINE instead of the full set of TimeSlices independent on their type, which is the result retrieved using a standard WFS 2.0 endpoint. Alternatively, filtering of entity BASELINE TimeSlices could be integrated into the internal processing logic of the enrichment service as demonstrated in Snowflake Software's implementation.

8.5 WPS 1.0 as an interface of choice for validation purposes

For this activity, the OGC WPS 1.0 service interface was used due to its flexibility as well as its fit for the effort. The purpose of the WPS was strictly for performing validation tasks (WFS was used for data retrieval). We followed the principle of avoiding definition of dedicated service endpoints for validation purpose and reuse of available standards. The ability of validation service to fetch input documents based on a given URL is further very useful characteristic provided in the OGC WPS 1.0 service standard.

The reason not to specify or use some service definition based on SOAP is one of complexity. For validation we do not use message-based security or any other kind of complex processing chain. Therefore, there was no need for additional SOAP features, which would justify the use of SOAP over the simple and reliable HTTP GET/POST service endpoint provided by WPS 1.0. Further WPS is an OGC standard. By using a WPS instance we contribute to OGC based interoperability frameworks.

Another option considered for service interface endpoint implementation was based on the WFS 2.0 specification. The initial idea was to use a WFS-T end-point to import a NOTAM into a local repository implicitly triggering validation in a background as part of WFS-T update execution. Validation results would be embedded into event document and made available via another WFS end-point. Unfortunately, the OGC WFS 2.0 interface has strict specified purpose, which is neither data processing nor data validation. This idea was rejected due to several reasons: semantic ambiguity, implementation complexity, and the required service execution logic, which couldn't be implemented as part of standard WFS 2.0 service endpoint implementation.

8.6 Validation result encoding

For validation result encoding the decision was made to use the AIXM 5.1 extensibility mechanism and embed validation results using ISO Geographic Metadata XML (GMD). The GMD is an XML schema implementation derived from ISO 19115. Although there is certain ambiguity in the interpretation of the implementation and the meta-model is much more complex and far beyond our needs, several significant schema elements were identified and utilized to encode validation results.

8.7 Validation and enrichment as part of the scenario workflow

The following assumptions were made in order to implement and perform the enrichment and validation. We assumed the previously enriched Digital NOTAM documents are used as input for the validation service. From the overall process perspective this means that the enrichment is performed in the first step and the results are used by the validation service. Indeed, already enriched documents were made available via a WFS interface provided by the enrichment service. The aviation client was acting as an integration component retrieving enriched events in the first step and using them as input parameter for validation.

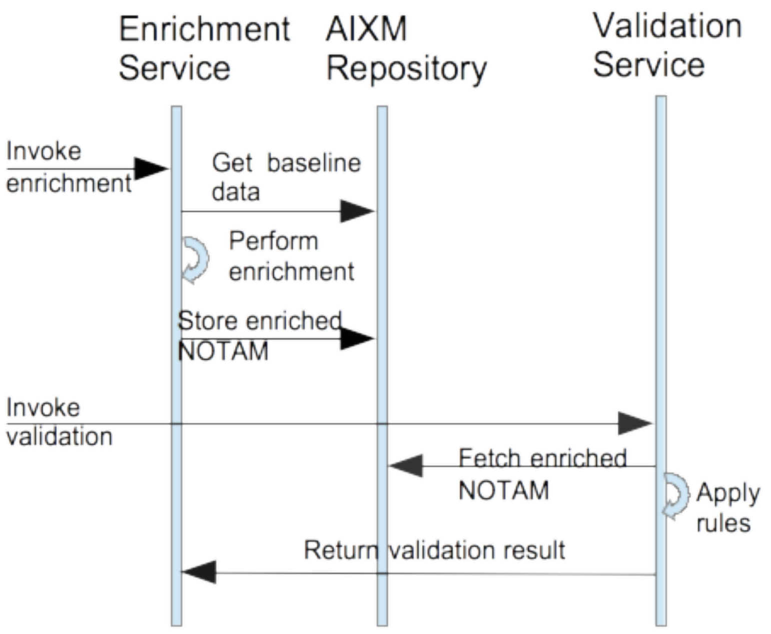
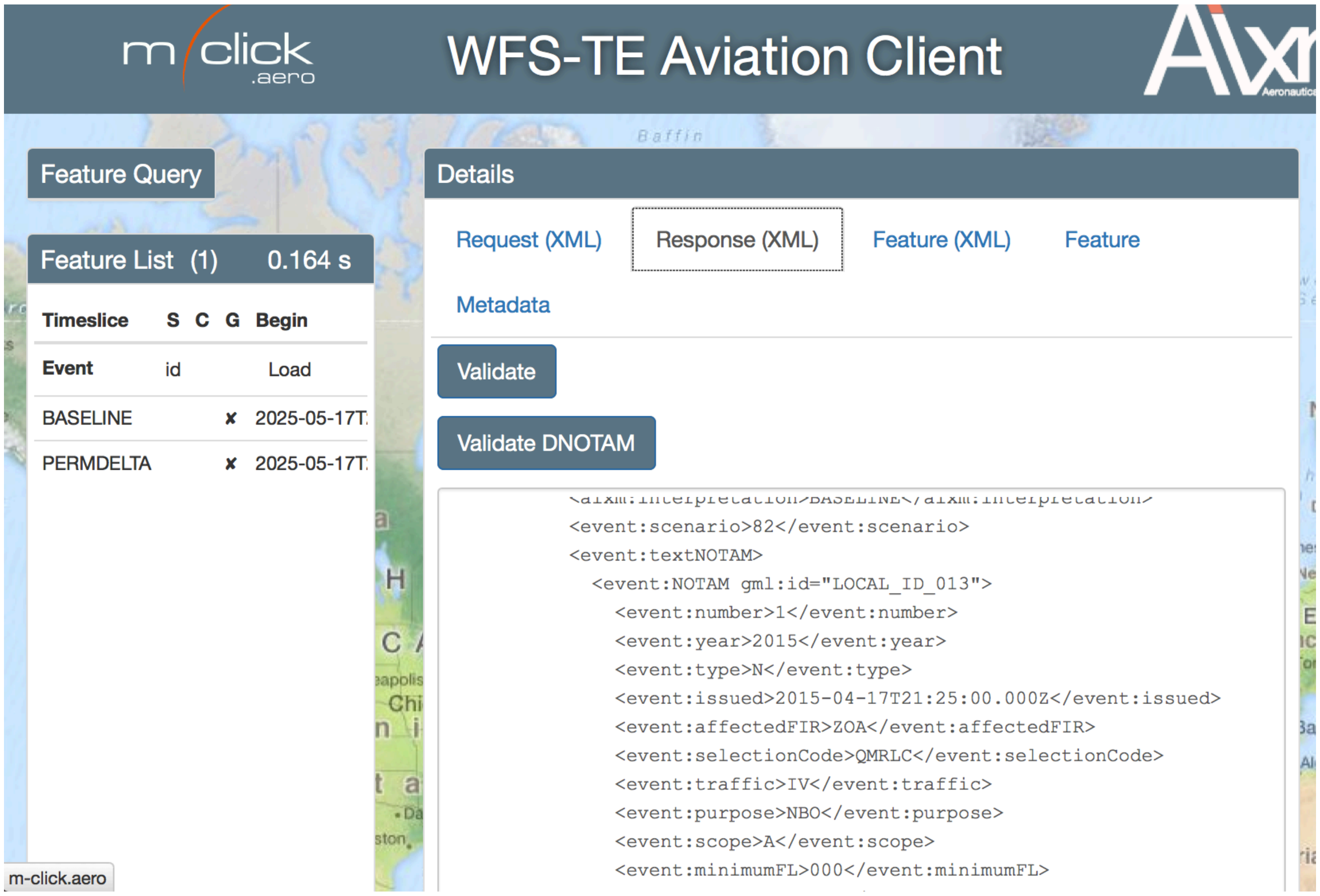


Figure 11 – Integrated Enrichment and Validation Service

The enrichment and validation services are integrated via an AIXM repository component accessible from both services. Enrichment results are stored in a repository and are provided to the validator using URLs (via WFS). The ability of validation services to fetch input document based on given URL is useful characteristic provided in the OGC WPS 1.0 service standard.



Timeslice	S	C	G	Begin
Event	id			Load
BASELINE		✘		2025-05-17T
PERMDELTA		✘		2025-05-17T

```
<aixm:interpretation>BASELINE</aixm:interpretation>
<event:scenario>82</event:scenario>
<event:textNOTAM>
  <event:NOTAM gml:id="LOCAL_ID_013">
    <event:number>1</event:number>
    <event:year>2015</event:year>
    <event:type>N</event:type>
    <event:issued>2015-04-17T21:25:00.000Z</event:issued>
    <event:affectedFIR>ZOA</event:affectedFIR>
    <event:selectionCode>QMRLC</event:selectionCode>
    <event:traffic>IV</event:traffic>
    <event:purpose>NBO</event:purpose>
    <event:scope>A</event:scope>
    <event:minimumFL>000</event:minimumFL>
```

Figure 12 - Integration into an aviation client

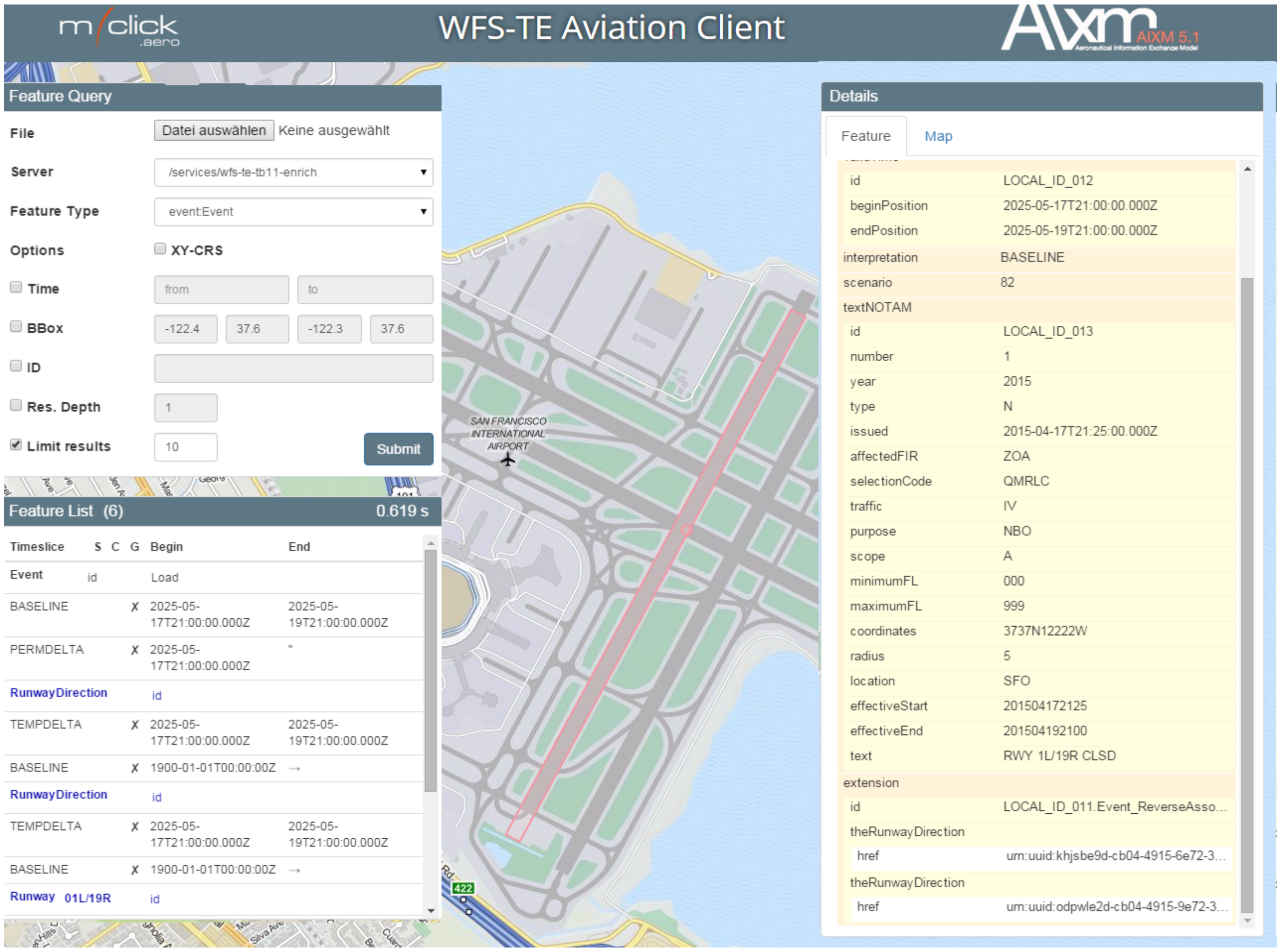
Although the enrichment and validation services are dedicated, autonomously operated entities, they share common infrastructure, such as the AIXM static data repositories. In some validation cases, the enrichment is an unavoidable precondition for validation in accordance with specified normative rules. For practical reasons, both services were implemented based on the same functionality core.

8.8 DNOTAM Event with reverse associations

In order to perform an advanced enrichment process with full integration of NOTAMs and relevant aeronautical entities and their timeslices m-click.aero provided an additional version of the enrichment service. The version includes certain modification in the event schema including bidirectional (reverse) associations between events and static aeronautical entities. This approach enabled WFS 2.0 based navigation through a network of static data and associated event messages stored in an intermediary aeronautical database for purposes of advanced visualization and processing.

An alternative to "reverse associations extension" would be to implement dedicated, additional computational logic for BASELINE timeslices retrieval. In Snowflake's enrichment service implementation involving runway closures ('RWY.CLS'), additional software components were deployed in order to retrieve the relevant features when direct references are not available. This is important for the retrieval of ‘child’ features that often contain the required geometries that are not referenced by other features within the scenario such as aixm:RunwayElement.

A screen snapshot from an aviation client has been provided below (a typical example for functions contained in a typical pre-flight briefing application):


m click .aero
WFS-TE Aviation Client
AIXM 5.1
Aeronautical Information Exchange Model
SAN FRANCISCO INTERNATIONAL AIRPORT

Feature Query

File	Datei auswählen Keine ausgewählt
Server	/services/wfs-te-tb11-enrich
Feature Type	event:Event
Options	☐ XY-CRS
☐ Time	from / to
☐ BBox	-122.4 / 37.6 / -122.3 / 37.6
☐ ID	
☐ Res. Depth	1
☑ Limit results	10

Submit

Feature List (6) 0.619 s

Timeslice	S	C	G	Begin	End
Event	id			Load	
BASELINE			✗	2025-05-17T21:00:00.000Z	2025-05-19T21:00:00.000Z
PERMDELTA			✗	2025-05-17T21:00:00.000Z	"
RunwayDirection				id	
TEMPDELTA			✗	2025-05-17T21:00:00.000Z	2025-05-19T21:00:00.000Z
BASELINE			✗	1900-01-01T00:00:00Z	→
RunwayDirection				id	
TEMPDELTA			✗	2025-05-17T21:00:00.000Z	2025-05-19T21:00:00.000Z
BASELINE			✗	1900-01-01T00:00:00Z	→
Runway 01L/19R				id	

Details

Feature Map

id	LOCAL_ID_012
beginPosition	2025-05-17T21:00:00.000Z
endPosition	2025-05-19T21:00:00.000Z
interpretation	BASELINE
scenario	82
textNOTAM	
id	LOCAL_ID_013
number	1
year	2015
type	N
issued	2015-04-17T21:25:00.000Z
affectedFIR	ZOA
selectionCode	QMRLC
traffic	IV
purpose	NBO
scope	A
minimumFL	000
maximumFL	999
coordinates	3737N12222W
radius	5
location	SFO
effectiveStart	201504172125
effectiveEnd	201504192100
text	RWY 1L/19R CLSD
extension	
id	LOCAL_ID_011.Event_ReverseAsso...
theRunwayDirection	
href	urn:uuid:khjsbe9d-cb04-4915-6e72-3...
theRunwayDirection	
href	urn:uuid:odpwle2d-cb04-4915-9e72-3...

Figure 13 – An aviation client depicts runway closure using reverse event associations

If for example a runway direction was modified, the corresponding notification will ideally be distributed as an AIXM 5.1 message containing an event element and runway’s TEMPDELTA timeslice, This piece of information is obviously not sufficient to portray the runway and surrounding environment (because it only contains the modification). In order to portray the fully operational picture, the event notification message must be enriched with additional aeronautical entities and timeslice elements. If required entities and their timeslices are already available in a local database aware of associations between aeronautical entities, this storage can perform implicit enrichment based on general functions of WFS-T 2.0 endpoint implementation and associations between stored entities.

A simple WFS 2.0 query containing a single event ID returns not only a NOTAM (displayed on the right side of Figure 13 in the yellow panel) but also associated entities: a single runway and two runway directions which appear in the feature list panel. These entities are resolved thanks to the enrichment with reverse associations. For example, the runway portrayal in the Figure 13 – An aviation client depicts runway closure using reverse event associationswas performed with assistance of enrichment service. Otherwise, the entities would remain unknown (or hard to retrieve) and the local stored NOTAM almost useless.

8.9 Geometry Construction for DNOTAM enrichment

Snowflake Software's implementation of the Digital NOTAM enrichment service allowed non-spatial features to be queried spatially via the WFS interface by aggregating child feature geometries where applicable. For example, aixm:Runway does not contain a geometry. In order to spatially query the aixm:Runway feature, a geometry for the runway is constructed based on the relevant aixm:RunwayElement features.

A similar process is undertaken for the event:Event feature gml:BoundedBy element. This is populated based on a minimum-bounding box around the related features; again this geometry construction is executed during the internal static data retrieval stage of the enrichment process, therefore allowing the enriched Digital NOTAMs to be queried spatially.

9 Identified Issues and Future Work

9.1 Validation Service Endpoint

While the OGC WPS 1.0 standard provides the best available option for the validation service, an alternative using the new WPS 2.0 might be considered.

9.2 Validation Result Data Format

The ISO GMD result data format has been proven to be useful but might also be seen as quite verbose and far beyond the validation service needs.

9.3 Use of WFS-TE for Enrichment

If aeronautical static data is available via WFS with Temporal Extension [WFS-TE], the processing of timeslice data in the enrichment service would be simplified due to more precise and rapid data retrieval. Using a standard WFS 2.0 without AIXM temporality awareness, an enrichment service will always receive all timeslices from an entity and will need to filter out those not relevant for the NOTAM. Alternatively, the filtering of time slices could integrated into the internal processing logic that is triggered on digital NOTAM insert.

9.4 NOTAM specific validation rules

The common purpose AIXM 5.1 validation rules were available during the work in the Testbed, but there is an issue related to the partial availability of mature (Digital NOTAM-specific) validation rules. In order to provide more comprehensive validation, sponsors provided several digital NOTAM specific rules in accordance with scenarios. They were validated together with an extensive list of common purpose AIXM 5.1 formatting rules.

9.5 AIXM 5.1 specific validation rules affecting NOTAM payloads

There were several issues affecting the validation process, which are related to the digital NOTAM payload timeslices. For the sake of clarity, the proposal to use only the BASELINE and TEMPDELTA timeslices was made assuming that at the data originator, every change on an entity automatically means creation of a new, fresh BASELINE TimeSlice. That affects both the modifications of NOTAMs, as well as the modifications of static data (either included directly in the NOTAM payload or retrieved during the process of enrichment).

9.6 Nil element declarations from GML 3.2

Declaring an element to be nil is an implementation of the “Void” data type of ISO/IEC 11404 [GPD], i.e. represents “an object whose presence is syntactically or semantically required, but carries no information in a given instance” [ISO/IEC 11404]. From an XML validation perspective, having both @xsi:nil=’true' and @xlink:href is valid. However,

@xsi:nil=’true' indicates that the property has no value in this instance and as a result, processing software will ignore the @xlink:href. For example, in Listing 2 attribute xlink:href will be ignored unless we remove the xsi:nill which will lead to schema inconsistency. This will have further consequences on the validation.

```
      <aixm:extension>
        <event:AirportHeliportExtension gml:id="ID_667">
          <event:theEvent xlink:href="#uuid.f50faabn" xsi:nil="true"/>
        </event:AirportHeliportExtension>
      </aixm:extension>
```

Listing 2 - xlink:href with a nil attribute

9.7 Identification of timeSlice elements

During the work on the WFS temporality extension, as well as on the validator, a recurring problem was observed: how to safely recognize the TimeSlice elements of any feature? The simple answer is: "All elements named aixm:timeSlice are TimeSlices.” However, some AIXM extensions such as DNOTAM define a new element such as "event:timeSlice," which is a different element because it has a different namespace. Every AIXM extension with custom feature types should always name their time slice elements "TimeSlice" so that a system can count on the local name, even if the namespace differs. However, this requirement cannot be enforced in the AIXM schema nor via Business Rules (because Business Rules operate on the data, while this is a rule about the extension schemas).

9.8 Bidirectional associations between AIXM 5.1 entities

Events communicate temporal or permanent changes of AIXM entities. That is the reason why entities have to be associated with events. The association is (as prescribed in the AIXM 5.1 schema) directed from child (entity) to parent element (event). A TEMPDELTA TimeSlice always references a Digital NOTAM Event feature.

In some cases a Digital NOTAM event is related to an aeronautical entity, which is further integrated into more complex structures. For example, an aixm:RunwayDirection feature belongs to an aixm:Runway which belongs to an aixm:AirportHeliport. Such relationships must be considered in both the enrichment and validation cases. Resolving such associations, the enrichment service can fetch all needed timeslices to create an enriched Digital NOTAM message. If some entity affected by a digital NOTAM message is referenced by other entities (for example, an aixm:AirportHeliport is referenced by related aixm:Runway entities, however the aixm:AirportHeliport does not reference the related aixm:Runway entities), the enrichment is not simple. Additional processing steps to compensate the lack of bidirectional associations or advanced logic in the enrichment process are required.

These steps can be integrated into internal processing logic that is triggered on insert of a Digital NOTAM. A sequential approach to entity and related entity BASELINE timeslice retrieval is required in order to retrieve all related entities. The processing logic should also be aware of Digital NOTAM scenarios and the expected contents of enriched Digital NOTAMs according to each scenario.

This problem is even more visible if both the Digital NOTAM and static data are stored in a repository and browsed by a client. The Digital NOTAM is not aware of entities it is related to. While a related entity is aware of a Digital NOTAM some associated entities as described above will still be missing.

In order to provide more flexible processing in both enrichment and validation, an advanced reverse association mechanism is needed (not only what the NOTAM concerns but also in general, for all AIXM 5.1 entities). Therefore, an enhanced AIXM version with bidirectional links between entities (reverse associations) like the unofficial MXIA, an AIXM 5.1 based schema with bidirectional extension, would simplify the enrichment and validation process.

9.9 Feature Identification in the Enrichment Service

The enrichment service performs identification and fetching of BASELINE timeslices based on event's TEMPDELTA timeslice. Obviously, it can only work if both timeslices have same source/origin and compatible key/identification methods (NOTAM Office or some authoritative static data provider). The recommended method is the use of xlink:href and UUID. Both validator and enrichment service assumes entity resolving using UUID and not via natural keys.

9.10 Types of Timeslices used for Validation

In this Testbed, the validation process was executed using enriched event documents, which contain BASELINE and TEMPDELTA timeslices. SNAPSHOT Timeslices were used only in FAA NOTAMs (which were provided already enriched). The enriched event document rules are more complex compared to those performed on SNAPSHOTs because they need to consider more than one TEMPDELTA. This is unavoidable because some information contained in TEMPDELTAs is relevant for the overall process of validation. For example, some rules requests certain types of attributes not to be modified in a PERMDELTA timeslice. Future work could explore the option to enrich a NOTAM with more consistent SNAPSHOT timeslices and perform the validation using more compact and less verbose validation rules.

9.11 Reverse association between events and aeronautical entities

Modification of the event schema is recommended in order to implement bidirectional associations between events and affected static data entities. Implementation of bidirectional associations between events and affected static data would simplify the enrichment process somewhat by removing advanced enrichment logic and integrating events into local or global static data repository. This issue and recommendation are related to the limitations in the AIXM 5.1 schema, the event specification extension, as well as the awareness of WFS service interface standard regarding the specific features of AIXM and the relations between entities and its temporality model. As future work we recommend further analysis regarding these issues.

10 Annex A – A single Schematron rule example (runway direction)

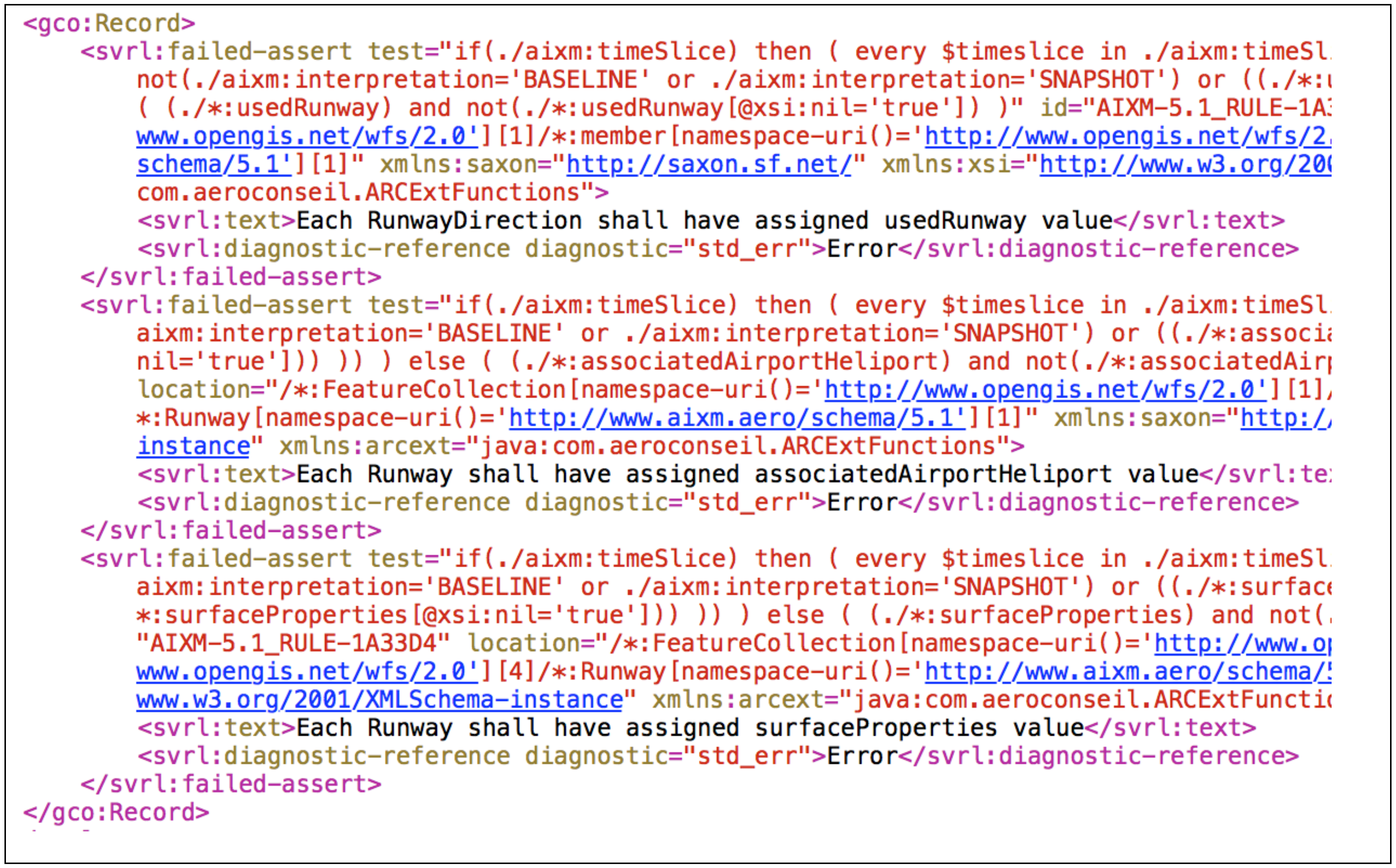
```
<rule context="aixm:RunwayDirection">
  <assert id="AIXM-5.1_RULE-1A2F70" test="every $x1 in current()/aixm:timeSlice/*/aixm:annotation/*/aixm:propertyName satisfies (not(not($x1[@xsi:nil=
      'true']) and not($x1 = ('designator','trueBearing','trueBearingAccuracy','magneticBearing','patternVFR','slopeTDZ','elevationTDZ',
      'elevationTDZAccuracy','approachMarkingType','approachMarkingCondition','classLightingJAR','precisionApproachGuidance','usedRunway',
      'startingElement','availability'))))">Each RunwayDirection.annotation.Note.propertyName shall not have assigned value other than ('designator',
      'trueBearing', 'trueBearingAccuracy', 'magneticBearing', 'patternVFR', 'slopeTDZ', 'elevationTDZ', 'elevationTDZAccuracy', 'approachMarkingType',
      'approachMarkingCondition', 'classLightingJAR', 'precisionApproachGuidance', 'usedRunway', 'startingElement', 'availability')</assert>
  <assert id="AIXM-5.1_RULE-1A3317" test="every $x1 in current() satisfies (for $c1 in count(for $x2 in $x1/aixm:timeSlice/*/aixm:usedRunway return if
      (not($x2[@xsi:nil='true'])) then 1 else ()) return ($c1 &gt;= 1))">Each RunwayDirection shall have assigned usedRunway value</assert>
  <assert id="AIXM-5.1_RULE-1A336B" test="every $x1 in current() satisfies (for $c1 in count(for $x2 in $x1/aixm:timeSlice/*/aixm:designator return if
      (not($x2[@xsi:nil='true'])) then 1 else ()) return ($c1 &gt;= 1))">Each RunwayDirection shall have assigned designator value</assert>
  <assert id="AIXM-5.1_RULE-1A85AF" test="every $x1 in current() satisfies (not(for $c1 in count(for $x2 in $x1/aixm:timeSlice/*/aixm:usedRunway return
      if (not($x2[@xsi:nil='true'])) then 1 else ()) return ($c1 &gt;= 1)) or (for $c2 in count(for $x3 in $x1/aixm:timeSlice/*/aixm:usedRunway return if
      (not($x3[@xsi:nil='true'])) then 1 else ()) return ($c2 = 1)))">It is obligatory that each RunwayDirection with assigned usedRunway value uses
      exactly one Runway</assert>
  <assert id="AIXM-5.1_RULE-1A85B0" test="every $x1 in current() satisfies (not(for $c1 in count(for $x2 in $x1/aixm:timeSlice/*/aixm:startingElement
      return if (not($x2[@xsi:nil='true'])) then 1 else ()) return ($c1 &gt;= 1)) or (for $c2 in count(for $x3 in $x1/aixm:timeSlice/*/aixm:
      startingElement return if (not($x3[@xsi:nil='true'])) then 1 else ()) return ($c2 = 1)))">It is obligatory that each RunwayDirection with assigned
      startingElement value hasStart exactly one RunwayElement</assert>
  <assert id="AIXM-5.1_RULE-D8D55" test="every $x1 in current() satisfies (not(for $c1 in count(for $x2 in $x1/aixm:timeSlice/*/aixm:elevationTDZ return
      if (not($x2[@xsi:nil='true'])) then 1 else ()) return ($c1 &gt;= 1)) or (for $c2 in count(for $x3 in $x1/aixm:timeSlice/*/aixm:elevationTDZ/@uom
      return if (not(not(string-length(normalize-space($x3))))) then 1 else ()) return ($c2 &gt;= 1)))">Each RunwayDirection with assigned elevationTDZ
      shall have assigned  elevationTDZ.uom value</assert>
  <assert id="AIXM-5.1_RULE-D8D56" test="every $x1 in current() satisfies (not(for $c1 in count(for $x2 in $x1/aixm:timeSlice/*/aixm:elevationTDZAccuracy
      return if (not($x2[@xsi:nil='true'])) then 1 else ()) return ($c1 &gt;= 1)) or (for $c2 in count(for $x3 in $x1/aixm:timeSlice/*/aixm:
      elevationTDZAccuracy/@uom return if (not(not(string-length(normalize-space($x3))))) then 1 else ()) return ($c2 &gt;= 1)))">Each RunwayDirection
      with assigned elevationTDZAccuracy shall have assigned  elevationTDZAccuracy.uom value</assert>
  <assert id="AIXM-5.1_RULE-E1578" test="every $x1 in current()/aixm:timeSlice satisfies (not((for $c3 in count(for $x2 in $x1/*/aixm:extension/event:
      theEvent return if ((for $c1 in count(for $x3 in //*[concat('#',@gml:id)=$x2/@xlink:href]/event:timeSlice/*/event:scenario return if ($x3 =
      'RWY.CLS') then 1 else ()) return ($c1 &gt;= 1)) and (for $c2 in count(for $x4 in //*[concat('#',@gml:id)=$x2/@xlink:href]/event:timeSlice/*/event:
      version return if ($x4 = '2.0') then 1 else ()) return ($c2 &gt;= 1))) then 1 else ()) return ($c3 &gt;= 1)) and (for $c4 in count(for $x5 in $x1/
      */aixm:availability/*/aixm:usage/*/aixm:operation return if (not($x5 = ('PERMIT','CONDITIONAL'))) then 1 else ()) return ($c4 &gt;= 1))))">It is
      prohibited that a RunwayDirection.timeSlice belongsTo Event with scenario equal-to 'RWY.CLS' and with version equal-to '2.0' and has
      availability.ManoeuvringAreaAvailability.usage.ManoeuvringAreaUsage.operation not equal-to ('PERMIT', 'CONDITIONAL')</assert>
  <assert id="AIXM-5.1_RULE-EB5A0" test="every $x1 in current()/aixm:timeSlice satisfies (not(for $c3 in count(for $x2 in $x1/*/aixm:extension/event:
      theEvent return if ((for $c1 in count(for $x3 in //*[concat('#',@gml:id)=$x2/@xlink:href]/event:timeSlice/*/event:scenario return if ($x3 =
      'RWY.CLS') then 1 else ()) return ($c1 &gt;= 1)) and (for $c2 in count(for $x4 in //*[concat('#',@gml:id)=$x2/@xlink:href]/event:timeSlice/*/event:
      version return if ($x4 = '2.0') then 1 else ()) return ($c2 &gt;= 1))) then 1 else ()) return ($c3 &gt;= 1)) or (for $c4 in count(for $x5 in $x1/*/
      aixm:availability return if (not($x5[@xsi:nil='true'])) then 1 else ()) return ($c4 = 1)) and (for $c5 in count(for $x6 in $x1/*/aixm:availability/
      */aixm:operationalStatus return if ($x6 = 'CLOSED') then 1 else ()) return ($c5 &gt;= 1)))">Each RunwayDirection.timeSlice that belongsTo Event
      with scenario equal-to 'RWY.CLS' and with version equal-to '2.0' shall have exactly one assigned availability value and shall have
      availability.ManoeuvringAreaAvailability.operationalStatus equal-to 'CLOSED'</assert>
  <assert id="AIXM-5.1_RULE-F2EBA" test="every $x1 in current() satisfies (not(for $c3 in count(for $x2 in $x1/aixm:timeSlice/*/aixm:extension/event:
      theEvent return if ((for $c1 in count(for $x3 in //*[concat('#',@gml:id)=$x2/@xlink:href]/event:timeSlice/*/event:scenario return if ($x3 =
      'RWY.CLS') then 1 else ()) return ($c1 &gt;= 1)) and (for $c2 in count(for $x4 in //*[concat('#',@gml:id)=$x2/@xlink:href]/event:timeSlice/*/event:
      version return if ($x4 = '2.0') then 1 else ()) return ($c2 &gt;= 1))) then 1 else ()) return ($c3 &gt;= 1)) or (for $c4 in count(for $x5 in $x1/
      aixm:timeSlice/*/aixm:interpretation return if ($x5 = 'TEMPDELTA') then 1 else ()) return ($c4 &gt;= 1)))">Each RunwayDirection that belongsTo
      Event with scenario equal-to 'RWY.CLS' and with version equal-to '2.0' shall have interpretation equal-to 'TEMPDELTA'</assert>
</rule>
```

11 Annex B – Validation Result Example

```
<gmd:MD_Metadata>
    <gmd:contact gco:nilReason="inapplicable"/>
    <gmd:dateStamp gco:nilReason="inapplicable"/>
    <gmd:identificationInfo gco:nilReason="inapplicable"/>
    <gmd:dataQualityInfo>
        <gmd:DQ_DataQuality>
            <gmd:scope gco:nilReason="inapplicable"/>
            <gmd:report>
                <gmd:DQ_DomainConsistency>
                    <gmd:nameOfMeasure>
                        <gco:CharacterString>OGC Testbed-11 DNOTAM Validation</gco:CharacterString>
                    </gmd:nameOfMeasure>
                    <gmd:measureIdentification>
                        <gmd:MD_Identifier>
                            <gmd:code>
                                <gco:CharacterString>m-click DNOTAM Validator.</gco:CharacterString>
                            </gmd:code>
                        </gmd:MD_Identifier>
                    </gmd:measureIdentification>
                    <gmd:dateTime>
                        <gco:DateTime>2015-04-27T11:45:10.924Z</gco:DateTime>
                    </gmd:dateTime>
                    <gmd:result>
                        <gmd:DQ_QuantitativeResult>
                            <gmd:valueType>
                                <gco:RecordType>Issues</gco:RecordType>
                            </gmd:valueType>
                            <gmd:valueUnit gco:nilReason="inapplicable"/>
                            <gmd:value>
                                <gco:Record>

                                    ....

                                </gco:Record>
                            </gmd:value>
                        </gmd:DQ_QuantitativeResult>
                    </gmd:result>
                </gmd:DQ_DomainConsistency>
            </gmd:report>
        </gmd:DQ_DataQuality>
    </gmd:dataQualityInfo>
</gmd:MD_Metadata>
```



```
<gco:Record>
    <svrl:failed-assert test="if(./aixm:timeSlice) then ( every $timeslice in ./aixm:timeSl
        not(./aixm:interpretation='BASELINE' or ./aixm:interpretation='SNAPSHOT') or ((./*:
        ( (./*:usedRunway) and not(./*:usedRunway[@xsi:nil='true']) )" id="AIXM-5.1_RULE-1A
        www.opengis.net/wfs/2.0'][1]/*:member[namespace-uri()='http://www.opengis.net/wfs/2
        schema/5.1'][1]" xmlns:saxon="http://saxon.sf.net/" xmlns:xsi="http://www.w3.org/20
        com.aeroconseil.ARCExtFunctions">
        <svrl:text>Each RunwayDirection shall have assigned usedRunway value</svrl:text>
        <svrl:diagnostic-reference diagnostic="std_err">Error</svrl:diagnostic-reference>
    </svrl:failed-assert>
    <svrl:failed-assert test="if(./aixm:timeSlice) then ( every $timeslice in ./aixm:timeSl
        aixm:interpretation='BASELINE' or ./aixm:interpretation='SNAPSHOT') or ((./*:associ
        nil='true'])) )) ) else ( (./*:associatedAirportHeliport) and not(./*:associatedAir
        location="/*:FeatureCollection[namespace-uri()='http://www.opengis.net/wfs/2.0'][1]
        *:Runway[namespace-uri()='http://www.aixm.aero/schema/5.1'][1]" xmlns:saxon="http:/
        instance" xmlns:arcext="java:com.aeroconseil.ARCExtFunctions">
        <svrl:text>Each Runway shall have assigned associatedAirportHeliport value</svrl:te
        <svrl:diagnostic-reference diagnostic="std_err">Error</svrl:diagnostic-reference>
    </svrl:failed-assert>
    <svrl:failed-assert test="if(./aixm:timeSlice) then ( every $timeslice in ./aixm:timeSl
        aixm:interpretation='BASELINE' or ./aixm:interpretation='SNAPSHOT') or ((./*:surfac
        *:surfaceProperties[@xsi:nil='true'])) )) ) else ( (./*:surfaceProperties) and not(
        "AIXM-5.1_RULE-1A33D4" location="/*:FeatureCollection[namespace-uri()='http://www.o
        www.opengis.net/wfs/2.0'][4]/*:Runway[namespace-uri()='http://www.aixm.aero/schema/
        www.w3.org/2001/XMLSchema-instance" xmlns:arcext="java:com.aeroconseil.ARCExtFuncti
        <svrl:text>Each Runway shall have assigned surfaceProperties value</svrl:text>
        <svrl:diagnostic-reference diagnostic="std_err">Error</svrl:diagnostic-reference>
    </svrl:failed-assert>
</gco:Record>
```

Revision history

Date	Release	Editor	Primary clauses modified	Description
2015-02-20	0.1	Aleksandar Balaban		First draft with content
2015-04-25	0.2.8	Aleksandar Balaban		Second draft, 80% finished
2015-05-13	1.0	Aleksandar Balaban		Final draft, 95% finished
2015-05-14	1.1	Aleksandar Balaban		Final draft with Snowflake contribution, 98% finished
2015-12-30	1.1a	Scott Simmons	All	Edits for final publication