

Open Geospatial Consortium

Publication Date: 2014-07-15

Approval Date: 2014-06-14

Posted Date: 2014-05-16

Reference number of this document: OGC 14-007

Reference URL for this document: <http://www.opengeospatial.net/doc/PER/testbed10/aviation-binding-aixm-tools>

Category: Public Engineering Report

Editor: Matthes Rieke

OGC[®] Testbed 10 Report on Aviation Binding AIXM to Development Tools

Copyright © 2014 Open Geospatial Consortium.

To obtain additional rights of use, visit <http://www.opengeospatial.org/legal/>.

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.

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

Abstract

This document is a deliverable of the OGC Testbed 10 (Testbed-10). Its contents cover the summary of the work carried out regarding the creation and evaluation of generated data bindings for the Aeronautical Information Exchange Model (AIXM) for established programming languages.

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.

Keywords

ogcdoc, ogc documents, ows10, aviation, aixm, jaxb

License Agreement

Permission is hereby granted by the Open Geospatial Consortium, ("Licensor"), free of charge and subject to the terms set forth below, to any person obtaining a copy of this Intellectual Property and any associated documentation, to deal in the Intellectual Property without restriction (except as set forth below), including without limitation the rights to implement, use, copy, modify, merge, publish, distribute, and/or sublicense copies of the Intellectual Property, and to permit persons to whom the Intellectual Property is furnished to do so, provided that all copyright notices on the intellectual property are retained intact and that each person to whom the Intellectual Property is furnished agrees to the terms of this Agreement.

If you modify the Intellectual Property, all copies of the modified Intellectual Property must include, in addition to the above copyright notice, a notice that the Intellectual Property includes modifications that have not been approved or adopted by LICENSOR.

THIS LICENSE IS A COPYRIGHT LICENSE ONLY, AND DOES NOT CONVEY ANY RIGHTS UNDER ANY PATENTS THAT MAY BE IN FORCE ANYWHERE IN THE WORLD.

THE INTELLECTUAL PROPERTY IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF THIRD PARTY RIGHTS. THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE DO NOT WARRANT THAT THE FUNCTIONS CONTAINED IN THE INTELLECTUAL PROPERTY WILL MEET YOUR REQUIREMENTS OR THAT THE OPERATION OF THE INTELLECTUAL PROPERTY WILL BE UNINTERRUPTED OR ERROR FREE. ANY USE OF THE INTELLECTUAL PROPERTY SHALL BE MADE ENTIRELY AT THE USER'S OWN RISK. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR ANY CONTRIBUTOR OF INTELLECTUAL PROPERTY RIGHTS TO THE INTELLECTUAL PROPERTY BE LIABLE FOR ANY CLAIM, OR ANY DIRECT, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM ANY ALLEGED INFRINGEMENT OR ANY LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR UNDER ANY OTHER LEGAL THEORY, ARISING OUT OF OR IN CONNECTION WITH THE IMPLEMENTATION, USE, COMMERCIALIZATION OR PERFORMANCE OF THIS INTELLECTUAL PROPERTY.

This license is effective until terminated. You may terminate it at any time by destroying the Intellectual Property together with all copies in any form. The license will also terminate if you fail to comply with any term or condition of this Agreement. Except as provided in the following sentence, no such termination of this license shall require the termination of any third party end-user sublicense to the Intellectual Property which is in force as of the date of notice of such termination. In addition, should the Intellectual Property, or the operation of the Intellectual Property, infringe, or in LICENSOR's sole opinion be likely to infringe, any patent, copyright, trademark or other right of a third party, you agree that LICENSOR, in its sole discretion, may terminate this license without any compensation or liability to you, your licensees or any other party. You agree upon termination of any kind to destroy or cause to be destroyed the Intellectual Property together with all copies in any form, whether held by you or by any third party.

Except as contained in this notice, the name of LICENSOR or of any other holder of a copyright in all or part of the Intellectual Property shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Intellectual Property without prior written authorization of LICENSOR or such copyright holder. LICENSOR is and shall at all times be the sole entity that may authorize you or any third party to use certification marks, trademarks or other special designations to indicate compliance with any LICENSOR standards or specifications.

This Agreement is governed by the laws of the Commonwealth of Massachusetts. The application to this Agreement of the United Nations Convention on Contracts for the International Sale of Goods is hereby expressly excluded. In the event any provision of this Agreement shall be deemed unenforceable, void or invalid, such provision shall be modified so as to make it valid and enforceable, and as so modified the entire Agreement shall remain in full force and effect. No decision, action or inaction by LICENSOR shall be construed to be a waiver of any rights or remedies available to it.

None of the Intellectual Property or underlying information or technology may be downloaded or otherwise exported or reexported in violation of U.S. export laws and regulations. In addition, you are responsible for complying with any local laws in your jurisdiction which may impact your right to import, export or use the Intellectual Property, and you represent that you have complied with any regulations or registration procedures required by applicable law to make this license enforceable.

Contents		Page
1	Introduction.....	1
1.1	Scope	1
1.2	Document contributor contact points	1
1.3	Revision history.....	1
1.4	Future work	2
1.5	Foreword	2
2	References.....	2
3	Conventions	3
3.1	Abbreviated terms	3
3.2	UML notation.....	3
4	OWS-10 Report on Aviation Binding AIXM to Development Tools overview.....	4
5	Technological Overview.....	4
5.1	Automated Data Binding Generation	4
5.2	Glassfish/Metro	5
5.3	Microsoft .NET Framework.....	5
6	Binding Generation Approach	6
6.1	JAXB	6
6.1.1	Required Schema Adjustments	6
6.1.2	Schema Generation	7
6.1.2.1	Command line	7
6.1.2.2	maven-jaxb2-plugin.....	8
6.1.3	Roundtrip Execution	10
6.1.3.1	Performance.....	11
6.1.3.2	Web Processing Service Interface	12
6.1.3.3	Issues	13
6.1.4	Language Integration	13
6.1.5	Event Service Integration.....	14
6.1.6	Source Code.....	14
6.2	.NET Framework.....	14
6.2.1	Required Schema Adjustments	15
6.2.2	Schema Generation	15
6.2.3	Roundtrip Execution	16
6.2.3.1	Performance.....	16
6.2.3.2	Issues	17
6.2.4	Language Integration	18
6.2.5	Source Code.....	18

7	General Assessments	19
8	Accomplishments.....	19
9	Lessons Learned.....	19
10	Recommendations.....	20

Figures		Page
Figure 1: Visual Studio Command Prompt - Binding Generation.		6
Figure 2: Roundtrip performance for JAXB bindings (one feature with one timeslice).		11
Figure 3: Roundtrip performance for JAXB bindings (one feature with 100 timeslices).		12
Figure 4: Roundtrip performance for C# bindings (one feature with one timeslice).		17
Figure 5: Roundtrip performance for C# bindings (one feature with 100 timeslices).		17

Tables		Page
Table 1: Required Adjustments to Schemas for JAXB generation.		7
Table 2: Roundtrip Issues for JAXB bindings.		13
Table 3: Required Adjustments for C# generation.		15
Table 4: Roundtrip Issues for C# bindings.		18

Listings		Page
Listing 1: Resolution of an element naming clash.		8
Listing 2: Maven project configuration for JAXB bindings.		9
Listing 3: Roundtrip using Java with JAXB.		10
Listing 4: Exemplary WPS Roundtrip Execution.		12
Listing 5: Extracting information from a Navaid feature.		14
Listing 6: Roundtrip Execution using C#.		16
Listing 7: Resolving the interpretation value of a NavaidTimeSlice.		18

OGC® Testbed 10 Report on Aviation Binding AIXM to Development Tools

1 Introduction

1.1 Scope

This OGC document provides an overview on the approach for automatically generating data bindings for the Aeronautical Information Exchange Model (AIXM), version 5.1. It covers the descriptions of the programming language environments for which these bindings have been created. Besides the documentation observed issues, the results are evaluated in terms of language semantic integration and performance. This report covers a general assessment of the applied approach and provides recommendations on how to foster the usage of data bindings for AIXM.

1.2 Document contributor contact points

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

Name	Organization
Matthes Rieke (MRI), editor (m.rieke<at>52north.org)	52°North GmbH

1.3 Revision history

Date	Release	Editor	Primary clauses modified	Description
2014-01-31	0.1.0	MRI	all	Initial draft report
2014-04-15	0.9.0	MRI	1.4, 5-10, Annex	Pre-final version of the report
2014-05-16	1.0.0	MRI		Edits for the final version of the report

1.4 Future work

Improvements in this document are desirable on the following topics:

- **Investigation on existing roundtrip issues** – issues with the contents of the resulting documents of a roundtrip should be investigated further, for both development environments.
- **Taking alternative approaches and languages into account** – Besides the assessed two development environments several other solutions exist for the generation of data bindings. As these might provide better results, an assessment would be valuable.
- **AIXM binding software suite** – a comprehensive software suite providing convenience methods on processing AIXM features (e.g. TimeSlice management) could be built on top of the created schema bindings. A basis for this could be the already developed wrapper components available as Open Source Software.

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.

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

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

In addition to this document, this report includes several XML Document files as specified in Annex A.

3 Conventions

3.1 Abbreviated terms

AIXM	Aeronautical Information Exchange Model
DNOTAM	Digital NOTAM
IDE	Integrated Development Environment
ES	Event Service
GML	Geography Markup Language
HTTP	Hypertext Transfer Protocol
JAXB	Java Architecture for XML Binding
NOTAM	Notice to Airmen
OGC	Open Geospatial Consortium
OWS-10	OWS Test bed Phase 10
UUID	Universally Unique Identifier
UML	Unified Modeling Language
WFS	Web Feature Service
WPS	Web Processing Service
WXXM	Weather Information Exchange Model
XML	Extensible Markup Language
XPath	XML Path Language

3.2 UML notation

Most diagrams that appear in this standard are presented using the Unified Modeling Language (UML) static structure diagram, as described in Subclause 5.2 of [OGC 06-121r3]. Besides static structure diagrams, sequence diagrams are used to illustrate information/data flow.

4 Testbed 10 Report on Aviation Binding AIXM to Development Tools overview

The Aeronautical Information Exchange Model (AIXM) has been established as a reliable encoding for ATM related feature data. Its version 5.1 is based on the Geographic Markup Language (GML), version 3.2.1. Since both AIXM and GML encompass a wide range of data structures as well as a flexible design in terms of extensibility and options, it is a challenging task for software developers to support every nuance of these.

Therefore, work on the automated creation of generated AIXM data bindings for established programming languages has been carried out within the OWS-10 test bed. This report summarizes the approach, the observed issues while creating these bindings, the integration into development tools (IDEs, application servers) and an evaluation in terms of language integration and performance. The focus lies on a Java based environment (namely Glassfish/Metro using JAXB) and C# as part of the Microsoft .NET Framework with its command line tools.

5 Technological Overview

The following sections provide a brief introduction to the fundamental concepts and the applied technologies, namely Glassfish/Metro and JAXB and the Microsoft .NET Framework

5.1 Automated Data Binding Generation

The process of automatically generating bindings for a certain data model can be understood as a subtype of *source code generation*. “Source code generation is the act of generating source code based on an ontological model such as a template and is accomplished with a programming tool such as a template processor or an integrated development environment (IDE)”¹. Though, the entry point for automated data binding generation is not a template or an abstract ontological model but a representation (within this work XML schema definitions) of a certain data model. XML has always been an important technology when defining web services and their interfaces as well as data exchange. Therefore, support for data bindings in existing programming languages plays a central role in the design of these services.

A software component providing automated data binding generation usually also provides convenience methods on serializing and deserializing XML documents. The deserialization into programming language representations with subsequent serialization of these representations back to XML is called “roundtrip” in the following.

¹ http://en.wikipedia.org/wiki/Automatic_programming#Source_code_generation.

For the C# language (as part of the Microsoft .NET Framework) several software solutions supporting this concept exist. The assessment of the “xsd.exe” tool, shipped with .NET, in combination with AIXM 5.1 is a major topic in this report. In addition, several other solutions for C# exist. These include, but are not limited to:

- Liquid XML Data Binding for .Net C# (commercial product)
- XmlObjectMapper - XML Data Binding Framework (open source project)
- Altova XMLSpy 2014 (commercial product)

In the Java world, several projects deal with automatically binding XML schemas against source code. The most important is the Java Architecture for XML Binding (JAXB) which is assessed using AIXM 5.1 within this report. Besides JAXB, other projects exist, including but not limited to:

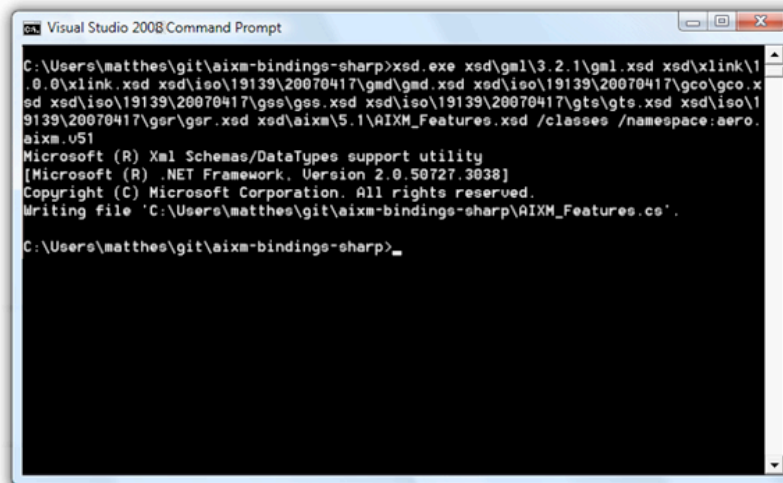
- Apache XMLBeans (open source project)
- JiBX (open source project)
- Altova XMLSpy 2014 (commercial product)

5.2 Glassfish/Metro

JAXB as part of GlassFish/Metro provides sophisticated and contemporary support for complex XML schema definitions. In contrast to other common solutions such as XMLBeans, Metro JAXB is highly configurable at both the code generation as well as in the (un-)marshalling. It provides the basic data interface binding layer for Metro, thus simplifying the creation of domain-specific web services conforming to the JAX-WS API. Schema generation can be achieved via command line tools or by using a build tool such as Apache Maven with corresponding plugins for JAXB.

5.3 Microsoft .NET Framework

Microsoft’s .NET environment provides the capability to create auto-generated source code for C# applications. It ships with the “xsd.exe” command line tool which can be executed manually or within the command line prompt of Microsoft Visual Studio (see Figure 1).



```

Visual Studio 2008 Command Prompt
C:\Users\matthes\git\aixm-bindings-sharp>xsd.exe xsd\gml\3.2.1\gml.xsd xsd\xlink\1
.0\xlink.xsd xsd\iso\19139\20070417\gmd\gmd.xsd xsd\iso\19139\20070417\gco\gco.x
sd xsd\iso\19139\20070417\gss\gss.xsd xsd\iso\19139\20070417\gts\gts.xsd xsd\iso\1
9139\20070417\ger\ger.xsd xsd\aixm\5.1\AIXM_Features.xsd /classes /namespace:aero.
aixm.u51
Microsoft (R) Xml Schemas/DataTypes support utility
[Microsoft (R) .NET Framework, Version 2.0.50727.3038]
Copyright (C) Microsoft Corporation. All rights reserved.
Writing file 'C:\Users\matthes\git\aixm-bindings-sharp\AIXM_Features.cs'.

C:\Users\matthes\git\aixm-bindings-sharp>_

```

Figure 1: Visual Studio Command Prompt - Binding Generation.

6 Binding Generation Approach

The approach on how the binding generation has been realized is similar for both development environments. The starting points are the officially available schemas as instances of XML Schema Definitions (XSD):

- AIXM 5.1: http://www.aixm.aero/public/standard_page/download.html (version date: 2010-02-01)
- GML 3.2.1: <http://schemas.opengis.net/gml/3.2.1/> (version date: 2012-07-21)

Based on these, environment-specific changes might be applied in order to fulfill the schema binding generation. The following sections described the required steps for JAXB and C#.

6.1 JAXB

The following sections illustrate the approach that has been implemented for the generation of JAXB bindings for AIXM 5.1.

6.1.1 Required Schema Adjustments

The JAXB Reference Implementation cannot process the original schemas as provided by EUROCONTROL and the OGC. Several adjustments need to be applied to the GML 3.2.1 as well as the AIXM 5.1 schemas. The following enumeration provides an overview of the required changes.

Table 1: Required Adjustments to Schemas for JAXB generation.

Issue	Change	Affected .xsd	Assessment
Remote schema imports result in multiple instances of the same namespace. The compiler breaks with a “Type is already defined” error message.	Replace all remote schemaLocation values with relative paths and provide referenced schemas locally ² .	GML 3.2.1, GMD, AIXM 5.1	Minor. Does not break XML instances → schemas are isomorphic.

6.1.2 Schema Generation

Schema generation can be achieved via command line tools or by using a build tool such as Apache Maven with corresponding plugins for JAXB. Both approaches are described in the following.

6.1.2.1 Command line

In order to get the command line generation running, a bit of configuration has to be done manually. The generation requires the issues described in section 6.1.1 to be addressed beforehand.

XML provides with the concept of namespaces the possibility to have multiple elements with the local name that are qualified using the namespace prefix:

```
<aixm:Airspace gml:id="test">
...
  <aixm:name>the aixm name</aixm:name>
  <gml:name>the gml name</gml:name>
...
</aixm:Airspace>
```

When generating bindings for JAXB, these two element definitions create a clash in the mapping against the methods and properties of the created binding classes. In order to work around such issues, JAXB provides the possibility to define custom mappings. An example of such mapping is illustrated in Listing 1. The command line tool takes these

² XML Catalogs did not resolve the issue as expected as the available catalog implementations were not able to resolve the schemas. Further investigation was out of scope for this report.

configuration files into account when mapping XML elements against class methods and properties.

Listing 1: Resolution of an element naming clash.

```
<jaxb:bindings xmlns:jaxb="http://java.sun.com/xml/ns/jaxb"
  xmlns:xjc="http://java.sun.com/xml/ns/jaxb/xjc"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  jaxb:extensionBindingPrefixes="xjc" version="2.1">
  <jaxb:bindings schemaLocation="../xsd/aixm/AIXM_Features.xsd"
    node="/xs:schema">
    ...
    <jaxb:bindings>
      <jaxb:bindings
        node="//xs:group[@name='AerialRefuellingPropertyGroup']
//xs:element[@name='name']">
        <jaxb:property name="AIXMName" />
      </jaxb:bindings>
    </jaxb:bindings>
    ...
  </jaxb:bindings>
</jaxb:bindings>
```

The actual execution of the binding generation can be triggered with a one line command line execution:

```
xjc src/main/xsd/aixm/AIXM_Features.xsd -b src/main/xjb -
extension -d .build
```

The schema files are located under `src/main/xsd` and the previously introduced configuration files are provided in the folder `src/main/xjb`. The result is stored in the `.build` folder and contains a single `.jar` file. This `.jar` file holds the schema bindings for AIXM 5.1 and all transient dependent schemas (such as GML 3.2.1, W3C XLink and GMD).

6.1.2.2 maven-jaxb2-plugin

Using a build tool such as Apache Maven has quiet decent benefits over the command line solution:

- the build is platform independent
- reproducibility

- existing bindings for dependent schemas (e.g. GML 3.2.1) can be reused; this can be achieved with an extension concept for JAXB, the episodes
- 3rd party dependencies are reflected in the artifact metadata (e.g. the JAXB runtime implementation)

Within the OWS-10 test bed a similar approach to the command tool schema generation has been followed. A single Maven artifact containing the bindings for AIXM 5.1 and all transient dependent schemas is the result. An excerpt of the configuration of the Maven build is provided in Listing 2.

Listing 2: Maven project configuration for JAXB bindings.

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.n52.aixm</groupId>
    <artifactId>52n-aixm-bindings</artifactId>
    <version>1.0.0-SNAPSHOT</version>
  </parent>

  <artifactId>aixm-v51-jaxb-uberBinding</artifactId>
  <description>52North JAXB bindings for AIXM 5.1 - including
GML 3.2.1, GMD and XLink</description>

  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
      </plugin>
      <plugin>
        <groupId>org.jvnet.jaxb2.maven2</groupId>
        <artifactId>maven-jaxb2-plugin</artifactId>
        <configuration>
          <schemaIncludes>
            <value>aixm/AIXM_Features.xsd</value>

            <value>aixm/message/AIXM_BasicMessage.xsd</value>
            <value>aixm/event/Event_Features.xsd</value>
          </schemaIncludes>
        </configuration>
      </plugin>
    </plugins>
  </build>

  <dependencies>
```

```

    <dependency>
      <groupId>com.sun.xml.bind</groupId>
      <artifactId>jaxb-impl</artifactId>
    </dependency>
  </dependencies>
</project>

```

6.1.3 Roundtrip Execution

The execution of a roundtrip within a Java program is a rather simple task in comparison to the creation of the bindings. Listing 3 illustrates the basic workflow of a roundtrip. An `Unmarshaller` object is created from the `JAXBContext` for the given package (as defined in the schema generation). This `Unmarshaller` is used to convert a data stream into the Java representation of a `Navaid` document. Subsequently, a `Marshaller` object is created in the same in order to convert the Java object back to an XML representation. The example uses a stream-based approach. JAXB also supports the processing of files or string representations of an XML document.

Listing 3: Roundtrip using Java with JAXB.

```

//imports of all required classes

//executed within an exemplary main method

JAXBContext jc = JAXBContext.newInstance("aero.aixm.v510");
Unmarshaller unmarshaller = jc.createUnmarshaller();

NavaidType navaid = (NavaidType)
    unmarshaller.unmarshal(
        getClass().getResourceAsStream("navaid.xml"));

Marshaller marshaller = jc.createMarshaller();
marshaller.marshal(navaid, resolveTargetOutputStream());

```

A sophisticated approach using Java Generics providing a set of convenience methods for dealing with different input and output types has also been designed and implemented. It is available within a source code project hosted at the 52°North GitHub repositories³.

³ <https://github.com/52North/aixm-bindings>.

6.1.3.1 Performance

In order to assess the JAXB roundtrip executions, a basic performance test has been implemented. The test iterates over a set of source files with different AIXM feature documents (Navaid, Airspace, AirportHeliport and RouteSegment) and measures the time delta required for doing a single roundtrip. Figure 2 and Figure 3 illustrate the result of the test execution. The peaks in the beginning of the time series can be related to the fact that the Java Virtual Machine has to load the binding classes into memory.

The first figure shows the results for a rather small document (one Feature with one TimeSlice), the second illustrates results for bigger documents (one Feature with 100 TimeSlices).

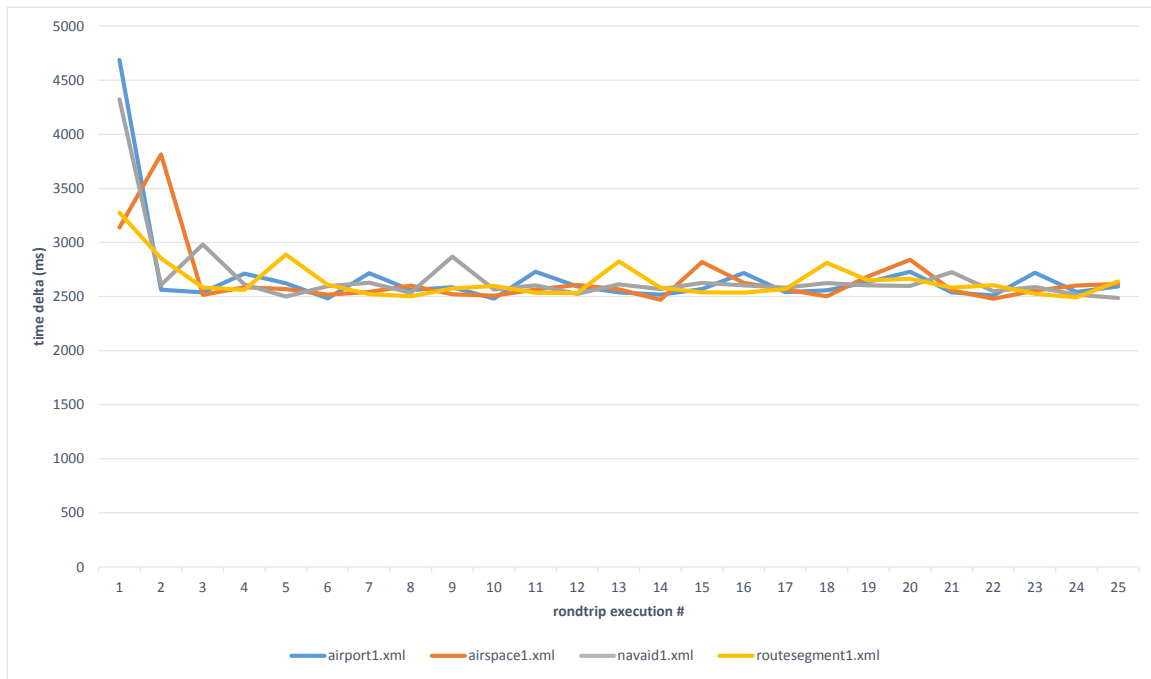


Figure 2: Roundtrip performance for JAXB bindings (one feature with one timeslice).

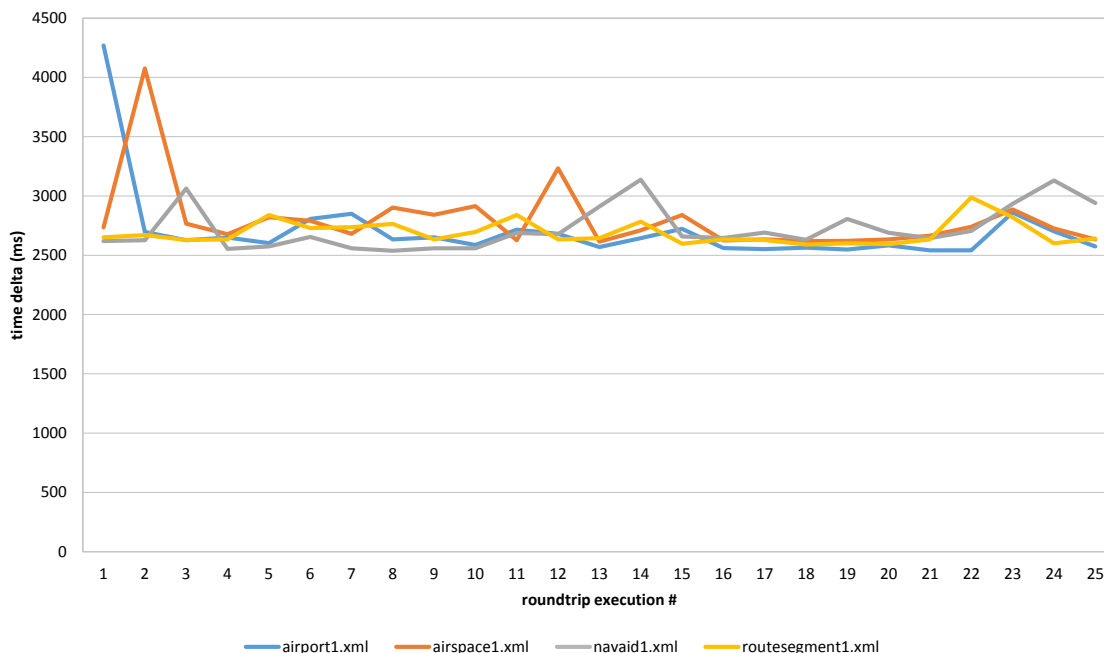


Figure 3: Roundtrip performance for JAXB bindings (one feature with 100 timeslices).

6.1.3.2 Web Processing Service Interface

The JAXB roundtrip method has been deployed as a WPS process. A demo instance is available at:

<http://ows.dev.52north.org:8080/wps/WebProcessingService?Request=DescribeProcess&Service=WPS&version=1.0.0&identifier=org.n52.aixm.roundtrip.wps.AIXMRoundtripAlgorithm>

Listing 4 provides an exemplary request for the Execute operation.

Listing 4: Exemplary WPS Roundtrip Execution.

```
<wps:Execute service="WPS" version="1.0.0"
xmlns:wps="http://www.opengis.net/wps/1.0.0"
xmlns:ows="http://www.opengis.net/ows/1.1"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/wps/1.0.0
  http://schemas.opengis.net/wps/1.0.0/wpsExecute_request.xsd">
  <ows:Identifier>
org.n52.aixm.roundtrip.wps.AIXMRoundtripAlgorithm</ows:Identifier>

  <wps>DataInputs>
    <wps:Input>
      <ows:Identifier
```

```

xmlns:ns1="http://www.opengis.net/ows/1.1">input</ows:Identifier>
  <wps:Data>
    <wps:ComplexData mimeType="application/xml">
      <inlinedAIXMFeature/>
    </wps:ComplexData>
  </wps:Data>
</wps:Input>
</wps>DataInputs>

<wps:ResponseForm>
  <wps:ResponseDocument>
    <wps:Output mimeType="application/xml" encoding="UTF-8">
      <ows:Identifier>output</ows:Identifier>
    </wps:Output>
  </wps:ResponseDocument>
</wps:ResponseForm>
</wps:Execute>

```

6.1.3.3 Issues

The following issues have been identified as a results of comparing the input and output documents of the roundtrip execution.

Table 2: Roundtrip Issues for JAXB bindings.

Issue	Available Workaround	Assessment
Some undefined objects (XLink attributes, gml:boundedBy) are added in the resulting XML document as empty elements or default attributes. An example is documented in Annex A.	Configuration in the JAXB Marshaller might overcome this issue. Investigation was out of scope for this report.	The empty elements do not break the semantics of the document. Uncritical issue.

6.1.4 Language Integration

The integration into the Java language is sophisticated. The generated code makes use of valuable concepts such as Annotations and Generics. Still accessing the information of the XML instance requires to follow some common patterns and possible gaps. For instance, accessing optional elements of an XML schema should be done with a certain set of `if`-statements, ensuring that no unexpected exception is thrown. An example of such pattern is illustrated in Listing 5.

6.1.5 Event Service Integration

The developed JAXB AIXM bindings have been integrated into the data processing layer of the 52°North Event Service. It is a fundamental pylon in the general OWS-10 Aviation service architecture providing publish/subscribe functionality to interested consumers (see [OGC 14-008] for further details).

When providing advanced subscription functionality such as spatial filtering the Event Service has to rely on robust data parsing components that are able to extract specific information from an XML document. The JAXB bindings fit well into such a component as they provide straight-forward access to properties of an XML instances. Listing 5 demonstrates how to access the information of a Navaid feature.

Listing 5: Extracting information from a Navaid feature.

```
List<NavaidTimeSlicePropertyType> slice = navaid.getTimeSlice();
String identifier = navaid.getIdentifier().getValue();

if (slice != null && !slice.isEmpty() &&
    slice.get(0).isSetNavaidTimeSlice()) {
    NavaidTimeSliceType timeSlice = slice.get(0).getNavaidTimeSlice();

    NavaidExtensionType extension = (NavaidExtensionType)
        timeSlice.getExtension().get(0).getAbstractNavaidExtension().getValue();

    String interpretation = timeSlice.getInterpretation();
}
```

6.1.6 Source Code

The work on JAXB binding generation lead to a source code project which allows the easy reproduction of the schema bindings with different tools. A simple Maven command (“mvn install” on sub-module “uberBinding”) creates a binary containing the AIXM 5.1 JAXB bindings. The source code is hosted at the 52°North GitHub repository:

<https://github.com/52North/aixm-bindings>

6.2 .NET Framework

The following sections illustrate the approach that has been implemented for the generation of C# bindings for AIXM 5.1.

6.2.1 Required Schema Adjustments

The xsd.exe schema compiler of the .NET Framework cannot process the original schemas as provided by EUROCONTROL and the OGC. Several adjustments need to be applied to the GML 3.2.1 schemas. The following enumeration provides an overview of the required changes.

Table 3: Required Adjustments for C# generation.

Issues	Change	Affected .xsd	Assessment
XML type lists (e.g. xs:double list) are not supported by xsd.exe.	Change type of datatype to xs:string.	GML 3.2.1	Critical. Break of semantics. Requires additional (manual) datatype conversions
Cyclic dependency includes break binding generation.	E.g. replace <include schemaLocation="gml.xsd" /> with the actual required sub-schemas.	GML 3.2.1	Minor. Does not break XML instances → schemas are isomorphic.

6.2.2 Schema Generation

Similar to the command line execution of JAXB all dependent XML schemas have to be provided locally so the xsd.exe tool is able to resolve the links. After applying the required adjustments to the schemas as described in the previous section, the actual execution of the binding generation can be triggered with a one line command line execution.

```
xsd.exe xsd\gml\3.2.1\gml.xsd xsd\xlink\1.0.0\xlink.xsd
      xsd\iso\19139\20070417\gmd\gmd.xsd
      xsd\iso\19139\20070417\gco\gco.xsd
      xsd\iso\19139\20070417\gss\gss.xsd
      xsd\iso\19139\20070417\gts\gts.xsd
      xsd\iso\19139\20070417\gsr\gsr.xsd
      xsd\xaixm\5.1\AIXM_Features.xsd /classes
      /namespace:aero.aixm.v51
```

The schema files are located under xsd. A target namespace can be defined. The result is one big C# source file that holds the schema bindings for AIXM 5.1 and all transient dependent schemas (such as GML 3.2.1, W3C XLink, GMD).

6.2.3 Roundtrip Execution

Deserialization and serialization is very similar to the implementation with JAXB. An `XmlSerializer` object with the specific AIXM feature type defined is created. This object is then used to deserialize the XML document into the C# object instance. Subsequent the object is again serialized into an XML representation. Again, this example uses streams (i.e. file based streams). The C# `XmlSerializer` also is capable of processing string representations.

Listing 6: Roundtrip Execution using C#.

```

XmlSerializer deserial = new XmlSerializer(typeof(NavaidType));

FileStream readFileStream = new
FileStream(@"C:\ows10\navaid.xml", FileMode.Open,
FileAccess.Read, FileShare.Read);
TextWriter writeFileStream = new StreamWriter(@"C:\ows10\navaid-
roundtripped.xml");

//the (un)marshalling
NavaidType navaid = (NavaidType)
deserial.Deserialize(readFileStream);
deserial.Serialize(writeFileStream, navaid);

```

6.2.3.1 Performance

Similar to the JAXB roundtrip execution a similar basic test has been implemented to assess the performance of the C# AIXM bindings. Figure 4 and Figure 5 illustrate the results. The roundtrip execution shows a very high performance in the range of a few milliseconds, probably due to memory optimizations within the .NET framework runtime.

The first figure shows the results for a rather small document (one Feature with one TimeSlice), the second illustrates results for bigger documents (one Feature with 100 TimeSlices).

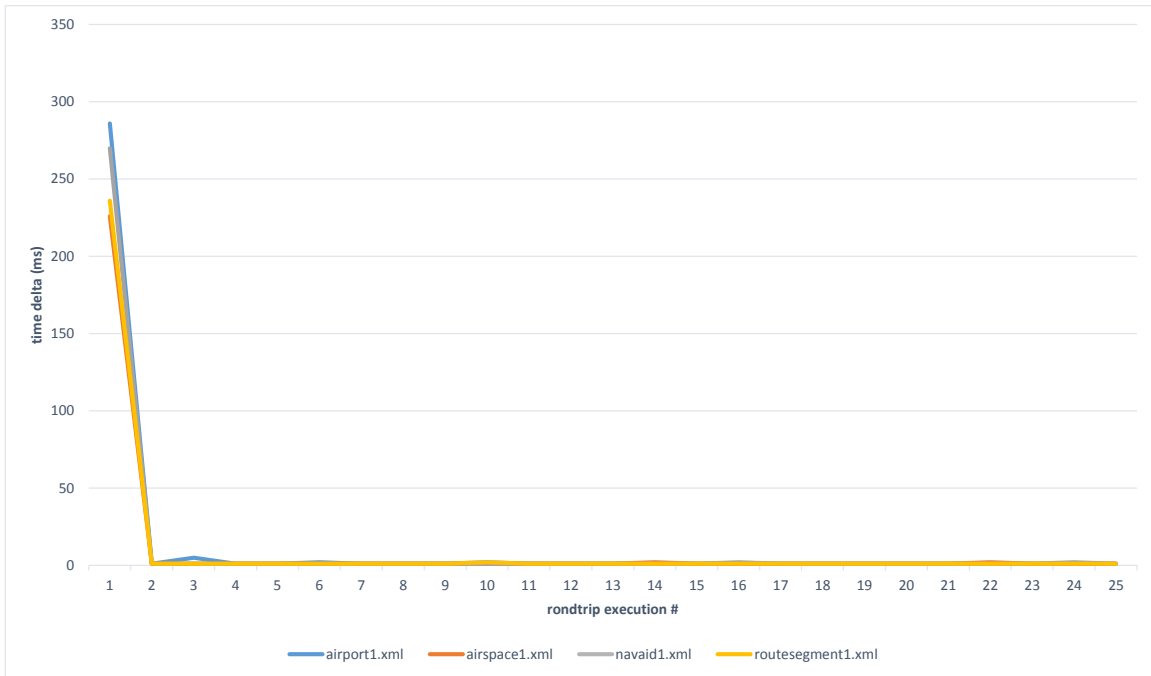


Figure 4: Roundtrip performance for C# bindings (one feature with one timeslice).

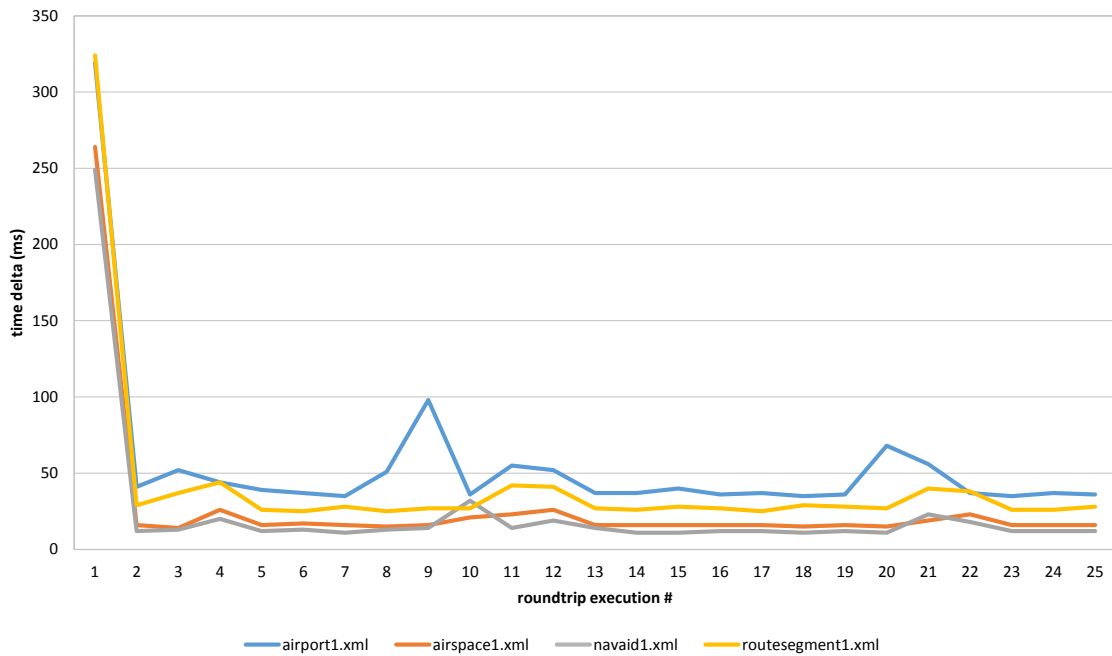


Figure 5: Roundtrip performance for C# bindings (one feature with 100 timeslices).

6.2.3.2 Issues

The following issues have been identified as a results of comparing the input and output documents of the roundtrip execution.

Table 4: Roundtrip Issues for C# bindings.

Issue	Available Workaround	Assessment
AbstractTimeGeometricPrimitive types are not (de)serialized.	N/A	Critical. Breaks most temporal functionality of XML instances.

6.2.4 Language Integration

The integration into the concepts of C# is done in a similar way as with JAXB. Accessing elements can be done very straightforward. Still, the same patterns as noted in section 6.1.4 must be taken care of, especially when dealing with a kind of lax schema as AIXM 5.1 is (many optional elements).

6.2.5 Source Code

The work on C# binding generation for the .NET Framework lead to a source code project which contains the bindings for AIXM 5 (class file “AIXM_Features.cs”). Additionally, it allows the easy reproduction of the schema bindings with a convenience command line script. The source code is hosted at the 52°North GitHub repository:

<https://github.com/52North/aixm-bindings-sharp>

Besides the actual schema binding generation exemplary implementations for apply the bindings have been developed. Similar to the JAXB source code project this lead to some convenience classes which abstract from the C# XML API specifics. Listing 7 demonstrates how to access the `interpretation` element of a `NavaidTimeSlice`.

Listing 7: Resolving the interpretation value of a NavaidTimeSlice.

```
GenericRoundtrip<NavaidType> rt =
    new GenericRoundtrip<NavaidType>();

NavaidType navaid = rt.DeSerialize(
    GenericRoundtrip<NavaidType>.ReadFileContents(
        "C:\\ows10\\navaid.xml"));

interpretation intr =
navaid.timeSlice[0].NavaidTimeSlice.interpretation;

Console.WriteLine(intr.ToString());
```


7 General Assessments

The creation of bindings for both development environments requires some adjustments to the available AIXM and GML schema documents. Albeit of these the created bindings are still compatible with existing AIXM 5.1 data as the changes are only to be applied on a technical level and not on the actual model of AIXM 5.1

Once the bindings are available the integration into the programming language makes it easy to access the properties of the features. The task of deserialization and serialization can be made a little more user friendly whenever a developer can expect to receive AIXM data only. Within this test bed some convenience methods have been developed within wrapper components. The source code is available as Open Source projects.

The performance of the actual roundtrip differs a significantly in both development environments. While the JAXB performance is still at an acceptable level, the C# realization shows very fast response times.

8 Accomplishments

During the work carried out on binding AIXM to development tools several accomplishments have been reached:

- generation of JAXB bindings
- generation of .NET C# bindings
- source code (in both environments) for
 - the generation data bindings
 - the execution of roundtrips (also incorporating a web-based approach using a WPS process)
 - providing some convenience methods when dealing with AIXM feature documents
- identification of compiler compatibility and required changes to schema
- basic performance assessment of roundtrips
- identification of issues with the contents of roundtrip executions.

9 Lessons Learned

The issues identified on the creation of data bindings for two different development environments highlights the need for pre-compiled data bindings of AIXM. Several

environment specific configurations were required in order to create reasonable binaries. Documentation of these is crucial in order to create a reusable workflow. As a result it was inevitable to host all programming and scripting efforts on an Open Source platform (i.e. GitHub).

10 Recommendations

During the work on this report, several aspects have been collected that should be taken into account when continuing the work on data bindings for AIXM:

- Provision of AIXM data bindings for Java and C# in alignment with future AIXM releases
- Besides the assessed two development environments several other solutions exist for the generation of data bindings. As these might provide better results, an assessment would be valuable
- investigation on the issues of the C# bindings with the time related elements of GML.

Summarizing, the provision of bindings in parallel to schemas would promote the use as binding creation can be cumbersome and is therefore recommended.

Annex A

XML Instances

JAXB Roundtrip – Original Document

```

<?xml version="1.0" encoding="UTF-8"?>
<aixm:Navaid xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:message="http://www.aixm.aero/schema/5.1/message"
  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:gstr="http://www.isotc211.org/2005/gstr" xmlns:aixm="http://www.aixm.aero/schema/5.1"
  xmlns:event="http://www.aixm.aero/schema/5.1/event" xmlns:aimsaa="urn:us:gov:dot:faa:aim:saa:5.1"
  xmlns:aimsua="urn:us:gov:dot:faa:aim:saa:sua:5.1"
  xsi:schemaLocation="http://www.aixm.aero/schema/5.1/message http://www.aixm.aero/schema/5.1/message/AIXM_BasicMessage.xsd
  http://www.aixm.aero/schema/5.1/event http://www.aixm.aero/schema/5.1/event/Event_Features.xsd"
  gml:id="urn-x.ows7.snowflake.amdb.navaid.fid-20bf8b95_127734f43d9_7e46">
  <gml:identifier codeSpace="faa_amdb_feature_id">539184F438BF20211D50BCE6F6C0093B8
  </gml:identifier>
  <gml:boundedBy>
  <gml:Envelope srsName="urn:ogc:def:crs:OGC:1.3:CRS84">
  <gml:lowerCorner>-71.41890556 41.60551087</gml:lowerCorner>
  <gml:upperCorner>-71.41890556 41.60551087</gml:upperCorner>
  </gml:Envelope>
  </gml:boundedBy>
  <aixm:featureMetadata>
  <gmd:MD_Metadata>
  <gmd:contact gco:nilReason="inapplicable" />
  <gmd:dateStamp>
  <gco:DateTime>2011-01-12T10:45:07.000Z</gco:DateTime>
  </gmd:dateStamp>
  <gmd:identificationInfo gco:nilReason="inapplicable" />
  </gmd:MD_Metadata>
  </aixm:featureMetadata>
  <aixm:timeslices>
  <aixm:NavaidTimeSlice
    gml:id="urn-x.ows7.snowflake.ts.amdb.navaid.fid-20bf8b95_127734f43d9_7e46">
    <gml:validTime>
    <gml:TimePeriod
      gml:id="urn-x.ows7.snowflake.tp.amdb.navaid.fid-20bf8b95_127734f43d9_7e46">
      <gml:beginPosition>2011-01-13T12:00:00.000Z</gml:beginPosition>
      <gml:endPosition indeterminatePosition="unknown" />
      </gml:TimePeriod>
    </gml:validTime>
    <aixm:interpretation>PERMDelta</aixm:interpretation>
    <aixm:sequenceNumber>1</aixm:sequenceNumber>
    <aixm:featureLifetime>
    <gml:TimePeriod
      gml:id="urn-x.ows7.snowflake.ft.amdb.navaid.fid-20bf8b95_127734f43d9_7e46">
      <gml:beginPosition>2011-01-13T12:00:00.000Z</gml:beginPosition>
      <gml:endPosition indeterminatePosition="unknown" />
      </gml:TimePeriod>
    </aixm:featureLifetime>
    <aixm:type>OTHER:Unknown</aixm:type>
    <aixm:designator>K0QU</aixm:designator>
    <aixm:name>PAPI</aixm:name>
    <aixm:location>
    <aixm:ElevatedPoint
      gml:id="urn-x.ows7.snowflake.geom.amdb.navaid.fid-20bf8b95_127734f43d9_7e46"
      srsName="urn:ogc:def:crs:OGC:1.3:CRS84">
      <gml:pos>-71.41890556 41.60551087</gml:pos>
      </aixm:ElevatedPoint>
    </aixm:location>
    <aixm:servedAirport xlink:href="#urn-x.ows7.snowflake.nasr_arp.14598" />
    <aixm:availability>
    <aixm:NavaidOperationalStatus
      gml:id="urn-x.ows7.snowflake.nos.amdb.navaid.fid-20bf8b95_127734f43d9_7e46">
      <aixm:timeInterval>
      <aixm:Timesheet
        gml:id="urn-x.ows7.snowflake.timesheet.amdb.navaid.fid-20bf8b95_127734f43d9_7e46">
        <aixm:timeReference>OTHER</aixm:timeReference>
        <aixm:startDate>01-01</aixm:startDate>
        <aixm:endDate>31-12</aixm:endDate>
        <aixm:day>ANY</aixm:day>
        <aixm:startTime>00:00</aixm:startTime>
        <aixm:endTime>24:00</aixm:endTime>
      </aixm:Timesheet>
      </aixm:timeInterval>
      <aixm:annotation>
      <aixm:Note
        gml:id="urn-x.ows7.snowflake.an.amdb.navaid.fid-20bf8b95_127734f43d9_7e46">
        <aixm:translatedNote>
        <aixm:LinguisticNote
          gml:id="urn-x.ows7.snowflake.aln.amdb.navaid.fid-20bf8b95_127734f43d9_7e46">
          <aixm:note lang="Eng">operational continuously</aixm:note>
          </aixm:LinguisticNote>
        </aixm:translatedNote>
      </aixm:Note>
      </aixm:annotation>
      <aixm:operationalStatus>OPERATIONAL</aixm:operationalStatus>
    </aixm:NavaidOperationalStatus>
    </aixm:availability>
  </aixm:NavaidTimeSlice>
</aixm:timeslices>
</aixm:Navaid>

```

JAXB Roundtrip – Resulting Document

Note the added xlink:type attributes.

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<aixm:Navaid xmlns:gml="http://www.opengis.net/gml/3.2"
  xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:aixm="http://www.aixm.aero/schema/5.1"
  xmlns:gco="http://www.isotc211.org/2005/gco" xmlns:gmd="http://www.isotc211.org/2005/gmd"
  xmlns:gts="http://www.isotc211.org/2005/gts" xmlns:event="http://www.aixm.aero/schema/5.1/event"
  xmlns:message="http://www.aixm.aero/schema/5.1/message"
  gml:id="urn-x.ows7.snowflake.amdb_navaid.fid-20bf8b95_127734f43d9_7e46">
  <gml:identifier codeSpace="faa_amdb_feature_id">5391B4F438BF2021D50BCEF6C0093B8
  </gml:identifier>
  <gml:boundedBy>
    <gml:Envelope srsName="urn:ogc:def:crs:OGC:1.3:CRS84">
      <gml:lowerCorner>-71.41890556 41.60551087</gml:lowerCorner>
      <gml:upperCorner>-71.41890556 41.60551087</gml:upperCorner>
    </gml:Envelope>
  </gml:boundedBy>
  <aixm:featureMetadata xlink:type="simple">
    <gmd:MD_Metadata>
      <gmd:contact gco:nilReason="inapplicable" xlink:type="simple" />
      <gmd:dateStamp>
        <gco:Date xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:nil="true" />
        <gco:DateTime>2011-01-12T10:45:07.000Z</gco:DateTime>
      </gmd:dateStamp>
      <gmd:identificationInfo gco:nilReason="inapplicable"
        xlink:type="simple" />
    </gmd:MD_Metadata>
  </aixm:featureMetadata>
  <aixm:timeslice>
    <aixm:NavaidTimeslice
      gml:id="urn-x.ows7.snowflake.ts.amdb_navaid.fid-20bf8b95_127734f43d9_7e46">
      <gml:validTime xlink:type="simple">
        <gml:TimePeriod
          gml:id="urn-x.ows7.snowflake.tp.amdb_navaid.fid-20bf8b95_127734f43d9_7e46">
          <gml:beginPosition>2011-01-13T12:00:00</gml:beginPosition>
          <gml:endPosition indeterminatePosition="unknown"></gml:endPosition>
        </gml:TimePeriod>
      </gml:validTime>
      <aixm:interpretation>PERMDelta</aixm:interpretation>
      <aixm:sequenceNumber>1</aixm:sequenceNumber>
      <aixm:featureLifetime xlink:type="simple">
        <gml:TimePeriod
          gml:id="urn-x.ows7.snowflake.ft.amdb_navaid.fid-20bf8b95_127734f43d9_7e46">
          <gml:beginPosition>2011-01-13T12:00:00.000Z</gml:beginPosition>
          <gml:endPosition indeterminatePosition="unknown"></gml:endPosition>
        </gml:TimePeriod>
      </aixm:featureLifetime>
      <aixm:type>OTHER:Unknown</aixm:type>
      <aixm:designator>K0QU</aixm:designator>
      <aixm:name>PAPI</aixm:name>
      <aixm:location>
        <aixm:ElevatedPoint srsName="urn:ogc:def:crs:OGC:1.3:CRS84"
          gml:id="urn-x.ows7.snowflake.geom.amdb_navaid.fid-20bf8b95_127734f43d9_7e46">
          <gml:pos>-71.41890556 41.60551087</gml:pos>
        </aixm:ElevatedPoint>
      </aixm:location>
      <aixm:servedAirport xlink:type="simple"
        xlink:href="#urn-x.ows7.snowflake.nasr_arp.14598" />
      <aixm:availability>
        <aixm:NavaidOperationalStatus
          gml:id="urn-x.ows7.snowflake.nos.amdb_navaid.fid-20bf8b95_127734f43d9_7e46">
          <aixm:TimeInterval>
            <aixm:Timesheet
              gml:id="urn-x.ows7.snowflake.timesheet.amdb_navaid.fid-20bf8b95_127734f43d9_7e46">
              <aixm:timeReference>OTHER</aixm:timeReference>
              <aixm:startDate>01-01</aixm:startDate>
              <aixm:endDate>31-12</aixm:endDate>
              <aixm:day>ANY</aixm:day>
              <aixm:startTime>00:00</aixm:startTime>
              <aixm:endTime>24:00</aixm:endTime>
            </aixm:Timesheet>
          </aixm:TimeInterval>
          <aixm:annotation>
            <aixm:Note
              gml:id="urn-x.ows7.snowflake.an.amdb_navaid.fid-20bf8b95_127734f43d9_7e46">
              <aixm:translatedNote>
                <aixm:LinguisticNote
                  gml:id="urn-x.ows7.snowflake.a1n.amdb_navaid.fid-20bf8b95_127734f43d9_7e46">
                  <aixm:note lang="Eng">operational continuously</aixm:note>
                </aixm:LinguisticNote>
              </aixm:translatedNote>
            </aixm:Note>
          </aixm:annotation>
          <aixm:operationalStatus>OPERATIONAL</aixm:operationalStatus>
        </aixm:NavaidOperationalStatus>
      </aixm:availability>
    </aixm:NavaidTimeslice>
  </aixm:timeslice>
</aixm:Navaid>

```