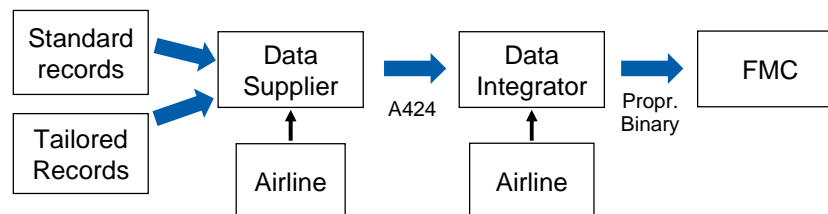


The ARINC 424A big picture and history of developments .....	1
ARINC 424 concept.....	1
NDBX concept.....	2
The ARINC 424A concept.....	3
ARINC 424A Output Formats .....	3
The ARINC 424A Big Picture.....	4
UML Modelling principles .....	5
Overview of the UML model.....	5
Overview of the ASCII Format.....	5
Overview of the XML-based formats .....	5
How to read the UML model - Example TBD .....	<b>Error! Bookmark not defined.</b>
Guidance on how to validate the UML model.....	<b>Error! Bookmark not defined.</b>
Guidance on how to validate the ASCII format.....	10
Guidance on how to validate the XML formats.....	<b>Error! Bookmark not defined.</b>
List of A424A resources .....	<b>Error! Bookmark not defined.</b>

## The ARINC 424A big picture and history of developments

### *ARINC 424 concept*

Navigation data are assembled from public sources like AIPs and airline specific, tailored re-records (e.g. company routes) by data suppliers (Figure 1). Customer defined extract parameters like the coverage area are used by data suppliers to generate airline navigation data files.



**Figure 1 ARINC 424 data chain**

The data format and encoding of these navigation data files has been standardized in the ARINC 424 specification “Navigation System Database” in 1975 [1]. The reason for the standardization was to avoid high costs for the industry to support several different formats. ARINC 424 databases are primarily used in Flight Management Computers (FMCs) as a basis of the flight plan, but also in other applications like computer flight planning systems or flight simulators.

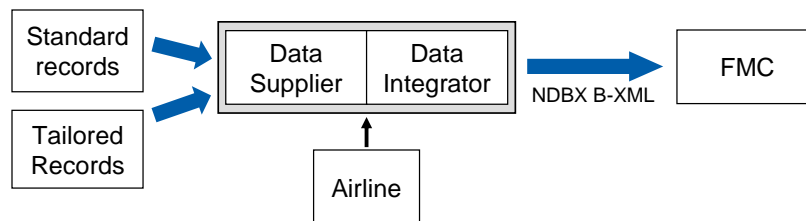
The final step in the ARINC 424 data chain is the packaging of navigation data into the FMC binary loadable image. The format of this image may be different for each vendor and FMC type. It has intentionally not been standardized in order to provide FMC manufacturers the ability to optimize their hardware and software designs for performance – which was a necessary step during the early years of FMS development.

The drawback of this approach is that many airlines with large mixed fleets and different avionics vendors have to deal with dozens of navigation database packages, though the content is always the same.

### ***NDBX concept***

The NDBX project “Navigation Data Base Open Standard” was started in 2006 by the AEEC through the initiative of Lufthansa German Airlines. The working group was attended by representatives of airlines, aircraft manufacturers, FMC suppliers, data suppliers, and government institutions.

The main objective of NDBX is the definition of an open standard database which is directly read-able by the FMC, not needing the separate conversion step to a specific FMC machine read-able/loadable format. This effectively combines the data supplier and data integrator role into one and creates a single point of contact for airlines (Figure 6).



**Figure 2 NDBX data chain**

The motivation for this approach is the logistics effort and associated costs to maintain, distribute and deliver dozens of different navigation data files to the designated airplanes even though the content of all these files is identical. This is mainly targeted for future FMCs.

A side effect of this concept is a better separation between the FMC and the navigation database, therefore making the data more available to other airborne applications as well.

The previous A424 working group decision not to pursue standardization of the FMC loadable format in order to support optimization of software and hardware is viewed as being obsolete today due to the advances in both avionics hardware and soft-ware. Other goals of NDBX are to support potential new requirements of modern flight deck technologies in a data driven world such as:

- delta/short loading,
- use of data-link,
- avionics database servers,
- off AIRAC cycle data delivery, and
- Integrated Modular Avionics.

### **NDBX content**

The basic approach of the NDBX project was to use modern data modeling approaches (UML) and data formats (XML). The XML format is converted into Binary-XML (BXML) [2] for smaller file sizes and easier parsing. It also contains indices for faster random access. The UML modeling of navigation data and mapping

to XML files was derived from the Eurocontrol AIXM 5.0 project [3]. The content definition is copied from ARINC 424 field definitions. Attribute coding was adapted where necessary, e.g. bearings are stored as float values (“359.4”) instead of a simple text string with decimal point suppressed (“3594”).

A major difference from the current ARINC 424 is the fact that NDBX records are restricted to data required by the FMC. Flight planning or simulation records are not covered, neither are fields used to-day only by the packaging software for data organization. The goal behind this decision is to minimize the NDBX package sizes. A practical problem of this approach is that FMC vendors have differing requirements of what is “FMC required data”, and an agreement is still under discussion.

### ***The ARINC 424A concept***

Comparing NDBX and ARINC 424 reveals that the content is basically the same, but as stated previously, NDBX does not contain all of the fields and records of 424. Additionally, coding and especially the format for FMCs are vendor specific to-day. During the development of NDBX a major concern was to ensure that NDBX did not create a second navigation data exchange standard with many identical and a few different elements compared to 424. This would have imposed unnecessary effort on the industry to maintain two parallel lines of documentation and development for the same data.

In order to avoid these issues and at the same time utilize the advantages of the both concepts, AEEC has made a decision to merge the 424 and the NDBX standards. This effort is in progress under the working title ARINC 424A “Navigation Database Standards”. The first release is scheduled for early 2010.

ARINC 424A will be based on a common UML model for all navigation data. This model will hold all existing specifications, documentation, and requirements from the existing ARINC 424, as well as specifications from NDBX. Automated scripts are used to derive schemas of the output formats and related documentation. The UML model is de-scribed in detail in the chapter “424A UML Data Model”.

Enterprise Architect was used for the development of the UML model. An HTML version of the model is available as well for easy access to the model without the use of the Enterprise Architect software.

### ***ARINC 424A Output Formats***

The first two output formats derived from the 424A UML model are an ASCII representation (identical to the existing ARINC 424), and a BXML representation (the NDBX format described above).

A third variant attracted attention in discussions with the NDB committee: an XML format containing the full 424 content instead of only the FMC required data as in the NDBX format.

This format could be used for flight planning or simulation as well, and it can replace today’s 424 ASCII files between data suppliers and data integrators. An advantage of this 424 XML format is the easy extendibility with new elements and attributes compared to problems encountered today when extending the fixed length ASCII records.

This enables three variations of data flow from the data supplier to the FMC (Figure 8).

1. The data supplier provides navigation data in ASCII format, which is converted by the data integrator into the specific format for the FMC. This option is identical to today's data chain. No changes are required for the data supplier, data integrator or FMC.
2. The data supplier provides navigation data in XML format, which is converted by the data integrator into the vendor specific format for the FMC. This option is very similar to today's data chain, but has the advantage that additional attributes can be added much easier to the XML format than to the fixed length ASCII format, while the individual FMC loadable formats are not changed
3. The data supplier provides navigation data in BXML format which is loadable directly into FMC. Option 3 represents the NDBX approach.

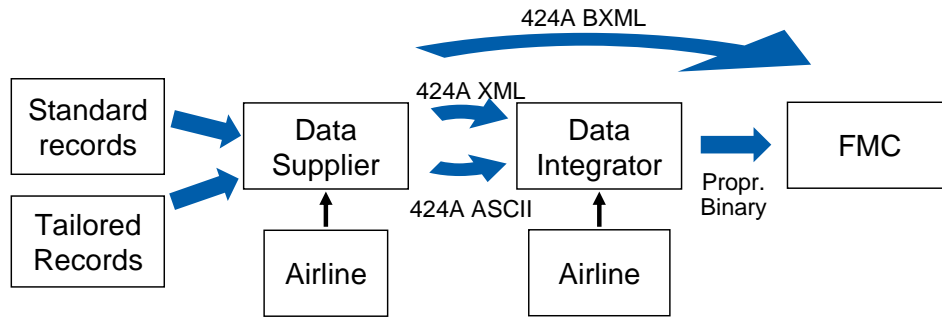
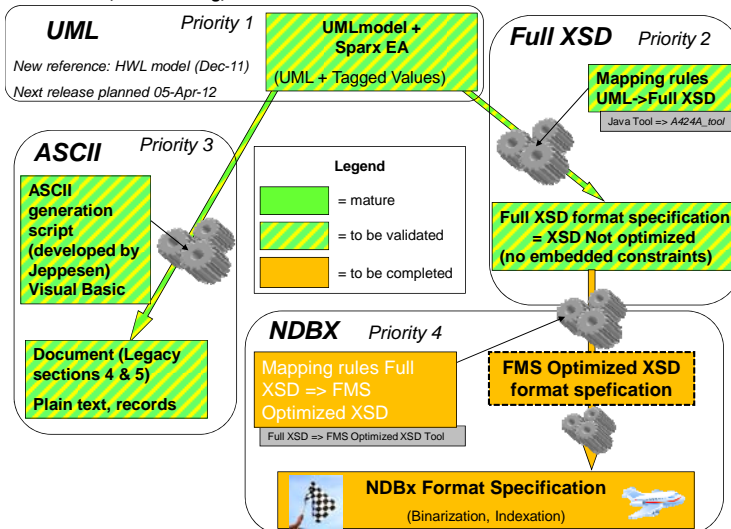


Figure 3 424A data chains

## The ARINC 424A Big Picture

New version, FRA meeting, Dec 2011



## Structure of the ARINC 424A document

### ARINC 424A: the new Table of Contents...

A424-19	A424A
<ul style="list-style-type: none"><li>• Section 1: Introduction</li><li>• Section 2: Glossary of Terms</li><li>• Section 3: Navigation Data</li><li>• <b>Section 4: Navigation Data Record Layout</b></li><li>• <b>Section 5: Navigation Data Field Definitions</b></li><li>• <b>Section 6: Encoding Standards</b></li><li>• Section 7: Naming Conventions</li><li>• Attachments<ul style="list-style-type: none"><li>▶ 1 Flow Diagram</li><li>▶ 2 Local Horizontal Reference ...</li><li>▶ 3 Navigation Data/File Data ...</li><li>▶ 4 Airway Minimum Altitudes ...</li><li>▶ 5 Path and Terminator</li></ul></li><li>• Appendixes<ul style="list-style-type: none"><li>▶ 1 Chronology and Bibliography</li><li>▶ 2 Straight-In Criteria</li><li>▶ 3 Subject Index</li></ul></li></ul>	<ul style="list-style-type: none"><li>• Section 1: Introduction</li><li>• Section 2: Glossary of Terms</li><li>• Section 3: Navigation Data</li><li>• <b>Section 4: Navigation Data Field Definitions</b></li><li>• <b>Section 5: Navigation Data ASCII Record ...</b></li><li>• <b>Section 6: Navigation Data XML Record ...</b></li><li>• <b>Section 7: Encoding Standards</b></li><li>• Section 8: Naming Conventions</li><li>• Attachments<ul style="list-style-type: none"><li>▶ 1 Flow Diagram</li><li>▶ 2 Local Horizontal Reference ...</li><li>▶ 3 Navigation Data/File Data ...</li><li>▶ 4 Airway Minimum Altitudes ...</li><li>▶ 5 Path and Terminator</li><li>▶ <b>6 UML to Full XSD Mapping Rules</b></li><li>▶ <b>7 Full XSD to FMS Optimized XSD Mapping Rules</b></li></ul></li><li>• Appendixes<ul style="list-style-type: none"><li>▶ 1 Chronology and Bibliography</li><li>▶ 2 Straight-In Criteria</li><li>▶ 3 Subject Index</li></ul></li></ul>

Page 2

© AIRBUS S.A.S. All rights reserved. Confidential and proprietary document.



## UML Modelling principles

- For the modelling principles please reference Jeppesen Processing Instruction

## Overview of the UML model

- For the modelling principles please reference Aeroconseil UML model Technical Report

## Overview of the ASCII Format

Mapping XML datatypes ⇔ A424 datatypes

*Include mapping rules UML => ASCII ?*

## Overview of the XML-based formats

*=> All*

*Rename “embedded XSD” to FMS optimised XSD.*

*Include mapping rules UML => full XML?*

## Validating the Full XSD format

The validation of the Arinc424A specification and the A424A UML model in particular can be performed by checking directly the Full XSD format, since a mistake in the UML model is likely to be reflected almost as is in the output formats. This chapter provides guidance on how to validate the A424A Full XSD format.

Validating the Full XSD schemas can be done in many different ways:

1. Review the document describing the UML to Full XSD Mapping Rules
2. Run the script that implements the mapping rules, extract the log file storing the errors spotted by the script and relate them to modelling issues in the UML
3. Check directly the content of the Full XSD schemas that are generated by the script
4. Create an A424A XML dataset compliant with the Full XSD schemas

The following input documents and tools are available to support this validation exercise:

- The document entitled *[2011.Dec.15] Mapping Rules UML to Full XSD v0.2* which explains how the various UML structures defined in the A424A UML model shall map in XML
- The script (developed in Java) implementing the UML to Full XSD mapping rules. A standalone version of this tool is available as well as the corresponding Eclipse project including the sources.
- The two XML schemas *A424A\_FULL\_XSD\_DataTypes.xsd* and *A424A\_FULL\_XSD\_Records.xsd* as generated by the script. These XML schemas are not declared valid because of several issues/inconsistencies in the UML model, that are reflected as is the schemas.
- Two additional XML schemas corresponding to the schemas generated by the script in which all the invalid lines are excluded, by putting them into simple comments. As a result, the two rectified XML schemas are declared valid and can be used as a the reference grammar to start building an XML dataset
- An example of valid XML dataset compliant with the rectified FULL XSD schemas which includes an instance of RunwayRecord and an instance of AirportRecord. This XML file can be used as a first baseline for creating additional instances of A424A records.

### **Running the script generating the Full XSD schemas from the mapping**

Here are examples of errors currently spotted by the script when generating the Full XSD grammar.

*ERROR: The UML attribute [MinimumAltitudeZone].endBearing has no tagged value REQUIRED\_BY\_FMS.*

=> This error can be included in the Excel file capturing the validation results, together with a recommendation whether this attribute should be included in the FMS Optimized XSD or not.

*ERROR: The property minimumAltitude defined for MinimumAltitudeZone is not typed properly in UML and will not be added to the Full XML format.*

=> In this case, the UML attribute shall be checked as well as the UML A424ADatatype used to type the attribute. Recommendations on how to solve the problem at the UML level can be of course proposed.

*ERROR: The multiplicity of the following association is invalid or not specified: Connector {[STARRecord] <--- [Approach] }. The multiplicity 0..1 will be used by default.*

=> In such a case, a recommended multiplicity should be included in the Excel file.

### **Checking the Full XSD schemas**

A traditional XML validation can be performed on the FULL XSD to check if the grammar itself is well formed and relevant. The schemas currently generated from the UML model contain structural deficiencies that prevent them from being declared valid. These deficiencies are most of the time due to issues in the UML model itself, but can be explained also by a bug in the generation script.

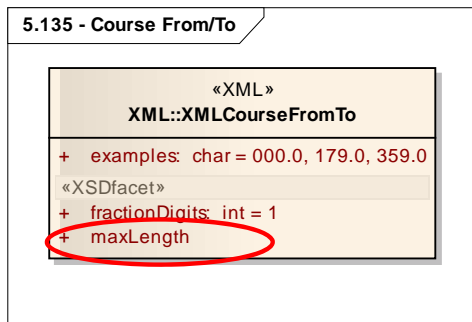
Here are examples of errors automatically spotted during a traditional XML validation.

#### **Example 1**

```
<simpleType name="XMLCourseFromTo">
  <restriction base="a424a:valAngle">
    <fractionDigits value="1"/>
    <maxLength value=""/>
  </restriction>
</simpleType>
```

**Value '' is not allowed for attribute 'value'.**

=> An empty value is simply not allowed. The problem comes from the UML model: the class XMLCoursefromTo shall be updated accordingly.

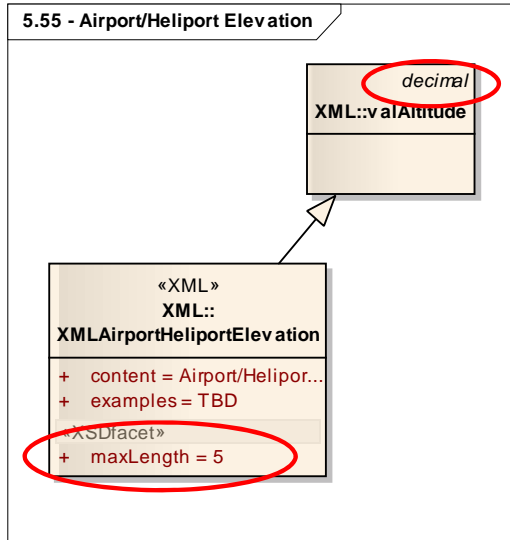


#### **Example 2**

```
<simpleType name="XMLAirportHeliportElevation">
  <restriction base="a424a:valAltitude">
    <maxLength value="5"/>
  </restriction>
</simpleType>
```

**Facet 'xsd:maxLength' is not allowed for simple type definition 'a424a:XMLAirportHeliportElevation'.**

=> The “parent” type valAltitude inherits from xsd:decimal, which means that XMLAirportHeliportElevation is also a decimal. Therefore, it can not carry the pattern maxLength which is reserved for string characters. If need be, a number of digits could be specified here. The modification shall be performed in the UML model anyway.

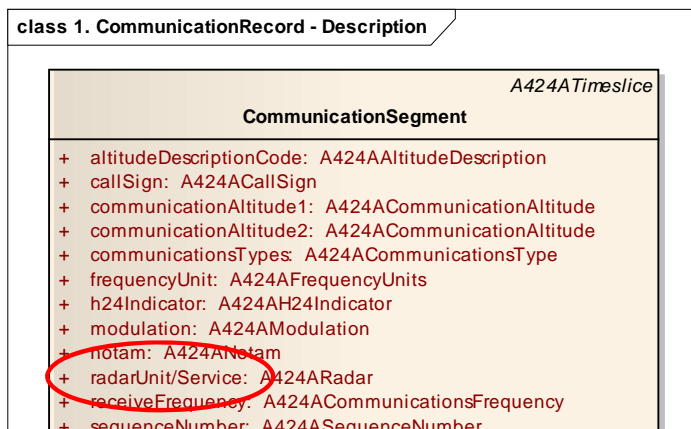


### Example 3

```
<element name="radarUnit/Service" type="a424a:XMLRadar" minOccurs="0">
  <annotation>
    <appinfo source="REQUIRED_BY_FMS">True</appinfo>
  </annotation>
</element>
```

**Value 'radarUnit/Service' is not allowed for attribute 'name'.**

=> Special characters are not allowed for element names. The name of the UML attribute should be modified to remove the character “/”.



## Creating an XML dataset compliant with the Full XSD schemas

An XML dataset is valid according to a schema if it conforms to the structure and content specified in that schema. Starting to create initial A424A Full XML dataset including instances of RunwayRecord, AirportRecord, NavaidRecord etc... compliant with the “rectified” Full XSD schemas will help to identify whether the expected characteristics of the runways, airports, navaids, etc... are correctly mapped into XML, and will also help to spot missing properties or invalid properties.

### Example

```
<a424a:A424AFULLXMLRecord xsi:type="a424a:RunwayRecordType">
  <a424a:identifier>001</a424a:identifier>
  <a424a:cycleDate>1212</a424a:cycleDate>
  <a424a:Runway>
    <a424a:customerAreaCode>EUR</a424a:customerAreaCode>
    <a424a:runwayBearingTrueIndicator>>false</a424a:runwayBearingTrueIndicator>
    <a424a:runwayDescription>Description of the Runway</a424a:runwayDescription>
    <a424a:runwayGradient>0.1</a424a:runwayGradient>
    <a424a:runwayLeftRightCentre>L</a424a:runwayLeftRightCentre>
    <a424a:runwayMagneticBearing>321</a424a:runwayMagneticBearing>
    <a424a:runwayNumber>32</a424a:runwayNumber>
    <a424a:LandingArea>
      <a424a:touchdownZoneElevation>50</a424a:touchdownZoneElevation>
    </a424a:LandingArea>
    <a424a:LTP>
      <a424a:datumCode>WGE</a424a:datumCode>
      <a424a:position>0.0 0.0</a424a:position>
    </a424a:LTP>
    <a424a:references_AirportRecord xlink:href="Link to the corresponding
AirportRecord"/>
    <a424a:references_LandingSystem>
    <a424a:references_GLSRecord xlink:href="Link to the corresponding
GLSRecord"/>
    </a424a:references_LandingSystem>
  </a424a:Runway>
</a424a:A424AFULLXMLRecord>
```

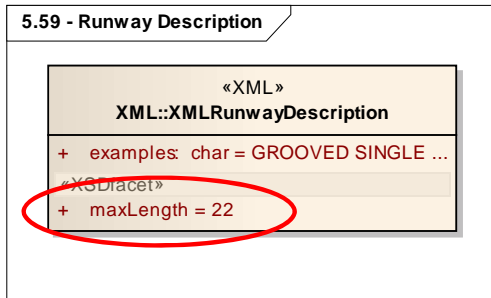
The validation of this XML data encoding of a runway against the rectified Full XSD schemas generates the following error message:

**Value 'Description of the Runway' is not allowed for element**

**<a424a:runwayDescription>.**

**Reason: Value 'Description of the Runway' violates 'maxLength' facet value '22'.**

=> The limitation to 22 characters of the runway description in the full XML is questionable (probably a copy/paste of the ASCII constraint). The recommendation in this case would be to remove the UML attribute maxLength from the class XMLRunwayDescription (no need to constraint the XML)



## Guidance on how to validate the ASCII format

In order to provide the 424-A Standard in its traditional 424 ASCII Representation, the NDBX Subgroup has developed a script to derive a text format of the UML Model. This script was contributed by Jeppesen (Christian Pschierer).

The resulting ASCII text version can be found in the A424-A Standard at Chapter 4 and 5. It is posted for your review. Please compare the 424-A ASCII Version against the 424-19 (not -20!) Grey Cover Document.

Please report any finding of divergences to the 424-19 format. Further, consider any additional needs you or your company would have for this format of the 4-24-A that you would find missing or less than ideal.