EUROPEAN ORGANISATION FOR THE SAFETY OF AIR NAVIGATION

EUROCONTROL

AIXM Primer

Edition	**:**	**4.5**
Edition Date	**:**	**20/03/2006**
Status	**:**	**Released Issue**
Class	**:**	**General Public**

EUROPEAN AIR TRAFFIC MANAGEMENT

AIXM Primer

Table of Contents

List of Figures

List of Tables

Document Identification Sheet

Table 1. DOCUMENT STATUS AND TYPE

STATUS		CLASSIFICATION	
Working Draft		General Public	✓
Draft		EATMP	
Proposed Issue		Restricted	
Released Issue	✓		

Table 2. ELECTRONIC BACKUP

INTERNAL REFERENCE NAME: AIXM_Primer_4.5.xml		
HOST SYSTEM	**MEDIA**	**SOFTWARE**
Microsoft Windows	Type: Hard disk	DocBook editor
	Media Identification:	

The following table identifies all management authorities who have successively approved the present issue of this document

Table 3. DOCUMENT APPROVAL

AUTHORITY	NAME AND SIGNATURE	DATE
Project Manager & Editor	Eduard POROSNICU	20/03/2006
Editor	Silas JONES	31/07/2005

The following table records the complete history of the successive editions of the present document.

Table 4. DOCUMENT CHANGE RECORD

EDITION	DATE	REASON FOR CHANGE	SECTIONS / PAGES AFFECTED
4.5	20/03/2006	Corrections in mapping table AICM-AIXM. Added note with regard to the fields used by the EAD for CRC calculation. Added explanation about "Obstacle in Airspace" in 2.3.	2.3, 6.3, Appendix A
4.5 - draft-2	28/11/2005	Updated mapping table AICM-AIXM	Appendix A
4.5 - draft-1	21/11/2005	Updated for AIXM 4.5	2.5, 3.1, 4, Appendix A
4.0 - draft	31/07/2005	Updated for AIXM 4.0	All
1.1	01/10/2002	Proposed Issue	All
1.0	05/06/2002	Draft Issue	All
0.5	30/05/2002	Document creation	All

Executive Summary

A primer[1] is “a small book containing basic facts about a subject, used especially when you are beginning to learn about that subject”. The AIXM Primer contains basic facts about the AIXM XML Schema - including things not described by the Schema itself - and its intended usage. It provides the essential information necessary to start an AIXM implementation.

1. Document Status

Edition 1.1 of this document was published as a Proposed Issue because the usage rules contained in the document needed to be confirmed by implementation. At the time of the publication, the most advanced AIXM implementation was through the European EAD Database (EAD) system-to-system interface (ESI).

The current edition has been updated based on feedback gathered from AIXM stakeholders since 2002.

Note that the edition number has "jumped" from 1.1 to 4.5; beginning with this version, the Primer edition number will match the version number of the AIXM to which it applies. Eventual revisions that refer to the same AIXM version will get a suffix indicating the revision number.

[1] Cambridge International Dictionary of English (http://dictionary.cambridge.org)

Chapter 1. Introduction

AIXM specifies an encoding rule that may be used for neutral aeronautical data interchange, based on the Extensible Markup Language (XML[1]). XML is a text format and the values of all data types have to be character encoded. The basic units of an XML document are *XML elements*. An element may have attributes and content. XML documents have a hierarchical structure.

The central component of the AIXM specification is the XML Schema (XSD). XML Schemas define a number of complex types, simple types and element declarations, which define the allowable structure and data instances of an XML document. As indicated in the W3C Recommendation[2] : "the purpose of a schema is to define a class of XML documents, and so the term 'instance document' is often used to describe an XML document that conforms to a particular schema. In fact, neither instances nor schemas need to exist as documents per se - they may exist as streams of bytes sent between applications, as fields in a database record, etc.". Within the scope of this document, XML documents/files that conform to the AIXM XML Schema will be referred to as *AIXM messages*.

In addition to a short presentation of the AIXM XML Schema capabilities, this document also introduces a set of usage rules. Ideally, an AIXM message that is valid against the Schema should be accepted by the receiving system without errors. However, a few operational constraints that need to be imposed on AIXM messages cannot be expressed using XML Schemas. This is particularly true for AIXM-Update messages, which allow previously exchanged data to be brought up to date without the need for reissuing a complete new data set. ~~The possibility to use XSLT stylesheets in order to check compliance of an AIXM-XML message with some of these rules will be investigated.~~

Another aspect of AIXM discussed in this document is its relationship to the AICM, the conceptual model from which the AIXM schema was derived.

[1]The XML Recommendation 1.0 was published by the World Wide Web Consortium. It is available at http://www.w3c.org/XML
[2]See “Basic Concepts” at “http://www.w3c.org/TR/xmlschema-0/#PO”

Chapter 2. AIXM and the AICM

2.1. Purpose of this Chapter

The purpose of this chapter is to explain the relationship between AIXM (the exchange model) and AICM (the conceptual model), and the similarities and differences between the two. The AICM is an entity-relationship model based on ICAO and industry standards, recommended practices, and data concepts from published aeronautical information products. It defines entities in terms of their properties and the relationships between them. It also describes data value domains (i.e. ranges) and data validation rules.

As explained in the Introduction, AIXM is an XML Schema encoding for exchanging aeronautical data. It was programmatically generated from the AICM entity-relationship data model, using a number of ad-hoc conversion rules. The following sections detail the reasons for and consequences of those rules.

It must be stressed that AIXM is only one particular implementation of the AICM - other exchange formats and implementations are possible.

2.2. Unique Identifiers

Generally, in order to utilise any dataset it is usually (though not always) necessary to be able to separately identify its constituent parts. Specifically, AIXM requires this ability to express relationships between features and to identify features that are the target of an update message. Unique identifiers associated with each feature instance are used to achieve this. By introducing unique identifiers AIXM differs from the AICM, because the conceptual model does not specify any.

Broadly speaking there are two alternatives when implementing a mechanism for uniquely identifying objects in a database, and the same is true for data formats such as AIXM: "surrogate" (artificial) or "natural" keys. The AIXM uses natural keys for the following reasons:

- The AIXM is intended for data exchange between loosely coupled systems. By not relying on artificial identifiers, which would be system specific, the internal workings of a source or destination system need not be known. This is important because data may originate from a number of sources.
- Changing an internal identifier in one system will not have an impact on other systems.
- The process of identifying the fields that constitute natural keys helped determine which AICM entities should be regarded as AIXM features, and which should be regarded as properties of features. Typically if an entity did not have an obvious natural key it was incorporated into another feature that did. Grouping entities in this way helped avoid having to create complicated "artificial natural" keys, such as it would have been the case for timesheets (for example).

Ideally, a feature's natural key would never change, but it is possible for this to happen. For example, a DME, which is uniquely identified by its position (amongst other things), could be resurveyed to a greater degree of accuracy, which would affect its natural key. The AIXM contains a mechanism for handling this aspect of natural keys - for details, see section 5.1.4.2.

The AIXM also makes provision for surrogate keys - see section 3.4. Surrogate keys may be useful in the case of data being exchanged between parties who have agreed on the form and the source of the keys, and when the values of those keys are persistent.

2.3. Feature-Entity Correspondence and Complex Properties

For every AIXM feature (cf. Chapter 3 for the definition of "AIXM feature") there exists a corresponding entity[1] in the entity-relationship data model, but the converse is not true.

Through the imposition of natural keys, it became clear that some entities should be regarded as complex properties of other entities. Within the AIXM Schema, such entities have been implemented as complex local elements of the appropriate feature type.

For example, the VOR_TIMESHEET entity corresponds to the <Vtt> child element in the declaration of the VorType, having as type TimetableNavaidType (which is a complex type).

In addition, not all entities in the AIXM entity-relationship data model are supported by the AIXM Schema. This is the case mainly for entities that model elements of the Integrated Aeronautical Information Package, such as AIP, NOTAM, AIC, AIP Supplements. The purpose of including them into the entity-relationship model was to show the relationships which exist between these documents and the so-called 'tabular data' (or 'structured data').

The eAIP[2] Project is looking into defining XML DTDs for these documents. The essential difference between AIXM and the eAIP is that AIXM focuses on the exchange of the structured aeronautical information, while the eAIP focuses on the exchange of the AIP document content, which is at 80% free textual information (no structure, no type).

Future versions of AIXM are expected to include a definition for the currently unsupported entity, TWY_INTERSECTION. For the moment, this entity should be regarded as a 'placeholder', an indication of future work that needs to be done.

Note that Obstacle in Airspace, which has been removed from the AICM in edition 3.3, still exists in the AIXM Snapshot Schema. It has been considered useful to preserve it in Snapshot messages, in case a data provider would like to communicate the list of all obstacles in a given FIR/UIR. This would be useful for AIP production, as there exist a table of En-route obstacles in ENR 5.4. It is intended to restore it to a later version of the AICM

Appendix A contains mappings for entity-to-feature correspondence.

There exist one important difference between the naming conventions for AIXM XML elements and the name of the corresponding entity in AICM. Short abbreviations of three letters are used as feature names, instead of the name of the AICM entity. The purpose was to reduce the typing effort of the programmers that will have to write code, frequently referring to feature names. Where available, 3 letter ICAO abbreviations have been used, such as for VOR, DME, ILS, MLS, NDB, RWY, etc.

Further details on AIXM naming conventions are given in chapter 3.

2.4. Data Types

Within the conceptual model all data types were based on domains, which specify valid content for particular kinds of value by restricting numbers to a particular range or text to a particular length etc. Every AICM domain has an equivalent data type in AIXM, although they were implemented in varying ways. Some, namely:

- NUMBER

[1] the word 'entity' is used within this document in the entity-relationship sense; it is not an 'XML entity'
[2] http://www.eurocontrol.int/ais/ahead/eaip

- ZNUMBER
- DEGREES1
- DEGREES2

were mapped directly to XML Schema types and restricted using facets. Other domains, such as:

- LATITUDE
- LONGITUDE
- DATE
- TIME
- DATETIME

were mapped as strings restricted by facet patterns. All of these types are used directly. Conversely, domains such as ALPHA were declared as base types in the schema, and were then restricted by other types. Mostly this construct was used for non-enumerated text types restricted by length, typically to restrict names etc. to a particular selection of letters.

One important aspect of data types to contrast the above paragraph with the implementation of the CHARACTER2 domain, which was mapped directly to the xsd:string type without restriction. It is used for textual descriptions and remarks, and permits the use of any UNICODE character. In relation with this, due attention should be paid to character encoding issues - see Appendix C.

2.5. Deprecated Elements

Certain AIXM feature types and properties (elements) have been deprecated in AIXM 4.0 and in AIXM 4.5. That means that, although the elements still exist in the XML Schema, their usage should be abandoned as soon as possible. The reason they are still in the schema is "technical backwards compatibility": i.e. to ensure that an AIXM message that is valid against a previous version is also valid against version 4.5.

Deprecated elements are marked in the AIXM XML Schema with <xsd:appinfo> annotation elements, as shown below. These tags can be used to direct an implementing system to handle deprecated elements as required.

```
<xsd:complexType name="DlnType">
  <xsd:annotation>
    <xsd:appinfo>DEPRECATED-4.0</xsd:appinfo>
    <xsd:documentation>[Deprecated] DME - Limitation - Version</xsd:documentation>
  </xsd:annotation>
...
</xsd:complexType>
```

An alternative, so-called "strict" version of AIXM 4.5, is also available, from which all deprecated elements have been removed.

2.6. AICM Rules

In addition to information about entities, attributes and relationships, the AICM also contains textual rules that cannot be represented using an entity-relationship model. These are classified into two types: "technical" and "business", with business rules further classified into subcategories.

A technical rule is one that is necessary in order to avoid data ambiguity, or, to put it another way, to protect systems from data they cannot handle. A typical example are rules requiring a unit-of-measure field to be present if a value field is present, such as shown below, taken from the DME definition in the AICM:

"If VAL_DISPLACE is specified, then UOM_DISPLACE is mandatory."

Business rules are defined as "simple atomic statements that define or constrain some aspect of business operations." They cannot be enforced in the AIXM XML Schema because there are often real-world situations in which they can be broken, although the data may be correct. In addition, business rules are often unenforceable using XML Schema. For example, the following rule, taken from the DESIGNATED_POINT definition (on the CODE_ID attribute) is mandated by ICAO but cannot be captured using XML Schema:

"If CODE_TYPE='ICAO', then CODE_ID should be unique world-wide."

It remains a task for every implementation of the model to decide to what degree such business rules can be enforced.

The first of the two technical rules that have been implemented since version 3.3 of the AIXM, govern the relationship between RWY (runway), RWY_DIRECTION (runway direction), and RWY_CLINE_POINT (runway centreline point).

"The RWY_CLINE_POINT used as threshold must be related to the same RWY as the current RWY_DIRECTION."

The XML schema originally contained the following construction for <Rdn> (runway direction):

```
<Rdn>
  <RdnUid>...</RdnUid>
  <RcpUid>...</RcpUid>
  <valTrueBrg>...</valTrueBrg>
  ...
</Rdn>
```

Requiring the <Rdn> element to have an <RcpUid> (runway centerline point identifier) created an explicit relationship to an existing runway centerline point . However, although both the <RdnUid> and the <RcpUid> types contain a <RwyUid>, the schema could not prevent a dataset from using - wrongly - the runway centerline point of one runway as the threshold position for another runway. To eliminate that possibility and to remove the duplicate <RwyUid>, the <RcpUid> element was replaced with just <geo_lat> (latitude) and <geo_long> (longitude) elements. Therefore, when receiving an <Rdn> element with explicit lat/long values, there must exist an <Rcp> for the same <Rwy> with *exactly* the same lat/long values; if this is not the case, then there is an error somewhere.

A similar rule governs FATO (final approach and take-off area), FATO_CLINE_POINT (FATO centreline point) and FATO_DIRECTION (FATO direction), and it has been encoded in a similar way.

"The FATO_CLINE_POINT used as threshold must be related to the same FATO as the current FATO_DIRECTION."

Chapter 3. Basic Concepts

3.1. AIXM Features Types

The basic AIXM information unit is the *AIXM feature*. A feature corresponds to a real thing in the aeronautical environment, such as a navaid, runway, route segment, airport etc.. Each feature may have attributes and relationships, all declared as child elements of the feature.

There are two levels in the definition of an AIXM feature. First, *feature types* are defined as complex types in the AIXM-Features.xsd sub-schema file. Starting with AIXM version 4.5, these complex types have un-abbreviated meaningful names, which are based on the content of the "full name" specified for every AICM entity, followed by the keyword "Type". For example, the AD_HP_OBSTACLE entity has "Obstacle at Aerodrome / Heliport" as full name. Therefore, the corresponding AIXM-Feature complex type is named "AerodromeHeliportObstacleType".

In the second step, when used in AIXM messages, AIXM feature types are used to declare AIXM *feature elements*. As explained in 2.3, short abbreviations of three letters are used for feature element names. They start with an upper case character, while the remaining 2 characters are in lower case (for example, 'Vor').

For example, there exist a complex type "AerodromeHeliportType", having child elements such as <txtName> - the name of the site, <codeIcao> - the ICAO Location Indicator, etc.:

```
<xsd:complexType name="AerodromeHeliportType">
  <xsd:annotation>
    <xsd:documentation>Aerodrome / Heliport</xsd:documentation>
  </xsd:annotation>
  <xsd:sequence>
    <xsd:element name="AhpUid" type="AerodromeHeliportUidType">
    …
  </xsd:sequence>
</xsd:complexType>
```

3.2. Feature Attributes

Attributes from the entity-relationship model correspond to local child elements in the declaration of the feature types. For example:

```
<xsd:element name="valFreq" type="valFreq">
  <xsd:annotation>
    <xsd:documentation>Frequency</xsd:documentation>
  </xsd:annotation>
</xsd:element>
```

<valFreq> is a local child element in the definition of the complex type VorType, having as type 'valFreq'. The latter is a simple type and corresponds to an entity-relationship attribute domain. All such simple types are defined in the AIXM-~~Update-~~DataTypes.xsd sub-schema ~~and the AIXM-Snapshot-DataTypes.xsd sub-schema~~.

In general, all attributes from the entity-relationship model have an equivalent feature attribute in AIXM. However, as the order of elements is significant in XML, many NO_SEQ attributes from the entity-relationship model have not been included into the AIXM-Features sub-schema. For example, the NO_SEQ attribute of the AIRSPACE_VERTEX entity does not exist in the definition of the ~~Avx~~ AirspaceBorderVertexType in the XML Schema. This is also related to the fact that using AIXM it is only possible to exchange data about an entire airspace border, not about individual border points (vertexes). If a

border point has changed, then the whole border data, including all its points, must be included in AIXM-Update messages. This is due to the difficulty to identify individual vertexes in order to update them separately. Adding a sequence number to every vertex was not considered necessary, because the order of the child elements in XML already gives an implicit sequence. In addition, sequence numbers are difficult to maintain when a vertex is added or deleted in the middle of the border definition. It was consider safer to require the whole updated border definition to be included in the AIXM-Update message.

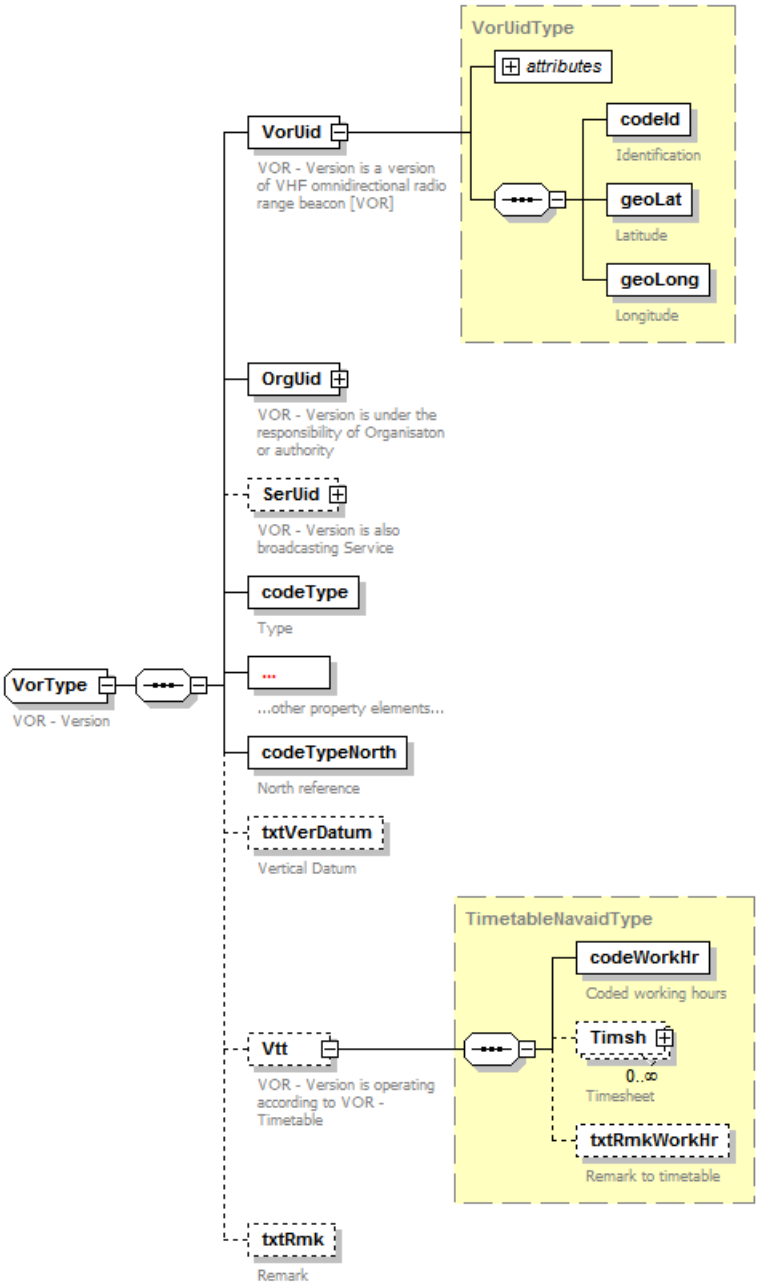
Feature attribute names have been derived from the corresponding attribute name in the entity-relationship model by adopting a *lowerCamelCase* style:

- removing the '_' character;
- converting the first token to lower case (it indicates a data type);
- converting the first character of the following tokens to upper case;
- converting the remaining characters of the other tokens to lower case.

For example, the CODE_TYPE_NORTH attribute of the VOR entity corresponds to the <codeTypeNorth> child element of the VorType. Using an XML Schema editor like XMLSpy[1], it is possible to represent in a diagram the definition of an AIXM feature and its attributes. For example, the diagram representing the VOR feature is included here.

[1] http://www.xmlspy.com

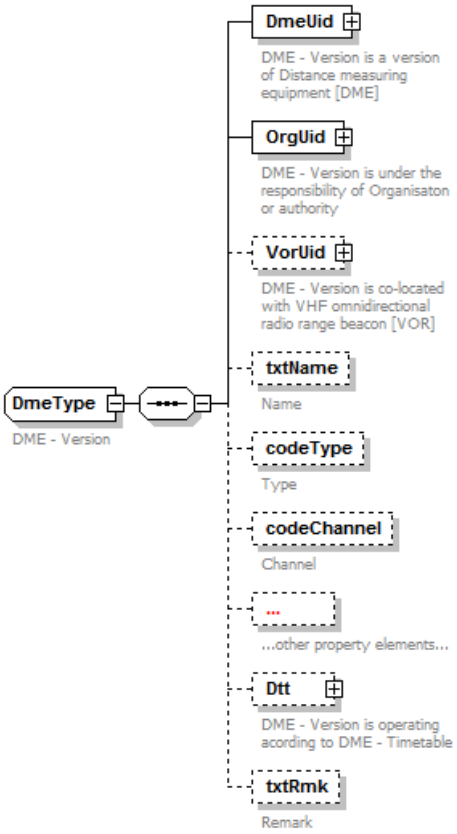
Figure 3.1. VOR Feature Diagram



3.3. Feature Identification

As described above, AIXM employs natural keys in order to uniquely identify feature instances. For example, the unique identifier of a VOR is composed of the position of the station (latitude and longitude) plus the radio identification. The unique identifier of a feature is declared as a separate complex type in the AIXM-Features.xsd sub-schema. For example, the unique identifier of the VOR feature is declared as the VorUidType complex type. The declaration of the VorType contains a child element named <VorUid>, having as type the VorUidType.

Figure 3.2. DME Feature Diagram



The AIXM Schema also allows for the use of an artificial unique identifier (surrogate key). The declaration of every *"FeatureUid"* complex type contains an optional attribute named 'mid' of type xsd:string ~~long~~, which can be used as artificial unique identifier in a given context. This context can be a dataset or an application domain.

An example of a dataset could be an AIXM encoded export of a complete aeronautical database. Unique artificial identifiers could be generated when the dataset is created and cease to exist after the dataset is processed by the receiving system. Such an artificial identifier will have a transient nature.

An application domain could result from an agreement between two or more systems to use AIXM with artificial identifiers. How exactly the artificial unique identifiers are constructed and resolved in order to be persistent remains to be agreed between such systems.

~~The European AIS Database (EAD), which employs AIXM as main data exchange format, will use natural keys as unique identifiers. The values of the 'mid' attributes, if present in an AIXM-Update file received by the EAD, will be ignored. However, the EAD will fill-in the 'mid' attributes for all AIXM-Update messages generated by the EAD[2].~~

How features are uniquely identified has consequences on two aspects of the Schema:

- how relationships are encoded and

~~[2] Due to internal EAD reasons, a number of features will 'borrow' the 'mid' value from their parent entities. At the time of the writing of this document, this was foreseen to be the situation for the following 'geometry type' features: Abd, Apg, Fpg, Tsg, Tlg, Rpg. They will use the same 'mid' value as their parent Ase, Apn, Fpa, Tsa, Tla and Rpa respectively. Please check Appendix A for the meaning of these 3-letter abbreviated feature names.~~

- how updates are encoded.

3.4. Feature Relationships

Relationships are encoded by including the natural key of the related feature.

For example, the collocation relationship between a DME and a VOR is encoded by a child element of with name <VorUid> and of type VorUid in the DmeType. A DME feature instance would contain at this place the natural key of the collocated VOR: its position (latitude and longitude) and its radio identification.

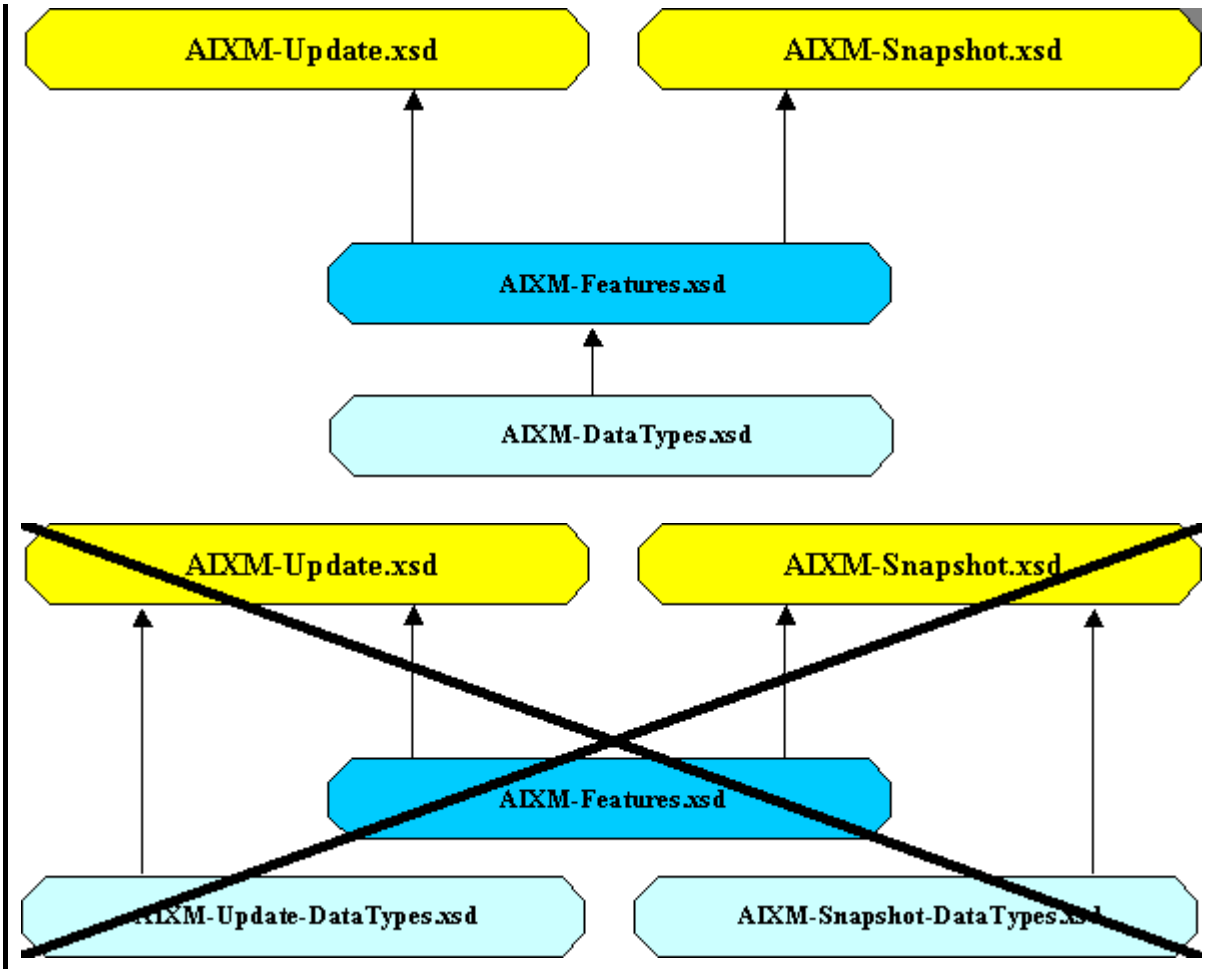
The same mechanism is applied for all relationships encoded in the AIXM Schema. As a rule, the name of a child element that corresponds to a relationship usually ends in (but sometimes only contains) "Uid".

Chapter 4. AIXM Schema Files

4.1. Inclusions

The diagram below indicates how the ~~five~~ four sub-schema files, which compose the AIXM Schema, are included in each other. For example, the AIXM-Update.xsd schema needs to include ~~the AIXM-Update-DataTypes.xsd file and~~ the AIXM-Features.xsd file, which includes the AIXM-DataTypes.xsd file.

Figure 4.1. AIXM Schema File Inclusions



4.2. AIXM Data Types

~~When included into AIXM-Update.xsd, t~~The AIXM-Features sub-schema uses data types declared in the AIXM~~-Update~~-DataTypes.xsd. For example, the <geoLat> child element of the VorUidType is declared as having the type geoLat, which is defined in AIXM~~-Update~~-DataTypes.xsd file.

~~The data type geoLat is defined differently in AIXM-Update-DataTypes.xsd and AIXM-Snapshot-DataTypes.xsd files. The difference consists in an optional 'chg' attribute, which is allowed for the attributes of a feature when appearing in update messages, as discussed in 5.1.~~

When used in an AIXM-Update message, feature attributes such as <geoLat>, can have an optional 'chg' attribute, as explained in 5.1.4.2. This attribute is declared at the level of the geoLat type, in the AIXM-DataTypes.xsd, through the intermediate of an attribute group, named 'Changes'. However, in order to

enable the use of the 'chg' attribute only for Update messages and not for Snapshot messages, the 'Changes' attribute group is left undefined in the AIXM-DataTypes.xsd file. It is defined only at a later stage, as follows:

- in the AIXM-Update.xsd file, the 'Changes' attribute group is defined as containing a 'chg' attribute
- in the AIXM-Snapshot.xsd file, the 'Changes' attribute group is defined as an empty group

Chapter 5. AIXM Message Types

Currently, the AIXM exchange format supports two types of data exchange:

- "Update" messages - containing data about new, changed or withdrawn aeronautical features;
- "Snapshot" messages - containing data about the versions of selected aeronautical features which are valid at a given date and time.

Other message types are likely to be added in future versions of AIXM. An example could be an AIXM-Query Schema, allowing to express queries with reference to AIXM features. Such queries would then be interpreted by the receiving system and the result could come back, for example, in AIXM-Snapshot format.

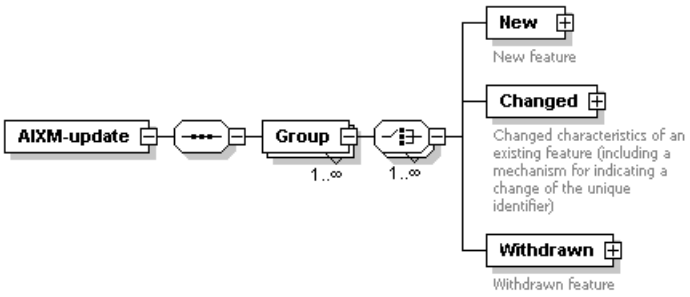
5.1. AIXM-Update

5.1.1. General

AIXM-Update messages are used to bring up to date previously exchanged data, without the need for reissuing a complete new data set.

An AIXM-Update XML message has as root the <AIXM-Update> element, which contains a sequence of one or more <Group> elements.

Figure 5.1. AIXM-Update Diagram



A system that implements AIXM may require a mechanism to guarantee that no messages are lost or left unprocessed by the receiving system. Otherwise, this could result in a lack of synchronisation between the sender and the receiver. The <AIXM-Update> element has two attributes that are intended to facilitate the implementation of such a synchronisation control mechanism: 'origin' and 'number', which shall contain the name of the issuing authority and a sequence number respectively. This allows a receiving system to check that it has received and successfully processed all the messages. A system issuing AIXM-Update messages should foresee a mechanism that allows receiving systems to request a copy of a previously sent message.

~~In the EAD implementation, the EAD System Interface (ESI[1]) is responsible for ensuring that all messages are correctly received and processed. At the time of the writing of this document, the 'number' attribute mentioned above was not foreseen to be used.~~

[1] ~~For details, please see the "ESI Decision paper" at http://www.eurocontrol.int/ead/downloads.html.~~

5.1.2. Groups

All changes included in one AIXM-Update message must have the same effective date, specified in the 'effective' attribute of the <AIXM-Update> element. Inside an Update message <Group> elements may be used to collate multiple modifications which have a common cause (specified in the 'Reason' attribute).

A <Group> contains one or more modifications. Each modification must belong to one of the following categories: new, changed or withdrawn. They occur as child elements of the <Group>, as shown in the diagram above.

The use of <Group> elements is optional: if a system does not require or it is not able to group updates by their common reason, it may create an AIXM-Update message that has only one <Group>, containing all changes.

In the EAD, AIXM-Update Groups are used in connection with the 'slots mechanism'[2]. 'EAD slots' may be regarded as containers, used to organise data updates issued by different originators, for the purpose of applying validation rules. 'Private slots' are used to group together updates issued by the same originator, having the same cause and the same effective date. 'Public slots' are then used to group together private slots that have the same effective date. A <Group> element has two optional attributes, 'Name' and 'SubName'. When used to send data to the EAD, these attributes can contain the identifier of the public and private slot respectively, in which the updates must be included in the EAD.

5.1.3. New feature

The <New> element contains data about one new feature instance, such as a new VOR, a new RWY or a new route. For example, a new type of fuel available at the Brussels airport could be encoded as follows:

```
…
<New>
  <Ful>
    <FulUid>
      <AhpUid>
        <codeId>EBBR</codeId>
      </AhpUid>
      <codeCat>AVGAS-LL</codeCat>
    </FulUid>
    <txtDescr>Octane 100 Low Lead aviation fuel</txtDescr>
    <txtRmk>Indicative price is 99.99 EUR / tone</txtRmk>
  </Ful>
</New>
…
```

The unique identifier (natural key) of the new feature instance is made of the child elements of <FulUid>: airport identifier ('EBBR') and fuel category ('AVGAS-LL'). If the receiving system already has a feature with the same natural key, it is expected to trigger an error message. How such error messages are handled and what is their content is not part of AIXM. This is considered to be implementation specific.

5.1.4. Changed Feature

A <Changed> element contains data about the modified attributes and relationships of one previously sent feature. As feature attributes and relationships are used in the composition of the unique identifier (natural key), a mechanism to indicate a change of the unique identifier has been included in the Schema.

[2] For details about EAD slots, please see the "EAD Slot Management" document available at "http://www.eurocontrol.int/ead/downloads.html "

5.1.4.1. Unchanged Natural Key

Changes that do not affect the natural key of a feature are encoded as in the example below. The Changed element contains first the natural key of the feature instance (<VorUid> in this example) and then the rest of its attributes and relationships.

```
…
<Changed>
  <Vor>
    <VorUid mid='187652432'>
      <codeId>BXL</codeId>
      <geoLat>514326.67N</geoLat>
      <geoLong>0032345.37E</geoLong>
    </VorUid>
    <OrgUid>
      <txtName>BELGIUM</txtName>
    </OrgUid>
    <txtName>BRUSSELS VOR</txtName>
    <codeType>DVOR</codeType>
    <valFreq chg="1">119.00</valFreq>
    <uomFreq>MHZ</uomFreq>
    <codeTypeNorth>MAG</codeTypeNorth>
    <valDeclination>1.3</valDeclination>
    <codeDatum>WGE</codeDatum>
    <Vtt>
      <codeWorkHr>H24</codeWorkHr>
    </Vtt>
    <txtRmk>This is just an example</txtRmk>
  </Vor>
</Changed>
…
```

The rule is that all feature attributes and relationships that have a value need to be included, both changed and unchanged ones. In the example above only the frequency of the VOR, encoded in the element <valFreq> has changed to 119.00 (MHz). However, all the other feature attributes and relationship that are not null must be included (note: as for a <New> feature). This is to avoid uncertainty and look-up overheads in determining the current value of a feature attribute or relationship.

The current version of AIXM Schema does not support NULL values. In an Update or Snapshot message, all feature attributes and relationships that are NULL (do not have a value) are simply not included. For example, if the name of a VOR becomes NULL (for whatever reason, such as because the VOR does no longer have a name or it is unknown), then the corresponding <txtName> field would be absent in the <Changed> message exemplified

In future versions, if it becomes necessary to allow NULL values, AIXM will need to have a specific schema construct in order for such messages to be valid. Also, a flag would be necessary in the AIXM-Update message in order to indicate whether or not an AIXM source is using NULL values.

5.1.4.2. 'chg' attribute

The attribute chg='1' of the <valFreq> element indicates that this is a value which has actually changed. However, the use of 'chg' attributes is optional. The 'use_chg' attribute of the root <AIXM-Update> element shall indicate if the changes are explicitly marked by a 'chg='1' attribute or not.

5.1.4.3. 'mid' attribute

As discussed at 3.3 above, the possibility to use artificial unique identifiers is foreseen in the Schema. In this example, the mid='187652432' attribute of the <VorUid> element contains an example of such

an artificial unique identifier. The 'mid' attribute is optional. The decision to use/not use artificial identifiers in AIXM-Update messages must be made by the implementers of systems which use AIXM as exchange format.

When data is uploaded into the EAD, the system will use natural keys in order to identify the records that are affected by the change. Even if present in the AIXM-Update message, the 'mid' attributes will be ignored. However, the EAD system will fill-in the 'mid' attributes when issuing AIXM-Update or AIXM-Snapshot messages. This might be useful for systems downloading data from the EAD.

5.1.4.4. Changed Natural Key

Changes that affect the natural key of the feature are encoded as in the example below. The <Changed> element shall contain first the previous natural key of the feature instance (<VorUid> in this example) and then new <Vor> element, including its new natural key and the rest of its attributes and relationships.

Again, as explained above, the 'mid' and the 'chg' attributes are optional.

```
…
<Changed>
  <VorUid mid='187652432'>                        <!-- the old natural key -->
    <codeId>BXL</codeId>
    <geoLat>514333.34N</geoLat>
    <geoLong>0032301.21E</geoLong>
  </VorUid>
  <Vor>
    <VorUid mid='187652432'>                 <!-- the new natural key -->
      <codeId>BXL</codeId>
      <geoLat chg="1">514326.67N</geoLat>
      <geoLong chg="1">0032345.37E</geoLong>
    </VorUid>
    <OrgUid>
      <txtName>BELGIUM</txtName>
    </OrgUid>
    <txtName>BRUSSELS VOR</txtName>
    <codeType>DVOR</codeType>
    <valFreq chg="1">119.00</valFreq>
    <uomFreq>MHZ</uomFreq>
    <codeTypeNorth>MAG</codeTypeNorth>
    <valDeclination>1.3</valDeclination>
    <codeDatum>WGE</codeDatum>
    <Vtt>
      <codeWorkHr>H24</codeWorkHr>
    </Vtt>
    <txtRmk>This is just an example</txtRmk>
  </Vor>
</Changed>
…
```

As mentioned before, the AIXM Schema was created starting from the AIXM entity-relationship model. If applied as such in a relational database, the model would not allow for changes of the natural key to occur. However, the AIXM Schema allows for changes of the natural keys. The AIXM entity-relationship model should not be used as such (1:1) for a database implementation. Its main purpose was to facilitate the generation of the AIXM Schema and it may contain modelling structures that are not appropriate for a straight-forward relational database implementation. Implementers of the AIXM entity-relationship model should perform a rationalisation of the model before implementing it in a physical database.

5.1.5. Withdrawn Feature

A <Withdrawn> element contains data about one feature instance, which will no longer exist starting from the effective date of the AIXM-Update message. The rule is that only the unique identifier of the feature instance must be specified in a <Withdrawn> element.

For example, a type of fuel which is no longer available at the Brussels airport could be encoded as it follows:

```
…
<Withdrawn>
  <FulUid>
    <AhpUid>
      <codeId>EBBR</codeId>
    </AhpUid>
    <codeCat>AVGAS</codeCat>
  </FulUid>
</Withdrawn>
…
```

5.1.6. Corrections to AIXM-Update messages

Data updates through AIXM-Update messages are normally issued long time in advance before their effective date. For operationally significant changes, the AIRAC cycle shall apply. Particular implementations might have even longer lead periods between the date when the changes are committed and their effective date.

In this document the term pending data is used to designate data, which has already been published (through an AIXM-Update message), but which has not reached yet the effective date.

There is an obvious need to be able to issue corrections to pending data. ~~This is especially true for implementations such as the EAD, which perform coherence checks involving pending changes of adjacent States.~~ AIXM-Update messages could be used as well in order to correct pending data. The convention would be that when a feature version with the same effective date and the same natural key is issued by a system, this new version overwrites the previously sent change.

There are some questions with how different combinations of the update primitives New/Changed/Withdrawn should be interpreted. For example, a 'withdrawn' of a pending 'new' feature should have as consequence that the feature will not be created at all. There are also combinations that do not make sense, such as sending a 'new' after a 'changed'.

However, particular implementations may define their specific mechanism for corrections. In future versions, it will be possible to include a correction mechanism in the AIXM Schema.

5.1.7. Aspects of Usage

5.1.7.1. Operational Rules

Ideally, an AIXM message that is valid against the Schema should be accepted by the receiving system without errors. However, a number of constraints that need to be imposed on AIXM messages cannot be expressed using an XML Schema. Such operational rules are specified further down.

For the purpose of applying the rules specified further down, all AIXM-Update messages issued by one source for the same effective date will be regarded as one message. This message is resulting from the concatenation of all the individual Groups under a single <AIXM-Update> element, taking into consideration their publication order.

XSLT stylesheets is a good candidate technology for the implementation of the compliance checks of AIXM messages with some of these rules.

5.1.7.1.1. Order of Related Features

Rule: In AIXM-Update messages, data about a <New> feature must occur before any relationship towards this new feature occurs.

This rule is applicable both within one AIXM-Update message and when considering all AIXM-Update messages issued by one originator for the same effective date.

This operational rule is primarily intended to allow for the use of event-based parsers, such as SAX, instead of object-based parsers such as DOM. The essential difference[3] is that a DOM parser works by loading the whole XML document into memory and constructing a DOM tree.

The problem with using DOM parsers is that AIXM-Update messages could be quite large and cause errors related to insufficient memory available. The SAX API can provide faster and less costly processing of XML data. However, they work only when you do not need to access all of the data in an XML document.

In order to eliminate the need to access the whole XML document in memory, a certain sequence of features needs to be imposed in an AIXM-Update message. This is related to relationships. For example, let's suppose that a new VOR/DME is created. As the collocation relationship is encoded in the <Dme> element through the inclusion of the <VorUid>, the <New> element containing the <Vor> shall be placed before the <New>element containing the <Dme>.

5.1.7.1.2. Natural Key Update

The use of natural keys in AIXM-Update messages together with the fact that a natural key may change, requires a few rules to be followed in order to eliminate ambiguities.

Related features should not be re-sent.

Rule: A change in the natural key of a feature F1 does not require the originator to send also <Changed> notifications for any other feature that has the natural key of F1 as descendant element.

For example, let's suppose that the natural key of an aerodrome/heliport changes. The natural key of the aerodrome/heliport appears as <AhpUid> child element in all the features that are related to that aerodrome, such as <Rwy>, <Twy>, <Ful>, etc. The question is, does the originator have to include all these related features in the AIXM-Update message?

The rule indicates that, if the natural key of the <Ahp> feature instance has changed, it is not necessary to re-send data about all the <Rwy>, <Twy>, <Ful>, etc. which are related to that aerodrome/heliport.

This rule is based on the assumption that most systems work internally with artificial unique identifiers. For such systems, no change occurs in the related features if there is a change of the natural key of the feature to which they are related. For example, no change occurs to the DME record if there is a change of the natural key of the collocated VOR.

[3] for example, see http://www-3.ibm.com/software/ts/tpf/pubs/xml16/xdvs.htm

Related elements should contain the new natural key.

Sometimes, it is necessary to include in AIXM-Update messages data about a feature, which is related to another feature whose natural key has changed at the same effective date. The question is, which natural key shall be included, the old one or the new one?

Rule: If a <Changed> notification for the natural key of a feature F1 was issued for an effective date DT1, then any subsequent AIXM-Update message for the same or a later effective date, including the current one, issued by the same originator, which contains a <New>, <Changed> or <Withdrawn> notification about a feature F2 having the natural key of F1 as descendant, shall contain the new values for the natural key of F1.

For example, the natural key of an aerodrome/heliport changes and let's suppose that the length of a RWY located at that aerodrome/heliport changes as well at the same effective date. If the <Changed> notification for the natural key of the <Ahp> feature has already been sent, then the <Changed> notification for the <Rwy> feature shall contain the new natural key in the <AhpUid> element.

The same rule applies for withdrawn features. If a RWY ceases to exist at the aerodrome, then the <Withdrawn> notification for the <Rwy> feature shall contain the new natural key of the <Ahp> feature, as it is included in the natural key of the <Rwy>.

5.1.7.2. Limited Geographical Areas

Users might be interested to receive from an AIXM data source messages covering only a limited geographical area, such as an FIR. This is possible with AIXM. However, there are two aspects that need to be mentioned.

- Certain types of feature do not have a geographical aspect and so cannot be collated in that way. For example, if an AIXM source is to provide downloads based on a geographical area, certain elements - such as Org_Auth - cannot be readily included. A choice must be made between including all non-related features or only directly referenced ones.

- Another potential issue can arise with update messages for limited geographical areas. An update can generate "change" messages for designated points that are unknown to the receiving system. This is because it is possible for the points to have "moved" into the area of interest (e.g. an FIR). Instead of a "changed" message, the receiving system expects a "new" message. This can also apply to children of the unknown feature, for example in the case of "changed" messages to an route segment usage <Rsu>, which is the child of an unknown route segment. Potential problems such as this one could be handled by implementing special procedures in the client application. For example, to resolve the above scenario, the system could treat the unknown feature in the "changed" message as if it were in a "new" message, and thus create the object locally.

5.1.7.3. Cascaded Updates

The AIXM does not compel applications to cascade the effects of changes or deletions to the database. For example, if a VOR is deleted, one could argue that all route segments etc. that use that VOR should also be deleted. The precise behaviour in circumstances such as this is left entirely as an application-level decision.

5.1.7.4. Related Features

The AIXM does not require referred features to be present in a message - this again is an application-level decision. For example, if an application did not want to store VORs, but just DMEs, its upload

software should simply ignore all references towards VORs. The related VORs would not have to be present in the parameter set or the AIXM download file.

5.2. AIXM-Snapshot

AIXM-Snapshot XML messages could be regarded as a kind of database reports, containing data about versions of aeronautical features which are valid at a specific moment (date and time). The root element is <AIXM-Snapshot>. It has attributes indicating the origin of the message, the date and time when it was issued and the date and time used as selection criteria (effective time). It contains elements such as <Vor>, <Dme>, <Rsg>, etc., as defined in the AIXM-Features.xsd sub-schema. Each such feature element contains data about the version of a feature. The values of all feature attributes correspond to the date and time specified in the 'effective' attribute of the <AIXM-Snapshot> element.

An extract from an AIXM snapshot message is given below:

```
<AIXM-Snapshot
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="AIXM-Snapshot.xsd"
      version="1.1"
      origin="EAD"
      created="2002-05-27T15:25:34"
      effective="2002-05-26T01:00:00">
  <Dme>
    <DmeUid mid='137456733'>
      <codeId>BXL</codeId>
      <geoLat>514327.45N</geoLat>
      <geoLong>0032346.21E</geoLong>
    </DmeUid>
    <OrgUid mid='178692030'>
      <txtName>BELGIUM</txtName>
    </OrgUid>
    <VorUid mid='187652432'>
      <codeId>BXL</codeId>
      <geoLat>514326.67N</geoLat>
      <geoLong>0032345.37E</geoLong>
    </VorUid>
    <codeChannel>15X</codeChannel>
    <codeDatum>WGE</codeDatum>
    <valElev>350</valElev>
    <uomDistVer>FT</uomDistVer>
    <Dtt>
      <codeWorkHr>H24</codeWorkHr>
    </Dtt>
  </Dme>
  …
</AIXM-Snapshot>
```

As mentioned in 3.4, feature relationships are encoded by including the natural key of the related feature. In the example above, the natural key of the State ('BELGIUM') responsible for the DME and the natural key of the collocated VOR (codeId, geoLat and geoLong) are included in the <Dme> element.

Chapter 6. Data Integrity

6.1. Introduction

In order to aid system implementers to achieve the ICAO requirements for CRC (cyclic redundancy check) protection, AICM has provision for CRC values, which are modelled by a 'VAL_CRC' attribute in a number of AICM entities (navaids, aerodrome/heliport, etc.). The following section describes the CRC algorithm. Appendix B provides sample Java code for implementing the CRC algorithm.

6.2. CRC Algorithm

The CRC checking is used in numerous systems to verify the integrity of the information during data transmission. If the computed CRC bits are different from the original (transmitted) CRC bits, then there has been an error occurred in the transmission. If there are identical, it can be assumed that no error occurred (there is a chance 1 in 4 billion that two different bit streams have the same CRC32).

The idea is that the data bits are treated as a data polynomial and the CRC bits represent the remainder of the division of the data polynomial by a fixed, known polynomial (called the CRC polynomial). The CRC32 polynomial used here is the ICAO-approved CRC32Q which is defined as fallows:

```
1 + x + x^3 + x^5 + x^7 + x^8 + x^14 + x^16 + x^22 + x^24 + x^31 + x^32
```

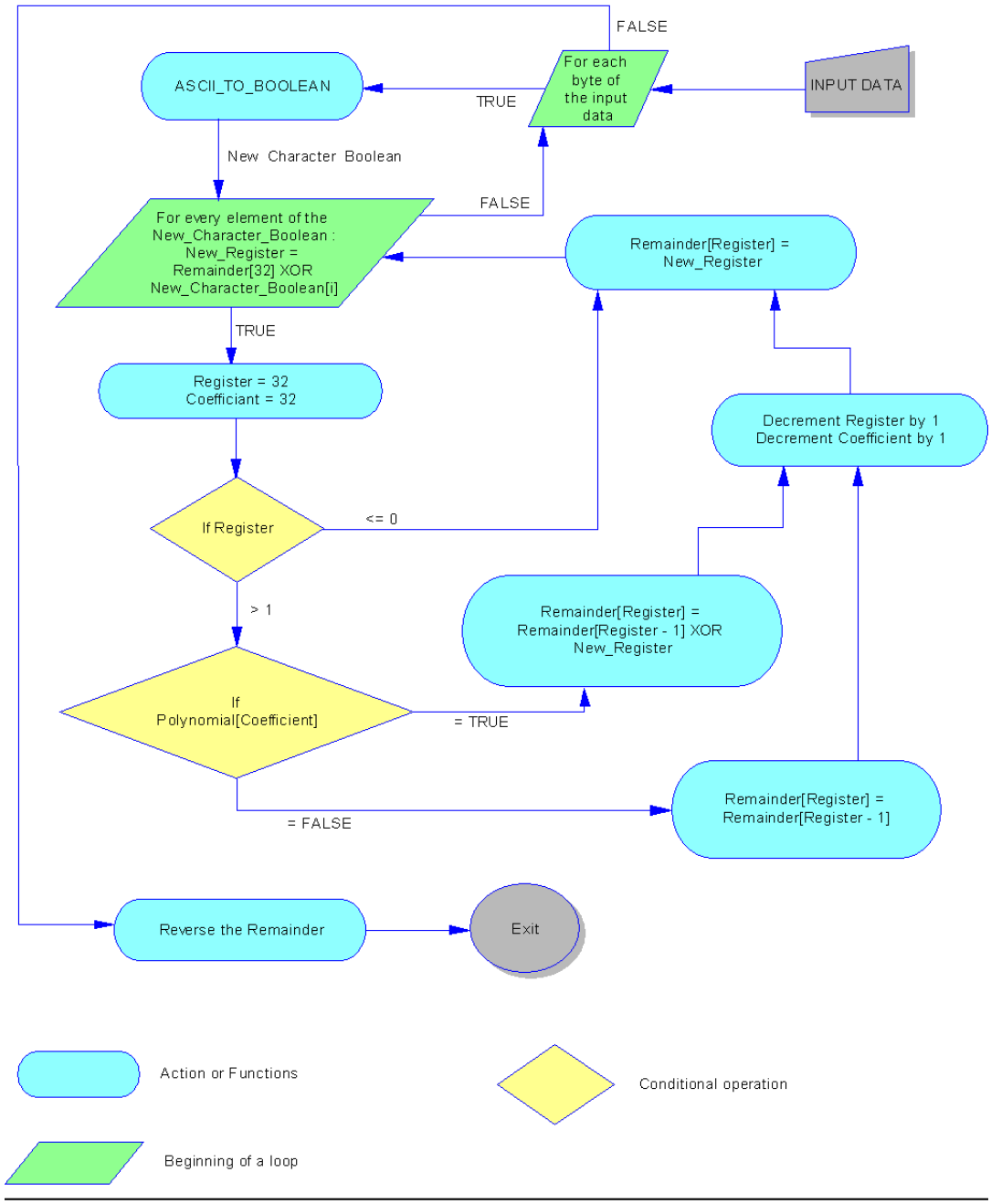
In a binary representation, where the coefficients are represented by a logical 1 we have:

```
11010101100000101000001010000001
```

For the calculation of the value of the CRC in AIXM messages, the bit stream used is composed through the concatenation of the XML fields concerned.

The algorithmic representation of the procedure of calculation of the CRC is represented below. The input data of this procedure is the bits streams defines above. The output data of this procedure is the remainder which contains the CRC (32 bits).

Figure 6.1. CRC Algorithm Flowchart



The function « ASCII_TO_BOOLEAN » converted the ASCII value of a character into a boolean array. For example if you take the ‘A’ character, the Hexadecimal representation of this ASCII value is 0x41. We have of binary representation:

```
01000001
```

the function « ASCII_TO_BOOLEAN » return for this character :

```
False, true, false, false, false, false, false, true.
```

To validate our algorithm, we developed a class java "CRC", which calculates value CRC of an introduced data. Here a table, this gives the result of this class according to different stream from entry.

Table 6.1. Example data for CRC class

Latitude	Longitude	Height	Geoid	CRC
480637N	-	-	-	A5A7C704
-	0163411E	-	-	A1AE5741
480637N	0163411E	-	-	A1BA30EE
-	-	782	-	6C297100
480637N	0163411E	782	-	6A259F4E
-	-	-	46.7	266D25C1
480637N	0163411E	-	46.7	2F866D6D
480637N	0163411E	782	46.7	5E5DC940

6.3. Note with regard to the EAD Implementation of AIXM

In the current EAD implementation (Releases 3 & 4), the bit stream used for the calculation of the CRC value in AIXM messages, composed by concatenating the following fields, in this order:

- Latitude (<geoLat>)
- Longitude (<geoLong>)
- Elevation (<valElev>)
- Geoid Undulation (<valGeoidUndulation>)

In cases where the elements representing the Elevation and the Geoid Undulation are “null” or the relevant features do not have attributes to take these values, the concatenation of latitude and longitude is used.

Appendix A. AIXM-AICM Mappings

A.1. Mapping of AIXM Features to AICM Entities, plus natural key compositions

Note: *For clarity, natural keys that are composed of a choice of relationships are listed without details of those choices. For example, an <Ana> (radio navigation aid at an aerodrome/heliport) must be related to an Aerodrome/Heliport and one of the following: TACAN, VOR, NDB, DME or MKR.*

Table A.1. AIXM-AICM Mappings

AIXM Feature Name	AIXM Complex Type	AICM Entity	Natural Keys
Aha	AerodromeHeliportAddressType	ADDRESS	Relationship to: AD_HP [Address is for Aerodrome / Heliport] Attribute: codeType Attribute: noSeq
Ahc	AerodromeHeliportColocationType	AD_HP_COLLOCATION	Relationship to: AD_HP [Aerodrome / Heliport - Co-location is having as 1st Aerodrome / Heliport] Relationship to: AD_HP [Aerodrome / Heliport - Co-location is having as 2nd Aerodrome / Heliport]
Ana	AerodromeHeliportNavaidType	AD_HP_NAV_AID	Relationship to: AD_HP [Radio navigation aid at Aerodrome / Heliport is associated with Aerodrome / Heliport] Relationship to: TACAN [Radio navigation aid at Aerodrome / Heliport is also Tactical air navigation beacon [TACAN]] Relationship to: VOR [Radio navigation aid at Aerodrome / Heliport is also VHF omnidirectional radio range beacon [VOR]] Relationship to: NDB [Radio navigation aid at Aerodrome / Heliport is also Non-directional radio beacon [NDB]] Relationship to: DME [Radio navigation aid at Aerodrome / Heliport is also Distance measuring equipment [DME]] Relationship to: MKR [Radio navigation aid at Aerodrome / Heliport is also Marker [MKR]]
Aho	AerodromeHeliportObstacleType	AD_HP_OBSTACLE	Relationship to: OBSTACLE [Obstacle at Aerodrome / Heliport is also Obstacle] Relationship to: AD_HP [Obstacle at Aerodrome / Heliport is affecting Aerodrome / Heliport]
Sah	AerodromeHeliportServiceType	SERVICE_AT_AD_HP	Relationship to: AD_HP [Service provided at an aerodrome/heliport is furnished for Aerodrome / Heliport] Relationship to: SERVICE [Service provided at an aerodrome/heliport is also Service]
Ahp	AerodromeHeliportType	AD_HP	Attribute: codeId
Ahu	AerodromeHeliportUsageType	AD_HP_USAGE	Relationship to: AD_HP []
Agl	AeronauticalGroundLightType	AERO_GND_LGT	Attribute: txtName Attribute: codeType

---	AircraftClassType	AIRCRAFT_CLASS	---
Aas	AirspaceAssociationType	AIRSPACE_ASSOCIATION	Relationship to: AIRSPACE [First member] Relationship to: AIRSPACE [Second member] Attribute: codeType
Ofa	AirspaceAuthorityType	AUTH_FOR_AIRSPACE	Relationship to: AIRSPACE [Authority responsible for airspace is responsible for Airspace] Relationship to: ORG_AUTH [Authority responsible for airspace is also Organisaton or authority]
---	AirspaceBorderCrossingType	AIRSPACE_BORDER_CROSSING	---
Abd	AirspaceBorderType	AIRSPACE_BORDER	Relationship to: AIRSPACE [Airspace border is describing the horizontal boundary of Airspace]
---	AirspaceBorderVertexType	AIRSPACE_VERTEX_TYPE	---
---	AirspaceCentrelineVertexType	AIRSPACE_CLINE_VERTEX	---
---	AirspaceCircularVertexType	AIRSPACE_CIRCLE_VERTEX	---
Acr	AirspaceCorridorType	AIRSPACE_CORRIDOR	Relationship to: AIRSPACE [Airspace corridor is describing the horizontal shape of Airspace]
Adg	AirspaceDerivedGeometryType	AIRSPACE_DERIV_GEOMETRY	Relationship to: AIRSPACE []
Oae	AirspaceObstacleType	---	---
Sae	AirspaceServiceType	SERVICE_IN_AIRSPACE	Relationship to: SERVICE [Service provided in airspace is also Service] Relationship to: AIRSPACE [Service provided in airspace is furnished within Airspace]
Ase	AirspaceType	AIRSPACE	Attribute: codeType Attribute: codeId
---	ApronGeometryPointType	APRON_SHAPE_POINT	---
Apg	ApronGeometryType	AD_HP_SURFACE_BORDER	Relationship to: APRON [Aerodrome / Heliport Surface border is describing the geometry of one Apron]
Als	ApronLightingSystemType	APRON_LGT_SYS	Relationship to: APRON [Apron lighting system is owned by one and only one Apron] Attribute: codePsn
Apn	ApronType	APRON	Relationship to: AD_HP [Apron is located at Aerodrome / Heliport] Attribute: txtName
---	CallsignType	CALLSIGN_DETAIL	---

Nsc	CheckpointType	NAV_SYS_CHECKPOINT	Attribute: codeType Relationship to: TWY_HOLDING_POSITION [Checkpoint is situated at TWY holding position] Relationship to: GATE_STAND [Checkpoint is situated at Parking position stand or gate]
Plc	CruisingLevelsColumnType	PREDEFINED_LVL_COLUMN	Relationship to: PREDEFINED_LVL_TABLE [Column of a table of criusing level is in Table of cruising level] Attribute: codeId
Plb	CruisingLevelsTableType	PREDEFINED_LVL_TABLE	Attribute: codeId
Dpn	DesignatedPointType	DESIGNATED_POINT	Attribute: codeId Attribute: geoLat Attribute: geoLong
---	DirectFlightClassType	DIRECT_FLIGHT_CLASS	---
---	DirectSegmentType	DIRECT_FLIGHT_SEGMENT	---
Dme	DmeType	DME	Attribute: codeId Attribute: geoLat Attribute: geoLong
Dli	DmeUsageLimitationType	DME_USAGE_LIMIT	Relationship to: DME [] Attribute: codeType
Rte	EnrouteRouteType	EN_ROUTE_RTE	Attribute: txtDesig Attribute: txtLocDesig
Fcp	FatoCentreLinePositionType	FATO_CLINE_POINT	Relationship to: FATO [FATO - centre line position is on the centre line of Final approach and take-off area [FATO]] Attribute: geoLat Attribute: geoLong
Fda	FatoDirectionApproach LightingType	FATO_DIRECTION_ALS	Relationship to: FATO_DIRECTION [Approach lighting system for FATO direction is serving Approach and take-off direction of a FATO] Attribute: codeType
Fdd	FatoDirectionDeclared DistanceType	FATO_DIRECTION_DECL_DIST	Relationship to: FATO_DIRECTION [Distance declared for FATO direction is for Approach and take-off direction of a FATO] Relationship to: TWY [Distance declared for FATO direction is associated with Taxiway [TWY]] Attribute: codeType Attribute: codeDayPeriod

Fdo	FatoDirectionObstacleType	FATO_DIRECTION_OBSTACLE	Relationship to: OBSTACLE [Obstacle for FATO direction is also Obstacle] Relationship to: FATO_DIRECTION [Obstacle for FATO direction is affecting Approach and take-off direction of a FATO]
Fds	FatoDirectionStarType	FATO_DIRECTION_STAR	Relationship to: STAR [FATO direction STAR is also Standard instrument arrival [STAR]] Relationship to: FATO_DIRECTION [FATO direction STAR is for Approach and take-off direction of a FATO]
Fdn	FatoDirectionType	FATO_DIRECTION	Relationship to: FATO [Approach and take-off direction of a FATO is using Final approach and take-off area [FATO]] Attribute: txtDesig
Fls	FatoLightingSystemType	FATO_DIRECTION_LGT_SYS	Relationship to: FATO_DIRECTION [FATO lighting system is owned by Approach and take-off direction of a FATO] Attribute: codePsn
---	FatoProtectionAreaGeometryPointType	FATO_PROTECT_SHAPE_POINT	---
Fpg	FatoProtectionAreaGeometryType	AD_HP_SURFACE_BORDER	Relationship to: FATO_PROTECT_AREA [Aerodrome / Heliport Surface border is describing the geometry of one FATO protection area]
Fpa	FatoProtectionAreaType	FATO_PROTECT_AREA	Relationship to: FATO_DIRECTION [FATO protection area is encompassing Approach and take-off direction of a FATO] Attribute: codeType
Fto	FatoType	FATO	Relationship to: AD_HP [Final approach and take-off area [FATO] is situated at Aerodrome/Heliport] Attribute: txtDesig
---	FlightClassType	FLIGHT_CLASS	---
---	FlowConditionElementLevelType	FLOW_COND_ELEMENT_LVL	---
---	FlowConditionElementType	FLOW_COND_ELEMENT	---
---	FlowConditionsCombinationType	FLOW_COND_COMBINATION	---
---	FlowRoutingElementLevelType	TFC_FLOW_RTE_ELEMENT_LVL	---
---	FlowRoutingElementType	TFC_FLOW_RTE_ELEMENT	---
---	FlowRoutingType	TFC_FLOW_RTE	---
Fqy	FrequencyType	FREQUENCY	Relationship to: SERVICE [Frequency is attributed to Service] Attribute: valFreqTrans

Ful	FuelType	FUEL	Relationship to: AD_HP [Fuel is available at Aerodrome / Heliport] Attribute: codeCat
Gsd	GateStandType	GATE_STAND	Relationship to: APRON [Parking position stand or gate is on Apron] Attribute: txtDesig
Gbr	GeographicalBorderType	GEO_BORDER	Attribute: txtName
---	GeographicalBorderVertexType	GEO_BORDER_VERTEX	---
Aga	GroundServiceAddressType	ADDRESS	Relationship to: AD_HP_GND_SER [Contact address is for Ground service] Attribute: codeType Attribute: noSeq
Ahs	GroundServiceType	AD_HP_GND_SER	Relationship to: AD_HP [Ground service is associated with Aerodrome / Heliport] Attribute: codeType
Shp	HoldingProcedureServiceType	SERVICE_ON_HOLDING_PROC	Relationship to: HOLDING_PROCEDURE [Service provided on holding procedure is furnished on Holding procedure] Relationship to: SERVICE [Service provided on holding procedure is also Service]
Hpe	HoldingProcedureType	HOLDING_PROCEDURE	Attribute: codeType Relationship to: TACAN [Significant point is at Tactical air navigation beacon [TACAN]] Relationship to: VOR [Significant point is at VHF omnidirectional radio range beacon [VOR]] Relationship to: DESIGNATED_POINT [Significant point is at Designated point] Relationship to: NDB [Significant point is at Non-directional radio beacon [NDB]] Relationship to: DME [Significant point is at Distance measuring equipment [DME]] Relationship to: MKR [Significant point is at Marker [MKR]]
---	IlsGlidePathType	ILS_GP	---
---	IlsLocalizerType	ILS_LLZ	---

Ils	IlsType	ILS	Relationship to: RWY_DIRECTION [Instrument landing system [ILS] is associated with RWY direction] Relationship to: FATO_DIRECTION [Instrument landing system [ILS] is associated with Approach and take-off direction of a FATO]
Sip	InstrumentApproachServiceType	SERVICE_ON_IAP	Relationship to: IAP [Service provided on IAP is furnished on Instrument approach procedure [IAP]] Relationship to: SERVICE [Service provided on IAP is also Service]
Iap	InstrumentApproachType	IAP	Relationship to: AD_HP [Instrument approach procedure [IAP] is associated with Aerodrome / Heliport] Attribute: txtDesig Attribute: codeCatAcft Attribute: codeTransId
Iue	InstrumentApproachUsage ConditionType	IAP_USAGE	Relationship to: IAP [IAP condition of usage is based on Instrument approach procedure [IAP]] Attribute: codeRteAvbl
---	LightGroupType	SURFACE_LGT_GROUP	---
Mkr	MkrType	MKR	Attribute: codeId Attribute: geoLat Attribute: geoLong
---	MlsAzimuthType	MLS_AZIMUTH	---
---	MlsElevationType	MLS_ELEVATION	---
Mls	MlsType	MLS	Relationship to: RWY_DIRECTION [Microwave landing system [MLS] is associated with RWY direction] Relationship to: FATO_DIRECTION [Microwave landing system [MLS] is associated with Approach and take-off direction of a FATO]

Mgp	MsaGroupType	MSA_GROUP	Relationship to: TACAN [Significant point is at Tactical air navigation beacon [TACAN]] Relationship to: VOR [Significant point is at VHF omnidirectional radio range beacon [VOR]] Relationship to: DESIGNATED_POINT [Significant point is at Designated point] Relationship to: NDB [Significant point is at Non-directional radio beacon [NDB]] Relationship to: DME [Significant point is at Distance measuring equipment [DME]] Relationship to: MKR [Significant point is at Marker [MKR]]
---	MsaType	MSA	---
Ain	NavaidAngularReferenceType	ANGLE_INDICATION	Relationship to: NAV_SYS_CHECKPOINT [Navaid - Angular reference is referring to Checkpoint] Relationship to: VOR [Navaid - Angular reference is using VHF omnidirectional radio range beacon [VOR]] Relationship to: NDB [Navaid - Angular reference is using Non-directional radio beacon [NDB]] Relationship to: TACAN [Navaid - Angular reference is using Tactical air navigation beacon [TACAN]] Relationship to: TACAN [Significant point is at Tactical air navigation beacon [TACAN]] Relationship to: VOR [Significant point is at VHF omnidirectional radio range beacon [VOR]] Relationship to: DESIGNATED_POINT [Significant point is at Designated point] Relationship to: NDB [Significant point is at Non-directional radio beacon [NDB]] Relationship to: DME [Significant point is at Distance measuring equipment [DME]] Relationship to: MKR [Significant point is at Marker [MKR]]
Din	NavaidDistanceIndicationtType	DISTANCE_INDICATION	
---	NavaidLimitationType	NAVAID_LIMITATION	---
Ndb	NdbType	NDB	Attribute: codeId Attribute: geoLat Attribute: geoLong

Nli	NdbUsageLimitationType	NDB_USAGE_LIMIT	Relationship to: NDB [] Attribute: codeType
Ntg	NitrogenType	NITROGEN	Relationship to: AD_HP [Nitrogen is available at Aerodrome / Heliport] Attribute: codeType
---	ObstacleClearanceAltitudeHeightType	OCA_OCH	---
Obs	ObstacleType	OBSTACLE	Attribute: geoLat Attribute: geoLong
Oil	OilType	OIL	Relationship to: AD_HP [Oil is available at Aerodrome / Heliport] Attribute: codeCat
Oaa	OrganisationAuthorityAddressType	ADDRESS	Relationship to: ORG_AUTH [Contact address is for Organisaton or authority] Attribute: codeType Attribute: noSeq
Oas	OrganisationAuthorityAssociationType	ORG_AUTH_ASSOC	Relationship to: ORG_AUTH [Organisaton or authority - Association is having as child Organisaton or authority] Relationship to: ORG_AUTH [Organisaton or authority - Association is having as parent Organisaton or authority] Attribute: codeType
Org	OrganisationAuthorityType	ORG_AUTH	Attribute: txtName
Oxg	OxygenType	OXYGEN	Relationship to: AD_HP [Oxygen is available at Aerodrome / Heliport] Attribute: codeType
Pfy	PassengerFacilityType	PASSENGER_FACILITY	Relationship to: AD_HP [Passenger facility is associated with Aerodrome / Heliport] Attribute: codeType Attribute: noSeq
---	ProcedureLegType	PROCEDURE_LEG	---
---	RoutePortionType	RTE_PORTION	---
Srs	RouteSegmentServiceType	SERVICE_ON_RTE_SEG	Relationship to: SERVICE [Service provided on route segment is also Service] Relationship to: RTE_SEG [Service provided on route segment is furnished on Route segment]
Rsg	RouteSegmentType	RTE_SEG	Relationship to: EN_ROUTE_RTE [Route segment is part of En-route route]
---	RouteSegmentUsageLevelType	RTE_SEG_USE_LVL	---

Rsu	RouteSegmentUsageType	RTE_SEG_USE	Relationship to: RTE_SEG [Route segment usage condition is based on Route segment] Attribute: codeRteAvbl Attribute: noSeq Attribute: codeDir
Rcp	RunwayCentreLinePositionType	RWY_CLINE_POINT	Relationship to: RWY [Position on the centre line of a RWY is on the center line of Runway [RWY]] Attribute: geoLat Attribute: geoLong
Rda	RunwayDirectionApproachLightingSystemType	RWY_DIRECTION_ALS	Relationship to: RWY_DIRECTION [RWY direction approach lighting system is serving RWY direction] Attribute: codeType
Rdd	RunwayDirectionDeclaredDistanceType	RWY_DIRECTION_DECL_DIST	Relationship to: RWY_DIRECTION [Declared distance for a RWY direction is for RWY direction] Relationship to: TWY [Declared distance for a RWY direction is associated with Taxiway [TWY]] Attribute: codeType Attribute: codeDayPeriod
Rls	RunwayDirectionLightingSystemType	RWY_DIRECTION_LGT_SYS	Relationship to: RWY_DIRECTION [RWY lighting system is owned by RWY direction] Attribute: codePsn
Rdo	RunwayDirectionObstacleType	RWY_DIRECTION_OBSTACLE	Relationship to: OBSTACLE [Obstacle for a RWY direction is also Obstacle] Relationship to: RWY_DIRECTION [Obstacle for a RWY direction is affecting RWY direction]
Rds	RunwayDirectionStarType	RWY_DIRECTION_STAR	Relationship to: STAR [RWY direction STAR is also Standard instrument arrival [STAR]] Relationship to: RWY_DIRECTION [RWY direction STAR is for RWY direction]
Rdn	RunwayDirectionType	RWY_DIRECTION	Relationship to: RWY [RWY direction is using Runway [RWY]] Attribute: txtDesig
---	RunwayProtectionAreaGeometryPointType	RWY_PROTECT_SHAPE_POINT	---
Rpg	RunwayProtectionAreaGeometryType	AD_HP_SURFACE_BORDER	Relationship to: RWY_PROTECT_AREA [Aerodrome / Heliport Surface border is describing the geometry of one RWY Protection area]

Rpa	RunwayProtectionAreaType	RWY_PROTECT_AREA	Relationship to: RWY_DIRECTION [RWY Protection area is encompassing RWY direction] Attribute: codeType
Rwy	RunwayType	RWY	Relationship to: AD_HP [Runway [RWY] is situated at Aerodrome / Heliport] Attribute: txtDesig
Ser	ServiceType	SERVICE	Relationship to: UNIT [Service is provided by Organisation unit providing services] Attribute: codeType Attribute: noSeq
Ssd	SidServiceType	SERVICE_ON_SID	Relationship to: SID [Service provided on SID is furnished on Standard instrument departure [SID]] Relationship to: SERVICE [Service provided on SID is also Service]
Sid	SidType	SID	Relationship to: AD_HP [Standard instrument departure [SID] is for Aerodrome / Heliport] Attribute: txtDesig Attribute: codeCatAcft Attribute: codeTransId
Sue	SidUsageType	SID_USAGE	Relationship to: SID [SID usage is based on Standard instrument departure [SID]] Attribute: codeRteAvbl
Spa	SignificantPointAirspaceType	SIGNIFICANT_POINT_IN_AS	Relationship to: AIRSPACE [Significant point in airspace is situated within Airspace] Relationship to: TACAN [Significant point is at Tactical air navigation beacon [TACAN]] Relationship to: VOR [Significant point is at VHF omnidirectional radio range beacon [VOR]] Relationship to: DESIGNATED_POINT [Significant point is at Designated point] Relationship to: NDB [Significant point is at Non-directional radio beacon [NDB]] Relationship to: DME [Significant point is at Distance measuring equipment [DME]] Relationship to: MKR [Significant point is at Marker [MKR]]

Spd	SpecialDateType	SPECIAL_DATE	Relationship to: ORG_AUTH [Special dates is associated with Organisaton or authority] Attribute: codeType Attribute: dateDay Attribute: dateYear
Sns	SpecialNavigationSystem Station-Type	SPEC_NAV_STATION	Relationship to: SPEC_NAV_SYS [Special navigation system station is part of Special navigation system] Attribute: txtName
Sny	SpecialNavigationSystemType	SPEC_NAV_SYS	Attribute: codeType Attribute: codeId
---	SpecifiedCruisingLevelType	PREDEFINED_LVL	---
Ssr	StarServiceType	SERVICE_ON_STAR	Relationship to: STAR [Service on STAR is furnished on Standard instrument arrival [STAR]] Relationship to: SERVICE [Service on STAR is also Service]
Sia	StarType	STAR	Relationship to: AD_HP [Standard instrument arrival [STAR] is for Aerodrome / Heliport] Attribute: txtDesig Attribute: codeCatAcft Attribute: codeTransId
Sse	StarUsageType	STAR_USAGE	Relationship to: STAR [Usage of particular STAR is based on Standard instrument arrival [STAR]] Attribute: codeRteAvbl
Swy	StopwayType	SWY	Relationship to: RWY_DIRECTION [Stopway [SWY] is designed for RWY direction]
Tcn	TacanType	TACAN	Attribute: codeId Attribute: geoLat Attribute: geoLong
Tli	TacanUsageLimitationType	TACAN_USAGE_LIMIT	Relationship to: TACAN [] Attribute: codeType Attribute: codeComp
Tcp	TaxiwayCentreLinePositionType	TWY_CLINE_POINT	Relationship to: TWY [Position on the centre line of a TWY is on the centreline of Taxiway [TWY]] Attribute: geoLat Attribute: geoLong
Thp	TaxiwayHoldingProcedureType	TWY_HOLDING_POSITION	Relationship to: TWY_CLINE_POINT [TWY holding position is at Position on the centre line of a TWY]

Tly	TaxiwayLightingSystemType	TWY_LGT_SYS	Relationship to: TWY [TWY lighting system is owned by Taxiway [TWY]] Attribute: codePsn
Twy	TaxiwayType	TWY	Relationship to: AD_HP [Taxiway [TWY] is situated at Aerodrome / Heliport] Attribute: txtDesig
---	TimetableNavaidType	TIMETABLE	---
---	TimetableType	TIMETABLE	---
---	TlofGeometryPointType	TLOF_SHAPE_POINT	---
Tlg	TlofGeometryType	TLOF_GEOMETRY	Relationship to: TLOF [Aerodrome / Heliport Surface border is describing the geometry of one Touch down and lift off area [TLOF]]
Tls	TlofLightingSystemType	TLOF_LGT_SYS	Relationship to: TLOF [TLOF lighting system is owned by Touch down and lift off area [TLOF]] Attribute: codePsn
---	TlofSafeAreaGeometryPointType	TLOF_SAFE_AREA_SHAPE_PT	---
Tsg	TlofSafeAreaGeometryType	AD_HP_SURFACE_BORDER	Relationship to: TLOF_SAFE_AREA [Aerodrome / Heliport Surface border is describing the geometry of one TLOF safe area]
Tsa	TlofSafeAreaType	TLOF_SAFE_AREA	Relationship to: TLOF [TLOF safe area is encompassing Touch down and lift off area [TLOF]]
Tla	TlofType	TLOF	Relationship to: AD_HP [Touch down and lift off area [TLOF] is situated at Aerodrome / Heliport] Attribute: txtDesig
Tfr	TrafficFlowRestrictionType	TFC_FLOW_RESTR	Attribute: codeId
Uac	UnitAssociationType	UNIT_ASSOC	Relationship to: UNIT [Unit association is having as child Organisation unit providing services] Relationship to: UNIT [Unit association is having as parent Organisation unit providing services] Attribute: codeType
Uas	UnitContactAddressType	ADDRESS	Relationship to: UNIT [Contact address is for Organisation unit providing services] Attribute: codeType Attribute: noSeq
Uni	UnitType	UNIT	Attribute: txtName
Vor	VorType	VOR	Attribute: codeId Attribute: geoLat Attribute: geoLong

Vli	VorUsageLimitationType	VOR_USAGE_LIMIT	Relationship to: VOR [] Attribute: codeType

Table A.2. Deprecated types

AIXM Feature - name	AIXM complex type	AICM Entity	Natural Keys
Aac	[Deprecated-4.0] AirspaceAssoc-Type	AIRSPACE_ASSOC	Relationship to: AIRSPACE [Airspace association is having as child Airspace] Relationship to: AIRSPACE [Airspace association is having as parent Airspace]
Dln	[Deprecated-4.0] DmeLimitation-Type	DME_LIMITATION	Relationship to: DME [DME - Limitation is limiting Distance measuring equipment [DME]] Attribute: codeType Attribute: valAngleFm Attribute: valAngleTo Attribute: valDistInner Attribute: valDistVerLower
Fao	[Deprecated-4.5] FatoOldType	FATO	Relationship to: TLOF [Final approach and take-off area [FATO] is having Touch down and lift off area [TLOF]]
Nln	[Deprecated-4.0] NdbLimitation-Type	NDB_LIMITATION	Relationship to: NDB [NDB limitation is limiting Non-directional radio beacon [NDB]] Attribute: codeType Attribute: valAngleFm Attribute: valAngleTo Attribute: valDistInner Attribute: valDistVerLower
Tln	[Deprecated-4.0] TacanLimitation-Type	TACAN_LIMITATION	Relationship to: TACAN [TACAN - Limitation is limiting Tactical air navigation beacon [TACAN]] Attribute: codeComponent Attribute: codeType Attribute: valAngleFm Attribute: valAngleTo Attribute: valDistInner Attribute: valDistVerLower
Vln	[Deprecated-4.0] VorLimitation-Type	VOR_LIMITATION	Relationship to: VOR [VOR limitation is limiting VHF omni-directional radio range beacon [VOR]] Attribute: codeType Attribute: valAngleFm Attribute: valAngleTo Attribute: valDistInner Attribute: valDistVerLower

Appendix B. CRC Java Code

B.1. How to execute the crc.java code

To use the class crc.java you must compile the below java code (B.2 Java code of crc.java). After you can execute the class via the following order :

```
%JAVA_HOME%\bin\java –cp %DIR% CRC %STREAM%
```

Where :

- %JAVA_HOME% is the directory where the jdk is installed
- %DIR% is the directory where the class crc.class can be found
- %STREAM% is the input stream used for the CRC calculation

B.2. Java code of crc.java

```
/*
 * Copyright (c) 2004, EUROCONTROL
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without modification,
 * are permitted provided that the following conditions are met:
 *   * Redistributions of source code must retain the above copyright notice, this
 *     list of conditions and the following disclaimer.
 *   * Redistributions in binary form must reproduce the above copyright notice,
 *     this list of conditions and the following disclaimer in the documentation
 *     and/or other materials provided with the distribution.
 *   * Neither the name of EUROCONTROL nor the names of its contributors may be used
 *     to endorse or promote products derived from this software without specific
 *     prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED.
 * IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT,
 * INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
 * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
 * WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGE.
 */

/**
 *
 * Encapsulates CRC computation.
 *
 * For reference, the parameters for this implementation, relative to the standard
 * parameters for CRC testing as described, for example,
 * <a href="http://www.gpfn.sk.ca/~rhg/csc8550s02/crc.html">here</a> or
 * <a href="http://rcswww.urz.tu-dresden.de/~sr21/crc.html">here</a> are as follows:
 * <ul><li>CRC order: 32</li>
 * <li>CRC polynomial: 814141AB</li>
 * <li>Initial CRC value: 00000000 (and the direct/non-direct distinction is
 * irrelevant).
 * </li>
```

```
 * <li>Final XOR value:  00000000 (and the CRC is not reversed before the final XOR).
 *  </li>
 * <li>The check box "Reverse input data bytes" should not be checked.  </li>
 * </ul>
 * With these settings, the CRC calculator at the first web address above agrees on
 * the computed CRC for the test data supplied in the ReadMe.doc file supplied by
 * Eurocontrol with the original version of this code.
 *
 *  @author Serge Hennaux for Pulsar Consulting S.A., Belgium
 *  @author Jo Calder for Mekon Ltd, UK.
 *
 */

/*
 * This version is a signficant refactoring and rationalization of the class CRC.java
 * released by Eurocontrol on 11/11/2004.
 *
 * Summary of changes by Mekon Ltd.
 * <ul><li>the class is encapsulated so as to be serially reusable and not to
 * require synchronization in a multi-threaded environment</li>
 * <li>standard Java conventions for the naming of variables and methods have been
 * adopted</li>
 * <li>data structures declared but not used effectively have been removed</li>
 * <li>data structures of length 33 have been normalized to length 32</li>
 * <li>code for mapping between character and boolean representations has been
 * simplified.</li>
 * </ul>
 *
 * These changes to CRC.java are copyright (C) Mekon Ltd, 2004.  Redistribution and
 * use of this version is permitted under the terms specified by Eurocontrol as
 * above, provided this second copyright notice is also retained (in the case of
 * source code distribution) or reproduced (in the case of binary distribution).
 *
 */


public class CRC {


  /**
   * The polynomial for CRC32 as an array of booleans.  The least significant bit
   * has the index 0.
   */

      public final static boolean CRC32POLY[] = {

        true,        /* 1 */
      true,        /* x */
      false,       /* x2 */
      true,        /* x3 */
      false,       /* x4 */
      true,        /* x5 */
      false,       /* x6 */
      true,        /* x7 */
      true,        /* x8 */
      false,       /* x9 */
      false,       /* x10 */
      false,       /* x11 */
      false,       /* x12 */
      false,       /* x13 */
      true,        /* x14 */
      false,       /* x15 */
      true,        /* x16 */
```

```
    false,       /* x17 */
    false,       /* x18 */
    false,       /* x19 */
    false,       /* x20 */
    false,       /* x21 */
    true,        /* x22 */
    false,       /* x23 */
    true,        /* x24 */
    false,       /* x25 */
    false,       /* x26 */
    false,       /* x27 */
    false,       /* x28 */
    false,       /* x29 */
    false,       /* x30 */
    true         /* x31 */
      };


private final static int POLYNOMIALLENGTH = CRC32POLY.length;

public static void main(String[] args) {

  if (args.length != 1) {
    System.out.println("Usage CRC [string to calculate CRC]");
    System.exit(1);
  }

  boolean[] crc = computeCRC(args[0]);

  show(crc);
}

/**
 *
 * Computes a CRC value for the given input.  Returns an array of booleans.
 * @param input
 *
 */
private static boolean[] computeCRC(String input) {
  char[] toProcess = input.toCharArray();
  boolean[] toReturn = new boolean[POLYNOMIALLENGTH];

  for (int z = 0; z < input.length(); z++) {
    boolean[] characterAsBoolean = charToBoolean(toProcess[z]);
    processChar(toReturn, characterAsBoolean);
  }
  return reverse(toReturn);
}

/**
 * Processes the next character from the input.
 */
private static void processChar(boolean[] remainder, boolean[] characterAsBoolean)
 {
  // remainder is always of length 32.
  for (int k = 0; k < 8; k++) {
    int count = POLYNOMIALLENGTH-1;
    boolean value = remainder[count] ^ characterAsBoolean[k];
    for (;count > 0;count--) {
      if (CRC32POLY[count] == true) {
        remainder[count] = remainder[count - 1] ^ value;
      } else {
        remainder[count] = remainder[count - 1];
```

```
      }
    }
    // notice that count at this point is always zero.
    remainder[count] = value;
  }
}

/**
 * Computes a hexadecimal CRC value for the supplied input.  Returns the
 * hexadecimal representation of the CRC for the supplied input.  The result will
 * contain no lower case letters.  (Cf AIXM definition of "Data Types for
 * Cyclic Redundancy Check Values (CRCV)".)
 * @param input
 * @return
 */
public static String computeCRCHex(String input) {
  String a = booleanToHex(reverse(CRC32POLY));
  boolean[] raw = computeCRC(input);
  return booleanToHex(raw);

}

/**
 * Reverses the booleans in an array.
 */
private static boolean[] reverse(boolean[] toProcess) {
  boolean[] toReturn = new boolean[toProcess.length];

  for (int k = toProcess.length-1; k > -1; k--) {
    toReturn[(toProcess.length-1)-k] = toProcess[k];
  }
  return toReturn;
}

// only used in main()
private static void show(boolean[] crcValue) {
  String buf = "";
  System.out.print("Resultat : ");
  for (int j = 0; j < crcValue.length; j++) {
    if (crcValue[j] == true)
      buf += "1";
    else
      buf += "0";
  }
  System.out.println(buf);
  System.out.println(booleanToHex(crcValue));
}

/**
 * Computes a boolean[8] representation of a char.  The bit ordering is from more
 * to less significant.
 * i.e.  <pre>string "4" = char 0x34 = 0011 0100 = [false, false, true, true,
 * false, true, false, false] </pre>
 *
 *
 */
private static boolean[] charToBoolean(char val) {
  boolean tB[] = new boolean[8];
  for (int i = 0; i < 8; i++) {
    if (bitIsSet(val, i)) {
      tB[7-i] = true;
    }
  }
```

```
    return tB;
  }


  /**
   * Indicates whether the ith bit of the supplied char is set (== 1).
   */
  private static boolean bitIsSet(char val, int i) {
    return (val & (int) Math.pow(2, i)) > 0;
  }

  /**
   * Computes the hexadecimal representation of the array.  The result will contain
   * no lower case letters.  (Cf AIXM definition of "Data Types for Cyclic
   * Redundancy Check Values (CRCV)".)
   */
  private static String booleanToHex(boolean[] array) {
    StringBuffer sb = new StringBuffer();
    for (int i = 0; i < array.length; i = i+4) {
      sb.append(decodeBoolean(i, array));
    }
    return sb.toString().toUpperCase();
  }

  /**
   * Computes the hexadecimal representation of the 4-bit segment of the array
   * commencing at the given offset.
   */
  private static String decodeBoolean(int offset, boolean[] array) {
    char toReturn = 0x0;
    for (int j = 0; j < 4; j++) {
      if (array[offset+(3-j)]) {
        toReturn += Math.pow(2, j);
      }
    }
    return Integer.toHexString(toReturn);
  }

}
```

Appendix C. Character Encoding Issues

C.1. Introduction

This appendix is intended as an introduction to the concept of character encoding, explaining what it is and why it is important, particularly with regard to XML documents.

C.2. What is a Character Encoding?

Simply put, a character encoding is a standard by which computers process character data. In order to fully appreciate what a character encoding is, and how they can potentially lead to problems, it is beneficial to know how they have evolved. Before the personal computer became widespread, most computer software processed character data using a standard called ASCII (American Standard Code for Information Interchange). ASCII represented every character in the English alphabet, plus punctuation characters etc. using the numbers between 32 and 127. Codes below 32 represented "unprintable" characters such as line-feeds. However, because there are eight bits to a byte, this meant that the codes from 128 to 255 were unallocated.

Table C.1. Printable ASCII Characters

	!	".	#	$	%	&	'	(	)	*	+	,	-	.	/
0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	

Inevitably, those codes were used in different ways by different people. And as computers became globally available, non-English-speaking users utilised those codes for their own alphabet characters. This led to a number of different encoding standards (known as OEM) being developed around the world, mostly as extensions of ASCII.

C.3. Unicode

Unicode is a standard, by the Unicode Consortium, which defines a character repertoire and character code intended to be fully compatible with ISO 10646, and an encoding for it. ISO 10646 is more general (abstract) in nature, whereas Unicode "imposes additional constraints on implementations to ensure that they treat characters uniformly across platforms and applications", as they say in section Unicode & ISO 10646 of the Unicode FAQ. Unicode was originally designed to be a 16-bit code, but it was extended so that currently code positions are expressed as integers in the hexadecimal range 0..10FFFF (decimal 0..1 114 111). That space is divided into 16-bit "planes". Until recently, the use of Unicode has mostly been limited to "Basic Multilingual Plane (BMP)" consisting of the range 0..FFFF. The ISO 10646 and Unicode character repertoire can be regarded as a superset of most character repertoires in use. However, the code positions of characters vary from one character code to another.

Originally, before extending the code range past 16 bits, the "native" Unicode encoding was UCS-2, which presents each code number as two consecutive octets m and n so that the number equals 256m+n. This means, to express it in computer jargon, that the code number is presented as a two-byte integer. According to the Unicode consortium, the term UCS-2 should now be avoided, as it is associated with the 16-bit limitations. UTF-32 encodes each code position as a 32-bit binary integer, i.e. as four octets. This is a very obvious and simple encoding. However, it is inefficient in terms of the number of octets

needed. If we have normal English text or other text which contains ISO Latin 1 characters only, the length of the Unicode encoded octet sequence is four times the length of the string in ISO 8859-1 encoding. UTF-32 is rarely used, except perhaps in internal operations (since it is very simple for the purposes of string processing). UTF-16 represents each code position in the Basic Multilingual Plane as two octets. Other code positions are presented using so-called surrogate pairs, utilizing some code positions in the BMP reserved for the purpose. This, too, is a very simple encoding when the data contains BMP characters only. Unicode can be, and often is, encoded in other ways, too, such as the following encodings:

UTF-7: Each character code is presented as a sequence of one or more octets in the range 0 - 127 ("bytes with most significant bit set to 0", or "seven-bit bytes", hence the name). Most ASCII characters are presented as such, each as one octet, but for obvious reasons some octet values must be reserved for use as "escape" octets, specifying the octet together with a certain number of subsequent octets forms a multi-octet encoded presentation of one character. There is an example of using UTF-7 later in this document.

UTF-8: Character codes less than 128 (effectively, the ASCII repertoire) are presented "as such", using one octet for each code (character) All other codes are presented, according to a relatively complicated method, so that one code (character) is presented as a sequence of two to four octets, each of which is in the range 128 - 255. This means that in a sequence of octets, octets in the range 0 - 127 ("bytes with most significant bit set to 0") directly represent ASCII characters, whereas octets in the range 128 - 255 ("bytes with most significant bit set to 1") are to be interpreted as really encoded presentations of characters.

IETF Policy on Character Sets and Languages (RFC 2277) favors UTF-8. It requires support to it in Internet protocols (and doesn't even mention UTF-7). Note that UTF-8 is efficient, if the data consists dominantly of ASCII characters with just a few "special characters" in addition to them, and reasonably efficient for dominantly ISO Latin 1 text.

C.4. Why Character Encoding is Important

Problems with character encoding can arise when files generated on one system are transferred and interpreted on another system - basically when data is exchanged. Because the upper 128 bytes were allocated differently between different encodings (usually along the same lines as language, but sometimes there were even different encodings within a single country), data could be interpreted incorrectly. For example, on some computers the character code 130 would display as é (e with an acute accent), but on computers sold in Israel it was the Hebrew letter Gimel.

This misinterpretation could have a range of effects depending on the software. At one end of the scale this could be manifested as an application incorrectly displaying or printing a character - this is a typical problem for word processors. At the other end of the scale an unexpected character code may be sufficient to crash an application. Although the latter may appear to be the more severe, in terms of data integrity it is the potentially undetected error that can be more problematic. This can happen quite easily if data is incorrectly interpreted and then rewritten or saved using another encoding. Equally problematic is the case in which the declared encoding is different to the actual encoding.

Obviously this has major implications for aeronautical data exchange if ICAO standards for the integrity of data are to be adhered to. Within AIXM, these kind of problems could effect all elements which do not restrict their content to simple characters. These are textual descriptions and textual remark.

C.5. Specifying Character Encodings in XML

The solution to the problem, or at least the beginning of the solution, is to know which encoding a string uses, whether it be in a file, in memory or in an e-mail. The remainder of the solution is for software to

be able to identify that encoding and to interpret it correctly, or handle failure gracefully. Whenever textual data is exchanged between systems, the sender and the recipient should agree on the character encoding used. The easiest way to do this is to specify the character encoding used in a header statement (of a sort dependent on the file type).

XML, being designed for data exchange, requires a character encoding be specified in a header statement in the first line of a file, as shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
```

XML typically uses UTF-8, a Unicode encoding. UTF-8 allows the inclusion of special characters from a multitude of language sets.

Of course, there is the potential problem that a piece of software would not know how to interpret such a header statement, and would therefore be unable to interpret the file. Because the header is always written in "basic" ASCII, without any kind of special characters, it is assumed that any parser would be able to read it. Once the encoding is known, the file can be processed fully and correctly.

Therefore, it is very important that the declared encoding of an AIXM message/file **is the actual encoding** used in the message/file.

- End of document -