AIXM 5.1 - Business Rules (data verification)

Using SBVR and Schematron

***Document Status***: Draft

***Publication date***: 08 November 2016

***version***: 0.7

**Aeronautical Information Exchange Model**

**(AIXM)**

***Copyright****: 2016 - EUROCONTROL and Federal Aviation Administration*

*All rights reserved.*

*This document and/or its content can be downloaded, printed and copied in whole or in part, provided that the above copyright notice and this condition is retained for each such copy.*

*For all inquiries, please contact:*

*Eduard POROSNICU - eduard.porosnicu@eurocontrol.int*

*Deborah COWELL - deborah.cowell@faa.gov*

|  |  |  |  |
| --- | --- | --- | --- |
| ***Edition*** | ***Status*** | ***Issue Date*** | ***Reason for Change*** |
| 0.1 | Working Draft | 25 July 2013 | Initial working draft. Derived from previous work by Pulsar Consulting (under contract for Eurocontrol) and from the “Guidance on Writing AIRM Constraints” - SESAR Project 08.01.03 deliverable, which is itself based on the previous Eurocontrol worl |
| 0.2 | Working Draft | 05 August 2013 | Updated after initial attempts to use the instructions and after discussion with group members, in particular issues raised by Michal Kadlec |
| 0.3 | Draft | 09 June 2014 | First public release, intended for a wider review. Also including an introductory part explaining the need for business rules, the experience of the previous AIXM model version and the approach proposed for AIXM 5 |
| 0.4 | Draft | 15 June 2015 | Second public release. Minor editorial changes. Some new keywords added. Further details about how to write paths to target NounConcepts. First examples of profiles. |
| 0.5 | Draft | 15 September 2015 | Updated public release. No significant changes to the document, just small editorial corrections. Issued together with version 0.5 of the rules. |
| 0.6 | Draft | July 2016 | Updated version. Editorial adjustments, such as grouping together all explanations related to the use of “has” in fact types. Additional explanations about the use of the SBVR standard to the AIXM Business Rules project. Inclusion of the “different instances” notation for NounConcepts. Removal of the special notation for <<choice>> classes. |
| 0.7 | Draft | 08 November 2016 | Public release, in relation with the publication of the Business Rules set 0.7. Additional explanation about the use of specialised classes as NounConcept. |

Contributors

This document reflects the contribution of technical and operational aeronautical information specialists, in particular from industry. Their contribution is hereby acknowledged.

|  |  |
| --- | --- |
| ***Name*** | ***Organisation*** |
| CLAY Michael | Frequentis |
| CHUMAKOV Marina | CNA (under contract for US FAA) |
| DIELS-GRABSCH Volker | m-click.aero |
| DRUARD Sophie | Pulsar Consulting |
| ECHTERHOFF Johannes | Interactive Instruments |
| FABBRI Davide | IDS |
| GERMAIN Francois | Thales |
| GHENCEA Andrei | Eurocontrol (trainee) |
| HAAS Markus | Comsoft |
| d’HULST Jean-Christophe | Pulsar Consulting |
| KADLEC Michal | Avitech |
| POROSNICU Eduard (editor) | Eurocontrol |
| RAZOV Aleksandr | Monitor Soft |
| SANCHEZ Antonio | GroupEAD |
| SCHNEIDER Markus | m-click.aero |
| SCHEUCHER Wolfgang | Solitec |
| SUZIC Robert | Eurocontrol |
| TANG Yauwu | MITRE Corporation |
| WALTL Michael | Frequentis |
| WILSON Scott | Eurocontrol |

Contents

[Executive Summary 7](#_Toc466369886)

[1. Introduction 8](#_Toc466369887)

[1.1. The need for Business Rules 8](#_Toc466369888)

[1.2. How to read this document 8](#_Toc466369889)

[2. Rules definition using SBVR 9](#_Toc466369890)

[2.1. Introduction 9](#_Toc466369891)

[2.2. SBVR Profile for AIXM 9](#_Toc466369892)

[2.2.1. SBVR concepts 9](#_Toc466369893)

[2.2.2. NounConcept - special situations 12](#_Toc466369894)

[2.2.3. Additional Fact-types 13](#_Toc466369895)

[2.2.4. Logical Operations 15](#_Toc466369896)

[2.2.5. Quantification 15](#_Toc466369897)

[2.2.6. Modality 16](#_Toc466369898)

[2.2.7. Additional SBVR keywords 17](#_Toc466369899)

[2.3. Methodology 19](#_Toc466369900)

[3. Schematron test implementation 21](#_Toc466369901)

[3.1. Introduction 21](#_Toc466369902)

[3.2. Schematron code - technical aspects 21](#_Toc466369903)

[3.2.1. XPath version 21](#_Toc466369904)

[3.2.2. XPath compatibility 21](#_Toc466369905)

[3.2.3. Use of Java extensions 21](#_Toc466369906)

[3.2.4. Feature references in the Schematron code 22](#_Toc466369907)

[3.2.5. Content of the Excel file 22](#_Toc466369908)

[3.3. Practical verification of AIXM data sets 22](#_Toc466369909)

[3.3.1. Testing 22](#_Toc466369910)

[4. AIXM Business Rules data set 24](#_Toc466369911)

[4.1. Excel file 24](#_Toc466369912)

[4.2. Profiles 24](#_Toc466369913)

[4.2.1. EAD Data Provider 24](#_Toc466369914)

[4.2.2. Event profile 24](#_Toc466369915)

[4.2.3. Event\_FAA profile 24](#_Toc466369916)

[4.2.4. Minimum air navigation data set profile 24](#_Toc466369917)

[4.2.5. Full air navigation data set profile 25](#_Toc466369918)

[5. References 26](#_Toc466369919)

[Annex A - License and Disclaimer 27](#_Toc466369920)

# Executive Summary

This document defines the AIXM 5.1 “business rules” concept, in particular how the rules are modelled and how they are provided to system developers. Such rules can be used to verify if AIXM XML data sets that are syntactically valid (against the AIXM XML Schema) are also semantically correct and can be used in confidence for a particular application.

The objective of the AIXM Business Rules project is to provide, in a standard format, an exhaustive set of operational constraints that may apply to (such as the requirements minimal data accuracy, etc.) or may be reflected onto (such as the rules for frequency pairing for VHF navaids, etc.) the aeronautical data. A second objective is to capture structural rules that are specific to the AIXM context (such as the relation between the type of TimeSlice and its validity period, etc.) and which are not enforced in the AIXM schema.

While the objective is to provide an exhaustive set of rules, only a subset of the rules might be relevant and needs to be enforced/checked for a particular application. For example, a rule that concerns mandatory feature properties might indicate that the frequency of a navaid is a required value. While for a charting or air navigation support application this is a necessary constraint, for a flight planning application this is not necessary. Therefore, profiles (subsets) of the AIXM Business Rules will be proposed for particular applications and/or AIXM user communities.

The Semantics of Business Vocabulary and Business Rules (SBVR) [[1]](#bx40i6rblbnl) standard is applied for writing the AIXM business rules, in relation with the AIXM UML logical data model. This means that the AIXM classes and their properties (attributes and associations), together with their definitions and data types, provide the the “business vocabulary” that is used as the basis for the definition of the AIXM business rules.

This document provides an ‘SBVR profile’, which is tailored to the AIXM needs and which is documented as a number of concepts and conventions applied in the writing of the AIXM business rules. This document is not intended as an exhaustive introduction in SBVR; it is mostly a “primer” document, giving the essential elements that need to be understood in order to:

* contribute to the writing of the AIXM 5.1 Business Rules in compliance with the SBVR methodology;
* read and understand the AIXM 5.1 Business Rules by those interested to review and/or implement such rules in a given system.

In addition to the SBVR definition, the AIXM Business Rules are also provided, where possible, as Schematron [[4]](#wxo2ufcod0ee) code. This is done for two reasons:

* in order to verify that the SBVR description of the rule is sufficiently clear and unambiguous in order permit its actual implementation as software code;
* to offer practical means by which an AIXM data set could be verified against the business rules, using software readily-available on the Web, most of it for free.

However, it shall be kept in mind that this Schematron encoding is missing for some rules for which the encoding is expected to be very complex or even impossible in Schematron. Any use of the Schematron coding provided with the AIXM Business Rules is subject to the license and disclaimer copied in [Annex A](#_cb4vzwi4gzj5).

# Introduction

## The need for Business Rules

The Semantics of Business Vocabulary and Business Rules (SBVR) [[1]](#bx40i6rblbnl) standard defines business rules as being “*a law or principle that operates within a particular sphere of knowledge, describing, or prescribing what is possible or allowable*” and that is also under the jurisdiction of a particular business. In the AIXM case, that jurisdiction is the aeronautical information domain. The aim of defining the AIXM Business Rules is to define what is possible or allowed in an AIXM data set, in particular with regard to data values.

The AIXM UML model defines the information items that are in the scope of the “aeronautical information” domain using UML class diagrams. This includes definitions for classes, attributes and associations between classes. For attributes, the constraints expressed as lists of values, range of values, pattern are also included in the AIXM UML model.

The AIXM XML schema is derived from the UML model and defines elements that correspond to the AIXM model classes, attributes and association role names. The values of XML elements and data types derived from UML attributes are constrained based on the data types defined in the UML model - list of values, data ranges and/or patterns.

More complex constraints, such as dependencies between the values of different attributes (sometimes in different classes), detection of ‘out of range’ values, mandatory properties for class of objects, etc. are not included in the UML model and do not appear in the XML Schema. Thus the need to document such more complex constraints as “business rules”.

Not all the AIXM Business Rules defined in this document are equally applicable in all communities of AIXM users. The most obvious examples are mandatory properties for AIXM features - in a flight planning community, there is no need to include in an AIXM data set the frequency/channel of a navigation aid. On the other side, such attributes are mandatory for data sets provided to an air navigation community. Therefore, the aim of this document is to document the largest possible set of candidate AIXM business rules, from which profiles for a particular community can be extracted. Such profiles are proposed in the [AIXM Business Rules data set](#_2cbw0r9eqh2d) chapter and the related Annexes.

## How to read this document

Operational/domain experts are recommended to read the [Executive Summary](#_t1fm751xtpp0), the [Introduction](#_i9159vu5ywh2) and [Rules definition using SBVR](#_uo0i0o1btd5a) (in particular the [Methodology](#_5cii9re7zovi) section). Then, go directly to the [AIXM Business Rules data set](#_2cbw0r9eqh2d) chapter.

The [Schematron reference implementation](#_4aoeu958dhze) chapter is intended for developers. However, it should be kept in mind that the use of Schematron is not the only option available for the implementation of the AIXM 5.1 business rules. The Schematron code will provided where possible (as an estimation, for 80% of the rules) as an example only.

# Rules definition using SBVR

## Introduction

The Semantics of Business Vocabulary and Business Rules (SBVR) [[1]](#bx40i6rblbnl) standard is used for the writing of the AIXM business rules in relation with the AIXM UML logical data model. This means that the AIXM classes and their properties (attributes and associations), together with their definitions and data types, provide the the “business vocabulary” that is used as the basis for the definition of the AIXM business rules.

This section defines an ‘SBVR profile’, which is tailored to the AIXM needs and which is documented as a number of concepts and conventions applied in the writing of the AIXM business rules. Two previous projects have provided input material for this document:

* work done by Pulsar Consulting, under contract for Eurocontrol, which is available on the AIXM wiki: Use of SBVR for AIXM [[2]](#kix.69aiguo4ozhz);
* Guidance on Writing AIRM Constraints [[3]](#qiucxh6y8iu8) provided by SESAR Project 08.01.03.

This document is not intended as an exhaustive introduction to SBVR. Apart from reading the full SBVR specification [[1]](#bx40i6rblbnl), the following ‘SBVR essentials” reading list is suggested:

* the first pages in Chapter 10 of the OMG SBVR specification [[1]](#bx40i6rblbnl) version 1.1;
* this [presentation available on the Web](https://www.researchgate.net/post/What_are_Semantics_of_Business_Vocabulary_and_Business_Rules_SBVR_Models_and_what_are_Information_System_Models) (in particular slides 5 and 6).

## SBVR Profile for AIXM

### SBVR concepts

The following table introduces the concepts used as part of the SBVR Profile for AIXM, including their graphical notation. Note that the use of some graphical notations is simplified as compared to the SBVR standard, such as double underline which is replaced with simple underline. This facilitates the use of standard office tools (such as Microsoft Excel) for the provision of the AIXM set of Business Rules.

|  |  |
| --- | --- |
| Concept | SBVR Definition:   * Unit of knowledge created by a unique combination of characteristics.   Representation in profile:   * Not represented – too general |
| **NounConcept** | SBVR Definition:   * Concept that is the meaning of a noun or noun phrase.   Representation in profile (i.e. what AIXM items may appear as NounConcept according to the SBVR profile):   * Represented by AIXM UML Classes and Properties, meaning that AIXM Class Name, Role Name or Attribute Name may appear as **NounConcept**. Due to specificities of the AIXM UML model, some special constructs are explained in the [NounConcept - Special situations](#_eeno4xdhez6v) section.   Style: **Bold, underlined** and **UpperCamelCase** or **lowerCamelCase** (depending on how the noun concept appears in the UML model). If several nouns are concatenated, then they should be separated by a dot (“.”) symbol.  Colour: **#008080**  Example: **AirportHeliport**, **Airspace.type**, **AirportHeliport.name**, **Runway.associatedAirportHeliport** |
| *Verb-concept*  *also known as*  *“Fact type”* | SBVR Definition:   * Concept that is the meaning of a verb phrase that involves one or more noun concepts and whose instances are all actualities.   Fact is a proposition that is taken as true.    <Fact-type> :: = <concept1> <verb> <concept2>    Verb Concepts:   * Business Facts * Relations amongst Concepts     Representation in profile (i.e. what AIXM items may appear as Verb-concept according to the SBVR profile):   * Represented by Name of an AIXM UML association.     Style: *italic*  Colour: *#0000ff*  Example: **AirportHeliport** *has* **name**, **Runway** *hasSurfaceDescribedBy* **SurfaceCharacetistics**. |
| Name | SBVR Definition:   * Name is a concept that corresponds to only one object [thing] used for the designation of an individual concept — a name. Names tend to be proper nouns.   Representation in profile (i.e. what AIXM items may appear as Name according to the SBVR profile):   * Represents UML Instances, Slots, Enumeration literals, and their assigned Properties * CodeList values     Style: surrounded by ‘simple quotes’  Colour: #339966  Example: ‘Sweden’, ‘CTA’, ‘CTA\_P’, ‘YES’, ‘NIL’  *Note: this style is different from the standard SBVR style for Name concepts (double underline) for a practical reasons - the double underline is not well supported on all the platforms and in all the software applications used for the development and maintenance of the AIXM Business Rules.*  ‘NIL’ is an additional Name concept used in the SBVR profile for AIXM. It indicates a void property (i.e. a property that does not have a value).  Sometimes, it is necessary to provide a “list of names” (such as ‘A’, ‘B’ or ‘C’) in order to indicate the allowed values for a property (which appears as a NounConcept in SBVR rules). The following style shall be used: opening-bracket, followed by each Name-value surrounded by simple quotes, separated by comma and ending with closing-bracket.  Example: (‘A’,‘B’,‘C’) (*note that the quotes and the brackets are also formatted, just to keep the editing simple*) |
| keyword | SBVR Definition   * are used to construct statements – the words that can be combined with other designations to form statements and definitions, see sections on [Logical Operations](#_dvdg981ah2s8), [Quantification](#_azany0a9mnmj), [Modality](#_7v6pywx3vgli) and [Additional SBVR keywords](#_dfvkvjxr1wm7), all these being part of the keyword concept.     Representation in profile:   * No particular mapping to UML model elements   Style: Usual text format  Colour: #ff9900  Example: Each - referring to universal quantifier. |

### NounConcept - special situations

When writing AIXM business rules in SBVR, apart from the simple cases where a class/role/attribute name is used, there are also some special cases, when several classes/associations are involved, as detailed in the following table.

|  |  |
| --- | --- |
| **Special case** | **NounConcept construction rules** |
| ***Concatenation*** - when necessary to identify an UML model element that is remote from the current class. For example, a rule defined for the **AirportHeliport** class that needs to refer to the **name** attribute of the associated **City** class. | The “.” symbol is used as separator between the concatenated noun concepts. The relative path needs to include all the intermediate association role names and class names, as in the following example: **servedCity.City.name**.  *Note that the separator is also formatted, just to keep the editing simple, although it is not part of respective noun concept.* |
| ***Association classes*** - Some associations are qualified by a number of properties depicted inside an ‘association class’. For example, the association between Airspace and another AirspaceVolume includes the AirspaceGeometryComponent association class. | When the NounConcept is inside the association class, the path includes first the name of the association role, concatenated with the association class name and with the property name concerned. For example: **geometryComponent.AirspaceGeometryComponent.operation.**  When the NounConcept is in the target association class or beyond, the path includes first the name of the association role, concatenated with the association class name, then the name of a fictitious theMyClass property name followed by the target class name. For example: **geometryComponent.AirspaceGeometryComponent.theAirspaceVolume.AirspaceVolume.lowerLimit.** |
| ***Specialisation*** - Inheritance is frequently used in the AIXM UML model. The class that is specialised is always stereotyped <<abstract>> and does not appear as such in the AIXM XML Schema. Only its non-abstract specialisations appear in the Schema. Inheritance could occur with more than one level of specialisation (for example, the Service class) | In general, the class that is specialised is used as NounConcept, followed by the “*specialisation*” fact-type and by the name of the derived class (also a NounConcept). For example: “... **NavaidEquipment** *specialisation* **TACAN**…”.  When there is no risk for confusion, the properties of the abstract (generalised) class are used as if they were directly properties of the specialised class. For example: “...**TACAN.location**...”, although location is a property of the <<abstract>> NavaidEquipment, which is inherited by the TACAN class. |
| ***Data type attributes*** - For example, most “Val…” data types come with a “uom” attribute. | The uom attribute appears directly in the path after the attribute name, prefixed with a ‘.’. For example, the rules that require the mandatory presence of a uom value for an elevation attribute use the following noun concept path: **elevation.uom**. |
| ***TimeSlice*** - Not present in the UML model, but appearing directly in AIXM XML Schema, based on the [AIXM Temporality Concept](http://www.aixm.aero/gallery/content/public/AIXM51/AIXM%20Temporality%201.0.pdf) (see in particular section 2.8 in that document). For example, rules that check the correctness of the validTime with regard to the type of TimeSlice. | The following properties introduced for each and every AIXM <<feature >> class may appear as noun concepts:   * **interpretation** * **sequenceNumber** * **correctionNumber** * **validTime** * **featureLifeime**   Note: in order to avoid confusions, these properties appear in the SBVR formulation as properties of **timeSlice.AIXM Feature TimeSlice**. |
| ***GML elements*** - Present only partially in the UML model, through the GM\_… classes. However, the details of these classes are missing in AIXM 5.1. For example, positions are encoded with the gml:pos element, while surfaces use much deeper structures, such as gml:Surface.gml:... | Specific GML 3.2.1 schema elements appear as NounConcepts, such as:   * **….Curve.segments.GeodesicString** |
| ***Different instances*** *-* When a rule concerns two different instances of the same NounConcept and it is necessary to distinguish between the two. For example, a rule for the uniqueness of 5-letter designators needs to refer to different instances of the DesignatedPoint class. | A number in square brackets is used in order to distinguish the two, such as in the following example: **DesignatedPoint**[1].  Note that this notation is different from the one used in the SBVR standard (subscript, such as **DesignatedPoint**1) for a practical reason - subscripts are not well supported in all systems and on all the platforms used for the definition and maintenance of the AIXM Business Rules. This situation might change in future and the graphical notation might be aligned with the one used in the SBVR standard. |

### Additional Fact-types

The following table introduces additional fact-types used in SBVR AIXM Profile. They are ‘additional’ in the sense that they do not appear explicitly as associative facts in the model.

Note: *the fact-types marked with an \* sign are not used for the moment. They are kept for completeness sake, some might become necessary later.*

|  |  |
| --- | --- |
| *is-property-of* | associative fact type that is defined with respect to a first given concept and a second given concept such that each instance *of* the fact type *is* an actuality that an instance *of* the first concept constitutes an essential quality of an instance *of* the second concept |
| *has* | describes the same associative fact as “*is-property-of*”, but in the opposite direction: an instance of the second concept constitutes an essential quality of an instance of the first concept   * ***“****has****” as fact type for the attributes of a class:*** there is an implicit association (fact type) between a class and each of its attributes. This can be expressed using both the *is property of* or the *has* fact type (depending on direction in which the rule is written). For example: “**name** *is property of* **AirportHeliport**” or “**Airspace** *has* **type**” * ***“****has****” as fact type for the properties of an associated <<object>> class:*** The AIXM UML model makes a difference between <<feature>> and <<object>> classes (stereotype). An <<object>> is seen as a complex property of another class. Therefore, "*has*" can be used as a simplified fact-type in situation such as the following:   + **Airspace** *has* **geometryComponent.AirspaceVolume.upperLimit** (“*has*” followed by the role name of the target class is used instead of the feature-object association name “*hasGeometry*”)   + **AirportHeliport** *has* **servedCity.City.name**  (“*has*” followed by the role name of the target class is used instead of the feature-object association name “*serves*”)   It is important to note that "*has*" as fact-type cannot be used when referring to the properties of another <<feature>> class, because the two exist independently from each other. They have a different timeline and traversing the association requires also selecting the relevant TimeSlices. This aspect is sufficiently important to be preserved in the SBVR text. In such situations, the association name must be used:   * + **Runway** *isSituatedAt* **AirportHeliport** (this cannot be replaced by “*Runway has associatedAirportHeliport.AirportHeliport*”!) |
| *is-descendant-of* | describes a deeper associative fact than “*is-property-of*”; an instance of the first concept constitutes a property of a property (and so on…) of an instance of the second concept |
| *has descendant* | describes the same associative fact as “*is-descendant-of*”, but in the opposite direction |
| *specialisation* | categorisation fact type (used to target a specific non-abstract class, that is a specialisation of a parent class; for example **NavaidEquipment** *specialisation* **VOR**)  or  contextualisation fact type |
| *categorization\** | a particular instance of the concept is also an instance of the category |
| *is-of-type\** | is used for expressing units of measurements (UOM) |

### Logical Operations

The following table introduces the logical operations (e.g. “and”, ”not”, “or”, etc.) used in the SBVR Profile for AIXM. The exact representation of the operations is in the third column (using keywords convention).

Note: *the keywords marked with an \* sign are not used for the moment. They are kept for completeness sake, some might become necessary later.*

|  |  |  |
| --- | --- | --- |
| **Logical Operation** | **SBVR Profile for ATM Representation** | **Meaning** |
| conjunction | p and q | True when all operands are true |
| disjunction | p or q | True when at least one operand is true |
| equivalence\* | p if and only if q | True when operands are either all true or all false |
| exclusive disjunction\* | p or q but not both | One operand is true and the other is false |
| implication | if p then q    The following equivalent formulation is sometimes used in this profile in order to improve the readability of the text:    “Each… shall…” | binary logical operation that operates on an antecedent and a consequent and that formulates that the meaning of the consequent is true if the meaning of the antecedent is true |
| logical negation | not p | True when the operand is false |
| nand\* | not both p and q | true when at least one operand is false |
| nor\* | neither p nor q | true when all operands are false |
| whether or not\* | p whether or not q | binary logical operation that has a consequent and an inconsequent and that formulates that the meaning the consequent is true regardless of the meaning the inconsequent. |

### Quantification

The following table introduces the quantifications (e.g. “at least one”) used as part of the SBVR Profile for AIXM. The second column contains the representation (using keywords convention).

|  |  |  |  |
| --- | --- | --- | --- |
| **quantifier**  **name** | **SBVR Profile for ATM Representation** | **logic**  **symbol** | **description** |
| universal | each x | ∀ | For each and every *x*, taken one at a time |
| existential | at least one x | ∃ | At least one *x* |
| exactly-one | exactly one x | ∃1 | There is exactly one (at least one and at most one) *x* |
| at-most-one | at most one x | ∃0..1 | There is at most one *x* |
| at-most-*n* | at most *n* x | ∃0..*n*  (*n* ≥ 1) | There is at most *n x Note: n* is always instantiated by a number ≥ 1. So this is really a set of quantifiers (*n* = 1, etc.) |
| at-least-*n* | at least *n* x | ∃*n..*  (*n* ≥ 1) | There is at least *n x Note: n* is always instantiated by a number ≥ 1. So this is really a set of quantifiers (*n* = 1, etc.) |
| at-least-2 | more than one x | ∃*n..*  (*n =* 2) | There is at least 2 *x.* |
| exactly-*n* | exactly *n* x | ∃*n*  (*n* ≥ 1) | There is at exactly (at least and at most) *n x Note: n* is always instantiated by a number ≥ 1. So this is really a set of quantifiers (*n* = 1, etc.) |
| numeric range | at least *n* and at most *m* x | ∃*n..m*  (*n* ≥ 1)  (*m* ≥ 2) | There is at least *n* and at most *m x* |

### Modality

The following table introduces the modalities (e.g. “It is obligatory”) used in the SBVR Profile for AIXM. The fourth column details the representation in the profile.

Note: *the keywords marked with an \* sign are not used for the moment. They are kept for completeness sake, some might become necessary later.*

|  |  |  |
| --- | --- | --- |
| **Modality** | **SBVR Profile for ATM** | **Reading (Verbalised as) after applying transformation using negations:** |
| **obligation** | It is obligatory that p  The following formulations are used in this profile in order to distinguish between a strict and loose enforcement of an obligation (based on the ICAO practice):  **p** shall …  **p** should ...  “shall” = obligation with strict enforcement level  “should” = obligation with loose enforcement level | *It is not permitted that not p* |
| **prohibition**  (negation of permission) | It is prohibited that p  In relation with the two forms of obligation (strict and loose enforcement,, based on the ICAO practice), the prohibition formulation can also take two forms:  **p** shall not …  **p** should not **...** | *It is obligatory that not p* |

Note: *Please be aware of the distinction between the use of the phrases “It is obligatory that”/”It is prohibited that” and the pattern “Each...shall/should”. The latter implies that the target entity or attribute has to exist, whereas this is not always the case. Therefore, the negative formulation is preferred in most cases, as it will check the value only when the entity or the attribute is actually present.*

*Example:*

Each Runway with assigned lengthStrip value shall have assigned lengthStrip.uom value equal-to (‘FT’,’M’)

*versus*

It is prohibited that a Runway with assigned lengthStrip value has lengthStrip.uom value not equal-to ('FT','M')

### Additional SBVR keywords

The following table introduces additional keywords that are used in this Profile.

|  |  |  |
| --- | --- | --- |
| **Keyword** | **Explanation** | **Way of use in SBVR AIXM Profile** |
| a, an | universal or existential quantification | depending on context based on English rules |
| assigned value | existential quantification (that the referent thing is not null) | for readability reasons (and to comply with the English language syntax), it is usually written as follows:  assigned **noun-concept** value |
| with | *has* fact type plus a condition on the property or descendant | used as a simplified form of:  p *has* q and q …some condition for q... |
| that | existential quantification for a part of a condition starting with a verb concept | typically used in front of the verb concept, between two noun concepts with associated conditions  ... p… some condition for p… that *verb* q … some condition for q … |
| equal-to | existential quantification plus a condition that the referent thing is the same thing as the referent of the term that comes next | typically used in front of a Name or a list of Names, when the item also has to exist |
| higher-than | existential quantification plus a condition that the referent thing is a numerical value that is larger than the referent of the term that comes next | typically used after a noun-concept in the form of a class attribute that takes numerical values |
| higher-or-equal-to | existential quantification plus a condition that the referent thing is a numerical value that is larger than or equal to the referent of the term that comes next | typically used after a noun-concept in the form of a class attribute that takes numerical values |
| lower-than | existential quantification plus a condition that the referent thing is a numerical value that is smaller than the referent of the term that comes next | typically used after a noun-concept in the form of a class attribute that takes numerical values |
| lower-or-equal-to | existential quantification plus a condition that the referent thing is a numerical value that is smaller than or equal to the referent of the term that comes next | typically used after a noun-concept in the form of a class attribute that takes numerical values |
| resolved-into | existential quantification plus a condition that the value of the noun-concept that precedes the keyword corresponds to an instance of the verb and noun-concept that come next | used to indicate the action of evaluating the value of an association role name instance and identifying the target feature instance, taking into consideration the associations implementation method used (according to the AIXM Feature Identification and reference document) |
| other-than | at-most-one quantification plus condition that the referent thing is not the same thing as the referent of the term that comes next | typically used in front of a Name or a list of Names, when the item may also be null |

## Methodology

This section is primarily intended for those involved in the writing of the SBVR text of the AIXM Business Rules. However, it might also provide interesting hints for the readers of the document, as it shows how a rule is being developed starting from a fact established by the model, a reference document (such as an ICAO Annex containing Standards and Recommended Practices) and applying the SBVR methodology.

The starting point for writing the SBVR text of a rule is the AIXM UML model, which describes “fact types” such as:

* “**Runway** *isSituatedAt* **AirportHeliport**”
* “**name** *is-property-of* **AirportHeliport**”
* etc.
* Some of these fact types are documented as associations between classes and there is an explicit verb (the name of the association, such as “*isSituatedAt*”). Some others are attributes of a class (and the verb that describes the fact type is implicitly “*is-property-of*”). Note that the graphical syntax for **NounConcepts** and *FactTypes* is used in these examples, as described earlier in the document.

In the first phase, the objective is to capture the various constraints that are used to define bounds, borders or limits on this set of facts. Any such rule shall have at its core a “fact type”. In addition, pure operative rules that are not directly related to a fact type established by the model might also have to be captured. For example, the required accuracy of the boundary points of a restricted area depends on whether the area is inside or outside a TMA. There is no fact type that relates a Restricted Area with a TMA, they are both instances of the Airspace class. However, the business concept of “ overlapping geometry” can be included in the business vocabulary of the aeronautical information domain and enables expressing unambiguously such rules.

Thus, a recipe for writing the SBVR text of a rule could be:

* write the rule in operational terms, trying to identify the main class in the AIXM UML model that is concerned (AirportHeliport, Airspace, Navaid, etc.).
* identify the fact types in the model to which the constraint refers (could be represented as an attribute of a class, an association of a class, or could be a more complex part of the model, involving 2-3 classes and their attributes, indirectly related);
* start building the rule text around that fact type:
  + add quantification to each noun concept involved;
  + add logical operations, if more conditions are involved;
  + add modality for the fact type.

For example:

* “It is obligatory that each **Runway** *isSituatedAt* exactly one **AirportHeliport**
* “It is obligatory that exactly one **name** *is-property-of* each **AirportHeliport**”

Keep in mind that obligations come in two forms in the aeronautical information domain: strict (expressed with “shall”) and loose (expressed with “should”).

The fact type “*is-property-of”* is maybe easier to understand if written in the opposite direction using *“has/have”.*

* “Each **AirportHeliport** shall *have* exactly one **name**”

In fact, the two rules mentioned above are already hard-coded in the AIXM UML model.The real goal is to document rules that are not already coded in the UML model. For example, a data consistency rule stating that “a heliport cannot have any associated runway”, is defined following the same steps:

* identify in the AIXM UML model the fact types concerned:
  + **Runway** *isSituatedAt* **AirportHeliport**
  + **Runway** *has* **type**
  + **AirportHeliport** *has* **type**
* the model already includes the constraint that the Airport type attribute can get a value from a predefined list: (‘AD’, ‘AH’,’HP’,’LS’,’OTHER’). The value that interests us in this case is ‘HP’ (heliport). Combining this constraint with the second fact type gives the following: **AirportHeliport** *has* **type** equal-to ‘HP’.
* adding modality and quantification gives the following rule: It is prohibited that a **Runway** *has* **type** equal-to ‘FATO’ and **Runway** *isSituatedAt* **AirportHeliport** and **AirportHeliport** *has* **type** equal-to ‘HP’
* in his form, the rule might a bit difficult to understand by domain experts. Therefore, it is recommended that a simpler form (using the keyword “with”) is used: “**Runway** with **type** equal-to ‘FATO’“ instead of “**Runway** *has* **type** equal-to ‘FATO’ and **Runway**...”
* Finally, the rule can be written as: “It is prohibited that a **Runway** with **type** equal-to ‘FATO’ *isSituatedAt* **AirportHeliport** with **type** equal-to ‘HP’”

It could be argued that the rule can be written more clear in the opposite direction: “Each AirportHeliport with type equal-to ‘HP’ shall not have associated Runway with type equal-to ‘FATO’". However, the fact-type established by the model is that 'Runway is situated at Airport'. All associations in the model have directionality. Therefore, it is better to take this into consideration when writing the rule. In XML, the Runway will point to the AirportHeliport, therefore the rule can be checked directly from the Runway towards the AirportHeliport.

# Schematron test implementation

## Introduction

In addition to the SBVR definition, where possible Schematron [[4]](#wxo2ufcod0ee) code is provided for the AIXM Business Rules provided with this document. This is done for two reasons:

* in order to verify that the SBVR description of the rule is sufficiently clear and unambiguous in order permit its actual implementation as software code;
* to offer practical means by which an AIXM data set could be verified against the business rules, using software readily-available on the Web, most of it for free.

It shall be kept in mind that this Schematron encoding is missing for some rules for which the encoding is expected to be very complex or even impossible. In addition, any use of the Schematron coding provided with the AIXM Business Rules is subject to the license and disclaimer copied in [Annex A](#_cb4vzwi4gzj5).

## Schematron code - technical aspects

Comprehensive introductions to Schematron are available on the Web, such as [[5] Introduction to Schematron](#7dno9zq4fx51) by Mulberry Technologies, Inc.

### XPath version

The Schematron code is in principle limited to XPath 2.0 “basic” conformance level, which is a W3C recommendation since January 2007 and is much richer than XPath 1.0.

Practically, this is ensured by checking that the Schematron code is usable with the XPath subset supported by the latest Saxon-HE version (using the ISO Schematron implementation for XSLT-2.0).

### XPath compatibility

To ensure maximum compatibility across different Schematron implementations, the XPath expressions should be written in the following style:

1. The Schematron context is always an absolute XPath - not a relative XPath. For example:  
     
   Right: //aixm:InitialLegTimeSlice/aixm:holding  
   Wrong: aixm:InitialLegTimeSlice/aixm:holding
2. The Schematron assertion always evaluates to a boolean value - not a number, string, node or node set. That is, the assertion should result in a clear *true* or *false* value, not something like an empty set of nodes which would then have to be interpreted as *false*. For example:  
     
   Right: @srsName='urn:ogc:def:crs:EPSG::4326'  
   Wrong: .[@srsName='urn:ogc:def:crs:EPSG::4326']

### Use of Java extensions

Java extensions are used in order to provide functions that would be hard, if not impossible, to write using just XPpath and related conditions. For example, geodetic distance and angle calculations functions are written in Java and called from the Schematron code.

The library that implements these extensions is available in the form of a .jar file: [ARCEngine.jar](https://drive.google.com/file/d/0BxlGN-YBj-q0amhYQnpQMjZHYnM/edit?usp=sharing).

### Feature references in the Schematron code

According to the AIXM [Feature Identification and Reference](http://www.aixm.aero/sites/aixm.aero/files/imce/AIXM51/aixm_feature_identification_and_reference-1.0.pdf) document, associations between features can be implemented with abstract or local references. In the current version, the Schematron code provided assumes that all associations are encoded as local references (xlink:href=”#...” referring to the gml:id value of the target feature).

### Content of the Excel file

In the Excel file (see next chapter), three fields are used to store a rule translation to Schematron code

* One field gives the Schematron context
* Another one gives the Schematron text

However, the so-called Schematron text is in fact only the xpath notation used within Schematron rules. The final Schematron code will need to include the given context and the given text within specific xml tags before a Schematron engine can make use of it.

A macro included in a separate “[EC-AIXM-Rules-Schematron-Excel\_to\_Schematron.xlsm](https://drive.google.com/open?id=0BxlGN-YBj-q0SzFrVmk1bmtQZ0U)” file can export the Schematron code in a ‘ready to use’ format.

Usage :

* Open the .xlsm file (*NB*: *do not rename the .xslm file!*)
* Hit the “Run Schematron generator” button
* On first popup : Select a valid AIXM-Rules-Mgr.Export.xml file from which you want to extract the Schematron. (*NB: valid means that you have no error when opening this file with excel!*)
  + *attention: the Excel Macro is hard-coded to extract the Schematron code from column Q. Due to the new columns (B,...) that include the flag for profiles, the Macro will not work properly. The current workaround is to open the .xml file in Excel, remove the columns B->... that contain the profile flags. Make sure that the Schematron code is in column Q and save the file in the original .xml format.*
* On second popup : Save the output into an “.sch” file
* On alert, click on “Delete” button
* Close the .xslm file without saving

The Schematron file you generated is ready-to-use in your validation engine.

The character x000D indicates carriage return in the UTF8 XML export file

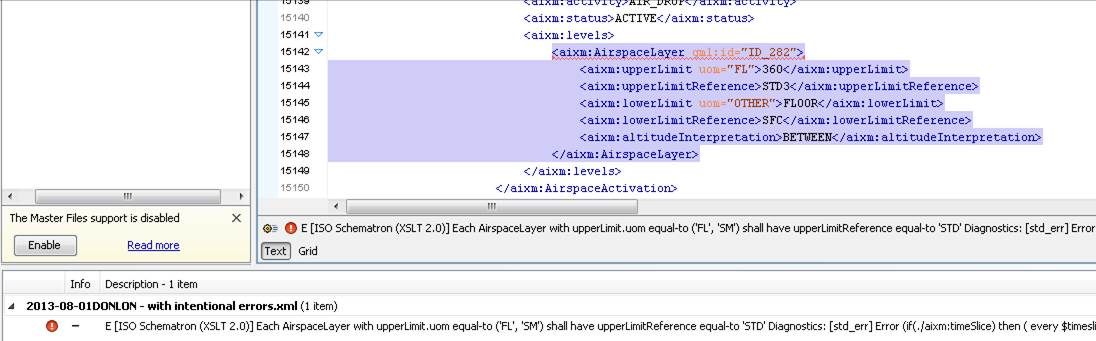
## Practical verification of AIXM data sets

### Testing

<oXygen/> XML Developer in Professional edition was used for testing the Schematron code

* <oXygen/> XML Developer is very interesting because, not only it allows testing the rules, but it also allows editing xpath code with auto-completion functions based on a provided xml or xsd
* Java extensions can be integrated in <oXygen/> XML Developer as well.

The following picture shows an example of a rule checking result in <oXygen/> XML Developer



# AIXM Business Rules data set

## Excel file

The reference set of AIXM 5.1 Business Rules is made available in the form of an Excel file, that contains both the SBVR formulation and, where available, the Schematron encoding of the rule. This Excel file is available here: [www.aixm.aero/page/business-rules](http://www.aixm.aero/page/business-rules).

## Profiles

The Excel file contains a pool of rules from which subsets (profiles) are then extracted for various applications. As explained in the Introduction, not all systems apply the same rules, in particular when it comes to minimal data - what is critical for a system might be of no relevance for another system. The error level of a rule may also depend on the profile.

The profile to which a rule belongs is indicated starting from column B. There is a column for each Profile about which information is stored in the tool used to produce this Excel file. Two error levels are used in the profiles (“Error” or “Warning”), which is aligned with the two levels of the diagnostics messages in Schematron.

### EAD Data Provider

This profile contains a selection of AIXM features and business rules that are imposed when providing data to the European AIS Database (EAD). These rules are classified into two error levels:

* Warning - indicates that the rule is part of the EAD Profile and failure to comply will generate a Warning message. However, the data will be accepted by the EAD.
* Error - indicates that the rule is part of the EAD Profile and failure to comply will generate an upload Error message. It indicates that the whole data set will be rejected. This is the case for rules that prevent data processing failures in EAD. For example, some AIXM 5.1 data needs to be mapped back to AIXM 4.5, which requires some mandatory attributes to be present in the upload file. Therefore, the rules that ensure the presence of these mandatory values are part of the EAD\_Error profile.

### Event profile

This profile will contain a selection of AIXM features and business rules that can be used in order to verify the correctness and completeness of Digital NOTAM data sets. This shall be included in the Digital NOTAM Event Specification (starting with [version 2.0](https://docs.google.com/document/d/1UDwGkTFnzuYaGEfKmkRrnDMKEDGfPAixNTaLpIO1qP0/edit?usp=sharing)).

### Event\_FAA profile

This profile will contain rules that are specific to the Digital NOTAM encoding scenarios of FAA (FNS-NDS) system. Such rules refer to an FAA scenario number.

### Minimum air navigation data set profile

--- example of a possible future profile ---

This profile shall contains a selection of AIXM features and business rules that are proposed to be applied when providing data sets that are used for international IFR air navigation applications (for example, en-route charts, flight planning, NOTAM validation, etc.), but excluding SID/STAR/Approach procedure data.

### Full air navigation data set profile

--- example of a possible future profile ---

This profile contains a selection of AIXM features and business rules that are proposed to be applied when providing data sets that are used for international IFR air navigation applications, including SID/STAR/Approach procedure data. The objective of this profile is to provide data which can be directly converted into ARINC 424 and be used by downstream actors for Flight Management Systems (FMS) data provision

# References

[1] “Semantics of Business Vocabulary and Business Rules”, (SBVR), v1.1, Object Modeling Group (OMG), September 2013. URL:<http://www.omg.org/spec/SBVR/1.1/>

[2] “AIXM Business Rules, SBVR Specifications - Abstract”, version 3, 25 February 2010, Pulsar Consulting under contract for Eurocontrol

[3] “Guidance on Writing AIRM Constraints”, SESAR Project 08.01.03, available from SESAR Joint Undertaking (SJU), [www.sesarju.eu](http://www.sesarju.eu/programme/workpackages/wp8)

[4] Schematron, ISO/IEC 19757-3:2006 Information technology -- Document Schema Definition Language (DSDL) -- Part 3: Rule-based validation -- Schematron, [www.schematron.com](http://www.schematron.com)

[5] Introduction to Schematron, Wendell Piez and Debbie Lapeyre, Mulberry Technologies, Inc., <http://www.mulberrytech.com/papers/schematron-Philly.pdf>

# Annex A - License and Disclaimer

For the use of the Schematron code provided with the AIXM Business Rules

====================================================================

Copyright (c) 2016, EUROCONTROL & FAA

=====================================

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 names of EUROCONTROL or FAA nor the names of their contributors may be used to endorse or promote products derived from this specification without specific prior written permission.

THIS SPECIFICATION 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.

==========================================

Editorial note: this license is an instance of the BSD license template as

provided by the Open Source Initiative:

http://www.opensource.org/licenses/bsd-license.php