AIXM 5

Temporality Model

Aeronautical Information Exchange Model (AIXM)

For all inquiries, please contact:

Brian MURPHY - brian.murphy@faa.gov

Eduard POROSNICU - eduard.porosnicu@eurocontrol.int

Part Edition No.		Part Edition Issue Date	Part Author	Reason for Change
0.1	Draft	24 APR 2007	Design Team	Initial Draft
0.2	Draft	04 JUN 2007	Design Team	Updated after discussions in St. Louis and Frankfurt.
0.3	Draft	10 JUN 2007	Design Team	Updated after comments from AIXM FG #8 Meeting and from Edna.
0.4	Proposed	15 JUL 2007	Design Team	Removed "Static" Time Slices from the model. Re-organised the presentation of the different kinds of Time Slices.
0.5	Proposed	12 NOV 2007	Design Team	Clean-up for first public version.
0.6	Proposed	01 FEB 2010	Design Team	Describe PropertiesWithSchedule concept introduced in AIXM 5.1 (see chapter 2.8) Include UML diagrams from AIXM 5.1 model
1.0	Released	15 SEP 2010	Design Team	Released version, to be used as baseline for AIXM 5.1 implementations.
1.1	Released	06 FEB 2019	AIXM CCB	Content has been re-organised in order to facilitate the reading for aeronautical information domain experts. Minor revisions and clarifications, in relation with AIXM 5.1.1. New section dedicated to temporality consistency rules. Examples moved out of this document and integrated in a reference implementation / test suite.

Table of Contents

Table of Contents

1. The need for a temporality model

Time is an essential aspect in the aeronautical information domain, where change notifications are usually provided well in advance of their effective dates. Aeronautical information systems are requested to store and to provide both the current situation and the future changes. The expired information needs to be archived for legal investigation purposes.

For operational reasons, a distinction is usually made between:

- ***permanent changes*** (the effect of which will last until the next permanent change or until the end of the lifetime of the feature) and
- ***temporary status*** (changes of a limited duration that are considered to be overlaid on the permanent state of the feature). For example, systems that produce printed aeronautical documentation (AIP, charts) tend to ignore temporary status information; only the static data is represented on such printed products.

A temporary change includes the concepts of overlay and reversion. When the temporary change ends, the overlay no longer applies and the feature baseline data remains the only applicable information.

Note that, from an operational point of view, "temporary status" also includes the concept of "temporary features". However, from the AIXM point of view, temporary features are in no way different from normal features. The feature is created and withdrawn, just that the life span is shorter than usual.

In order to satisfy the temporal requirements of aeronautical information systems, AIXM must include an exhaustive temporality model, which enables a precise representation of the states and events of aeronautical features. In particular, this shall enable the development and the implementation of digital NOTAM. In this concept, the free text contained in a NOTAM message is replaced with structured facts, which enable the automated processing of the information.

A general temporal model needs to be uniformly applied to all aeronautical feature types and be abstracted from the task of modeling object properties. At the conceptual level, the model needs to describe the temporal evolution of the features, as they occur in the real world. This shall be done in compliance with the following rules:

- Completeness - all temporal states must be representable;
- Minimalism - use of minimal number of elements;
- Consistency - no reuse of elements with different meaning;
- Context-free - meaning of (atomic) elements independent of context; no functional dependency of (atomic) elements at the data encoding level;

The data exchange specification shall support the conceptual temporality model. In addition, convenience elements ("views") may be introduced in the data exchange specification in order to facilitate the operations. This means that the data exchange specification may deviate from the "minimalism" rule.

2. How to read this document

Aeronautical information domain experts are suggested to start by reading the Temporality Use Cases section on the AIXM website, which provides an exhaustive set of examples of temporality use cases. It includes both regular situations (such as new feature, changed properties, withdrawn feature, etc.) and exceptional situations (such as postponed change, cancelled withdraw, etc.). Those examples shows how “time slices” are used for encoding the states and the changes that occur in the lifetime of an aeronautical information feature.

The “time slices” concept is at the heart of the AIXM Temporality Model. If interested in more technical aspects of the Temporality Concept, domain experts should also read the chapters 3 and 4, which explain how the reasons for the “time slice” approach and provide detailed usage rules.

System developers are suggested to read the document in the full sequence order of the chapters, while also looking at the coding examples provided in the Temporality Use Cases section on the AIXM website:

- Chapter 3 explains the time slice concept starting from the operational needs of the aeronautical information domain;
- Chapter 4 provides a summary of the AIXM temporality model, together with usage rules and data verification rules;
- Chapter 5 contains some frequently asked questions and their answers.

3. Building the Temporality Model

In order to explain the AIXM Temporality Model, this chapter follows a step-by-step approach, in which the elements that compose this model are added progressively in order to satisfy the operational needs of the aeronautical information domain.

3.1 (step 1) Time varying properties

There are two levels at which aeronautical feature instances are affected by time:

- Every feature has a start of life and an end of life;
- The properties of a feature can change within the lifetime of the feature; this includes the possibility for a property to have a null value for certain periods.

The start of life and the end of life may also be considered as feature properties (attributes). This gives the following high-level list of properties for any AIXM feature:

- a global unique identifier;
- the start of life (date and time);
- the end of life (date and time);
- attributes and associations that qualify, quantify or relate in some form that feature.

It is considered that any feature property may change in time, except for the global unique identifier. This is a key principle of the AIXM Temporality model.

The first step in the construction of the AIXM temporality model is represented by the diagram below, which shows the values of a features' properties (P1, P2, … P5) along a timeline.

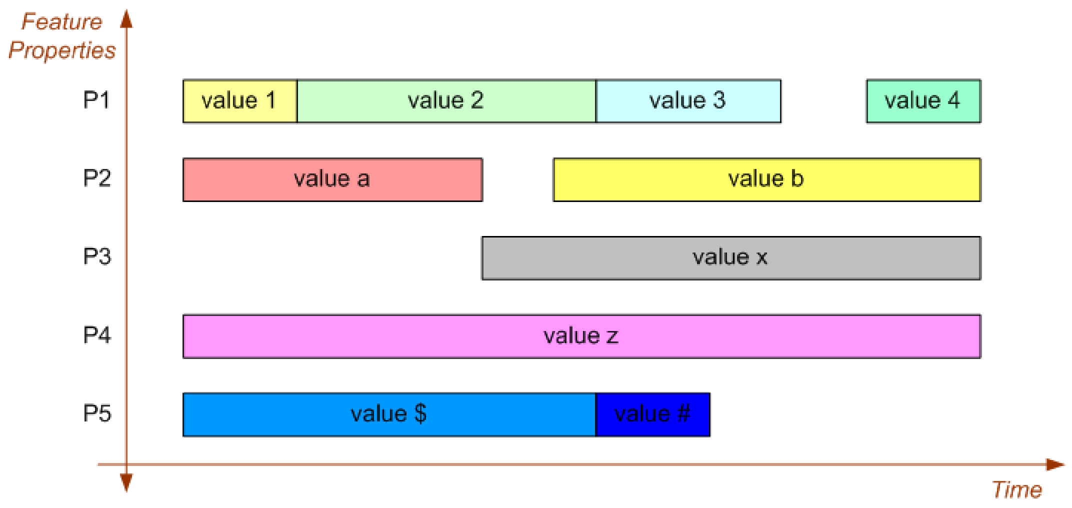


Figure 1 – Time varying properties

3.2 (step 2) The Time Slice model

The temporality model adopted by AIXM describes feature events and states. A state is the feature property set valid over a time period. According to the GML standard [OGC 07-036] section 14.5.3 "*States are captured by time-stamped instances of a feature*".

An event is a change of one or more feature properties, which is defined in the GML standard as "*an action that occurs at an instant or over an interval of time*". In the diagram below events are located at the vertical cuts while states are represented as the feature property set between events.

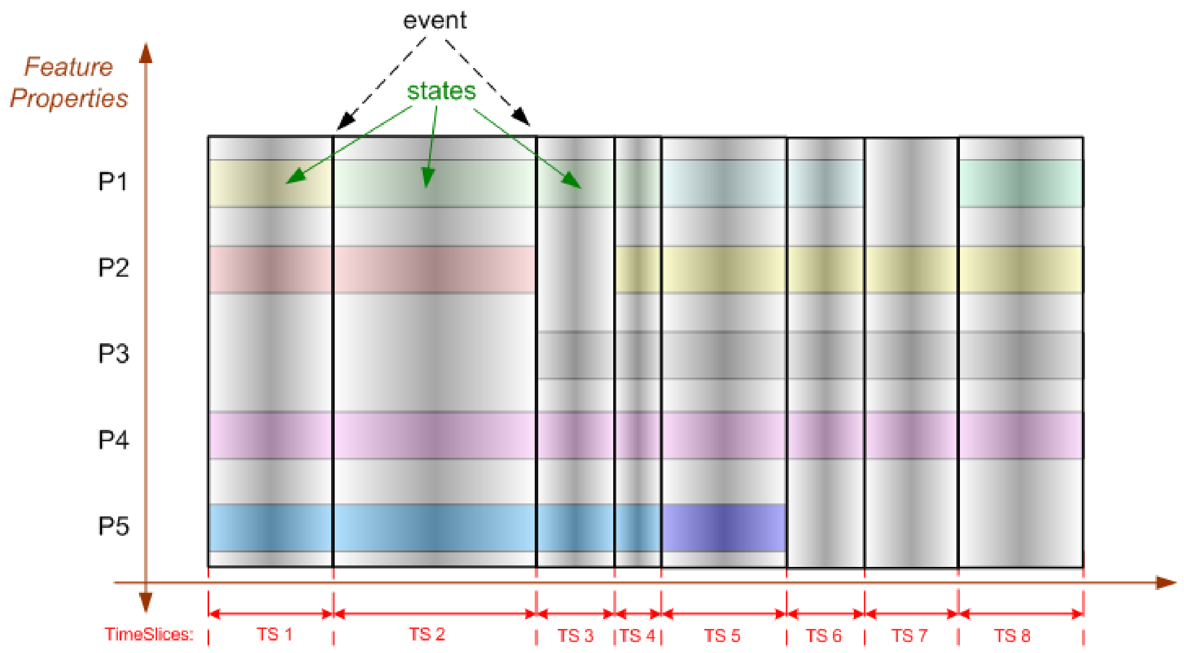


Figure 2 – Events and States

In order to describe the feature properties during states and events, the time varying properties of every AIXM feature are encapsulated in a container called "AIXM TimeSlice".

The history of the feature is described with "state" Time Slices, each containing the values of the time varying properties between two consecutive changes (events). Each Time Slice has specified validity period and one constant value for each property. In an UML diagram, the basic Time Slice concept is represented as below:

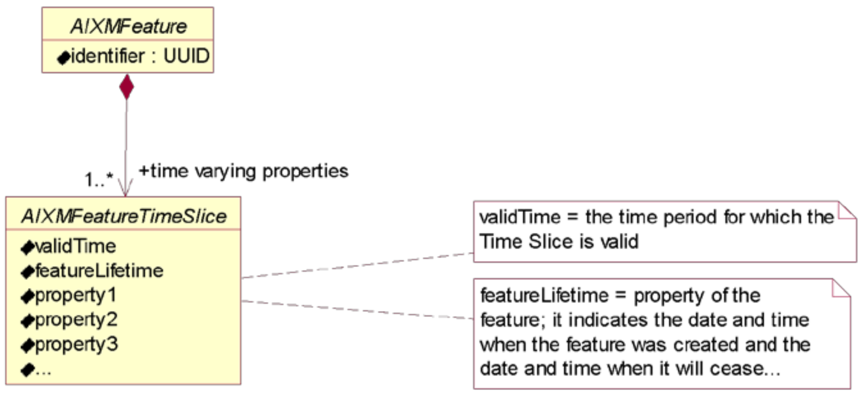

AIXMFeature
identifier : UUID
1..*
+time varying properties
AIXMFeatureTimeSlice
validTime
featureLifetime
property1
property2
property3
...
validTime = the time period for which the Time Slice is valid
featureLifetime = property of the feature; it indicates the date and time when the feature was created and the date and time when it will cease...

Figure 3 – AIXMFeatureTimeSlice

There exist several possible TimeSlices types, as detailed in the following sections.

3.3 (step 3) Temporary changes

Aeronautical features may be affected by temporary events, such as a navaid being out of service, a runway being closed, a restricted area becoming active, etc. All such events generate temporary changes in the values of one or more feature properties. This includes the possibility to temporarily give a value to a property that does not have baseline value or to temporarily remove a baseline value. At the end of the temporary event, the values of these properties remain the ones provided by the baseline TimeSlice.

In order to model temporary events, the basic temporality model previously defined in this document needs to be refined by differentiating between two kinds of Time Slices:

- ***Baseline*** = a kind of Time Slice that describes the feature state (the set of all features' properties) as result of a permanent change.
- ***Temporary*** = a kind of Time Slice that describes the transitory overlay of a feature state during a temporary event.

From the "payload" point of view, there exists an essential difference between Baseline and Temporary Time Slices:

- A Baseline Time Slice includes the values of all time varying feature properties that are defined for the time of validity of the Time Slice; for example, in the diagram below, TS2 will include the values of p1, p2, p4 and p5;
- A Temporary Time Slice includes just the values of the properties that are temporarily changed; for example, in the diagram below, TS "temp" will include just p4="value w". For this reason, temporary Time Slices are called "Temporary Delta" Time Slices.

Note: a temporary change could also consist in a feature property becoming temporarily undefined (no value). For this purpose, feature properties can also get a 'nil' value.

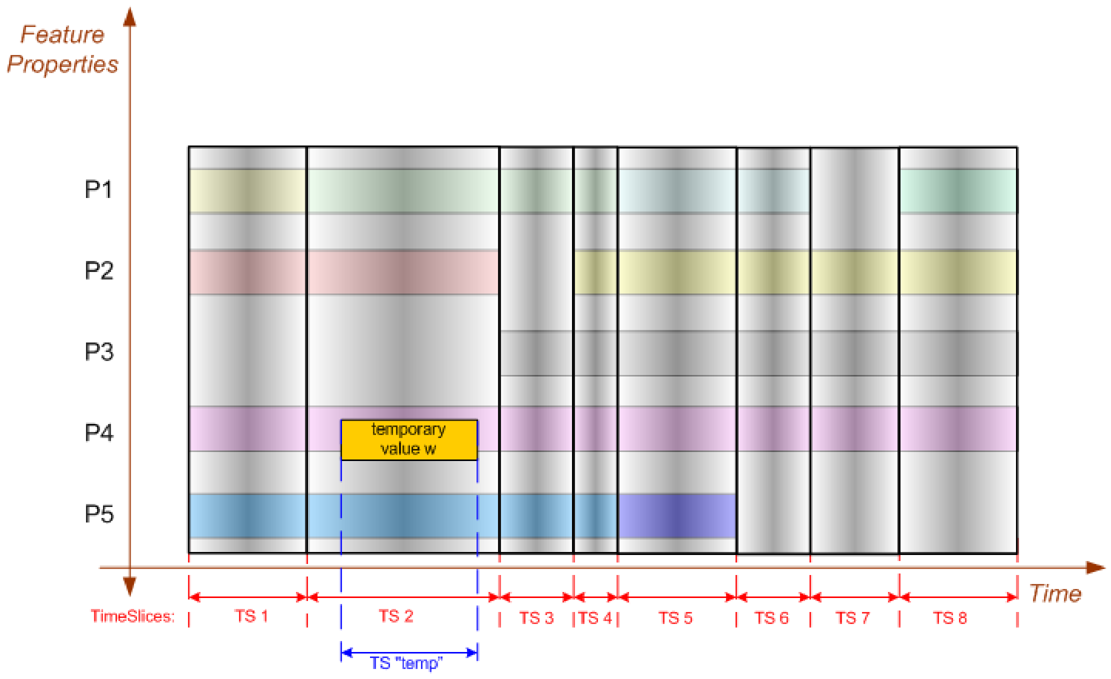


Figure 4 - Baseline and Temporary TimeSlices

One reason for temporary Time Slices to contain strictly the modified properties is to avoid creating dependencies between temporary events running in parallel. **It is a key principle of the model that only the affected properties are included in Temporary Delta Time Slices.**

With regard to the UML model, as the Temporary Delta Time Slices need to be distinguished from the baseline ones, an additional attribute is necessary in the AIXMFeatureTimeSlice class. This attribute is named "interpretation" and indicates the type of Time Slice - BASELINE (Baseline) or TEMPDELTA (Temporary), as shown in the figure below.

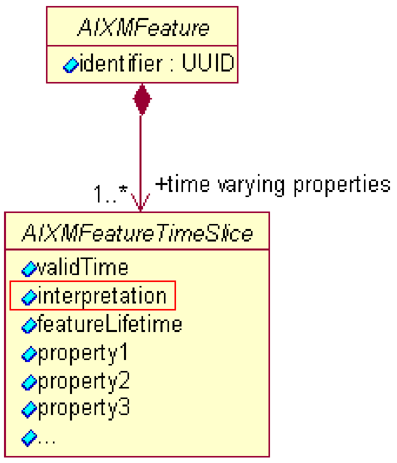


Figure 5 – FeatureTimeSlice with "interpretation" property

The essential benefit brought by TEMPDELTA Time Slices is that they enable the encoding of "digital NOTAM". A TEMPDELTA Time Slice will contain the values of all feature properties that are overlaying for a limited time period the baseline values.

The temporality model described up to this point complies with the rules for completeness, minimalism, consistency and context-free mentioned in Chapter 1. Using BASELINE and TEMPDELTA Time Slices, it is possible to describe the temporal evolution of the time varying properties of aeronautical features, covering both permanent states and temporary events.

3.4 (step 4) Current Status - SNAPSHOT Time Slices

The model with only BASELINE and TEMPDELTA TimeSlice types lacks the possibility to communicate the current status of a feature, which results when merging the baseline data with any temporary data that is effective at that moment in time. In order to reply to a query that requests the information on a feature at a point in time, an additional kind of Time Slice is included in the model. This is named "SNAPSHOT" and will carry the result of merging the effective BASELINE information with all overlaying TEMPDELTA that are effective at that moment in time. Typically, a SNAPSHOT Time Slice will have a Time Instant as validTime.

- SNAPSHOT = A kind of Time Slice that describes the state of a feature at a time instant, as result of combining the valid BASELINE Time Slice at that time instant with all eventual valid TEMPDELTA Time Slices at the same time instant.

Note that for a SNAPSHOT, the sequenceNumber and the correctionNumber (see section 3.6 further in this document) properties shall be left empty.

3.5 (step 5) Data exchange – need for PERMDELTA Time Slices

Although the complete lifetime of a feature can be encoded using only BASELINE and TEMPDELTA Time Slices, a Time Slice that represents permanent change events is also introduced in the model, for convenience. This will be called Permanent Delta (PERMDELTA).

- PERMDELTA = A kind of Time Slice that describes the difference in a feature state as result of a permanent change.

The ***end of life can now be communicated with a PERMDELTA Time Slice in which the featureLifetime/endPosition gets a value***. Symmetrically, the start of life can also be communicated with a PERMDELTA Time Slice, in which the featureLifetime/startPosition property and the other feature properties get their initial values. Being modeled as formal events, the start of life and the end of life can be postponed or advanced (this requires a mechanism for updating an ‘event’ Time Slice, which will be discussed later in this paper).

A second advantage of PERMDELTA Time Slices is that it gives the possibility to explicitly indicate the properties that are changed at the time of validity of the TimeSlice. The data provider is the best positioned to know the list of changed properties and the PERMDELTA Time Slice gives the possibility to communicate this information to interested clients. This facilitates the implementation of systems that are not interested in changes of certain feature properties. For example, charting applications - a PERMDELTA affecting properties that do not appear on the chart could be ignored.

From a conceptual point of view, a PERMDELTA Time Slice occurs at the edge between any two consecutive BASELINE Time Slices and it contains values strictly for the changed properties.

Figure 6 shows the four types of TimeSlice that are part of the AIXM temporality Concept

.

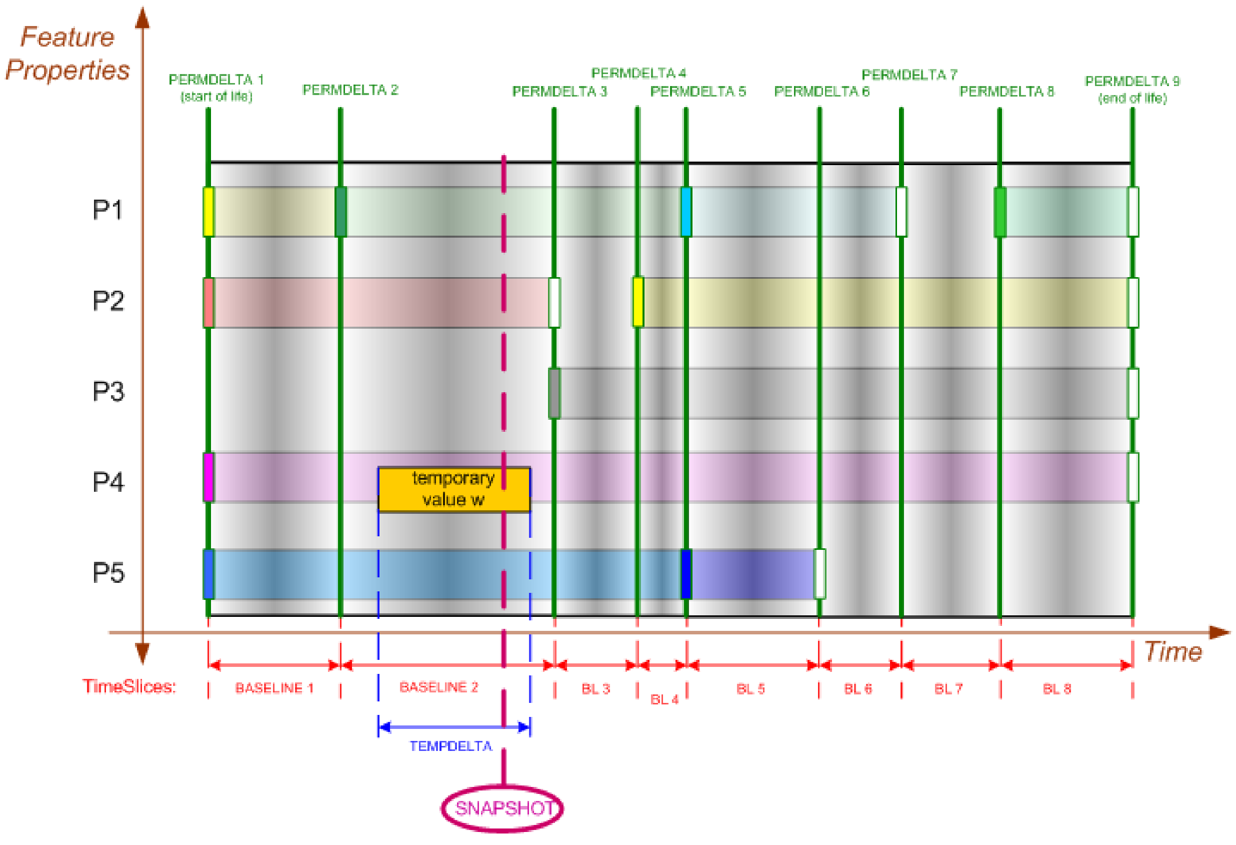


Figure 6 – The four types of TimeSlices

3.6 (step 6) Data exchange – corrections

In the aeronautical world, we need to communicate information about events that are planned to take place in future. Inevitably, the reality might be different from the initial planning and it might be necessary to update the already communicated information.

As in AIXM the properties of a feature are encapsulated in Time Slices, this means that we need a mechanism for updating/correcting a previously communicated feature Time Slice. First, a key is necessary for the identification of the Time Slice concerned. For this purpose, a ***“sequenceNumber”***

attribute is introduced in the model, playing the role of unique identifier for each Time Slice inside a feature. Then, a "***correctionNumber***" is used in order to sort the eventual corrections in their chronological order (see Figure 7 in the next section).

If necessary to correct a previously communicated Time Slice, an update of the Time Slice will be provided, having the same sequence number but a higher correction number. As a consequence, if there exist more than one Time Slice with the same sequence number related to a given feature, the one with the highest correction number will be considered as "valid" (meaning usable by a consumer system), the other ones as "invalid" (thus not to be used).

The sequenceNumber does not play any ordering role, it is just an identifier, allowing to indicate which previous TimeSlice (with the same interpretation and that belongs to the same feature) is superseded by a "correction" Time Slice.

3.7 The AIXM *TimeSlice* Model in summary

The UML representation of the final AIXM 5 Feature Time Slice model is provided below:

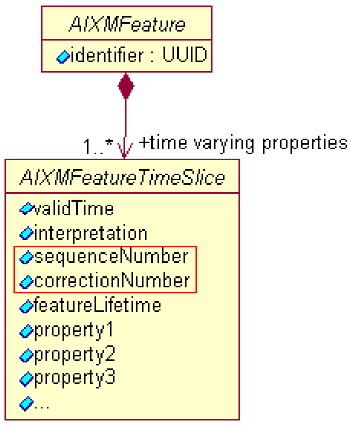


Figure 7 – Complete AIXMFeatureTimeSlice model

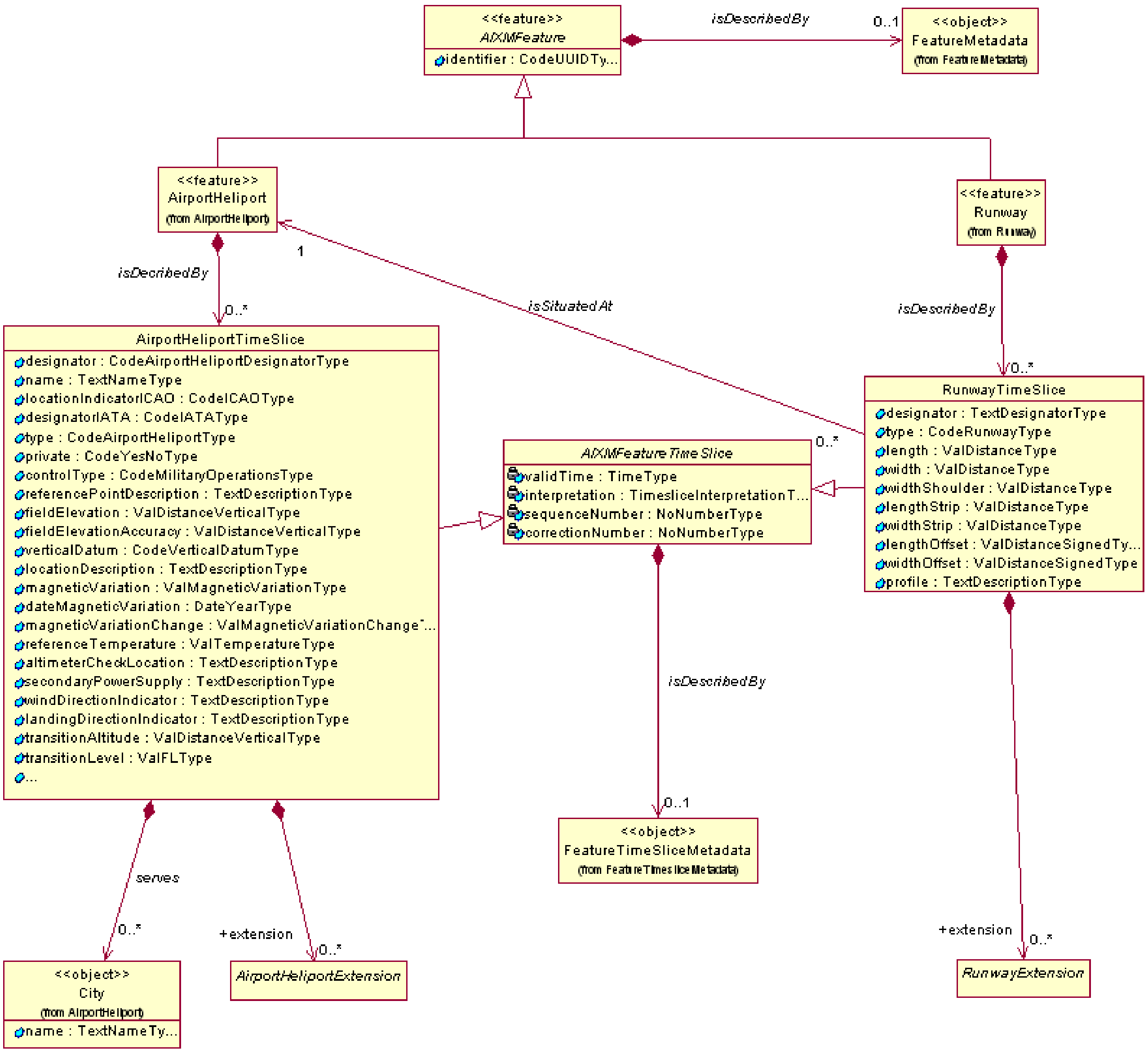
The following Time Slice types are used in the AIXM:

- ***BASELINE = a kind of Time Slice that describes the feature state (the set of all feature's properties) as result of a permanent change.***
- ***PERMDELTA = A kind of Time Slice that describes the difference in a feature state as result of a permanent change.***
- ***TEMPDELTA = a kind of Time Slice that describes the transitory overlay of a feature state during a temporary event.***
- ***SNAPSHOT = A kind of Time Slice that describes the state of a feature at a time instant, being the result of the combination of the valid BASELINE Time Slice for the same time instant with all eventual valid TEMPDELTA Time Slices for the same time instant.***

It shall be noted that the AIXM TimeSlice model is based on the "append only" principle. AIXM TimeSlices cannot be deleted or modified. Instead, they are superseded by other TimeSlices, which have the same sequenceNumber and higher correction numbers. It is the responsibility of the AIXM data sources to ensure the temporal coherence of the TimeSlice information. It shall also be kept in mind that the sequenceNumber does not play any ordering role, it is only a unique identifier.

3.8 Temporality applied to the Abstract Model

The AIXM UML model contains a set of abstract classes that are used as templates for the features and objects defined in AIXM. When applying the Time Slice concept, as described in this document, this would trigger the split of every UML class that represents a feature into a main class and a "FeatureTimeSlice" class, as shown in the following diagram.



<<feature>>
AIXMFeature
identifier : CodeUUIDTy...
isDescribedBy
0..1
<<object>>
FeatureMetadata
(from FeatureMetadata)
<<feature>>
AirportHeliport
(from AirportHeliport)
<<feature>>
Runway
(from Runway)
1
isDecribedBy
0..*
isSituatedAt
isDescribedBy
0..*
AirportHeliportTimeSlice
designator : CodeAirportHeliportDesignatorType
name : TextNameType
locationIndicatorICAO : CodeICAOType
designatorIATA : CodeIATAType
type : CodeAirportHeliportType
private : CodeYesNoType
controlType : CodeMilitaryOperationsType
referencePointDescription : TextDescriptionType
fieldElevation : ValDistanceVerticalType
fieldElevationAccuracy : ValDistanceVerticalType
verticalDatum : CodeVerticalDatumType
locationDescription : TextDescriptionType
magneticVariation : ValMagneticVariationType
dateMagneticVariation : DateYearType
magneticVariationChange : ValMagneticVariationChange...
referenceTemperature : ValTemperatureType
altimeterCheckLocation : TextDescriptionType
secondaryPowerSupply : TextDescriptionType
windDirectionIndicator : TextDescriptionType
landingDirectionIndicator : TextDescriptionType
transitionAltitude : ValDistanceVerticalType
transitionLevel : ValFLType
...
0..*
AIXMFeatureTimeSlice
validTime : TimeType
interpretation : TimesliceInterpretationT...
sequenceNumber : NoNumberType
correctionNumber : NoNumberType
RunwayTimeSlice
designator : TextDesignatorType
type : CodeRunwayType
length : ValDistanceType
width : ValDistanceType
widthShoulder : ValDistanceType
lengthStrip : ValDistanceType
widthStrip : ValDistanceType
lengthOffset : ValDistanceSignedTy...
widthOffset : ValDistanceSignedType
profile : TextDescriptionType
isDescribedBy
0..1
<<object>>
FeatureTimeSliceMetadata
(from FeatureTimesliceMetadata)
serves
0..*
+extension
0..*
+extension
0..*
<<object>>
City
(from AirportHeliport)
name : TextNameTy...
AirportHeliportExtension
RunwayExtension

Figure 15 - Model expanded with explicit TimeSlice classes

The UML diagram shows how each and every `<<feature>>` inherits from the abstract AIXMFeature class. The concrete features are described by TimeSlices which have properties. The TimeSlice inherits from the abstract AIXMFeatureTimeSlice class.

The diagram also shows that each AIXM Feature may have FeatureMetadata and each TimeSlice may have FeatureTimeSliceMetadata. Finally, each TimeSlice may contain an Extension. The Extension mechanism allows each user of AIXM 5 to define and use his own specific attributes and classes, in addition to the core AIXM ones.

The diagram above is quite complex. If applied to the whole set of AIXM classes, it might undermine the readability of the UML diagrams, as a separate "TimeSlice" class and the necessary associations would have to be added for each <<feature>> class. **Therefore, the design team of the AIXM 5 model has**

decided to provide a simplified AIXM UML model, without visible inheritance of all features from the abstract AIXMFeature and without visible *SomeFeature*TimeSlice classes. However, the split and into SomeFeatureTimeSlice classes is assumed to exist and it is applied when converting from the UML model to the XML Schema of AIXM.

4. Temporality Rules

4.1 Overview

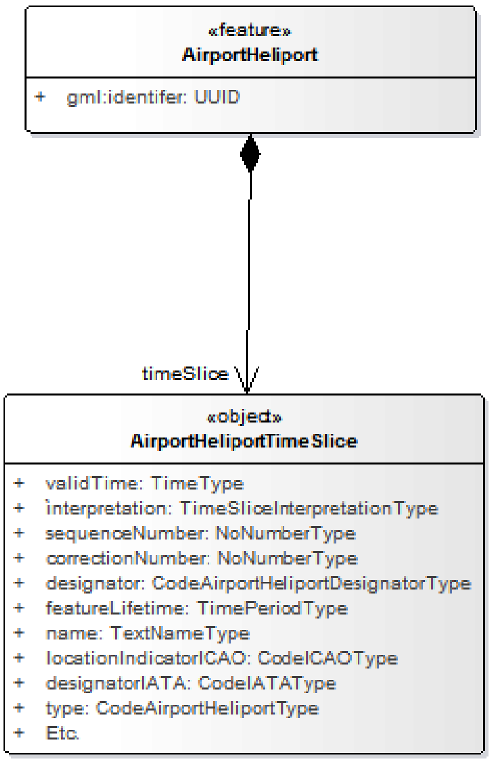
The essence of the AIXM Temporality concept is summarised in the following UML class diagram.

All properties (attributes and associations) of a feature are encapsulated in a TimeSlice that has a timestamp, which can be a time instant or a time period. Every feature has a "featureLifetime" property, which models that start and end of life of an aeronautical information feature.

The only feature property that is situated outside the TimeSlice and cannot have a start/end of validity is the gml:identifier.

AIXM TimeSlices model supports the coding of both feature states and feature events. The type of TimeSlice is specified by the "interpretation" attribute:

- (state) BASELINE TimeSlices are valid for a time period and provide the static information - reference values for each property for this time period;
- (event) TEMPDELTA TimeSlices are valid also for a time period and provide the dynamic information - temporary values for some properties
- (state) SNAPSHOT TimeSlices are valid at a time instant and provide the status at that moment in time by merging the BASELINE values with the eventual TEMPDELTA values


«feature»
AirportHeliport
+ gml:identifer: UUID
timeSlice
«object»
AirportHeliportTimeSlice
+ validTime: TimeType
+ interpretation: TimeSliceInterpretationType
+ sequenceNumber: NoNumberType
+ correctionNumber: NoNumberType
+ designator: CodeAirportHeliportDesignatorType
+ featureLifetime: TimePeriodType
+ name: TextNameType
+ locationIndicatorICAO: CodeICAOType
+ designatorIATA: CodeIATAType
+ type: CodeAirportHeliportType
+ Etc.

- (event) PERMDELTA TimeSlices are valid at a time instant and provide the difference between two consecutive BASELINE.

In order to enable the provision of corrections to previously provided data, each AIXM TimeSlice includes:

- a sequenceNumber, which plays the role of unique identifier for a TimeSlice, in the context of the AIXM feature to which it belongs;
- a correctionNumber, which indicates the order in which the corrections have been applied, in case several TimeSlices have the same interpretation and sequenceNumber - the feature TimeSlice with the highest correctionNumber value is considered the only valid one.

The TimeSlice types mentioned above and the sequence/correction number are primarily intended for coding released information. The coding of potential ("what if") information would probably require additional properties and/or lists of values, which are not included in the current AIXM Temporality Concept.

4.2 Definitions

The following terms are used in relation with the TimeSlice temporality model:

- ***correction TimeSlice*** *= designates the TimeSlice that has the same sequenceNumber and a higher correctionNumber than another TimeSlice of the same feature and having the same interpretation value*
- ***valid TimeSlice*** *= designates either the single TimeSlice or the one with the highest correctionNumber between the TimeSlices with the same interpretation and with the same sequenceNumber.*
- ***active TimeSlice*** *= designates the TimeSlice that has its beginPosition at or before the current system date/time and the endPosition after the current system date/time or an indeterminate endPosition*

4.3 Data coding rules

The AIXM temporality concept, as summarised above and as described in the previous chapters, implies a number of rules for the content of the various types of feature Time Slices. In order to enable the verification of AIXM data sets for compliance, the Temporality Concept rules are explicitly provided in this chapter. A more structured description of these rules is included in the general set of AIXM Business Rules, as provided for each AIXM version, using the Semantics of Business Vocabulary and Business Rules (SBVR) standard.

Editorial conventions

- ***shall*** = when used in this Chapter, it indicates a rule, which must be applied by all AIXM implementations in order to ensure a minimal level of interoperability.
- ***should*** = when used in this Chapter, it indicates a recommendation, which is meant to facilitate the interoperability of various AIXM implementations. Recipients of AIXM data expect to receive both data which complies and data which does not comply with the recommendation.

Rule Id	***Title***	***Description***
TS_001	BASELINE - always time period	Each BASELINE TimeSlice shall have validTime.TimePeriod or no validTime. *Justification: basic rule of the temporality concept. With regard to the situations where validTime is empty, see section "Canceling a Time Slice (abandoned changes)"*
TS_002	PERMDELTA - always time instant	Each PERMDELTA TimeSlice shall have validTime.TimeInstant or no validTime. *Justification: basic rule of the temporality concept. With regard to the situations where validTime is empty, see "Canceling a Time Slice (abandoned changes)"*
TS_003	TEMPDELTA - always time period	Each TEMPDELTA TimeSlice shall have validTime.TimePeriod or no validTime. *Justification: basic rule of the temporality concept. With regard to the situations where validTime is empty, see "Canceling a Time Slice (abandoned changes)"*
TS_004	SNAPSHOT - always time instant	Each SNAPSHOT TimeSlice shall have validTime.TimeInstant.

		Note: For technical reasons, some systems calculate the SNAPSHOT in advance for time periods. See OGC 12-027r3 WFS Temporal Extension.
TS_005	Timeslice (excluding Snapshot) - unique interpretation/sequence/correction	Two TimeSlices of the same feature and with the same interpretation value (BASELINE, PERMDELTA, TEMPDELTA) shall not have the same combination of sequenceNumber and correctionNumber. *Justification: basic rule of the temporality concept. A duplicate would mean conflicting information as there is no other indication about which Time Slice is the valid one.*
TS_006	SNAPSHOT - no sequence and correction	A SNAPSHOT TimeSlice shall not have assigned correctionNumber and sequenceNumber values. *Justification: basic rule of the temporality concept. A SNAPSHOT is not expected to be corrected, therefore no need for sequence and correction numbers.*
TS_007	Timeslice (excluding Snapshot) - correctionNumber recommended	TimeSlices with interpretation value BASELINE, PERMDELTA, or TEMPDELTA) should have a not null correctionNumber. *Justification: This would simplify the sorting algorithm in order to identify the valid TimeSlice, as an empty correctionNumber would otherwise need special treatment.*
TS_008	Timeslice - '0' not for a real correction	If a TimeSlice has correctionNumber equal-to '0', then another TimeSlice of the same feature and with the same interpretation value (BASELINE, PERMDELTA, TEMPDELTA) shall not have a null correctionNumber. Justification: according to the usage rules, an empty correctionNumber is assumed to be treated as value "0" by the sorting algorithms that identify the valid TimeSlice.
TS_009	Valid BASELINE cannot overlap	Valid BASELINE TimeSlices of the same feature shall not have overlapping or intersecting validTime periods. *Justification: basic rule of the temporality concept. Otherwise, the baseline status could not be determined.*
TS_010	Valid PERMDELTA cannot overlap	Valid PERMDELTA TimeSlices of the same feature shall not have the same TimeInstant. *Justification: basic rule of the temporality concept, as the PERMDELTA is defined as the difference between two consecutive BASELINE.*
TS_011	Valid TEMPDELTA cannot overlap on same property	Valid TEMPDELTA TimeSlices of the same feature shall not have overlapping or intersecting validTime periods and modify the same property. *Justification: the current NOTAM practice forbids two NOTAM to refer to the same subject/condition. Also, it becomes complicated to identify the actual value of the property.*

		Note: This rule was introduced in version 1.1 of the Temporality Concept in order to facilitate the calculation of SNAPSHOT TimeSlices (feature status at a moment in time). The previous version allowed overlapping TEMPDELTA TimeSlides. Therefore, a very limited number of existing Digital NOTAM implementations might not comply with this rule.
TS_012	TimeSlice validity shall be UTC	Each validTime.TimeInstant.timePosition, validTime.TimePeriod.beginPosition or validTime.TimePeriod.endPosition shall have "Z" as timezone. *Justification: basic rule of the temporality concept, based on the aeronautical information practice.* *Note: TimePeriod and TimeInstant are restricted through the OGC 12-028r1 GML Profile for Aviation to "dateTime" data type only.*
TS_013	Feature lifetime shall be UTC	Each featureLifeTime.TimePeriod.beginPosition or featureLifeTime.TimePeriod.beginPosition shall have "Z" as timezone *Justification: basic rule of the temporality concept, based on the aeronautical information practice.*
TS_014	FeatureLifetime start not after TimeSlice validTime start	The begin position of a featureLifetime shall not be after the begin position of any valid TimeSlice *Justification: Otherwise there would be a contradiction between the status of the feature (properties would have values) at the start of validity of the TimeSlice and the start of the featureLifetime, which indicates that the feature does not exist.*
TS_015	FeatureLifetime end not before TimeSlice validTime end	The end position of a featureLifetime shall not be before the end position of any valid TimeSlice *Justification: Otherwise there would be a contradiction between the status of the feature (properties would have values) at the end of validity of the TimeSlice and the end of the featureLifetime, which indicates that the feature does not exist anymore.*
TS_016	FeatureLifetime not allowed in TEMPDELTA	A TEMPDELTA TimeSlice shall not contain featureLifetime *Justification: It does not make sense to modify temporarily the start or end of lifetime of a feature. That would contradict the meaning of the lifetime concept.*
TS_017	Active TimeSlices cannot be modified	The validTime.TimePeriod.beginPosition of a BASELINE or TEMPDELTA TimeSlice (except for correction TimeSlices only about the end of validity) shall be after the date/time of the issue of the TimeSlice. *Note: this is only applicable to data provider systems. A data user system might receive or process the data with a delay that range from milliseconds to days or even weeks.* *Justification: It is not allowed to use a correction TimeSlice for modifying properties (other than end of validity) for a TimeSlice that is already*

		active. Instead, a new TimeSlice with a validTime.TimePeriod begin position *after* the current date/time *(thus with a different sequenceNumber) shall be used.*
TS_018	PERMDELTA valid in the past cannot be corrected	The validTime.TimeInstant of a PERMDELTA TimeSlice shall be after the time position when it was issued. <u>Note</u>*: this is only applicable to data provider systems. A data user system might receive or process the data with a delay that range from milliseconds to days or even weeks.* <u>*Justification:*</u> *Basic principle of the temporality concept - that only the present and the future can be corrected (with a TimeSlice having a higher correction number and the same sequence number).*
TS_019	Abandoned sequences cannot be reinstated	If there exist a TimeSlice with validTime unspecified, there cannot exist a TimeSlice with the same sequenceNumber and a higher correctionNumber <u>*Justification*</u>*: in order to keep the verification rules as simple as possible, Reuse of "abandoned change" sequenceNumber could complicate the wording of some other rules.*
TS_020	24:00:00 cannot be used	The string "T24" cannot appear in the value of TimeInstant.timePosition, TimePeriod.beginPosition or timePeriod.endPosition

4.4 Usage rules

4.4.1 Time period interpretation (start and end time instants)

BASELINE and TEMPDELTA TimeSlices have a time interval as period of validity. This is actually expressed with two time instants - a beginPosition and an endPosition on the timeline. It is important to clarify how the start and end time instants are considered with regard to the time interval - are they included or excluded from the time interval? This is particularly important when querying AIXM data sources using a time instant or time period criteria.

AIXM uses the gml:TimePeriod element, defined in the GML schema. The semantic of this element is defined by ISO 19108 TM_Period: "*The period is a one-dimensional geometric primitive that represents extent in time. The period is equivalent to a curve in space. Like a curve, <u>it is an open interval</u> bounded by beginning and end points (instants), and has length (duration). Its location in time is described by the temporal positions of the instants at which it begins and ends; its duration equals the temporal distance between those two temporal positions.*"

In the aeronautical information domain, when a change is announced, it is considered that it already applies at the time instant when it starts. For example, if a NOTAM announces an airspace closure starting at 16:00, then the airspace should be shown in status 'closed' already at 16:00:00.00000. The same principle is applied to static data, when a changed or new element is considered to already exist at the effective time, such as the start of an AIRAC cycle.

When a feature is withdrawn, it is considered to no longer exist at time instant for which the withdrawal is announced. For NOTAM, the end of validity is sometimes expressed with hh:59 in order to indicate that the effective time is up to but not including (hh+1):00. However, that requires an additional convention, that the end minute is included, which is not usually applied in numerical comparisons in standard software.

Therefore, the convention that best matches the current aeronautical information practice is to consider that time periods that express a period of validity include the start time instant and exclude the end time instant. This convention is applied in all known AIXM implementations.

This means that when used in AIXM, the semantic of gml:TimePeriod will deviate from the ISO 19108 standard by considering that the start time is actually included in the period. Thus, for TimeSlice.validTime and for featureLifetime.validTime the following convention shall be applied:

- the beginPosition of a TimePeriod is considered as included in the time period;
- the endPosition of a TimePeriod is considered as excluded from the time period.

The endPosition remains excluded from the time period, which is in line with ISO 19108. This also avoids an overlap in the validity of two consecutive TimeSlices, for example two BASELINE.

A secondary aspect concerns the 'end of day' formulation in a TimeInstant or as endPosition of a TimePeriod. ICAO Annex 15 refers in § 1.2.3.1 to the ISO 8601 International Standard for "*Data elements and interchange formats — Information interchange — Representation of dates and times*". The currently published version 8601:2004 allows hour values from 0 to 24:00. However, this will be changed in the new version of the standard as the Draft 8601-1 (2016) amendment proposed to remove the "24" from the allowed values for hours. The ICAO Annex 5 attachment E also indicates that "*Hours should be represented by two digits from 00 to 23*", which implicitly excludes the use of "24:00". Therefore, the value 24:00:00 shall not be used in AIXM as TimeInstant or as TimePeriod beginPosition or endPosition value. Instead, 00:00:00 of the next day shall be used. This is also specified as a coding rule (TS 020) in section 4.3.

4.4.2 BASELINE Time Slices with indeterminate end of validity

The operationally significant changes in the aeronautical information domain are regulated by the AIRAC cycle. Usually, when a permanent change is communicated, it is unknown when the next permanent change will take place. Therefore, it triggers the encoding of a BASELINE with an unknown end of validity. This is expressed in GML as "<gml:endPosition indeterminatePosition="unknown"/>". This BASELINE will cover the period until the next permanent change. Implicitly, when the next change occurs, the previous BASELINE gets an end of validity and needs to be updated/corrected.

The situation may be represented as in the diagram below. The first BASELINE, created at the start of life of the feature, initially has an unknown end of validity. It is represented on this diagram as "BASELINE 1", assuming that it has sequenceNumber=1.

When the permanent change "PERMDELTA 2" occurs, the validity of the initial BASELINE ends and a new BASELINE takes over. To completely represent the history of the feature, a corrected version of the first BASELINE is instantiated (having the same sequenceNumber=1 and also a correctionNumber=1), this time with a known end of validity. The newly created BASELINE has sequenceNumber=2 and no correction yet.

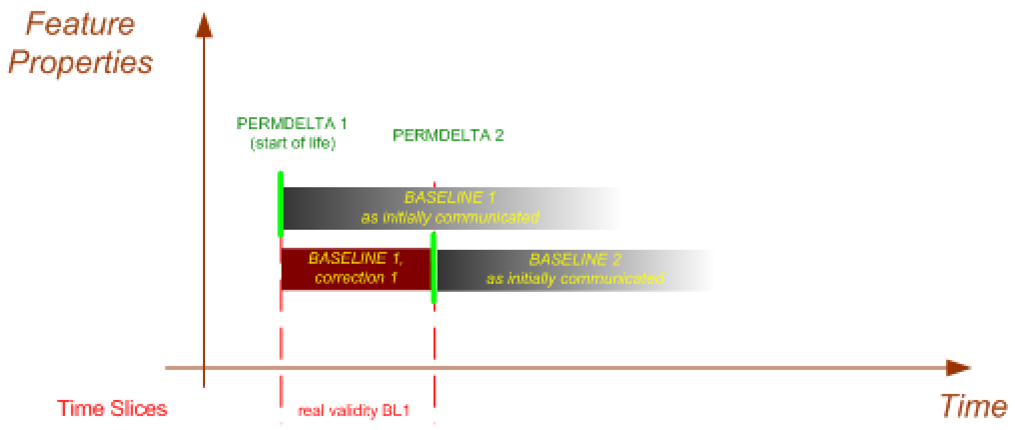


Figure 16 – Previous BASELINE correction as result of a PERMDELTA

4.4.3 SequenceNumber values

As explained in 3.6, the sequenceNumber is primarily used as an identifier of the TimeSlice, in order to apply a correction. Therefore, the sequenceNumber shall be unique per feature and per type of TimeSlice and should be persistent. It is not allowed to change the sequenceNumber of a TimeSlice because this could break the link with a correction TimeSlice and there is no mechanism in AIXM that would enable notifying the change of a sequenceNumber.

For example, sequenceNumbers could be allocated for each AIXM feature starting from “1” and could be incremented by 1 unit (“2”, “3”, “4”, etc.) each time that a new TimeSlice of that kind is encoded for the same feature:

- The initial PERMDELTA that creates the feature to have sequenceNumber=1 and the first BASELINE that results also has sequenceNumber=1;
- The second PERMDELTA (the first change of the feature after its creation) to have sequenceNumber=2 and the resulting BASELINE to also have sequenceNumber=2, etc.
- Then, the first TEMPDELTA that occurs to have sequenceNumber=1, the next one sequenceNumber=2, etc.

The result of this example is visible in Figure 17 – Complete history of a feature.

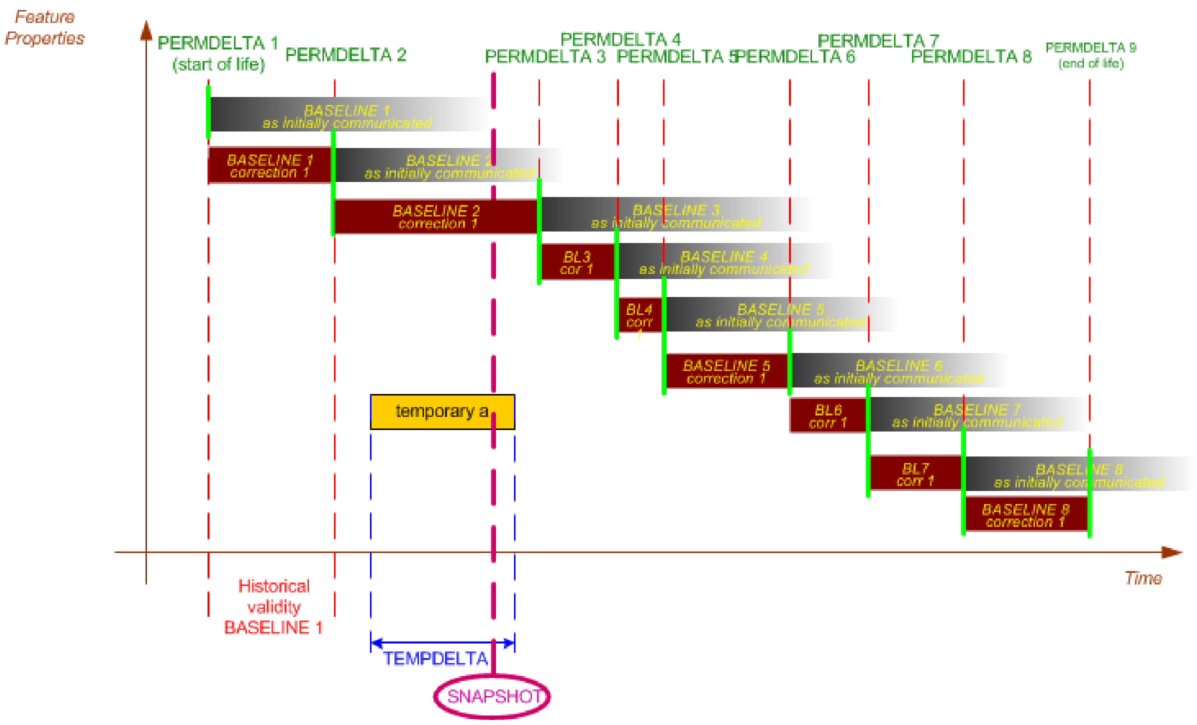

Feature Properties
PERMDELTA 1 (start of life)
PERMDELTA 2
PERMDELTA 3
PERMDELTA 4
PERMDELTA 5
PERMDELTA 6
PERMDELTA 7
PERMDELTA 8
PERMDELTA 9 (end of life)
BASELINE 1 as initially communicated
BASELINE 1 correction 1
BASELINE 2 as initially communicated
BASELINE 2 correction 1
BASELINE 3 as initially communicated
BL3 cor 1
BASELINE 4 as initially communicated
BL4 corr 1
BASELINE 5 as initially communicated
BASELINE 5 correction 1
BASELINE 6 as initially communicated
BL6 corr 1
BASELINE 7 as initially communicated
BL7 corr 1
BASELINE 8 as initially communicated
BASELINE 8 correction 1
temporary a
Historical validity BASELINE 1
TEMPDELTA
SNAPSHOT
Time

Figure 17 – Complete history of a feature

4.4.4 Correction number values

Correction numbers are used in order to identify which TimeSlice of a given feature and of a given type of TimeSlice, from those that have the same sequenceNumber, is the latest valid one. A simple numerical comparison is expected to be used by applications. However, the correctionNumber is not mandatory according to the AIXM schema. This can complicate the sorting by number algorithm in case an initial TimeSlice is left with an empty correctionNumber. Therefore, it is recommended that first instance of a TimeSlice (other than Snapshot) has correction number = 0. Then, the first correction of this time slice should have correction number = 1, and incremented by 1 unit at each correction.

For example, it should be (for each type of TimeSlice, except SNAPSHOT):

- first TimeSlice sequenceNumber= 1, correctionNumber = 0
- with its first correction sequenceNumber= 1, correctionNumber=1, etc.
- then a second TimeSlice sequenceNumber= 2, correctionNumber = 0
- with its first correction sequenceNumber= 2, correctionNumber=1, etc.

The following way of using correctionNumber is incorrect:

- first sequenceNumber= 1, no correctionNumber
- then sequenceNumber= 1, correctionNumber=0

4.4.5 Data feature versus real-world object

The featureLifetime property refers to the data about the feature, not the feature itself in the real world! For example, an obstacle that is re-surveyed might appear as a new feature in a database. However, that obstacle might have existed since a long time. From this point of view, the featureLifetime for legacy data might contain the start date when that information starts to exist in the system that is providing the AIXM 5.1 data.

In case of old data that is being migrated, the start of a featureLifetime might be unknown. In this case, there is no obligation to insert a fictitious featureLifetime start, it may be left empty. However, the start of lifetime can never be after the start of validity of any TimeSlice.

4.4.6 PERMDELTA and TEMPDELTA - included properties

By definition, PERMDELTA and TEMPDELTA Time Slices shall contain strictly the values that are updated. A value that is not new/changed/removed shall therefore not appear in a TEMPDELTA or PERMDELTA. The following sections explain how this concept is applied in particular cases, such as complex properties, extensions, etc.

4.4.6.1 “Delta” for simple properties

This rule applies only to features, as explained in the next sub-section. The following table indicates what shall be included, depending on the type of change:

Type of update	***The “Delta” must contain***
new value (for a property that was empty)	the new value
changed value (for a property that already had a value)	the new value
deletion (for a property that had a value and that become empty)	the property shall be present but empty and shall have the attribute nil=true. In addition, it should have a relevant nilReason (inapplicable, missing, unknown, withheld or other:withdrawn)

4.4.6.2 “Delta” for complex properties

Many AIXM features have complex properties that are represented as aggregated classes in the UML model). For example, an AirportHeliport has an associated ElevatedPoint class with role “ARP”.

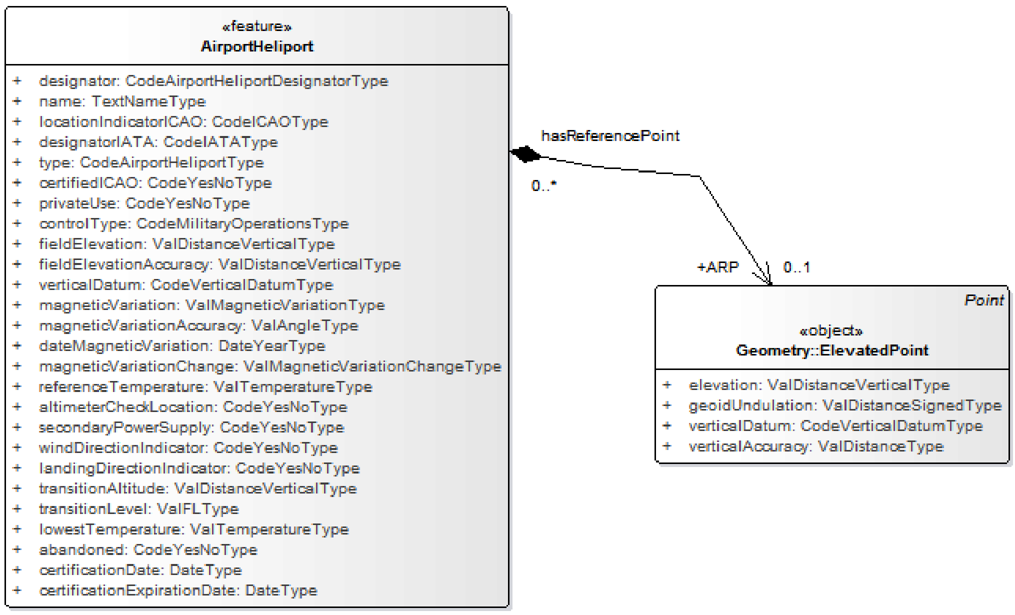

«feature»
AirportHeliport
+ designator: CodeAirportHeliportDesignatorType
+ name: TextNameType
+ locationIndicatorICAO: CodeICAOType
+ designatorIATA: CodeIATAType
+ type: CodeAirportHeliportType
+ certifiedICAO: CodeYesNoType
+ privateUse: CodeYesNoType
+ controlType: CodeMilitaryOperationsType
+ fieldElevation: ValDistanceVerticalType
+ fieldElevationAccuracy: ValDistanceVerticalType
+ verticalDatum: CodeVerticalDatumType
+ magneticVariation: ValMagneticVariationType
+ magneticVariationAccuracy: ValAngleType
+ dateMagneticVariation: DateYearType
+ magneticVariationChange: ValMagneticVariationChangeType
+ referenceTemperature: ValTemperatureType
+ altimeterCheckLocation: CodeYesNoType
+ secondaryPowerSupply: CodeYesNoType
+ windDirectionIndicator: CodeYesNoType
+ landingDirectionIndicator: CodeYesNoType
+ transitionAltitude: ValDistanceVerticalType
+ transitionLevel: ValFLType
+ lowestTemperature: ValTemperatureType
+ abandoned: CodeYesNoType
+ certificationDate: DateType
+ certificationExpirationDate: DateType
hasReferencePoint
0..*
+ARP
0..1
Point
«object»
Geometry::ElevatedPoint
+ elevation: ValDistanceVerticalType
+ geoidUndulation: ValDistanceSignedType
+ verticalDatum: CodeVerticalDatumType
+ verticalAccuracy: ValDistanceType

Figure 18

The question is: what do PERMDELTA or TEMPDELTA Time Slices contain for such situations?

By definition, “delta” Time Slices shall contain strictly the values of the affected feature properties and this rule applies only to features. Objects are considered complex types of a feature property and have to be included in full in a “delta” Time Slice, if the encapsulating feature property is changed. The following table indicates what shall be included, depending on the type of change.

Type of update - for a property included in a complex “object”	***The “Delta” must contain***
new value (for a property that was empty)	the complete complex property (object) with the new value for that property
changed value (for a property that already had a value)	the complete complex property (object) with the new value for that property
deletion (for a property that had a value and that become empty)	the complete complex property (object) with that property also present, but empty. That property shall have the attribute nil=true. In addition, it should have a relevant nilReason (inapplicable, missing, unknown, withheld or other:withdrawn)

Feature properties are all the feature attributes and all the associations for which the feature has the navigability (indicated as an arrow pointing from the feature class towards another class). For example, in the previous class diagram, the properties of the AirportHeliport feature are all attributes (designator, name, …, certificationExpirationDate) and also the “ARP” property, given by the role played by class ElevatedPoint in the association hasReferencePoint. The “ARP” property of the AirportHeliport is a complex one, composed of several attributes. If a temporary or permanent change occurs inside the ElevatedPoint (for example, a modification of the elevation value), then the modified ElevatedPoint shall be included in full in the TEMPDELTA or PERMDELTA Time Slice.

4.4.6.3 “Delta” for multi-occurring properties

An equivalent rule applies for feature properties that can occur multiple times. In the AIXM UML model, such properties are encapsulated in an Object, which is related with the feature class by a 0..* association. For example, an AirportHeliport may serve 0..* Cities, as indicated in the following diagram. This means that the property “serves” of the AirportHeliport feature is potentially multi-occurring.

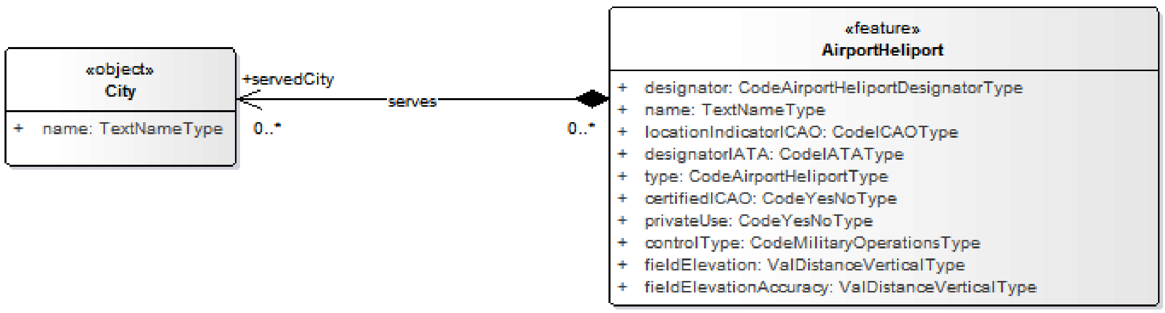

«object»
City
+ name: TextNameType
+servedCity
serves
0..*
0..*
«feature»
AirportHeliport
+ designator: CodeAirportHeliportDesignatorType
+ name: TextNameType
+ locationIndicatorICAO: CodeICAOType
+ designatorIATA: CodeIATAType
+ type: CodeAirportHeliportType
+ certifiedICAO: CodeYesNoType
+ privateUse: CodeYesNoType
+ controlType: CodeMilitaryOperationsType
+ fieldElevation: ValDistanceVerticalType
+ fieldElevationAccuracy: ValDistanceVerticalType

Figure 19

The rule is that, in a PERMDELTA or TEMPDELTA Time Slice, multi-occurring properties shall be provided with all occurrences included. Therefore, if an AirportHeliport had, for example, two served cities and this needs to be permanently changed to three cities, all the three "servedCity" properties have to be included in a PERMDELTA.

The following table indicates what shall be included, depending on the type of change.

Type of update - for a multi-occurring property	***The "Delta" must contain***
additional value	all occurrences shall be included, with the new value being added to the existing ones
changed value for one of the occurrences	all occurrences shall be included, with the changed value for the affected occurrence
removed occurrence	all remaining occurrences shall be included and there shall not be empty elements for the removed occurrences.
all occurrences are removed	one occurrence shall be included, but empty. That property shall have the attribute nil=true. In addition, it should have a relevant nilReason (inapplicable, missing, unknown, withheld or other:withdrawn)

4.4.6.4 "Delta" for extensions

A special situation occurs in the case of AIXM extensions, which are also multi-occurring properties in the 'aixm:'[1] namespace. However, below that level, the actual extension properties have their own dedicated namespace. The rule for multi-occurring properties shall still be applied, but only within the dedicated extension namespace.

For example, if a BASELINE TimeSlice includes:

- aixm:extension/ext1:Object1
- aixm:extension/ext2:Object1

If a PERMDELTA or TEMPDELTA TimeSlice intends to modify only ext1:property1, then it shall contain:

- aixm:extension/ext1:Object1 (the modified value)

It is not necessary to include ext2:Object1 because that is in another namespace. It can be considered as "another property".

The following table indicates what shall be included, depending on the type of change.

Type of update - for a property included in an extension	***The "Delta" must contain***
new value (for a property that was empty)	the extension under that namespace, with the new value for that property and must include all other

[1] Actually "aixm:" is a namespace prefix. The true AIXM namespace is "http://www.aixm.aero/schema/5.1" (for version 5.1). The "aixm:" prefix is used as alias in this section, for editorial convenience.

	properties that have a value *For example: aixm:extension/ext1:Object1.propertyA=”valueA”*
changed value (for a property that already had a value)	the extension under that namespace, with the new value for that property and must include all other properties that have a value *For example: aixm:extension/ext1:Object1.propertyA=”valueB”*
deletion (for a property that had a value and that become empty)	the extension under that namespace, with that property also present, but empty. That property shall have the attribute nil=true. In addition, it should have a relevant nilReason (inapplicable, missing, unknown, withheld or other:withdrawn) *For example: aixm:extension/ext1:Object1.propertyA(xsi:nil=true, nilReason=’inapplicable’)*

4.4.7 Properties with schedule (Timesheets)

Some feature properties may have their own cyclic variation in time according to an established schedule. For example, a navaid can be operational during day time and unserviceable during night time; a restricted airspace could be active every day from 09:00 till 17:00; etc.

To model such situations, the concept of “properties with schedule” has been introduced since AIXM version 5.1. It associates the properties that have cyclic varying values with a “Timesheet” that describes the times when each value is applicable for those attributes. At the feature level, all the properties that change according to an established schedule must be isolated in a separate class, which inherits from an abstract class called “PropertiesWithSchedule”.

The coding guidelines for schedules and their relation with the validity of a TimeSlice are explained in the general AIXM Coding Guidelines.

4.4.8 Canceling a Time Slice (abandoned changes)

In the aeronautical information domain, changes are normally announced in advance of the effective date. Inevitably, this leads to situations where a future change needs to be cancelled, after it was already communicated. This section provides rules for TimeSlice content when encoding the following situations:

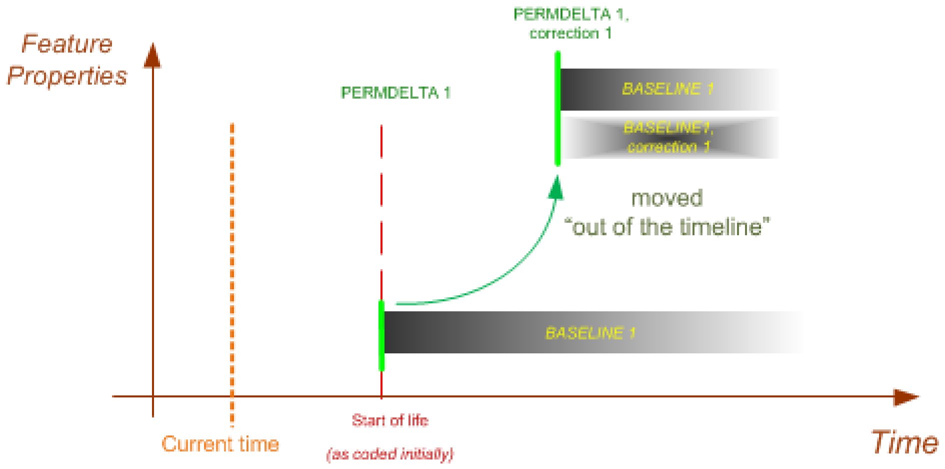
- Abandoning the creation of a feature before its effective date;
- Abandoning a permanent change before its effective date;
- Abandoning a temporary change before its effective date;
- Abandoning the withdrawal of a feature before its effective date.

It was discussed in section 3.6 that the postponement/advancement of an event requires a correction TimeSlice, using the sequenceNumber property as key for identifying the Time Slice concerned. The same concept is applied in the case of cancelled future changes. One or more correction TimeSlice, depending on the actual case, are provided when encoding the cancellation of a future change.

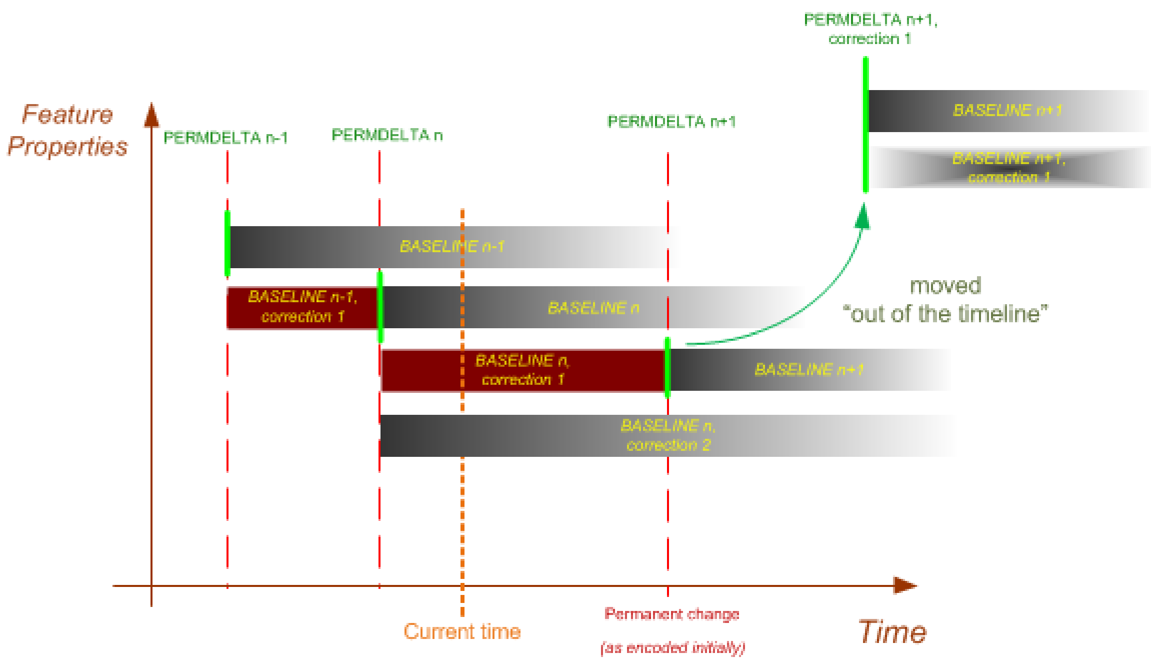
In most situations, a correction TimeSlice will need to "remove from the feature timeline" a TimeSlice that was coded with the initial change. As a general rule, this is done by leaving empty the validTime property and setting the value of its nilReason attribute to "inapplicable". It is not possible to explicitly use the xsi:nil="true" for gml:validTime, because it is not defined in the GML schema for this element.

Depending on the type of change that is abandoned, the following additional rules shall be applied:

- ***abandoned feature creation*** - the cancellation of a future feature creation/commissioning before its effective date shall be coded with a correction TimeSlice that has empty validTime with nilReason="inapplicable", as shown on the following diagram. If a PERMDELTA was also provided, the same needs to be done for the PERMDELTA TimeSlice.

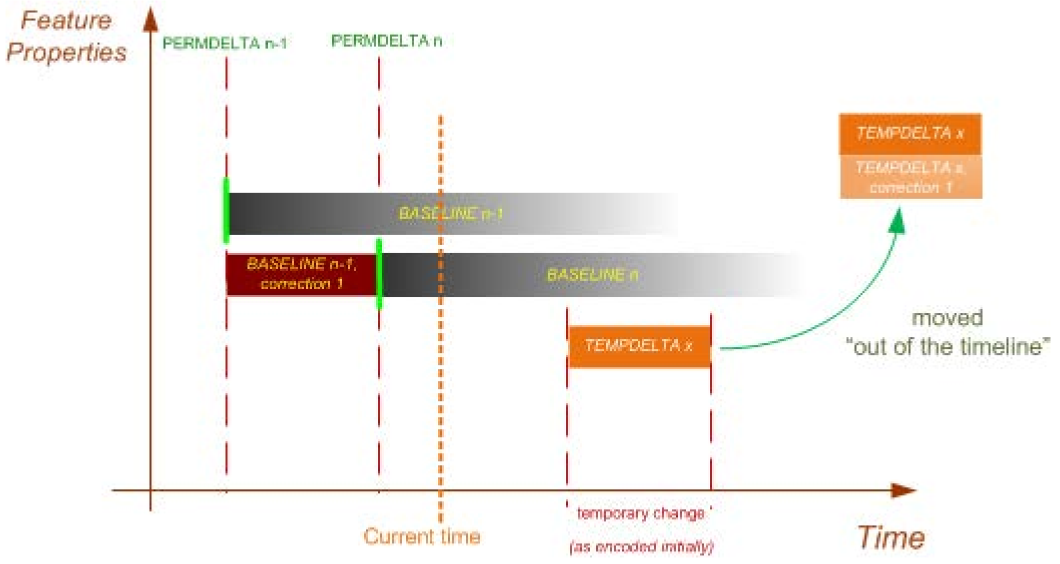


- ***abandoned permanent change*** - the cancellation of a feature permanent change before its effective date shall be coded as shown in the following diagram:
  - first, the future TimeSlices ("n+1" on the diagram) are moved "out of the timeline" with a correction TimeSlice that has an empty validTime property with nilReason="inapplicable";
  - second, with a correction of the previous TimeSlice ("n, correction 2" on the diagram) in order to extend its validTime beyond the time of the abandoned change.

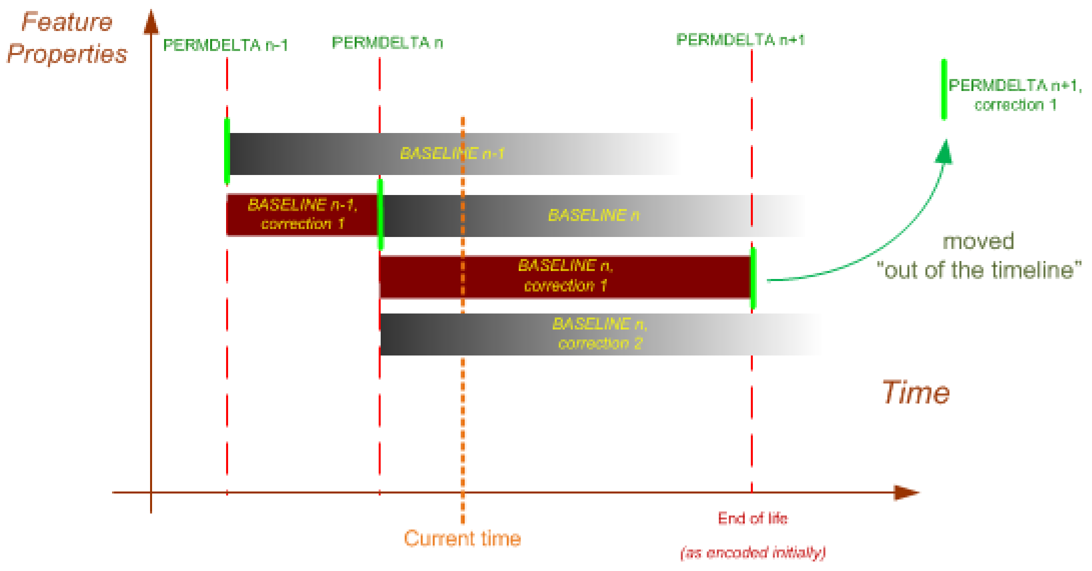


There also exist a simpler alternative, which would consist into leaving the TimeSlice n+1 in place on the timeline, but correcting its values to be equal to the ones of the previous TimeSlice n. That would practically create a 'no change' TimeSlice. The disadvantage of this approach is that it would hide the information that the change was actually abandoned. This approach would also result in an "empty" PERMDELTA (n+1, correction 1).

- ***abandoned temporary change*** - the cancellation of a temporary change before its effective time shall be coded with a correction TimeSlice that has empty validTime with nilReason="inapplicable". This moves the TEMPDELTA TimeSlice "out of the timeline", as shown in the following diagram.


Feature Properties
PERMDELTA n-1
PERMDELTA n
TEMPDELTA x
TEMPDELTA x, correction 1
BASELINE n-1
BASELINE n-1, correction 1
BASELINE n
TEMPDELTA x
moved "out of the timeline"
Current time
temporary change (as encoded initially)
Time

- ***abandoned feature withdrawal*** - the cancellation of a future withdrawal/decommissioning before its effective date shall be coded with a correction TimeSlice that sets back the end of the validity of the previously existing TimeSlice to índeterminate, as shown in the following diagram. In addition, if PERMDELTA are also provided, a correction TimeSlice for PERMDELTA n+1 needs to be coded. This shall have an empty validTime with a nilReason attribute set to “inapplicable”.



4.5 Implementation considerations - limited sub-set of Timeslices

The AIXM temporal model provides considerable flexibility for systems that implement temporality. A system that tried to fully implement the AIXM temporality model and all possibilities for cancelling/correcting TimeSlices could be very complex. Therefore, some systems implementing AIXM might decide to limit the temporality use cases that they support. For example:

- Some systems might only store BASELINE Time Slice data and disregard any temporary changes. Examples include AIP publication, paper chart publishers and ARINC 424 based systems.
- Some systems may only transmit and store temporary changes. Examples include the NOTAM systems. However, such systems need to refer to a source of BASELINE data.
- Some systems may only require periodic snapshots providing the current state of the system. An example is a passive monitoring system designed to report system status at selected time intervals.
- Some systems may want a new “snapshot” after every change without making a distinction between a temporary and a permanent change. Examples include traffic management and flight plan processing systems.
- Some systems may be developed to process and interpret all of the temporal components and provide users with Baseline, Deltas and Snapshot Time Slices at any given moment in time.

When limitations are applied, it is the responsibility of the community concerned to to negotiate specific temporal data exchange requirements as well as to integrate temporality into their internal subsystems.

5. Frequently asked questions

This section provides explanations for aspects of the temporality concept that were raised during the initial development of the AIXM temporality concept.

What was the temporality model of past AIXM versions?
AIXM 3.x and 4.x provide limited temporality support. It is possible to encode the feature state at a point in time (AIXM-Snapshot message) and to communicate baselines (AIXM-Update). AIXM 3.x and 4.x do not support the direct encoding of the temporary status information; it would have to be done as a sequence of two baselines, one changing the properties and the second one reverting to the previous situation. But this does not allow distinguishing between real permanent changes and temporary status information.
In addition, AIXM 3.x and 4.x embed temporality in the exchange message rather than in the feature itself. Consequently, temporality was a property of the message rather than a property of the aeronautical feature. The message properties describe how receiving systems must interpret the message content.
The limited capabilities to transmit temporal information using Update and Snapshot messages in AIXM 3.x and 4.x have led to the development of this more complete AIXM 5 Temporality Concept, at feature level.

Can the start of life and the end of life properties of a feature vary in time?
At first sight, probably not. A feature is created at a moment in time and will cease to exist at another moment in time. But this is true only when considering the already known history of a feature. When exchanging data about the future, there might be situations where the start/end of life is planned to happen at a certain date/time and this date might change.
Therefore, start/end of life of a feature have to be included in the list of time varying properties.

Why not a validity period for each property?
Instead of grouping property values in Time Slices, another approach could be a temporal model where every property gets it own validity period.
The first argument against this approach is that, in general, the properties of a feature do not change independently from each other. There exist operational constraints that link the values of some properties with the values of other properties. Therefore, several properties would have anyhow to be grouped together, with a common validity period.
The second reason is that changes in the aeronautical world are regulated by the AIRAC cycle. This imposes that significant operational changes occur at predefined dates, in order to ensure the predictability of the aeronautical environment and to allow time for the users to accommodate with the changes. In general, aeronautical features have stable property values between AIRAC cycle dates.
Therefore, grouping together the properties in Time Slice with a unique validity period is a simplified temporal model, which supports well the operational requirements.

Why not considering the temporary change as a sequence of two permanent changes?
Using a Time Slice model with BASELINEs only, the interval TS2 would have to be split into 3 new Time Slices, for example TS2a, TS2b and TS2c. In this approach, the temporary situation would be modeled as a sequence of two permanent changes. The disadvantage of this solution is that the information about the temporary nature of the value “w” would be lost. There exist aeronautical applications, such as charting and AIP production, which normally ignore the temporary changes. Such applications need to know if a value is temporary or part of the baseline.
Also, temporary events, such as the activation of a restricted area, have a life of their own: first the activation is requested, than planned for a time interval maybe different from what was requested, than active for maybe a shorter time than planned, etc. In order to correctly model the life of temporary events, they need to be modeled as such and not hidden behind fictitious permanent changes.

Does the model really need to support schedules?
It is obvious that schedules exist in the aeronautical data domain. The question is whether the TimeSlice concept is sufficient or not for also coverings such situations.
Theoretically, the TimeSlice model without schedules can be used for a feature (such as a navaid) that has a property (such as operational hours) that changes cyclically (such as operational every day from 06:00 – 22:00). But this means that either a dedicated BASELINE or a TEMPDELTA is encoded each time when the operationalStatus changes from operational to unserviceable. This would generate either 730 BASELINE TimeSlices or one BASELINE Timeslice and 365 TEMPDELTA TimeSlices in a year, which is a significant inconvenience. In addition, the "cyclical" aspect would not be immediately visible.
Therefore, the TimeSlice model needs to be complemented with a "schedules" concept, which enables to model directly the cyclic variation of the values of one or more feature properties.

Is there any alternative to introducing the "properties with schedule" concept?
Another solution could be to include "schedules" in the TimeSlice concept and make a schedule usable for any feature. That would have two disadvantages.
If an attribute, such as the value of a declared distance, has one value during day and another value during night, each of the two values would need to be part of a different Baseline. Each of the two Baseline would have a schedule that would indicate when they are applicable. But the two Baseline would have overlapping validity times. This would significantly complicate the Temporality concept of AIXM. The analysis also shows that, frequently, schedules really concern just one or two attributes. Having the schedule at the level of the feature would hide this important aspect.
Therefore, the introduction of the attribute with schedule concept is considered the most convenient approach.

References

1. Aeronautical Information Exchange Model (AIXM), Exchange Model goals, requirements and design, December 2006, www.aixm.aero
2. Aeronautical Information Conceptual Model, Edition 1.0, Ref. AIS.ET2.ST01.2000-02, 01 October 1997 (Eurocontrol Extranet, OneSky Teams)
3. "Dynamic Features" Tim Wilson and David Burggraf. September 29, 2005. Contract deliverable to FAA from Galdos Systems Inc.
4. GML: Geography Markup Language. Ron Lake, David S. Burggraf, Milan Trninic, Laurie Rae. Wiley 2004.
5. Temporal Features, James Ressler, Northrop Grumman TASC, OPENGIS PROJECT DOCUMENT #06-076
6. Geographic information - Geography Markup Language (GML), ISO 19136:2007(E) 2007-03-12
7. AIXM Primer. 4.5 draft 2 Edition. EATMP-xxxxxx-xx. Nov. 28, 2005. EUROCONTROL.
8. Annex 15 to the Convention on International Civil Aviation - Aeronautical Information Services. 12th Edition. ICAO. July 2004.
9. AIXM 5.1 - Business Rules (data verification), Using SBVR and Schematron, version 0.7, 08 November 2016.