AIXM 5

Temporality Model

Aeronautical Information Exchange Model (AIXM)

For all inquiries, please contact:

Navin VEMBAR - Navin.Vembar@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.

Table of Contents

1. The need for a temporality model

Time is an essential aspect on the aeronautical information world, where change notifications are usually made well in advance of their effective dates. Aeronautical information systems are usually 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[1] 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).

A temporary change includes the concepts of overlay and reversion. The temporary change is overlaid on the permanent feature state. When the temporary change ends, the temporary changes no longer apply and we revert back to the permanent feature state.

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 The feature is created and withdrawn, just that the life span is shorter than usual.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**. By digital NOTAM we mean replacing the free text contained in a NOTAM message with structured facts, which enable the automated processing of the information.

A general temporal model should be uniformly applied to all aeronautical feature types and the temporality concept should be abstracted from the task of modeling object properties. At the conceptual level, the model should 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 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.

[1] 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.

2. 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.

2.1 (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 not be defined over a time period.

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 assumption 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 feature's properties (P1, P2, … P5) along a timeline.

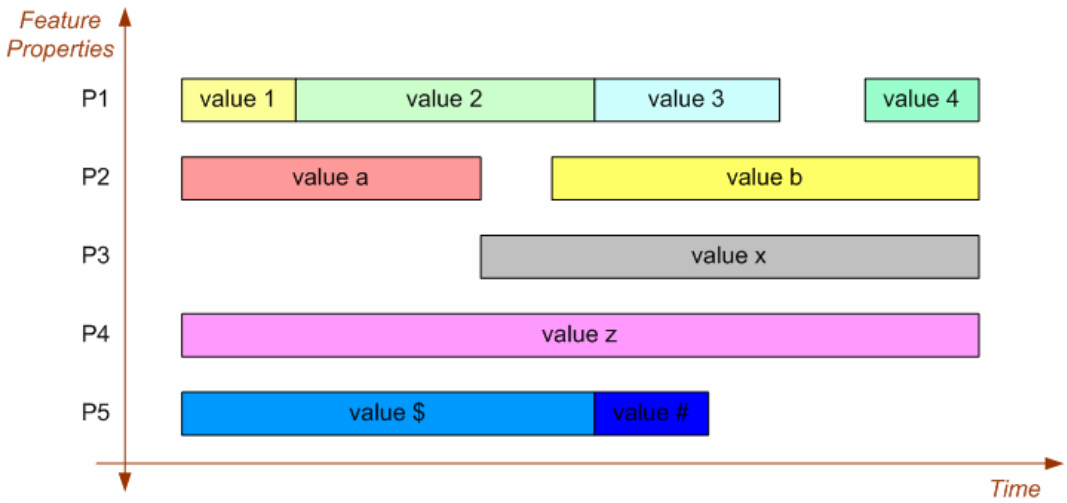


Figure 1 – Time varying properties

Discussion: ***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, we have to include the start/end of life of a feature in the time varying properties list.

2.2 (step 2) The Time Slice model

The temporality model adopted2 by AIXM describes feature events and states. An event is a change of one or more feature properties. A state is the feature property set valid over a time period. An event occurs at the transition between states. In the diagram below events are located at the vertical cuts while states are represented as the feature property set between events.

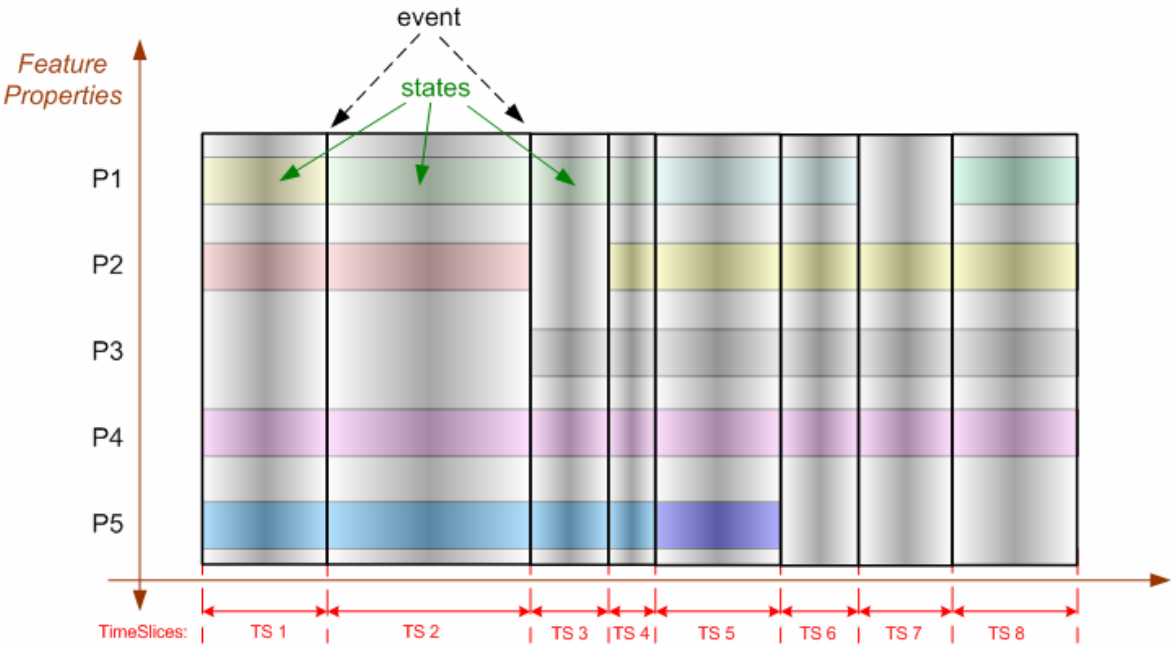


Figure 2 – Events and States

In order to describe the feature properties during states and events, the time varying properties of every feature are encapsulated in a container called "Time Slice". 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 maximum one value for each property and one specified validity period. In an UML diagram, the basic Time Slice concept is represented as below:

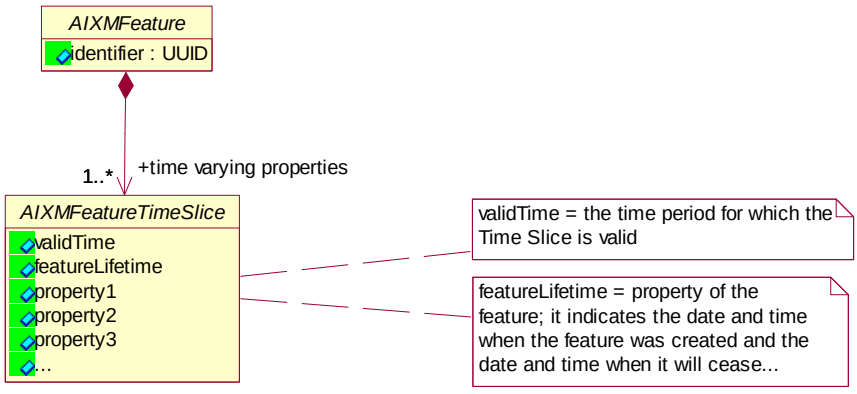


Figure 3 – AIXMFeatureTimeSlice

[2] The AIXM Time Slice model is based on the ISO 19136 (GML) timeslice concept.

Discussion: ***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.

2.3 (step 3) Temporary events – digital NOTAM

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. At the end of the temporary event, the values of these properties are reversed to their static values.

In order to model temporary events, we need to specialise the basic temporality model defined at step 2 by differentiating between two kinds of Time Slices:

- ***Baseline*** = a kind of Time Slice that describes the feature state (the set of all feature's 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 a "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.

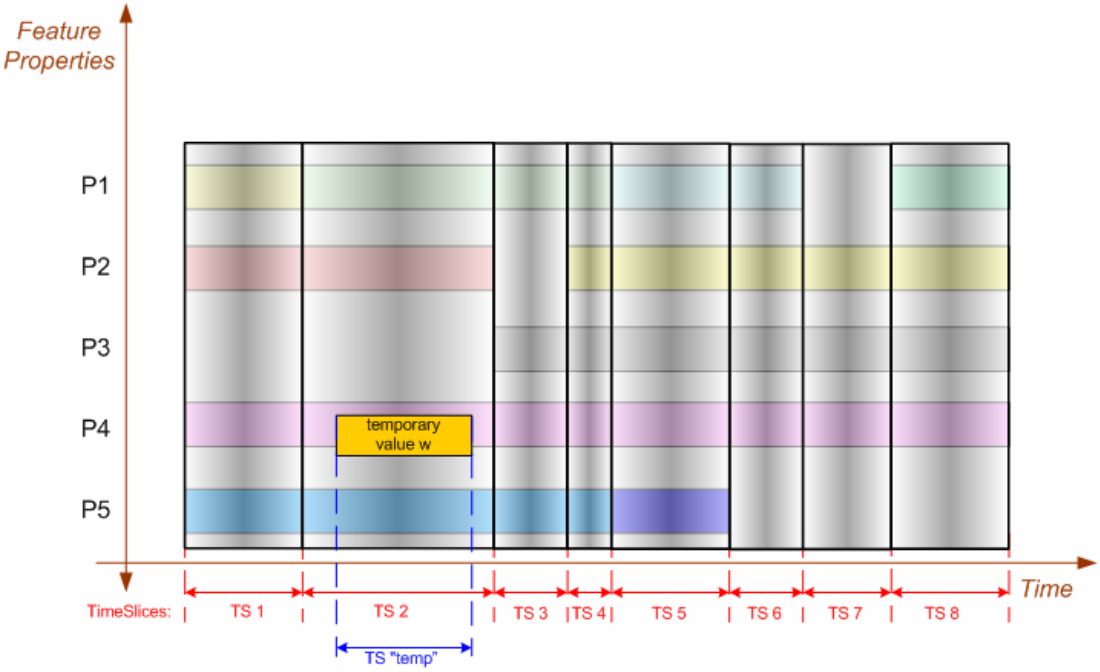


Figure 4 - BASELINE and TEMPDELTA TimeSlices

One reason for temporary Time Slices to contain strictly the modified properties is to avoid confusions that could result from overlapping temporary events. When several temporary delta overlap in time, complicated rules would be necessary in order to decide which values to include for not affected properties. Should the values of the baseline Time Slice be included? Or should the other temporary changes be considered? Therefore, the model is clearer if 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 is named “interpretation” and indicates the type of Time Slice - BASELINE or TEMPDELTA, as shown in the figure below.

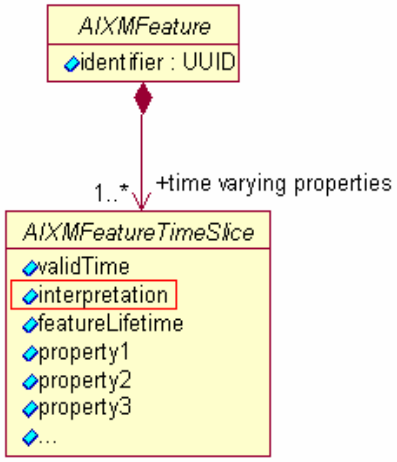


Figure 5 – FeatureTimeSlice with “interpretation” property

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.

Discussion: ***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.

2.4 (step 4) Current Status - SNAPSHOT Time Slices

The temporality model described up to this point complies with the rules for completeness, minimalism, consistency and context-free mentioned at the end of section 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.

However, the model is slightly inconvenient for a real life implementation, because it 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. For convenience, we need an additional kind of Time Slice to be included in the model. This will be named "SNAPSHOT" and will carry the result of merging the effective BASELINE information with all overlaying TEMPDELTA that are effective at a 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 actual BASELINE Time Slice effective at that time instant with all TEMPDELTA Time Slices that are effective at that time instant.

Note that for a SNAPSHOT, the correctionNumber and the sequenceNumber properties shall not be populated.

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

Another kind of Time Slice that will be introduced for convenience is in support to systems that need to notify the clients about data updates. There exist two types of applications:

1. "Pull" Systems - provide an interface by which a client can query the aeronautical information and extract the results of the query;
2. "Push" Systems - generate and transmit to the client notifications about aeronautical information changes.

For "push" systems, it is difficult to use only these three kinds of Time Slice for communicating (generating and transmitting) information about the future. For example, how to communicate information about the end of life (decommissioning) of a feature?

Using BASELINE Time Slices for this purpose would require communicating an 'update' of at least the last sent Time Slice, with an updated value of the 'endOfLife' property (encoded as featureLifetime/endPosition). This would also require interpretation rules such as "if the endOfLife

is equal with the end of validity of the Time Slice, then this means that the feature is permanently withdrawn". The postponement of a withdrawn, which is operationally possible, would require a second update of the endOfLife, which might become complicated to interpret.

A more convenient solution is to include in the temporality model a Time Slice that represents permanent change events. 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 relatively simply 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 client systems no longer need to compare the previously received BASELINE Time Slice with the new one in order to detect the changed attributes. This process may be time consuming and even error prone. The data originator 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 will easily 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.

All these kinds of Time Slices are described in Figure 6.

Conceptually, there exists a direct dependence between PERMDELTA and BASELINE Time Slices. However, this does not mean that the BASELINE Time Slice needs to be effectively instantiated after each PERMDELTA. In an implementation, it is possible, for example, to "accumulate" PERMDELTA Time Slices. The instantiation of a new BASELINE might occur, for example, after each third PERMDELTA affecting a feature.

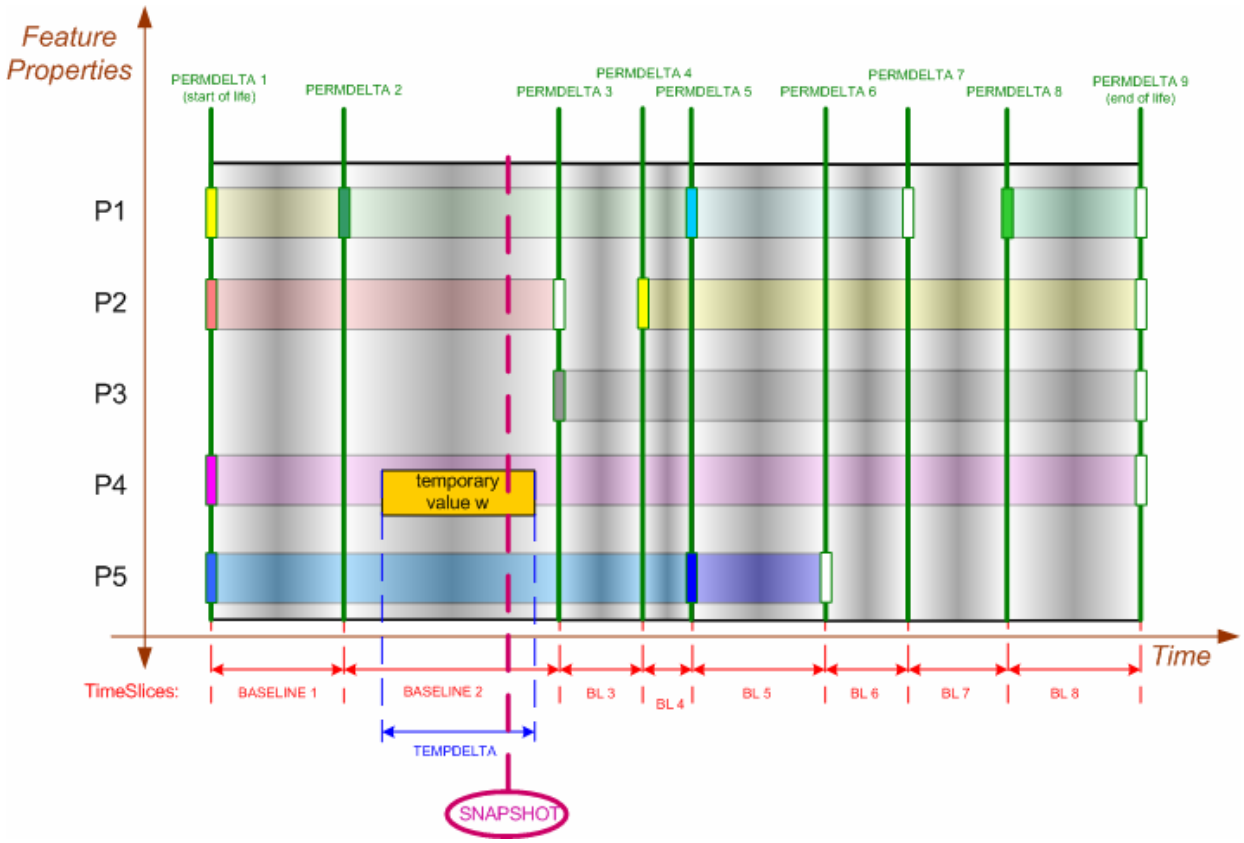


Figure 6 – Four types of TimeSlices

2.6 (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 ***"sequence number" attribute is introduced in the model, playing the role of unique identifier for each Time Slice*** inside a feature.

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 higher correction number will be considered valid.

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

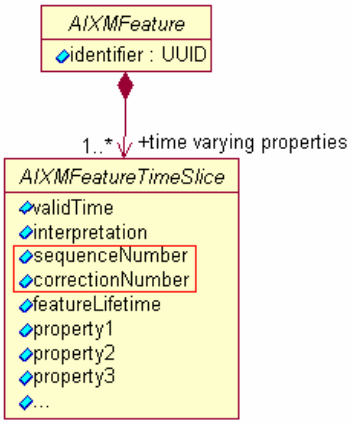


Figure 7 – Complete AIXMFeatureTimeSlice model

To summarise, 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 overlay of a fe***ature state during a temporary event.
- ***SNAPSHOT*** = A kind of Time Slice that describes the state of a feature at a time instant, as result of combining the actual BASELINE Time Slice (valid at that time instant) with all eventual TEMPDELTA Time Slices that are effective at that time instant.

Discussion: ***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 should 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.

2.7 Properties with schedule

The Temporality Model described up to this point works well for features that have properties with constant values during their time of validity. In some cases, one or more properties of a feature 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 in AIXM 5.1. The idea is to associate the properties that have cyclic varying values with a "Timesheet" that describes the times when each value is applicable for those attributes. The concept of Timetable/Timesheet already existed in AIXM 3.x and 4.x. It was inherited as such in AIXM 5.0, where it was found to sometimes overlap with the TimeSlice concept. Therefore, a re-thinking of the role of Timesheets was necessary in AIXM 5.1 (see Change Proposal 5.1-35, which provides a more detailed analysis of the need for schedules in AIXM).

Discussion: ***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.

At the feature level, all the properties that change according to an established schedule must be isolated in a separate class, as illustrated below with class NavaidOperationalStatus. This class inherits from an abstract class called "PropertiesWithSchedule".

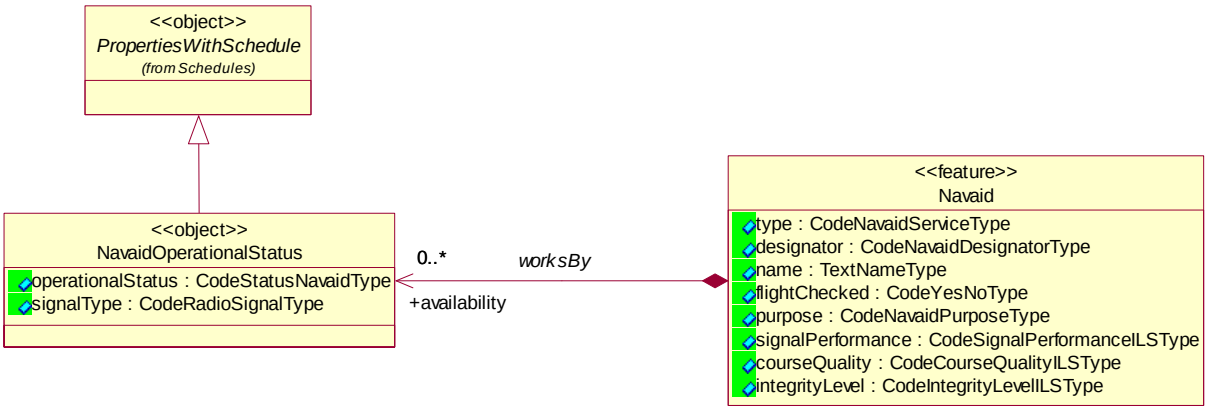


Figure 8 - Properties with schedule model

The abstract class PropertiesWithSchedule is then associated with Timesheet(s).

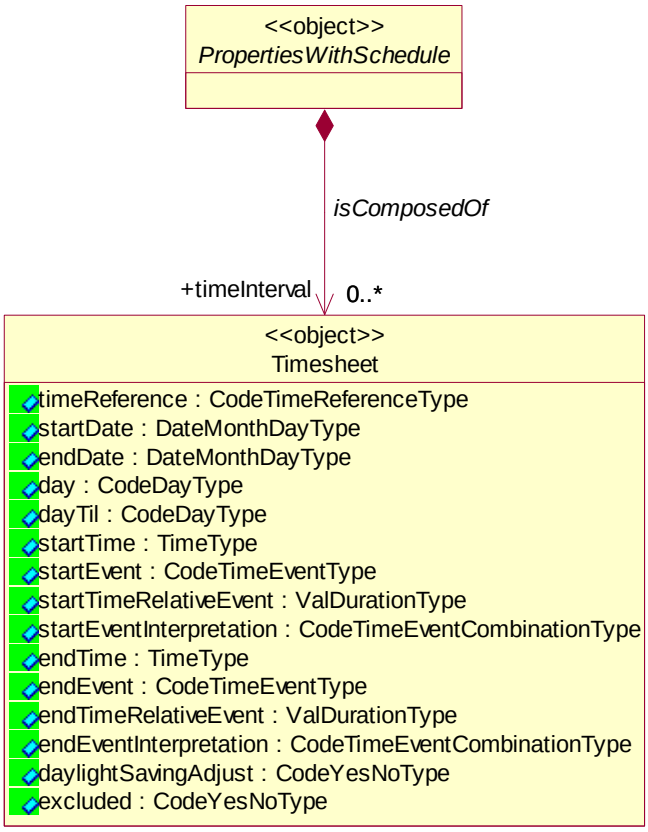


Figure 9 - Timesheet class

The Timesheet class contains the time reference system (UTC-12 to UTC +14), daylight saving indicator and provides the possibility to include/exclude specified dates and times. It can, for instance, represent:

- a single repetitive time period, such as "each Monday from 10:00 to 16:00";
- a single time block spreading over several days, such as "From each Monday 10:00 till Thursday at sunset"
- a date range, such as "every year from 15 OCT to 15 MAY";
- etc…

Discussion: ***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.

The introduction of the PropertiesWithSchedule requires clear rules for interpreting the various combinations that can occur between TimeSlice types, their validity and property schedules. The risk is that the value of a property may be undefined if the schedules associated with a BASELINE leaves "holes". In the AIS operations of today, for properties that change values according to a schedule, it is quite common to specify only the "main" value, such as "operational", "active", etc.. For example, it is indicated that the "navaid operates every day from 06:00 – 22:00", but it is not explicitly indicated what is the status of the navaid between 22:00 – 06:00. Operational people will assume that the navaid is not operating between 22:00 – 06:00.

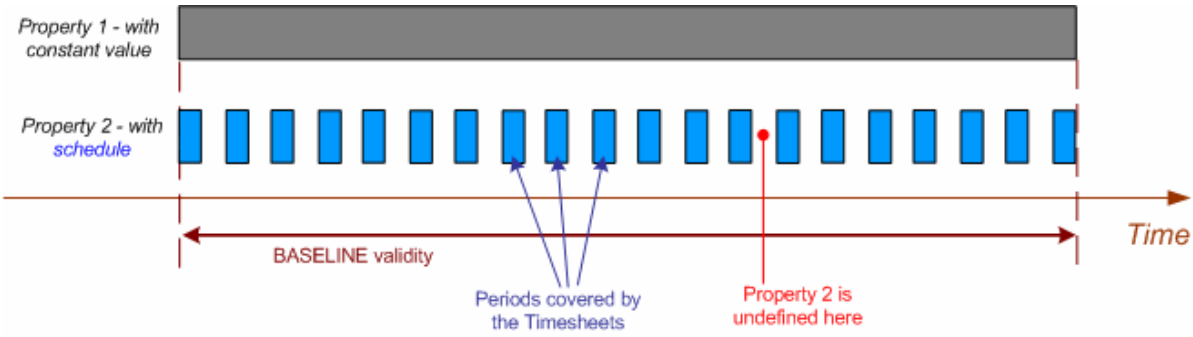


Figure 10 - Schedule with holes

As machines cannot make "assumptions", for digital data processing, it is safer if the "non-operating" times are also stated explicitly, so that schedules associated with the values of the property do not leave any holes. Therefore, it is recommended that BASELINE Timeslices contain only fully defined properties with schedule, which indicate explicitly what the property value is at every moment within the validity time of the TimeSlice. If one or more Timesheets are associated with a property with schedule, than the value of the property shall be considered as undefined at any moment not covered by a Timesheet.

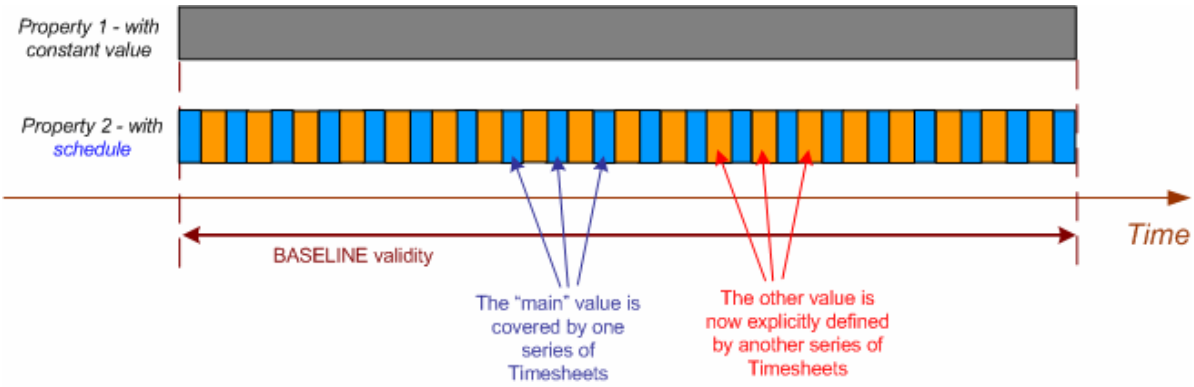


Figure 11 - Fully defined property with schedule

Timesheets that leave gaps can also occur for TEMPDELTA TimeSlices. By definition, any property contained in a TEMPDELTA overrides the value of the equivalent BASELINE property, for the duration of validity of the TEMPDELTA. Therefore, as a general principle, the times encoded in Timesheets contained in TEMPDELTA TimeSlices also replace in full the times encoded in the equivalent BASELINE Timesheets.

The situation is simple and clear if the TEMPDELTA does not have Timesheets or if these Timesheets cover the whole period of applicability of the TEMPDELTA, as presented in the following diagrams:

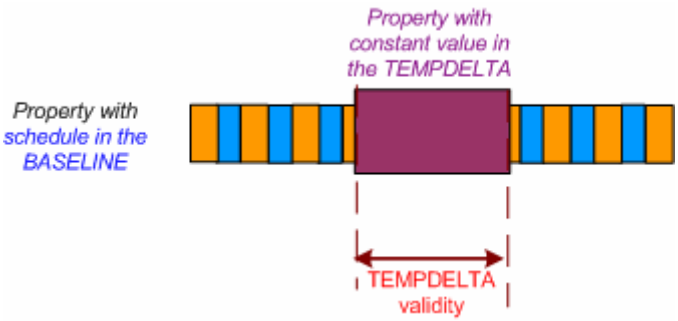


Figure 12 - TEMPDELTA constant value overrides the BASELINE schedule

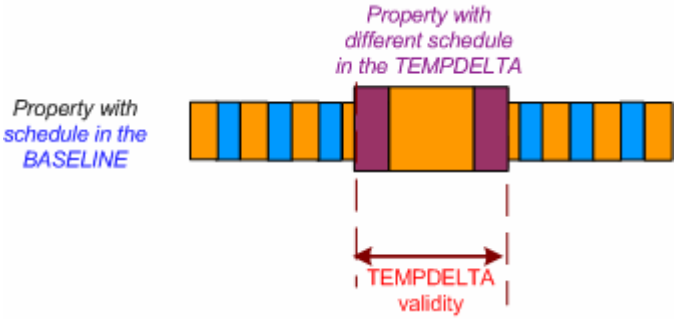


Figure 13 - TEMPDELTA schedule overrides the BASELINE schedule

A situation that can lead to difficulties in interpretation is when the Timesheets associated with the TEMPDELTA leave gaps (times where the value of the property is not explicitly specified), as in the following diagram:

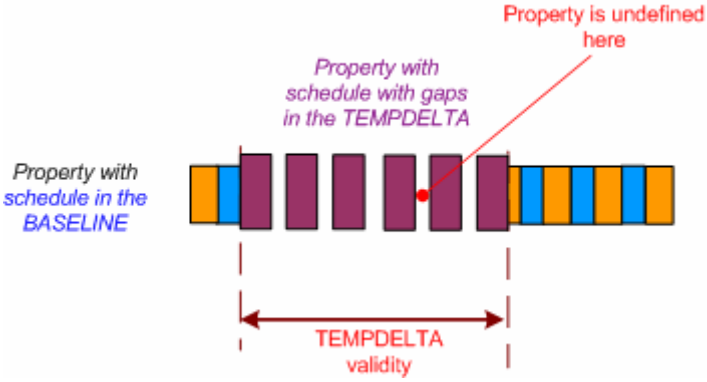


Figure 14 - TEMPDELTA schedule gaps leave the property undefined

The temptation could be to consider that the BASELINE situation applies in the gaps left by the Timesheets associated with the TEMPDELTA. But this would be in conflict with the general principle that the TEMPDELTA values replace the BASELINE values in full. Therefore, if a TEMPDELTA schedule leaves gaps (periods for which the value is not explicitly provided), then it shall be considered that the property has an unspecified value at those time periods.

From the examples above, it is therefore recommended that TEMPDELTA schedules do not leave unspecified periods (gaps) within the time of applicability of the TEMPDELTA.

2.8 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.

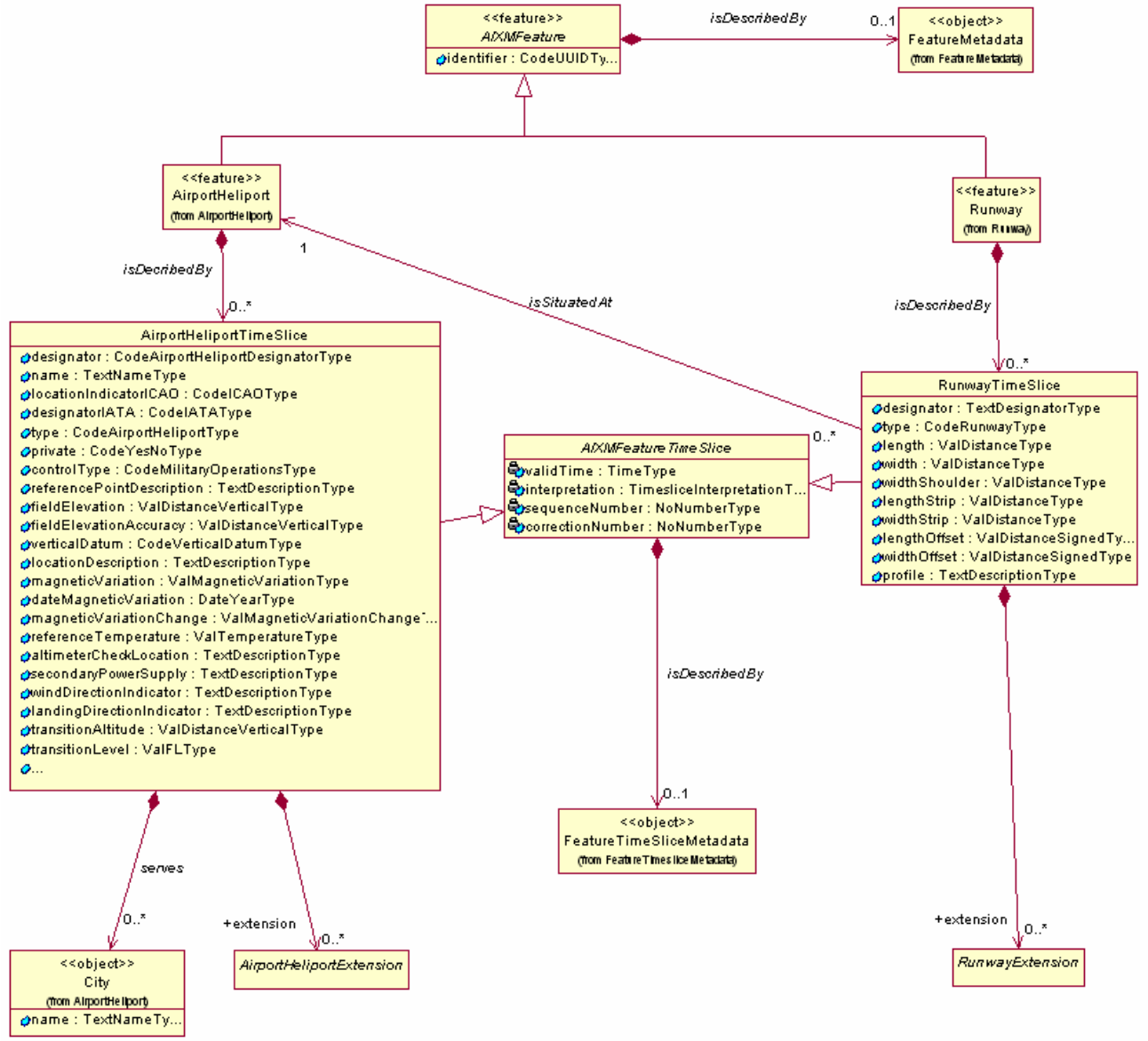


Figure 15 - Model expanded with explicit TimeSlice classes

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 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, when converting from the UML model to the XML Schema of AIXM.

3. Application aspects

3.1 BASELINE Time Slices with undetermined 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 BASLINE 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.

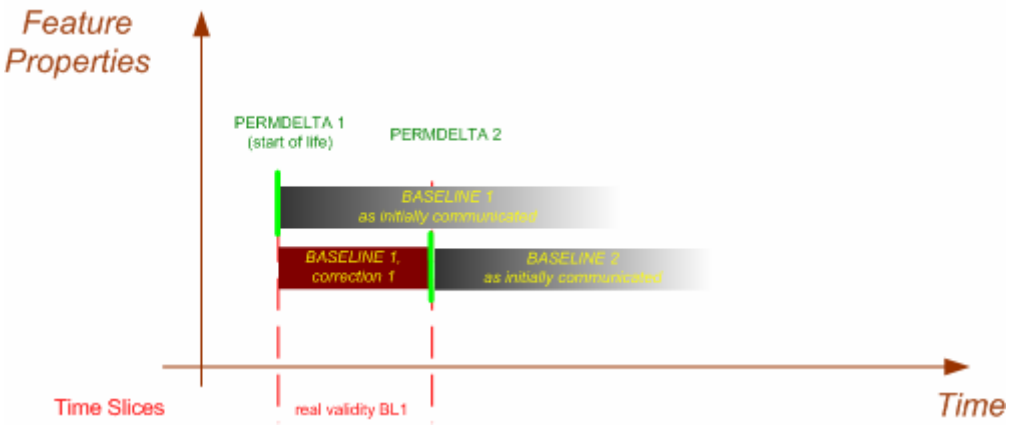


Figure 16 – Previous BASELINE correction as result of a PERMDELTA

3.2 SequenceNumber values

As explained in 2.6, the sequenceNumber is primarily used as an identifier of the TimeSlice, in order to apply a correction. Therefore, the sequenceNumber shall be unique per 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.

A secondary aspect is that sequenceNumbers can also be used to give some chronological information about the time when that TimeSlice was issued (not in which order it becomes valid!).

Therefore, it is recommended that sequenceNumbers are allocated starting from "1" and are incremented by 1 unit ("2", "3", "4", etc.) each time that a new TimeSlice of that kind is encoded:

- The initial PERMDELTA that creates the feature to have sequenceNumber=1 and the first BASELINE that results to also have 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 recommendation is visible in Figure 17 – Complete history of a feature.

3.3 Feature end of life

As explained in 2.5, the PERMDELTA TimeSlices have been introduced in order to facilitate the notification of the end of life / decommissioning / withdraw of a feature. This shall be encoded as a PERMDELTA that changes the featureLifetime/../endPosition property (of the BASELINE TimeSlice valid at the withdraw time) from "undetermined" into a precise date and time value. The effective date of the PERMDELTA shall be equal to the end of life value. No other property of the feature is included in the PERMDELTA in this case, as this PERMDELTA will not result into the establishment of a new BASELINE, just in a correction of the last active BASELINE.Extended to the complete history of the feature, the correction of the initially communicated BASELINES up to the end of life of the feature can be represented as in the following diagram.

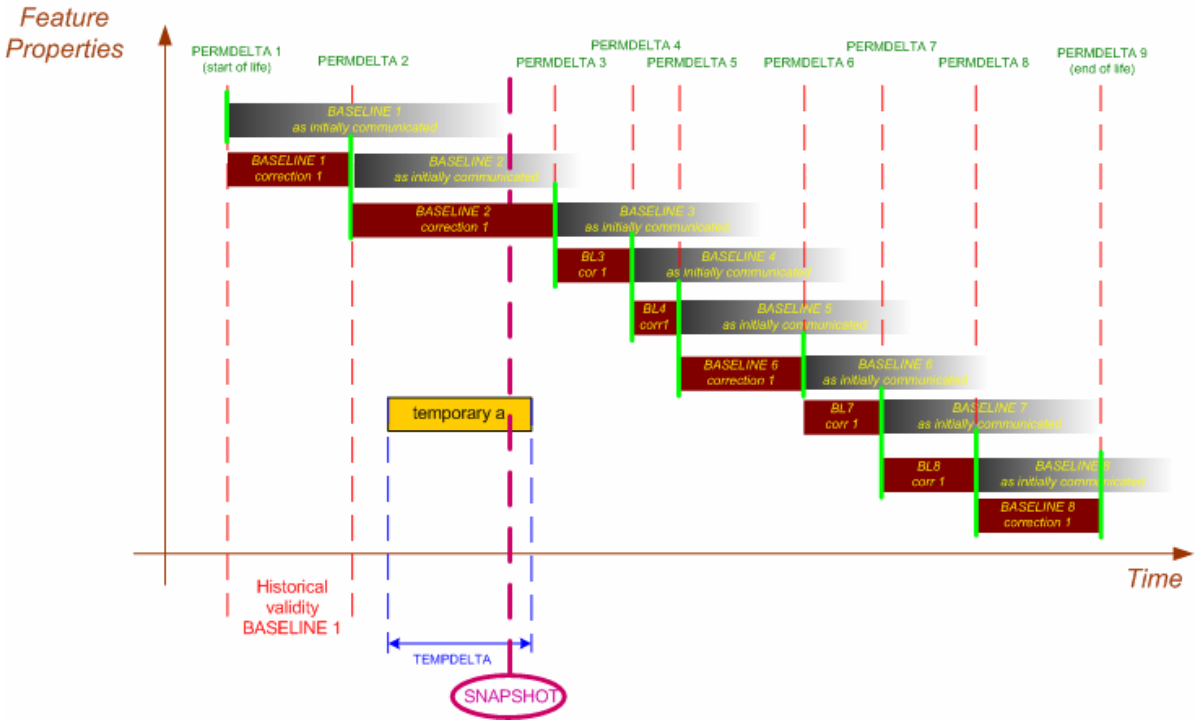


Figure 17 – Complete history of a feature

3.4 "Delta" for complex properties

Many AIXM features have complex properties that are made of zero or more component classes (represented as aggregated classes in the UML model, 0..*). For example, an AirportHeliport has an associated AirportHeliportAvailability which is "composedOf" zero or more AirportHeliportUsage.

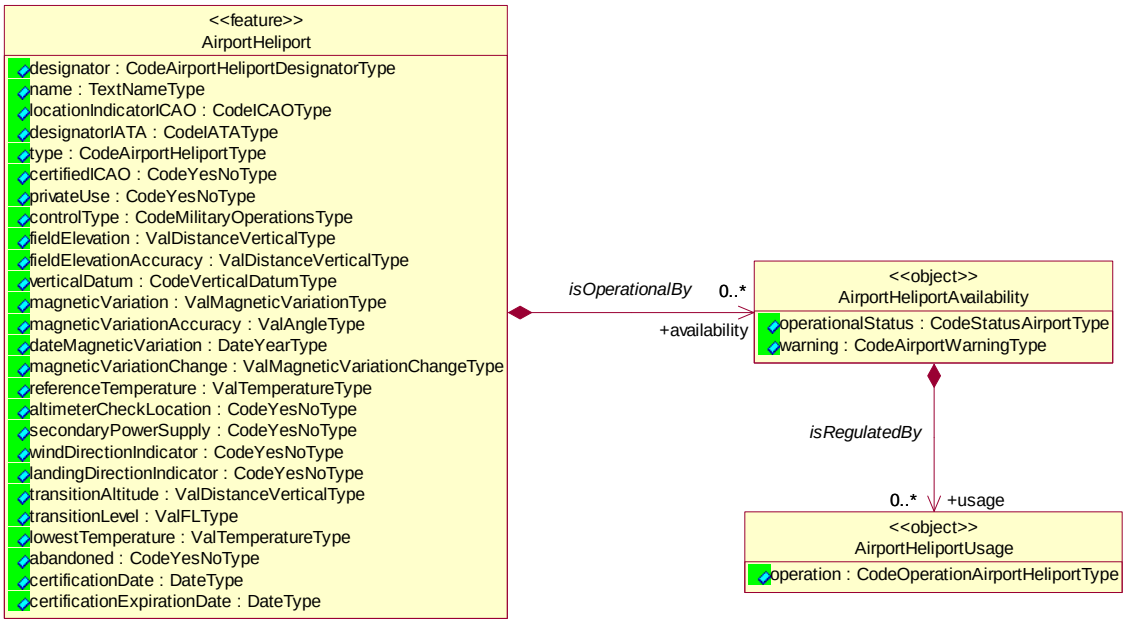


Figure 18

The question is: what should 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 consider 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. This will be explained further down with an example.

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 "availability" property, given by the role played by class AirportHeliportAvailability in the association isOperationalBy. The "availability" property of the Airspace is a complex one, composed of several AirportHeliportUsage. If a temporary or permanent change occurs inside the AirportHeliportAvailability (for example, a modification of one of its composing AirportHeliportUsage), then the modified AirportHeliportAvailability shall be included in full in the TEMPDELTA or PERMDELTA Time Slice.

3.5 "Delta" for multi-occurring properties

An equivalent rule applies for feature properties that can occur multiple times. In UML, 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.

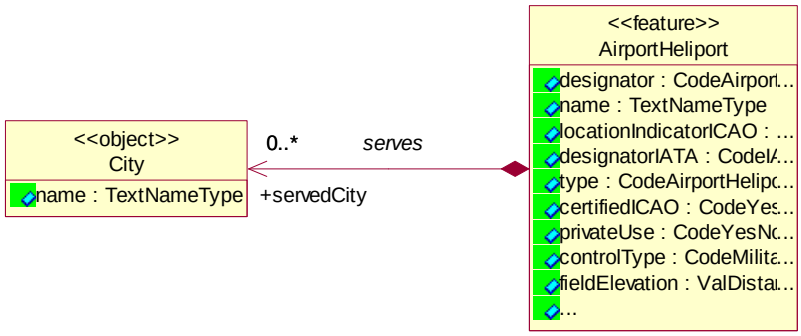


Figure 19

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.

3.6 Identifying the feature affected by a DELTA Time Slice

A Time Slice is always encoded as child element of a feature. As every AIXM feature has 'gml:identifier' property, this should be sufficient for this purpose. This is supposed to be a global unique identifier (of type UUID), which provides an unambiguous key for every AIXM Feature.

However, the global unique identifiers are likely to not exist for some time. In this situation, there are two possibilities:

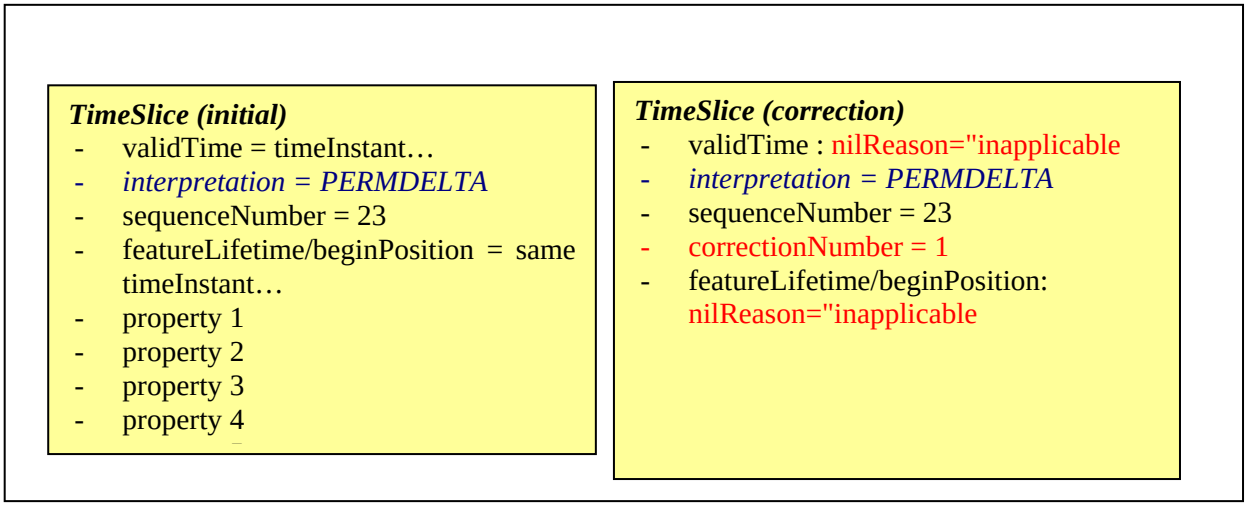
- Either use the gml:identifier property for encoding a local unique identifier (an artificial key), specific to the data originator. In this case, the PERMDELTA and TEMPDELTA Time Slices can be operationally received only from the same originator who has provided the BASELINE data. Using PERMDELTA/TEMPDELTA from another data source would inevitably break the chain, as different identifiers would be used.
- Or, in addition to the PERMDELTA or TEMPDELTA Time Slice, include in the AIXM file a SNAPSHOT Time Slice, which contains some properties (a "natural key") that are sufficient for identifying the feature. The fact that the SNAPSHOT contains just some natural key properties and does not contain all properties is not in conflict with the definition of a SNAPSHOT TimeSlice, because a SNAPSHOT represents the view that a particular system has on that feature, which might be an incomplete view. The recipient of the data will have to query his local system and identify the feature that has the same values at that moment in time, thus being identified as the target for the update.

3.7 Canceling a Time Slice (abandoned changes)

For aeronautical information systems that work in "push" mode, the primary means for generating and providing information about a change is by PERMDELTA and TEMPDELTA Time Slices. The question is what procedures shall be applied in the case of a change in the planning, such as:

- Abandoning the commissioning/decommissioning of a feature (before its effective date)
- Abandoning a permanent change (before its effective date)
- Abandoning a temporary change (before its effective date)

It was already discussed (see 2.6) that the postponement/advancement of an event requires a correction to a TimeSlice, using the sequenceNumber property as key for identifying the Time Slice concerned. The sequenceNumber will also be used for identifying the PERMDELTA or TEMPDELTA that needs to be abandoned. To clearly indicate that the change contained in the TimeSlice has been canceled, the gml:validTime property will be empty and it will have the nilReason attribute set to "inapplicable". For example, if a PERMDELTA for SomeFeature has been provided, with sequenceNumber "23", in order to cancel it, a second PERMDELTA with the same sequenceNumber and a higher correctionNumber has to be issued, as below



Note that this Time Slice cancellation does not affect 'pull' systems, such as Web Services or WFS, where the system provides the most current information, following an on-line request by the client. The client is not supposed to refer to or compare the results with the results of a previous query.

3.8 Overlapping TimeSlices and corrections

The sequenceNumber and correctionNumber are used to resolve and interpret overlapping timeSlices. Consider the scenario shown in the figure below where a Feature's Status property is changed repeatedly over several overlapping time intervals. Each temporary change receives a sequenceNumber. In the example, one of the Time Slices is corrected, leading to a duplicated sequenceNumber and different correctionNumber.

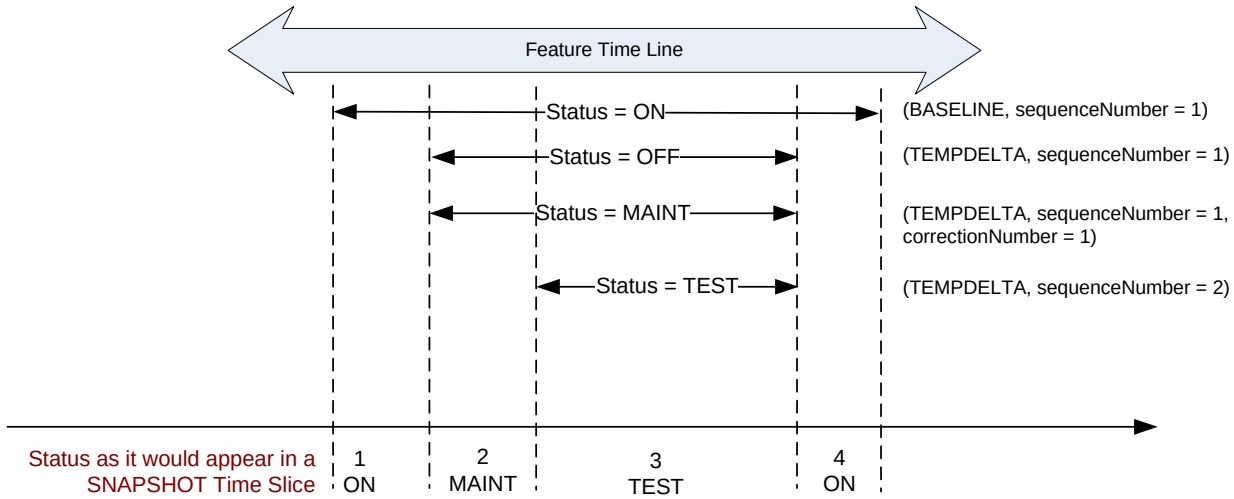


Figure 20 Example of TimeSlice corrections and overlapping

At the edges of each temporal event we can identify transitions to different feature versions. The combination of Time Slice type and sequenceNumber can be used to unambiguously identify the value of the Feature's Status property at each moment in time.

To determine the value of a property at a given time or over a given time interval the following rules should be used:

1. Identify the BASELINE that is valid at that moment in time, by looking at it's validTime. If several exist, they should all have the same sequenceNumber and different correctionNumbers. Take the one with the highest correctionNumber;
2. Identify all TEMPDELTAS that are effective at the specified time ;
3. Sort the TEMPDELTAs by increasing sequenceNumber;
4. Apply the TEMPDELTAs to the feature from low sequenceNumber to high sequenceNumber.
    a. When two or more deltas have the same sequenceNumber apply the delta with the highest correctionNumber.

The possibility to resolve overlapping TEMPDELTAs using the sequenceNumber and correctionNumber shows how cancellations and corrections can be communicated. In this example, TEMPDELTA sequenceNumber = 1 is initially used to communicate that the feature Status = OFF. Later, a Time Slice correction is transmitted using the same sequenceNumber = 1 but with a correctionNumber = 1; it corrects the feature state to Status = MAINT. However, the final status is later given by the TEMPDELTA with sequence number 2 which indicates the feature Status = TEST.

3.9 Other implementation considerations

The conceptual temporal model described in the previous section provides considerable flexibility for systems that implement temporality. A system that tried to fully implement the AIXM temporality model would be very complex. However there is no requirement for systems implementing AIXM to support all kinds of Time Slices. For example:

- Some systems may 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 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 that can process and interpret all of the temporal components and provide users with Baseline, Deltas and Snapshot Time Slices at any given moment in time.

AIXM contains a complete temporal model; however, as the examples illustrate it is the responsibility of interacting systems to negotiate specific temporal data exchange requirements as well as to integrate temporality into their internal subsystems.

4. Usage examples

4.1 Navaid example

Figure 21 illustrates the temporal model by showing a transmission frequency change for a navigation aid (VOR AML, from 112.0 MHz to 113.2 MHz). Normally, this change should occur at an AIRAC cycle date. Usually, the change requires the navaid to be out of service for a certain time, then to be on test on the new frequency. The temporary status is communicated at present through NOTAM messages.

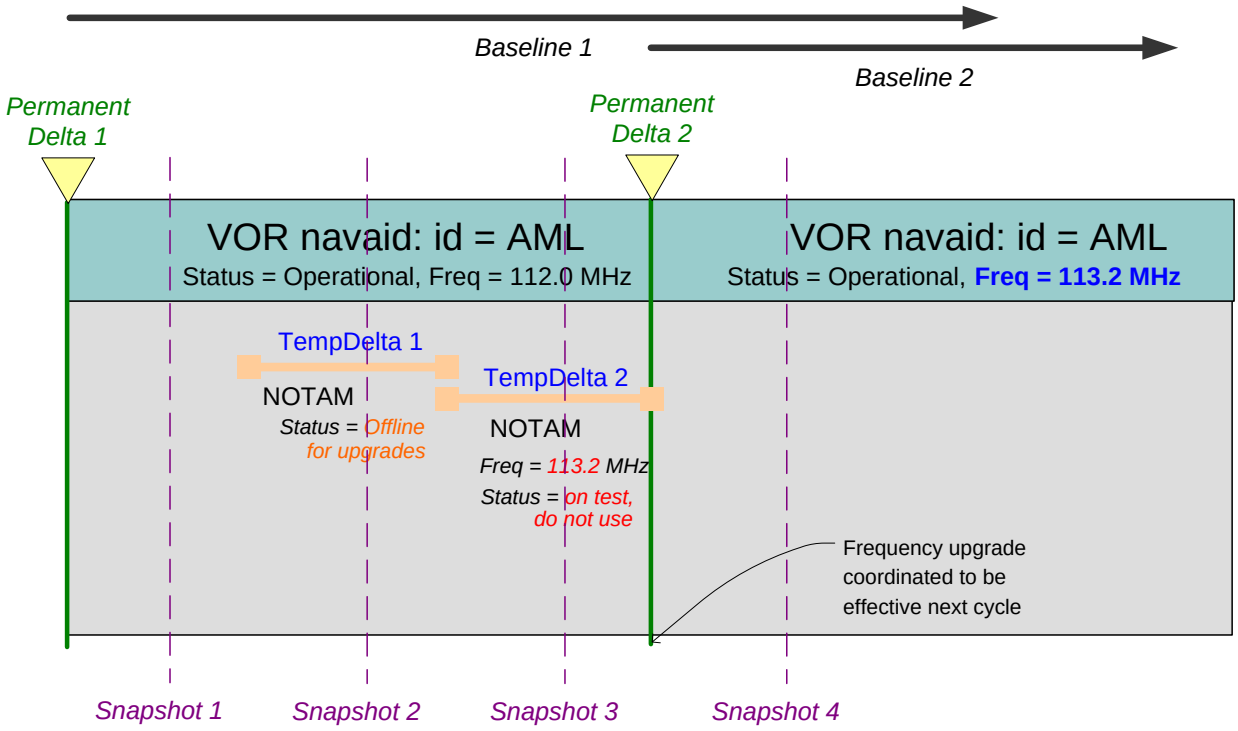


Figure 21

Based on this diagram we can identify the following temporal components:

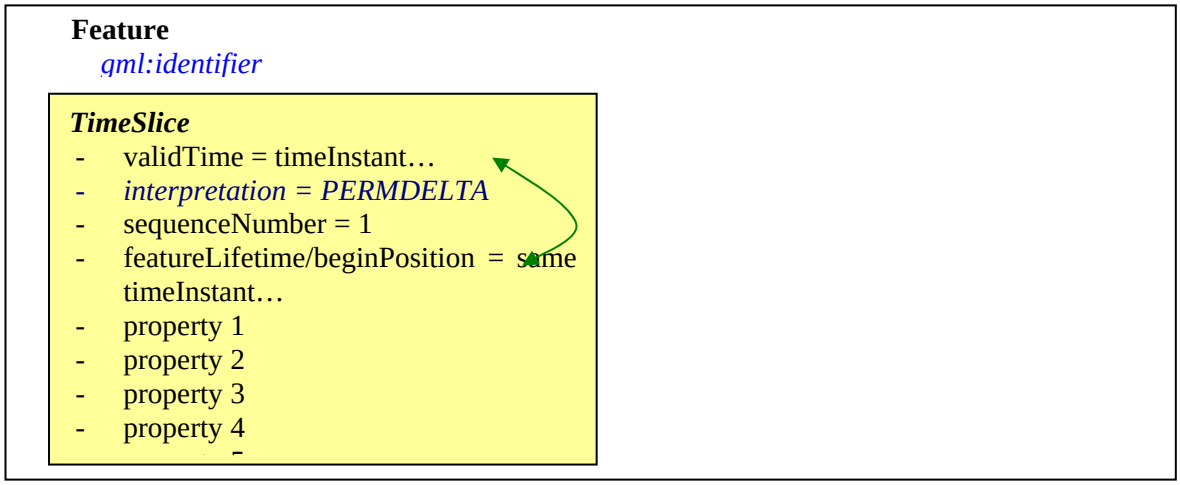
- The diagram shows two BASELINE Time Slices. The first baseline has a NAVAID frequency of 112.0 MHz and is valid since some time in the past; the second baseline has the new frequency of 113.2 MHz and is valid starting from the AIRAC cycle date.
- A PERMDELTA can be used to describe the permanent state change, which is the AML VOR frequency change. For completeness sake, the previous PERMDELTA that has preceded the first BASELINE (1) is also shown.
- Each transitory event can be expressed as a TEMPDELTA that changes the Operational Status of the navaid and eventually the frequency.
- Based on the PERMDELTA and the TEMPDELTA delta Time Slices shown in the diagram, four different versions for the “current status of the feature” may exist. Each “current status” version begins and ends at the boundary of a Permanent or Temporary Delta and may be presented as a Time Slice of type SNAPSHOT.

Depending on the temporal implementation employed by the exchanging systems, different methods can be used to communicate feature changes. In the interest of global standardization, the rest of this

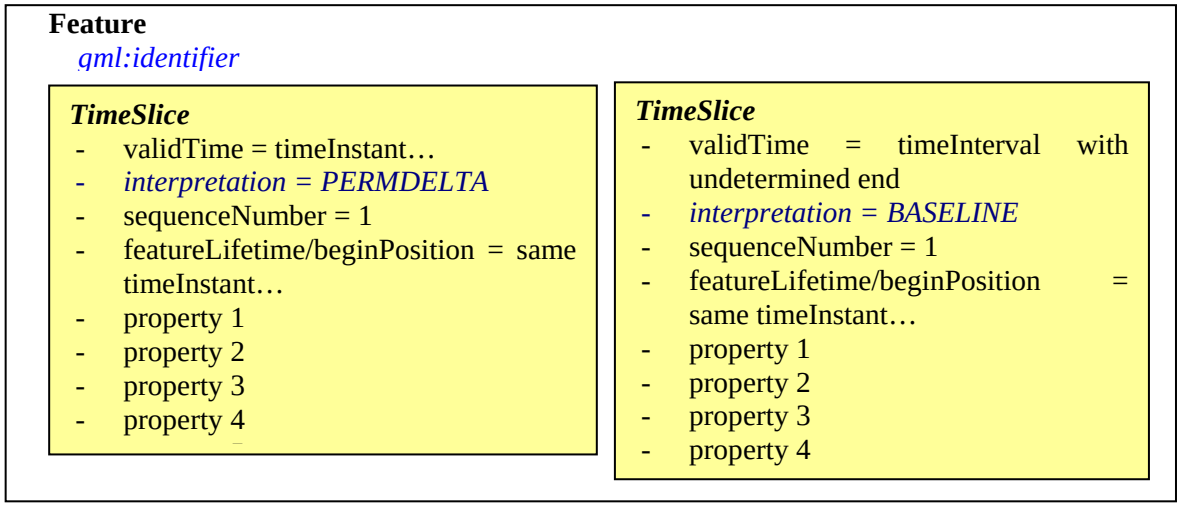
section provides some recommendations. They are relevant especially for "push" type applications, which generate and provide notifications in the form of TEMPDELTA and PERMDELTA Time Slices.

4.2 Feature creation (commissioning)

The start of life of a feature (also known as "commissioning") is modelled as a PERMDELTA which gives an initial value to the startOfLife property and to all other feature properties that are defined. The validTime of the PERMDELTA shall be the effective date and time when the feature is commissioned.



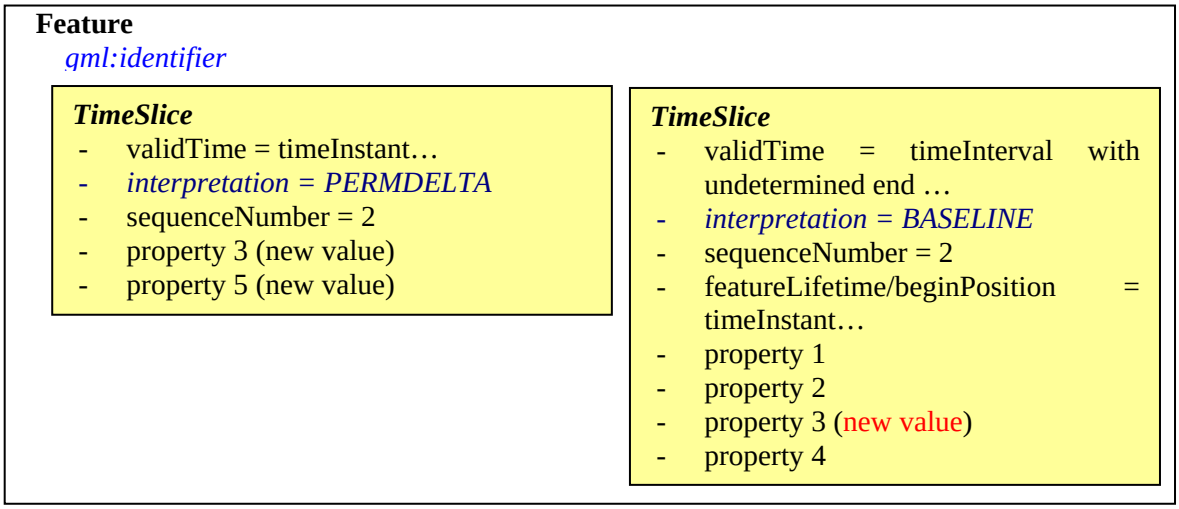
Optionally, if this has been requested by the user, a BASELINE Time Slice, containing the same property values as the PERMDELTA (as they are the result of the PERMDELTA) may also be included. The validTime of the BASELINE shall be a timeInterval with the end "undetermined".



4.3 Permanent change

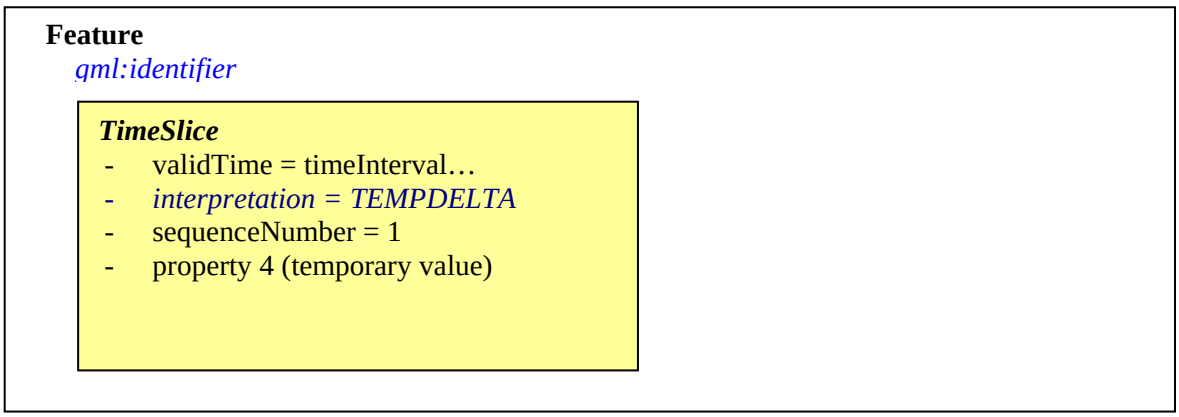
A permanent change is modeled as a PERMDELTA Time Slice, containing all properties that change their values. The validTime of the PERMDELTA shall be the effective date and time of the change.

Optionally, a BASELINE Time Slice can be included, containing all the properties that have a value as they result after the PERMDELTA. The validTime of the new BASELINE shall be a timeInterval with the end "undetermined".

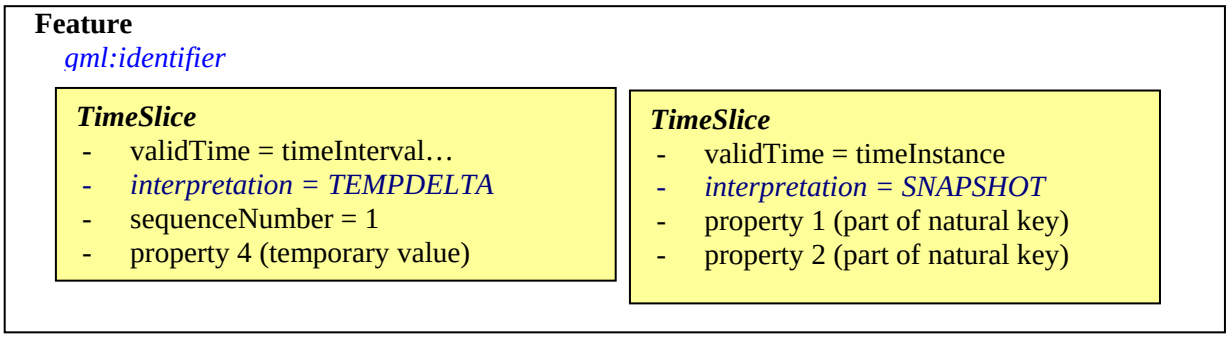


4.4 Digital NOTAM

A temporary state of a feature is encoded as a TEMPDELTA Time Slice, containing all properties that temporarily change their values. The validTime of the PERMDELTA shall indicate the start and the end of effectivity for the temporary state. The end can be undetermined.

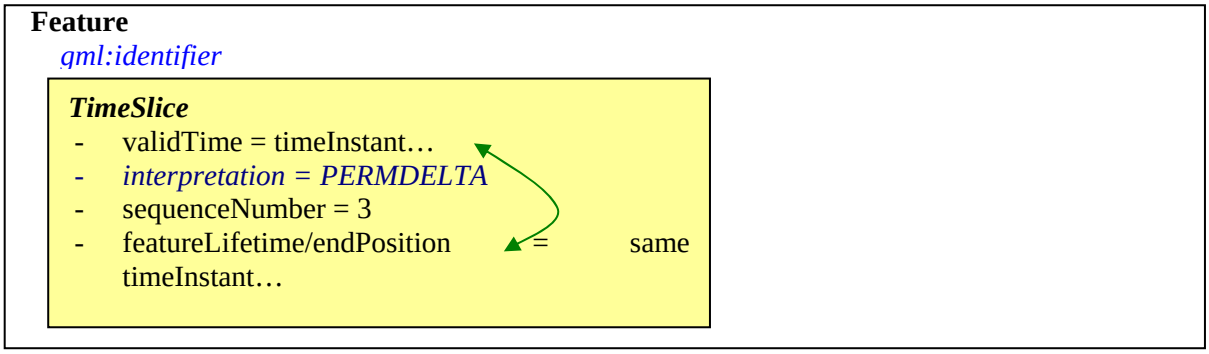


Optionally, a SNAPSHOT Time Slice can be included in the data set (used as "natural key").

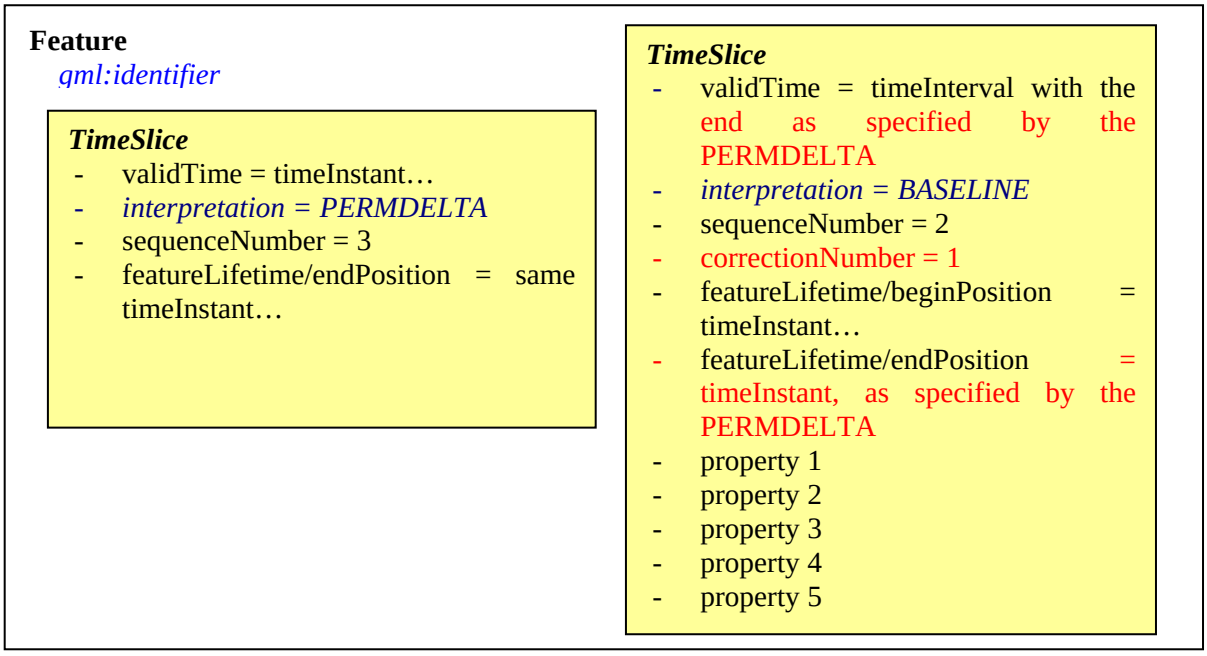


4.5 End of life (decommissioning)

The end of life of a feature (also known as "permanent withdrawn" or "decommissioning") is modelled as a PERMDELTA which gives a value to the featureLifetime/endPosition.



Optionally, the correction of the latest BASELINE can be included (if requested by the client).



4.6 Complete feature histories

The Timeslice model can be used to transmit history of a feature by transmitting the sequence of changes that occur to the feature's property. The feature history can be the past history or future history.

Figure 22 shows an example history of a fictional VOR navigation aid. The navigation aid has the following events:

- Jan 7, 2006: Commissioned
- Jan 23 – Feb 18, 2006: Temporary frequency change
- Feb 11 – Mar 9, 2006: Temporary offline
- Feb 22, 2006: Change in magnetic variation
- Mar 27, 2006: Change in frequency

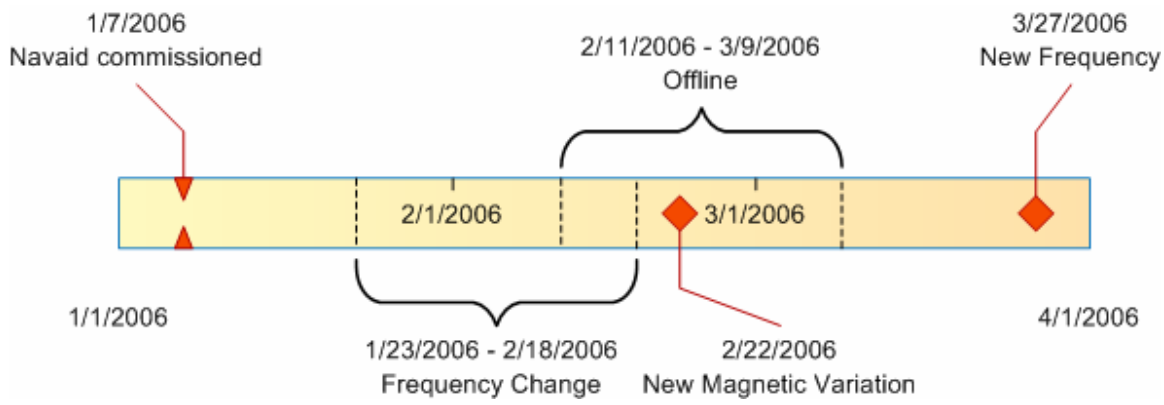


Figure 22: Fictitious example - history of a VOR navigation aid.

Using the Time Slice model we could represent the history of the VOR navigation aid as a series of five Time Slices, as shown in Figure 23. Three Time Slices are used to represent states and two are used to represent temporary events. Notice that overlapping events are encoded as separate Time Slices. The PERMDELTA Time Slices are not shown in this example.

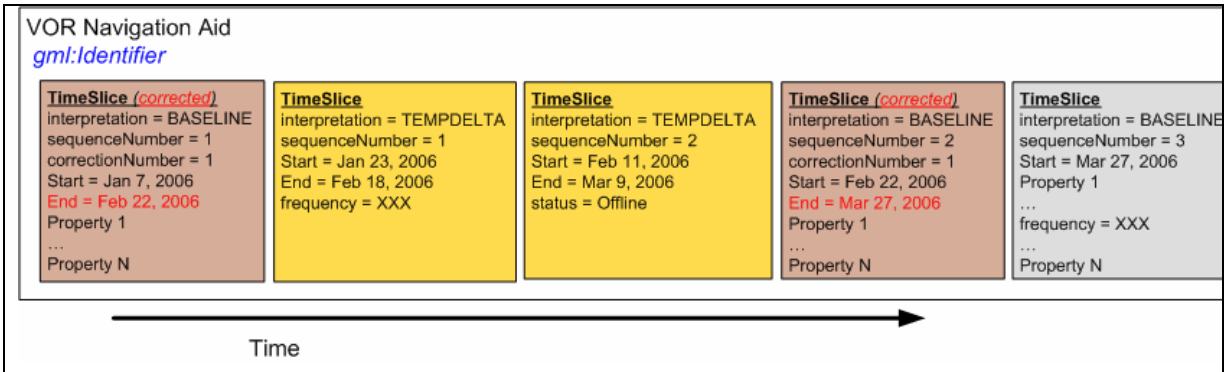


Figure 23: TimeSlices for the VOR navigation aid history

This approach to modeling history is equivalent to the recommended approach for GML 3.2 [4]. In actual implementations of the AIXM Timeslice mode, communicating histories can lead to very large messages. These large messages might be a problem for some resource constrained system. Although implementation issues are outside the scope of this design document we want to point out that the disadvantage of message size should be weighed against the value of standardization and compliance with GML. In most situations, the value of standardization may outweigh the loss of message efficiency.

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.