EUROCONTROL

NM 19.5.0 - NOP/B2B Reference Manuals - AirspaceServices

Edition No.	:	19.5.0.4.87	
Edition Issue Date	:	12/11/2015	
Author	:	NM NOP/B2B Team	
Reference	:	B2B/19.5.0/Airspace	
Copy No.	:		← stamp here

Document Control

Copyright Notice

Document Identification

Full Title:	NM 19.5.0 - NOP/B2B Reference Manuals - AirspaceServices
Total Number of Pages:	206

Table of Contents

List of Figures

List of Tables

References

External

[1] AIXM: Aeronautical Information Exchange Model (http://www.aixm.aero)

NM

[2] NOP/B2B Reference Manuals - CommonServices

[3] AIXM5.1 Temporality Model Profile for NM B2B (AIXMTemporalityModelProfileForNMB2B.pdf)

[4] ADR Extension Model (ADRReference.tar.gz)

Terminology

Main Abbreviations and Acronyms

AUP (Airspace Use Plan)

ADR (Airspace Data Repository)

UUP (Updated airspace Use Plan)

CADF (Central Airspace Data Function operators)

CDR (Conditional Route)

DMEAN (Dynamic Management of the European Airspace Network)

EAUP (European Airspace Use Plan)

EUUP (European Updated airspace Use Plan)

FIR (Flight Information Region)

RSA (Restricted Airspace)

UIR (Upper Information Region)

CIAM (Collaborative Interface for Airspace Management)

ASM (Airspace Management)

ADR-E (ADR Extension to AIXM 5.1)

Chapter 1. Introduction

1.1. Identification

(1) This document forms part of the set of the NM 19.5.0 - NOP/B2B Reference Manuals, which all together form the NM 19.5.0 - NOP/B2B Documentation.

(2) Its reference is B2B/19.5.0/Airspace.

(3) Its title is NM 19.5.0 - NOP/B2B Reference Manuals - AirspaceServices.

Chapter 2. Context

2.1. Introduction

(1) The AirspaceServices NOP/B2B service group is intended to provide services related to the management and sharing of Airspace data (e.g. airspaces, routes, aerodromes, etc.) as used by the NM systems.

(2) The AirspaceServices group consists of two types of services:

a) AirspaceStructureService: for retrieving up-to-date airspace data from the CACD database. The CACD database is the repository for the environment data (a.k.a. airspace data) used in the NM systems to perform Flight Planning and Flow Management. This data includes AIP concepts (such as Routes, Points and Aerodromes), and non-AIP concepts (such as Flows, RAD Restrictions and Traffic Volumes).
AIP concepts such as Airspaces may differ slightly from the AIP definition: for example when the AIP in defining an Airspace reads "follow the border between country X and Y", this must be translated into a real geometry that can be interpreted by the NM systems.

b) AirspaceAvailabilityService: for querying and modifying the airspace availability information; this includes the Flexible Use of Airspace (AUP/UUP and EAUP/EUUP).

(3) The Airspace services make use of AIXM 5.1/ADR-E types when possible (ADR-E stands for ADR Extension, see below). This does not mean that all data types defined in this service group are AIXM 5.1 or ADR-E types, as other service groups (Flight and Flow) use non-AIXM types, and because Airspace querying services must still use, for example, traditional ICAO ids (not UUIDs that do not support wildcards).

(4) Documentation about AIXM 5.1 can be found in [1].

(5) The ADR Extension is based on a UML model which is published on the Eurocontrol OneSky website (see [4]).

2.2. AIXM 5.1 / ADR-E

(1) This paragraph describes how the AIXM 5.1 model is used in terms of:

- Container message
- Temporality
- Feature/Object identification and referencing
- Subset of the AIXM 5.1 model and ADR-E used in the Airspace services

2.2.1. ADR Message

(1) The ADR Message is the container for exchanging AIXM 5.1 Features.

(2) An ADR Message may contain 0..n Features.

(3) An ADR Message is represented in the NM Exchange Model as typedef.

```
typedef<String> airspace:ADRMessageType;
```

(4) Depending on the service, the ADR Message may be embedded in the B2B reply as part of the message or returned as a file. When returned as a file, each file contains a single ADR Message. When embedded in the B2B reply, the reply may contain one or more ADR Messages.

2.2.2. Temporality

(1) The AIXM 5.1 Temporality model defines four TimeSlice types:

a) BASELINE

b) SNAPSHOT

c) PERMDELTA

d) TEMPDELTA

(2) The current NM services make use of all four timeslice types as follows:

- The AirspaceStructure services make use of BASELINE and PERMDELTA timeslices.
- The AirspaceAvailability services make use of SNAPSHOT and TEMPDELTA timeslices.

(3) The usage of these TimeSlice types conforms to the AIXM 5.1 recommendations:

- BASELINE TimeSlices are used to exchange the lifetime (or part of it) of the Airspace data (see AirspaceStructureService).
- PERMDELTA TimeSlices are used to exchange permanent changes to the Airspace data, i.e. changes that may or not follow the AIRAC cycles and are effective permanently (see AirspaceStructureService).
- TEMPDELTA TimeSlices are used to exchange temporary changes to the Airspace data, i.e. changes that do not follow the AIRAC cycles and are effective only for a short period of time (see AirspaceAvailabilityService).
- SNAPSHOT TimeSlices are used for Feature identification as an alternative to the UUID.

(4) The temporality model used in the AirspaceStructure services is explained in detail in [3].

2.2.3. Feature / Object identification

(1) NM follows the AIXM 5.1 recommendations for Feature and Object identification.

(2) AIXM 5.1 Features and Objects must have a valid `gml:id`.

(3) A `gml:id` is unique within the scope of an XML document. The `gml:id` is never persisted in the NM systems and cannot be used to identify a Feature or an Object outside of the XML document in which it is defined.

(4) In the airspace data published by NM, the use of the `gml:id` for Feature identification is limited to the AirspaceAvailability services only. In the AirspaceStructure services the Feature identification is done exclusively by means of the `gml:identifier`.

(5) In addition to a `gml:id`, AIXM 5.1 Features must have a valid `gml:identifier`.

(6) The `gml:identifier` is used as a persistent identifier for a Feature. A `gml:identifier` will always refer to the same Feature.

(7) The UUID is normally used as `gml:identifier` and in such cases the codeSpace `attribute` is set to "urn:uuid:". However for some Feature types a valid UUID could not be used. This is the case for those feature types that do not map to entities in the CACD database but are artificially created during the export to AIXM 5.1. Here is a list of such Feature types:

a) AirportHeliportCollocation -- The collocation of two airports.

b) AngleIndication -- The angle attribute of a *Reference Point* from a *Navaid*.

c) DistanceIndication -- The distance attribute of a *Reference Point* from a *Navaid*.

d) RouteSegment -- When it is part of a *NAT Track*.

e) AirTrafficManagementService -- The *Air Traffic Management Service* associated to the service provider defined by a *Unit*.

f) StandardLevelColumn

g) StandardLevelTable

(8) For example the AngleIndication and DistanceIndication in CACD are simple attributes of a Reference Point. They do not map to stand-alone entities. However when exporting the Reference Point to AIXM 5.1 the angle becomes a separate AngleIndication feature, which requires generating a new unique identifier. If afterwards the Reference Point is updated in CACD with a new angle value, when re-exporting to AIXM, it is important not to export a new AngleIndication without being able to stop the life of the previous one. Being able to identify a previously exported feature becomes paramount. Hence when a Feature does not exist as an entity in CACD, it cannot be assigned a new random UUID but it must be given a deterministic `gml:identifier` which allows referring to the same feature afterwards. Depending on the Feature type, different algorithms are used to generate such an identifier. These algorithms are explained below for each Feature when needed.

(9) In those cases in which a valid UUID cannot be used, the `codeSpace` value is set to "*urn:x-nmb2b:*".
Note that this is in line with the IANA reccomendations on URN Namespace Identifiers (NID). In particular RFC 2611 defines three categories of URN namespaces:

a) *Formal:*

i) It requires registration at IANA through RFC review process.

b) *Informal:*

i) It requires registration at IANA via template submission;

ii) It has the form *urn-<number>* (e.g. *urn-2)*, where *<number>* is assigned by IANA

c) *Experimental:*

i) It does not require any registration at IANA;

ii) There is no provision for avoiding namespace collisions;

iii) It has the form *x-<name>* (it starts with *x-*).

(10) The URN namespace used by NM falls into the *Experimental* category.

(11) All UUID values used in ADR Messages are originated and maintained by NM. In other words, NM applications serve and consume only NM UUIDs.

(12) The UUID is represented in the NM Data Model as:

```
typedef<string> common:UUID;
```

(13) In few cases in the AirspaceAvailabilityService the feature identification is done through a SNAPSHOT TimeSlice instead of the UUID. When this is the case it is explicitly documented (see AirspaceAvailabilityService).

2.2.4. Feature References

(1) In the ADR Message NM only makes use of Feature references and never Object references.

(2) Feature references in the ADR Message conform to the AIXM 5.1 recommendations.

(3) The NM general principle about Feature references is that they are always expressed as "remote references", i.e. via xlink:href to a gml:identifier (UUID), unless where explicitly stated otherwise (see AirspaceAvailabilityService).

(4) In few cases NM makes use of "local references" using xlink:href with an xpointer to a gml:id. When this is the case it is explicitly documented (see AirspaceAvailabilityService) as this represents a deviation from the NM general principle.

(5) When Feature references are used in the NM Data Model they are of type

```
typedef<string> common::UUID;
```

2.2.5. AIXM 5.1 and ADR Extension coverage

(1) This paragraph describes which subset of the AIXM 5.1 model and ADR-E is used by the Airspace services. For each exported feature and Object type, the list of exported attributes and associations is provided.

(2) Whenever a Feature/Object/Data type is not part of the AIXM 5.1 core but it has been introduced in the ADR-E, the heading of the corresponding section will have the name of the Feature/Object followed by "(ADR-E)" indicating that AIXM 5.1 is extended.
Examples: "IntermediateSignificantPoint Object (ADR-E)", "CodeConditionalRouteType (ADR-E)".

(3) Whenever a Feature/Object that is already part of the AIXM 5.1 core is extended, the heading of the corresponding section will not have the name of the Feature/Object followed by "(ADR-E)". Instead the new attributes and associatons mentioned in the section will be followed by "(ADR-E) to highlight that those attributes exist only in the ADR-E".
Example: The Airspace Feature section will be called "Airspace Feature" but the attribute "level1" will be documented as "level1 (ADR-E)" indicating that the attribute was added to the existing Airspace Feature and can be found in the ADR-E.

(4) The following concepts are explained:

a) Features

b) Objects

c) Data Types

d) Miscellania

2.2.5.1. Features

2.2.5.1.1. AirportHeliport Feature

(1) The exported attributes are:

a) name

b) locationIndicatorICAO

c) designatorIATA

d) controlType

(2) The exported associations are:

a) servedCity

b) ARP

(3) Example:

```
<aixm:AirportHeliportTimeSlice gml:id="ID_167_1385510754492_3483">
  <aixm:name>BRUSSELS/BRUSSELS-NATIONAL</aixm:name>
  <aixm:locationIndicatorICAO>EBBR</aixm:locationIndicatorICAO>
  <aixm:designatorIATA>BRU</aixm:designatorIATA>
  <aixm:controlType>CIVIL</aixm:controlType>
  <aixm:servedCity>
    <aixm:City gml:id="ID_167_1385510754492_3484">
      <aixm:name>BRUSSELS</aixm:name>
    </aixm:City>
  </aixm:servedCity>
  <aixm:ARP>
    <aixm:ElevatedPoint gml:id="ID_167_1385510754492_3485">
      <gml:pos srsName="urn:ogc:def:crs:EPSG::4326">50.90138888888889 4.484444444444445</gml:pos>
    </aixm:ElevatedPoint>
  </aixm:ARP>
</aixm:AirportHeliportTimeSlice>
```

2.2.5.1.2. AirportHeliportCollocation Feature

(1) This is an artificial Feature created during the AIXM 5.1 export that does not exist as a stand-alone entity in CACD.

(2) Feature identification (see Feature/Object identification): The `gml:identifier` of an *AirportHeliportCollocation* is the concatenation of the feature type and the UUID of the dependent *AirportHeliport*. This is sufficient because in CACD a dependent Aerodrome (called a child) can have only one host aerodrome (called the parent).
Example:

```
AirportHeliportCollocation_c608da02-859e-4c93-a228-73da81d686c9
```

(3) There are no exported attributes.

(4) The exported associations are:

a) hostAirport

b) dependentAirport

(5) Example:

```
<aixm:AirportHeliportCollocationTimeSlice gml:id="ID_167_1385510754492_3484">
  <aixm:hostAirport xlink:href="urn:uuid:35b44a15-2cb5-455d-98e0-1f2cc09b3160"/>
  <aixm:dependentAirport xlink:href="urn:uuid:2fc069c4-3a18-46f2-9ea8-a77c96701fc9"/>
</aixm:AirportHeliportCollocationTimeSlice>
```

2.2.5.1.3. AirportHeliportSet Feature (ADR-E)

It represents a set of aerodromes, which may be listed explicitly or defined via a pattern id.

(1) The exported attributes are:

a) airportHeliport

To explicitly add aerodromes to an AirportHeliportSet

b) airportHeliportSetPattern
To implicitly add groups of aerodromes to an AirportHeliportSet based on a pattern in the designator. The value is a string of alfabetic characters and represents the first letters of the ICAO identifier. The semantic is therefore the following: "include all aerodromes whose ICAO identifier starts with the pattern". For example a pattern such as "EB" includes all aerodromes whose ICAO designator starts with 'EB'.

(2) The exported associations are:

a) airportHeliport

(3) Example showing an AirportHeliportSet composed of 4 explicit aerodromes plus all Aerodromes whose ICAO identifier starts with "EGA":

```
<AirportHeliportSetTimeSlice id="ID_690_1423477720241_130">
  <airportHeliport href="urn:uuid:b6f0ff0c-397c-4140-bf25-c47830a3ddc0"/>
  <airportHeliport href="urn:uuid:ed6ffa29-2e87-439e-a899-d942791ac9cd"/>
  <airportHeliport href="urn:uuid:c3c8b035-26fd-4c9d-a5c9-7dd177d14c08"/>
  <airportHeliport href="urn:uuid:b4274096-7463-40ee-a6b9-008c4fa19834"/>
  <airportHeliportSetPattern>
    <AirportHeliportSetPattern id="ID_690_1423477720241_133">
      <pattern>EGA</pattern>
    </AirportHeliportSetPattern>
  </airportHeliportSetPattern>
</AirportHeliportSetTimeSlice>
```

2.2.5.1.4. Airspace Feature

(1) The exported attributes are:

a) type

b) designator

c) localType

d) designatorICAO

e) flexibleUse (ADR-E)

f) level1 (ADR-E): the airspace is manageable at the strategic level. The act of defining and reviewing as required the national airspace policy taking into account national and international airspace requirement.

g) level2 (ADR-E): the airspace is manageable at the pre-tactical level. The act of conducting operational management within the framework of pre-determined existing ATM structure and procedures defined in level1 and of reaching specific agreement between civil and military authorities involved.

h) level3 (ADR-E): the airspace is manageable at the tactical level.

i) isFBZ (ADR-E): this attribute is only exported when the Airspace.type is one of ('D', 'R', 'P', 'TSA', 'TRA', 'RCA', 'CBA') or Airspace.type is 'D_OTHER' and Airspace.localType is one of ('MRA', 'MTA'). These types correspond to the CACD RestrictedAirspace(RSA). When the attribute isFBZ=='YES', then the Airspace is an FPL Buffer Zone.

j) fbzDefaultActive (ADR-E): this attribute is only exported when Airspace.isFBZ=='YES'. The attribute fbzDefaultActive is used in the context of FUA.

   i) AMA: AMC Manageable Area. This corresponds to Airspace.level1=="YES" and level2=="YES". The airspace can be activated in a flexible way for use by the military or other special users after due coordination between military and civilian airspaces during the times defined in the availability. When there is no allocation for this airspace in the AUP/UUP, then the airspace is considered as available for civilian traffic during the availability.

   ii) NAM: Non AMC Manageable Area. This corresponds to Airspace.level1=="YES" and level2=="NO". The airspace can be activated by the military or other special users without prior coordination with the civilian users, i.e. AMC during the times defined in the availability. When there is no allocation for this airspace in the AUP/UUP, then the airspace is considered as closed for civilian traffic during the availability.

   When an airspace managed by an AMC is not allocated, then there is an implicit 'allocation' according to whether the airspace is AMA/NAM. The question arises as whether the RSA or the surrounding FBZ should be used for that. In order to answer that question, the FBZ airspace has an attribute fbzDefaultActive. When fbzDefaultActive=='YES', the FBZ availability will be used for the implicit allocation, otherwise the RSA availability will be used.

(2) The exported associations are:

a) geometryComponent

b) activation: refers only to AirspaceActivations with status=AVBL_FOR_ACTIVATION. The related Timesheet (PropertiesWithSchedule) contains a Time Schedule (see Timesheet Time Schedule).

c) nearby (ADR-E): refers to RoutePortions potentially extended with a range(AirspaceLayer Object). When the RSA Airspace is allocated, the nearby RoutePortions are considered to be so near that need to be closed.

d) offload (ADR-E): refers to RoutePortions potentially extended with a range(AirspaceLayer Object). When the RSA Airspace is allocated, the offload RoutePortions are considered to be an alternative so they are opened.

e) notAffected (ADR-E): refers to RoutePortions. When the RSA Airspace is allocated, these RoutePortions are considered as not affected (neither opened nor closed).

f) rsaActivation (ADR-E): refers to AirspaceActivations with status=ACTIVE. In reality they are the result of the publication of an AUP/UUP. The related Timesheet (PropertiesWithSchedule) contains a Time Period (see Timesheet Time Period).

g) ownerRSA (ADR-E): only set when Airspace.isFBZ=='YES'. The owning RSA in indicated with a uuid.

(3) There are basically three levels of composition:

a) Airblocks

b) Airspaces composed of Airblocks associated with AltitudeRange to give a 3D volume

c) Airspaces composed of other Airspaces

2.2.5.1.4.1. Airblocks

(1) An Airblock describes a horizontal projection. The type of this Airspace is "PART".

(2) An Airblock is 'anonymous' in the sense that it has no designator.

(3) An Airblock is a building block for 3D Airspaces.

(4) The horizontal projection is coded as a flat AIXM AirspaceVolume. None of the attributes of the AirspaceGeometryComponent are set. The geometrical description is defined by the associated Surface object (GML /ISO19107).

(5) The Surface uses srsName="urn:ogc:def:crs:EPSG::4326".

(6) A GM_Surface is composed of an array of Patch objects. The Patch object that is used is PolygonPatch.

(7) The PolygonPatch describes the exterior with a LinearRing.

(8) The vertices are:

a) Point Features which reference a UUID

b) Positions which reference coordinates

(9) Example:

```
<aixm:AirspaceTimeSlice gml:id="ID_171_1385510754499_5">
  <aixm:type>PART</aixm:type>
  <aixm:geometryComponent>
    <aixm:AirspaceGeometryComponent gml:id="ID_171_1385510754499_6">
      <aixm:theAirspaceVolume>
        <aixm:AirspaceVolume gml:id="ID_171_1385510754499_7">
          <aixm:horizontalProjection>
            <aixm:Surface gml:id="ID_171_1385510754499_8" srsName="urn:ogc:def:crs:EPSG::4326">
              <gml:patches>
                <gml:PolygonPatch>
                  <gml:exterior>
                    <gml:LinearRing>
                      <gml:pointProperty xlink:href="urn:uuid:da598262-..."/>
                      <gml:pointProperty xlink:href="urn:uuid:bbc5c700-..."/>
                      <gml:pos>56.66833333333334 -10.0</gml:pos>
                    </gml:LinearRing>
                  </gml:exterior>
                </gml:PolygonPatch>
```

```
            </gml:patches>
          </aixm:Surface>
        </aixm:horizontalProjection>
      </aixm:AirspaceVolume>
    </aixm:theAirspaceVolume>
  </aixm:AirspaceGeometryComponent>
</aixm:geometryComponent>
</aixm:AirspaceTimeSlice>
```

2.2.5.1.4.2. Airspaces composed of Airblocks

(1) An Airspace composed of Airblocks and AltitudeRange is mapped to an AIXM Airspace having AirspaceVolumes.

(2) There can be multiple associated AirspaceVolumes.

(3) The AirspaceVolume defines the altitude range through the upperLimit and lowerLimit attributes.

(4) For the first (in order of appearance) AirspaceVolume, the corresponding AirspaceGeometry-Component has its "operation" attribute set to "BASE".

(5) For subsequent (in order of appearance) AirspaceVolumes, the corresponding AirspaceGeometry-Component has its "operation" attribute set to "UNION".

(6) As the AirspaceVolume depends on an Airblock, the AirspaceVolumeDependency has its "dependency" attribute set to "HORZ_PROJECTION".

(7) Example:

```
<aixm:AirspaceTimeSlice gml:id="ID_171_1385510754499_41296">
  <aixm:type>SECTOR</aixm:type>
  <aixm:designator>BIRDES</aixm:designator>
  <aixm:designatorICAO>YES</aixm:designatorICAO>
  <aixm:geometryComponent>
    <aixm:AirspaceGeometryComponent gml:id="ID_171_1385510754499_41297">
      <aixm:operation>BASE</aixm:operation>
      <aixm:theAirspaceVolume>
        <aixm:AirspaceVolume gml:id="ID_171_1385510754499_41298">
          <aixm:upperLimit uom="FT">UNL</aixm:upperLimit>
          <aixm:upperLimitReference>MSL</aixm:upperLimitReference>
          <aixm:lowerLimit uom="FL">55</aixm:lowerLimit>
          <aixm:lowerLimitReference>STD</aixm:lowerLimitReference>
          <aixm:contributorAirspace>
            <aixm:AirspaceVolumeDependency gml:id="ID_171_1385510754499_41299">
              <aixm:dependency>HORZ_PROJECTION</aixm:dependency>
              <aixm:theAirspace xlink:href="urn:uuid:a2cf60ce-8fe9-4ee1-913f-06cc0a9bdb84"/>
            </aixm:AirspaceVolumeDependency>
          </aixm:contributorAirspace>
        </aixm:AirspaceVolume>
      </aixm:theAirspaceVolume>
    </aixm:AirspaceGeometryComponent>
  </aixm:geometryComponent>
  <aixm:geometryComponent>
    <aixm:AirspaceGeometryComponent gml:id="ID_171_1385510754499_41300">
      <aixm:operation>UNION</aixm:operation>
      <aixm:theAirspaceVolume>
        <aixm:AirspaceVolume gml:id="ID_171_1385510754499_41301">
          <aixm:upperLimit uom="FT">UNL</aixm:upperLimit>
          <aixm:upperLimitReference>MSL</aixm:upperLimitReference>
```

```
          <aixm:lowerLimit uom="FL">55</aixm:lowerLimit>
          <aixm:lowerLimitReference>STD</aixm:lowerLimitReference>
          <aixm:contributorAirspace>
            <aixm:AirspaceVolumeDependency gml:id="ID_171_1385510754499_41302">
              <aixm:dependency>HORZ_PROJECTION</aixm:dependency>
              <aixm:theAirspace xlink:href="urn:uuid:418f6ebc-1bfe-41f0-ac9b-d70f584f3375"/>
            </aixm:AirspaceVolumeDependency>
          </aixm:contributorAirspace>
        </aixm:AirspaceVolume>
      </aixm:theAirspaceVolume>
    </aixm:AirspaceGeometryComponent>
  </aixm:geometryComponent>
</aixm:AirspaceTimeSlice>
```

2.2.5.1.4.3. Airspaces composed of other Airspaces

(1) An Airspace composed of other Airspaces defines its volume by AirspaceVolumes.

(2) There can be multiple associated AirspaceVolumes.

(3) For the first AirspaceVolume, the corresponding AirspaceGeometryComponent has its "operation" attribute set to "BASE".

(4) For subsequent AirspaceVolumes, the corresponding AirspaceGeometryComponent has its "operation" attribute set to "UNION".

(5) As the AirspaceVolume depends on an Airspace, the AirspaceVolumeDependency has its "dependency" attribute set to "FULL_GEOMETRY".

(6) Example:

```
<aixm:AirspaceTimeSlice gml:id="ID_171_1385510754499_198517">
  <aixm:type>NAS</aixm:type>
  <aixm:designator>EB</aixm:designator>
  <aixm:designatorICAO>YES</aixm:designatorICAO>
  <aixm:geometryComponent>
    <aixm:AirspaceGeometryComponent gml:id="ID_171_1385510754499_198518">
      <aixm:operation>BASE</aixm:operation>
      <aixm:theAirspaceVolume>
        <aixm:AirspaceVolume gml:id="ID_171_1385510754499_198519">
          <aixm:contributorAirspace>
            <aixm:AirspaceVolumeDependency gml:id="ID_171_1385510754499_198520">
              <aixm:dependency>FULL_GEOMETRY</aixm:dependency>
              <aixm:theAirspace xlink:href="urn:uuid:9be9ab99-3df5-4251-9cb9-fba72afeb751"/>
            </aixm:AirspaceVolumeDependency>
          </aixm:contributorAirspace>
        </aixm:AirspaceVolume>
      </aixm:theAirspaceVolume>
    </aixm:AirspaceGeometryComponent>
  </aixm:geometryComponent>
  <aixm:geometryComponent>
    <aixm:AirspaceGeometryComponent gml:id="ID_171_1385510754499_198521">
      <aixm:operation>UNION</aixm:operation>
      <aixm:theAirspaceVolume>
        <aixm:AirspaceVolume gml:id="ID_171_1385510754499_198522">
          <aixm:contributorAirspace>
            <aixm:AirspaceVolumeDependency gml:id="ID_171_1385510754499_198523">
              <aixm:dependency>FULL_GEOMETRY</aixm:dependency>
              <aixm:theAirspace xlink:href="urn:uuid:265ffc98-d66f-454a-811b-3b2f66d490e6"/>
            </aixm:AirspaceVolumeDependency>
```

```
          </aixm:contributorAirspace>
        </aixm:AirspaceVolume>
      </aixm:theAirspaceVolume>
    </aixm:AirspaceGeometryComponent>
  </aixm:geometryComponent>
</aixm:AirspaceTimeSlice>
```

2.2.5.1.5. AirTrafficManagementService Feature

(1) This is an artificial Feature created during the AIXM 5.1 export that does not exist as a stand-alone entity in CACD.

(2) Feature identification (see Feature/Object identification): The `gml:identifier` for *AirtrafficManagementService* is composed of the feature type and the UUID of the *Unit*.
Example:

```
AirTrafficManagementService_4cfcafb8-1841-405c-9c75-454dafd8e5d4
```

(3) None of the attributes are exported.

(4) The exported associations are:

a) serviceProvider: this is always a reference to a Unit Feature of type OTHER:__ADR__AMC.

b) clientAirspace

(5) Example:

```
<aixm:AirtrafficManagementServiceTimeSlice gml:id="ID_171_1385510754499_198518">
  <aixm:serviceProvider xlink:href="urn:uuid:4cfcafb8-1841-405c-9c75-454dafd8e5d4"/>
  <aixm:clientAirspace xlink:href="urn:uuid:27b59518-f53c-4ccf-9c38-0495935946c9"/>
  <aixm:clientAirspace xlink:href="urn:uuid:11f90918-73dd-450a-832a-ca5c2b0d061d"/>
</aixm:AirtrafficManagementServiceTimeSlice>
```

2.2.5.1.6. AngleIndication Feature

(1) This is an artificial Feature created during the AIXM 5.1 export that does not exist as a stand-alone entity in CACD.

(2) Feature identification (see Feature/Object identification): The `gml:identifier` for *AngleIndication* contains the feature type and the UUID of the *Reference Point*. It is not necessary to include the UUID of the *Navaid* because the reference point can refer to only one Navaid at a time.
Example:

```
AngleIndication_529f213e-0568-4334-86c0-8bb1a268b9dc
```

(3) The exported attributes are:

a) angle

b) angleType

(4) The exported associations are:

a) fix
This association refers to a DesignatedPoint of type OTHER:__ADR__REFERENCE.

b) pointChoice
The pointChoice always refers to a Navaid.

(5) Example:

```
<aixm:AngleIndicationTimeSlice gml:id="ID_171_1385510754499_198518">>
  <aixm:angle>10</aixm:angle>
  <aixm:angleType>MAG</aixm:angleType>
  <aixm:fix xlink:href="urn:uuid:c608da02-859e-4c93-a228-73da81d686c9"/>
  <aixm:pointChoice_navaidSystem xlink:href="urn:uuid:529f213e-0568-4334-86c0-8bb1a268b9dc"/>
</aixm:AngleIndicationTimeSlice>
```

2.2.5.1.7. ArrivalLeg Feature

(1) The exported attributes are:

a) upperLimitAltitude and upperLimitReference

b) lowerLimitAltitude and lowerLimitReference

(2) The exported associations are:

a) startPoint

b) endPoint

c) arrival

(3) Example:

```
<aixm:ArrivalLegTimeSlice gml:id="ID_172_1385510754499_780064">
  <aixm:upperLimitAltitude uom="FL">20</aixm:upperLimitAltitude>
  <aixm:upperLimitReference>STD</aixm:upperLimitReference>
  <aixm:lowerLimitAltitude uom="FT">GND</aixm:lowerLimitAltitude>
  <aixm:lowerLimitReference>MSL</aixm:lowerLimitReference>
  <aixm:startPoint>
    <aixm:TerminalSegmentPoint gml:id="ID_172_1385510754499_780064">
      <aixm:pointChoice_navaidSystem xlink:href="urn:uuid:69ed4c7b-d34c-457c-a780-3baed58fe767"/>
    </aixm:TerminalSegmentPoint>
  </aixm:startPoint>
  <aixm:endPoint>
    <aixm:TerminalSegmentPoint gml:id="ID_172_1385510754499_780065">
      <aixm:pointChoice_airportReferencePoint xlink:href="urn:uuid:dd2a6f3f-bd9b-436e-98..."/>
    </aixm:TerminalSegmentPoint>
  </aixm:endPoint>
  <aixm:arrival xlink:href="urn:uuid:28b8122d-ca51-4cbb-aa5f-b78d859099c9"/>
</aixm:ArrivalLegTimeSlice>
```

2.2.5.1.8. DepartureLeg Feature

(1) The exported attributes are:

a) upperLimitAltitude and upperLimitReference

b) lowerLimitAltitude and lowerLimitReference

(2) The exported associations are:

a) startPoint

b) endPoint

c) departure

(3) Example:

```
<aixm:DepartureLegTimeSlice gml:id="ID_172_1385510754499_780098">
  <aixm:upperLimitAltitude uom="FL">110</aixm:upperLimitAltitude>
  <aixm:upperLimitReference>STD</aixm:upperLimitReference>
  <aixm:lowerLimitAltitude uom="FT">GND</aixm:lowerLimitAltitude>
  <aixm:lowerLimitReference>MSL</aixm:lowerLimitReference>
  <aixm:startPoint>
    <aixm:TerminalSegmentPoint gml:id="ID_172_1385510754499_780099">
      <aixm:pointChoice_fixDesignatedPoint xlink:href="urn:uuid:28ac6496-46bf-4c13-8293-..."/>
    </aixm:TerminalSegmentPoint>
  </aixm:startPoint>
  <aixm:endPoint>
    <aixm:TerminalSegmentPoint gml:id="ID_172_1385510754499_780100">
      <aixm:pointChoice_navaidSystem xlink:href="urn:uuid:aca80964-9e4f-4e59-970b-..."/>
    </aixm:TerminalSegmentPoint>
  </aixm:endPoint>
  <aixm:departure xlink:href="urn:uuid:056b539f-40d5-455b-a3a3-41baf7ceb71d"/>
</aixm:DepartureLegTimeSlice>
```

2.2.5.1.9. DesignatedPoint Feature

(1) The exported attributes are:

a) designator
The designator is only defined for the types:

i) ICAO

ii) TERMINAL

iii) OTHER:__ADR__REFERENCE

b) type

c) name

(2) The exported association is:

a) location

(3) In AIXM 5.1 the possible types of a DesignatedPoint are listed in the core enumeration data type `CodeDesignatedPointType`. NM makes use of three additional types, which are listed below. These additional types follow the AIXM 5.1 convention for extending an enumeration, i.e. by using the syntax "OTHER:...".

a) OTHER:__ADR__REFERENCE
This type of point is a derived point. They support the definitions of Routes. The corresponding AngleIndication and DistanceIndication refer to it as their "fix".

b) OTHER:__ADR__BOUNDARY
This type of point lays on a Route at the boundary of two information regions. This type is not yet defined by ICAO. EAD exports these points with the type COORD. EAD uses the SignificantPointInAirspace Feature to link the DesignatedPoint and the Airpace together with a relativeLocation attribute set to "BORDER".

c) OTHER:__ADR__RADAR
This type of point represents a radar located at an Aerodrome. No such points are currently exported.

2.2.5.1.10. DistanceIndication Feature

(1) This is an artificial Feature created during the AIXM 5.1 export that does not exist as a standalone entity in CACD.

(2) Feature identification (see Feature/Object identification): The `gml:identifier` for *DistanceIndication* contains the feature type and the UUID of the *Reference Point*. It is not necessary to include the UUID of the *Navaid* because the reference point can refer to only one Navaid at a time. Example:

```
DistanceIndication_529f213e-0568-4334-86c0-8bb1a268b9dc
```

(3) The exported attribute is:

a) distance

(4) The exported associations are:

a) fix
This association refers to a DesignatedPoint of type OTHER:__ADR__REFERENCE.

b) pointChoice
The pointChoice always refers to a Navaid.

(5) Example:

```
<aixm:DistanceIndicationTimeSlice gml:id="ID_172_1385510754499_780099">
  <aixm:distance uom="NM">14</aixm:distance>
```

```
  <aixm:fix xlink:href="urn:uuid:c608da02-859e-4c93-a228-73da81d686c9"/>
  <aixm:pointChoice_navaidSystem xlink:href="urn:uuid:529f213e-0568-4334-86c0-8bb1a268b9dc"/>
</aixm:DistanceIndicationTimeSlice>
```

2.2.5.1.11. FlightRestriction Feature

(1) The exported attributes are:

a) designator

b) type

c) instruction: this corresponds to the <txtDescr> in AIXM 4.5

d) processingIndicator (ADR-E)

e) enabled (ADR-E)

f) usage (ADR-E): indicates whether the TimeSlice is operational or not.

g) isFUA (ADR-E): indicates whether the FlightRestriction is a FUARestriction or not.

h) fuaDefaultActive (ADR-E): indicates that this FlightRestriction should be activated by default when its dependent RSA airspace is allocated by an AUP/UUP. The creation/update of an AUP/UUP must indicates which FlightRestrictions are activated or not.

(2) The exported associations are:

a) flight

b) regulatedRoute

c) annotation: this corresponds to the <txtOprGoal> in AIXM 4.5. It is exported as a Note with propertyName="instruction", purpose="REMARK" (see example)

(3) Example:

```
<aixm:FlightRestrictionTimeSlice gml:id="ID_168_1385510754493_5">
  <aixm:designator>DS2000A</aixm:designator>
  <aixm:type>FORBID</aixm:type>
  <aixm:instruction>GITER NOT AVAILABLE $FOR TRAFFIC ARR ESMS</aixm:instruction>
  <aixm:flight>
    <aixm:FlightConditionCombination gml:id="ID_168_1385510754493_6">
      <!-- not expanded here -->
    </aixm:FlightConditionCombination>
  </aixm:flight>
  <aixm:regulatedRoute>
    <!-- not expanded here -->
  </aixm:regulatedRoute>
  <aixm:annotation>
    <aixm:Note gml:id="ID_4941_1381916120390_9">
      <aixm:propertyName>instruction</aixm:propertyName>
      <aixm:purpose>REMARK</aixm:purpose>
      <aixm:translatedNote>
        <aixm:LinguisticNote gml:id="ID_4941_1381916120390_10">
```

```
          <aixm:note>TO SEGREGATE ARR ESMS TO DEP EKCH</aixm:note>
        </aixm:LinguisticNote>
      </aixm:translatedNote>
    </aixm:Note>
  </aixm:annotation>
</aixm:FlightRestrictionTimeSlice>
```

2.2.5.1.12. Flow Feature (ADR-E)

(1) A Flow identifies a pattern of traffic and it is used inside a ReferenceLocation or a TrafficVolume (see below).

(2) It catches flights by defining where they come from, where they are directed and what they cross. This is done by combining Flow Elements.

(3) A Flow Element is in fact a location, which can be one of the following:

- An AirportHeliport
- An AirportHeliportSet
- An Airspace
- A Significant Point (i.e. either a Navaid or a DesignatedPoint)

This location is not to be confused with the Reference Location. A Flow Element is a location used to define how the Flow is composed. For example a Flow may be defined as a sequence of Flow Elements such as the following:

a) Flights crossing point DIRBO

b) And then landing in EDDV

The order of the FlowElements is paramount.

(4) More precisely a Flow is defined by sequences of upstream and downstream Flow Elements, where the concepts of upstream and downstream are to be intended with respect to the Reference Location to which the Flow will be linked. This allows the same Flow to be linked to several Reference Locations or Traffic Volumes without having to redefine the Flow Elements in each Reference Location/Traffic Volume.

(5) For example, the following snippet defines a Flow named "EBBR>" (note that the character '>' is escaped as '>' in order to be included in an XML document) with a single downstream Flow Element represented by the EBBR airport (The UUID f44fb7b2-a883-4e4d-b741-97b2d1879ae5 is the 'EBBR' aerodrome). The sematic of this Flow is that it captures all flights departing from EBBR.

```
<Flow id="ID_74_1425258197437_15290">
  <identifier codeSpace="urn:uuid:">0ab1205a-0692-4798-8068-40c15c7e1e6f</identifier>
  <timeSlice>
    <FlowTimeSlice id="ID_74_1425258197437_15291">
      <validTime>
        <TimePeriod id="ID_74_1425258197437_15292">
          <beginPosition>2003-12-25T00:00:00</beginPosition>
          <endPosition indeterminatePosition="unknown"/>
```

```
        </TimePeriod>
      </validTime>
      <interpretation>BASELINE</interpretation>
      <featureLifetime>
        <TimePeriod id="ID_74_1425258197437_15293">
          <beginPosition>2003-12-25T00:00:00</beginPosition>
          <endPosition indeterminatePosition="unknown"/>
        </TimePeriod>
      </featureLifetime>
      <flowId>EBBR&gt;</flowId>
      <upstreamFlowElement>
        <FlowLocationElement id="ID_74_1425258197437_15294">
          <index>1</index>
          <locationChoice_airportHeliport href="urn:uuid:f44fb7b2-a883-4e4d-b741-97b2d1879ae5"/>
        </FlowLocationElement>
      </upstreamFlowElement>
    </FlowTimeSlice>
  </timeSlice>
</Flow>
```

(6) The Flow in the above example can now be used in any ReferenceLocation or Traffic Volume. For example it is used in the ReferenceLocation 'LFRRZU' (which represents a particular sector) to capture flights traversing the secrtor LFRRZU which departed from the 'EBBR' airport.

(7) The exported attributes are:

a) flowId

(8) The exported associations are:

a) downstreamFlowElement

b) upstreamFlowElement

(9) Example showing a Flow with one downstream FlowElement and two upstream FlowElements:

```
<FlowTimeSlice id="ID_101_1423664342095_20">
  <flowId>18GWC&gt;SO</flowId>
  <downstreamFlowElement>
    <FlowLocationElement id="ID_101_1423664342095_23">
      <index>3</index>
      <locationChoice_airportHeliportSet href="urn:uuid:b7ed0827-57a6-489a-8fad-6788b1616ee0"/>
    </FlowLocationElement>
  </downstreamFlowElement>
  <upstreamFlowElement>
    <FlowLocationElement id="ID_101_1423664342095_24">
      <index>1</index>
      <locationChoice_airspace href="urn:uuid:f61af630-fe9e-4847-9757-9ba4d36dcd82"/>
    </FlowLocationElement>
  </upstreamFlowElement>
  <upstreamFlowElement>
    <FlowLocationElement id="ID_101_1423664342095_25">
      <index>2</index>
      <locationChoice_navaid href="urn:uuid:f967e31c-5b26-4c4e-8a9d-9c85291f62ee"/>
    </FlowLocationElement>
  </upstreamFlowElement>
</FlowTimeSlice>
```

2.2.5.1.13. Navaid Feature

(1) The exported attributes are:

a) type

b) designator

c) name

(2) The exported association is:

a) location

(3) Example:

```
<aixm:NavaidTimeSlice gml:id="ID_170_1385510754495_73754">
  <aixm:type>NDB_MKR</aixm:type>
  <aixm:designator>AA</aixm:designator>
  <aixm:name>AASIAAT</aixm:name>
  <aixm:location>
    <aixm:ElevatedPoint gml:id="ID_170_1385510754495_73755">
     <gml:pos srsName="urn:ogc:def:crs:EPSG::4326">68.72305555555556 -52.78472222222222</gml:pos>
    </aixm:ElevatedPoint>
  </aixm:location>
</aixm:NavaidTimeSlice>
```

2.2.5.1.14. OrganisationAuthority Feature

(1) The exported attributes are:

a) name

b) designator

c) type
The type is always STATE.

(2) Example:

```
<aixm:OrganisationAuthorityTimeSlice gml:id="ID_170_1385510754495_73755">
  <aixm:name>KINGDOM OF BELGIUM</aixm:name>
  <aixm:designator>BELGIUM</aixm:designator>
  <aixm:type>STATE</aixm:type>
</aixm:OrganisationAuthorityTimeSlice>
```

2.2.5.1.15. ReferenceLocation Feature (ADR-E)

(1) A Reference Location can be one of the following entities:

- An Airspace
- An AirportHeliport

- An AirportHeliportSet
- A SignificantPoint (either a DesignatedPoint or a Navaid)

(2) One Reference Location can be used in multiple Traffic Volumes.

(3) If the location is a Significant Point, it has an associattion to a Flight Level Range (AirspaceLayer)

(4) A Reference Location can have associated Flows (see also Flow Feature and TrafficVolume Feature).
This allows reusing the same Reference Location and the same Flows in multiple Traffic Volumes without having to duplicate the Flows in each Traffic Volume.
When this happens, a Traffic Volume associated to this Reference Location may have

- *Linked Flows*, i.e. Flows which are directly linked to the Traffic Volumes,
- *Associated Flows*, i.e. Flows which are associated to the Reference Location

These concepts will be further explained below, when discussing Traffic Volumes.

(5) The exported attributes are:

a) referenceLocationId

b) category
Possible values are: ARR = Arrival, DEP = Departure, ALL = Arrival and Departure.
The Category is always set to ALL when the location is an Airspace or a Significant Point, whereas it can take any of the three values if the location is an AirportHeliport or an AirportHeliportSet.

(6) The exported association is:

a) location
A reference to one of: Airspace, AirportHeliport, AirportHeliportSet, DesignatedPoint, Navaid.

b) airspaceLayer
Present only when the location is a Significant Point.

(7) Example of a ReferenceLocation to an Airpace

```
<ReferenceLocationTimeSlice id="ID_704_1423477742163_19">
  <category>ALL</category>
  <referenceLocationId>ASEBTCWXG</referenceLocationId>
  <locationChoice_airspace href="urn:uuid:a875ce9c-512b-40ba-9ac6-e85032350cb0"/>
</ReferenceLocationTimeSlice>
```

(8) Example of a ReferenceLocation to a DesignatedPoint

```
<ReferenceLocationTimeSlice id="ID_193_1423620275860_25039">
  <category>ALL</category>
  <referenceLocationId>SPABLOMG</referenceLocationId>
  <airspaceLayer>
    <AirspaceLayer id="ID_193_1423620275860_25042">
      <upperLimit uom="FT">UNL</upperLimit>
```

```
      <upperLimitReference>MSL</upperLimitReference>
      <lowerLimit uom="FT">GND</lowerLimit>
      <lowerLimitReference>MSL</lowerLimitReference>
    </AirspaceLayer>
  </airspaceLayer>
  <locationChoice_designatedPoint href="urn:uuid:aa19ebe5-485c-40f9-83a5-2aa3e5ae88d1"/>
</ReferenceLocationTimeSlice>
```

2.2.5.1.16. Route Feature

(1) The exported attributes are:

a) designatorPrefix

b) designatorSecondLetter

c) designatorNumber

(2) None of the associations are exported.

(3) Example:

```
<aixm:RouteTimeSlice gml:id="ID_170_1385510754495_73755">
  <aixm:designatorPrefix>K</aixm:designatorPrefix>
  <aixm:designatorSecondLetter>H</aixm:designatorSecondLetter>
  <aixm:designatorNumber>501</aixm:designatorNumber>
</aixm:RouteTimeSlice>
```

2.2.5.1.17. RouteSegment Feature

(1) Feature identification (see Feature/Object identification): This Feature type exists as an entity in CACD when it is part of an ATS Route, but not if it is part of a NAT Track. This means that when the RouteSegment is part of an ATS Route it has its own UUID but when it is part of a NAT Track then it needs an artificially generated `gml:identifier` which is the concatenation of the NAT Track's UUID, plus the UUIDs of the first and last point of the segment in this order.
Example:

```
Route_024bb6f8-3265-472a-9988-c765f519bcef.
      7ae44b19-3827-4ce9-8fd1-ac20ca5e2ead.
      f24473c7-85f8-4329-965f-4057bfa7c60c
```

(2) The exported attributes are:

a) upperLimit and upperLimitReference

b) lowerLimit and lowerLimitReference

(3) The exported associations are:

a) start

b) routeFormed

c) end

d) availability: these are the "permanent" availabilities, i.e. the Timesheets (PropertiesWithSchedule) express a Time Schedule (see Timesheet Time Schedule).
The RouteAvailabilityExtension (ADR-E) is used to code the availability.

e) cdrUpdate (ADR-E): these are the temporary overriding availabilities, i.e. the Timesheets (PropertiesWithSchedule) express a TimePeriod (see Timesheet Time Period).

(4) Example:

```
<aixm:RouteSegmentTimeSlice gml:id="ID_172_1385510754499_82">
  <aixm:upperLimit uom="FL">430</aixm:upperLimit>
  <aixm:upperLimitReference>STD</aixm:upperLimitReference>
  <aixm:lowerLimit uom="FL">185</aixm:lowerLimit>
  <aixm:lowerLimitReference>STD</aixm:lowerLimitReference>
  <aixm:start>
    <aixm:EnRouteSegmentPoint gml:id="ID_172_1385510754499_82">
      <aixm:pointChoice_fixDesignatedPoint xlink:href="urn:uuid:f24473c7-85f8-4329-965f-..."/>
    </aixm:EnRouteSegmentPoint>
  </aixm:start>
  <aixm:routeFormed xlink:href="urn:uuid:024bb6f8-3265-472a-9988-c765f519bcef"/>
  <aixm:end>
    <aixm:EnRouteSegmentPoint gml:id="ID_172_1385510754499_83">
      <aixm:pointChoice_fixDesignatedPoint xlink:href="urn:uuid:7ae44b19-3827-4ce9-8fd1-..."/>
    </aixm:EnRouteSegmentPoint>
  </aixm:end>
  <aixm:availability/><!-- not expanded here -->
</aixm:RouteSegmentTimeSlice>
```

(5) RouteSegments form a Route. In the CACD model they are grouped together into portions. The portions can be in the down or in the up direction. Up and Down portions do not necessarily match. E.g. a Route A-B-C-D-E-F-G could be organised in portions with the following segments:

Portion	**Constituting RouteSegments**
PortionForward=1	A-B
PortionForward=2	B-C C-D D-E
PortionForward=3	E-F F-G
PortionBackward=1	F-G
PortionBackward=2	C-D D-E E-F
PortionBackward=3	A-B B-C

Table 2.1. Partitioning of CACD Routes into portions.

(6) Example:

```
  <aixm:RouteSegmentTimeSlice>
  <!-- start B -->
  <!-- end C -->
  <!-- availability -->
```

```
  <aixm:annotation>
    <aixm:Note gml:id="ID_NOTE_1">
      <aixm:purpose>REMARK</aixm:purpose>
      <aixm:translatedNote>
        <aixm:LinguisticNote gml:id="ID_NOTE_1_1">
          <aixm:note>PortionForward=2</aixm:note>
        </aixm:LinguisticNote>
      </aixm:translatedNote>
    </aixm:Note>
  </aixm:annotation>
  <aixm:annotation>
    <aixm:Note gml:id="ID_NOTE_2">
      <aixm:purpose>REMARK</aixm:purpose>
      <aixm:translatedNote>
        <aixm:LinguisticNote gml:id="ID_NOTE_2_1">
          <aixm:note>PortionBackward=1</aixm:note>
        </aixm:LinguisticNote>
      </aixm:translatedNote>
    </aixm:Note>
  </aixm:annotation>
</aixmRouteSegmentTimeSlice>
```

2.2.5.1.18. SpecialDate Feature

(1) The exported attributes are:

a) type
The type is either HOL or BUSY_FRI.

b) dateDay

c) dateYear

d) name

(2) The exported association is:

a) authority

(3) Example specific day:

```
<aixm:SpecialDateTimeSlice gml:id="ID_172_1385510754499_82">
  <aixm:type>HOL</aixm:type>
  <aixm:dateDay>19-06</aixm:dateDay>
  <aixm:dateYear>2014</aixm:dateYear>
  <aixm:name>CORPUS CHRISTI DAY (M)</aixm:name>
  <aixm:authority xlink:href="urn:uuid:11bf3600-dcba-4448-85d9-259b3b5e88b7"/>
</aixm:SpecialDateTimeSlice>
```

(4) Example re-occurring holiday:

```
<aixm:SpecialDateTimeSlice gml:id="ID_172_1385510754499_82">
  <aixm:type>HOL</aixm:type>
  <aixm:dateDay>25-12</aixm:dateDay>
  <aixm:name>CHRISTMAS DAY</aixm:name>
  <aixm:authority xlink:href="urn:uuid:2ada7e48-c90d-4f2c-b33e-3f75dd995566"/>
</aixm:SpecialDateTimeSlice>
```

(5) Examply busy Friday:

```
<aixm:SpecialDateTimeSlice gml:id="ID_172_1385510754499_82">
  <aixm:type>BUSY_FRI</aixm:type>
  <aixm:dateDay>25-10</aixm:dateDay>
  <aixm:dateYear>2013</aixm:dateYear>
  <aixm:authority xlink:href="urn:uuid:609bcbaf-0960-45da-9e80-645ecdf499f5"/>
</aixm:SpecialDateTimeSlice>
```

2.2.5.1.19. StandardInstrumentArrival Feature

(1) The exported attributes are:

a) designator

b) instruction

(2) The exported associations are:

a) availability

b) airportHeliport

c) connectingPoint (ADR-E)

d) initialApproachFix (ADR-E)

(3) Example:

```
<aixm:StandardInstrumentArrivalTimeSlice gml:id="ID_172_1385510754499_784149">
  <instruction>VIA UP/P7</instruction>
  <availability>
    <!-- not expanded here -->
  </availability>
  <airportHeliport href="urn:uuid:02876331-3e92-4cea-a67f-a4cf6cf9aefd"/>
  <designator>ABBOT1B</designator>
  <extension>
    <StandardInstrumentArrivalExtension id="ID_142_1423664916078_10">
      <connectingPoint>
        <TerminalSegmentPoint id="ID_142_1423664916078_11">
          <pointChoice_fixDesignatedPoint href="urn:uuid:260a38b8-61ec-4f0e-91d8-d9c79270f461"/>
        </TerminalSegmentPoint>
      </connectingPoint>
      <connectingPoint>
        <TerminalSegmentPoint id="ID_142_1423664916078_12">
          <pointChoice_fixDesignatedPoint href="urn:uuid:376efb12-ca41-46fb-895e-e1a8060210d1"/>
        </TerminalSegmentPoint>
      </connectingPoint>
      <connectingPoint>
        <TerminalSegmentPoint id="ID_142_1423664916078_13">
          <pointChoice_fixDesignatedPoint href="urn:uuid:fe43cb87-cd23-47ad-9a60-60f25cfe451b"/>
        </TerminalSegmentPoint>
      </connectingPoint>
      <connectingPoint>
        <TerminalSegmentPoint id="ID_142_1423664916078_14">
          <pointChoice_fixDesignatedPoint href="urn:uuid:64464f67-bb36-4d72-b664-1acf24332780"/>
        </TerminalSegmentPoint>
      </connectingPoint>
    </StandardInstrumentArrivalExtension>
```

```
  </extension>
</StandardInstrumentArrivalTimeSlice>
```

2.2.5.1.20. StandardInstrumentDeparture Feature

(1) The exported attributes are:

a) designator

b) instruction

(2) The exported associations are:

a) availability

b) airportHeliport

c) connectingPoint (ADR-E)

(3) Example:

```
<aixm:StandardInstrumentDepartureTimeSlice gml:id="ID_172_1385510754499_780091">
  <aixm:instruction>FL 110 MAX -----&gt; ONLY DEST LFCI/FL110 MAX</aixm:instruction>
  <aixm:availability/><!-- not expanded here -->
  <aixm:airportHeliport xlink:href="urn:uuid:5ba9179b-3306-486e-8970-edce59758396"/>
  <aixm:designator>AB5E</aixm:designator>
  <aixm:extension>
    <adr:StandardInstrumentDepartureExtension gml:id="ID_172_1385510754499_780092">
      <adr:connectingPoint>
        <aixm:TerminalSegmentPoint gml:id="ID_172_1385510754499_780093">
          <aixm:pointChoice_navaidSystem xlink:href="urn:uuid:8b359a52-bf2c-4dc6-af1a-..."/>
        </aixm:TerminalSegmentPoint>
      </adr:connectingPoint>
      <adr:connectingPoint>
        <aixm:TerminalSegmentPoint gml:id="ID_172_1385510754499_780094">
          <aixm:pointChoice_navaidSystem xlink:href="urn:uuid:aca80964-9e4f-4e59-970b-..."/>
        </aixm:TerminalSegmentPoint>
      </adr:connectingPoint>
    </adr:StandardInstrumentDepartureExtension>
  </aixm:extension>
</aixm:StandardInstrumentDepartureTimeSlice>
```

2.2.5.1.21. TrafficVolume Feature (ADR-E)

(1) A Traffic Volume represents a volume of air traffic.
It is used for monitoring the amount of air traffic over a given object (e.g. an Airspace) called Reference Location, so that a Regulation can be applied if the load is higher than the available capacity.

(2) A Traffic Volume is always associated to exactly one Reference Location (see ReferenceLocation Feature above).

(3) In a simplistic approach, it would seem enough to define a capacity for the Reference Location and count all the flights entering that location in a given unit of time (which is the hour). In reality

not all flights crossing a location contribute to the complexity of the traffic in the same way: for example if the majority of the traffic is in the southern part of an airspace and only few flights cross the northern part, it would be desirable to set a specific monitoring for the southern flights alone.

(4) This is why a Traffic Volume can be refined with Flows (see Flow Feature above).

(5) A Traffic Volume is therefore the combination of one Reference Location and potentially multiple Flows.

(6) It is worth noticing here that a Reference Location can itself have Flows (see ReferenceLocation Feature).

- The Flows defined in the Reference Location are called *Associated Flows*
- The Flows defined in the Traffic Volume are called *Linked Flows*

(7) When defining the Linked Flows in a Traffic Volume, these can be combined in several ways. Each Linked Flow can be:

- *Included*
  Flights matching the Flow are included in the volume of air traffic.
  IMPORTANT: If at least one Included Flow is defined, then the volume of air traffic is defined solely by the Linked Flows. In other words if there is at least one Included Flow then a flight to be in the Traffic Volume must be in one of the Included Flows.
- *Excluded*
  Flights matching the Excluded Flows are excluded from the volume and therefore do not contribute to the counts.
- *Exempted*
  Flights matching an Exempted Flow are not affected by regulations. Exempted Flows partecipate in the counts provided that no Included Flows are defined (if any Included Flows were defined only flights matching those Flows would be included in the counts, see above).
- *Included Exempted*
  Flights matching an Included Exempted Flow always partecipate in the counts but are not affected by regulations. The Flow behaves both as an Included Flow and an Exempted Flow at the same time.

(8) Traffic Volumes can be active or not according to a timetable. This reflects in principle the Sector Configurations and the rationale behind is that the amount of traffic changes according to the period of the year, the day of the week and the time of the day. So for example in a time frame when the traffic is relatively low, a single Traffic Volume could suffice, whereas in a time of high load, the same volume could be split into smaller Traffic Volumes to allow a more granular monitoring. So according to a timetable, the large Traffic Volume could be made inactive and the smaller ones active.

(9) When a Traffic Volume is not active the counts are not computed.

(10) The exported attributes are:

a) tvId

(11) The exported associations are:

a) activation

b) linkedFlow

(12) Example:

```
<TrafficVolumeTimeSlice id="ID_705_1423477842671_179">
  <tvId>EGPESTX</tvId>
  <referenceLocation href="urn:uuid:00495049-ecc1-4936-a1d0-35bbbdaaf2fe"/>
  <activation>
    <TrafficVolumeActivation id="ID_705_1423477842671_182">
      <!-- not expanded here -->
    </TrafficVolumeActivation>
  </activation>
  <linkedFlow>
    <TrafficVolumeLinkedFlow id="ID_705_1423477842671_186">
      <role>EXCLUDED</role>
      <theFlow href="urn:uuid:c9c75b61-4c5e-4e4f-be75-cdef7b94f258"/>
    </TrafficVolumeLinkedFlow>
  </linkedFlow>
</TrafficVolumeTimeSlice>
```

2.2.5.1.22. TrafficVolumeSet Feature (ADR-E)

(1) Represents a set of Traffic Volumes as a single entity.

(2) The exported attributes are:

a) tvSetId

(3) The exported associations are:

a) trafficVolume
Traffic Volume included in this TrafficVolumeSet

(4) Example:

```
<TrafficVolumeSetTimeSlice id="ID_46_1423644956269_3">
  <tvSetId>AEROEDNY</tvSetId>
  <trafficVolume href="urn:uuid:64b3ec4b-f673-4709-9771-4517fb70b72b"/>
  <trafficVolume href="urn:uuid:ba60a7d2-11e1-4948-9c0f-566f2db6a23e"/>
  <trafficVolume href="urn:uuid:4b2a81eb-f2dc-4e82-b296-00a406fc9850"/>
</TrafficVolumeSetTimeSlice>
```

2.2.5.1.23. Unit Feature

(1) The exported attributes are:

a) name

b) type

The type is always OTHER:__ADR__AMC.

c) designator

(2) None of the associations are exported.

(3) Example:

```
<aixm:UnitTimeSlice gml:id="ID_172_1385510754499_780092">
  <aixm:name>BELGIUM</aixm:name>
  <aixm:type>OTHER:__ADR__AMC</aixm:type>
  <aixm:designator>EBBRZAMC</aixm:designator>
</aixm:UnitTimeSlice>
```

2.2.5.2. Objects

2.2.5.2.1. AirspaceActivation Object

(1) The exported attribute is:

a) status

(2) The exported associations are:

a) levels

b) hostAirspace (ADR-E)
The Information Region in which the (military) Airspace is located.

c) requestor (ADR-E)

d) fuaRestriction (ADR-E)
The activation of the associated FUARestrictions (see <FUARestrictionActivation>)

(3) Typical example from AirspaceActivation in the retrieval of an AUP (AirspaceAvailability Service):

```
<aixm:activation>
  <aixm:AirspaceActivation gml:id="ID_4247_1381851018763_6">
    <aixm:status>ACTIVE</aixm:status>
    <aixm:levels>
      <aixm:AirspaceLayer gml:id="ID_4247_1381851018763_7">
        <aixm:upperLimit uom="FL">55</aixm:upperLimit>
        <aixm:upperLimitReference>STD</aixm:upperLimitReference>
        <aixm:lowerLimit uom="FT">GND</aixm:lowerLimit>
        <aixm:lowerLimitReference>MSL</aixm:lowerLimitReference>
      </aixm:AirspaceLayer>
    </aixm:levels>
    <aixm:extension>
      <adr:AirspaceActivationExtension
            xmlns:adr="http://www.aixm.aero/schema/5.1/extensions/ADR"
            gml:id="ID_4247_1381851018763_8">
        <adr:hostAirspace
            xmlns:xlink="http://www.w3.org/1999/xlink"
            xlink:href="urn:uuid:126db09b-215c-4eee-981b-27063b544b4a"/>
      </adr:AirspaceActivationExtension>
    </aixm:extension>
```

```
  </aixm:AirspaceActivation>
</aixm:activation>
```

(4) Another typical example is the description of the availability of a (military) Airspace (AirspaceStructure Service):

```
<aixm:activation>
  <aixm:AirspaceActivation gml:id="ID_171_1385510754499_169471">
    <aixm:status>AVBL_FOR_ACTIVATION</aixm:status>
    <aixm:extension>
      <adr:PropertiesWithScheduleExtension gml:id="ID_171_1385510754499_169472">
        <adr:timeTable>
          <adr:TimeTable gml:id="ID_171_1385510754499_169473">
            <!-- not expanded here -->
          </adr:TimeTable>
        </adr:timeTable>
      </adr:PropertiesWithScheduleExtension>
    </aixm:extension>
  </aixm:AirspaceActivation>
</aixm:activation>
```

2.2.5.2.2. AirspaceLayer Object

(1) It is primarily used in the RouteSegmentExtension

(2) The exported attributes are:

a) upperLimit

b) upperLimitReference

c) lowerLimit

d) lowerLimitReference

(3) There are no associations.

(4) An example shows the special behaviour:

```
<adr:RouteSegmentExtension gml:id="ID_172_1385510754499_98">
  <adr:levels gml:id="ID_172_1385510754499_99">
    <aixm:upperLimit uom="FL">430</aixm:upperLimit>
    <aixm:upperLimitReference>STD</aixm:upperLimitReference>
    <aixm:lowerLimit uom="FL">185</aixm:lowerLimit>
    <aixm:lowerLimitReference>STD</aixm:lowerLimitReference>
  </adr:levels>
</adr:RouteSegmentExtension>
```

2.2.5.2.3. FlightConditionCombination Object

(1) The exported attribute is:

a) logicalOperator

(2) The exported association are:

a) element

b) timeInterval

(3) Example:

```
<aixm:FlightConditionCombination gml:id="ID_168_1385510754493_6">
  <aixm:timeInterval>
    <aixm:Timesheet gml:id="ID_168_1385510754493_7">
      <!-- not expanded here -->
    </aixm:Timesheet>
  </aixm:timeInterval>
  <aixm:logicalOperator>ANDNOT</aixm:logicalOperator>
  <aixm:element>
    <!-- not expanded here -->
  </aixm:element>
  <aixm:element>
    <!-- not expanded here -->
  </aixm:element>
</aixm:FlightConditionCombination>
```

2.2.5.2.4. FlightConditionElement Object

(1) None of the attributes are exported.

(2) The exported associations are:

a) flightCondition

b) operationalCondition

c) flightLevel

d) borderCrossingCondition (ADR-E)

e) airportHeliportSetCondition (ADR-E)

(3) Example of a FlightConditionElement crossing an Airspace reference location between certain levels:

```
<aixm:FlightConditionElement gml:id="ID_168_1385510754493_11">
  <aixm:flightCondition_airspaceCondition xlink:href="urn:uuid:d664b2c4-8b52-4387-b26a-..."/>
  <aixm:operationalCondition>
    <aixm:FlightConditionCircumstance gml:id="ID_168_1385510754493_12">
      <aixm:referenceLocation>YES</aixm:referenceLocation>
      <aixm:relationWithLocation>XNG</aixm:relationWithLocation>
    </aixm:FlightConditionCircumstance>
  </aixm:operationalCondition>
  <aixm:flightLevel>
    <aixm:FlightRestrictionLevel gml:id="ID_168_1385510754493_13">
      <aixm:upperLevel uom="FL">115</aixm:upperLevel>
      <aixm:upperLevelReference>STD</aixm:upperLevelReference>
      <aixm:lowerLevel uom="FT">GND</aixm:lowerLevel>
      <aixm:lowerLevelReference>MSL</aixm:lowerLevelReference>
    </aixm:FlightRestrictionLevel>
```

```
  </aixm:flightLevel>
</aixm:FlightConditionElement>
```

(4) Example of a FlightConditionElement using the borderCrossingCondition (ADR-E):

```
<aixm:FlightConditionElement gml:id="ID_168_1385510754493_3062">
  <aixm:operationalCondition>
    <aixm:FlightConditionCircumstance gml:id="ID_168_1385510754493_3063">
      <aixm:referenceLocation>YES</aixm:referenceLocation>
      <aixm:relationWithLocation>XNG</aixm:relationWithLocation>
    </aixm:FlightConditionCircumstance>
  </aixm:operationalCondition>
  <aixm:flightLevel>
    <aixm:FlightRestrictionLevel gml:id="ID_168_1385510754493_3064">
      <aixm:upperLevel uom="FL">245</aixm:upperLevel>
      <aixm:upperLevelReference>STD</aixm:upperLevelReference>
      <aixm:lowerLevel uom="FT">GND</aixm:lowerLevel>
      <aixm:lowerLevelReference>MSL</aixm:lowerLevelReference>
    </aixm:FlightRestrictionLevel>
  </aixm:flightLevel>
  <aixm:extension>
    <adr:FlightConditionElementExtension gml:id="ID_168_1385510754493_3065">
      <adr:borderCrossingCondition>
        <adr:AirspaceBorderCrossingObject gml:id="ID_168_1385510754493_3066">
          <adr:exitedAirspace xlink:href="urn:uuid:d664b2c4-8b52-4387-b26a-4f993180c545"/>
          <adr:enteredAirspace xlink:href="urn:uuid:6013c06c-6cba-4a83-b0dd-b9e527c86920"/>
        </adr:AirspaceBorderCrossingObject>
      </adr:borderCrossingCondition>
    </adr:FlightConditionElementExtension>
  </aixm:extension>
</aixm:FlightConditionElement>
```

(5) Example of a FlightConditionElement using the airportHeliportSetCondition (ADR-E):

```
<aixm:FlightConditionElement gml:id="ID_168_1385510754493_3062">
  <aixm:operationalCondition>
    <aixm:FlightConditionCircumstance gml:id="ID_168_1385510754493_3063">
      <aixm:referenceLocation>YES</aixm:referenceLocation>
      <aixm:relationWithLocation>DEP</aixm:relationWithLocation>
    </aixm:FlightConditionCircumstance>
  </aixm:operationalCondition>
  <aixm:extension>
    <adr:FlightConditionElementExtension gml:id="ID_168_1385510754493_3065">
      <adr:airportHeliportSetCondition xlink:href="urn:uuid:d664b2c4-8b52-4387-b26a-4f..."/>
    </adr:FlightConditionElementExtension>
  </aixm:extension>
</aixm:FlightConditionElement>
```

2.2.5.2.5. FlowLocationElement Object

(1) The exported attribute is:

a) index
Defines the order of the FlowElements in the upstream or downstream sequence.

(2) The exported association is:

a) location

The location that identifies this FlowElement.

(3) Example of two FlowLocationElements representing the downstream and upstream flow location elements of a Flow:

```
<downstreamFlowElement>
  <FlowLocationElement id="ID_699_1423477726446_12">
    <index>2</index>
    <locationChoice_navaid href="urn:uuid:bf538f1c-21bd-4629-a893-2a1b8292cb8a"/>
  </FlowLocationElement>
</downstreamFlowElement>
<upstreamFlowElement>
  <FlowLocationElement id="ID_699_1423477726446_13">
    <index>1</index>
    <locationChoice_designatedPoint href="urn:uuid:07765def-1d2b-4c9d-8b39-997a61f4cb49"/>
  </FlowLocationElement>
</upstreamFlowElement>
```

2.2.5.2.6. FUARestrictionActivation Object (ADR-E)

(1) The exported attributes are

a) active: indication that the FUA restriction (theFlightRestriction) is activated during the AirspaceActivation.

b) remark: a remark specific to why the FUA restriction was (not) activated. The remark is only used in the context of AirspaceAvailability Service ((E)AUP/UUP related). In the context of Airspace.rsaActivation (ADR-E), the remark field is not exported.

(2) The exported association is:

a) theFlightRestriction: reference to the FUA restriction (using uuid)

(3) Example:

```
<aixm:extension>
  <adr:AirspaceActivationExtension gml:id="ID_171_1385510754499_169472_1">
    <adr:fuaRestriction>
      <adr:FUARestrictionActivation gml:id="ID_171_1385510754499_169472_2">
        <adr:active>YES</adr:active>
        <!-- remark field not used in the context of Airspace.rsaActivation -->
        <adr:theFlightRestriction xlink:href="urn:uuid:9f7051bf-a004-4d51-b3a4-df4283fe11cd"/>
      </adr:FUARestrictionActivation>
    </adr:fuaRestriction>
    <!-- potentially more fuaRestrictions -->
  </adr:AirspaceActivationExtension>
</aixm:extension>
```

2.2.5.2.7. IntermediateSignificantPoint Object (ADR-E)

(1) There are no attributes.

(2) The exported association is:

a) pointChoice

(3) The IntermediateSignificantPoint (ADR-E) is only used by the RoutePortionExtension.

(4) Example:

```
<adr:IntermediateSignificantPoint gml:id="ID_168_1385510754493_741">
  <adr:pointChoice_fixDesignatedPoint xlink:href="urn:uuid:1b3950ed-6e59-4a0e-a732-49..."/>
</adr:IntermediateSignificantPoint>
```

2.2.5.2.8. ProcedureAvailability Object

(1) The exported attributes is:

a) status
The value is always set to USABLE.

(2) The exported association is:

a) timeInterval

(3) Example:

```
<aixm:ProcedureAvailability gml:id="ID_172_1385510754499_780273">
  <aixm:timeInterval>
    <aixm:Timesheet gml:id="ID_172_1385510754499_780273_1">
      <!-- not expanded here -->
    </aixm:Timesheet>
  </aixm:timeInterval>
  <aixm:status>USABLE</aixm:status>
</aixm:ProcedureAvailability>
```

2.2.5.2.9. PropertiesWithSchedule Object

(1) There are no attributes.

(2) The exported association is:

a) timeInterval

(3) Example: see Timesheet Object.

2.2.5.2.10. RouteAvailability Object

(1) The exported attributes are:

a) direction

b) status

c) conditionalRouteType (ADR-E)

(2) The exported associations are:

a) timeInterval

b) levels

c) hostAirspace (ADR-E)
This is only used in the context of an AUP, to indicate in which Information Region the RouteSegment is located.

(3) Example of a RouteAvailability defining the schedule of CDR usage:

```
<aixm:RouteAvailability gml:id="ID_172_1385510754499_2371">
  <aixm:timeInterval>
    <aixm:Timesheet gml:id="ID_172_1385510754499_2371">
      <!-- not expanded here, see Timesheet (schedule definitions) -->
    </aixm:Timesheet>
  </aixm:timeInterval>
  <aixm:direction>FORWARD</aixm:direction>
  <aixm:status>COND</aixm:status>
  <aixm:levels>
    <aixm:AirspaceLayer gml:id="ID_172_1385510754499_2372">
      <aixm:upperLimit uom="FL">245</aixm:upperLimit>
      <aixm:upperLimitReference>STD</aixm:upperLimitReference>
      <aixm:lowerLimit uom="FL">95</aixm:lowerLimit>
      <aixm:lowerLimitReference>STD</aixm:lowerLimitReference>
      <aixm:discreteLevelSeries xlink:href="urn:uuid:SLC_ODD_IFR"/>
    </aixm:AirspaceLayer>
  </aixm:levels>
  <aixm:extension>
    <adr:RouteAvailabilityExtension gml:id="ID_172_1385510754499_2373">
      <adr:conditionalRouteType>CDR_3</adr:conditionalRouteType>
    </adr:RouteAvailabilityExtension>
  </aixm:extension>
</aixm:RouteAvailability>
```

2.2.5.2.11. RoutePortion Object

(1) There are no attributes.

(2) The exported associations are:

a) start

b) referencedRoute

c) end

d) intermediatePoint (ADR-E)

(3) Example:

```
<aixm:RoutePortion gml:id="ID_168_1385510754493_739">
  <aixm:start_navaidSystem xlink:href="urn:uuid:682543b6-97c6-4bf7-b95c-21a1e5b8c998"/>
  <aixm:referencedRoute xlink:href="urn:uuid:77af856b-078c-484b-a20a-3216cef1be2d"/>
  <aixm:end_fixDesignatedPoint xlink:href="urn:uuid:a7b98cb4-4e28-4258-8a4e-d8b5ab3d95d0"/>
  <aixm:extension>
    <adr:RoutePortionExtension gml:id="ID_168_1385510754493_740">
      <adr:intermediatePoint>
        <adr:IntermediateSignificantPoint gml:id="ID_168_1385510754493_741">
          <adr:pointChoice_fixDesignatedPoint xlink:href="urn:uuid:1b3950ed-6e59-4a0e-a73..."/>
        </adr:IntermediateSignificantPoint>
```

```
      </adr:intermediatePoint>
    </adr:RoutePortionExtension>
  </aixm:extension>
</aixm:RoutePortion>
```

2.2.5.2.12. Timesheet Object

2.2.5.2.12.1. Time Period

(1) The core aixm:Timesheet doesn't allow to express a time period because the Timesheet.startDate and Timesheet.endDate are of type aixm:DateMonthDayType.

(2) The Timesheet was extended with a gml:validTime to have the possibility to express a time period. An alternative extension would have been to extend the Timesheet with a startYear and endYear, but this alternative was rejected because it would confuse too much the experienced AIXM user.

(3) The following Timesheet properties are set with fixed values.

Property	Value
timeReference	UTC
day	ANY
excluded	NO

Table 2.2. Timesheet properties with fixed values when defining a time period

(4) Example expressing the time period 2013/11/05 20:30 --> 2013/11/05 21:00.

```
<aixm:Timesheet gml:id="ID_50_1352812184610_10_1">
  <aixm:timeReference>UTC</aixm:timeReference>
  <aixm:day>ANY</aixm:day>
  <aixm:excluded>NO</aixm:excluded>
  <aixm:extension>
    <adr:TimesheetExtension gml:id="ID_50_1352812184610_10_2">
      <gml:validTime>
        <gml:TimePeriod gml:id="ID_40_1377648348105_83">
          <gml:beginPosition>2013-11-05T20:30:00</gml:beginPosition>
          <gml:endPosition>2013-11-05T21:00:00</gml:endPosition>
        </gml:TimePeriod>
      </gml:validTime>
    </adr:TimesheetExtension>
  </aixm:extension>
</aixm:Timesheet>
```

2.2.5.2.12.2. Time Schedule

(1) The Time Schedule information in CACD can be represented by a string like "2012/10/18 -> 2014/02/06 -----67 00:00 06:00". The meaning of this is: between the Time Period 2012/10/18 (midnight) -> 2014/02/06 (midnight), on Saturdays(6) and Sundays(7) from 00:00 until 06:00.

(2) The Time Schedule information is always part of an AIXM Object that is derived from PropertiesWithSchedule.

(3) Apart from the days of the week, holidays, the day before/after holidays and busy Fridays are possible.

(4) For each day in a string like "2012/10/18 -> 2014/02/06 -----67 00:00 06:00", a separate aixm:timeInterval element is needed.

(5) The following Timesheet properties are set:

Property	**Value**
timeReference	UTC
day	'day of the week or special day'
startTime	'start time'
endTime	'end time'
excluded	NO
gml:validTime (ADR-E)	'the surrounding time period'

Table 2.3. Timesheet properties when defining a Time Schedule

(6) Example "2012/10/18 -> 2014/02/06 -----67 00:00 06:00":

```
<aixm:timeInterval>
  <aixm:Timesheet gml:id="ID_50_1352812184610_8_1">
    <aixm:timeReference>UTC</aixm:timeReference>
    <aixm:day>SAT</aixm:day>
    <aixm:startTime>00:00</aixm:startTime>
    <aixm:endTime>06:00</aixm:endTime>
    <aixm:excluded>NO</aixm:excluded>
    <aixm:extension>
      <adr:TimesheetExtension gml:id="ID_50_1352812184610_8_2">
        <gml:validTime>
          <gml:TimePeriod gml:id="ID_40_1377648348105_81">
            <gml:beginPosition>2012-10-18T00:00:00</gml:beginPosition>
            <gml:endPosition>2014-02-06T00:00:00</gml:endPosition>
          </gml:TimePeriod>
        </gml:validTime>
      </adr:TimesheetExtension>
    </aixm:extension>
  </aixm:Timesheet>
</aixm:timeInterval>
<aixm:timeInterval>
  <aixm:Timesheet gml:id="ID_50_1352812184610_9_1">
    <aixm:timeReference>UTC</aixm:timeReference>
    <aixm:day>SUN</aixm:day>
    <aixm:startTime>00:00</aixm:startTime>
    <aixm:endTime>06:00</aixm:endTime>
    <aixm:excluded>NO</aixm:excluded>
    <aixm:extension>
      <adr:TimesheetExtension gml:id="ID_50_1352812184610_9_2">
        <gml:validTime>
          <gml:TimePeriod gml:id="ID_40_1377648348105_82">
            <gml:beginPosition>2012-10-18T00:00:00</gml:beginPosition>
            <gml:endPosition>2014-02-06T00:00:00</gml:endPosition>
          </gml:TimePeriod>
        </gml:validTime>
      </adr:TimesheetExtension>
    </aixm:extension>
```

```
  </aixm:Timesheet>
</aixm:timeInterval>
```

(7) For summertime and wintertime periods, the gml:validTime (ADR-E) is not used. Instead the core AIXM startDate, endDate are used with the values 'SDLST' and 'EDLST'. The daylightSavingAdjust is set explicitly to 'NO'.
Examples:

```
<!-- WINTERTIME start EDLST end SDLST -->
<aixm:timeInterval>
  <aixm:Timesheet gml:id="uuid.7033a5ab-b30c-4daa-82c3-68ff420e1330">
    <aixm:timeReference>UTC</aixm:timeReference>
    <aixm:startDate>EDLST</aixm:startDate>
    <aixm:endDate>SDLST</aixm:endDate>
    <aixm:day>MON</aixm:day>
    <aixm:startTime>08:30</aixm:startTime>
    <aixm:endTime>17:00</aixm:endTime>
    <aixm:daylightSavingAdjust>NO</aixm:daylightSavingAdjust>
    <aixm:excluded>NO</aixm:excluded>
  </aixm:Timesheet>
</aixm:timeInterval>
<!-- SUMMERTIME start SDLST end EDLST -->
<aixm:timeInterval>
  <aixm:Timesheet gml:id="uuid.7033a5ab-b30c-4daa-82c3-68ff420e1330">
    <aixm:timeReference>UTC</aixm:timeReference>
    <aixm:startDate>EDLST</aixm:startDate>
    <aixm:endDate>SDLST</aixm:endDate>
    <aixm:day>MON</aixm:day>
    <aixm:startTime>07:30</aixm:startTime>
    <aixm:endTime>16:00</aixm:endTime>
    <aixm:daylightSavingAdjust>NO</aixm:daylightSavingAdjust>
    <aixm:excluded>NO</aixm:excluded>
  </aixm:Timesheet>
</aixm:timeInterval>
```

2.2.5.2.12.3. Default schedules

(1) Time Schedules can be very complex. E.g. in defining the CDRs, it is not allowed to have gaps in the schedule. In order to simplify the definitions, it is possible to define a 'default' schedule. The default schedule is defined as any time, subtracting some Time Schedules from this default.

(2) The default schedules is represented as a Timesheet with the following properties:

Property	**Value**
timeReference	UTC
day	ANY
excluded	NO
gml:validTime (ADR-E)	'the surrounding time period'

Table 2.4. Timesheet properties of the default schedule

(3) From the default schedule, the other Timeschedules are subtracted by setting the Timesheet.excluded property to YES.

(4) Example:

```
<aixm:timeInterval>
  <aixm:Timesheet gml:id="ID_50_1352812184610_7_1">
    <aixm:timeReference>UTC</aixm:timeReference>
    <aixm:day>ANY</aixm:day>
    <aixm:excluded>NO</aixm:excluded>
    <aixm:extension>
      <adr:TimesheetExtension gml:id="ID_50_1352812184610_8_2">
        <gml:validTime>
          <gml:TimePeriod gml:id="ID_40_1377648348105_81">
            <gml:beginPosition>2012-10-18T00:00:00</gml:beginPosition>
            <gml:endPosition>2014-02-06T00:00:00</gml:endPosition>
          </gml:TimePeriod>
        </gml:validTime>
      </adr:TimesheetExtension>
    </aixm:extension>
  </aixm:Timesheet>
</aixm:timeInterval>
<aixm:timeInterval>
  <aixm:Timesheet gml:id="ID_50_1352812184610_8_1">
    <aixm:timeReference>UTC</aixm:timeReference>
    <aixm:day>SAT</aixm:day>
    <aixm:startTime>00:00</aixm:startTime>
    <aixm:endTime>06:00</aixm:endTime>
    <aixm:excluded>YES</aixm:excluded>
    <aixm:extension>
      <adr:TimesheetExtension gml:id="ID_50_1352812184610_8_2">
      <gml:validTime>
        <gml:TimePeriod gml:id="ID_40_1377648348105_81">
          <gml:beginPosition>2012-10-18T00:00:00</gml:beginPosition>
          <gml:endPosition>2014-02-06T00:00:00</gml:endPosition>
        </gml:TimePeriod>
      </gml:validTime>
      </adr:TimesheetExtension>
    </aixm:extension>
  </aixm:Timesheet>
</aixm:timeInterval>
<aixm:timeInterval>
  <aixm:Timesheet gml:id="ID_50_1352812184610_9_1">
    <aixm:timeReference>UTC</aixm:timeReference>
    <aixm:day>SUN</aixm:day>
    <aixm:startTime>00:00</aixm:startTime>
    <aixm:endTime>06:00</aixm:endTime>
    <aixm:excluded>YES</aixm:excluded>
    <aixm:extension>
      <adr:TimesheetExtension gml:id="ID_50_1352812184610_9_2">
        <gml:validTime>
          <gml:TimePeriod gml:id="ID_40_1377648348105_82">
            <gml:beginPosition>2012-10-18T00:00:00</gml:beginPosition>
            <gml:endPosition>2014-02-06T00:00:00</gml:endPosition>
          </gml:TimePeriod>
        </gml:validTime>
      </adr:TimesheetExtension>
    </aixm:extension>
  </aixm:Timesheet>
</aixm:timeInterval>
```

2.2.5.2.13. TrafficVolumeActivation Object (ADR-E)

(1) The time-based definition of when the TrafficVolume is active (see Timesheet Object).

(2) Example:

```
<activation>
  <TrafficVolumeActivation id="ID_705_1423477842671_182">
    <timeInterval>
      <Timesheet id="ID_705_1423477842671_183">
        <timeReference>UTC</timeReference>
        <day>ANY</day>
        <startTime>06:00</startTime>
        <endTime>21:00</endTime>
        <extension>
          <TimesheetExtension id="ID_705_1423477842671_184">
            <validTime>
              <TimePeriod id="ID_705_1423477842671_185">
                <beginPosition>2014-05-29T00:00:00</beginPosition>
                <endPosition indeterminatePosition="unknown"/>
              </TimePeriod>
            </validTime>
          </TimesheetExtension>
        </extension>
      </Timesheet>
    </timeInterval>
  </TrafficVolumeActivation>
</activation>
```

2.2.5.3. Data types

2.2.5.3.1. CodeConditionalRouteType (ADR-E)

(1) The following information is also available from Eurocontrol ATM Lexicon.

(2) A Conditional Route may have more than one category, and those categories may change at specified times:

a) Category One: Permanently Plannable CDR.
CDR_1 routes are available for flight planning during times published in the relevant AIP.

b) Category Two: Non-Permanently Plannable CDR.
CDR_2 routes may be available for flight planning. Flights may only be planned on a CDR_2 in accordance with conditions published daily in the AUP.

c) Category Three - Not Plannable CDR.
CDR_3 routes are not available for flight planning; however, ATC Units may issue tactical clearances on such route segments.

2.2.5.3.2. CodeProcessingIndicatorType (ADR-E)

(1) This enumerator facilitates the detection of the 'kind' of FlightRestriction.

(2) Instead of parsing the whole FlightCondition to find out the 'kind' of FlightRestriction, the CodeProcessingIndicatorType puts you on the right track.

a) AD_CP: This is an Aerodrome DCT or Aerodrome Connecting Points FlightRestriction. This FlightRestriction corresponds to the FlightRestrictions in RAD Appendix 5.

b) FRA_DCT: FRA DCT FlightRestriction. The FlightCondition is either an Airspace or an AirspaceBorder.

c) FPR: Flight Profile FlightRestriction. These kind of FlightRestrictions reflect known agreements between ATCs to transfer flights between them (often formulated into "letters of agreement" (LOAs). They are used by NM to calculate a vertical flight profile which is compliant with these LOAs. They are not used to invalidate flight plans.

d) RAD_DCT: RAD DCT FlightRestriction. This kind of FlightRestriction defines direct flight limitations traversing or crossing the border an Airspace.

e) TFR: Traffic Flow FlightRestriction. This the most commonly found FlightRestriction as defined in the RAD.

f) OTHER:__ADR__AD_FLIGHT_RULE: This is not really a FlightRestriction. It is about IFR/VFR rules that apply on an AirportHeliport. The information is cast into a fake FlightCondition to express the prohibition to depart from or arrive at AirportHeliports that are not equipped with the corresponding IFR hardware.

g) OTHER:__ADR__FLIGHT_PROPERTY_ON_TP: This is not really a FlightRestriction. It is about which kind or airplane effectively can use a Procedure.

2.2.5.4. Miscellania

2.2.5.4.1. gml:pos

(1) A Latitude/Longitude is expressed using the WGE(WGS-84) datum. Latitude is provided first. Both latitude/longitude use a decimal notation instead of minutes/seconds.

(2) In AIXM (because it is based on GML) the datum must be encoded in the srsName attribute as "urn:ogc:def:crs:EPSG::4326".

(3) Example:

```
<gml:pos srsName="urn:ogc:def:crs:EPSG::4326">56.66833333333334 -10.0</gml:pos>
```

This corresponds to 56°40'6"N, 10°W.

2.2.5.4.2. StandardLevel encoding

(1) The StandardLevelColumns and StandardLevelTables are hardcoded in the Route domain.

(2) When defining the RouteAvailability, the levels (AirspaceLayer) will refer to a discreteLevelSeries, which is a reference to a StandardLevelColumn Feature.

(3) The CodeLevelSeriesType is extended with the value OTHER:__ADR__UNIDIRECTIONAL to express that both EVEN and ODD can be used because the RouteSegment is unidirectional.

(4) In total 10 StandardLevelColumns are defined to express the possible combinations of CodeLevelSeriesType and StandardColumnTable.

(5) The following table shows the possible combinations:

	IFR	IFR_RVSM	VFR	VFR_RVSM
EVEN	X	X	X	X
ODD	X	X	X	X
UNIDIRECTIONAL	X	X	-	-

Table 2.5. Possible combinations of CodeLevelSeriesType and StandardColumnTable

(6) Example (definition of ODD/VFR combination):

```
<aixm:StandardLevelColumn gml:id="ID_172_1385510754499_70">
  <gml:identifier codeSpace="urn:uuid:">SLC_ODD_VFR</gml:identifier>
  <aixm:timeSlice>
    <aixm:StandardLevelColumnTimeSlice gml:id="ID_172_1385510754499_71">
      <gml:validTime>
        <gml:TimePeriod gml:id="ID_172_1385510754499_72">
          <gml:beginPosition>1970-01-01T00:00:00</gml:beginPosition>
          <gml:endPosition indeterminatePosition="unknown"/>
        </gml:TimePeriod>
      </gml:validTime>
      <aixm:interpretation>BASELINE</aixm:interpretation>
      <aixm:featureLifetime>
        <gml:TimePeriod gml:id="ID_172_1385510754499_73">
          <gml:beginPosition>1970-01-01T00:00:00</gml:beginPosition>
          <gml:endPosition indeterminatePosition="unknown"/>
        </gml:TimePeriod>
      </aixm:featureLifetime>
      <aixm:series>ODD</aixm:series>
      <aixm:levelTable xlink:href="urn:uuid:SLT_VFR"/>
    </aixm:StandardLevelColumnTimeSlice>
  </aixm:timeSlice>
</aixm:StandardLevelColumn>
```

Chapter 3. Port Types

3.1. AirspaceAvailabilityService Port Type

3.1.1. Overview

3.1.1.1. Intro

(1) The AirspaceAvailabilityService provides services for both querying and modifying the airspace availability. These services are of two kinds:

a) Management and sharing of AUP

b) Access to EAUP

3.1.1.2. AUP Management Details

(1) The management and sharing of AUP cannot be understood outside of the existing operational ASM process, nor the CIAM.

(2) "Airspace Use Plan", contains the decision of an AMC on the temporary allocation of the airspace within its jurisdiction for a specific time period.

(3) The convention for operational users is to make a naming distinction between the first AUP of a chain or baseline AUP commonly referred as the "AUP" and the subsequent AUPs in a chain referred as the UUPs (Updated AUPs).

(4) Due to the equivalence of data structures, in this document, the term AUP refers to both, unless otherwise stated.

(5) The concept of "AUP chain" is defined as the sequence of AUPs for a given day and AMC. It is made of:

a) The AUP baseline of the day

b) The ordered list of its subsequent versions (often called UUPs, but these are also AUPs)
The following rules apply:

a) Any AUP with a validity period comprised in the time span [6:00 day D, 6:00 day D+1 (the next day)] is a member of the AUP-Chain of day D

b) The validity period of the Baseline AUP is always: from 6:00 day D till 6:00 D+1

(6) An AUP has a state at any point in time: DRAFT, READY or RELEASED:

a) An AUP can be created/updated via NOP/B2B in state DRAFT or READY

b) Releasing an AUP is only permitted to the CADF
The following rule applies:

a) AUPs become immutable (cannot be updated anymore) once in RELEASED state

(7) The retrieval (unique) key of an AUP chain (AUPChain type) is the (day, AMC id) pair.

(8) AUPSummary contains its unique identification, its validity period, its state and its last update timestamp, among others.

(9) By definition, a "post-ops AUP chain" is an AUP chain of which the 06:00 AM end is in the past.

(10) In this document, the phrase "CDR update" is synonym to the phrase "CDR opening/closure".

(11) A NIL AUP is a Baseline AUP that does not contain CDR updates, nor RSA allocations and is explicitly flagged as such (AUPSummary.nilAUP set to true).

(12) The requests of the AirspaceAvailabilityService for AUP are:

a) AUPChainRetrievalRequest / AUPChainRetrievalReply

b) AUPRetrievalRequest / AUPRetrievalReply

c) AUPCreationRequest / AUPCreationReply

d) AUPUpdateRequest / AUPUpdateReply

e) AUPValidationRequest / AUPValidationReply

f) AUPDeletionRequest / AUPDeletionReply

g) AUPRSAAllocationExpansionRequest / AUPRSAAllocationExpansionReply

h) AUPServiceConfigurationRequest / AUPServiceConfigurationReply

(13) Airspace data:

a) The AUPs and EAUPs published by and retrieved from NM are encoded using the AIXM 5.1 exchange model (see above).

b) The convention used for Feature identification and referencing within the AirspaceAvailability service differs from the one used for the AirspaceStructure service in the following:

i) it makes use of a SNAPSHOT timeslice for features identification;

ii) it uses the `gml:id` attribute for feature referencing.

c) The actual CDR opening/closures and RSA activations are expressed as a TEMPDELTA timeslice.

d) Even if the Airspace data used by the client to prepare the AUP has a "local" origin, the final AUP sent to the NM must use the UUIDs published by NM otherwise the AUP will be rejected; therefore local and NM data must be correlated by the client

e) The UUIDs sent to NM must belong to the corresponding AIRAC data published by NM.

(14) Cross-AIRAC AUP:

a) An AUP validity period (from D 06:00 to D+1 06:00) may cross AIRAC boundaries

b) When provided to NM a CDR update or RSA allocation crossing the AIRAC switch (i.e. midnight of the last day of the running AIRAC), must be split in two:

i) A CDR update/RSA allocation ending at midnight

ii) and a CDR update/RSA allocation beginning at midnight
The "pre-midnight" part must comply with the running AIRAC data definition; the "post-midnight" part must comply with the next AIRAC data definition

(15) AUP Availability Period:

a) All of the AUP requests (including the AUPChainRetrieval) must be performed within the AUP Availability Period. The AUP Availability Period is defined as the on-going AIRAC, plus, once day 23 of the on-going AIRAC has been reached, and not before, the next AIRAC.

(16) Typical sequence of events:

a) The AUP is prepared in advance by the client (AMC) with his own application

b) Once the time to coordinate comes the AUP must have been created -in NM- in DRAFT status

c) The AUP can be updated while in DRAFT status.

d) Once the AUP is considered ready for publication the AMC updates to READY.

i) If the AUP is a Baseline AUP: once the AUP is in READY the AUP can be updated (in READY or even back to DRAFT) before a cut-off time (cut-off time COT2)

ii) If the AUP is not a Baseline AUP (i.e. UUP): the AUP cannot be updated once in READY

e) Once all AUPs are in READY status, which is enforced by the CADF (cut-off time COT1), CADF takes the responsibility for the AUP from then onwards. In coordination with the clients, CADF modifies AUPs that still require manual intervention and finally publishes all AUPs. Upon publication, the AUP status is changed to RELEASED.

f) Once RELEASED, the AUPs become immutable and the EAUP is published

(17) The operational meaning of the AUP status must be respected.

(18) The following diagram depicts the possible state transitions.

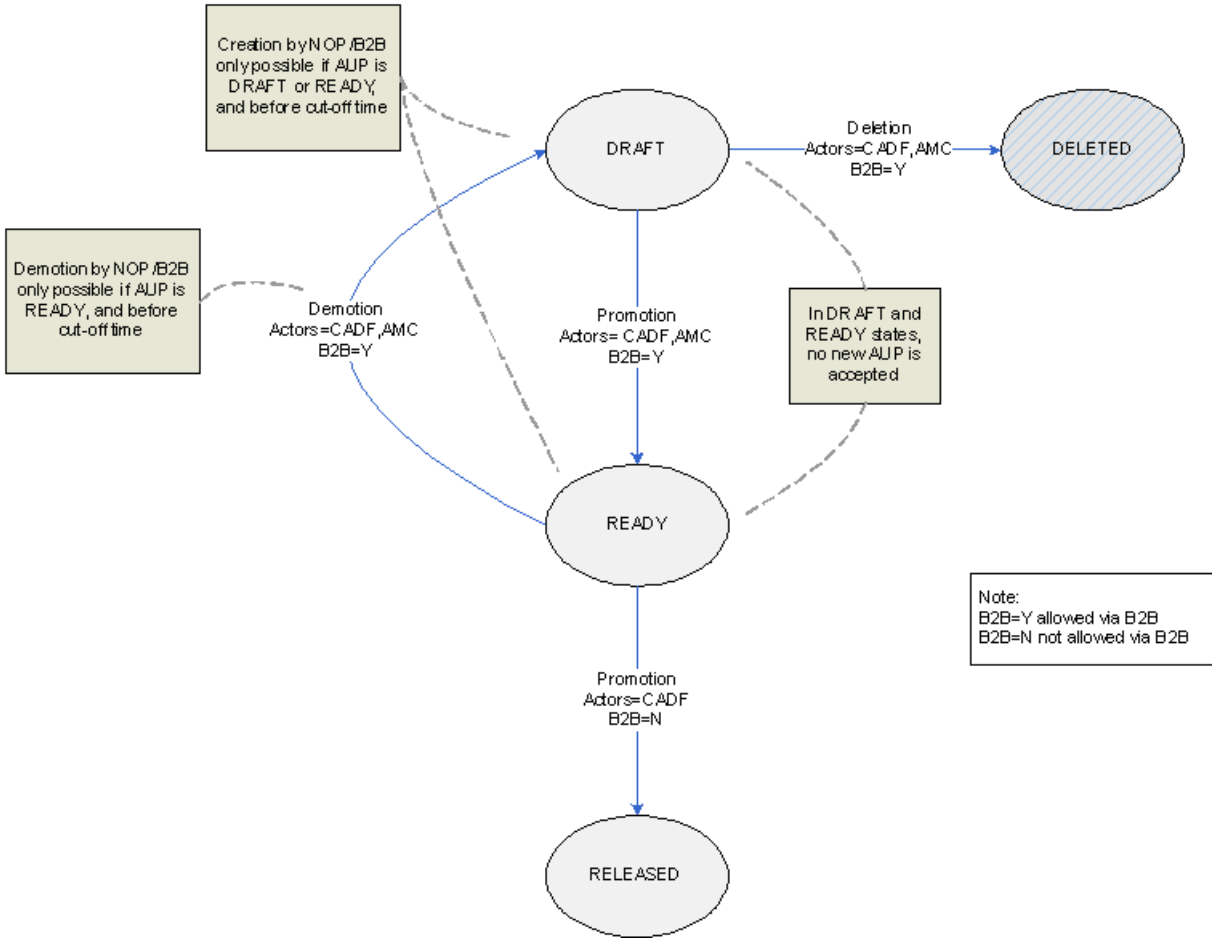

Figure 3.1. AUP Status Transitions

(19) Attention: A NIL AUP MUST be created directly in status READY. This is a legacy constraint.

(20) Status transitions have operational cut-off times agreed in advance between the AMC and the CADF in conformance with the operational procedures.

(21) Creation of an AUP in DRAFT or READY by an AMC is only possible before a cut-off time (COT1), after the cut-off time no creation/modification of AUP is allowed and it is up to the CADF to create at the NM the AUP in coordination with the AMC.

(22) Update of an AUP is also only possible before a cut-off time, but different cut-off times apply depending on the transition and the AUP:

a) If the status transition is from DRAFT to DRAFT COT1 applies. (same for Deletion when the AUP to be deleted is DRAFT)

b) If the status transition is from DRAFT to READY it must be done before COT1 too

c) If the status transition is from READY to READY or READY to DRAFT (same for Deletion when the AUP we want to delete is in READY):

i) For Baseline AUP, the cut-off time (COT2) is not just operational but technical too; it is configured in the NM system in accordance to the latest operational procedures (therefore to be communicated by the CADF to the AMC). The NM system prevents any update after such a cut-off time. In case modifications are required after such a cut-off time the AMC coordinates with the CADF, which updates at the NM premises.

ii) For non-baseline AUP: not applicable, see below.

d) For non-baseline AUP (i.e. UUP), once an AUP is READY, it cannot be modified by the AMC. Therefore it is most convenient for the AMC to work with the DRAFT and only update to READY when sure that the content is finalised (which is the meaning of READY).

(23) Note that as of today COT1 > COT2, which means that there is a period between COT1 and COT2 in which a Baseline AUP can be updated from DRAFT to READY, but if the AUP is already in READY it cannot be modified.

(24) Remark: No specific values for COT1 and COT2 are given as they are expected to change with the introduction of the rolling UUP process.

3.1.1.3. EAUP Access Details

(1) An EAUP is made of the simplified concatenated merged CDR updates and RSA allocations of all RELEASED AUPs for all AMCs.

(2) In a nutshell:

a) The "EAUP chain" is defined as the sequence of EAUPs for a given day. It is made of:

i) The EAUP baseline of the day

ii) The ordered list of its subsequent versions (often called EUUPs, but these are also EAUPs)

b) All EAUPs of an EAUP chain are released EAUPs

c) EAUPs are immutable, i.e. they do not change after release

d) The retrieval (unique) key of an EAUP chain (EAUPChain type) is its day

e) EAUPChain is made of its date and a list of EAUP summaries (EAUPSummary type)

f) EAUPSummary contains its unique identification, its release timestamp and its validity period

g) The service allows for querying the contents of an EAUP (CDR openings and closures, and RSA allocations)

(3) The requests of the AirspaceAvailabilityService for EAUP are:

a) EAUPChainRetrievalRequest / EAUPChainRetrievalReply

b) EAUPCDRRequest / EAUPCDRReply

c) EAUPCDRCompareRequest / EAUPCDRCompareReply

d) EAUPRSARequest / EAUPRSAReply

e) EAUPRSACompareRequest / EAUPRSACompareReply

(4) The replies in the EAUP related services are self-contained, i.e. all the Features are definded in the resulting ADRMessageType.

(5) As an example I will dig into the service retrieveEAUPCDRs.

(6) Example of the call:

```
<?xml version="1.0" encoding="UTF-8"?>
<airspace:EAUPCDRRequest
    xmlns:airspace="eurocontrol/cfmu/b2b/AirspaceServices">
  <endUserId>joe</endUserId>
  <sendTime>2012-09-10 06:53:20</sendTime>
  <eaupId>
    <chainDate>2012-09-19</chainDate>
    <sequenceNumber>1</sequenceNumber>
  </eaupId>
</airspace:EAUPCDRRequest>
```

(7) The reply has the following skeleton:

```
<?xml version="1.0" encoding="UTF-8"?>
<airspace:EAUPCDRReply
    xmlns:airspace="eurocontrol/cfmu/b2b/AirspaceServices"
    xmlns:common="eurocontrol/cfmu/b2b/CommonServices"
    xmlns:adr="http://www.aixm.aero/schema/5.1/extensions/ADR"
    xmlns:gml="http://www.opengis.net/gml/3.2"
    xmlns:aixmmsg="http://www.aixm.aero/schema/5.1/message"
    xmlns:aixm="http://www.aixm.aero/schema/5.1"
    xmlns:xlink="http://www.w3.org/1999/xlink">
  <requestReceptionTime>2012-09-19 15:05:55</requestReceptionTime>
  <requestId>B2B_CUR:2487</requestId>
  <sendTime>2012-09-19 15:05:55</sendTime>
  <status>OK</status>
  <data>
    <cdrOpeningsClosures gml:id="ID_991_1348067155492_96">
      <aixmmsg:hasMember>
        <!-- not expanded here -->
      </aixmmsg:hasMember>
      <aixmmsg:hasMember>
        <!-- not expanded here -->
      </aixmmsg:hasMember>
    </cdrOpeningsClosures>
  </data>
</airspace:EAUPCDRReply>
```

(8) The content is a list of RouteSegment openings and closures. The RouteSegments are not identified by a UUID. Instead they are identified by feature identifying properties in a SNAPSHOT RouteSegmentTimeSlice.

(9) The RouteSegment uses the following identifying properties:

a) start

b) routeFormed

c) end

(10) Example of a RouteSegmentTimeSlice SNAPSHOT:

```
  <aixm:timeSlice>
    <aixm:RouteSegmentTimeSlice gml:id="ID_991_1348067155492_97">
      <gml:validTime>
        <gml:TimeInstant gml:id="ID_991_1348067155492_98">
          <gml:timePosition>2012-09-19T15:05:55</gml:timePosition>
        </gml:TimeInstant>
      </gml:validTime>
      <aixm:interpretation>SNAPSHOT</aixm:interpretation>
      <aixm:start>
        <aixm:EnRouteSegmentPoint gml:id="ID_991_1348067155492_99">
          <aixm:pointChoice_fixDesignatedPoint
              xlink:href="urn:uuid:#xpointer(id('ID_991_1348067155492_4'))"/>
        </aixm:EnRouteSegmentPoint>
      </aixm:start>
      <aixm:routeFormed
          xlink:href="urn:uuid:#xpointer(id('ID_991_1348067155492_1'))"/>
      <aixm:end>
        <aixm:EnRouteSegmentPoint gml:id="ID_991_1348067155492_100">
          <aixm:pointChoice_fixDesignatedPoint
              xlink:href="urn:uuid:#xpointer(id('ID_991_1348067155492_5'))"/>
        </aixm:EnRouteSegmentPoint>
      </aixm:end>
    </aixm:RouteSegmentTimeSlice>
  </aixm:timeSlice>
```

(11) Notice that each feature identifying property uses an xpointer reference to a gml:id.

(12) In the above example the start of the RouteSegment refers to gml:id="ID_991_1348067155492_4". Therefore further in the cdrOpeningsClosure you will find:

```
  <aixmmsg:hasMember>
    <aixm:DesignatedPoint gml:id="ID_991_1348067155492_4">
      <aixm:timeSlice>
        <aixm:DesignatedPointTimeSlice gml:id="ID_991_1348067155492_488">
          <gml:validTime>
            <gml:TimeInstant gml:id="ID_991_1348067155492_489">
              <gml:timePosition>2012-09-19T15:05:55</gml:timePosition>
            </gml:TimeInstant>
          </gml:validTime>
          <aixm:interpretation>SNAPSHOT</aixm:interpretation>
          <aixm:designator>REMBA</aixm:designator>
          <aixm:type>ICAO</aixm:type>
        </aixm:DesignatedPointTimeSlice>
      </aixm:timeSlice>
    </aixm:DesignatedPoint>
  </aixmmsg:hasMember>
```

(13) The DesignatedPoint feature is in this case identified using DesignatedPointTimeSlice SNAPSHOT. More in particular the designator brings the answer as to which point it concerned.

(14) Similarly we find for the referred Route:

```
  <aixmmsg:hasMember>
    <aixm:Route gml:id="ID_991_1348067155492_1">
      <aixm:timeSlice>
        <aixm:RouteTimeSlice gml:id="ID_991_1348067155492_404">
          <gml:validTime>
            <gml:TimeInstant gml:id="ID_991_1348067155492_405">
              <gml:timePosition>2012-09-19T15:05:55</gml:timePosition>
            </gml:TimeInstant>
          </gml:validTime>
          <aixm:interpretation>SNAPSHOT</aixm:interpretation>
          <aixm:designatorSecondLetter>M</aixm:designatorSecondLetter>
          <aixm:designatorNumber>624</aixm:designatorNumber>
        </aixm:RouteTimeSlice>
      </aixm:timeSlice>
    </aixm:Route>
  </aixmmsg:hasMember>
```

(15) So the RouteTimeSlice SNAPSHOT exposes the designator (through designator[Prefix|SecondLetter|Number]).

(16) In order to define the opening or closure, a RouteSegmentTimeSlice TEMPDELTA is needed:

```
  <aixm:RouteSegmentTimeSlice gml:id="ID_991_1348067155492_101">
    <gml:validTime>
      <gml:TimePeriod gml:id="ID_991_1348067155492_102">
        <gml:beginPosition>2012-09-19T09:45:00</gml:beginPosition>
        <gml:endPosition>2012-09-19T12:15:00</gml:endPosition>
      </gml:TimePeriod>
    </gml:validTime>
    <aixm:interpretation>TEMPDELTA</aixm:interpretation>
    <aixm:availability>
      <aixm:RouteAvailability gml:id="ID_991_1348067155492_103">
        <aixm:levels>
          <aixm:AirspaceLayer gml:id="ID_991_1348067155492_104">
            <aixm:upperLimit>175</aixm:upperLimit>
            <aixm:upperLimitReference>MSL</aixm:upperLimitReference>
            <aixm:lowerLimit>145</aixm:lowerLimit>
            <aixm:lowerLimitReference>MSL</aixm:lowerLimitReference>
          </aixm:AirspaceLayer>
        </aixm:levels>
        <aixm:extension>
          <adr:RouteAvailabilityExtension gml:id="ID_991_1348067155492_105">
            <adr:conditionalRouteType>CDR_1</adr:conditionalRouteType>
            <adr:hostAirspace
                xlink:href="urn:uuid:#xpointer(id('ID_991_1348067155492_2'))"/>
          </adr:RouteAvailabilityExtension>
        </aixm:extension>
      </aixm:RouteAvailability>
    </aixm:availability>
  </aixm:RouteSegmentTimeSlice>
```

(17) This xml snippet expresses that the RouteSegment will be (CDR_1) closed between 2012-09-19 09:45 and 2012-09-19 12:15 between flightlevels F145-F175. The hostAirspace is the Airspace is which the RouteSegment is located.

(18) The Airspace is also defined further in the file by an AirpaceTimeSlice SNAPSHOT:

```
  <aixm:Airspace gml:id="ID_991_1348067155492_2">
    <aixm:timeSlice>
      <aixm:AirspaceTimeSlice gml:id="ID_991_1348067155492_510">
```

```
        <gml:validTime>
          <gml:TimeInstant gml:id="ID_991_1348067155492_511">
            <gml:timePosition>2012-09-19T15:05:55</gml:timePosition>
          </gml:TimeInstant>
        </gml:validTime>
        <aixm:interpretation>SNAPSHOT</aixm:interpretation>
        <aixm:type>FIR</aixm:type>
        <aixm:designator>EBBU</aixm:designator>
        <aixm:designatorICAO>YES</aixm:designatorICAO>
      </aixm:AirspaceTimeSlice>
    </aixm:timeSlice>
  </aixm:Airspace>
```

3.1.2. AUP Chain Retrieval

3.1.2.1. SOAP

(1) The associated SOAP operation is:

```
AUPChainRetrievalReply retrieveAUPChain(
    AUPChainRetrievalRequest request
    )
```

3.1.2.2. `AUPChainRetrievalRequest`

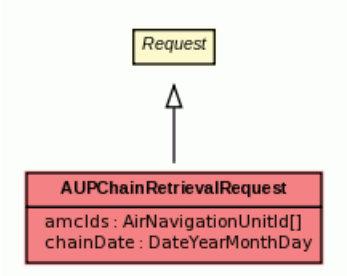

Figure 3.2. `AUPChainRetrievalRequest` Class Diagram

(1) Request to retrieve one or more AUP chains from its date (i.e. from the release date of its AUP baseline) and one or more owning AMCs.

(2) Can be performed at any time (provided the service is available).

(3) Client applications must take into account that post-ops AUP chains are immutable: they will not gain or lose AUPs, and the AUPs they contain will not be modified anymore. Consequently, NM requires the client applications to avoid retrieving the same post-ops AUP chains repeatedly.

(4) Regarding mutable AUP chains (i.e. tactical and pre-tactical), NM requires client applications not to poll the service with high frequency, i.e. certainly not more than every minute.

(5) Inherits from: `Request`

(6) Attributes:

a) **AirNavigationUnitId[] amcIds** *(Optional)*
The ANU ids of the AMCs of which the AUP chain is requested. Default is all.
Constraint: Size must be comprised between 0 and ∞.

b) **DateYearMonthDay chainDate** *(Mandatory)*
The date of the requested AUP chain.

3.1.2.3. AUPChainRetrievalReply

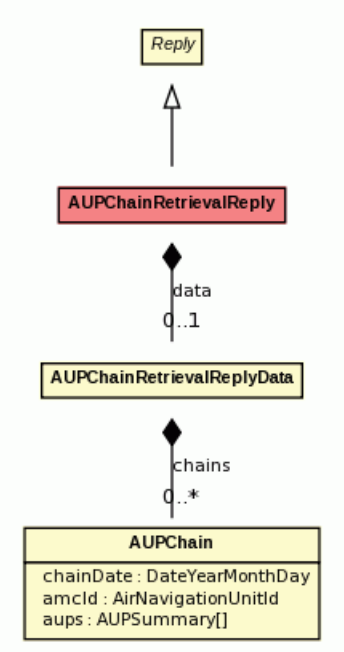

Figure 3.3. AUPChainRetrievalReply Class Diagram

(1) Reply returned in response to AUPChainRetrievalRequest

(2) Inherits from: Reply

(3) Attributes:

a) **AUPChain[] chains** *(Mandatory)*
The retrieved AUP chains. The array can be empty.
REMARK: We return OBJECT_NOT_FOUND if the AUPChain has never been created in the NM system, being for the AUPChain of today, or in 3 months. Note that the situation is slightly different for a "past" AUPChain: it is immutable (won't change anymore) so that if at the end of the day it contains no AUP we know for sure that it is empty and the NM system creates it empty. We then return an AUPChain with OK status and an empty AUP/UUP list.
Constraint: Size must be comprised between 0 and ∞.

3.1.3. AUP Retrieval

3.1.3.1. SOAP

(1) The associated SOAP operation is:

```
AUPRetrievalReply retrieveAUP(
    AUPRetrievalRequest request
    )
```

3.1.3.2. `AUPRetrievalRequest`

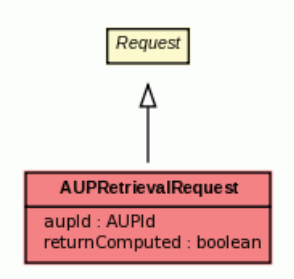

Figure 3.4. `AUPRetrievalRequest` Class Diagram

(1) Request to retrieve an AUP from its unique id.

(2) Can be performed at any time (provided the service is available).

(3) Client applications must take into account that RELEASED AUPs are immutable: they will not be modified anymore. Consequently, NM requires the client applications to avoid retrieving the same RELEASED AUPs repeatedly.

(4) Regarding non-RELEASED AUPs, NM requires client applications not to poll the service with high frequency, i.e. certainly not more than every 10 minutes.

(5) Inherits from: `Request`

(6) Attributes:

a) **`AUPId` `aupId`** *(Mandatory)*
Unique id of the requested AUP: found in AUPSummary.aupId.

b) **`boolean` `returnComputed`** *(Optional)*
Specifies if computed AUP entries are to be returned in addition to manual AUP entries, which are always returned as part of an AUP. False by default.

3.1.3.3. `AUPRetrievalReply`

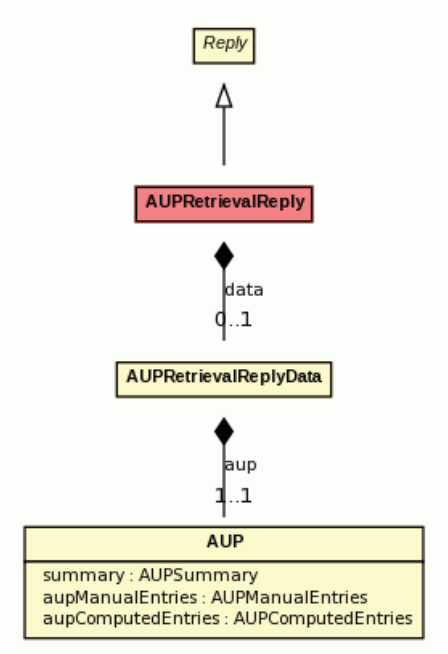

Figure 3.5. `AUPRetrievalReply` Class Diagram

(1) Reply returned in response to AUPRetrievalRequest.

(2) Inherits from: `Reply`

(3) Attributes:

a) **`AUP` `aup`** *(Mandatory)*
The retrieved AUP. If returnComputed is true in the request, both manual and computed AUP entries are returned.

3.1.4. AUP Creation

3.1.4.1. SOAP

(1) The associated SOAP operation is:

```
AUPCreationReply createAUP(
    AUPCreationRequest request
    )
```

3.1.4.2. `AUPCreationRequest`

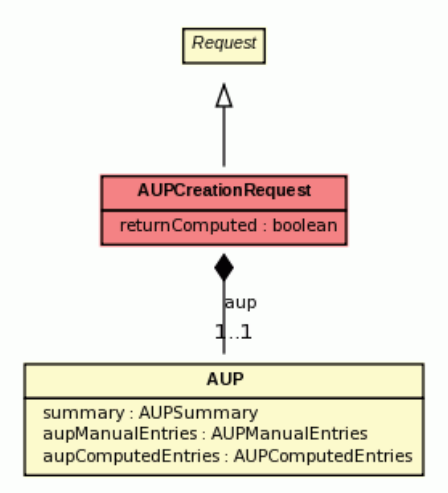

Figure 3.6. `AUPCreationRequest` Class Diagram

(1) Request to validate a new AUP and, on success, to create it.

(2) Via NOP/B2B, an AUP can only be created by an AMC, and is thereby owned by the AMC: the AUP can then be updated by a user (certificate) associated to that AMC only.

(3) This service is constrained in terms of timing/process. See AUP Status Transitions.

(4) Inherits from: `Request`

(5) Attributes:

a) **`AUP aup`** *(Mandatory)*
An AUP containing manual AUP entries only, i.e. its aupComputedEntries must be null.

b) **`boolean returnComputed`** *(Optional)*
Specifies if computed AUP entries are to be returned in addition to manual AUP entries, which are always returned as part of an AUP. False by default.

3.1.4.3. `AUPCreationReply`

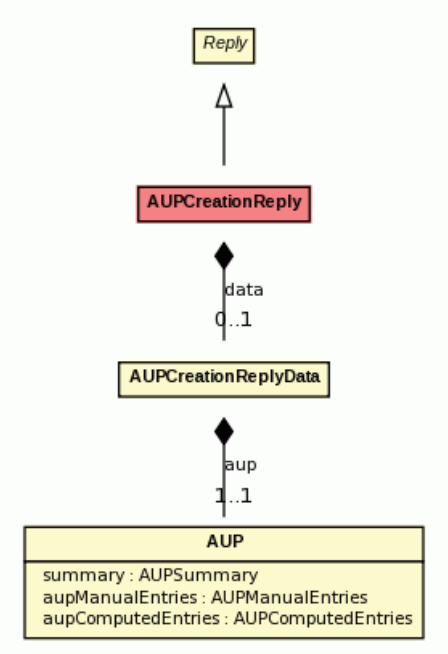

Figure 3.7. `AUPCreationReply` Class Diagram

(1) Reply returned in response to AUPCreationRequest.

(2) Inherits from: `Reply`

(3) Attributes:

a) **`AUP` `aup`** *(Mandatory)*
The created AUP. If returnComputed is true in the request, both manual and computed AUP entries are returned.

3.1.5. AUP Update

3.1.5.1. SOAP

(1) The associated SOAP operation is:

```
AUPUpdateReply updateAUP(
    AUPUpdateRequest request
    )
```

3.1.5.2. `AUPUpdateRequest`

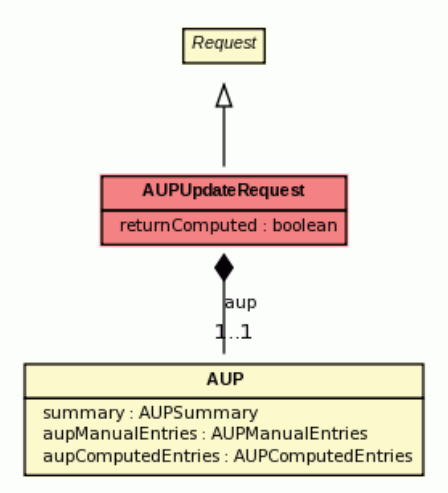

Figure 3.8. `AUPUpdateRequest` Class Diagram

(1) Request to validate a new AUP and, on success, to update it.

(2) Via NOP/B2B, an AUP can only be updated by its owning AMC.

(3) This service is constrained in terms of timing/process. See AUP Status Transitions

(4) Note that if the intention of the client is simply to update the status of a previously created AUP, the original AUP must be provided again.

(5) AUPSummary.lastUpdate serves in the concurrency control mechanism: in order to overwrite a previously saved AUP in DRAFT or READY state, the provided AUPSummary.lastUpdate must match the AUPSummary.lastUpdate of the AUP version being updated. Hence, when updating an AUP, the caller must pass the lastUpdate of the previous AUP version that he knows.

(6) Inherits from: `Request`

(7) Attributes:

a) **`AUP aup`** *(Mandatory)*
The updated AUP, containing manual AUP entries only, i.e. its aupComputedEntries must be null.

b) **`boolean returnComputed`** *(Optional)*
Specifies if computed AUP entries are to be returned in addition to manual AUP entries, which are always returned as part of an AUP. False by default.

3.1.5.3. `AUPUpdateReply`

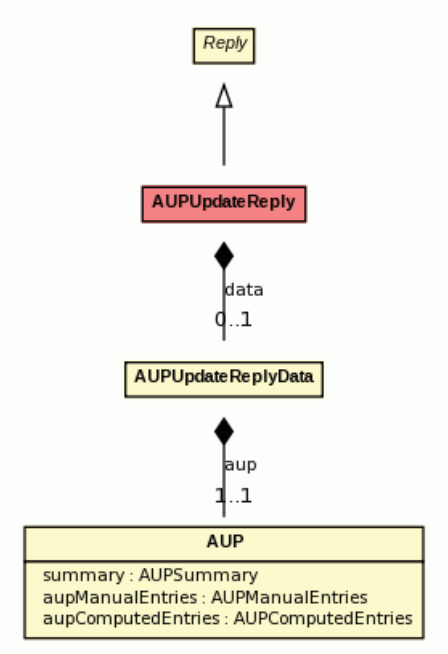

Figure 3.9. `AUPUpdateReply` Class Diagram

(1) Reply returned in response to AUPUpdateRequest.

(2) Inherits from: `Reply`

(3) Attributes:

a) **`AUP` `aup`** *(Mandatory)*
In case of update success, the saved AUP. If returnComputed is true in the request, both manual and computed AUP entries are returned.

3.1.6. AUP Validation

3.1.6.1. SOAP

(1) The associated SOAP operation is:

```
AUPValidationReply validateAUP(
    AUPValidationRequest request
    )
```

3.1.6.2. `AUPValidationRequest`

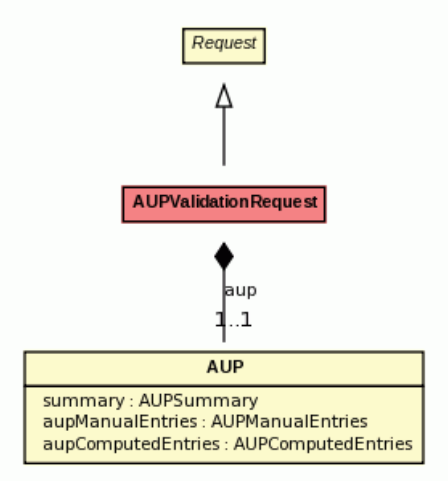

Figure 3.10. `AUPValidationRequest` Class Diagram

(1) Request to validate an AUP.

(2) Can be performed at any time (provided the service is available).

(3) No transaction takes place: the AUP is neither created or updated. The validation service is meant for the customer to validate an AUP at any time, e.g. to work on an AUP prior to persisting it within the NM system.

(4) Inherits from: `Request`

(5) Attributes:

a) **`AUP`** **`aup`** *(Mandatory)*
The AUP to be validated, containing manual AUP entries only, i.e. its aupComputedEntries must be null.

3.1.6.3. `AUPValidationReply`

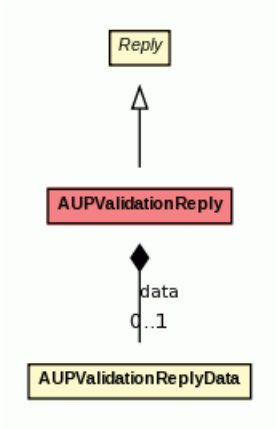

Figure 3.11. `AUPValidationReply` Class Diagram

(1) Reply returned in response to AUPValidationRequest.

(2) Inherits from: `Reply`

3.1.7. AUP Deletion

3.1.7.1. SOAP

(1) The associated SOAP operation is:

```
AUPDeletionReply deleteAUP(
    AUPDeletionRequest request
    )
```

3.1.7.2. `AUPDeletionRequest`

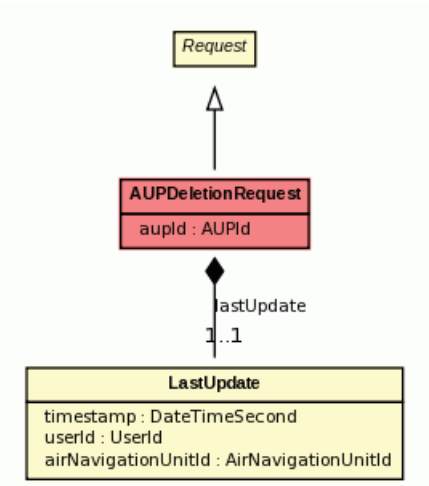

Figure 3.12. `AUPDeletionRequest` Class Diagram

(1) Request to delete an existing AUP.

(2) Via NOP/B2B, an AUP can only be deleted by the AMC owning the AUP.

(3) Deleting an AUP can only be done when updating is possible (See AUP Status Transitions).

(4) Inherits from: `Request`

(5) Attributes:

a) **`AUPId` `aupId`** *(Mandatory)*
Id of the AUP to be deleted.

b) **`LastUpdate` `lastUpdate`** *(Mandatory)*
See AUPUpdateRequest.lastUpdate.

3.1.7.3. `AUPDeletionReply`

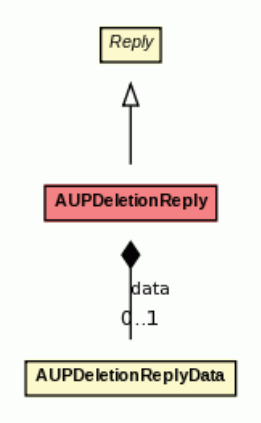

Figure 3.13. `AUPDeletionReply` Class Diagram

(1) Reply returned in response to AUPDeletionRequest.

(2) Inherits from: `Reply`

3.1.8. AUP RSA Allocation Expansion

3.1.8.1. SOAP

(1) The associated SOAP operation is:

```
AUPRSAAllocationExpansionReply expandRSAAllocations(
    AUPRSAAllocationExpansionRequest request
    )
```

3.1.8.2. `AUPRSAAllocationExpansionRequest`

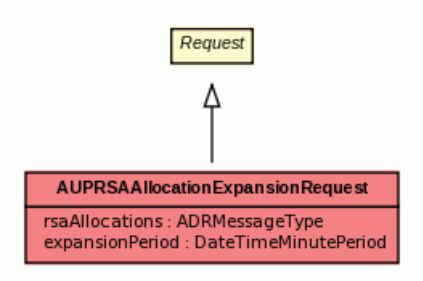

Figure 3.14. `AUPRSAAllocationExpansionRequest` Class Diagram

(1) Request to obtain the following expansion.

(2) Can be performed at any time (provided the service is available).

(3) The expansion algorithm or simply "expansion" computes a list of CDR opening/closures based on:

(4) a) A provided list of RSA allocations, and

b) The pre-defined RSA allocations as stored in NM, and

c) The pre-defined relationships between RSA and CDRs (is-nearby, is-not-affected, etc) as stored in NM

d) A period for which the calculation on the affected routes will done

(5) The output list of CDR openings/closures is labelled implicit to distinguish it from CDR openings and closures managed by the customer.

(6) This request does not imply any update transaction within the NM system.

(7) It is up to the client to extract from the returned CDR updates those of interest to him, and include those in the AUP to be created/updated.

(8) Remark: CHMI users can select the computed CDR updates of interest while creating an AUP. This results in the automatic inclusion of the implicitCDRs CDR updates in the AUP upon saving. The implicitCDRs list of CDR updates is readable by B2B users per AUP. However, B2B users will never generate an AUP with a separate list of implicitCDRs, i.e. all CDR updates in an AUP from a B2B user are always considered explicit.

(9) Inherits from: `Request`

(10) Attributes:

a) **`ADRMessageType rsaAllocations`** *(Optional)*
List of explicit RSA allocations input to the expansion algorithm.

b) **DateTimeMinutePeriod expansionPeriod** *(Mandatory)*
Period of time input to the expansion algorithm.
Constraint: See INVALID_EXPANSION_PERIOD

(11) Constraint:

a)

Name	INVALID_EXPANSION_PERIOD
Attribute	expansionPeriod
Description	Period must be greater or equal to 1 second.

3.1.8.3. AUPRSAAllocationExpansionReply

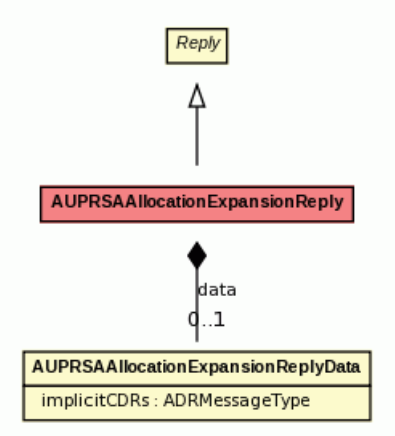

Figure 3.15. AUPRSAAllocationExpansionReply Class Diagram

(1) Reply returned in response to AUPRSAAllocationExpansionRequest.

(2) Inherits from: Reply

(3) Attributes:

a) **ADRMessageType implicitCDRs** *(Optional)*
The list of implicit CDRs computed by applying the expansion algorithm.

3.1.9. AUP Service Configuration

3.1.9.1. SOAP

(1) The associated SOAP operation is:

```
AUPServiceConfigurationReply getAUPServiceConfiguration(
    AUPServiceConfigurationRequest request
    )
```

3.1.9.2. `AUPServiceConfigurationRequest`

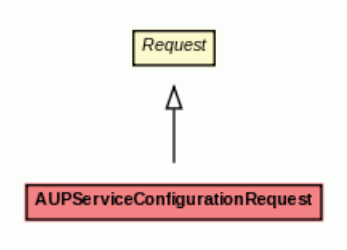

Figure 3.16. `AUPServiceConfigurationRequest` Class Diagram

(1) Request to obtain the dynamic part of the AUP service configuration.

(2) Can be performed at any time (provided the service is available).

(3) The AUP service configuration data is provided a long time prior to its applicability, hence NM requires the client applications not to retrieve it with high frequency, certainly not more than once every 10 minutes.

(4) Inherits from: `Request`

3.1.9.3. `AUPServiceConfigurationReply`

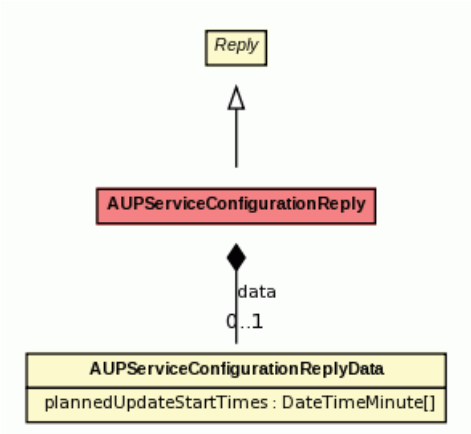

Figure 3.17. `AUPServiceConfigurationReply` Class Diagram

(1) Reply returned in response to AUPServiceConfigurationRequest.

(2) Inherits from: `Reply`

(3) Attributes:

a) **`DateTimeMinute[] plannedUpdateStartTimes`** *(Mandatory)*
Next planned update (UUP) start times. At the moment, these are possibly:

i) One planned update start time for the current AUP chain

ii) One planned update start time for the next AUP chain
Constraint: Size must be comprised between 0 and ∞.

3.1.10. EAUP Chain Retrieval

3.1.10.1. SOAP

(1) The associated SOAP operation is:

```
EAUPChainRetrievalReply retrieveEAUPChain(
    EAUPChainRetrievalRequest request
    )
```

3.1.10.2. `EAUPChainRetrievalRequest`

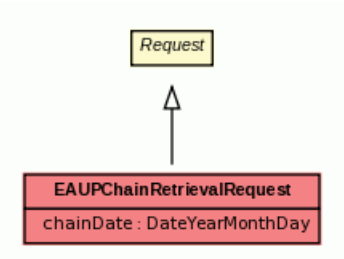

Figure 3.18. `EAUPChainRetrievalRequest` Class Diagram

(1) Request to retrieve an EAUP chain from its date, i.e. from the release date of its EAUP baseline.

(2) Customers must take into account that post-ops (i.e. post-tactical) released EAUP chains are immutable: they will not gain or lose EAUPs, and the EAUPs they contain will not be modified anymore. Consequently, NM requires its customers to undertake their best effort to avoid repeatedly retrieving the same post-ops EAUP chain.

(3) Given that some hours always elapse between two successive EAUP releases, NM requires its customers not to poll the service with high frequency, i.e. certainly not more than every minute, a lower frequency being preferred.

(4) Inherits from: `Request`

(5) Attributes:

a) **`DateYearMonthDay` `chainDate`** *(Mandatory)*
The date of the requested EAUP chain.
Constraint: See `INVALID_CHAIN_DATE`

(6) Constraint:

a)

Name	`INVALID_CHAIN_DATE`
Attribute	`chainDate`

Description	Valid chain dates are: i) D-1 (pre-tactical, tomorrow) ii) D (tactical, today) iii) [D + 1 (yesterday), D +15 months] (post-ops)

3.1.10.3. `EAUPChainRetrievalReply`

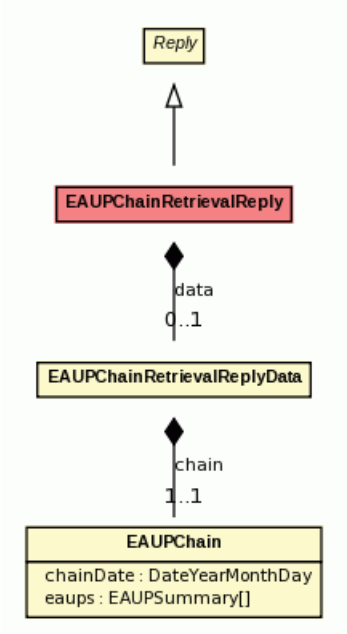

Figure 3.19. `EAUPChainRetrievalReply` Class Diagram

(1) Reply returned in response to EAUPChainRetrievalRequest.

(2) The returned EAUPChain contains EAUP summaries, each containing among others the EAUP identification to be used subsequently to retrieve a complete EAUP or to query its contents.

(3) Inherits from: `Reply`

(4) Attributes:

a) **`EAUPChain chain`** *(Mandatory)*
The retrieved EAUP chain.

3.1.11. EAUP CDR Retrieval

3.1.11.1. SOAP

(1) The associated SOAP operation is:

```
EAUPCDRReply retrieveEAUPCDRs(
    EAUPCDRRequest request
    )
```

3.1.11.2. `EAUPCDRRequest`

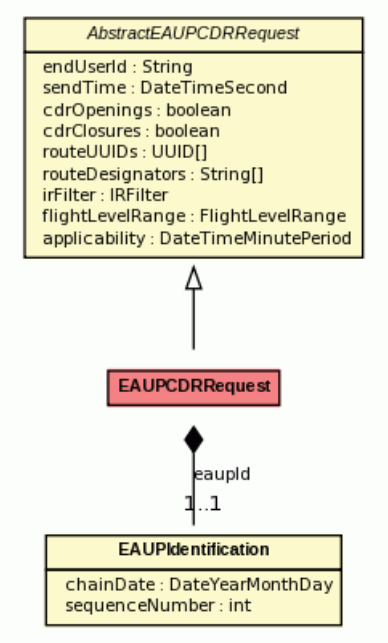

Figure 3.20. `EAUPCDRRequest` Class Diagram

(1) Used to retrieve the CDR openings and/or closures within a given EAUP, while possibly applying a filter on the returned result set, as described in AbstractEAUPCDRRequest, from which this request inherits.

(2) The queried EAUP is identified using the EAUPIdentification from the EAUPSummary returned as part of an EAUPChain.

(3) Inherits from: `AbstractEAUPCDRRequest`

(4) Attributes:

a) **`EAUPIdentification` `eaupId`** *(Mandatory)*
The identification of the EAUP, extracted (and left unchanged) from an EAUPSummary. If no other attribute is specified in this request, all the CDR openings and closures of the EAUP are returned. Mandatory.

3.1.11.3. `EAUPCDRReply`

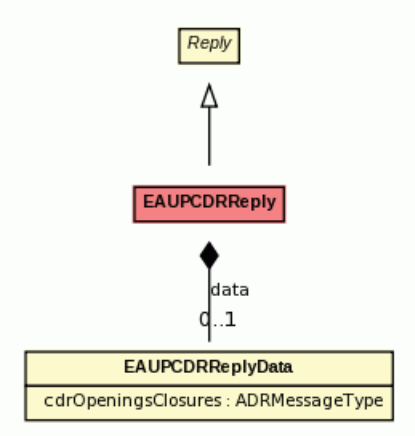

Figure 3.21. `EAUPCDRReply` Class Diagram

(1) Reply returned in response to EAUPCDRRequest.

(2) Inherits from: `Reply`

(3) Attributes:

a) **`ADRMessageType` `cdrOpeningsClosures`** *(Optional)*
The list of CDR openings and closures matching the request. Empty if no CDR opening/closure matches the request.

3.1.12. EAUP CDR Comparison

3.1.12.1. SOAP

(1) The associated SOAP operation is:

```
EAUPCDRCompareReply compareEAUPCDRs(
    EAUPCDRCompareRequest request
    )
```

3.1.12.2. `EAUPCDRCompareRequest`

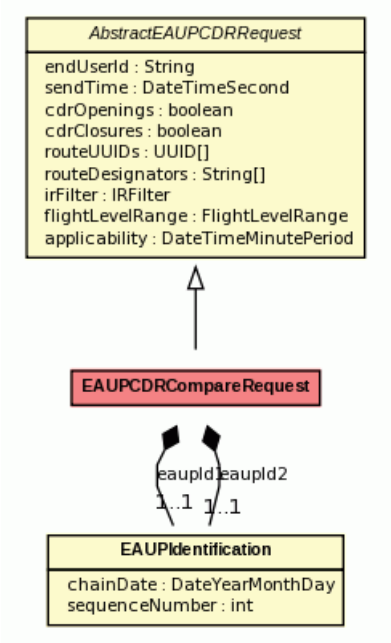

Figure 3.22. `EAUPCDRCompareRequest` Class Diagram

(1) Used to retrieve the CDR openings and closures that the two given EAUPs have in common, and those that are in one of these EAUPs only, while possibly applying a filter on the returned result set, as described in AbstractEAUPCDRRequest.

(2) The queried EAUPs are identified using the EAUPIdentification from the EAUPSummary returned as part of an EAUPChain.

(3) Inherits from: `AbstractEAUPCDRRequest`

(4) Attributes:

a) **`EAUPIdentification` `eaupId1`** *(Mandatory)*
The identification of the first EAUP, extracted (and left unchanged) from an EAUPSummary.
Constraint: See `EAUP_IDS_CANNOT_BE_THE_SAME`

b) **`EAUPIdentification` `eaupId2`** *(Mandatory)*
The identification of the second EAUP, extracted (and left unchanged) from an EAUPSummary.
Constraint: See `EAUP_IDS_CANNOT_BE_THE_SAME`

(5) Constraint:

a)

Name	`EAUP_IDS_CANNOT_BE_THE_SAME`
Attributes	`eaupId1`, `eaupId2`

Description	`eaupId1` and `eaupId2` cannot be the same.

3.1.12.3. `EAUPCDRCompareReply`

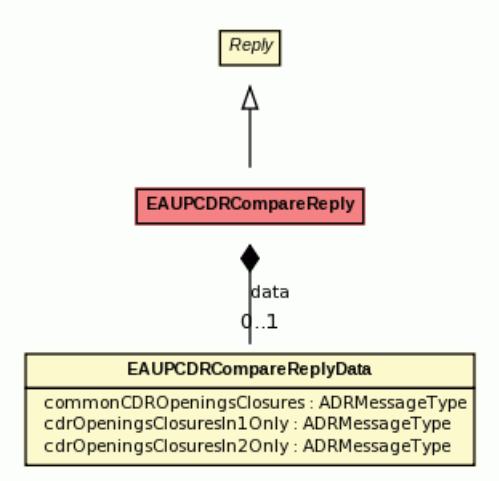

Figure 3.23. `EAUPCDRCompareReply` Class Diagram

(1) Reply returned in response to EAUPCDRCompareRequest.

(2) The three lists below are mandatory, i.e. cannot be null, but are left empty if no matching CDR openings or closures were found. The only circumstances where the lists are left null are those corresponding to request failures, as described in the CommonServices NOP/B2B Reference Manual.

(3) Inherits from: `Reply`

(4) Attributes:

a) **`ADRMessageType commonCDROpeningsClosures`** *(Optional)*
The list of CDR openings and closures matching the request and that are common to the two requested EAUPs.

b) **`ADRMessageType cdrOpeningsClosuresIn1Only`** *(Optional)*
The list of CDR openings and closures matching the request and that only appear in the EAUP identified by eaupId1.

c) **`ADRMessageType cdrOpeningsClosuresIn2Only`** *(Optional)*
The list of CDR openings and closures matching the request and that only appear in the EAUP identified by eaupId2.

3.1.13. EAUP RSA Retrieval

3.1.13.1. SOAP

(1) The associated SOAP operation is:

```
EAUPRSAReply retrieveEAUPRSAs(
    EAUPRSARequest request
    )
```

3.1.13.2. `EAUPRSARequest`

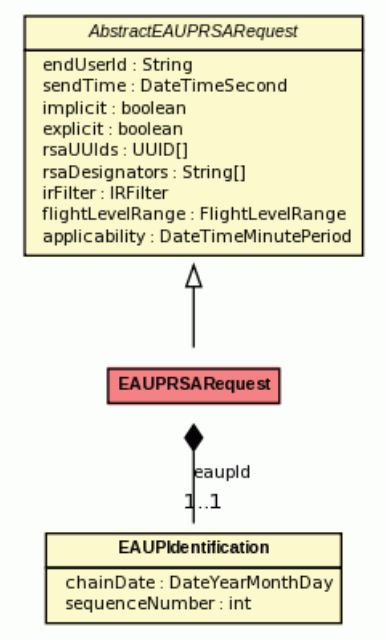

Figure 3.24. `EAUPRSARequest` Class Diagram

(1) Used to retrieve the RSA allocations within a given EAUP, while possibly applying a filter on the returned result set, as described in AbstractEAUPRSARequest, from which this request inherits.

(2) The queried EAUP is identified using the EAUPIdentification from the EAUPSummary returned as part of an EAUPChain.

(3) Inherits from: `AbstractEAUPRSARequest`

(4) Attributes:

a) **`EAUPIdentification` `eaupId`** *(Mandatory)*
The identification of the EAUP, extracted (and left unchanged) from an EAUPSummary. If no other attribute is specified in this request, all the RSA allocations of the EAUP are returned.

3.1.13.3. `EAUPRSAReply`

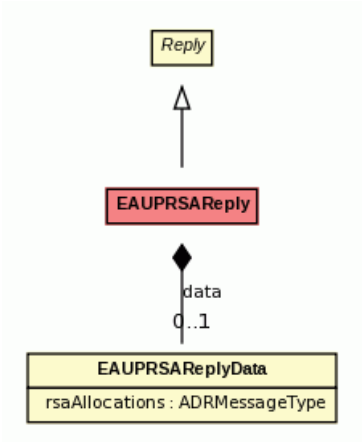

Figure 3.25. `EAUPRSAReply` Class Diagram

(1) Reply returned in response to EAUPRSARequest.

(2) Inherits from: `Reply`

(3) Attributes:

a) **`ADRMessageType`** **`rsaAllocations`** *(Optional)*
The list of RSA allocations matching the request.

3.1.14. EAUP RSA Comparison

3.1.14.1. SOAP

(1) The associated SOAP operation is:

```
EAUPRSACompareReply compareEAUPRSAs(
    EAUPRSACompareRequest request
    )
```

3.1.14.2. EAUPRSACompareRequest

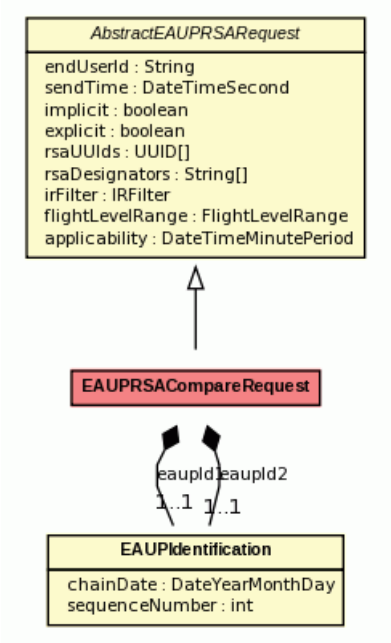

Figure 3.26. EAUPRSACompareRequest Class Diagram

(1) Used to retrieve the RSA allocations that the two given EAUPs have in common, and those that are in one of these EAUPs only, while possibly applying a filter on the returned result set, as described in AbstractEAUPRSARequest.

(2) The queried EAUPs are identified using the EAUPIdentification from the EAUPSummary returned as part of an EAUPChain.

(3) Inherits from: `AbstractEAUPRSARequest`

(4) Attributes:

a) **`EAUPIdentification` eaupId1** *(Mandatory)*
The identification of the first EAUP, extracted (and left unchanged) from an EAUPSummary.
Constraint: See `EAUP_IDS_CANNOT_BE_THE_SAME`

b) **`EAUPIdentification` eaupId2** *(Mandatory)*
The identification of the second EAUP, extracted (and left unchanged) from an EAUPSummary.
Constraint: See `EAUP_IDS_CANNOT_BE_THE_SAME`

(5) Constraint:

a)

Name	EAUP_IDS_CANNOT_BE_THE_SAME
Attributes	`eaupId1`, `eaupId2`

Description	`eaupId1` and `eaupId2` cannot be the same.

3.1.14.3. `EAUPRSACompareReply`

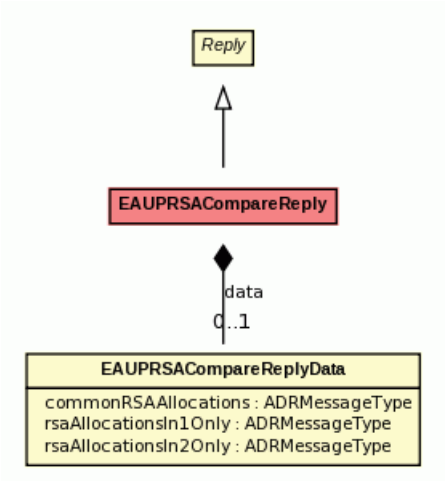

Figure 3.27. `EAUPRSACompareReply` Class Diagram

(1) Reply returned in response to EAUPRSACompareRequest.

(2) The three lists below are mandatory, i.e. cannot be null, but are left empty if no matching RSA allocations were found. The only circumstances where the lists are left null are those corresponding to request failures, as described in the CommonServices NOP/B2B Reference Manual.

(3) Inherits from: `Reply`

(4) Attributes:

a) **`ADRMessageType` `commonRSAAllocations`** *(Optional)*
The list of RSA allocations matching the request and that are common to the two requested EAUPs.

b) **`ADRMessageType` `rsaAllocationsIn1Only`** *(Optional)*
The list of RSA allocations matching the request and that only appear in the EAUP identified by eaupId1.

c) **`ADRMessageType` `rsaAllocationsIn2Only`** *(Optional)*
The list of RSA allocations matching the request and that only appear in the EAUP identified by eaupId2.

3.1.15. Manageable Routes For AMC

3.1.15.1. SOAP

(1) The associated SOAP operation is:

```
AUPGetManageableRoutesForAMCReply getManageableRoutesForAMC(
    AUPGetManageableRoutesForAMCRequest request
    )
```

3.1.15.2. AUPGetManageableRoutesForAMCRequest

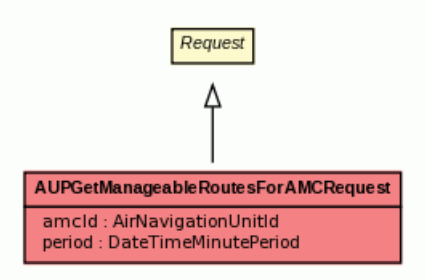

Figure 3.28. AUPGetManageableRoutesForAMCRequest Class Diagram

(1) An AMC is responsible for the management of Elementary and Composed Manageable Airspaces.

(2) This service returns the potential openable and closeable Routes.

(3) Inherits from: Request

(4) Attributes:

a) **AirNavigationUnitId amcId** *(Mandatory)*
The id to identify the AMC.

b) **DateTimeMinutePeriod period** *(Mandatory)*
The period to consider. Typically the validity of an AUP or a part of that validity period.

3.1.15.3. AUPGetManageableRoutesForAMCReply

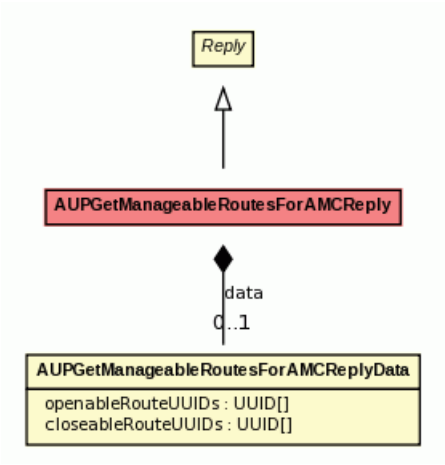

Figure 3.29. AUPGetManageableRoutesForAMCReply Class Diagram

(1) Reply returned in response to AUPGetManageableRoutesForAMCRequest.

(2) Inherits from: `Reply`

(3) Attributes:

a) **`UUID[] openableRouteUUIDs`** *(Mandatory)*
Routes which contain CDRs to open.
Constraint: Size must be comprised between 0 and ∞.

b) **`UUID[] closeableRouteUUIDs`** *(Mandatory)*
Routes which contain CDRs to close
Constraint: Size must be comprised between 0 and ∞.

3.1.16. Manageable Route Segments For AMC And Route

3.1.16.1. SOAP

(1) The associated SOAP operation is:

```
AUPGetManageableRouteSegmentsForAMCAndRouteReply getManageableRouteSegmentsForAMCAndRoute(
    AUPGetManageableRouteSegmentsForAMCAndRouteRequest request
    )
```

3.1.16.2. `AUPGetManageableRouteSegmentsForAMCAndRouteRequest`

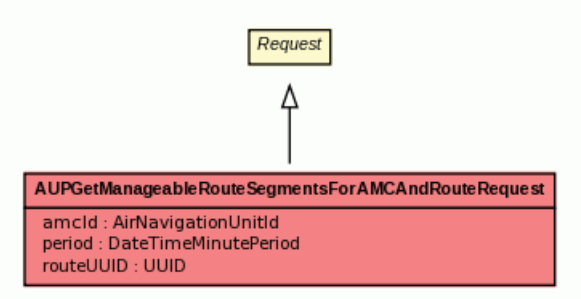

Figure 3.30. `AUPGetManageableRouteSegmentsForAMCAndRouteRequest` Class Diagram

(1) Return the RouteSegments that can be managed for the given AMC and Route.

(2) Inherits from: `Request`

(3) Attributes:

a) **`AirNavigationUnitId amcId`** *(Mandatory)*
The id to identify the AMC.

b) **`DateTimeMinutePeriod period`** *(Mandatory)*
The period to consider. Typically the validity of an AUP or a part of that validity period.

c) **`UUID routeUUID`** *(Mandatory)*

The Route UUID.

3.1.16.3. AUPGetManageableRouteSegmentsForAMCAndRouteReply

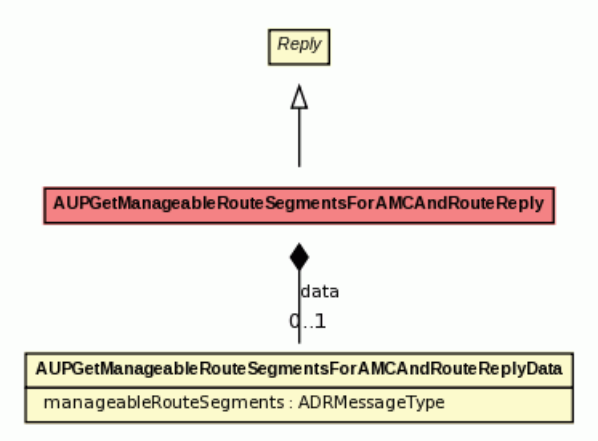

Figure 3.31. AUPGetManageableRouteSegmentsForAMCAndRouteReply Class Diagram

(1) Reply returned in response to AUPGetManageableRouteSegmentsForAMCAndRouteRequest

(2) Inherits from: Reply

(3) Attributes:

a) **ADRMessageType manageableRouteSegments** *(Optional)*
The RouteSegments that can be managed.

3.2. AirspaceStructureService Port Type

3.2.1. Overview

3.2.1.1. Business

(1) The Airspace Data published by NM is composed of the following AIXM5.1 Feature Types :

AIXM 5.1 Feature types
AirportHeliport AirportHeliportCollocation AirportHeliportSet
AngleIndication DesignatedPoint DistanceIndication Navaid
ArrivalLeg DepartureLeg Route

AIXM 5.1 Feature types
RouteSegment StandardInstrumentArrival StandardInstrumentDeparture StandardLevelColumn StandardLevelTable
Airspace
AirTrafficManagementService OrganisationAuthority SpecialDate Unit
FlightRestriction
Flow ReferenceLocation TrafficVolume TrafficVolumeSet

Table 3.1. AIXM 5.1 domain types to AIXM feature types mapping

(2) The granularity of the data retrieved by means of this port type will be at the level of individual AIXM 5.1 Feature types.

3.2.1.1.1. Concepts and definitions

(1) *AIXM5.1 Temporality Model Profile for NM B2B*
Defines the rules governing the temporality aspects during the export of the CACD data into AIXM 5.1 (see the document [3] for more details).

(2) *Complete AIXM Dataset*
It is a consistent and self contained set of AIXM 5.1 Features representing the complete NM Airspace Data as it is known at a given point in time.
Remarks:

a) A Complete AIXM Dataset is published every day by NM and is made available as a set of AIXM5.1 files (one per Feature Type) exposed by this port type.

b) A Complete AIXM Dataset is associated with an Airspace Data Update Id (see below) that represents the latest Update included in the dataset.

c) This Update Id must be used to query for subsequent Updates.

d) The AIXM5.1 Features included in the Complete AIXM Dataset contain only BASELINE Timeslices.

(3) *Incremental AIXM Dataset*
It is a consistent set of AIXM 5.1 Features that represents an update of the NM Airspace Data. The content of the Incremental AIXM Dataset is the set of updated Features.

Remarks:

a) Several Incremental AIXM Datasets may be published by NM every day.

b) Each Incremental AIXM Dataset is associated with an Airspace Data Update Id (see below) that corresponds to the Airspace Data Update included in the dataset.

c) Each Incremental AIXM Dataset has a reference to the previous Airspace Data Update Id, so the Incremental AIXM Datasets form a contiguous chain.

d) The Incremental AIXM Datasets must be applied in the order explicitly specified by NM and one Incremental AIXM Dataset should not be applied if the previous Incremental AIXM Dataset has not been applied.

e) An Incremental AIXM Dataset can be downloaded as BASELINE or PERMDELTA timeslices. It is up to the data consumer to decide the type of timeslices:

- If the data consumer asks for BASELINE timeslices, each Feature included in the Dataset will contain all known BASELINE timeslices which are not in the past with respect to the operational AIRAC at the time of the publication of the Dataset.
- If the data consumer asks for PERMDELTA timeslices, each Feature included in the Dataset will contain the set of PERMDELTA timeslices, computed according to the *AIXM5.1 Temporality Model Profile for NM B2B*.

f) In both cases the Incremental AIXM Dataset will only contain the updated Features.

(4) *Airspace Data Update*
It is a modification to the NM Airspace Data. It is published as an Incremental AIXM Dataset. One Airspace Data Update may affect one or more Airspace Data entities and therefore result into one or more AIXM5.1 Feature updates. An Airspace Data Update may happen as a result of:

a) Changes performed between two consecutive AIRAC cycles;

b) Changes performed during an operational AIRAC cycle;

Remark:
Each Airspace Data Update has a unique identifier (an Airspace Data Update Id, or simply Update Id)

(5) The Complete AIXM Datasets and Incremental AIXM Datasets are such that: The Complete AIXM Dataset published at day D is the result of applying all Incremental AIXM Datasets of day D-1 to the Complete AIXM Dataset of day D-1 (see picture below).

3.2.1.1.2. Dataset publication

(1) The publication of Complete and Incremental AIXM Datasets by NM is summarized by the following picture.

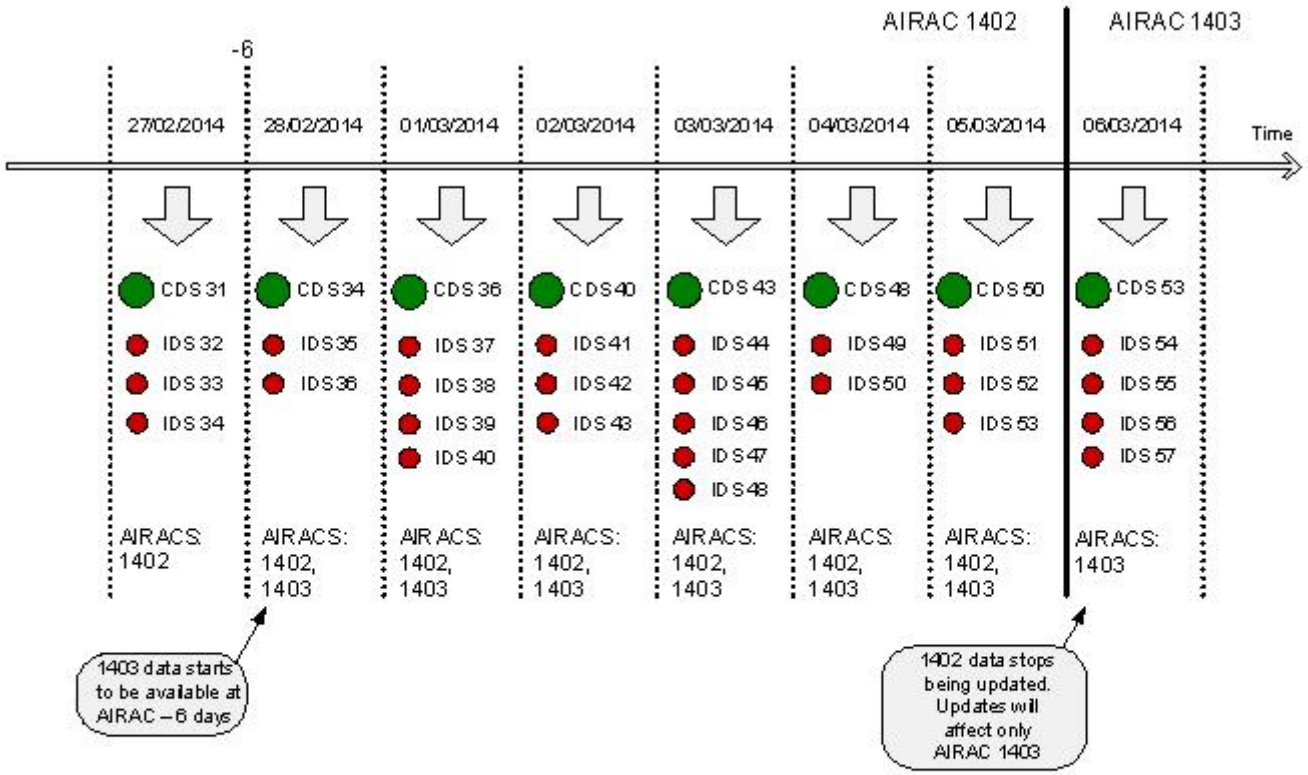

Figure 3.32. Datasets Publication Diagram

(2) The example shows a simplified scenario of dataset daily publication from 7 days before the AIRAC switch to one day after the AIRAC switch (transition between AIRAC 1402 to AIRAC 1403)

(3) One Complete AIXM Dataset is published every day (shown as green circles: CDS stands for Complete AIXM Dataset)

(4) Several Incremental AIXM Datasets are published every day (shown as red circles: IDS stands for Incremental AIXM Dataset)

(5) Each dataset, Complete or Incremental, is associated with an Airspace Data Update Id that uniquely identifies an update to the NM Airspace Data.

(6) Each Incremental AIXM Dataset corresponds to a single Airspace Data Update.

(7) The Update Id associated to a Complete AIXM Dataset corresponds to the latest update included in the dataset: for example CDS 43 is up-to-date with the Airspace Data Update 43, i.e. it contains all the Updates up to 43 included.

(8) The AIRAC numbers shown at the bottom of the picture show which AIRACS may be affected by the Updates: normally the data corresponding to an AIRAC starts to be available 6 days before the AIRAC switch. It means that the datasets published between AIRAC -6 and AIRAC day may contain data both for the current AIRAC and the next. In the example shown in the picture above, the datasets (both Complete and Incremental) published on the 28/02 may contain changes to both AIRAC 1402 and 1403.This information is useful to easily identify all the datasets associated to a particular AIRAC.

(9) The Complete and Incremental AIXM Datasets are made available through the following methods exposed by the *Airspace Structure Port Type:*

a) queryCompleteAIXMDatasets()

b) queryIncrementalAIXMDatasets()
These two methods are explained in detail below. They both return a list of available datasets according to the specified query parameters.

(10) Each dataset is composed of a number of zipped files, one per AIXM5.1 Feature type.

(11) NM requires the service consumers not to massively poll the service to know when there are new datasets available: the service consumer should not query for datasets more than once every 5 minutes.

(12) One of the primary objectives of this service is to provide data that can be automatically processed by ASM tools. However, applicability timetables must be exported as entered by the user. This implies usage of complex time expressions such as weekly expressions (e.g.THU, FRI) and special days (e.g. holidays); in other words, applicability timetables require an interpretation.

(13) See the ADR-Extension Document for more details on the AIXM 5.1 features and properties that are published.

3.2.1.1.2.1. Recommended workflows

(1) The service offers a certain level of flexibility in order to allow many possible use cases.

(2) However NM workflow recommends the following two workflows:

a) Periodic download of Complete AIXM Datasets

b) Follow the NM Airspace Data updates

3.2.1.1.2.1.1. Periodic download of Complete AIXM Datasets

(1) This is the simplest scenario in which a data consumer gets a Complete AIXM Dataset at regular (or not) intervals.

(2) In this case the data consumer will invoke the method `queryCompleteAIXMDatasets()` for a given AIRAC or a given date.

(3) In this case the data consumer will not make use of the UpdateId and will only deal with BASELINE timeslices.

3.2.1.1.2.1.2. Follow the NM Airspace Data updates

(1) In this workflow the data consumer will first download a Complete AIXM Dataset and after that will follow the evolution of the NM Airspace Data by downloading the Updates as Incremental AIXM Datasets:

(2) In this case the data consumer will:

a) Invoke the method `queryCompleteAIXMDatasets()` and download one Complete AIXM Dataset, for example the first available dataset for a given AIRAC.

b) Start polling the NM systems regularly at an interval not smaller than 5 minutes, by querying the method `queryIncrementalAIXMDatasets()` with the lastKnownUpdateId (the first time it is the UpdateId received with the Complete AIXM Dataset, the subsequent times it is the one associated to each downloaded Incremental AIXM Dataset.

c) The data consumer can also decide to fully re-synchronize with the NM systems by re-invoking the `queryCompleteAIXMDatasets()` method.

3.2.1.2. AirportHeliportSet extension

(1) RAD Appendix 2 (Area Definitions) defines collections of AirportHeliports as they are referred to in RAD Restrictions.

(2) Therefore it makes sense to consider these collections of AirportHeliports as a separate Features and refer to these collections of AirportHeliports in FlightRestrictions. The AirportHeliportSet is a new Feature that expresses the concept of the "Area Definition" in the RAD Appendix 2.

(3) The composition of the AirportHeliportSet is defined by:

a) either explicitly listing all the members AirportHeliports using the hasAiportHeliport association,

b) and/or using a pattern match using the hasAirportHeliportSetPattern association. An example of the pattern is "EB", meaning all the Belgian AirportHeliports.

(4) As the AirportHeliportSet is a Feature, it has Timeslices. This implies that the composition of an AirportHeliportSet can change in time.

(5) Special attention should be paid to AirportHeliportSets that use patterns. When a new AirportHeliport is created (e.g. Berlin), then there might be an effect on AirportHeliportSets that use the patter "ED".

3.2.1.3. Airspace extension

(1) The main reasons to extend the Airspace Feature were:

- to export additional information in the FUA context (see below);
- to express whether an Airspace is for operationl use (see below);

3.2.1.3.1. Airspace extension in FUA context

3.2.1.3.1.1. Flexible Use and related levels

(1) *FlexibleUse*: the Airspace is not designated as either military or civil airspace but should be used flexibly on a day-to-day basis. Consequently, any necessary airspace segregation should be only of a temporary nature.

(2) *Level1*: the Airspace is manageable at the strategic level. The act of defining and reviewing as required the national airspace policy taking into account national and international airspace requirements. The RSA activation is determined by the Airspace.activation.

(3) *Level2*: the Airspace is manageable at the pre-tactical level. The act of conducting operational management within the framework of pre-determined existing ATM structure and procedures defined in Level1 and of reaching specific agreement beween civil and military authorities involved. An RSA of Level2 must be included in AUP/UUP to become activated.

(4) Level3: the airspace is manageable at the tactival level.

(5) Some business rules:

a) If FlexibleUse = YES, then at least one of Level1/2/3 must also be YES

b) If FlexibleUse = NO, then all of Level1/2/3 must be NO

c) If Level2 = YES, then Level1 must also be YES

(6) The FUA Airspaces can be categorized as:

a) *NAM*: Airspaces which can be activated by the military or other special users without prior coordination with the civilian users, i.e. AMC during the times defined in the availability. This corresponds to Level1 = YES, Level2 = NO.

b) *AMA*: Airspaces which can be activated in a flexible way for use by the military or other special users after due coordination between military and civilian airspaces during the times defined in the availability. This corresponds to Level1 = YES, Level2 = YES.

3.2.1.3.1.2. Nearby, Offload and NotAffected

(1) *Nearby AirRoutes*. An AirRoute can be classified as a nearby Route for zero or more RSAs. The relationship is meant to help AMCs in the decision process for which Routes can be opened/closed by an RSA Activation. As this relationship is bi-directional, it suffices that the RSA/RouteSegment

vertical limit combination exists in one direction. Activating an RSA implies closing the Nearby AirRoutes.

(2) *Offload AirRoutes*. An AirRoute can be classified as an Offload Route for zero or more RSAs. The relationship is meant to help AMCs in the decision process for which Routes can be opened/closed by an RSA Activation. Activating an RSA implies opening an Offload AirRoute.

(3) *NotAffected AirRoutes*. An AirRoute can be classified as a not-affected-by-RSA-activation for zero or more RSAs. The relationship is meant to help AMCs in the decision process for which Routes can be opened/closed by an RSA Activation. Activating an RSA has no impact on the NotAffected AirRoute, even if it geometrically crosses the RSA.

3.2.1.3.1.3. Activations

(1) The availability of the RSA is expressed with the association `Airspace.activation` where the `AirspaceActivation.status = AVBL_FOR_ACTIVATION`. Note that the AIXM 5.1 attribute is called activation but it is in fact used to express the availability.

(2) The activation of an RSA is expressed in the `Airspace.rsaActivation` in the `AirspaceExtension`, with `AirspaceActivation.status = ACTIVE`.

3.2.1.3.2. Airspace Operational Usage

3.2.1.3.2.1. Description

(1) The majority of airspaces modeled in CACD are for operational use. However, in addition to those airspaces, NM also uses other airspaces that, although they may not be operational at some moment, they may become operational at some point in time. The reasons behind the use of such airspaces are varied, some may be experimental, some may be for contingency, others to allow reacting quickly to crisis situations, etc.

(2) It would be desirable not to export such airspaces because they are not of general interest when not operational. However, if they do become operational they would have to be exported. Also note that these airspaces may potentially switch from operational to not-operational several times. Therefore exporting an airspace only when it is operational and not exporting it when not-operational would create "holes" in the feature's lifetime. The AIXM 5.1 model does not foresee "holes" in the lifetime of a feature.

(3) For this reason the following approach has been chosen: such airspaces, or any such feature in general, is always exported and the notion of being or not operational is exported by means of a new attribute `Airspace.usage` in the `AirspaceExtension` class which may take the following values:

- `OPERATIONAL`: meaning that the feature is for operational use.
- `WITHHELD`: meaning that the feature is not for operational use and for this reason it has been withheld.

In addition to this new `usage` attribute, in the WITHHELD timeslices all other attributes are nullified, to make it more explicit that the feature is not to be used (hence the word "withheld").

(4) The following four scenarios are possible:

a) The feature is new and it is not operational:

i) The feature is exported and all its properties in the only timeslice are nullified

ii) The `usage` attribute is set to `WITHHELD`

b) The feature is new and it is operational:

i) The feature is exported and its only timeslice has its properties set

ii) The `usage` attribute is set to `OPERATIONAL`

c) The feature is changed from operational to non-operational:

i) A new timeslice is exported and all its properties are nullified

ii) The `usage` attribute is set to `WITHHELD`

d) The feature is changed from non-operational to operational:

i) A new timeslice is exported with its properties set

ii) The `usage` attribute is set to `OPERATIONAL`

(5) UML representation:

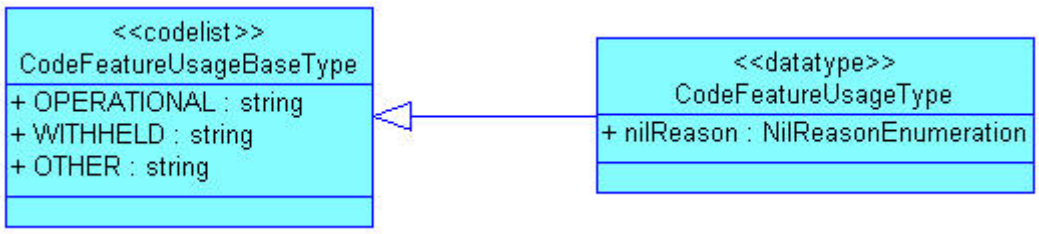

Figure 3.33. FeatureUsage

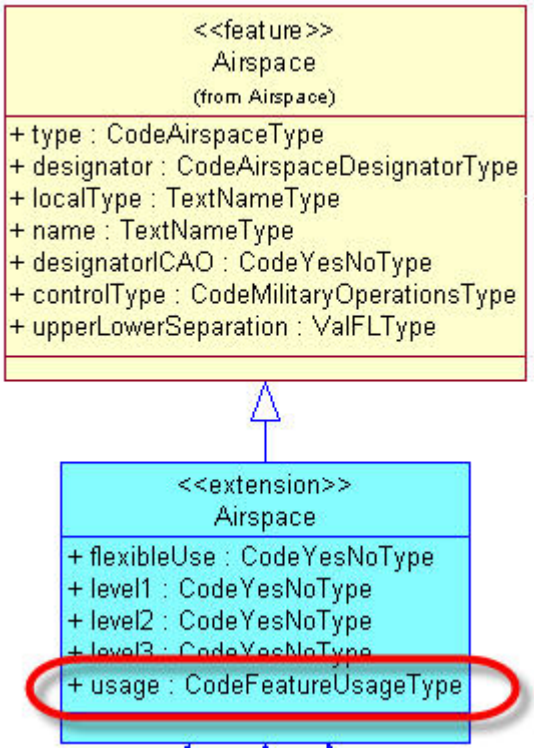

Figure 3.34. Airspace Extension

(6) XSD:

```
<simpleType name="CodeFeatureUsageBaseType">
  <union>
    <simpleType>
      <restriction base="xsd:string">
        <enumeration value="WITHHELD"/>
        <enumeration value="OPERATIONAL"/>
      </restriction>
    </simpleType>
    <simpleType>
      <restriction base="string">
        <pattern value="OTHER(:(\w|_){1,58})?"/>
      </restriction>
    </simpleType>
  </union>
</simpleType>
<complexType name="CodeFeatureUsageType">
  <simpleContent>
    <extension base="adr:CodeFeatureUsageBaseType">
      <attribute name="nilReason" type="gml:NilReasonEnumeration"/>
    </extension>
  </simpleContent>
</complexType>
```

(7) Example -- WITHHELD as PERMDELTA:

```
<adrmsg:hasMember>
  <aixm:Airspace gml:id="ID_36_1398421189613_755">
    <gml:identifier codeSpace="urn:uuid:">d7064a20-6b6f-4bc6-a946-5bb3cd887c7b</gml:identifier>
    <aixm:timeSlice>
      <aixm:AirspaceTimeSlice gml:id="ID_36_1398421189613_756">
        <gml:validTime>
          <gml:TimeInstant>
            <gml:timePosition>2006-06-08T00:00:00</gml:timePosition>
          </gml:TimeInstant>
        </gml:validTime>
        <aixm:interpretation>PERMDELTA</aixm:interpretation>
        <aixm:type> xsi:nil="true" nilReason="withheld"> </aixm:type>
        <aixm:designator xsi:nil="true" nilReason="withheld"> </aixm:designator>
        <aixm:designatorICAO xsi:nil="true" nilReason="withheld"> </aixm:designatorICAO>
        <aixm:geometryComponent xsi:nil="true" nilReason="withheld"> </aixm:geometryComponent>
        <aixm:extension>
          <adrext:AirspaceExtension gml:id="ID_36_1398421189613_762">
            <adrext:usage>WITHHELD</adrext:usage>
          </adrext:AirspaceExtension>
        </aixm:extension>
      </aixm:AirspaceTimeSlice>
    </aixm:timeSlice>
  </aixm:Airspace>
</adrmsg:hasMember>
```

(8) Example -- WITHHELD as BASELINE:

```
<adrmsg:hasMember>
  <aixm:Airspace gml:id="ID_36_1398421189613_755">
    <gml:identifier codeSpace="urn:uuid:">d7064a20-6b6f-4bc6-a946-5bb3cd887c7b</gml:identifier>
    <aixm:timeSlice>
      <aixm:AirspaceTimeSlice gml:id="ID_36_1398421189613_756">
        <gml:validTime>
          <gml:TimePeriod gml:id="ID_36_1398421189613_757">
            <gml:beginPosition>2006-06-08T00:00:00</gml:beginPosition>
            <gml:endPosition indeterminatePosition="unknown"/>
          </gml:TimePeriod>
        </gml:validTime>
        <aixm:interpretation>BASELINE</aixm:interpretation>
        <aixm:featureLifetime>
          <gml:TimePeriod gml:id="ID_36_1398421189613_758">
            <gml:beginPosition>2006-06-08T00:00:00</gml:beginPosition>
            <gml:endPosition indeterminatePosition="unknown"/>
          </gml:TimePeriod>
        </aixm:featureLifetime>
        <aixm:extension>
          <adrext:AirspaceExtension gml:id="ID_36_1398421189613_762">
            <adrext:usage>WITHHELD</adrext:usage>
          </adrext:AirspaceExtension>
        </aixm:extension>
      </aixm:AirspaceTimeSlice>
    </aixm:timeSlice>
  </aixm:Airspace>
</adrmsg:hasMember>
```

(9) Example -- OPERATIONAL:

```
<adrmsg:hasMember>
  <aixm:Airspace gml:id="ID_36_1398421189613_755">
    <gml:identifier codeSpace="urn:uuid:">d7064a20-6b6f-4bc6-a946-5bb3cd887c7b</gml:identifier>
    <aixm:timeSlice>
      <aixm:AirspaceTimeSlice gml:id="ID_36_1398421189613_756">
        <gml:validTime>
          <gml:TimePeriod gml:id="ID_36_1398421189613_757">
            <gml:beginPosition>2006-06-08T00:00:00</gml:beginPosition>
            <gml:endPosition indeterminatePosition="unknown"/>
          </gml:TimePeriod>
        </gml:validTime>
        <aixm:interpretation>BASELINE</aixm:interpretation>
        <aixm:featureLifetime>
          <gml:TimePeriod gml:id="ID_36_1398421189613_758">
            <gml:beginPosition>2006-06-08T00:00:00</gml:beginPosition>
            <gml:endPosition indeterminatePosition="unknown"/>
          </gml:TimePeriod>
        </aixm:featureLifetime>
        <aixm:type>TMA</aixm:type>
        <aixm:designator>LEBBTMA</aixm:designator>
        <aixm:designatorICAO>YES</aixm:designatorICAO>
        <aixm:geometryComponent>
          <aixm:AirspaceGeometryComponent gml:id="ID_36_1398421189613_759">
            <aixm:operation>BASE</aixm:operation>
            <aixm:theAirspaceVolume>
              <aixm:AirspaceVolume gml:id="ID_36_1398421189613_760">
                <aixm:contributorAirspace>
                  <aixm:AirspaceVolumeDependency gml:id="ID_36_1398421189613_761">
                    <aixm:dependency>FULL_GEOMETRY</aixm:dependency>
                    <aixm:theAirspace xlink:href="urn:uuid:8627b55f-5f3e-4490-9a87-1a03aa409f0c"/>
                  </aixm:AirspaceVolumeDependency>
                </aixm:contributorAirspace>
              </aixm:AirspaceVolume>
            </aixm:theAirspaceVolume>
          </aixm:AirspaceGeometryComponent>
        </aixm:geometryComponent>
        <aixm:extension>
          <adrext:AirspaceExtension gml:id="ID_36_1398421189613_762">
            <adrext:usage>OPERATIONAL</adrext:usage>
          </adrext:AirspaceExtension>
        </aixm:extension>
      </aixm:AirspaceTimeSlice>
    </aixm:timeSlice>
  </aixm:Airspace>
</adrmsg:hasMember>
```

3.2.1.4. AirspaceActivation extension

(1) It contains the Information Region in which the airspace is located and the AMC requesting the activation. It also allows specifying which associated FUARestrictions are to be activated.

3.2.1.5. CodeFeatureUsage extension

(1) Possible values are `OPERATIONAL` and `WITHHELD`.

(2) For information about its usage refer to <Airspace Operational Usage>

3.2.1.6. FlightCondition extension

(1) `hasAirspaceBorderCrossing`:
The reason for this extension is that the core AIXM AirspaceBorderCrossing is a Feature, whereas we rather see this an <<Object>>. So it is a replacement for the FlightConditionElementChoice.borderCrossingCondition.

(2) `hasAirportHeliportSet`:
Some FlightRestrictions in the RAD are expressed as departing from or arriving at a collection of AirportHeliports as described in RAD Appendix 2.

3.2.1.7. FlightRestriction extension

3.2.1.7.1. processingIndicator

3.2.1.7.1.1. AerodromeConnectingPoints

(1) processingIndicator = AD_CP.

(2) These FlightRestrictions are described in RAD Appendix 5.

(3) The FlightCondition is a combination of arriving and/or departing AirportHeliport and a DirectFlightClass.

(4) The FlightRouting describes the DCT connecting points to the AirportHeliport.

3.2.1.7.1.2. FRA DCT

(1) processingIndicator = FRA_DCT.

(2) FlightRoutings are limited to:

a) DirectFlightSegment

b) SignificantPoint

(3) The FlightCondition can be:

a) Crossing an Airspace
A SignificantPoint in the FlowRouting has to be interpreted as an intermediate point in the crossed Airspace from the FlightCondition.

b) Crossing an AirspaceBorder
A SignificantPoint in the FlowRouting has to be interpreted as and Entry and/or Exit Point depending on the following.

i) The FlightCondition consists of one AirspaceBorder AsExit - AsEntry.

A) For the Airspace AsExit, the SignificantPoint is an Exit Point.

B) For the Airspace AsEntry, the SignificantPoint is an Entry Point.

ii) The FlightCondition consists of two AirspaceBorders As1 - As2 and As2 - As1. The SignificantPoint is an Entry/Exit Point.

(4) In order to find all the Entry/Exit Points, all the FRA-DCT FlightRestrictions that contains AirspaceBorder FlightConditions have to be considered.

3.2.1.7.1.3. Flight Profile Restriction

(1) processingIndicator = FPR.

(2) These FlightRestrictions do not invalidate flight plans.

3.2.1.7.1.4. RAD DCT

(1) processingIndicator = RAD_DCT.

(2) The FlightCondition can be:

a) Airspace

b) AirspaceBorder

(3) The FlightConditionCirumstance has fixed values:

a) referenceLocation = YES

b) relationWithLocation = YES

(4) It is possible to express a DCT limit with allowed and disallowed segments. But this means you have to define two FlightRestictions.

a) The first FlightRestriction expresses the DCT limit in an Airspace (FlightCondition) and the allowed FlightRoutings (DCT segments longer than the DCT limit).

b) The second FlightRestriction expresses a FlightCondition on the same Airspace but without DCT limit. The FlightRoutings are the disallowed DCT segments (DCT segments shorter than the DCT limit specified in the other FlightRestriction).

3.2.1.7.1.5. Traffic Flow Restriction

(1) processingIndicator = TFR.

(2) These FlightRestrictions originate from the RAD, AIP, ENV-COR, NOTAMs and other communication between ANSPs and NM.

3.2.1.7.1.6. Aerodrome Flight Rule

(1) processingIndicator = OTHER:__ADR_AD_FLIGHT_RULE.

(2) The CACD model differs from the AIXM model. The AirportHeliportAvailability with regards to IFR usage is modelled in CACD as a FlightRestriction.

(3) A Flight Plan will be rejected if that Flight Plan specifier IFR and arriving/departing AirportHeliport is not supporting IFR at that time of the day.

(4) The CACD data are modelled into a FlightRestriction with a FlightCondition arriving to /departing from an AirportHeliport.

(5) IFPS will return the FlightRestriction designator when the Flight Plan violates the FlightRestriction.

3.2.1.7.1.7. Flight Properties on Procedures

(1) processingIndicator = OTHER:__ADR_FLIGHT_PROPERTY_ON_TP.

(2) Here again, the CACD models differs from the AIXM model. The Procedure.aircraftCharacteristic is modelled in CACD as a FlightRestriction.

(3) A Flight Plan will be rejected when the SID/STAR doesn't support the AircraftCharacteristic.

(4) The FlightRestriction is modelled as a FlightCondition arriving to / departing from an AirportHeliport and FlightRoutings forbidding/imposing the use of some SID/STARs.

3.2.1.7.2. enabled

(1) When enabled = NO, this means that the FlightRestriction is not active. This is typically used for re-occurring events, like annual exhibitions.

3.2.1.7.3. usage

3.2.1.7.3.1. Description

(1) It expresses the same concept already described for Airspaces (see <Airspace Operational Usage>).

(2) Example -- WITHHELD as PERMDELTA:

```
<adrmsg:hasMember>
  <aixm:FlightRestriction gml:id="ID_14_1398421187379_2">
    <gml:identifier codeSpace="urn:uuid:">cca9c008-a718-4d5d-9339-1bf013fbc94a</gml:identifier>
    <aixm:timeSlice>
      <aixm:FlightRestrictionTimeSlice gml:id="ID_14_1398421187379_3">
        <gml:validTime>
          <gml:TimeInstant>
            <gml:timePosition>2014-04-03T00:00:00</gml:timePosition>
          </gml:TimeInstant>
        </gml:validTime>
        <aixm:interpretation>PERMDELTA</aixm:interpretation>
        <aixm:instruction xsi:nil="true" nilReason="withheld"> </aixm:instruction>
        <aixm:extension>
          <adrext:FlightRestrictionExtension gml:id="ID_14_1398421187379_125">
            <adrext:usage>WITHHELD</adrext:usage>
          </adrext:FlightRestrictionExtension>
        </aixm:extension>
      </aixm:FlightRestrictionTimeSlice>
    </aixm:timeSlice>
  </aixm:FlightRestriction>
```

```
</adrmsg:hasMember>
```

(3) Example -- WITHHELD as BASELINE:

```
<adrmsg:hasMember>
  <aixm:FlightRestriction gml:id="ID_14_1398421187379_2">
    <gml:identifier codeSpace="urn:uuid:">cca9c008-a718-4d5d-9339-1bf013fbc94a</gml:identifier>
    <aixm:timeSlice>
      <aixm:FlightRestrictionTimeSlice gml:id="ID_14_1398421187379_3">
        <gml:validTime>
          <gml:TimePeriod gml:id="ID_14_1398421187379_4">
            <gml:beginPosition>2014-04-03T00:00:00</gml:beginPosition>
            <gml:endPosition indeterminatePosition="unknown"/>
          </gml:TimePeriod>
        </gml:validTime>
        <aixm:interpretation>BASELINE</aixm:interpretation>
        <aixm:featureLifetime>
          <gml:TimePeriod gml:id="ID_14_1398421187379_5">
            <gml:beginPosition>2014-04-03T00:00:00</gml:beginPosition>
            <gml:endPosition indeterminatePosition="unknown"/>
          </gml:TimePeriod>
        </aixm:featureLifetime>
        <aixm:extension>
          <adrext:FlightRestrictionExtension gml:id="ID_14_1398421187379_125">
            <adrext:usage>WITHHELD</adrext:usage>
          </adrext:FlightRestrictionExtension>
        </aixm:extension>
      </aixm:FlightRestrictionTimeSlice>
    </aixm:timeSlice>
  </aixm:FlightRestriction>
</adrmsg:hasMember>
```

(4) Example -- OPERATIONAL:

```
<adrmsg:hasMember>
  <aixm:FlightRestriction gml:id="ID_14_1398421187379_2">
    <gml:identifier codeSpace="urn:uuid:">cca9c008-a718-4d5d-9339-1bf013fbc94a</gml:identifier>
    <aixm:timeSlice>
      <aixm:FlightRestrictionTimeSlice gml:id="ID_14_1398421187379_3">
        <gml:validTime>
          <gml:TimePeriod gml:id="ID_14_1398421187379_4">
            <gml:beginPosition>2014-04-03T00:00:00</gml:beginPosition>
            <gml:endPosition indeterminatePosition="unknown"/>
          </gml:TimePeriod>
        </gml:validTime>
        <aixm:interpretation>BASELINE</aixm:interpretation>
        <aixm:featureLifetime>
          <gml:TimePeriod gml:id="ID_14_1398421187379_5">
            <gml:beginPosition>2014-04-03T00:00:00</gml:beginPosition>
            <gml:endPosition indeterminatePosition="unknown"/>
          </gml:TimePeriod>
        </aixm:featureLifetime>
        <aixm:designator>DS5525G</aixm:designator>
        <aixm:type>MANDATORY</aixm:type>
        <aixm:instruction>
          DEP EKCH:THE USE OF SIDS IS MANDATORY EXCEPT FOR DEST.
          WITHIN THE COPENHAGENGROUP,MALMO GROUP
          INFO:
```

```
          MANDATES THE ONLY POSSIBILITIES FOR JET AC,BEING
          BETUD,KEMAX,LANGO,MIKSI,NEXEN,ODN,SIMEG AND VEDAR SIDS
          OUTSIDE THE ACTIVATION OF THE RSA MULTEX/EKD352/53
        </aixm:instruction>
        <aixm:flight>
          <aixm:FlightConditionCombination gml:id="ID_14_1398421187379_6">
            ...
          </aixm:FlightConditionCombination>
        </aixm:flight>
        <aixm:regulatedRoute>
          <aixm:FlightRestrictionRoute gml:id="ID_14_1398421187379_23">
            ...
          </aixm:FlightRestrictionRoute>
        </aixm:regulatedRoute>
        <aixm:regulatedRoute>
          <aixm:FlightRestrictionRoute gml:id="ID_14_1398421187379_23">
            ...
          </aixm:FlightRestrictionRoute>
        </aixm:regulatedRoute>
        <aixm:extension>
          <adrext:FlightRestrictionExtension gml:id="ID_14_1398421187379_125">
            <adrext:processingIndicator>TFR</adrext:processingIndicator>
            <adrext:enabled>NO</adrext:enabled>
            <adrext:usage>OPERATIONAL</adrext:usage>
          </adrext:FlightRestrictionExtension>
        </aixm:extension>
      </aixm:FlightRestrictionTimeSlice>
    </aixm:timeSlice>
  </aixm:FlightRestriction>
</adrmsg:hasMember>
```

3.2.1.8. RouteAvailability extension

(1) The `conditionalRouteType` is used to express a RouteAvailability as follows:

a) CDR_1: Conditional Route Type 1, normally available for flight planning, but can be closed.

b) CDR_2: Conditional Route Type 2, normally not available for flight planning, but can be opened.

c) CDR_3: Conditional Route Type 3, potentially available for ATC re-routeing.

(2) The hostAirspace is used in the AirspaceAvailability Service to express in which FIR(s) the RouteSegments are located.

3.2.1.9. RoutePortion extension

(1) The `referencedProcedure` allows to define a 'ProcedurePortion'. Instead of having a RoutePortion between any two points of Route, this extension allows to expressing a portion between any two points on SID/STAR.

(2) The `range` allows to express some altitudes related to the RoutePortion. This is used to express vertical limits when defining nearBy and offLoad associations between Airspaces and RoutePortions.

(3) The `intermediatePoint`(s) is used when defining FlightConditions and FlightRoutings where the order (sequence) of the SignificantPoints to be traversed is important, but not Route dependent.

3.2.1.10. RouteSegment extension

(1) The `verticalLimits` describe during which part of the day the portions of the route exist, i.e. it is possible to declare a Route as non-existing, e.g. during the night to allow DCT in the Airspace.

(2) The `cdrUpdate` association is used to export the AIXM TEMP_DELTAs related to the publication of the AUP/UUP.

3.2.1.11. StandardInstrumentArrival extension

(1) The `connectingPoint`(s) in a STAR are the published ICAO Points (Navigation Aid or Waypoint) that may serve as connecting point to the en-route network. In other words a flight may join the STAR only at these points.

(2) The `initialApproachFix`(s) in a STAR is a point that connects the Arrival Procedure to the Instrument Approach Procedure.

3.2.1.12. StandardInstrumentDeparture extension

(1) The `connectingPoint`(s) in a SID are the published ICAO Points (Navigation Aid or Waypoint) that may serve as connecting point to the en-route network. In other words a flight may leave the SID only at these points.

3.2.1.13. TimeSheet extension

(1) The granularity of the CACD Timeslices are at AIRAC boundaries. This granularity is not sufficient to express permanent changes during the AIRAC, e.g. permanent changes to the CDR definitions.

(2) There are two types of time information inside the CACD Timeslices:

a) Timetables that express when a property is defined, i.e during certain times of some days of the week. In CACD these timetables are expressed with a string like "2012/10/18->2014/04/03 -----67 06:00 - 10:00". It is a bit more complex than that, because it is also possible to express a holiday, day-before-holidy, busy-Friday...

b) A time period when a property needs is valid, i.e. typically when AIXM would use a TEMP_DELTA to express Route closures/openings and Airpace allocations.

(3) The AIXM Timesheet (part of PropertiesWIthSchedule) doesn't cover the concept of 'year'. There is 'startDate' and 'endDate' but there is no calendar year involved.

(4) It was chosen to extend the AIXM.Timesheet with a gml:validTime element in order to express the fact that CACD properties have sub-AIRAC validities.

3.2.1.14. NAT Tracks publication

3.2.1.14.1. Description

(1) NAT Tracks are used to fly over the Atlantic Ocean. They are published daily by Shanwick Center.

a) The Daytime - Westbound OTS NAT tracks are available between 11:30 - 19:00. The most northerly starts with designator 'A'.

b) The Nighttime - Eastbound OTS NAT tracks are available between 01:00 - 08:00. The most southerly start with the designator 'Z'.

(2) The same NAT track (same UUID) may change shape over different days. For example one day it may be composed of some segments, and another day by other segments. In other words, the same segment may be used in the NAT track in one day, and not used in another day, and then used again.
This creates a conceptual problem: when a RouteSegment is not used in any NAT track, it still continues to exist in CACD with its own UUID and it may become part of a NAT track later in the future. This behaviour must be reflected in the exported AIXM features. A feature cannot have holes in its lifetime (this would also cause problems in the computation of PERMDELTAs). So rather than omitting timeslices, the chosen solution was to always export the RouteSegment timeslices as follows:

a) when the RouteSegment is in use, i.e. it is part of a NAT track, a new timeslice is created with a reference to the route (attribute `routeFormed`) and its availability;

b) when the RouteSegment is not in use, i.e. it is not part of any NAT tracks, a new timeslice is created without a reference to any NAT track (attribute `routeFormed` omitted or null) and with no availability;

This behaviour should also be quite intuitive because it reflects the reality.

(3) The availability of a RouteSegment corresponds to the NAT Signal Period, which is as follows:
For Westbound:

a) lifetime from 19:00 till 19:00

b) NAT Signal Period from 11:30 till 19:00

For Eastbound:

a) lifetime from 08:00 till 8:00

b) NAT Signal Period from 01:00 till 08:00

3.2.1.14.2. Publication

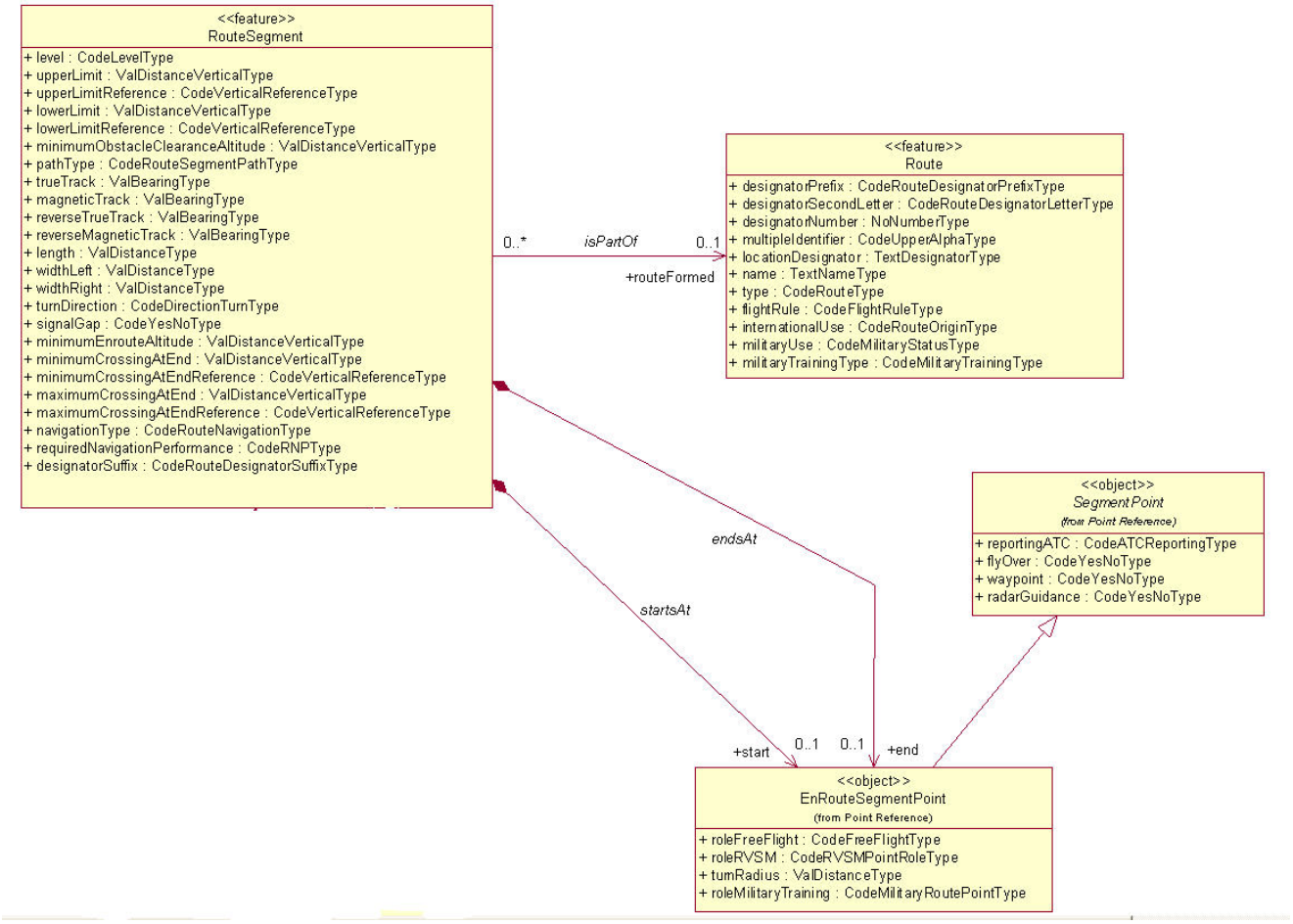

Figure 3.35. AIXM 5.1 features and objects used to publish the NAT Tracks

3.2.1.14.2.1. aixm:Route

(1) The attribute `aixm:Route.name` will contain the ICAO id of the OTS route (e.g. 'NATZ')

(2) The attribute `aixm:Route.type` will be always 'NAT'.

3.2.1.14.2.2. aixm:RouteSegment

(1) RouteSegments will tend to have many timeslices towards the end of an AIRAC cycle. Potentially they may have up to 1 timeslice per day.

(2) Each `aixm:RouteSegment` feature merely contains a `aixm:RouteSegment.start`, a `aixm:RouteSegment.end` and a `aixm:RouteSegment.availability`.

(3) The `aixm:RouteSegment.routeFormed` will contain the reference to the NAT track. IMPORTANT: note that this may be empty in some timeslices, see explanation above.

(4) The `aixm:RouteSegment.availability.direction` will be always 'FORWARD'.

(5) The `aixm:RouteSegment.availability.status` will be always 'OPEN'.

(6) The `aixm:RouteSegment.availability.timeInterval` is expressed using the AIXM extension `adr:TimesheetExtension`. It is a Period where:

a) The `timeReference` is always 'UTC'.

b) The `day` is always 'ANY'.

c) The `exluded` attribute is always 'NO'.

d) The period is expressed using the `gml:TimePeriod`.

3.2.1.14.2.3. Publication Example

```
<adrmsg:ADRMessage
    gml:id="ID_51_1352812234877_1"
    xmlns:adrmsg="http://www.eurocontrol.int/cfmu/b2b/ADRMessage"
    xmlns:adr="http://www.aixm.aero/schema/5.1/extensions/ADR"
    xmlns:gml="http://www.opengis.net/gml/3.2"
    xmlns:aixm="http://www.aixm.aero/schema/5.1"
    xmlns:xlink="http://www.w3.org/1999/xlink">
  <adrmsg:hasMember>
    <aixm:Route gml:id="ID_40_1377648348105_74">
      <gml:identifier codeSpace="urn:uuid:">6b2791d6-c61d-4d8a-8fef-eb74b4bd07e3</gml:identifier>
      <aixm:timeSlice>
        <aixm:RouteTimeSlice gml:id="ID_40_1377648348105_75">
          <gml:validTime>
            <gml:TimePeriod gml:id="ID_40_1377648348105_76">
              <gml:beginPosition>2013-07-25T00:00:00</gml:beginPosition>
              <gml:endPosition indeterminatePosition="unknown"/>
            </gml:TimePeriod>
          </gml:validTime>
          <aixm:interpretation>BASELINE</aixm:interpretation>
          <aixm:featureLifetime>
            <gml:TimePeriod gml:id="ID_40_1377648348105_77">
              <gml:beginPosition>2013-07-25T00:00:00</gml:beginPosition>
              <gml:endPosition indeterminatePosition="unknown"/>
            </gml:TimePeriod>
          </aixm:featureLifetime>
          <aixm:name>NATZ</aixm:name>
          <aixm:type>NAT</aixm:type>
        </aixm:RouteTimeSlice>
      </aixm:timeSlice>
    </aixm:Route>
  </adrmsg:hasMember>
  <adrmsg:hasMember>
    <aixm:RouteSegment gml:id="ID_40_1377648348105_78">
      <gml:identifier
      codeSpace="urn:uuid:">Route_6b2791d6-c61d-4d8a-8fef-eb74b4bd07e3.
                            51855585-f528-41f1-b254-4b7c139d46a8.
                            2ca64b3c-535f-47f0-a57f-adb3cda37600</gml:identifier>
      <aixm:timeSlice>
        <aixm:RouteSegmentTimeSlice gml:id="ID_40_1377648348105_79">
          <gml:validTime>
            <gml:TimePeriod gml:id="ID_40_1377648348105_80">
              <gml:beginPosition>2013-07-24T08:00:00</gml:beginPosition>
              <gml:endPosition>2013-07-25T08:00:00</gml:endPosition>
```

```
            </gml:TimePeriod>
          </gml:validTime>
          <aixm:interpretation>BASELINE</aixm:interpretation>
          <aixm:featureLifetime>
            <gml:TimePeriod gml:id="ID_40_1377648348105_81">
              <gml:beginPosition>2013-07-24T08:00:00</gml:beginPosition>
              <gml:endPosition>2013-07-25T08:00:00</gml:endPosition>
            </gml:TimePeriod>
          </aixm:featureLifetime>
          <aixm:start>
            <aixm:EnRouteSegmentPoint gml:id="ID_40_1377648348105_82">
              <!--YQX -->
              <aixm:pointChoice_fixDesignatedPoint
              xlink:href="urn:uuid:51855585-f528-41f1-b254-4b7c139d46a8"/>
            </aixm:EnRouteSegmentPoint>
          </aixm:start>
          <aixm:routeFormed xlink:href="urn:uuid:024bb6f8-3265-472a-9988-c765f519bcef"/>
          <aixm:end>
            <aixm:EnRouteSegmentPoint gml:id="ID_40_1377648348105_83">
              <!-- KOBEV -->
              <aixm:pointChoice_fixDesignatedPoint
              xlink:href="urn:uuid:2ca64b3c-535f-47f0-a57f-adb3cda37600"/>
            </aixm:EnRouteSegmentPoint>
          </aixm:end>
          <aixm:availability>
            <aixm:RouteAvailability gml:id="ID_40_1377648348105_84">
              <!-- Timesheet -->
              <aixm:timeInterval>
                <aixm:Timesheet gml:id="ID_50_1352812184610_10_1">
                  <aixm:timeReference>UTC</aixm:timeReference>
                  <aixm:day>ANY</aixm:day>
                  <aixm:excluded>NO</aixm:excluded>
                  <aixm:extension>
                    <adr:TimesheetExtension gml:id="ID_50_1352812184610_10_2">
                      <gml:validTime>
                        <gml:TimePeriod gml:id="ID_40_1377648348105_83">
                          <gml:beginPosition>2013-07-25T01:00:00</gml:beginPosition>
                          <gml:endPosition>2013-07-25T08:00:00</gml:endPosition>
                        </gml:TimePeriod>
                      </gml:validTime>
                    </adr:TimesheetExtension>
                  </aixm:extension>
                </aixm:Timesheet>
              </aixm:timeInterval>
              <aixm:direction>FORWARD</aixm:direction>
              <aixm:status>OPEN</aixm:status>
            </aixm:RouteAvailability>
          </aixm:availability>
        </aixm:RouteSegmentTimeSlice>
      </aixm:timeSlice>
    </aixm:RouteSegment>
  </adrmsg:hasMember>
  <!--  other RouteSegments -->
</adrmsg:ADRMessage>
```

3.2.1.15. Pre-Tactical and Tactical updates of the Daily Plan.

3.2.1.15.1. CACD Retrievals vs. Tactical Situation

(1) The current situation is that the airspace data involved in the Daily Plan is maintained in two systems:

a) The NM CACD (airspace) system for AIRAC definitions

b) NM flow system for tactical updates

(2) The export of related CACD objects (consistently with the export of other CACD objects) does not take into account the tactical updates done via the NM flow system (pre-tactical and tactical situations). Hence:

a) When retrieving a daily plan (see below), the caller receives the superset of CACD values on periods that were not updated (pre-)tactically and of the tactical updates.

b) The caller can only input the tactical updates. These should not include the CACD values (providing the CACD values has the effect of overwriting all that has been planned in CACD see below).

c) To retrieve a Daily Plan:

- Retrieve a Sector Configuration Plan;
- Retrieve a Capacity Plan;
- Retrieve a Traffic Volume Activation Plan;
- Retrieve an OTMV Plan;
- Retrieve a Runway Configuration Plan.

3.2.1.15.2. Daily Plan Update Pattern

(1) The pattern used for all tactical updates is the update of the entire list of values for a day. This list of values must be a consistent and complete time partition for the whole day. For each time period the user can specify whether new values or the existing CACD values should be used.

(2) More precisely, the client system will be able to update in one shot:

a) The plan of runway configurations for an entire day and an aerodrome;

b) The plan of sector configuration activations for an entire day and an airspace (AUA or sector cluster);

c) The plan of capacities for an entire day and a traffic volume;

d) The plan of traffic volume activations for an entire day and a traffic volume;

e) The plan of OTMVs for an entire day and a traffic volume for a duration;

(3) As mentioned above, each daily plan is a complete time partition of the day, meaning that each daily plan update which is made of values defined over periods must be such that its periods do not overlap and cover the whole day. Providing a daily plan that is not a time partition will result in an error.

(4) The data value associated to each period of the plan must be:

a) either an indication that for this period the CACD data is to be used (note that for some plans, e.g. for some capacity plans, there is no data defined in CACD for certain locations/traffic volumes)

b) or the specific values to be used for that period.

(5) So, it is up to the B2B client to either always pass the plan with the full CACD values (which are copied over, hence overwritten) when not updated, or to pass only the actual tactical updates and indications that CACD values should be used in the absence of tactical updates.

(6) Any of the plans mentioned above may be updated via B2C and/or B2B, and in both cases by different operators. When an operator updates a plan via B2C, the next B2B retrieve plan operation will include these changes done via B2C. The pattern used on the backend side to deal with concurrent updates is the following:

a) Each daily plan is returned with a data id that expresses a data version number (equivalent to a timestamp)

b) Before updating a daily plan, the updater must first get the plan and subsequently pass the associated data id when updating it. IMPORTANT: note that this data id is also related to the dataset in use, i.e. a data id obtained from a dataset cannot be used with another dataset: doing so would result in an error.

c) A concurrent update is defined as an update that took place earlier (i.e. before the update that the updater wants to execute now) but after the timestamp associated to the data id passed within the update to execute now.
For example:

i) A B2B client shows in a local screen a daily plan.corresponding to dataId I1

ii) A NM client in a parallel modifies the same daily plan (for the same location and updated periods) via B2B or B2C -> latest version in NM systems : dataId I2.

iii) The B2B client end-user modifies some values of the plan and tries to commit them (includes sending to NM; changes wrt I1)

iv) As the B2B client end-user started from dataId I1 but the periods that were modified were also conflictingly modified in parallel CONFLICTING_UPDATE ReplyStatus is returned

d) From the concurrency perspective, a daily plan update is successful if:

i) There was no concurrent update, or

ii) There were identical concurrent updates, and/or

iii) There were concurrent updates that did not involve time periods overlapping with the time periods for which there is a change in the newly updated plan.

e) IMPORTANT: NM insists that the B2B client only does a tactical update to a plan in case something has changed for that plan. Updates that do not change anything are logged. In

case too many of such non-updates are logged, then access can be restricted. This ensures that the backend responsible for the (pre-) tactical updates, does not get overloaded with non-updates.

(7) The `data id` is an opaque identifier of the version of the global state of the backend system related to CACD or tactical updatable related data (not pure flight data). Whenever `dataId` is passed in an update request, the system verifies if there have been conflicting updates between what the B2B client tried to update (wrt the state of the system linked to the dataid) and the latest state. Note that the `dataId` represent the global state of the backend system (not linked to specific locations). It changes continuously (between subsequent retrievePlan requests). However the fact that it changes continuously does not impact the B2B client, as it is only used to detect if there have been conflicting parallel updates between the latest state and what the B2B client changed in the update request.

(8) There are basically 2 ways (operationally) to use `dataId`:

a) In case B2C/NMOC (via phone coordination) will not be used anymore to update daily plans concerning the B2B client (nor in tactical, nor in forecast (PREDICT) and B2B is not used by any other systems to update the concerned daily plans and the master repository of the data is inside the B2B client's systems, then the B2B client can just before each update first do a query to retrieve the relevant daily plan (and its associated `dataId`.
So the detailed steps for a sector configuration update would be:

i) Retrieve the sector configuration plan for the AUA (Reply S1 includes dataId1)

ii) Produce the `sectorConfigurationPlanUpdateRequest` based on the client's local systems (using dataId1 and optionally merging/keeping S1 past data (if NM systems could/would be used as contingency repository and as such past data could have become different))

iii) `B2B.updateRunwayConfigurationPlan` with this sectorConfigurationPlanUpdate-Request. This usage pattern, would on each update systematically wipe out any changes applied via B2C/NMOC/other B2B systems). Note that in case NM systems are used as contingency repository, on restart the B2B client may want to import the data again into the local system.

b) In case the B2C/NMOC (via phone coordination) could still occasionally be used to update the concerned daily plans (e.g. contingency) or B2B is used by other systems to update the concerned daily plans or the B2B client system is not the master repository of the data (e.g. runway configurations that could be done by FMP or by tower), then the dataId (in combination with the CONFLICTING_UPDATE error reply) can be used to detect conflicting parallel updates and report those to the end user so that he can decide what to do.
So the detailed steps for a sector configuration update would be :

i) On the first update for a specific AUA A for a specific day X, the B2B client would first do a retrieveSectorConfigurationPlan to get a dataId A1. In the returned sectorConfigurationPlan there should not be any periods that have a data source tactical and that are different with the data in the local system. In case of differences, the operator is notified and needs to decide what values to choose.

ii) B2B.updateRunwayConfigurationPlan is used with dataId A1. The reply contains a dataId A2.

iii) Each next update for AUA A for day X, would use the dataId returned by the previous update (A2,A3,...).

iv) If the NM systems detect a conflicting parallel update between the time corresponding to the dataId A1/A2.. and the latest NM state wrt AUA A and day X, the reply contains a CONFLICTING_UPDATE error. In that case the client system would warn the operator that a conflicting parallel update has occurred and would show the local data and the NM data to allow the operator to choose (and optionally update any local system as well)

v) If AUA B for Day X is updated, then data Id B1 needs to be associated to AUA B . This usage pattern, would make sure that no changes are lost un-expectedly and the operator is in full control if updates are done via B2C or by the NM plan transfer of forecast data into operational.
Note that this second pattern is a bit more robust (operationally speaking) and can also handle the case were B2C is no longer used (point a above).
Note that it is this second pattern that a.o. NM systems use: when an AUA I is updated, first the NM data is shown and in the screen data the associated dataId I1 is kept. (the user has seen the NM data corresponding to dataId I1).
When the user applies his changes, this data Id I1 is then used in the `sectorConfugrationPlanUpdateRequest`. In case there were parallel conflicting updates, the user is notified and he needs to redo his update (including first looking at the latest NM data). The main reason behind: multiple operators at different terminals can do conflicting updates and they need to be notified.
Note that technically the B2B client can also use this pattern to detect parallel conflicting updates between different operators inside the client's own organization. (in case it is needed/useful).

3.2.1.15.3. Transactions and Errors

(1) If any change in the plan fails to be successfully processed by the NM system, the whole new plan is rejected so that the previous version of the plan remains unchanged.

(2) In some cases, the B2B layer can not do all validations involved in tactical updates. Some validations are done by the backend system. These errors are reported as a reply with ReplyStatus INVALID_INPUT and the error message string describes the problem. This error message string cannot be considered by the B2B client as part of the B2B contract (that string may change any time).

(3) As seen below, some error conditions are permanent in the sense that retrying the transaction later will never solve the problem (typically because the data in the request violates some validation rule), and a few others are temporary, e.g. the transaction failed because the NM system is itself in some temporary state that prevents it from processing the request (e.g. the daily plan is being transferred see below). In any case, the temporary error conditions are always well identified within the reply so that the B2B client knows when it is worth retrying a few minutes later NM insists

that the updates that failed due to a temporary condition are only retried a few minutes later, not a few seconds later and even less a few milliseconds later.

(4) Note that violations of the input constraints mentioned in the B2B model below will be reported in a structured way so that the B2B client will be able to decide what to do with them.

(5) Also note that permanent errors due to unknown references to airspace elements (like aerodrome ids, sector configurations ids, traffic volume ids) will NOT be returned formally. They are returned as a reply with ReplyStatus OBJECT_NOT_FOUND and an error message describing what attribute can not be found.

3.2.1.15.4. Allowed Times for Retrievals and Updates

3.2.1.15.4.1. Forecast and Operational Datasets

(1) The forecast and operational datasets are concepts that the NM customers (ANSPs in particular) are already familiar with. In short, the NM system prepares the plan between D-6 (6 days in advance) and D-1 (1 day in advance) within the forecast dataset, which is transferred to the operational dataset on D-1 around 16:00 UTC.

(2) The plan remains available in the forecast dataset after transfer, until the end of D (day of operations), even though it does not evolve anymore in that dataset.

(3) To summarize:

a) The plan can be updated:

i) In the forecast dataset: in [D-5 (5 days in the future), D-1 16:00 UTC]

ii) In the operational dataset: at any point in time on D-1 and D via B2B, D-1 updates are only allowed after the plan has been transferred (the ability to update the plan in the operational dataset before the plan has been transferred exists but is reserved to the NM OPS Room and in exceptional circumstances)

b) The plan can be retrieved:

i) From the forecast dataset: at any time in [D-5, D]

ii) From the operational dataset: at any point in time in [D-1, D].
The most up-to-date plan for a specific day X can be found in operational if the plan has already been transferred for day X. If the plan has not been transferred yet for day X, then the most up-to-date plan for day X can be found in forecast.

(4) Regarding the plan transfer, the updates to the forecast dataset are rejected from a cut-off time on D-1 that is 16:00 UTC; the NM OPS Room may then fine tune the plan for a little while, and finally transfers it to the operational dataset. The time elapsed between the cut-off time of the forecast dataset and the time at which the plan has been actually transferred to the operational dataset is typically around 10 minutes. This elapsed time may exceptionally be longer though, e.g. in case of unexpected change of the network situation (like unexpected strike or weather

conditions), so that the plan can exceptionally be transferred a few hours after the cut-off time.
The B2B client designer is invited to take this variability into consideration.
Updates to the plan, in the forecast or operational datasets, are rejected during this elapsed time.

(5) The B2B caller knows whether the plan has been transferred or not via the `planTransferred` attribute of a plan. Similarly, the B2B caller knows that the forecast update cut-off time has been reached or not via the `planCutOffReached` attribute of a plan.
This will be tentatively solved in NM 19.0.

3.2.1.15.4.2. Past

(1) A tactical update is rejected if it attempts to update the past. More precisely, the period associated to any update in the plan must start after the clock of the NM system. So when doing an tactical update at e.g. 13:00 for the remainder of day D, the clientschedule needs to also contain all the periods before 13:00 unaltered: i.e. as they were in a previously retrieved/send clientschedule. So all past periods marked with CACD need to be kept CACD (i.e. AIRSPACE datasource) and past all periods marked as tactically updated need to be kept tactically updated (TACTICAL datasource with the corresponding past data). Changing for a period, the datasource AIRSPACE (CACD) to TACTICAL is considered a change (because a TACTICAL updated period hides the CACD (AIRSPACE) data for that period and as such subsequent CACD changes are "ignored" for a tactically updated period)
Hence, an attempt to modify the past results in a permanent error.
This error will be formally identified as such if the modification is requested for D+1 (yesterday) or earlier.
But in case the attempt is made to modify the past within D, the resulting permanent error will not be formally identified as an attempt to update the past within D: the only way for the B2B client to investigate why the request failed will be to log and exploit the returned informal error message (string).

(2) In order to support changes on existing tactical updates, an existing period, starting in the past and ending in the future, can be split into a shorter period, starting at the same moment as the initial one but ending earlier, as long as the first new period still finishes in the future (i.e. the second new period starts in the future), and the data associated to the first new period is left unchanged.

(3) Retrievals in the past are limited to the past within D, i.e. the whole daily plan is returned for D but the B2B client cannot request the plans for D+1 or earlier.
This permanent error will be formally identified.

3.2.1.15.5. Simulations

(1) NM supports "flow simulation" features, where NM prepares simulations (airspace, traffic, strike scenarios). A simulation is basically a sandboxed environment containing some environment data, some traffic and a plan. It allows different actors to look at effects of a specific user scenario that is simulated and allows the different actors to interact with the simulation (for example: updating runway activations/sector configurations, regulations). In addition (selected) changes can be pushed back to the server.
There can be more than 1 such simulations ongoing at the same time, each evaluating different What-If scenarios. Each such simulation is identified by a simulationId.

(2) For this reason, the dataset types presented below are not limited to "forecast" and "operational" as indicated above, but include also "simulation" datasets. More details can be found below within the `DatasetType` and `Dataset` type descriptions.

(3) Note that there is no service yet that allows to retrieve the simulations and their simulationIds.

3.2.2. Query Complete AIXM data sets

3.2.2.1. SOAP

(1) The associated SOAP operation is:

```
CompleteAIXMDatasetReply queryCompleteAIXMDatasets(
    CompleteAIXMDatasetRequest request
    )
```

3.2.2.2. `CompleteAIXMDatasetRequest`

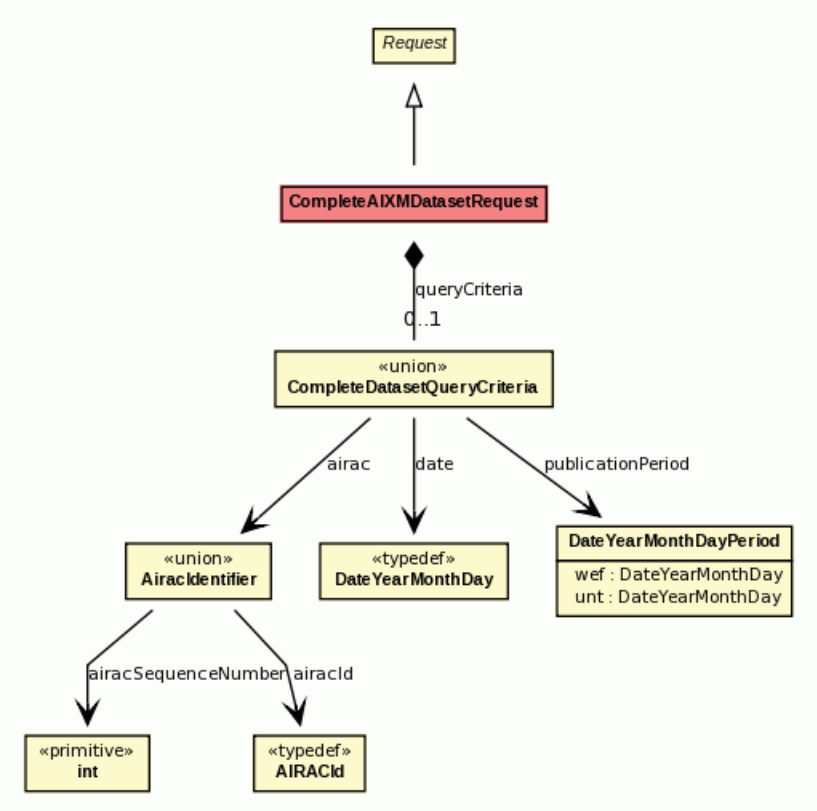

Figure 3.36. `CompleteAIXMDatasetRequest` Class Diagram

(1) Request to query complete AIXM Datasets.

(2) Inherits from: `Request`

(3) Attributes:

a) **`CompleteDatasetQueryCriteria queryCriteria`** *(Optional)*
The criteria by which to query for Complete AIXM Datasets. Default is any criteria.

3.2.2.3. `CompleteAIXMDatasetReply`

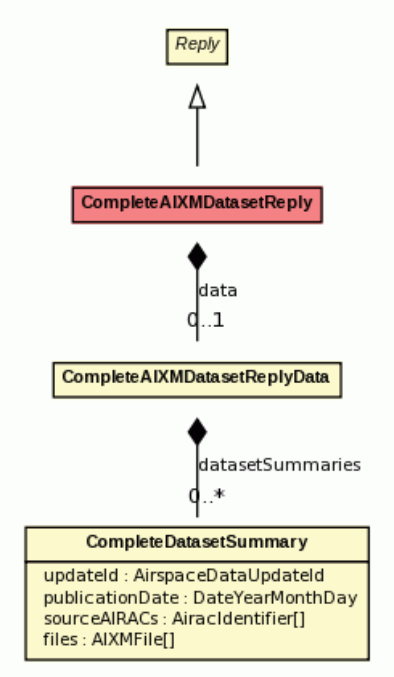

Figure 3.37. `CompleteAIXMDatasetReply` Class Diagram

(1) Reply returned in response to CompleteAIXMDatasetRequest

(2) The service returns a list of datasets available for downloads. More precisely it returns a list of CompleteDatasetSummary objects. Each summary contains relevant information about the dataset.

(3) A Complete AIXM Dataset is a set of AIXM 5.1 files, one per AIXM 5.1 feature type.

(4) Note that the service does not return the content of each file, but only the file identifiers. Each file must then be downloaded separately (see document for Common Services, FileService Port Type).

(5) When querying the datasets for a given AIRAC, the service returns all the Complete AIXM Datasets published for that AIRAC: this means from six days before the AIRAC switch until the end of the cycle.

(6) Past timeslices which are no longer relevant to the correspondent AIRACS are not included in the dataset.

(7) Inherits from: `Reply`

(8) Attributes:

a) **`CompleteDatasetSummary[] datasetSummaries`** *(Mandatory)*
The references to the requested ADR AIXM files.
Constraint: Size must be comprised between 0 and ∞.

3.2.3. Query Incremental AIXM data sets

3.2.3.1. SOAP

(1) The associated SOAP operation is:

```
IncrementalAIXMDatasetReply queryIncrementalAIXMDatasets(
    IncrementalAIXMDatasetRequest request
    )
```

3.2.3.2. `IncrementalAIXMDatasetRequest`

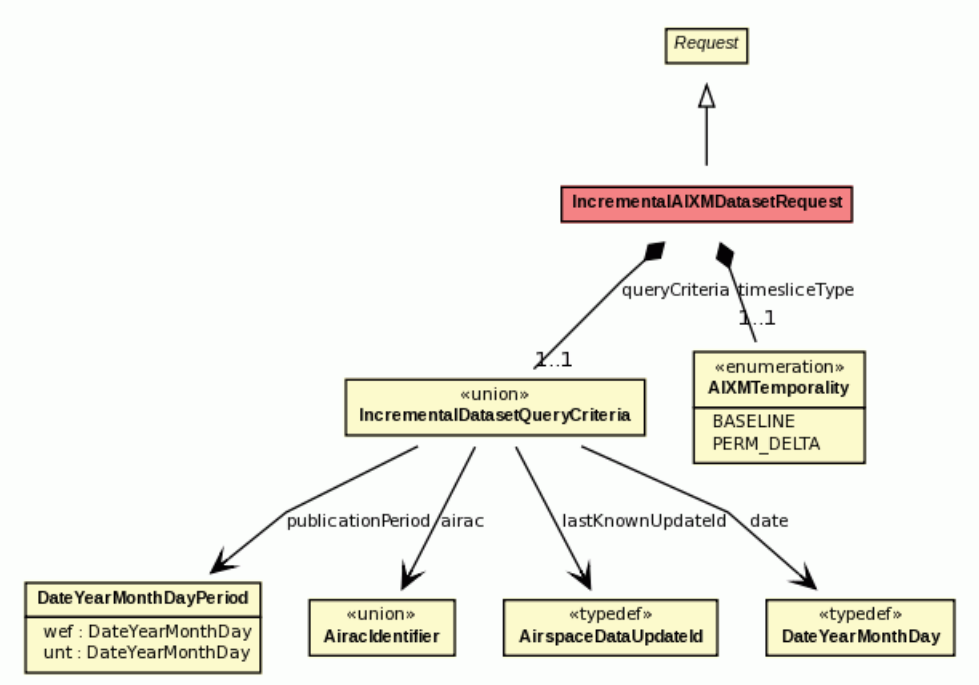

Figure 3.38. `IncrementalAIXMDatasetRequest` Class Diagram

(1) Request to query incremental AIXM Datasets.

(2) In the most common scenario the consumer provides the last known UpdateId and the service returns all available Incremental AIXM Datasets newer than the given UpdateId.

(3) The UpdateId provided by the caller must be a valid UpdateId previously obtained through:

a) either the download of a Complete AIXM Dataset.

b) or the retrieval of an Incremental AIXM Dataset.

(4) The method also allows using other query parameters in order not to limit the possible use cases.

(5) Inherits from: `Request`

(6) Attributes:

a) **`IncrementalDatasetQueryCriteria` `queryCriteria`** *(Mandatory)*
The criteria by which to query for Incremental AIXM Datasets.

b) **`AIXMTemporality` `timesliceType`** *(Mandatory)*
It allows the data consumer to choose whether to get the dataset as BASELINE or PERM-DELTA timeslices.

3.2.3.3. `IncrementalAIXMDatasetReply`

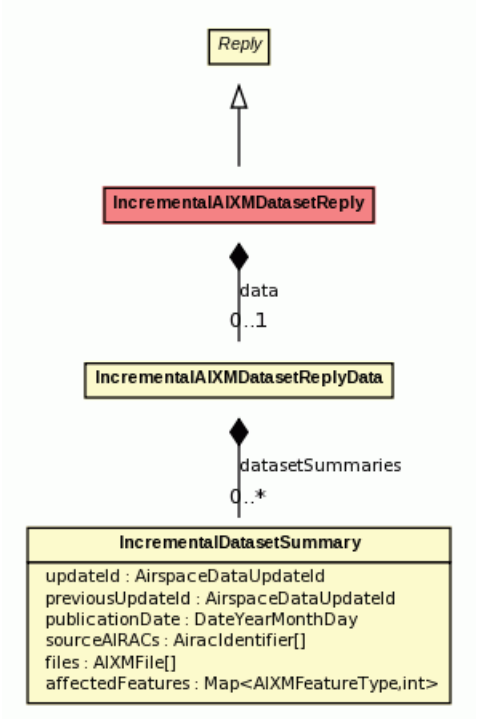

Figure 3.39. `IncrementalAIXMDatasetReply` Class Diagram

(1) Reply returned in response to IncrementalAIXMDatasetRequest

(2) The service returns a list of available Incremental AIXM Datasets. More precisely it returns a list of Incremental AIXM Dataset Summaries.

(3) Inherits from: `Reply`

(4) Attributes:

a) **`IncrementalDatasetSummary[]` `datasetSummaries`** *(Mandatory)*
A collection of summaries of the available Incremental AIXM Datasets.

Constraint: Size must be comprised between 0 and ∞.

3.2.4. Retrieve a Sector Configuration Plan

3.2.4.1. SOAP

(1) The associated SOAP operation is:

```
SectorConfigurationPlanRetrievalReply retrieveSectorConfigurationPlan(
    SectorConfigurationPlanRetrievalRequest request
    )
```

3.2.4.2. `SectorConfigurationPlanRetrievalRequest`

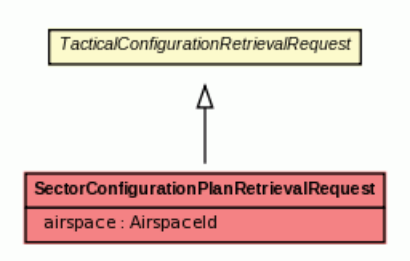

Figure 3.40. `SectorConfigurationPlanRetrievalRequest` Class Diagram

(1) Request to retrieve the sector configuration plan for a given AUA or sector cluster on a given day.

(2) Inherits from: `TacticalConfigurationRetrievalRequest`

(3) Attributes:

a) **`AirspaceId` `airspace`** *(Mandatory)*
AUA or sector cluster for which the sector configuration plan is requested.

3.2.4.3. `SectorConfigurationPlanRetrievalReply`

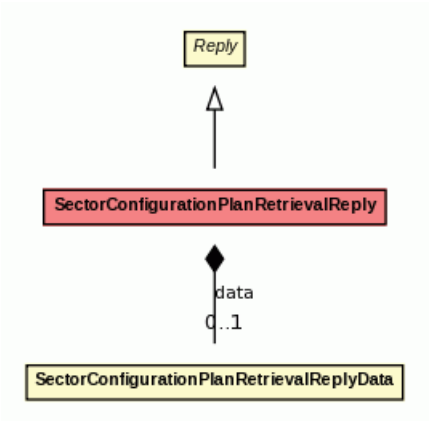

Figure 3.41. `SectorConfigurationPlanRetrievalReply` Class Diagram

(1) Reply to a SectorConfigurationPlanRetrievalRequest.

(2) Special error conditions:

(3) a) `OBJECT_NOT_FOUND` If the airspace is neither an AUA, nor a sector cluster, or if there is no sector configuration defined in CACD for this AUA/sector cluster.

(4) Inherits from: `Reply`

(5) Attributes:

a) **`SectorConfigurationPlan` `plan`** *(Mandatory)*
The complete sector configuration plan for a given AUA or sector cluster on a given day.

3.2.5. Update a Sector Configuration Plan

3.2.5.1. SOAP

(1) The associated SOAP operation is:

```
SectorConfigurationPlanUpdateReply updateSectorConfigurationPlan(
    SectorConfigurationPlanUpdateRequest request
    )
```

3.2.5.2. `SectorConfigurationPlanUpdateRequest`

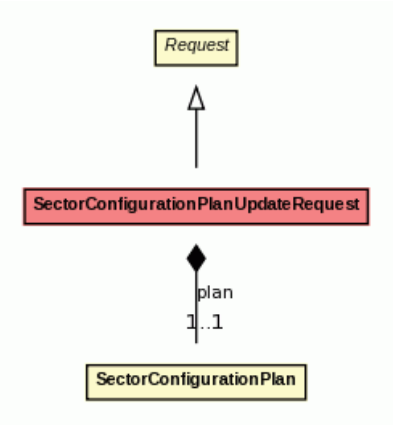

Figure 3.42. `SectorConfigurationPlanUpdateRequest` Class Diagram

(1) Request to update the sector configuration plan for a given AUA or sector cluster on a given day.

(2) Inherits from: `Request`

(3) Attributes:

a) **`SectorConfigurationPlan` `plan`** *(Mandatory)*
Sector configuration plan reflecting the requested updates.

3.2.5.3. `SectorConfigurationPlanUpdateReply`

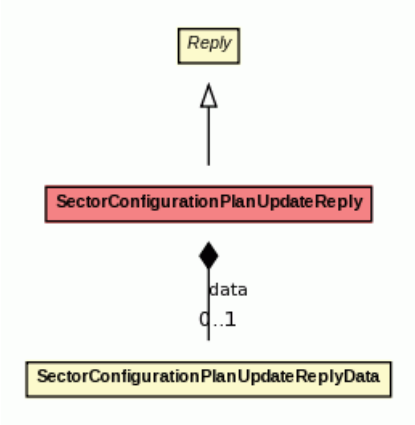

Figure 3.43. `SectorConfigurationPlanUpdateReply` Class Diagram

(1) Reply to a SectorConfigurationPlanUpdateRequest.

(2) Special error conditions:

(3) a) `INVALID_DATASET` [Temporary error] If the FORECAST update was rejected due to D-1 forecast update cut-off or if the OPERATIONAL update was rejected due to D-1 plan not transferred yet.

b) `CONFLIGTING_UPDATE` [Permanent error] If the update failed due to incompatible concurrent changes.

c) `OBJECT_NOT_FOUND` [Permanent error] If the airspace is neither an AUA, nor a sector cluster, or if there is no sector configuration defined in CACD for this AUA/sector cluster.

d) Note that there are also the INVALID_INPUT replyStatus errors, that cover the errors detected by the backend (See Transaction and Errors). For example : errors when modifying the past (See Airspace Past)

(4) Inherits from: `Reply`

(5) Attributes:

a) **`SectorConfigurationPlan plan`** *(Mandatory)*
The complete sector configuration plan for a given AUA or sector cluster on a given day, resulting from the update.

3.2.6. Retrieve a Capacity Plan

3.2.6.1. SOAP

(1) The associated SOAP operation is:

```
CapacityPlanRetrievalReply retrieveCapacityPlan(
    CapacityPlanRetrievalRequest request
    )
```

3.2.6.2. `CapacityPlanRetrievalRequest`

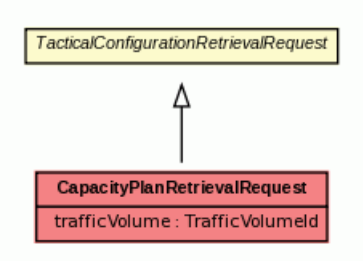

Figure 3.44. `CapacityPlanRetrievalRequest` Class Diagram

(1) Request to retrieve the capacity plan for a given traffic volume on a given day.

(2) Inherits from: `TacticalConfigurationRetrievalRequest`

(3) Attributes:

a) **`TrafficVolumeId` `trafficVolume`** *(Mandatory)*
The traffic volume for which the capacity plan is requested.

3.2.6.3. `CapacityPlanRetrievalReply`

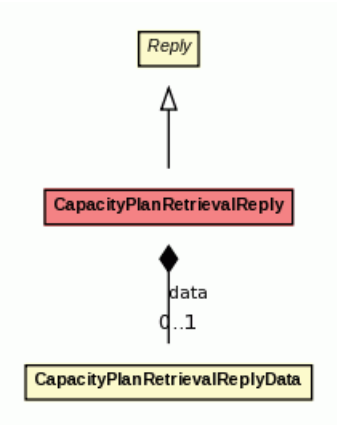

Figure 3.45. `CapacityPlanRetrievalReply` Class Diagram

(1) Reply to a CapacityPlanRetrievalRequest.

(2) Inherits from: `Reply`

(3) Attributes:

a) **`CapacityPlan` `plan`** *(Mandatory)*

The complete capacity plan for a given traffic volume on a given day.

3.2.7. Update a Capacity Plan

3.2.7.1. SOAP

(1) The associated SOAP operation is:

```
CapacityPlanUpdateReply updateCapacityPlan(
    CapacityPlanUpdateRequest request
    )
```

3.2.7.2. `CapacityPlanUpdateRequest`

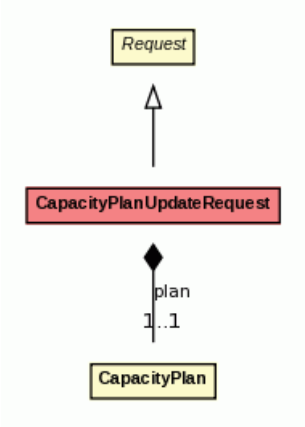

Figure 3.46. `CapacityPlanUpdateRequest` Class Diagram

(1) Request to update the capacity plan for a given traffic volume on a given day.

(2) Inherits from: `Request`

(3) Attributes:

a) **`CapacityPlan` `plan`** *(Mandatory)*
Capacity plan reflecting the requested updates.

3.2.7.3. `CapacityPlanUpdateReply`

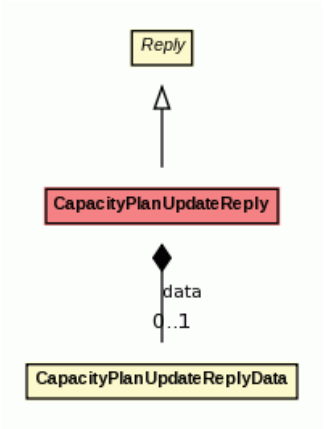

Figure 3.47. `CapacityPlanUpdateReply` Class Diagram

(1) Reply to a CapacityPlanUpdateRequest.

(2) Special error conditions:

(3) a) `INVALID_DATASET` [Temporary error] If the FORECAST update was rejected due to D-1 forecast update cut-off or if the OPERATIONAL update was rejected due to D-1 plan not transferred yet.

b) `CONFLIGTING_UPDATE` [Permanent error] If the update failed due to incompatible concurrent changes.

c) `OBJECT_NOT_FOUND` [Permanent error] If the Traffic Volume is not known.

d) Note that there are also the INVALID_INPUT replyStatus errors, that cover the errors detected by the backend (See Transaction and Errors). For example : errors when modifying the past (See Airspace Past)

(4) Inherits from: `Reply`

(5) Attributes:

a) **`CapacityPlan plan`** *(Mandatory)*
The complete capacity plan for a given traffic volume on a given day, resulting from the update.

3.2.8. Retrieve a Traffic Volume Activation Plan

3.2.8.1. SOAP

(1) The associated SOAP operation is:

```
TrafficVolumeActivationPlanRetrievalReply retrieveTrafficVolumeActivationPlan(
    TrafficVolumeActivationPlanRetrievalRequest request
    )
```

3.2.8.2. `TrafficVolumeActivationPlanRetrievalRequest`

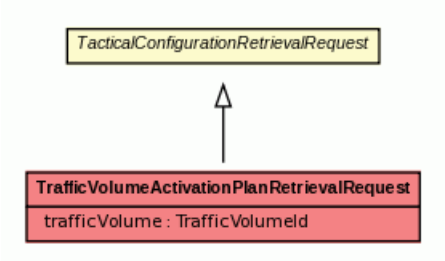

Figure 3.48. `TrafficVolumeActivationPlanRetrievalRequest` Class Diagram

(1) Request to retrieve the traffic volume activation plan for a given traffic volume on a given day.

(2) Inherits from: `TacticalConfigurationRetrievalRequest`

(3) Attributes:

a) **`TrafficVolumeId` `trafficVolume`** *(Mandatory)*
The traffic volume for which the activation plan is requested.

3.2.8.3. `TrafficVolumeActivationPlanRetrievalReply`

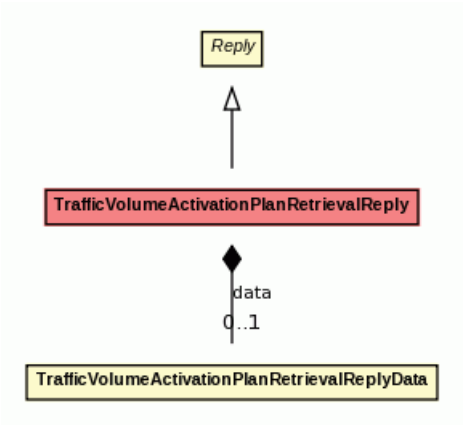

Figure 3.49. `TrafficVolumeActivationPlanRetrievalReply` Class Diagram

(1) Reply to a TrafficVolumeActivationPlanRetrievalRequest.

(2) Inherits from: `Reply`

(3) Attributes:

a) **`TrafficVolumeActivationPlan` `plan`** *(Mandatory)*

The complete activation plan for a given traffic volume on a given day.

3.2.9. Update a Traffic Volume Activation Plan

3.2.9.1. SOAP

(1) The associated SOAP operation is:

```
TrafficVolumeActivationPlanUpdateReply updateTrafficVolumeActivationPlan(
    TrafficVolumeActivationPlanUpdateRequest request
    )
```

3.2.9.2. `TrafficVolumeActivationPlanUpdateRequest`

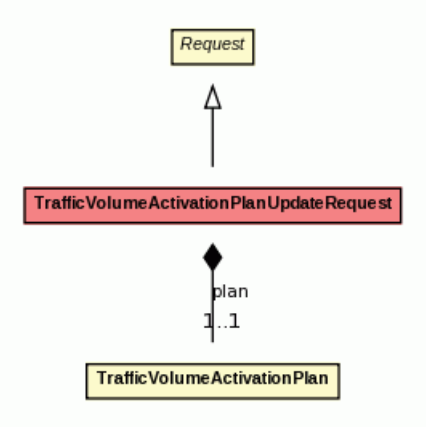

Figure 3.50. `TrafficVolumeActivationPlanUpdateRequest` Class Diagram

(1) Request to update the traffic volume activation plan for a given traffic volume on a given day.

(2) Inherits from: `Request`

(3) Attributes:

a) **`TrafficVolumeActivationPlan` `plan`** *(Mandatory)*
Traffic volume activation plan reflecting the requested updates.

3.2.9.3. `TrafficVolumeActivationPlanUpdateReply`

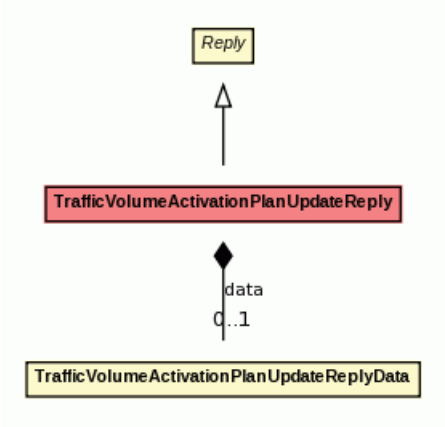

Figure 3.51. `TrafficVolumeActivationPlanUpdateReply` Class Diagram

(1) Reply to a TrafficVolumeActivationPlanUpdateRequest.

(2) Special error conditions:

(3) a) `INVALID_DATASET` [Temporary error] If the FORECAST update was rejected due to D-1 forecast update cut-off or if the OPERATIONAL update was rejected due to D-1 plan not transferred yet.

b) `CONFLIGTING_UPDATE` [Permanent error] If the update failed due to incompatible concurrent changes.

c) `OBJECT_NOT_FOUND` [Permanent error] If the Traffic Volume is not known.

d) Note that there are also the INVALID_INPUT replyStatus errors, that cover the errors detected by the backend (See Transaction and Errors). For example : errors when modifying the past (See Airspace Past)

(4) Inherits from: `Reply`

(5) Attributes:

a) **`TrafficVolumeActivationPlan plan`** *(Mandatory)*
The complete traffic volume activation plan for a given traffic volume on a given day, resulting from the update.

3.2.10. Retrieve an OTMV Plan

3.2.10.1. SOAP

(1) The associated SOAP operation is:

```
OTMVPlanRetrievalReply retrieveOTMVPlan(
    OTMVPlanRetrievalRequest request
    )
```

3.2.10.2. `OTMVPlanRetrievalRequest`

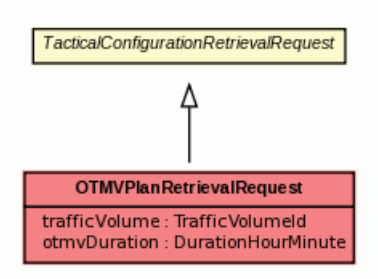

Figure 3.52. `OTMVPlanRetrievalRequest` Class Diagram

(1) Request to retrieve for a given day, the OTMV plans for a given traffic volume (for all applicable OTMV durations) or to retrieve the OTMV plan for a specific (traffic volume, OTMV duration) pair.

(2) Inherits from: `TacticalConfigurationRetrievalRequest`

(3) Attributes:

a) **`TrafficVolumeId` `trafficVolume`** *(Mandatory)*
The traffic volume for which the OTMV plan is requested.

b) **`DurationHourMinute` `otmvDuration`** *(Optional)*
Selects the OTMVs applying for the given traffic volume according to their OTMV duration. When not specified, all duration are considered.

3.2.10.3. `OTMVPlanRetrievalReply`

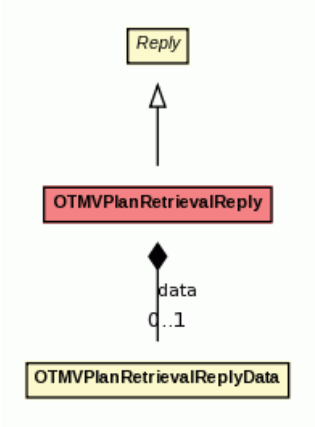

Figure 3.53. `OTMVPlanRetrievalReply` Class Diagram

(1) Reply to a OTMVPlanRetrievalRequest.

(2) Special error conditions:

(3) a) `OBJECT_NOT_FOUND` If the Traffic Volume is not known or if the associated reference location is not an airspace.

(4) Inherits from: `Reply`

(5) Attributes:

a) **`OTMVPlan` `plan`** *(Mandatory)*
The complete OTMV plan for a given (traffic volume, OTMV duration) pair on a given day.

3.2.11. Update an OTMV Plan

3.2.11.1. SOAP

(1) The associated SOAP operation is:

```
OTMVPlanUpdateReply updateOTMVPlan(
    OTMVPlanUpdateRequest request
    )
```

3.2.11.2. `OTMVPlanUpdateRequest`

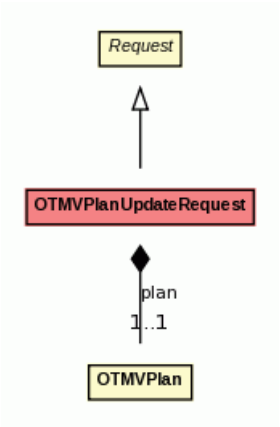

Figure 3.54. `OTMVPlanUpdateRequest` Class Diagram

(1) Request to update the OTMV plan for a given (traffic volume, OTMV duration) pair on a given day.

(2) Inherits from: `Request`

(3) Attributes:

a) **`OTMVPlan` `plan`** *(Mandatory)*
OTMV plan reflecting the requested updates.

3.2.11.3. `OTMVPlanUpdateReply`

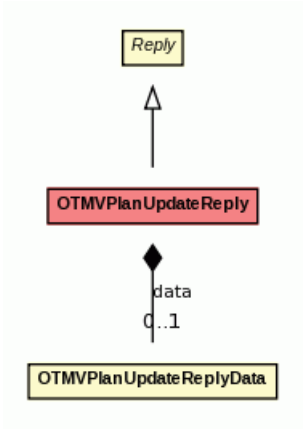

Figure 3.55. `OTMVPlanUpdateReply` Class Diagram

(1) Reply to a OTMVPlanUpdateRequest.

(2) Special error conditions:

(3) a) `INVALID_DATASET` [Temporary error] If the FORECAST update was rejected due to D-1 forecast update cut-off or if the OPERATIONAL update was rejected due to D-1 plan not transferred yet.

b) `CONFLIGTING_UPDATE` [Permanent error] If the update failed due to incompatible concurrent changes.

c) `OBJECT_NOT_FOUND` [Permanent error] If the Traffic Volume is not known or if the associated reference location is not an airspace.

d) Note that there are also the INVALID_INPUT replyStatus errors, that cover the errors detected by the backend (See Transaction and Errors). For example : errors when modifying the past (See Airspace Past)

(4) Inherits from: `Reply`

(5) Attributes:

a) **`OTMVPlan plan`** *(Mandatory)*
The complete OTMV plan for a given (traffic volume, OTMV duration) pair on a given day, resulting from the update.

3.2.12. Retrieve a runway configuration plan

3.2.12.1. SOAP

(1) The associated SOAP operation is:

```
RunwayConfigurationPlanRetrievalReply retrieveRunwayConfigurationPlan(
    RunwayConfigurationPlanRetrievalRequest request
    )
```

3.2.12.2. `RunwayConfigurationPlanRetrievalRequest`

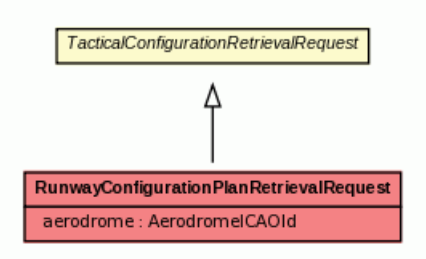

Figure 3.56. `RunwayConfigurationPlanRetrievalRequest` Class Diagram

(1) Request to retrieve the runway configuration plan for a given aerodrome on a given day.

(2) Inherits from: `TacticalConfigurationRetrievalRequest`

(3) Attributes:

a) **`AerodromeICAOId aerodrome`** *(Mandatory)*
The aerodrome for which the runway configuration plan is requested.

3.2.12.3. `RunwayConfigurationPlanRetrievalReply`

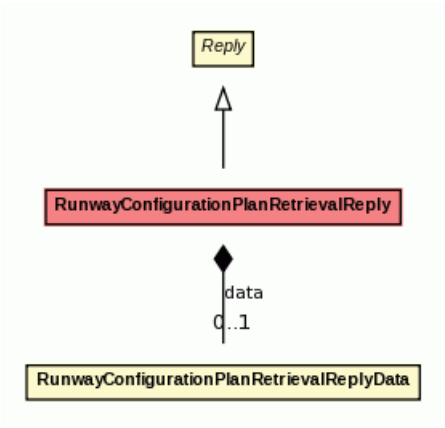

Figure 3.57. `RunwayConfigurationPlanRetrievalReply` Class Diagram

(1) Reply to a RunwayConfigurationPlanRetrievalRequest.

(2) Inherits from: `Reply`

(3) Attributes:

a) **`RunwayConfigurationPlan plan`** *(Mandatory)*

The complete runway configuration plan for a given aerodrome on a given day.

3.2.13. Update a runway configuration plan

3.2.13.1. SOAP

(1) The associated SOAP operation is:

```
RunwayConfigurationPlanUpdateReply updateRunwayConfigurationPlan(
    RunwayConfigurationPlanUpdateRequest request
    )
```

3.2.13.2. `RunwayConfigurationPlanUpdateRequest`

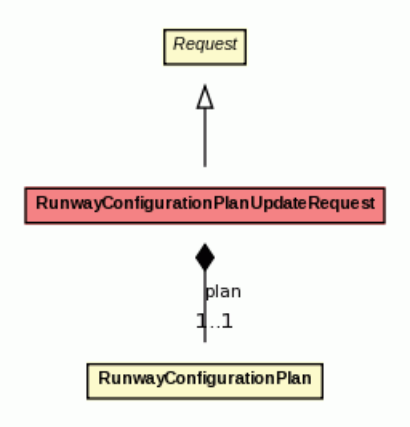

Figure 3.58. `RunwayConfigurationPlanUpdateRequest` Class Diagram

(1) Request to update the runway configuration plan for a given aerodrome on a given day.

(2) Inherits from: `Request`

(3) Attributes:

a) **`RunwayConfigurationPlan` `plan`** *(Mandatory)*
Runway configuration plan reflecting the requested updates.

3.2.13.3. `RunwayConfigurationPlanUpdateReply`

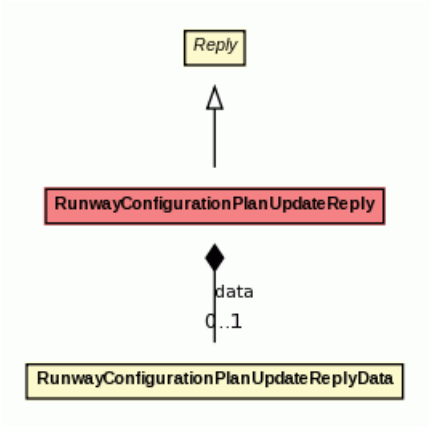

Figure 3.59. `RunwayConfigurationPlanUpdateReply` Class Diagram

(1) Reply to a RunwayConfigurationPlanUpdateRequest.

(2) Special error conditions:

(3) a) `INVALID_DATASET` [Temporary error] If the FORECAST update was rejected due to D-1 forecast update cut-off or if the OPERATIONAL update was rejected due to D-1 plan not transferred yet.

b) `CONFLIGTING_UPDATE` [Permanent error] If the update failed due to incompatible concurrent changes.

c) `OBJECT_NOT_FOUND` [Permanent error] If the aerodrome is not known.

d) `INVALID_INPUT` [Permanent error] If an unknown runway for this aerodrome is used , or if not all runways are specified for TACTICAL updated periods..

e) Note that there are also the INVALID_INPUT replyStatus errors, that cover the errors detected by the backend (See Transaction and Errors). For example : errors when modifying the past (See Airspace Past)

(4) Inherits from: `Reply`

(5) Attributes:

a) **`RunwayConfigurationPlan plan`** *(Mandatory)*
The complete runway configuration plan for a given aerodrome on a given day.

Chapter 4. Data Types

4.1. <<abstract>> AbstractEAUPCDRRequest

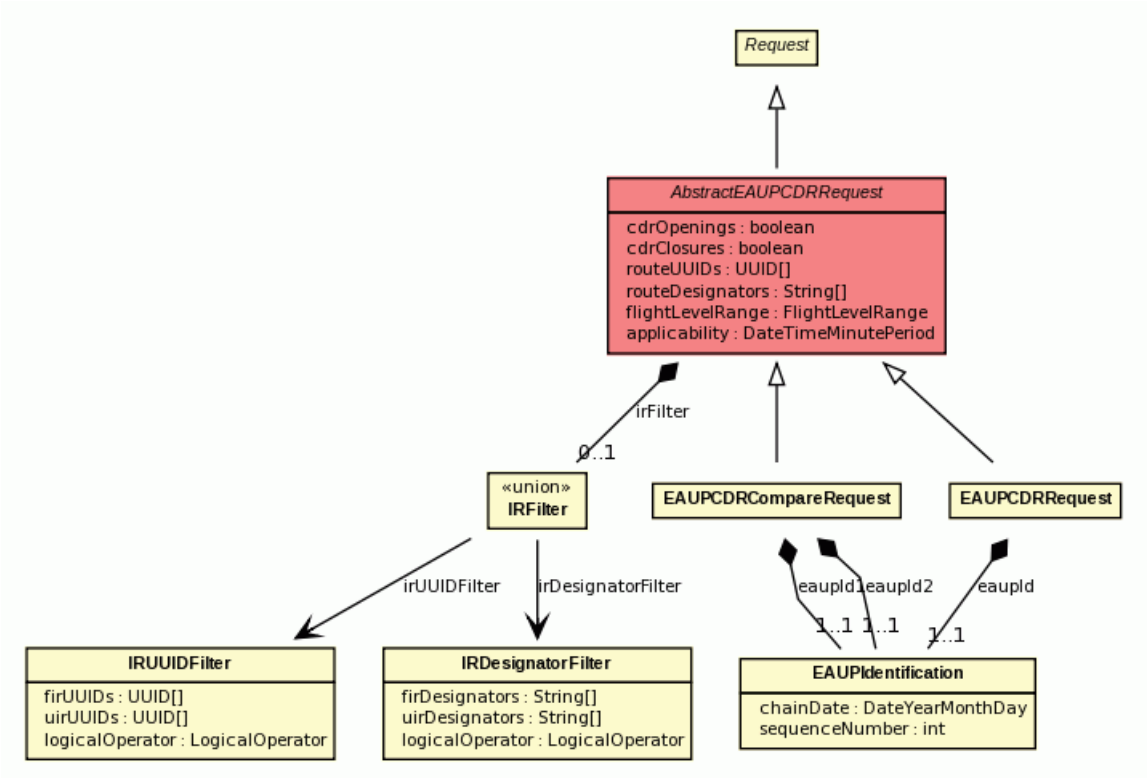

Figure 4.1. <<abstract>> AbstractEAUPCDRRequest Class Diagram

(1) Used to retrieve the CDR openings and closures within an EAUP, or between EAUPs, while possibly applying a filter on the returned result set, i.e. keep only the CDR openings/closures for:

(2)
a) A CDR type (i.e. opening and/or closure), and

b) A list of route wildcards, and

c) A list of FIR and/or UIR UUIDs, and

d) A list of FIR and/or UIR designator wildcards, and

e) A flight level range, and

f) An applicability period

(3) The logical AND operator applies between all the query attributes described below.

(4) Since released EAUPs are immutable, i.e. their contents will not be modified anymore, NM requires its customers to undertake their best effort to avoid repeatedly launching the same requests on the same EAUP.

(5) Inherits from: `Request`.

(6) Attributes:

a) **`boolean cdrOpenings`** *(Optional)*
If true, CDR openings are returned. True by default.

b) **`boolean cdrClosures`** *(Optional)*
If true, CDR closures are returned. True by default.

c) **`UUID[] routeUUIDs`** *(Optional)*
Query attribute on route UUIDs. The default meaning is "all route UUIDs".
Constraints:

i) Size must be comprised between 1 and ∞.

ii) See `ROUTE_UUIDS_CANNOT_CONTAIN_DUPLICATE`

d) **`string[] routeDesignators`** *(Optional)*
Query attribute on route designators. Each string item in the array can be a full route designator or a wildcard for a route designator. Supported wildcards are limited to at least one character and the star sign ("*") at the end of the expression. The default meaning is "all routes".
Constraints:

i) Size must be comprised between 1 and ∞.

ii) Item Pattern: `(UALPHA|DIGIT){1,7}|(UALPHA|DIGIT){1,6}*`

iii) See `ROUTE_DESIGNATORS_CANNOT_CONTAIN_DUPLICATE`

e) **`IRFilter irFilter`** *(Optional)*
Used to filter the IR airspaces on which CDR openings/closures apply, based on UUIDs or on IR designators.

f) **`FlightLevelRange flightLevelRange`** *(Optional)*
Query attribute on flight level range. The CDR opening/closure matches this query attribute if its flight level range and the given flight level range overlap. Be aware that the FlightLevelRange is right-opened, i.e. if e.g. a CDR opening flight level range is [300, 400 [and the caller queries on flight level range [400, 500 [, the CDR opening does not match the query. The default meaning is "any flight level range".

g) **`DateTimeMinutePeriod applicability`** *(Optional)*
Query attribute on CDR opening/closure applicability period. The CDR opening/closure matches this query attribute if its applicability period and the given applicability period overlap. Note that time period are left-closed and right-opened, i.e. no match if obtained if the CDR opening/closure applicability period starts at the time corresponding to the end of the query attribute. The default meaning is "any applicability".

(7) Constraints:

a)

Name	ROUTE_UUIDS_CANNOT_CONTAIN_DUPLICATE
Attribute	routeUUIDs
Description	If specified, the array cannot be empty and does not accept duplicates.

b)

Name	ROUTE_DESIGNATORS_CANNOT_CONTAIN_DUPLICATE
Attribute	routeDesignators
Description	Query attribute on route designators. Each string item in the array can be a full route designator or a wildcard for a route designator. Supported wildcards are limited to at least one character and the star sign ("*") at the end of the expression. The default meaning is "all routes".

(8) Extended by: EAUPCDRRequest, EAUPCDRCompareRequest.

4.2. <<abstract>> AbstractEAUPRSARequest

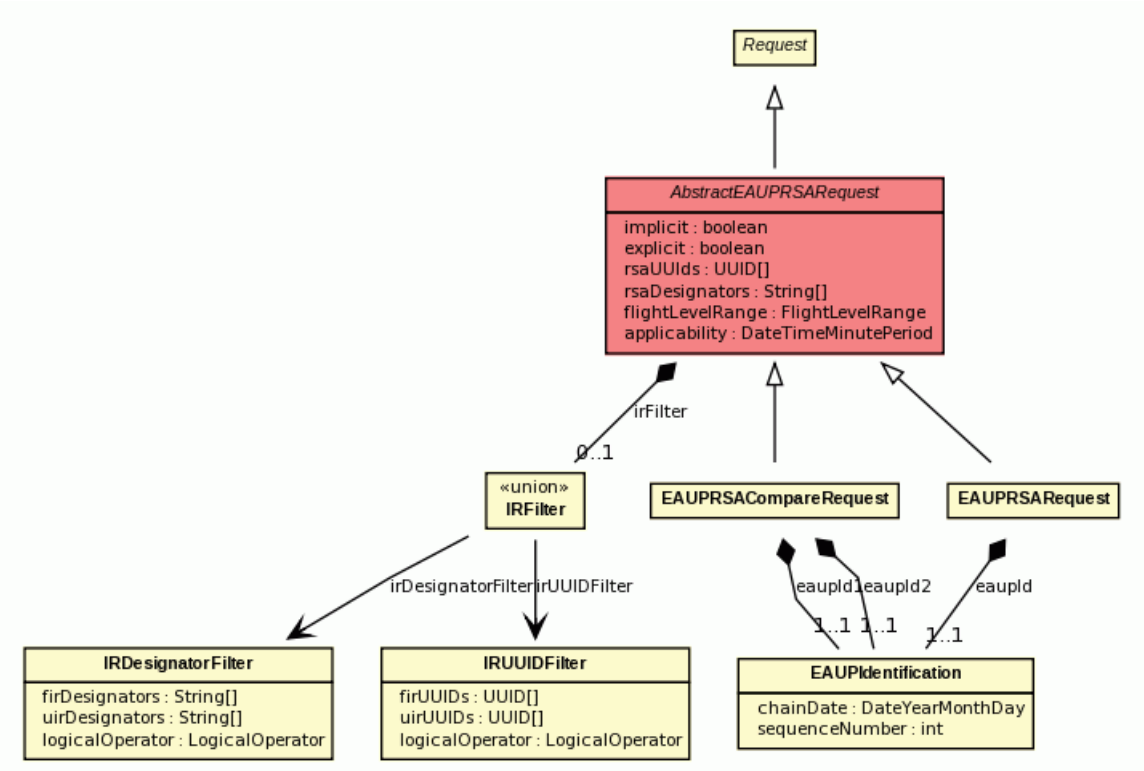

Figure 4.2. <<abstract>> AbstractEAUPRSARequest Class Diagram

(1) Used to retrieve the RSA allocations within an EAUP, or between EAUPs, while possibly applying a filter on the returned result set, i.e. keep only the RSA allocations for:

(2) a) An RSA allocation type (i.e. implicit and/or explicit), and

b) A list of RSA wildcards, and

c) A list of FIR and/or UIR UUIDs, and

d) A list of FIR and/or UIR designator wildcards, and

e) A flight level range, and

f) An applicability period

(3) The logical AND operator applies between all the query attributes described below.

(4) Since released EAUPs are immutable, i.e. their contents will not be modified anymore, NM requires its customers to undertake their best effort to avoid repeatedly launching the same requests on the same EAUP.

(5) Inherits from: `Request`.

(6) Attributes:

a) **`boolean implicit`** *(Optional)*
If true, implicit RSA allocations are returned. True by default.

b) **`boolean explicit`** *(Optional)*
If true, explicit RSA allocations are returned. True by default.

c) **`UUID[] rsaUUIds`** *(Optional)*
Query attribute on RSA UUIDs. The default meaning is "all RSA UUIDs".
Constraints:

i) Size must be comprised between 1 and ∞.

ii) See `RSA_UUIDS_CANNOT_CONTAIN_DUPLICATE`

d) **`string[] rsaDesignators`** *(Optional)*
Query attribute on RSA designators, i.e. ICAO-compliant RSA designator. Each string item in the array can be a full RSA designator or a wildcard for a RSA designator. Supported wildcards are limited to at least one character and the star sign ("*") at the end of the expression. The default meaning is "all RSAs".
Constraints:

i) Size must be comprised between 1 and ∞.

ii) Item Pattern: `(UALPHA|DIGIT){1,7}|(UALPHA|DIGIT){1,6}*`

iii) See `RSA_DESIGNATORS_CANNOT_CONTAIN_DUPLICATE`

e) **`IRFilter irFilter`** *(Optional)*
Used to filter the IR airspaces on which RSA allocations apply, based on UUIDs or on IR designators.

f) **FlightLevelRange flightLevelRange** *(Optional)*
Query attribute on flight level range. The RSA allocation matches this query attribute if its flight level range and the given flight level range overlap. Be aware that the FlightLevelRange is right-opened, i.e. if e.g. an RSA allocation flight level range is [300, 400 [and the caller queries on flight level range [400, 500 [, the RSA allocation does not match the query. The default meaning is "any flight level range".

g) **DateTimeMinutePeriod applicability** *(Optional)*
Query attribute on RSA allocation applicability period. The RSA allocation matches this query attribute if its applicability period and the given applicability period overlap. Note that time period are left-closed and right-opened, i.e. no match if obtained if the RSA allocation applicability period starts at the time corresponding to the end of the query attribute. The default meaning is "any applicability".

(7) Constraints:

a)

Name	RSA_UUIDS_CANNOT_CONTAIN_DUPLICATE
Attribute	rsaUUIds
Description	If specified, the array cannot be empty and does not accept duplicates.

b)

Name	RSA_DESIGNATORS_CANNOT_CONTAIN_DUPLICATE
Attribute	rsaDesignators
Description	If specified, the array cannot be empty and does not accept duplicates.

(8) Extended by: EAUPRSARequest, EAUPRSACompareRequest.

4.3. typedef<eurocontrol.cfmu.cua.b2b.aixm.ADRMessage> ADRMessageType

(1) This is an envelope of Features. The envelope can be empty. Depending on the service, the envelope can contain different types of Features.

(2) Used by: EAUPRSAReply, AUPComputedEntries, EAUPRSACompareReply, EAUPCDRReply, EAUPMessage, AUPGetManageableRouteSegmentsForAMCAndRouteReply, AUPRSAAllocationExpansionReply, AUPManualEntries, AUPRSAAllocationExpansionRequest, EAUPCDRCompareReply.

4.4. typedef<string> AerodromeIATAId

(1) IATA identifier of an aerodrome.

(2) Examples:BRU, CDG,...

(3) Pattern: UALPHA{3}

(4) Used by: FlightIATAId.

4.5. typedef<string> AerodromeIATAOrICAOId

(1) IATA (3 characters) or ICAO (4 characters) identifier of an aerodrome.

(2) Pattern: `UALPHA{3,4}`

(3) Used by: ACC3Accreditation, AerodromeIdAndRole.

4.6. typedef<string> AerodromeICAOId

(1) ICAO id of an Aerodrome.

(2) Pattern: `UALPHA{4}`

(3) Used by: AlternateAerodrome, AerodromeDAL, RunwayConfigurationPlanRetrievalRequest, FlightPointRoute, FourDTrajectoryPoint, Flight, Aerodrome, AirportPlanningInformationRequest, FlightKeys, ReferenceLocationAerodrome, ReclearanceInFlight, FlightPoint, FlightListByAerodromeRequest, RunwayConfigurationPlan, TrafficCountsByAerodromeRequest.

4.7. typedef<string> AerodromeSetId

(1) NM unique id of the AerodromeSet.

(2) Pattern: `ANY{1,8}`

(3) Used by: FlightListByAerodromeSetRequest, TrafficCountsByAerodromeSetRequest, ReferenceLocationAerodromeSet.

4.8. typedef<string> AIRACId

(1) Identifier of the AIRAC based on the year and a sequence number in the year.

(2) Its format is "YYSS": where YY are the last two digits of the year and SS the two digits for the AIRAC sequence number within the year. Example: "1203" is the third AIRAC of 2012.

(3) Pattern: `DIGIT{4}`

(4) Used by: AiracIdentifier.

4.9. AiracIdentifier

(1) Identification of a airac, either via a airac id or via a airac sequence number.

(2) Choices:

a) **AIRACId airacId**
Specific AIRAC id.

b) **`int airacSequenceNumber`**
Specific AIRAC sequence number.

(3) Used by: CompleteDatasetSummary, IncrementalDatasetQueryCriteria, CompleteDatasetQueryCriteria, IncrementalDatasetSummary.

4.10. `typedef<long> AirspaceDataUpdateId`

(1) Airspace data update identifier.

(2) Range: `]- ∞,∞[`.

(3) Used by: CompleteDatasetSummary, IncrementalDatasetQueryCriteria, IncrementalDatasetSummary.

4.11. `typedef<string> AirspaceId`

(1) NM unique id of the Airspace.

(2) Pattern: `ANY{1,12}`

(3) Used by: AvoidViaAirspaceReroutingConstraint, ReferenceLocationAirspace, SectorConfigurationPlanRetrievalRequest, FlightAirspace, FlightListByAirspaceRequest, RoutingAssistanceRequest, TrafficCountsByAirspaceRequest, SectorConfigurationPlan, DeltaEntry.

4.12. `<<enumeration>> AirspaceType`

(1) Enumerates airspace types

(2) Values:

Value	Description
`AOI`	Area Of Interest
`AOP`	Area Of Protection
`AREA`	NM or Eurocontrol Defined Area
`AUA`	ATC Unit Airspace
`AUAG`	ATC Unit Airspace Group
`CDA`	Client Defined Airspace
`CLUS`	Cluster
`CRSA`	Composed Manageable Airspace.
`CS`	Collapsed Sector
`ERSA`	Elementary Manageable Airspace
`ES`	Elementary Sector
`FIR`	Flight Information Region

Value	Description
IFPZ	IFPS Zone
NAS	National Airspace
REG	Region

Table 4.1. <<enumeration>> AirspaceType

(3) Used by: FlightAirspace.

4.13. AirSpeed

(1) Representation of a true Airspeed, together with its unit.

(2) Attributes:

a) **AirSpeed_DataType speed** *(Mandatory)*
The true airspeed value, in the provided unit. Must be in [0, 9999].

b) **SpeedUnit unit** *(Mandatory)*
Speed unit.

(3) Used by: Flight, FourDTrajectoryPoint, WindVector, AirSpeedOrInitial.

4.14. typedef<int> AirSpeed_DataType

(1) Airspeed data type.

(2) Range:] - ∞,∞[.

(3) Used by: AirSpeed.

4.15. AirSpeedOrInitial

(1) Represents a AIR_SPEED or INITIAL air speed value.

(2) Choices:

a) **void INITIAL**
The INITIAL flight level value.

b) **AirSpeed AIR_SPEED**
The AIR_SPEED value.

(3) Used by: LevelAndSpeedReroutingConstraint.

4.16. <<enumeration>> AIXMDatasetType

(1) Enumerates the supported AIXM datasets

(2) Values:

a) **COMPLETE**
The Complete AIXM dataset

b) **INCREMENTAL**
The Incremental AIXM dataset

(3) Used by: AIXMDatasetMessagePayload, AIXMDatasetMessageFilter.

4.17. <<enumeration>> AIXMFeatureType

(1) The Feature type.

(2) Values:

a) **AirTrafficManagementService**

b) **AirportHeliport**

c) **AirportHeliportCollocation**

d) **AirportHeliportSet**

e) **Airspace**

f) **AngleIndication**

g) **ArrivalLeg**

h) **DepartureLeg**

i) **DesignatedPoint**

j) **DistanceIndication**

k) **FlightRestriction**

l) **Flow**

m) **Navaid**

n) **OrganisationAuthority**

o) **ReferenceLocation**

p) **`Route`**

q) **`RouteSegment`**

r) **`SpecialDate`**

s) **`StandardInstrumentArrival`**

t) **`StandardInstrumentDeparture`**

u) **`StandardLevelColumn`**

v) **`StandardLevelTable`**

w) **`TrafficVolume`**

x) **`TrafficVolumeSet`**

y) **`Unit`**

(3) Used by: IncrementalDatasetSummary.

4.18. `AIXMFile`

(1) Represents a ADR AIXM file for a given DATE, DATA_SET_TYPE, UPDATE_ID, NM_RELEASE, AIXM feature type and temporality.

(2) Inherits from: `File`.

(3) Used by: CompleteDatasetSummary, IncrementalDatasetSummary.

4.19. `<<enumeration>> AIXMTemporality`

(1) The temporality of the data in accordance with AIXM 5.1 or higher model.

(2) Values:

a) **`BASELINE`**

b) **`PERM_DELTA`**

(3) Used by: IncrementalAIXMDatasetRequest.

4.20. `AUP`

(1) Represents an AUP, i.e. either indeed a (baseline) AUP or a UUP.

(2) Attributes:

a) **`AUPSummary summary`** *(Mandatory)*

Contains the summary information of the AUP/UUP.
Constraint: See INCONSISTENT_AUP_MANUAL_ENTRIES_AND_SUMMARY_NIL_AUP

b) **AUPManualEntries aupManualEntries** *(Optional)*
The list of manual AUP entries managed by the client application.
Constraint: See INCONSISTENT_AUP_MANUAL_ENTRIES_AND_SUMMARY_NIL_AUP

c) **AUPComputedEntries aupComputedEntries** *(Contextual)*
The list of computed AUP entries as computed by the NM system.
Presence:

i) Must be null in AUPCreationRequest, AUPUpdateRequest, AUPValidationRequest

ii) Optional otherwise.

(3) Constraint:

a)

Name	INCONSISTENT_AUP_MANUAL_ENTRIES_AND_SUMMARY_NIL_AUP
Attributes	summary, aupManualEntries
Context	AUPCreationRequest, AUPValidationRequest, AUPUpdateRequest
Description	The attribute aupManualEntries must be null if summary.nilAUP is true. Cannot be null otherwise.

(4) Used by: AUPUpdateReply, AUPCreationRequest, AUPValidationRequest, AUPUpdateRequest, AUPRetrievalReply, AUPCreationReply.

4.21. AUPChain

(1) Represents an AUP chain, i.e. the AUP baseline of a (AMC, day) pair and its subsequent versions (UUPs) in the day.

(2) Attributes:

a) **DateYearMonthDay chainDate** *(Mandatory)*
The date of the AUP chain.

b) **AirNavigationUnitId amcId** *(Mandatory)*
The ANU id of the AMC to which this AUP chain belongs.

c) **AUPSummary[] aups** *(Mandatory)*
The ordered list of AUP summaries in the chain. The list is ordered according to the sequence of versions: the first summary in the list is the baseline AUP, the second one is the first update version after the baseline AUP, and so forth. The array can be empty.
Constraint: Size must be comprised between 0 and ∞.

(3) Used by: AUPChainRetrievalReply.

4.22. AUPComputedEntries

(1) AUP entries that are not manual, i.e. computed by the NM system based on default RSA availability (implicitRSAs) and/or based on the result of the expansion via CHMI.

(2) Attributes:

a) **ADRMessageType implicitCDRs** *(Optional)*
The list of implicit CDR openings and closures of this AUP. Is available (not null) when an AMC has executed the expansion via CHMI.
Presence:

i) Must be null if `summary.expandedAUP` is false, or if `summary.nilAUP` is true.

ii) Cannot be null (but can be empty) otherwise.

b) **ADRMessageType mergedCDRs** *(Optional)*
The list of merged CDR openings and closures of this AUP. Computed based on merging the explicit CDRs and the implicit CDRs (if any) according to the following criteria (simplified): Merge all CDR updates for the same route, CDR type and source that overlap or touch in flight level range, applicability period or CDR update portion, where "merge" means taking the union of overlapping and touching elements. E.g. periods 09:00 until 12:00 and 10:00 until 14:00 are merged into 09:00 until 14:00.
Presence:

i) Must be null if `summary.nilAUP` is true.

ii) Cannot be null otherwise.

c) **ADRMessageType implicitRSAs** *(Optional)*
The list of implicit RSA allocations of this AUP. Implicit RSA are non manageable airspaces that are automatically allocated based on the default definition existing in NM. If a non manageable Airspace is allocated explicitly instead, it will not be included in this list.
Presence:

i) Must be null if `summary.nilAUP` is true.

ii) Cannot be null otherwise.

(3) Used by: AUP.

4.23. typedef<string> AUPId

(1) Unique id of an AUP, allocated by the NM system.

(2) Pattern: `HEXA{24}`

(3) Used by: AUPDeletionRequest, AUPSummary, AUPRetrievalRequest.

4.24. AUPManualEntries

(1) AUP Entries, i.e. CDR openings/closures and RSA allocations, to be provided by the client to NM.

(2) The NM system does not support cross-AIRAC AUP entries, i.e. an AUPRSAAllocation or AUP-CDROpeningClosure cannot have a validity period crossing an AIRAC boundary (midnight on an AIRAC date). Consequently, it is the client's responsibility to "cut" the AUP entries within an AUP to comply with this constraint.

(3) Attributes:

a) **ADRMessageType cdrs** *(Optional)*
The list of explicit CDR openings and closures of this AUP.

b) **ADRMessageType rsas** *(Optional)*
The list of explicit RSA allocations of this AUP.

(4) Used by: AUP.

4.25. <<enumeration>> AUPState

(1) Enumerates the possible states of an AUP.

(2) Values:

a) **DRAFT**
After initial upload and successful validation the AUP is in status DRAFT.

b) **READY**
Once the AUP is in DRAFT status operational coordination between adjacent AMCs and FMPs takes place. Once the coordination is completed the AUP is promoted to READY status by the AMC.

c) **RELEASED**
AUPs in READY are promoted to "RELEASED" by the CADF after the CADF himself has validated the AUPs for completeness and non-contradiction.

(3) Used by: AUPSummary.

4.26. AUPSummary

(1) Represents an AUP summary, i.e. all its associated data apart from its main contents (CDR openings/closures and RSA allocations).

(2) Attributes:

a) **AUPId id** *(Contextual)*
Unique id that the NM system associates to the AUP.
Presence:

i) Must be null in AUPCreationRequest, AUPValidationRequest

ii) Mandatory in AUPUpdateRequest

iii) Optional otherwise.

b) **AUPId originatingAupId** *(Optional)*
Unique id of the originating AUP.
It must be the value of the use plan this UUP is based on.
Constraints:

i) See ORIGINATING_AUP_ID_CANNOT_BE_NULL_IF_UUP_WRITE

ii) See ORIGINATING_AUP_ID_MUST_BE_NULL_IF_AUP_WRITE

c) **DateYearMonthDay chainDate** *(Mandatory)*
The chain date of the chain to which this AUP belongs. Mandatory.

d) **AirNavigationUnitId amcId** *(Contextual)*
The ANU id of the AMC to which this AUP belongs. Associated to the AUP at creation time by the NM system. Must be equal to the caller's ANU id in any service that modifies an existing AUP.
Presence:

i) Must be null in AUPCreationRequest, AUPValidationRequest

ii) Mandatory in AUPUpdateRequest

iii) Optional otherwise.

e) **AUPType aupType** *(Mandatory)*
Specifies if this AUP is a BASELINE (AUP) or an UPDATE (UUP). Redundant, used to check that the client and server share the same understanding of what the object represents, especially at creation time.

f) **DateTimeMinutePeriod validityPeriod** *(Mandatory)*
The period in which this AUP is valid. When saving an AUP of type AUPType.BASELINE, must be [06:00, 06:00 [; when saving an AUP of type AUPType.UPDATE, must be [S, 06:00 [where S is posterior or equal to the start time of the validity period of the predecessor.

g) **DateTimeMinute releaseTime** *(Contextual)*
The time at which the AUP was released.
Presence:

i) Must be null in AUPCreationRequest, AUPUpdateRequest, AUPValidationRequest

ii) Optional otherwise.

h) **AUPState aupState** *(Mandatory)*

Current state of this AUP.
Constraints:

i) See DRAFT_STATUS_NOT_ALLOWED_FOR_NIL_AUP

ii) See RELEASED_STATUS_NOT_ALLOWED_IN_WRITE_MODE

i) **boolean nilAUP** *(Mandatory)*
Indicates whether this AUP is nil or not.

j) **string remark** *(Mandatory)*
Short remark associated to the AUP.
Constraints:

i) Pattern: `(UALPHA|DIGIT|/| ){0,128}`

ii) See INVALID_REMARK

k) **string[] note** *(Mandatory)*
Additional information on the AUP.
Constraints:

i) Size must be comprised between 0 and 25.

ii) Item Pattern: `(UALPHA|DIGIT| ){0,60}`

l) **boolean expandedAUP** *(Mandatory)*
Indicates whether the AUP contains implicit CDRs as a result of the AMC running the expansion via CHMI. Attention: the value of this attribute is not affected by the B2B - AUP expand service. Mandatory -- ignored in all input AUPs.

m) **LastUpdate lastUpdate** *(Contextual)*
Last update information -- set by system.
Presence:

i) Must be null in AUPCreationRequest, AUPValidationRequest

ii) Mandatory in AUPUpdateRequest

iii) Optional otherwise.

(3) Constraints:

a)

Name	ORIGINATING_AUP_ID_MUST_BE_NULL_IF_AUP_WRITE
Attribute	originatingAupId
Context	AUPCreationRequest, AUPValidationRequest, AUPUpdateRequest
Description	Must be null when it is an AUP.

b)

Name	ORIGINATING_AUP_ID_CANNOT_BE_NULL_IF_UUP_WRITE
Attribute	originatingAupId
Context	AUPCreationRequest, AUPValidationRequest, AUPUpdateRequest
Description	Cannot be null when it is an UUP.

c)

Name	RELEASED_STATUS_NOT_ALLOWED_IN_WRITE_MODE
Attribute	aupState
Context	AUPCreationRequest, AUPValidationRequest, AUPUpdateRequest
Description	Must be either DRAFT or READY in all write services -- can be DRAFT, READY or RELEASED in read-only services.

d)

Name	DRAFT_STATUS_NOT_ALLOWED_FOR_NIL_AUP
Attribute	aupState
Context	AUPCreationRequest, AUPValidationRequest, AUPUpdateRequest
Description	DRAFT status is not allowed for a nil AUP.

e)

Name	INVALID_REMARK
Attribute	remark
Context	AUPCreationRequest, AUPValidationRequest, AUPUpdateRequest
Description	According to the current CIAM/CHMI process, the AUP remark must start with the phrase "NIL AUP " or "NIL UUP " if nilAUP is true and cannot start with these phrases otherwise; the "AUP" or "UUP" bit must match the actual AUP type. In order to remove this constraint from the client applications, the NOP/B2B system prefixes the given remark value with the appropriate phrase (hence 8 characters) when nilAUP is true. As a consequence the character set is: i) (UALPHA|DIGIT|/|){0,128} in output ii) (UALPHA|DIGIT|/|){0,120} in input

(4) Used by: AUPChain, AUP.

4.27. <<enumeration>> AUPType

(1) Enumerates the possible types of an AUP.

(2) Values:

a) **BASELINE**

actual AUP, starting an AUPChain

b) **`UPDATE`**
actual UUP, following an AUP or UUP in an AUPChain

(3) Used by: AUPSummary.

4.28. `typedef<int> Capacity`

(1) Capacity on a traffic volume, expressed as a number of flights per hour.

(2) Range: `[0,998[`.

(3) Used by: PlannedCapacity.

4.29. `CapacityPlan`

(1) Capacity plan for a given traffic volume on a given day.

(2) A capacity plan is a special plan in the sense that, there can be cases where no capacity is defined in CACD. So there exist traffic volumes for which the capacity is not known at all or not known for some periods.

(3) In addition, regulation measures can overrule the capacity values defined (either in CACD or by the B2B update capacity service). This (optional) overruling regulation info can be found back in the `nmSchedule` attribute. However in the client schedule the non regulated capacities are maintained (just in case the regulation can be cancelled before the end of its applicability period).

(4) In a retrieval context, the plan is said to be 'complete' in the sense that it contains all the plan entries from all involved data sources (including NO_DATA data source in case no info is known).

(5) In an update context, the plan can be complete (if the B2B client designer decided to overwrite the default CACD values) or limited to the (full list) of (pre-)tactical updates with the gaps marked as AIRSPACE datasource (to obtain a complete time partition).

(6) In any case, periods in the time partition marked as AIRSPACE datasource correspond to removing any potential (pre-)tactical update and hence reset the corresponding values to the CACD definition for that period.

(7) Inherits from: `TacticalConfigurationPlan`.

(8) Attributes:

a) **`TrafficVolumeId trafficVolume`** *(Mandatory)*
Traffic volume to which these capacities apply.

b) **`Set<PlannedCapacity> nmSchedule`** *(Contextual)*
The plan resulting from the superimposition of all constraints from all data sources, which the NM system is using as plan for its own calculations.

The NM schedule exposes the complete time partition of the configurations for the day, i.e. the data coming from the various data sources contributing to the NM view of the plan: these data sources are defined in PlanDataSource.
The NM schedule cannot be updated directly by the caller; it is the outcome of updates via the various data sources.
The possible values of dataSource are limited to `NO_DATA`, `AIRSPACE`, `TACTICAL` and `MEASURE` - the `MEASURE` value being used to express that the capacity is derived from a regulation.
Presence:

i) Must be null in `CapacityPlanUpdateRequest`, `OTMVPlanUpdateRequest`, `RunwayConfigurationPlanUpdateRequest`, `SectorConfigurationPlanUpdateRequest`, `TrafficVolumeActivationPlanUpdateRequest`

ii) Mandatory otherwise.

Constraint: Size must be comprised between 0 and ∞.

c) **`Set<PlannedCapacity> clientSchedule`** *(Mandatory)*
(Pre-)tactical capacities associated to their applicability period, as maintained by the client. This plan contains only the updated configurations together with an indication that the default CACD values apply when not updated (cf. PlanDataSource). The actual CACD values for these CACD periods can be found in the `nmSchedule`
In an update context, the clientSchedule can be complete (if the B2B client designer decided to overwrite the default CACD values) or limited to the actual differences with regards to the CACD defaults. This is a B2B client designer's decision and depends on how CACD wants to be used in combination with the B2B. If the clientSchedule only contains the actual differences with regards to the CACD defaults, then the clientschedule still needs to contain the full list of (pre-) tactical updates and for the non (pre-) tactically updated periods, an explicit indication that the CACD values need to be used (but without repeating the CACD values themselves). So in any case, the clientschedule needs to be a complete time partition for the full day.
In any case, it is of drastic importance to understand that an entry in the schedule (i.e. a period) overwrites all CACD values in that period.
The possible values of dataSource are limited to `AIRSPACE` and `TACTICAL` - the `AIRSPACE` value meaning that the value associated to the applicabilityPeriod is the CACD one (i.e. CACD or that there is no data defined in the CACD), and the `TACTICAL` value meaning that this plan entry corresponds to the explicit tactical update expressed via capacity.
Constraints:

i) Size must be comprised between 0 and ∞.

ii) See `INCOMPLETE_SCHEDULE`

(9) Constraint:

a)

Name	`INCOMPLETE_SCHEDULE`
Attribute	`clientSchedule`

Description	`clientSchedule` has gaps and/or overlaps in the time partition or is not covering exactly one day.

(10) Used by: CapacityPlanRetrievalReply, CapacityPlanUpdateRequest, CapacityPlanUpdateReply.

4.30. `CompleteDatasetQueryCriteria`

(1) The criteria by which to query for Complete AIXM Datasets.

(2) Choices:

a) **`AiracIdentifier airac`**
Specific AIRAC (AIRAC id or AIRAC sequence number) for which datasets are requested. Only datasets related to the specified AIRAC are returned. Normally the data effective at a particular AIRAC is made available by NM 6 days before the AIRAC switch, so a query for all datasets of a given AIRAC may return from 0 to 34 datasets (28 days of AIRAC + 6 days in advance).

b) **`DateYearMonthDay date`**
Allows querying for datasets based on their publication date.

c) **`DateYearMonthDayPeriod publicationPeriod`**
Allows querying for datasets based on their publication date: only datasets published within the given period will be returned.

(3) Used by: CompleteAIXMDatasetRequest.

4.31. `CompleteDatasetSummary`

(1) Describes a Complete AIXM Dataset.

(2) Attributes:

a) **`AirspaceDataUpdateId updateId`** *(Mandatory)*
The id of the latest update included in the data set. This updateId is the key to be used to query for subsequent updates (Incremental AIXM Datasets).
Remark: This key is not supposed to be manipulated by the consumer. It is a kind of opaque key exchanged between the consumer and the provider.

b) **`DateYearMonthDay publicationDate`** *(Mandatory)*
The date in which the dataset was made available by NM.

c) **`AiracIdentifier[] sourceAIRACs`** *(Mandatory)*
This is an array of either 1 or 2 elements that contains the identifiers of the AIRAC cycles potentially affected by the data set.
Constraint: Size must be comprised between 1 and 2.

d) **`AIXMFile[] files`** *(Mandatory)*

The list of AIXM file ids that compose the dataset.
Constraint: Size must be comprised between 0 and ∞.

(3) Used by: CompleteAIXMDatasetReply, AIXMDatasetMessagePayload.

4.32. DBEOrPublishedPointId

(1) Represents a DBE or Published point identifier.

(2) Choices:

a) **DBEPointId DBE**
The DBE point identifier.

b) **PublishedPointId PUBLISHED**
The PUBLISHED point identifier.

(3) Used by: DeltaLevelReroutingConstraint, AvoidViaPointReroutingConstraint, LevelAndSpeedReroutingConstraint.

4.33. DBEPoint

(1) Represents a non-published dbe point.

(2) Inherits from: NonPublishedPoint.

(3) Attributes:

a) **DBEPointId dbePointId** *(Mandatory)*
DBE point identifier.
Constraint: See UNSUPPORTED_POINT_TYPE

(4) Constraint:

a)

Name	UNSUPPORTED_POINT_TYPE
Attribute	dbePointId
Context	FlightListByAerodromeReply, FlightListByAerodromeSetReply, FlightListByAircraftOperatorReply, FlightListByAirspaceReply, FlightListByHotspotReply, FlightListByKeysReply, FlightListByMeasureReply, FlightListByPointReply, FlightListByTrafficVolumeReply, FlightRetrievalReply
Description	TheDBEPoint is supported only in FLIGHT_LIST_REPLY context.

4.34. typedef<string> DBEPointId

(1) Old Point id still in use by IPFS and EFTMS.

(2) Pattern: `(UALPHA|DIGIT|*){1,5}`

(3) Used by: DBEOrPublishedPointId, DBEPoint, ReferenceLocationDBEPoint.

4.35. EAUPChain

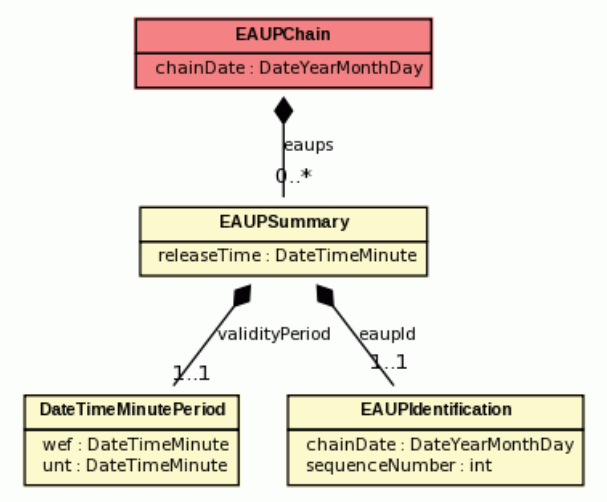

Figure 4.3. EAUPChain Class Diagram

(1) Represents an EAUP chain, i.e. the EAUP baseline of a day and its subsequent versions in the day.

(2) Attributes:

a) **DateYearMonthDay chainDate** *(Mandatory)*
The date of the EAUP chain. Valid chain dates are:

i) D-1 (pre-tactical, tomorrow)

ii) D (tactical, today)

iii) [D+1 (yesterday), D+15 months] (post-ops)

b) **EAUPSummary[] eaups** *(Mandatory)*
The ordered list of EAUP summaries in the chain. The list is ordered according to the sequence of versions: the first summary in the list is the baseline EAUP, the second one is the first update version after the baseline EAUP, and so forth. This ordering is recalled in the EAUPIdentification through a sequence number. The array can be empty.
REMARK: We return OBJECT_NOT_FOUND if the EAUPChain has never been created in the NM system, being for the EAUPChain of today, or in 3 months. Note that the situation is slightly different for a "past" EAUPChain: it is immutable (won't change anymore) so that if at the end of the day it contains no EAUP we know for sure that it is empty and the NM system creates it empty. We then return an EAUPChain with OK status and an empty EAUP/EUUP list.
Constraint: Size must be comprised between 0 and ∞.

(3) Used by: EAUPChainRetrievalReply.

4.36. EAUPIdentification

(1) Represents the unique id of a released EAUP.

(2) Attributes:

a) **DateYearMonthDay chainDate** *(Mandatory)*
The date of the chain to which this EAUP belongs.
Constraint: See INVALID_CHAIN_DATE

b) **int sequenceNumber** *(Mandatory)*
The position of the EAUP in its chain. The baseline occupies position 0.
Constraint: Range: `[0,∞[`.

(3) Constraint:

a)

Name	INVALID_CHAIN_DATE
Attribute	chainDate
Context	EAUPCDRCompareRequest, EAUPCDRRequest, EAUPRSACompareRequest, EAUPRSARequest
Description	The chainDate must be a date which belongs to interval [today - 15 months, today + 2 days].

(4) Used by: EAUPRSACompareRequest, EAUPSummary, EAUPRSARequest, EAUPCDRRequest, EAUPCDRCompareRequest.

4.37. EAUPSummary

(1) Represents an EAUP summary, i.e. all its associated data apart from its main contents (CDR openings/closures and RSA allocations).

(2) Attributes:

a) **DateTimeMinute releaseTime** *(Mandatory)*
The time at which the EAUP has been released and therefore became available to the caller of this service.

b) **DateTimeMinutePeriod validityPeriod** *(Mandatory)*
The period in which this EAUP is valid.

c) **EAUPIdentification eaupId** *(Mandatory)*
The unique id of the EAUP. This is the object to be subsequently used for retrieving/querying the contents of the EAUP.

(3) Used by: EAUPMessage, EAUPChain.

4.38. <<enumeration>> ErrorCategory

(1) Error categories for this service group.

(2) Values:

a) **FUA**
for all input validation errors related to AUP/EAUP

4.39. <<enumeration>> ErrorType

(1) Error types for this service group.

(2) Values:

a) **AUP_ALREADY_EXISTS**
The referred AUP already exists -- no parameter

b) **AUP_AMC_BID_NOT_FOUND**
The AMC must exist in NM -- no parameter.

c) **AUP_AMC_DELEGATED_ROUTE_PORTION_NOT_USED**
This is a warning indicating that the AUP/UUP does not use a route portion that was the delegated to the corresponding AMC

d) **AUP_AMC_MUST_EXIST**
AMC must exist for the whole lifetime of AUP -- no parameter

e) **AUP_CDR_UPDATE_AMC_NOT_RESPONSIBLE**
The originator AMC must be completely responsible for the CDR update segment of the CDR update -- parameters:

i) "ROUTE": Route the route segment is part of given by its NM unique id

ii) "START_POINT": Start point of the route segment given by its NM unique id

iii) "END_POINT": End point of the route segment given by its NM unique id

f) **AUP_CDR_UPDATE_CLOSURE_CDR_TYPE**
A CDR closure must refer to route segments that are closeable in at least one direction, i.e. ATS and CDR1 in one direction, and not CDR2 in the other direction-- parameters:

i) "ROUTE": Route the route segment is part of given by its NM unique id

ii) "START_POINT": Start point of the route segment given by its NM unique id

iii) "END_POINT": End point of the route segment given by its NM unique i

g) **AUP_CDR_UPDATE_FL_RANGE**
A singleton altitude range cannot be opened/closed -- parameters:

i) "ROUTE": Route the route segment is part of given by its NM unique id

ii) "START_POINT": Start point of the route segment given by its NM unique id

iii) "END_POINT": End point of the route segment given by its NM unique id

h) **AUP_CDR_UPDATE_FL_RANGE_ERROR**
The provided FL Range must be valid e.g. lower limit < upper limit -- parameters:

i) "LOWERLIMIT": Lower limit of the FL range.

ii) "UPPERLIMIT": Upper limit of the FL range.

iii) "ROUTE": Route the route segment is part of given by its NM unique id

iv) "START_POINT": Start point of the route segment given by its NM unique id

v) "END_POINT": End point of the route segment given by its NM unique id

i) **AUP_CDR_UPDATE_FL_TOO_LARGE**
n CDR update, FL range must be inside the FL range of the route -- parameters:

i) "ROUTE": Route the route segment is part of given by its NM unique id

ii) "START_POINT": Start point of the route segment given by its NM unique id

iii) "END_POINT": End point of the route segment given by its NM unique id

j) **AUP_CDR_UPDATE_LOWER_CRUISING_LEVEL**
In CDR update, if there is no cruising FL between the lower limit of the CDR update and the lower limit of the default route availability, the lower limit of the CDR update must be extended to the lower limit of the default route availability -- parameters:

i) "ROUTE": Route the route segment is part of given by its NM unique id

ii) "START_POINT_1": Start point of the first route segment given by its NM unique id

iii) "END_POINT_1": End point of the first route segment given by its NM unique id

iv) "START_POINT_2": Start point of the second route segment given by its NM unique id

v) "END_POINT_2": End point of the second route segment given by its NM unique id

vi) "FLIGHT_LEVEL": Altitude given by a flight level value

vii) "EXPECTED_FL": Expected lower limit of the CDR update given by a flight level value

k) **AUP_CDR_UPDATE_LOWER_FL_NOT_CRUISING_LEVEL**
For bi-directional route segments, the lower limit of the CDR update must be such that the CDR update covers the lower cruising level of the route segment -- parameters:

i) "ROUTE": Route the route segment is part of given by its NM unique id

ii) "START_POINT_1": Start point of the first route segment given by its NM unique id

iii) "END_POINT_1": End point of the first route segment given by its NM unique id

iv) "START_POINT_2": Start point of the second route segment given by its NM unique id

v) "END_POINT_2": End point of the second route segment given by its NM unique id

vi) "FLIGHT_LEVEL": Altitude given by a flight level value

vii) "EXPECTED_FL": Expected lower limit of the CDR update given by a flight level value

l) **AUP_CDR_UPDATE_NOTAM_CLOSURE**
A CDR closure must refer to route segments that are closed by NOTAM for the whole FL range and applicability period -- parameters:

i) "ROUTE": Route the route segment is part of given by its NM unique id

ii) "START_POINT": Start point of the route segment given by its NM unique id

iii) "END_POINT": End point of the route segment given by its NM unique id

m) **AUP_CDR_UPDATE_NO_CRUISING_LEVEL**
In CDR update, the FL range must include at least one cruising level -- parameters:

i) "ROUTE": Route the route segment is part of given by its NM unique id

ii) "START_POINT": Start point of the route segment given by its NM unique id

iii) "END_POINT": End point of the route segment given by its NM unique id

n) **AUP_CDR_UPDATE_OPENING_CDR_TYPE**
A CDR opening must refer to route segments that can be opened in at least one direction, i.e. CDR2, and not ATS or CDR1 in the other -- parameters:

i) "ROUTE": Route the route segment is part of given by its NM unique id

ii) "START_POINT": Start point of the route segment given by its NM unique id

iii) "END_POINT": End point of the route segment given by its NM unique id

o) **AUP_CDR_UPDATE_OVERLAP**
Applicability period and FL range of CDR updates cannot overlap -- parameters:

i) "ROUTE": Route the route segment is part of given by its NM unique id

ii) "START_POINT": Start point of the route segment given by its NM unique id

iii) "END_POINT": End point of the route segment given by its NM unique id

p) **AUP_CDR_UPDATE_OVERLAPS_AIRAC_SWITCH**

q) **AUP_CDR_UPDATE_OVERLAPS_BETWEEN_AMCS**
A CDR update conflicts with a CDR update from another AMC. -- parameters:

i) "ROUTE": NM unique id of the route from the CDR update

ii) "START_POINT": NM unique id of the start point from the CDR update

iii) "END_POINT": NM unique id of the end point from the CDR update

iv) "ROUTE": NM unique id of the route from the CDR update of the other AMC

v) "START_POINT": NM unique id of the start point from the CDR update of the other AMC

vi) "END_POINT": NM unique id of the end point from the CDR update of the other AMC

vii) "AMC": AMC id of the other AMC

r) **AUP_CDR_UPDATE_PERIOD_AUP_MISMATCH**

s) **AUP_CDR_UPDATE_POINT_IN_ROUTE**
The points forming the route segment of CDR update must be part of the route -- parameters:

i) "ROUTE": Route the route segment is part of given by its NM unique id

ii) "START_POINT": Start point of the route segment given by its NM unique id

iii) "END_POINT": End point of the route segment given by its NM unique id

t) **AUP_CDR_UPDATE_PTS_ROUTE**
The CDR update is not allowed for a PTS route -- parameters:

i) "ROUTE": Route the route segment is part of given by its NM unique id

ii) "START_POINT": Start point of the route segment given by its NM unique id

iii) "END_POINT": End point of the route segment given by its NM unique id

u) **AUP_CDR_UPDATE_ROUTE_MUST_EXIST**
The route must exist for the whole lifetime of the CDR update -- parameters:

i) "ROUTE": Route the route segment is part of given by its NM unique id

ii) "START_POINT": Start point of the route segment given by its NM unique id

iii) "END_POINT": End point of the route segment given by its NM unique id

v) **AUP_CDR_UPDATE_STATUS_CONFLICT**

w) **AUP_CDR_UPDATE_UPPER_CRUISING_LEVEL**

In CDR update, if there is no cruising FL between the upper limit of the CDR update and the upper limit of the default route availability, the upper limit of the CDR update must be extended to the upper limit of the default route availability -- parameters:

i) "ROUTE": Route the route segment is part of given by its NM unique id

ii) "START_POINT_1": Start point of the first route segment given by its NM unique id

iii) "END_POINT_1": End point of the first route segment given by its NM unique id

iv) "START_POINT_2": Start point of the second route segment given by its NM unique id

v) "END_POINT_2": End point of the second route segment given by its NM unique id

vi) "FLIGHT_LEVEL": Altitude given by a flight level value

vii) "EXPECTED_FL": Expected upper limit of the CDR update given by a flight level value

x) **AUP_CDR_UPDATE_UPPER_FL_NOT_CRUISING_LEVEL**
In CDR update, for bi-directional route segments, the upper limit of the CDR update must be such that the CDR update covers a cruising level -- parameters:

i) "ROUTE": Route the route segment is part of given by its NM unique id

ii) "START_POINT_1": Start point of the first route segment given by its NM unique id

iii) "END_POINT_1": End point of the first route segment given by its NM unique id

iv) "START_POINT_2": Start point of the second route segment given by its NM unique id

v) "END_POINT_2": End point of the second route segment given by its NM unique id

vi) "FLIGHT_LEVEL": Altitude given by a flight level value

vii) "EXPECTED_FL": Expected upper limit of the CDR update given by a flight level value

y) **AUP_CDR_UPDATE_VALID_POINT_TYPES**
The start and end points of merged opened or closed route segment (one or more segments) must be way points or navigation aid points (this validation does not refer to manual CDR updates) -- parameters:

i) "ROUTE": Route the route segment is part of given by its NM unique id

ii) "START_POINT": Start point of the route segment given by its NM unique id

iii) "END_POINT": End point of the route segment given by its NM unique id

z) **AUP_CDR_UPDATE_WRONG_ROUTE_TYPE**
CDR updates are not allowed on oceanic routes -- parameters:

i) "ROUTE": Route the route segment is part of given by its NM unique id

ii) "START_POINT": Start point of the route segment given by its NM unique id

iii) "END_POINT": End point of the route segment given by its NM unique id

aa) **AUP_DOES_NOT_EXIST**
The referred AUP must exist - no parameter

ab) **AUP_EMPTY**
The AUP is not nil and does not contain any AUP manual entry -- no parameter

ac) **AUP_FBZ_ALLOCATION_MUST_HAVE_FUA_ALLOCATION**
An FBZ allocation must have at least one activated FUA activation. -- parameters:

i) "RESTRICTED_AIRSPACE_1": NM unique id of the FBZ

ad) **AUP_FL_RANGE_UOM_ERROR**
The unit of measurement is not correct (see companion doc) -- no parameter.

ae) **AUP_FUA_ACTIVATION_FOR_DISABLED_RESTRICTION**
The FUA restriction of the FUA allocation must be enabled in the AIRAC of the activation. -- parameters:

i) "FUA_RESTRICTION": NM unique id of the FUA restriction which is not enabled

af) **AUP_FUA_ACTIVATION_FUA_MUST_EXIST**
The FUA Restriction that is being activated doesn't exist (yet or anymore) in the AIRAC of the RSA allocation. -- parameters:

i) "FUA_RESTRICTION": NM unique id of the FUA restriction

ii) "RESTRICTED_AIRSPACE": NM unique id of the RSA

ag) **AUP_FUA_ACTIVATION_IS_NOT_RELATED_TO_RSA**
The FUA restriction of the FUA allocation must have the allocated RSA as its reference location. -- parameters:

i) "FUA_RESTRICTION": NM unique id of the FUA restriction

ii) "RESTRICTED_AIRSPACE": NM unique id of the RSA

ah) **AUP_FUA_ACTIVATION_REMARK_INVALID_CHARS**
The FUA restriction activation remark contains invalid characters. -- parameters:

i) "FUA_RESTRICTION": NM unique id of the FUA restriction

ai) **AUP_FUA_ACTIVATION_REMARK_INVALID_FORMAT**
The FUA activation remark is composed of a limited set of characters. Lowercase characters are not part of this limited set. -- parameters:

i) "FUA_RESTRICTION": NM unique id of the FUA restriction

aj) **AUP_FUA_ACTIVATION_REMARK_TOO_LONG**
The size of the FUA activation remark is limited to 128 characters. -- parameters:

i) "FUA_RESTRICTION": NM unique id of the FUA restriction

ak) **AUP_INVALID_PERIOD**
The provided period is invalid.

al) **AUP_NIL_AUP_MANUAL_MUST_BE_NULL**
In a nil AUP, AUP manual entries must be null -- no parameter

am) **AUP_NIL_AUP_NOT_EMPTY**

an) **AUP_NIL_AUP_STATE**
A nil AUP cannot be DRAFT state -- no parameter

ao) **AUP_ONLY_ONE_SEGMENT_PROVIDED_AT_AIRAC_SWITCH**
Only one AUP segment was provided at AIRAC switch.

ap) **AUP_ORIGINATING_ID_EMPTY**
When creating a UUP, the originating AUP id should not be empty.

aq) **AUP_ORIGINATING_ID_INVALID**
The originating AUP id is invalid. Valid AUP ids are the results of a creation or query.

ar) **AUP_ORIGINATING_ID_NON_EMPTY**
When creating an AUP, the originating AUP id should be empty.

as) **AUP_ORIGINATING_ID_NON_UPDATEABLE**
Read only originating AUPs cannot be updated.

at) **AUP_OUTSIDE_AVAILABILITY_PERIOD**
The AUP and its content must be defined within the availability period -- no parameter

au) **AUP_PERIOD_BETWEEN_6_AND_6_AM**
Aup period should be 6AM of first day till 6AM following day

av) **AUP_RSA_ALLOCATIONS_OVERLAP**
Two geometrically overlapping RSAs under the responsibility of on AMC must not have RSA allocations overlapping in time and flight level. -- parameters:

i) "RESTRICTED_AIRSPACE_1": NM unique id of the first RSA

ii) "WEF_TIME": start time of the time overlap (e.g. 10:00)

iii) "TIL_TIME": end time of the time overlap (e.g. 11:00)

iv) "RESTRICTED_AIRSPACE_2": NM unique id of the second RSA

v) "LOWER_FL": lower flight level of the flight level overlap

vi) "UPPER_FL": upper flight level of the flight level overlap

aw) **AUP_RSA_ALLOCATIONS_OVERLAP_OTHER_AMC**
Two geometrically overlapping RSAs under the responsibility of two different AMCs must not have RSA allocations overlapping in time and flight level range. -- parameters:

i) "RESTRICTED_AIRSPACE_1": NM unique id of the first RSA

ii) "WEF_TIME": start time of the time overlap (e.g. 10:00)

iii) "TIL_TIME": end time of the time overlap (e.g. 11:00)

iv) "RESTRICTED_AIRSPACE_2": NM unique id of the second RSA

v) "LOWER_FL": lower flight level of the flight level overlap

vi) "UPPER_FL": upper flight level of the flight level overlap

ax) **AUP_RSA_ALLOCATION_AMC_NOT_RESPONSIBLE**
The originator AMC must be responsible for the RSA of the RSA allocations -- no parameter

ay) **AUP_RSA_ALLOCATION_COMPOSED_OVERLAP**
Applicability period and FL range of (composed) RSA allocations cannot overlap with (composing) RSA allocations -- parameters:

i) "RESTRICTED_AIRSPACE_COMPOSED": Composed restricted airspace given by its NM unique id

ii) "RESTRICTED_AIRSPACE_COMPOSING": Composing restricted airspace given by its NM unique id

az) **AUP_RSA_ALLOCATION_FL_RANGE**
In RSA allocation, a singleton altitude range cannot be allocated -- no parameter

ba) **AUP_RSA_ALLOCATION_FL_RANGE_ERROR**
The provided FL Range must be valid e.g. lower limit < upper limit -- parameters:

i) "LOWERLIMIT": Lower limit of the FL range.

ii) "UPPERLIMIT": Upper limit of the FL range.

iii) "AIRSPACE": The Airspace that is allocated given by its NM unique id.

bb) **AUP_RSA_ALLOCATION_MUST_HAVE_FUA_ALLOCATION**
All the FUA restrictions which have the RSA as their reference location should be explicitly mentioned as activated or not. -- parameters:

i) "RESTRICTED_AIRSPACE": NM unique id of the RSA

ii) "FUA_RESTRICTION": NM unique id of the FUA restriction that is not in the list of FUA activations

bc) **AUP_RSA_ALLOCATION_OVERLAP**
Applicability period and FL range of RSA allocations cannot overlap -- no parameter

bd) **AUP_RSA_ALLOCATION_PERIOD_AUP_MISMATCH**
The RSA allocation period must be within the validity period of the AUP -- no parameter

be) **AUP_RSA_ALLOCATION_PERIOD_RSA_MISMATCH**
The RSA allocation period must be within the availability period of the RSA -- no parameter

bf) **AUP_RSA_ALLOCATION_RSA_FBZ_OVERLAP**
When both FBZ and its owner RSA are allocated, the allocations must not overlap in time or flight level range. -- parameters:

i) "RESTRICTED_AIRSPACE_1": NM unique id of the FBZ

ii) "RESTRICTED_AIRSPACE_2": NM unique id of the owner RSA

bg) **AUP_RSA_ALLOCATION_RSA_MUST_EXIST**
The RSA must exist for the whole lifetime of the RSA allocation -- no parameter

bh) **AUP_RSA_ALLOCATION_VERTICAL_LIMITS**
Vertical limits of an RSA allocation must be within the vertical limits of the RSA -- no parameter

bi) **AUP_RSA_UPDATE_OVERLAPS_AIRAC_SWITCH**

bj) **AUP_UNDEFINED_UUP_START_TIME**
The start time of an UUP must have been defined by the CADF -- no parameter

bk) **AUP_USES_ROUTE_PORTION_DELEGATED_TO_OTHER_AMC**
This is a warning indicating that the AUP/UUP is using a route portion that was delegated to another AMC.

bl) **AUP_UUP_NO_CLOSURE_WITH_NOTAM**
An AUP closure is not at all covered by a NOTAM closure

bm) **AUP_UUP_PARTIALLY_COVERED_BY_NOTAM**
An AUP closure is only partially covered by NOTAM closures

bn) **AUP_UUP_PARTIALLY_COVERS_NOTAM**
A NOTAM closure is only partially covered by AUP closures

bo) **AUP_VALID_STATUS**
The AUP must have a valid status -- no parameter

bp) **EXPAND_WRONG_PERIOD**

The period provided for expansion should end at 06:00 AM and cannot be longer than 24 hours -- no parameter

bq) **INVALID_UUID**
The provided AUP uuid must have a valid format -- no parameter

br) **TIMESTAMP_MISMATCH**
The provided AUP timestamp does not match the timestamp of the existing AUP -- no parameter

bs) **UUP_CDR_CLOSURE_LEAD_TIME_TOO_SHORT**
Users should be warned of any new or extended (FL range or applicability) CDR1/ATS route closure with lead time of less than configured value

bt) **UUP_CDR_RECLOSURE_FOUND**
When promoting UUP to the Ready status the users must be warned of any occurrence of CDR2 closure of a route portion which was previously open by AUP or UUP

bu) **UUP_CDR_RECLOSURE_LEAD_TIME_TOO_SHORT**
Users should be warned of any new or extended (FL range or applicability) CDR2 re-closure with lead time of less than configured value

bv) **UUP_RSA_ALLOCATION_LEAD_TIME_TOO_SHORT**
Users should be warned of any new or extended (FL range or applicability) RSA allocation with lead time of less than configured value

bw) **UUP_WILL_CANCEL_PREVIOUS_AUP_UUP**
This is a warning to indicate that all RSA allocations and CDR updates from the previous AUP/UUP will be overwritten.

4.40. typedef<string> FIRICAOId

(1) ICAO id of an FIR.

(2) Pattern: `UALPHA{4}`

(3) Used by: EstimatedElapsedTimeAtLocation.

4.41. FlightLevel

(1) Flight level value together with its flight level unit.

(2) Attributes:

a) **FlightLevelUnit unit** *(Mandatory)*
Unit used to express this flight level.
Constraints:

i) See LEVEL_A_F_NOT_BETWEEN_0_AND_999

ii) See LEVEL_S_M_NOT_BETWEEN_0_AND_3050

b) **FlightLevel_DataType level** *(Optional)*
Flight level value in the given unit.
Constraints:

i) See LEVEL_A_F_NOT_BETWEEN_0_AND_999

ii) See LEVEL_GROUND_CEILING_MUTUALLY_EXCLUSIVE

iii) See LEVEL_S_M_NOT_BETWEEN_0_AND_3050

c) **boolean ground** *(Optional)*
Indicates the ground flight level, i.e. 0.
Constraint: See LEVEL_GROUND_CEILING_MUTUALLY_EXCLUSIVE

d) **boolean ceiling** *(Optional)*
Indicates the highest flight level, which corresponds to level F999.
Constraint: See LEVEL_GROUND_CEILING_MUTUALLY_EXCLUSIVE

(3) Constraints:

a)

Name	LEVEL_A_F_NOT_BETWEEN_0_AND_999
Attributes	unit, level
Description	When unit is FlightLevelUnit.F or FlightLevelUnit.A, then level must be in [0, 1000 [.

b)

Name	LEVEL_S_M_NOT_BETWEEN_0_AND_3050
Attributes	unit, level
Description	When unit is FlightLevelUnit.S or FlightLevelUnit.M, then level must be in [0, 3050].

c)

Name	LEVEL_GROUND_CEILING_MUTUALLY_EXCLUSIVE
Attributes	level, ground, ceiling
Description	level, ground and ceiling are mutually exclusive: exactly one of them must be not null.

(4) Used by: FourDPoint, AirFiledData, FlightLevelRange, Flight, RequestedFlightLevel, FlightLevelOrInitial, FlightAirspace, FlightPoint, FlightRestriction.

4.42. typedef<int> FlightLevel_DataType

(1) FlightLevel data type.

(2) Range:]- ∞,∞[.

(3) Used by: FlightLevel.

4.43. FlightLevelOrInitial

(1) Represents a FLIGHT_LEVEL or INITIAL flight level value.

(2) Choices:

a) **void INITIAL**
The INITIAL flight level value.

b) **FlightLevel FLIGHT_LEVEL**
The FLIGHT_LEVEL value.

(3) Used by: LevelAndSpeedReroutingConstraint.

4.44. FlightLevelRange

(1) Interval of flight levels.

(2) The range is left-closed and right-opened, i.e. a flight level range from 100 to 200 is interpreted as [100, 200 [.

(3) Attributes:

a) **FlightLevel min** *(Mandatory)*
Bottom boundary of the flight level range.

b) **FlightLevel max** *(Mandatory)*
Top boundary of the flight level range.

(4) Used by: TrafficVolumeLocation, AbstractEAUPCDRRequest, AbstractEAUPRSARequest, FlightListByPointRequest, TrafficCountsByPointRequest.

4.45. <<enumeration>> FlightLevelUnit

(1) Enumerates the units in which a flight level can be expressed.

(2) Values:

Value	Description
A	Altitude in hundreds of feet (as in "A330")
F	Standard flight level (as in "F330")
M	Altitude in tens of meters (as in "S1130")
MM	Altitude in meters.

Value	Description
S	Standard metric level in tens of meters (as in "S1130")
SM	Standard metric level in meters.

Table 4.2. <<enumeration>> FlightLevelUnit

(3) Used by: FlightLevel.

4.46. <<enumeration>> FlightPlanProcessing

(1) Indicates the kind of restriction with regards to flight plan processing.

(2) Values:

a) **AERODROME_FLIGHT_RULE**
This restriction defines that arrivals to or departures from the aerodrome reference location must be conducted under VFR

b) **DCT_LIMIT**
This restriction indicates a DCT segment limit for FIRs and for Aerodromes, as well as the general exceeding limit, specific DCT segments which are longer but nevertheless allowed can be defined and DCT segments which are shorter but still not allowed can also be defined

c) **FRA_DCT_LIMIT**
This is an extension of DCT_LIMIT for FRA ("Free Route Airspace") - in addition to the DCT_LIMIT restrictions, it defines FRA entry/exit and intermediate points

d) **PROFILE_TUNING**
Represents "letters of agreements" i.e. agreements between ATCs to transfer flights between them - when met they should only be used for profile tuning (avoid/force airspace penetration)

e) **RAD**
Indicates a RAD restriction that can be used by NM to invalidate a flight plan

f) **SSR_CODE_ALLOCATION**
Denotes a restriction which will be used in the allocation of SSR Codes (by CCAMS)

g) **TP_AIRCRAFT_TYPE_CLASSIFICATION**
This restriction indicates if terminal procedures are restricted to given aircraft type classification e.g. propellers only or jets only

(3) Used by: FlightRestriction.

4.47. GeoPoint

(1) Represents a non-published point expressed via its position.

(2) Inherits from: NonPublishedPoint.

(3) Attributes:

a) **Position position** *(Mandatory)*
Position of the point.

4.48. ICAOPoint

(1) Either a published point id, or an explicit non-published point.

(2) Choices:

a) **PublishedPointId pointId**
Published point id.

b) **NonPublishedPoint nonPublishedPoint**
Non-published point and therefore expressed inline.

(3) Used by: AirFiledData, PointDAL, EstimatedElapsedTimeAtLocation, FlightPoint, EnrouteDelay, OtherAerodromeDesignation.

4.49. IncrementalDatasetQueryCriteria

(1) The criteria by which to query for Incremental AIXM Datasets.

(2) Choices:

a) **AirspaceDataUpdateId lastKnownUpdateId**
This is the latest UpdateId known by the consumer.

b) **AiracIdentifier airac**
The AIRAC the datasets refer to.

c) **DateYearMonthDay date**
Allows querying for datasets based on their publication date.

d) **DateYearMonthDayPeriod publicationPeriod**
Allows querying for datasets based on their publication date: only datasets published within the given period will be returned.

(3) Used by: IncrementalAIXMDatasetRequest.

4.50. IncrementalDatasetSummary

(1) Describes an Incremental AIXM Dataset.

(2) Attributes:

a) **AirspaceDataUpdateId updateId** *(Mandatory)*
The id of the Airspace Data Update included in this data set.

b) **AirspaceDataUpdateId previousUpdateId** *(Mandatory)*
The id of the previous Airspace Data Update: to ensure continuity in the chain of Updates.

c) **DateYearMonthDay publicationDate** *(Mandatory)*
The date in which the dataset was made available by NM.

d) **AiracIdentifier[] sourceAIRACs** *(Mandatory)*
This is an array of either 1 or 2 elements that contains the identifiers of the AIRAC cycles included in the data set.
Constraint: Size must be comprised between 1 and 2.

e) **AIXMFile[] files** *(Mandatory)*
The list of AIXM file ids that compose the dataset.
Constraint: Size must be comprised between 0 and ∞.

f) **Map<AIXMFeatureType,int> affectedFeatures** *(Optional)*
This gives some information about the content of the update: it says for each AIXM 5.1 Feature Type how many features are affected by this Update.
This allows for the following scenario: if for example an Incremental AIXM Dataset only contains changes to FlightRestriction features and the consumer of the data is not interested in FlightRestrictions he may decide not to download this Incremental AIXM Dataset. The map can be null, denoting an update to the NM Airspace Data that does not map to any AIXM 5.1 Feature or property currently exported.
Constraints:

i) Size must be comprised between 0 and ∞.

ii) Item Range: `[0,∞[`.

(3) Used by: AIXMDatasetMessagePayload, IncrementalAIXMDatasetReply.

4.51. IRDesignatorFilter

(1) Represents a filter on IR airspace designator (ICAO-compliant FIR/UIR location indicators) wildcards.

(2) Attributes:

a) **string[] firDesignators** *(Optional)*
FIR designator wildcards. Each string item in the array can be a full FIR designator or a wildcard for a FIR designator. Supported wildcards are limited to at least one character and the star sign ("*") at the end of the expression. If null, all FIRs match.
Constraints:

i) Size must be comprised between 1 and ∞.

ii) Item Pattern: `UALPHA{4}|UALPHA{1,3}*`

iii) See FIRDESIGNATORS_AND_UIRDESIGNATORS_CANNOT_BE_BOTH_NULL

iv) See FIRDESIGNATORS_CANNOT_CONTAIN_DUPLICATE

b) **`string[] uirDesignators`** *(Optional)*
UIR designator wildcards. Each string item in the array can be a full UIR designator or a wildcard for a UIR designator. Supported wildcards are limited to at least one character and the star sign ("*") at the end of the expression. If null, all UIRs match.
Constraints:

i) Size must be comprised between 1 and ∞.

ii) Item Pattern:`UALPHA{4}|UALPHA{1,3}*`

iii) See FIRDESIGNATORS_AND_UIRDESIGNATORS_CANNOT_BE_BOTH_NULL

iv) See UIRDESIGNATORS_CANNOT_CONTAIN_DUPLICATE

c) **`LogicalOperator logicalOperator`** *(Optional)*
Specifies if AND or OR is meant between firDesignators and uirDesignators. AND by default.

(3) Constraints:

a)

Name	FIRDESIGNATORS_AND_UIRDESIGNATORS_CANNOT_BE_BOTH_NULL
Attributes	firDesignators, uirDesignators
Description	`firDesignators` and `uirDesignators` cannot be both null.

b)

Name	FIRDESIGNATORS_CANNOT_CONTAIN_DUPLICATE
Attribute	firDesignators
Description	If specified, the array cannot be null, and cannot contain null or duplicate items.

c)

Name	UIRDESIGNATORS_CANNOT_CONTAIN_DUPLICATE
Attribute	uirDesignators
Description	If specified, the array cannot be null, and cannot contain null or duplicate items.

(4) Used by: IRFilter.

4.52. `IRFilter`

(1) Used to filter the IR airspaces on which CDR openings/closures apply, based on either UUIDs or IR designators.

(2) Choices:

a) **IRUUIDFilter irUUIDFilter**
Filter the IR airspaces based on UUIDs.

b) **IRDesignatorFilter irDesignatorFilter**
Filter the IR airspaces based on IR designators.

(3) Used by: AbstractEAUPCDRRequest, AbstractEAUPRSARequest.

4.53. IRUUIDFilter

(1) Represents a filter on IR airspace UUIDs.

(2) Attributes:

a) **UUID[] firUUIDs** *(Optional)*
Matching FIR UUIDs. If null, all FIR UUIDs match.
Constraints:

i) Size must be comprised between 1 and ∞.

ii) See FIRUUIDS_AND_UIRUUIDS_CANNOT_BE_BOTH_NULL

iii) See FIRUUIDS_CANNOT_CONTAIN_DUPLICATE

b) **UUID[] uirUUIDs** *(Optional)*
Matching UIR UUIDs.If null, all UIR UUIDs match.
Constraints:

i) Size must be comprised between 1 and ∞.

ii) See FIRUUIDS_AND_UIRUUIDS_CANNOT_BE_BOTH_NULL

iii) See UIRUUIDS_CANNOT_CONTAIN_DUPLICATE

c) **LogicalOperator logicalOperator** *(Optional)*
Specifies if AND or OR is meant between firUUIDs and uirUUIDs. AND by default.

(3) Constraints:

a)

Name	FIRUUIDS_AND_UIRUUIDS_CANNOT_BE_BOTH_NULL
Attributes	firUUIDs, uirUUIDs
Description	firUUIDs and uirUUIDs cannot be both null.

b)

Name	FIRUUIDS_CANNOT_CONTAIN_DUPLICATE
Attribute	firUUIDs
Description	If specified, the array cannot be null, and cannot contain null or duplicate items.

c)

Name	UIRUUIDS_CANNOT_CONTAIN_DUPLICATE
Attribute	uirUUIDs
Description	If specified, the array cannot be null, and cannot contain null or duplicate items.

(4) Used by: IRFilter.

4.54. <<enumeration>> LoadState

(1) Enumeration of possible load states.

(2) Values:

a) **HIGH_THRESHOLD**

b) **LOW_THRESHOLD**

c) **NORMAL**

d) **OVERLOAD**

e) **UNDEFINED**

(3) Used by: LoadStateAtReferenceLocation.

4.55. <<enumeration>> Network

(1) Enumerates the possible network types used for flight message exchange.

(2) Values:

a) **AFTN**

b) **OTHER**

c) **SITA**

(3) Used by: NetworkAddress.

4.56. NetworkAddress

(1) Address on a network.

(2) Attributes:

a) **Network network** *(Mandatory)*
Type of network.

b) **NetworkAddress_DataType address** *(Mandatory)*
Examples:EGGOZDZX, LOVVZRZO,...

(3) Used by: MessageOriginator.

4.57. typedef<string> NetworkAddress_DataType

(1) Network Address data type.

(2) Pattern: `ANY{1,8}`

(3) Used by: NetworkAddress.

4.58. <<abstract>> NonPublishedPoint

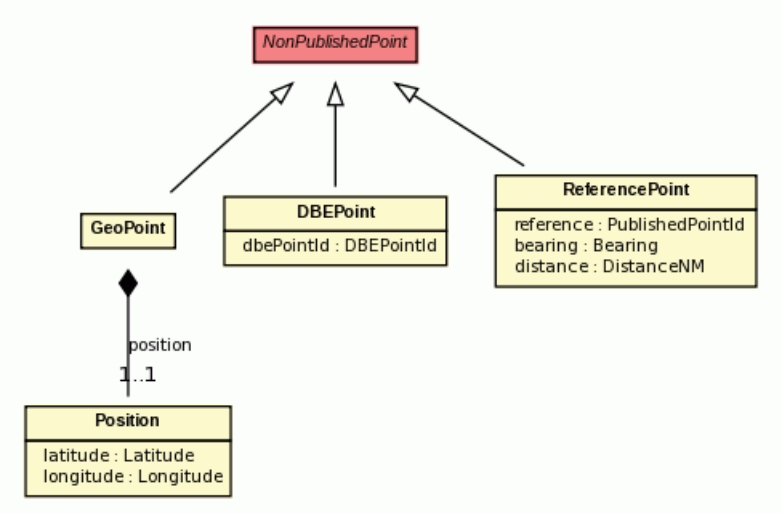

Figure 4.4. <<abstract>> NonPublishedPoint Class Diagram

(1) Represents a non-published point, i.e. expressed inline (not via a reference).

(2) Extended by: DBEPoint, ReferencePoint, GeoPoint.

(3) Used by: ICAOPoint, OtherAerodromeDesignation.

4.59. OTMV

(1) Definition of an OTMV.

(2) Attributes:

a) **TrafficVolumeId trafficVolume** *(Mandatory)*
Traffic volume to which this OTMV applies.

b) **DurationHourMinute otmvDuration** *(Mandatory)*

The duration of this OTMV.

c) **OTMVPeak peak** *(Optional)*
The peak configuration of this OTMV.

d) **OTMVSustained sustained** *(Optional)*
The sustained configuration of this OTMV.

e) **string remark** *(Optional)*
Constraint: Pattern: TEXT{1,255}

(3) Used by: PlannedOTMV.

4.60. OTMVPeak

(1) OTMV peak data.

(2) Attributes:

a) **OTMVThreshold threshold** *(Mandatory)*
Peak threshold of an OTMV.

(3) Used by: OTMV.

4.61. OTMVPlan

(1) OTMV plan for a given traffic volume on a given day.

(2) An OTMV plan is a special plan in the sense that for a given traffic volume there can be multiple OTMV durations. For each of these durations there exists a plan covering the full day (completely independent of any other duration). In update mode, only one duration can be updated in a single request. For a specific (traffic volume, OTMV duration) plan, there can be cases where there is no data known. So there are (traffic volume, OTMV duration) pairs where there is NO_DATA at all or only for some periods.

(3) In a retrieval context, the plan for a (traffic volume, OTMV duration) is said to be 'complete' in the sense that it contains all the plan entries from all involved data sources (including NO_DATA data source in case no info is known).

(4) In an update context, the plan can be complete (if the B2B client designer decided to overwrite the default CACD values) or limited to the (full list) of (pre-)tactical updates with the gaps marked as AIRSPACE (meaning in update context: either NO_DATA or CACD) datasource (to obtain a complete time partition).

(5) In any case, periods in the time partition marked with datasource AIRSPACE correspond to removing any potential (pre-)tactical update and hence reset the corresponding values to the CACD definition for that period.

(6) Inherits from: TacticalConfigurationPlan.

(7) Attributes:

a) **TrafficVolumeId trafficVolume** *(Mandatory)*
Traffic volume to which this OTMV plan applies.

b) **Map<DurationHourMinute,OTMVPlanForDuration> otmvPlans** *(Mandatory)*
The set of durations for which there are OTMV updates and for each duration, the OTMV plan.
If only one specific duration was requested, then the map will only contain the OTMV plan for that duration.
Constraints:

i) Size must be comprised between 0 and ∞.

ii) See INCOMPLETE_SCHEDULE

iii) See INVALID_SCHEDULE

iv) See ONLY_ONE_ENTRY_CAN_BE_UPDATED_IN_PLAN

(8) Constraints:

a)

Name	INCOMPLETE_SCHEDULE
Attribute	otmvPlans
Context	CapacityPlanUpdateRequest, OTMVPlanUpdateRequest, RunwayConfigurationPlanUpdateRequest, SectorConfigurationPlanUpdateRequest, TrafficVolumeActivationPlanUpdateRequest
Description	clientSchedule has gaps and/or overlaps in the time partition or is not covering exactly one day.

b)

Name	ONLY_ONE_ENTRY_CAN_BE_UPDATED_IN_PLAN
Attribute	otmvPlans
Context	CapacityPlanUpdateRequest, OTMVPlanUpdateRequest, RunwayConfigurationPlanUpdateRequest, SectorConfigurationPlanUpdateRequest, TrafficVolumeActivationPlanUpdateRequest
Description	Only one entry in the otmvPlans map attribute can be updated (i.e., for one duration).

c)

Name	INVALID_SCHEDULE
Attribute	otmvPlans
Context	CapacityPlanUpdateRequest, OTMVPlanUpdateRequest, RunwayConfigurationPlanUpdateRequest, SectorConfigurationPlanUpdateRequest, TrafficVolumeActivationPlanUpdateRequest

Description	The duration key used in the `otmvPlans` map attribute has to be equal to the duration of all OTMVs linked to that duration key.

(9) Used by: OTMVPlanUpdateRequest, OTMVPlanUpdateReply, OTMVPlanRetrievalReply.

4.62. `OTMVPlanForDuration`

(1) The OTMV plan for a specific duration.

(2) Attributes:

a) **`Set<PlannedOTMV> nmSchedule`** *(Contextual)*
The plan resulting from the superimposition of all constraints from all data sources, which the NM system is using as plan for its own calculations.
The NM schedule exposes the complete time partition of the configurations for the day, i.e. the data coming from the various data sources contributing to the NM view of the plan: these data sources are defined in PlanDataSource.
The NM schedule cannot be updated directly by the caller; it is the outcome of updates via the various data sources.
The possible values of dataSource are limited to `NO_DATA`, `AIRSPACE` and `TACTICAL`.
Presence:

i) Must be null in `CapacityPlanUpdateRequest`, `OTMVPlanUpdateRequest`, `RunwayConfigurationPlanUpdateRequest`, `SectorConfigurationPlanUpdateRequest`, `TrafficVolumeActivationPlanUpdateRequest`

ii) Mandatory otherwise.
Constraint: Size must be comprised between 0 and ∞.

b) **`Set<PlannedOTMV> clientSchedule`** *(Mandatory)*
(Pre-)tactical OTMVs associated to their applicability period, as maintained by the client. This plan contains only the updated configurations together with an indication that the default CACD values (AIRSPACE datasource) apply when not updated (cf. PlanDataSource). The actual CACD values for these CACD periods can be found in the `nmSchedule`
In an update context, the `clientSchedule` can be complete (if the B2B client designer decided to overwrite the default CACD values) or limited to the actual differences with regards to the CACD defaults. This is a B2B client designer's decision and depends on how CACD wants to be used in combination with the B2B. If the `clientSchedule` only contains the actual differences with regards to the CACD defaults, then the `clientschedule` still needs to contain the full list of (pre-) tactical updates and for the non (pre-) tactically updated periods, an explicit indication that the CACD values need to be used (but without repeating the CACD values themselves). So in any case, the `clientschedule` needs to be a complete time partition for the full day.
In any case, it is of drastic importance to understand that an entry in the schedule (i.e. a period) overwrites all CACD values in that period.
The possible values of dataSource are limited to `AIRSPACE` and `TACTICAL` - the `AIRSPACE` value meaning that the value associated to the applicabilityPeriod is the CACD one (i.e.

CACD or that there is no data defined in the CACD), and the `TACTICAL` value meaning that this plan entry corresponds to the explicit tactical update expressed via otmv.
Constraint: Size must be comprised between 0 and ∞.

(3) Used by: OTMVPlan.

4.63. `OTMVSustained`

(1) OTMV sustained data.

(2) Attributes:

a) **`OTMVThreshold threshold`** *(Mandatory)*
Sustained threshold of an OTMV.

b) **`int crossingOccurrences`** *(Mandatory)*
Number of crossing occurrences of the sustained threshold within `elapsed`, before this OTMV triggers an alert.
Constraint: Range: `[1,9999[`.

c) **`DurationHourMinute elapsed`** *(Mandatory)*
Duration of the time window on which `crossingOccurrences` are counted.

(3) Used by: OTMV.

4.64. `typedef<int> OTMVThreshold`

(1) Threshold type used in OTMVs, in flights per user-defined OTMV duration.

(2) Range: `[0,9998[`.

(3) Used by: OTMVSustained, OTMVPeak.

4.65. `<<strict enumeration>> PlanDataSource`

(1) A source of data for a plan, within or outside NM.

(2) Values:

a) **`AIRSPACE`**
Data from the NM Airspace system (CACD), the data is either baselined with the AIRAC or results from a live update.

b) **`MEASURE`**
Data resulting from the application of a measure in the NM system.

c) **`NO_DATA`**
There is no data defined in NM.

d) **`TACTICAL`**

Following a tactical update, typically, from the NOP user (B2B or B2C).

(3) Used by: PlannedCapacity, RunwayConfiguration, PlannedSectorConfigurationActivation, PlannedRunwayConfigurations, PlannedOTMV, PlannedTrafficVolumeActivation.

4.66. PlannedCapacity

(1) An entry within a capacity plan.

(2) Attributes:

a) **DateTimeMinutePeriod applicabilityPeriod** *(Mandatory)*
The time period in the plan to which this entry applies.

b) **PlanDataSource dataSource** *(Mandatory)*
The data source of this entry in the plan.
Constraints:

i) See INVALID_DATASOURCE

ii) See VALUE_CANNOT_BE_NULL

iii) See VALUE_MUST_BE_NULL

c) **Capacity capacity** *(Optional)*
Capacity
Constraints:

i) See VALUE_CANNOT_BE_NULL

ii) See VALUE_MUST_BE_NULL

(3) Constraints:

a)

Name	VALUE_CANNOT_BE_NULL
Attributes	dataSource, capacity
Description	capacity cannot be null if dataSource is TACTICAL. For nmSchedule: capacity cannot be null if dataSource is different from NO_DATA.

b)

Name	VALUE_MUST_BE_NULL
Attributes	dataSource, capacity
Description	capacity must be null if dataSource is NO_DATA. For clientSchedule: capacity must be null if dataSource is not TACTICAL.

c)

Name	INVALID_DATASOURCE
Attribute	dataSource
Description	NO_DATA is not a valid dataSource in an update context. MEASURE is not a valid dataSource in an update context.

(4) Used by: CapacityPlan.

4.67. PlannedOTMV

(1) An entry within an OTMV plan.

(2) Attributes:

a) **DateTimeMinutePeriod applicabilityPeriod** *(Mandatory)*
The time period in the plan to which this entry applies.

b) **PlanDataSource dataSource** *(Mandatory)*
The data source of this entry in the plan.
Constraints:

i) See INVALID_DATASOURCE

ii) See VALUE_CANNOT_BE_NULL

iii) See VALUE_MUST_BE_NULL

c) **OTMV otmv** *(Optional)*
OTMV
Constraints:

i) See VALUE_CANNOT_BE_NULL

ii) See VALUE_MUST_BE_NULL

(3) Constraints:

a)

Name	VALUE_CANNOT_BE_NULL
Attributes	dataSource, otmv
Description	otmv cannot be null if dataSource is TACTICAL. For nmSchedule: otmv cannot be null if dataSource is different from NO_DATA.

b)

Name	VALUE_MUST_BE_NULL
Attributes	dataSource, otmv
Description	otmv must be null if dataSource is NO_DATA.

	For `clientSchedule`: `otmv` must be null if `dataSource` is not `TACTICAL`.

c)

Name	`INVALID_DATASOURCE`
Attribute	`dataSource`
Description	`NO_DATA` is not a valid `dataSource` in an update context. `MEASURE` is not a valid `dataSource` in an update context.

(4) Used by: OTMVPlanForDuration.

4.68. PlannedRunwayConfigurations

(1) An entry within an aerodrome runway configuration plan.

(2) Attributes:

a) **DateTimeMinutePeriod applicabilityPeriod** *(Mandatory)*
The time period in the plan to which this entry applies.

b) **PlanDataSource dataSource** *(Mandatory)*
The data source of this entry in the plan.

c) **Set<RunwayConfiguration> runwayConfigurations** *(Contextual)*
Full set of runway configurations for an aerodrome within `applicabilityPeriod`
Presence:

i) Optional in `CapacityPlanUpdateRequest`, `OTMVPlanUpdateRequest`, `RunwayConfigurationPlanUpdateRequest`, `SectorConfigurationPlanUpdateRequest`, `TrafficVolumeActivationPlanUpdateRequest`

ii) Mandatory otherwise.
Constraint: Size must be comprised between 0 and ∞.

(3) Used by: RunwayConfigurationPlan.

4.69. PlannedSectorConfigurationActivation

(1) An entry within a sector configuration plan - its presence in the plan denotes the activation of the associated sector configuration.

(2) Attributes:

a) **DateTimeMinutePeriod applicabilityPeriod** *(Mandatory)*
The time period in the plan to which this entry applies.

b) **PlanDataSource dataSource** *(Mandatory)*
The data source of this entry in the plan.

Constraints:

i) See INVALID_DATASOURCE

ii) See VALUE_CANNOT_BE_NULL

iii) See VALUE_MUST_BE_NULL

c) **SectorConfigurationId sectorConfigurationId** *(Optional)*
Unique Id of the sector configuration, as known in the NM system.
Constraints:

i) See VALUE_CANNOT_BE_NULL

ii) See VALUE_MUST_BE_NULL

(3) Constraints:

a)

Name	VALUE_CANNOT_BE_NULL
Attributes	dataSource, sectorConfigurationId
Description	sectorConfigurationId cannot be null if dataSource is TACTICAL. This is applicable for both nmSchedule and clientSchedule.

b)

Name	VALUE_MUST_BE_NULL
Attributes	dataSource, sectorConfigurationId
Description	sectorConfigurationId must be null if dataSource is not TACTICAL. This is only applicable for the clientSchedule.

c)

Name	INVALID_DATASOURCE
Attribute	dataSource
Description	NO_DATA is not a valid dataSource in an update context. MEASURE is not a valid dataSource in an update context.

(4) Used by: SectorConfigurationPlan.

4.70. PlannedTrafficVolumeActivation

(1) Denotes whether a traffic volume is active or not during a period.

(2) Attributes:

a) **DateTimeMinutePeriod applicabilityPeriod** *(Mandatory)*
The time period in the plan to which this entry applies.

b) **PlanDataSource dataSource** *(Mandatory)*
The data source of this entry in the plan.
Constraints:

i) See INVALID_DATASOURCE

ii) See VALUE_CANNOT_BE_NULL

iii) See VALUE_MUST_BE_NULL

c) **boolean active** *(Optional)*
Indicates if the traffic volume is active or not during `applicabilityPeriod`
Constraints:

i) See VALUE_CANNOT_BE_NULL

ii) See VALUE_MUST_BE_NULL

(3) Constraints:

a)

Name	VALUE_CANNOT_BE_NULL
Attributes	dataSource, active
Description	active cannot be null if dataSource is TACTICAL. For nmSchedule: active must be true if dataSource is AIRSPACE.

b)

Name	VALUE_MUST_BE_NULL
Attributes	dataSource, active
Description	active must be null if dataSource is NO_DATA. For clientSchedule: active must be null if dataSource is not TACTICAL.

c)

Name	INVALID_DATASOURCE
Attribute	dataSource
Description	NO_DATA is not a valid dataSource in an update context. MEASURE is not a valid dataSource in an update context.

(4) Used by: TrafficVolumeActivationPlan.

4.71. ProcedureICAOId

(1) ICAO structured id for a departure/arrival procedure.

(2) Attributes:

a) **PublishedPointId endPoint** *(Mandatory)*

Id of the end point of the procedure.

b) **`int versionNumber`** *(Mandatory)*
Version number of the procedure.
Constraint: Range:] - ∞,∞[.

c) **`string routeIndicator`** *(Mandatory)*
Indicator that distinguishes the different procedures using the same point. Must be exactly one character in ["A", "Z"].

(3) Used by: CDMProvisionalInfo, FlightPointRoute, UpdateDPIRequest, CDMInfo.

4.72. `typedef<string> PublishedPointId`

(1) ICAO identifier for way points and navigation aid points.

(2) Pattern: `(UALPHA|DIGIT){1,5}`

(3) Used by: FourDTrajectoryPoint, FlightListByPointRequest, TrafficCountsByPointRequest, DBEOrPublishedPointId, ICAOPoint, RoutingAssistanceRequest, ReferencePoint, ReferenceLocationPublishedPoint, ProcedureICAOId.

4.73. `<<abstract>> ReferenceLocation`

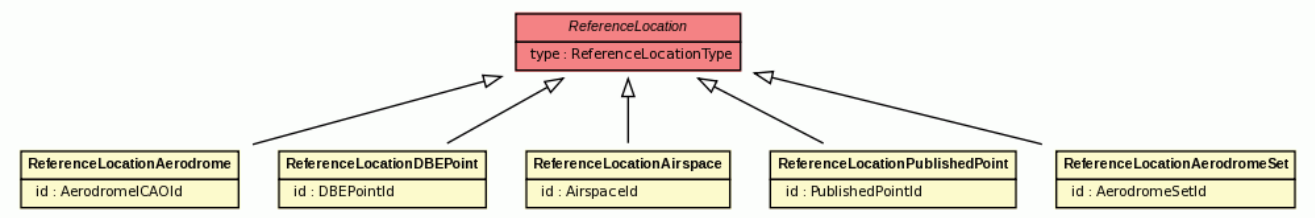

Figure 4.5. `<<abstract>> ReferenceLocation` Class Diagram

(1) Abstract reference location, i.e. a reference to an aerodrome, an aerodrome set, an airspace or a point.

(2) Attributes:

a) **`ReferenceLocationType type`** *(Mandatory)*
Specifies the type of the reference location and therefore the attribute above selected to pass the location id.

(3) Extended by: ReferenceLocationDBEPoint, ReferenceLocationAerodrome, ReferenceLocationPublishedPoint, ReferenceLocationAerodromeSet, ReferenceLocationAirspace.

(4) Used by: TrafficVolumeLocation, FlightHotspotLocation, RegulationOrMCDMOnly, FlightAtfcmMeasureLocation, FlightRegulationLocation.

4.74. ReferenceLocationAerodrome

(1) Reference to an aerodrome.

(2) The type must be ReferenceLocationType.AERODROME.

(3) Inherits from: ReferenceLocation.

(4) Attributes:

a) **AerodromeICAOId id** *(Mandatory)*
ICAO id of the referenced aerodrome.

4.75. ReferenceLocationAerodromeSet

(1) Reference to an aerodrome set.

(2) The type must be ReferenceLocationType.AERODROME_SET.

(3) Inherits from: ReferenceLocation.

(4) Attributes:

a) **AerodromeSetId id** *(Mandatory)*
Id of the referenced aerodrome set.

4.76. ReferenceLocationAirspace

(1) Reference to an airspace.

(2) The type must be ReferenceLocationType.AIRSPACE.

(3) Inherits from: ReferenceLocation.

(4) Attributes:

a) **AirspaceId id** *(Mandatory)*
Id of the referenced airspace.

4.77. ReferenceLocationDBEPoint

(1) Reference to a DBE point (internal to NM).

(2) The type must be ReferenceLocationType.DBE_POINT.

(3) Inherits from: ReferenceLocation.

(4) Attributes:

a) **DBEPointId id** *(Mandatory)*

Id of the referenced DBE point.

4.78. ReferenceLocationPublishedPoint

(1) Reference to a published point.

(2) The type must be ReferenceLocationType.PUBLISHED_POINT.

(3) Inherits from: ReferenceLocation.

(4) Attributes:

a) **PublishedPointId id** *(Mandatory)*
Id of the referenced published point.

4.79. <<enumeration>> ReferenceLocationType

(1) Enumerates reference location types.

(2) Values:

a) **AERODROME**

b) **AERODROME_SET**

c) **AIRSPACE**

d) **DBE_POINT**

e) **PUBLISHED_POINT**

(3) Used by: ReferenceLocation.

4.80. ReferencePoint

(1) Represents a non-published point expressed via a bearing and distance with regards to a reference published point.

(2) Inherits from: NonPublishedPoint.

(3) Attributes:

a) **PublishedPointId reference** *(Mandatory)*
Reference point.

b) **Bearing bearing** *(Mandatory)*
Bearing with regards to the reference point.

c) **DistanceNM distance** *(Mandatory)*
Distance to the reference point.

4.81. typedef<string> RestrictionId

(1) Unique id of a restriction, allocated by NM.

(2) Pattern: `(UALPHA|DIGIT){1,10}`

(3) Used by: FlightRestriction.

4.82. typedef<string> RouteId

(1) ICAO identifier of a route.

(2) Pattern: `(UALPHA|DIGIT){1,7}`

(3) Used by: FlightPointRoute, TrajectorySegment.

4.83. RunwayConfiguration

(1) A runway configuration at some point in time.

(2) Attributes:

a) **RunwayId runway** *(Mandatory)*
Runway to which this configuration applies.

b) **RunwayUsage usage** *(Optional)*
Usage of the runway in this configuration.

c) **PlanDataSource runwayUsageDataSource** *(Mandatory)*
The data source of the `runwayUsage` for this entry in the plan.
Note that NM prefers not to have any CACD (runwayUsageDataSource = AIRSPACE) runway usage when tactically updating runway configurations. If an AIRSPACE runwayUsageDataSource applies, then this means that in NM profile computations, the RunwayUsage is considered unknown (as it is difficult to predict accurately up to one month in advance the wind/runway activation schedule).

d) **DurationHourMinute departureTaxiTime** *(Optional)*
Departure taxi time.
Note that the departure taxi time must be specified even if the `usage` is `ARRIVAL` or `INACTIVE`, so that in the exceptional case where the runway would be used in another way (indicated via `DPI`) the NM system could still use this taxi time for computing the flight.

e) **PlanDataSource departureTaxiTimeDataSource** *(Mandatory)*
The data source of the `departureTaxiTime` for this entry in the plan.

f) **DurationHourMinute timeToInsertInSequence** *(Optional)*
Time to insert aircraft in sequence at the aerodrome of departure.

Note that the timeToInsertInSequence must be specified even if the usage is ARRIVAL or INACTIVE, so that in the exceptional case where the runway would be used in another way (indicated via DPI) the NM system could still use this taxi time for computing the flight.

g) **PlanDataSource timeToInsertInSequenceDataSource** *(Mandatory)*
The data source of the timeToInsertInSequence for this entry in the plan.

h) **DurationHourMinute timeToRemoveFromSequence** *(Optional)*
Time to remove aircraft from sequence at the aerodrome of departure.
Note that the timeToRemoveFromSequence must be specified even if the usage is ARRIVAL or INACTIVE, so that in the exceptional case where the runway would be used in another way (indicated via DPI) the NM system could still use this taxi time for computing the flight.

i) **PlanDataSource timeToRemoveFromSequenceDataSource** *(Mandatory)*
The data source of the timeToRemoveFromSequence for this entry in the plan.

j) **DurationHourMinute arrivalTaxiTime** *(Optional)*
Arrival taxi time.
Note that the departure taxi time must be specified even if the usage is DEPARTURE or INACTIVE.

k) **PlanDataSource arrivalTaxiTimeDataSource** *(Mandatory)*
The data source of the arrivalTaxiTime for this entry in the plan.

(3) Used by: PlannedRunwayConfigurations.

4.84. RunwayConfigurationPlan

(1) Runway configuration plan for a given aerodrome on a given day.

(2) A RunwayConfiguration plan is a special plan in the sense that all known runways need to be specified for a given period (PlannedRunwayConfiguration) or else the entire period needs to be marked CACD (AIRSPACE PlanDataSource). In addition, for a (pre-)tactically updated period it is possible to (pre-)tactically update selected runway attributes for selected runways. So each runway attribute (usage, departureTaxTime,...) has an associated dataSource attribute allowing to indicate if that attribute needs to be updated or if CACD data is to be used for that attribute, for that runway and for that specific period. The only constraint is: if a period (PlannedRunwayConfiguration) is marked as TACTICAL updated, then at least one of the attributes of one of the runways needs to be TACTICAL updated (otherwise INVALID_INPUT replyStatus).

(3) In addition the RunwayUsage attribute is special in the sense that all runways for a specific TACTICAL updated period need to be either all CACD, or else a mix of DEPARTURE/ARRIVAL/DEPARTURE_ARRIVALl/INACTIVE).

(4) Note that NM prefers not to have any CACD (AIRSPACE datasource) RunwayUsage when tactically updating runway configurations. If a CACD RunwayUsage applies, then this means that in NM profile computations, the RunwayUsage is considered unknown (as it is difficult to predict accurately up to one month in advance the wind/runway activation schedule).

(5) In a retrieval context, the plan is said to be 'complete' in the sense that it contains all the plan entries from all involved data sources.

(6) In an update context, the plan can be complete (if the B2B client designer decided to overwrite the default CACD values) or limited to the (full list) of (pre-)tactical updates with the gaps marked as AIRSPACE (meaning in update context: CACD) datasource (to obtain a complete time partition).

(7) In any case, periods in the time partition marked with datasource AIRSPACE, correspond to removing any potential (pre-)tactical update and hence reset the corresponding values to the CACD definition for that period.

(8) Inherits from: `TacticalConfigurationPlan`.

(9) Attributes:

a) **`AerodromeICAOId` `aerodrome`** *(Mandatory)*
Aerodrome to which this runway configuration plan belongs.

b) **`Set<RunwayId>` `knownRunwayIds`** *(Mandatory)*
The list of runways that NM knows for the aerodrome, regardless to their usage, e.g. an inactive runway is part of this set.
Constraint: Size must be comprised between 0 and ∞.

c) **`Set<PlannedRunwayConfigurations>` `nmSchedule`** *(Contextual)*
The plan resulting from the superimposition of all constraints from all data sources, which the NM system is using as plan for its own calculations.
The NM schedule exposes the complete time partition of the configurations for the day, i.e. the data coming from the various data sources contributing to the NM view of the plan: these data sources are defined in PlanDataSource.
The NM schedule cannot be updated directly by the caller; it is the outcome of updates via the various data sources.
Presence:

i) Must be null in `CapacityPlanUpdateRequest`, `OTMVPlanUpdateRequest`, `RunwayConfigurationPlanUpdateRequest`, `SectorConfigurationPlanUpdateRequest`, `TrafficVolumeActivationPlanUpdateRequest`

ii) Mandatory otherwise.

Constraint: Size must be comprised between 0 and ∞.

d) **`Set<PlannedRunwayConfigurations>` `clientSchedule`** *(Mandatory)*
(Pre-)tactical runway configuration of the aerodrome associated to their applicability period, as maintained by the client. This plan contains only the updated configurations together with an indication that the default CACD values apply when not updated (cf. PlanDataSource). The actual CACD values for these CACD periods can be found in the `nmSchedule`.
In an update context, the clientSchedule can be complete (if the B2B client designer decided to overwrite the default CACD values) or limited to the actual differences with regards to the CACD defaults. This is a B2B client designer's decision and depends on how CACD wants to be used in combination with the B2B. If the clientSchedule only contains the actual

differences with regards to the CACD defaults, then the clientschedule still needs to contain the full list of (pre-) tactical updates and for the non (pre-) tactically updated periods, an explicit indication that the CACD values need to be used (but without repeating the CACD values themselves). So in any case, the clientschedule needs to be a complete time partition for the full day.
In any case, it is of drastic importance to understand that an entry in the schedule (i.e. a period):

i) Must be complete in the sense that all the aerodrome runways must be present in the entry

ii) The runway configuration values provided for that period overwrite all CACD values for that same period

Constraints:

i) Size must be comprised between 0 and ∞.

ii) See `INCOMPLETE_SCHEDULE`

(10) Constraint:

a)

Name	`INCOMPLETE_SCHEDULE`
Attribute	`clientSchedule`
Description	`clientSchedule` has gaps and/or overlaps in the time partition or is not covering exactly one day.

(11) Used by: RunwayConfigurationPlanUpdateRequest, RunwayConfigurationPlanRetrievalReply, RunwayConfigurationPlanUpdateReply.

4.85. `typedef<string> RunwayId`

(1) Unique id of a runway within an aerodrome.

(2) Pattern: `DIGIT{2}(L|R|C| ){0,1}`

(3) Used by: RunwayConfiguration, RunwayConfigurationPlan.

4.86. `<<enumeration>> RunwayUsage`

(1) Possible usages of a runway.

(2) Values:

a) **`ARRIVAL`**
The runway is used for arrivals only.
Note that an arrival runway is not considered by the NM system for departure when processing flights, but `DPI` (Departure Planning Information) messages are still able to use an arrival runway.

b) **DEPARTURE**
The runway is used for departures only.

c) **DEPARTURE_ARRIVAL**
The runway is used for both departures and arrivals.

d) **INACTIVE**
The runway is neither used for depatures or arrivals.
Note that an inactive runway is not considered by the NM system for departure when processing flights, but DPI (Departure Planning Information) messages are still able to use an inactive runway.

(3) Used by: RunwayConfiguration.

4.87. typedef<string> SectorConfigurationId

(1) Unique id of a sector configuration.

(2) Pattern: `(UALPHA|DIGIT|.){1,6}`

(3) Used by: PlannedSectorConfigurationActivation, SectorConfigurationPlan.

4.88. SectorConfigurationPlan

(1) Sector configuration plan for a given AUA or sector cluster on a given day.

(2) In a retrieval context, the plan is said to be 'complete' in the sense that it contains all the plan entries from all involved data sources.

(3) In an update context, the plan can be complete (if the B2B client designer decided to overwrite the default CACD values) or limited to the (full list) of (pre-)tactical updates with the gaps marked as AIRSPACE datasource (to obtain a complete time partition).

(4) In any case, periods in the time partition marked with datasource AIRSPACE correspond to removing any potential (pre-)tactical update and hence reset the corresponding values to the CACD definition for that period.

(5) Inherits from: TacticalConfigurationPlan.

(6) Attributes:

a) **AirspaceId airspace** *(Mandatory)*
AUA or sector cluster to which these sector configurations belong.

b) **Map<SectorConfigurationId,Set<AirspaceId>> knownConfigurations** *(Contextual)*
The set of sector configuration ids that NM knows for the AUA or sector cluster, and for each sector configuration, the set of sectors that compose it.
Presence:

i) Must be null in CapacityPlanUpdateRequest, OTMVPlanUpdateRequest, RunwayConfigurationPlanUpdateRequest, SectorConfigurationPlanUpdateRequest, TrafficVolumeActivationPlanUpdateRequest

ii) Mandatory otherwise.

Constraints:

i) Size must be comprised between 0 and ∞.

ii) Item size must be comprised between 0 and ∞.

c) **Set<PlannedSectorConfigurationActivation> nmSchedule** *(Contextual)*

The plan resulting from the superimposition of all constraints from all data sources, which the NM system is using as plan for its own calculations.

The NM schedule exposes the complete time partition of the configurations for the day, i.e. the data coming from the various data sources contributing to the NM view of the plan: these data sources are defined in PlanDataSource.

The NM schedule cannot be updated directly by the caller; it is the outcome of updates via the various data sources.

The possible values of dataSource are limited to AIRSPACE and TACTICAL - the AIRSPACE value meaning that the value associated to the applicabilityPeriod is the CACD one, and the TACTICAL value meaning that this plan entry corresponds to the explicit tactical update expressed via sectorConfigurationId.

Presence:

i) Must be null in CapacityPlanUpdateRequest, OTMVPlanUpdateRequest, RunwayConfigurationPlanUpdateRequest, SectorConfigurationPlanUpdateRequest, TrafficVolumeActivationPlanUpdateRequest

ii) Mandatory otherwise.

Constraint: Size must be comprised between 0 and ∞.

d) **Set<PlannedSectorConfigurationActivation> clientSchedule** *(Mandatory)*

(Pre-)tactical sector configuration activations of the AUA or sector cluster associated to their applicability period, as maintained by the client. This plan contains only the updated configurations together with an indication that the default CACD values apply when not updated (cf. PlanDataSource). The actual CACD values for these CACD periods can be found in the nmSchedule

In an update context, the clientSchedule can be complete (if the B2B client designer decided to overwrite the default CACD values) or limited to the actual differences with regards to the CACD defaults. This is a B2B client designer's decision and depends on how CACD wants to be used in combination with the B2B. If the clientSchedule only contains the actual differences with regards to the CACD defaults, then the clientschedule still needs to contain the full list of (pre-) tactical updates and for the non (pre-) tactically updated periods, an explicit indication that the CACD values need to be used (but without repeating the CACD values themselves). So in any case, the clientschedule needs to be a complete time partition for the full day.

In any case, it is of drastic importance to understand that an entry in the schedule (i.e. a period) overwrites all CACD values in that period.

The possible values of dataSource are limited to AIRSPACE and TACTICAL - the AIRSPACE value meaning that the value associated to the applicabilityPeriod is the CACD one, and the TACTICAL value meaning that this plan entry corresponds to the explicit tactical update expressed via sectorConfigurationId.
Constraints:

i) Size must be comprised between 0 and ∞.

ii) See INCOMPLETE_SCHEDULE

(7) Constraint:

a)

Name	INCOMPLETE_SCHEDULE
Attribute	clientSchedule
Description	clientSchedule has gaps and/or overlaps in the time partition or is not covering exactly one day.

(8) Used by: SectorConfigurationPlanRetrievalReply, SectorConfigurationPlanUpdateRequest, SectorConfigurationPlanUpdateReply.

4.89. <<enumeration>> SpeedUnit

(1) Enumerates the supported speed units.

(2) Values:

Value	Description
FEET_PER_MINUTE	Feet per minute.
KILOMETERS_PER_HOUR	Kilometers per hour.
KNOTS	Nautical miles.
MACH_NUMBER	Mach number. Related to the speed of sound.
UNDEFINED	Undefined.

Table 4.3. <<enumeration>> SpeedUnit

(3) Used by: AirSpeed.

4.90. <<abstract>> TacticalConfigurationPlan

(1) Common information for all configuration plans on a given day.

(2) Attributes:

a) **PlanDataId dataId** *(Mandatory)*
Opaque identifier representing the version of this plan.

Constraint: See INCONSISTENT_DATAID_AND_DATASET_TYPE

b) **Dataset dataset** *(Mandatory)*
Dataset to which the plan belongs.
Constraints:

i) See INCONSISTENT_DATAID_AND_DATASET_TYPE

ii) See INCONSISTENT_DAY_AND_DATASET_TYPE

c) **DateYearMonthDay day** *(Mandatory)*
Day for which this plan is valid.
Constraint: See INCONSISTENT_DAY_AND_DATASET_TYPE

d) **boolean planTransferred** *(Optional)*
Indicates if the plan has been transferred to the OPERATIONAL dataset. When false, it means that the most up-to-date data can be found in the FORECAST dataset.

e) **boolean planCutOffReached** *(Optional)*
Indicates if the plan can still be updated in the FORECAST dataset, i.e. if the forecast cut-off time has been reached or not.

(3) Constraints:

a)

Name	INCONSISTENT_DATAID_AND_DATASET_TYPE
Attributes	dataId, dataset
Description	The dataId must match the one returned by the corresponding retrieval request for the given dataset.

b)

Name	INCONSISTENT_DAY_AND_DATASET_TYPE
Attributes	day, dataset
Description	The day must be in [D-5, D] depending on dataset type i.e.: i) day must be in [D-5, D] in an FORECAST context. ii) day must be in [D-1, D] in an OPERATIONAL context. iii) day has no range constraint in a SIMULATION context.

(4) Extended by: SectorConfigurationPlan, TrafficVolumeActivationPlan, CapacityPlan, RunwayConfigurationPlan, OTMVPlan.

4.91. <<abstract>> TacticalConfigurationRetrievalRequest

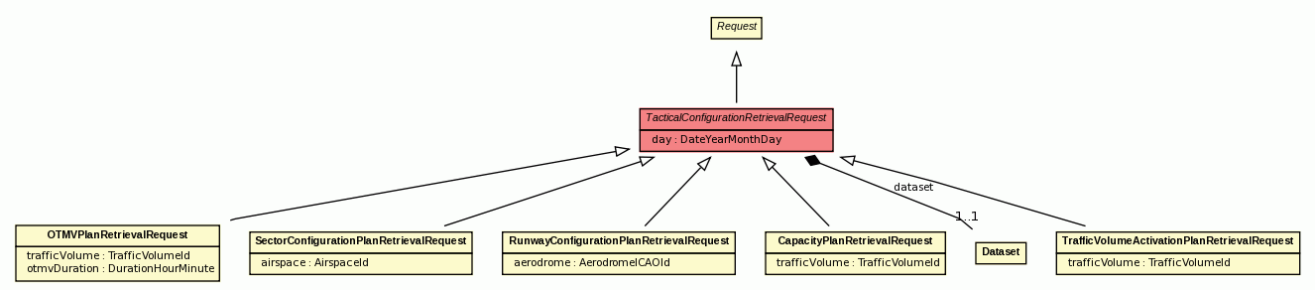

Figure 4.6. <<abstract>> TacticalConfigurationRetrievalRequest Class Diagram

(1) Abstract request to retrieve a tactical configuration (runway, sector, capacity, TV, OTMV) on a given day.

(2) Inherits from: Request.

(3) Attributes:

a) **Dataset dataset** *(Mandatory)*
Dataset for which the tactical configuration is requested.
Constraint: See INCONSISTENT_DAY_AND_DATASET_TYPE

b) **DateYearMonthDay day** *(Mandatory)*
Day for which the tactical configuration is requested.
Constraint: See INCONSISTENT_DAY_AND_DATASET_TYPE

(4) Constraint:

a)

Name	INCONSISTENT_DAY_AND_DATASET_TYPE
Attributes	day, dataset
Description	The day must be in [D-5, D] depending on dataset type i.e.: i) day must be in [D-5, D] in an FORECAST context. ii) day must be in [D-1, D] in an OPERATIONAL context. iii) day has no range constraint in a SIMULATION context.

(5) Extended by: OTMVPlanRetrievalRequest, TrafficVolumeActivationPlanRetrievalRequest, SectorConfigurationPlanRetrievalRequest, CapacityPlanRetrievalRequest, RunwayConfigurationPlanRetrievalRequest.

4.92. TrafficVolumeActivationPlan

(1) Activation plan for a given traffic volume on a given day.

(2) A TrafficVolumeActivation plan is a special plan in the sense that, there can be cases where there is no data known. So there exist traffic volumes for which there is no data at all or only for some periods. In addition, non-activation is not defined in CACD. Therefore the absence of CACD data (NO_DATA datasource) means either no data known or in-active. Either way, the absence of data is considered by NM systems as an in-active traffic volume. In addition, sector configuration activations (tactically updated or CACD defined) also over-rule CACD activations of a traffic volume. They can activate or de-activate a traffic volume (also marked by AIRSPACE datasource). The above activations can be over-ruled (set active or in-active) by the tactical updates (B2B or HMI). In addition regulation measures can activate a traffic volume (over-ruling either CACD or B2B/HMI updates). The consolidated info can be found back in the `nmSchedule` attribute.

(3) So the (de-)activation of a traffic volume is determined hierarchically by NO_DATA (least priority), CACD (including derived from sector config), tactical updates (a.o. via B2B) , regulation measures (highest priority). In the client schedule the non regulated traffic volumes activations are maintained (just in case the regulation can be cancelled before the end of its applicability period).

(4) In a retrieval context, the plan is said to be 'complete' in the sense that it contains all the plan entries from all involved data sources (including NO_DATA data source in case no info is known).

(5) In an update context, the plan can be complete (if the B2B client designer decided to overwrite the default CACD values/sector configuration derived activation) or limited to the (full list) of (pre-)tactical updates with the gaps marked as AIRSPACE (meaning in update context: NO_DATA or CACD or derived from sector config) datasource (to obtain a complete time partition).

(6) In any case, periods marked with datasource AIRSPACE in the time partition correspond to removing any potential (pre-)tactical update and hence reset the corresponding values to the NO_DATA/CACD/"derived from sector config" definition for that period.

(7) Inherits from: `TacticalConfigurationPlan`.

(8) Attributes:

a) **`TrafficVolumeId` `trafficVolume`** *(Mandatory)*
Traffic volume to which this activation plan applies.

b) **`Set<PlannedTrafficVolumeActivation>` `nmSchedule`** *(Contextual)*
The plan resulting from the superimposition of all constraints from all data sources, which the NM system is using as plan for its own calculations.
The NM schedule exposes the complete time partition of the configurations for the day, i.e. the data coming from the various data sources contributing to the NM view of the plan: these data sources are defined in PlanDataSource.
The NM schedule cannot be updated directly by the caller; it is the outcome of updates via the various data sources.
In `nmSchedule` the possible values of dataSource are limited to `NO_DATA`, `AIRSPACE`, `TACTICAL` and `MEASURE`.
Note that NO_DATA in nmClientSchedule means either inactive or that no data has been specified.
Presence:

i) Must be null in CapacityPlanUpdateRequest, OTMVPlanUpdateRequest, RunwayConfigurationPlanUpdateRequest, SectorConfigurationPlanUpdateRequest, TrafficVolumeActivationPlanUpdateRequest

ii) Mandatory otherwise.

Constraint: Size must be comprised between 0 and ∞.

c) **Set<PlannedTrafficVolumeActivation> clientSchedule** *(Mandatory)*

(Pre-)tactical traffic volume activations associated to their applicability period, as maintained by the client. This plan contains only the updated configurations together with an indication (AIRSPACE datasource) that the default CACD/sector config derived values apply when not updated (cf. PlanDataSource). The actual activity for these airspace datasource periods can be found in the nmSchedule

In an update context, the clientSchedule can be complete (if the B2B client designer decided to overwrite the default CACD values) or limited to the actual differences with regards to the AIRSPACE defaults. This is a B2B client designer's decision and depends on how CACD wants to be used in combination with the B2B. If the clientSchedule only contains the actual differences with regards to the AIRSPACE defaults, then the clientschedule still needs to contain the full list of (pre-) tactical updates and for the non (pre-) tactically updated periods, an explicit indication that the AIRSPACE values need to be used (but without repeating the NO_DATA/CACD/sector config derived values themselves). So in any case, the clientschedule needs to be a complete time partition for the full day.

In any case, it is of drastic importance to understand that an entry in the schedule (i.e. a period) overwrites all CACD/sector config derived values in that period.

The possible values of dataSource are limited to AIRSPACE and TACTICAL - the AIRSPACE value meaning that the value associated to the applicabilityPeriod is the CACD one (i.e. CACD or the TV activation derived from a sector configuration or that there is no data defined in CACD), and the TACTICAL value meaning that this plan entry corresponds to the explicit tactical update expressed via active.

Constraints:

i) Size must be comprised between 0 and ∞.

ii) See INCOMPLETE_SCHEDULE

(9) Constraint:

a)

Name	INCOMPLETE_SCHEDULE
Attribute	clientSchedule
Description	clientSchedule has gaps and/or overlaps in the time partition or is not covering exactly one day.

(10) Used by: TrafficVolumeActivationPlanRetrievalReply, TrafficVolumeActivationPlanUpdateReply, TrafficVolumeActivationPlanUpdateRequest.

4.93. typedef<string> TrafficVolumeId

(1) NM unique identifier for a traffic volume.

(2) Pattern: `(ALPHA|DIGIT){1,8}`

(3) Used by: TrafficVolumeActivationPlan, HotspotId, OTMVPlan, TrafficVolumeLocation, TrafficVolumeActivationPlanRetrievalRequest, FlightListByTrafficVolumeRequest, CapacityPlanRetrievalRequest, CapacityPlan, TrafficCountsByTrafficVolumeRequest, OTMVPlanRetrievalRequest, HotspotListRequest, OTMV, ATFCMSituationRegulation, HotspotPlan.

4.94. typedef<string> TrafficVolumeIdWildcard

(1) NM identifier for a traffic volume, with basic wildcard support ("*" is replaced by 0 or more characters).

(2) Pattern: `(ALPHA|DIGIT|*){1,8}`

(3) Used by: MeasureListRequest, RegulationMessageFilter.

4.95. typedef<string> TrafficVolumeSetId

(1) NM unique identifier for a traffic volume set.

(2) Pattern: `(ALPHA|DIGIT){1,8}`

(3) Used by: TrafficVolumeLocation, RegulationOrMCDMOnlyListRequest, RegulationOrMCDMOnly.

4.96. typedef<string> TrafficVolumeSetIdWildcard

(1) Either a full traffic volume set id, or a simple wildcard for traffic volume set ids.

(2) Pattern: `(ALPHA|DIGIT|*){1,8}`

(3) Used by: MeasureListRequest, RegulationMessageFilter.

Chapter 5. PRE-OPS Testing

5.1. AUP/UUP testing

5.1.1. Introduction

(1) The B2B services on AUP/UUP are:

a) Creation of AUP/UUP

b) Update of AUP/UUP

c) Deletion of AUP/UUP

d) Functional expansion

e) Promotion to Ready

f) Demotion from Ready

(2) Manage AUP/UUP Release is and remains accessible via (CIAM) CHMI only and is limited to the CADF user. It is defined as:

a) Promotion to Released

b) Demotion from Released

c) Set next UUP WEF

(3) Each AMC is responsible of its area of concern. The CACD system will check if the AMC id matches the AMC id associated with the RSA data present in the CACD database.

(4) In order to test the UUPs management, the associated AUPs must be in status Released. To arrive to this status, all the AUPs of all the AMCs must have been promoted to status Ready before the CADF can promote all AUPs to status Released.

(5) The AUP/UUPs data are validated against the CACD data that are kept up to date by LUs.

(6) The test setup described below takes these requirements into account and tries to replace the human activities by scripts.

5.1.2. PRE-OPS platform setup

(1) On the Pre-Ops platform, there is no human intervention. We therefore need to automate the setup of the PRE-OPS test platform.

(2) To get accurate validation of AUP/UUPs, the CACD system that belongs to the test platform need data close to OPS. To do so, we will synchronize the PRE-OPS CACD system with the OPS CACD system.

(3) The AUPs/UUPs from the OPS CACD system will not be applied.

(4) The AUPs OPS will be imported every day at 18:00 for DAY+1. The AUP imported will keep their status RELEASED.

(5) Then the UUP time will be set to DAY+1 06:00.

(6) At DAY+1, the external users can test UUPs on the AUPs imported the day before and that until DAY 18:00 where the AUPs will be again importer from OPS.

(7) The external users can always test the create/delete/validate/update of AUPs for any days after day DAY. DAY+1 becomes inaccessible for AUPs testing after the import of the AUPs from OPS at 18:00.

(8) The only restriction is that external user cannot test UUPs on AUPs that they have created on the PRE-OPS test platform.

DOCUMENT FINAL PAGE

To properly report any fault, or to propose a modification concerning the present document, please refer to:

- for faults, the Systems Incident Management Procedure, ref. STD-CM/PRO/SIMP
- for changes, the IT Change Management Process, ref. STD/ITSM/CHG