NM 19.5.0 - NOP/B2B Reference Manuals - FlightServices

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

Document Control

Copyright Notice

Document Identification

Full Title:	NM 19.5.0 - NOP/B2B Reference Manuals - FlightServices
Total Number of Pages:	205

Table of Contents

References **13**

Terminology **14**

1 Introduction **15**
1.1 Identification 15

2 Context **16**

3 Port Types **17**
3.1 FlightPreparationService Port Type 17
3.1.1 Overview 17
3.1.2 Flight Plan Validation 17
3.1.2.1 SOAP 17
3.1.2.2 `FlightPlanValidationRequest` 17
3.1.2.3 `FlightPlanValidationReply` 18
3.1.3 Extended Flight Plan Validation 18
3.1.3.1 SOAP 18
3.1.3.2 `ExtendedFlightPlanValidationRequest` 19
3.1.3.3 `ExtendedFlightPlanValidationReply` 20
3.1.4 Routing Assistance 20
3.1.4.1 SOAP 20
3.1.4.2 `RoutingAssistanceRequest` 21
3.1.4.3 `RoutingAssistanceReply` 24
3.2 FlightFilingService Port Type 25
3.2.1 Overview 25
3.2.2 Flight Plan Creation 25
3.2.2.1 SOAP 25
3.2.2.2 `FlightPlanCreationRequest` 26
3.2.2.3 `FlightPlanCreationReply` 27
3.2.3 Flight Plan Update 28
3.2.3.1 SOAP 28
3.2.3.2 `FlightPlanUpdateRequest` 28
3.2.3.3 `FlightPlanUpdateReply` 29
3.2.4 Flight Plan Cancellation 30
3.2.4.1 SOAP 30
3.2.4.2 `FlightPlanCancellationRequest` 30
3.2.4.3 `FlightPlanCancellationReply` 31
3.2.5 Flight Delay 31
3.2.5.1 SOAP 31
3.2.5.2 `FlightDelayRequest` 32
3.2.5.3 `FlightDelayReply` 33
3.2.6 Flight Departure 33
3.2.6.1 SOAP 33
3.2.6.2 `FlightDepartureRequest` 34

3.2.6.3 FlightDepartureReply ... 35
3.2.7 Flight Arrival ... 35
3.2.7.1 SOAP ... 35
3.2.7.2 FlightArrivalRequest ... 36
3.2.7.3 FlightArrivalReply ... 37
3.2.8 Filing Status ... 37
3.2.8.1 SOAP ... 37
3.2.8.2 FilingStatusRequest ... 38
3.2.8.3 FilingStatusReply ... 38
3.2.9 Extended Flight Plan Creation ... 39
3.2.9.1 SOAP ... 39
3.2.9.2 ExtendedFlightPlanCreationRequest ... 40
3.2.9.3 ExtendedFlightPlanCreationReply ... 41
3.2.10 Extended Flight Plan Update ... 41
3.2.10.1 SOAP ... 41
3.2.10.2 ExtendedFlightPlanUpdateRequest ... 42
3.2.10.3 ExtendedFlightPlanUpdateReply ... 44
3.3 FlightManagementService Port Type ... 45
3.3.1 Overview ... 45
3.3.1.1 Operations ... 45
3.3.1.2 Forecast and Operational Datasets ... 46
3.3.1.3 Simulation Datasets ... 46
3.3.1.4 Proposal Flights ... 46
3.3.1.5 DPI - Departure Planning Information ... 47
3.3.1.5.1 IMPORTANT REMARK ... 47
3.3.1.5.2 Introduction ... 48
3.3.1.5.3 General Overview ... 49
3.3.1.5.4 Documents ... 50
3.3.2 Flight Plan List ... 50
3.3.2.1 SOAP ... 50
3.3.2.2 FlightPlanListRequest ... 50
3.3.2.3 FlightPlanListReply ... 54
3.3.3 Flight List by Keys ... 55
3.3.3.1 SOAP ... 55
3.3.3.2 FlightListByKeysRequest ... 55
3.3.3.3 FlightListByKeysReply ... 59
3.3.4 Flight List by Aircraft Operator ... 59
3.3.4.1 SOAP ... 59
3.3.4.2 FlightListByAircraftOperatorRequest ... 60
3.3.4.3 FlightListByAircraftOperatorReply ... 61
3.3.5 Flight List by Aerodrome ... 61
3.3.5.1 SOAP ... 61
3.3.5.2 FlightListByAerodromeRequest ... 62
3.3.5.3 FlightListByAerodromeReply ... 63
3.3.6 Flight List by Aerodrome Set ... 63
3.3.6.1 SOAP ... 63
3.3.6.2 FlightListByAerodromeSetRequest ... 64

3.3.6.3 FlightListByAerodromeSetReply 65
3.3.7 Flight List by Airspace 65
3.3.7.1 SOAP 65
3.3.7.2 FlightListByAirspaceRequest 66
3.3.7.3 FlightListByAirspaceReply 67
3.3.8 Flight List by Point 67
3.3.8.1 SOAP 67
3.3.8.2 FlightListByPointRequest 68
3.3.8.3 FlightListByPointReply 69
3.3.9 Flight List by Traffic Volume 69
3.3.9.1 SOAP 69
3.3.9.2 FlightListByTrafficVolumeRequest 70
3.3.9.3 FlightListByTrafficVolumeReply 71
3.3.10 Flight List by Measure 71
3.3.10.1 SOAP 71
3.3.10.2 FlightListByMeasureRequest 72
3.3.10.3 FlightListByMeasureReply 73
3.3.11 Flight List by Hotspot 73
3.3.11.1 SOAP 73
3.3.11.2 FlightListByHotspotRequest 74
3.3.11.3 FlightListByHotspotReply 75
3.3.12 Flight Retrieval 75
3.3.12.1 SOAP 75
3.3.12.2 FlightRetrievalRequest 76
3.3.12.3 FlightRetrievalReply 79
3.3.13 Early DPI 80
3.3.13.1 SOAP 80
3.3.13.2 EarlyDPIRequest 80
3.3.13.3 EarlyDPIReply 80
3.3.14 Target DPI Target 80
3.3.14.1 SOAP 80
3.3.14.2 TargetDPITargetRequest 81
3.3.14.3 TargetDPITargetReply 81
3.3.15 Target DPI Sequence 81
3.3.15.1 SOAP 81
3.3.15.2 TargetDPISequencedRequest 82
3.3.15.3 TargetDPISequencedReply 82
3.3.16 ATC DPI 82
3.3.16.1 SOAP 82
3.3.16.2 ATCDPIRequest 83
3.3.16.3 ATCDPIReply 83
3.3.17 Cancel DPI 83
3.3.17.1 SOAP 83
3.3.17.2 CancelDPIRequest 84
3.3.17.3 CancelDPIReply 84
3.4 FlightSafetyService Port Type 84
3.4.1 Overview 84

3.4.2 Flight Safety 85
3.4.2.1 SOAP 85
3.4.2.2 ACC3AccreditationListReplacementRequest 85
3.4.2.3 ACC3AccreditationListReplacementReply 86

4 Data Types 87
4.1 ACC3Accreditation 87
4.2 typedef<string> ACC3AccreditationId 87
4.3 Aerodrome 87
4.4 AerodromeDAL 87
4.5 AerodromeIdAndRole 88
4.6 typedef<string> AerodromeName_DataType 88
4.7 typedef<string> AerodromeNameLocationDescription_DataType 88
4.8 <<enumeration>> AerodromeRole 88
4.9 AerodromesOfDestination 89
4.10 typedef<string> AircraftIATAId 89
4.11 typedef<string> AircraftICAOId 89
4.12 AircraftIdentification 90
4.13 AircraftIdentificationUpdate 90
4.14 typedef<string> AircraftOperatorIATAId 91
4.15 typedef<string> AircraftOperatorIATAOrICAOId 91
4.16 typedef<string> AircraftOperatorICAOId 91
4.17 typedef<string> AircraftOperatorName_DataType 91
4.18 <<enumeration>> AircraftPerformanceCategory 91
4.19 typedef<string> AircraftRegistrationMark 92
4.20 AircraftType 92
4.21 typedef<string> AircraftTypeICAOId 92
4.22 AirFiledData 93
4.23 <<abstract>> AirportPlanningInformationRequest 94
4.24 AlternateAerodrome 95
4.25 typedef<string> AlternateAerodrome_DataType 95
4.26 <<enumeration>> ATFMMessageType 95
4.27 typedef<string> AtsUnitId_DataType 98
4.28 BasicTrajectoryData 98
4.29 CDM 99
4.30 CDMInfo 99
4.31 CDMProvisionalInfo 101
4.32 <<enumeration>> CDMStatus 102
4.33 <<enumeration>> CfmuFlightType 102
4.34 typedef<string> Colour_DataType 103
4.35 <<enumeration>> CTOTLimitReason 103
4.36 DatalinkCapabilities 104
4.37 typedef<string> DataLinkCapabilities_DataType 104
4.38 <<enumeration>> DeicingStatus 104
4.39 <<enumeration>> DelayCharacteristics 105
4.40 DeltaEntry 105

4.41 <<enumeration>> DepartureAirportType 106
4.42 DepartureData 106
4.43 <<abstract>> DeparturePlanningInformationRequest 107
4.44 <<enumeration>> DepartureStatus 107
4.45 DepartureTolerance 108
4.46 Dinghies 108
4.47 DistanceAtLocation 108
4.48 EnrouteDelay 109
4.49 <<enumeration>> EntryExit 109
4.50 EquipmentCapabilityAndStatus 109
4.51 <<enumeration>> EquipmentStatus 113
4.52 EstimatedElapsedTimeAtLocation 113
4.53 <<enumeration>> EURSTSIndicator 114
4.54 ExclusionFromRegulations 115
4.55 typedef<string> ExtendedAircraftICAOId 115
4.56 ExtendedFlightPlan 116
4.57 ExtendedFlightPlanUpdate 117
4.58 <<enumeration>> FAMStatus 118
4.59 typedef<string> FilingId 119
4.60 <<abstract>> FilingReplyData 119
4.61 <<abstract>> FilingRequest 120
4.62 FilingResultQueued 121
4.63 FilingResultRejected 121
4.64 FilingResultValid 121
4.65 <<enumeration>> FilingRule 122
4.66 <<enumeration>> FilingStatus 122
4.67 Flight 122
4.68 FlightAirspace 130
4.69 <<enumeration>> FlightDataset 131
4.70 FlightEvent 131
4.71 <<enumeration>> FlightEventType 132
4.72 <<enumeration>> FlightField 135
4.73 FlightIATAId 139
4.74 FlightIdentificationInput 139
4.75 FlightIdentificationOutput 140
4.76 FlightKeys 140
4.77 <<abstract>> FlightListByLocationReplyData 142
4.78 <<abstract>> FlightListByLocationRequest 143
4.79 <<enumeration>> FlightListByMeasureMode 144
4.80 <<abstract>> FlightListReplyData 145
4.81 <<abstract>> FlightListRequest 146
4.82 FlightOperationalLogEntry 151
4.83 <<enumeration>> FlightOperationalLogEntryType 151
4.84 FlightOrFlightPlan 152
4.85 FlightPerformanceData 153

4.86 FlightPlan 154
4.87 FlightPlanHistory 159
4.88 FlightPlanHistoryInfo 159
4.89 FlightPlanInput 161
4.90 <<enumeration>> FlightPlanMessageStatus 161
4.91 <<enumeration>> FlightPlanMessageType 162
4.92 FlightPlanOrInvalidFiling 162
4.93 FlightPlanUnion 163
4.94 FlightPlanUpdate 163
4.95 <<enumeration>> FlightPlanValidationRequestedField 164
4.96 FlightPoint 165
4.97 FlightPointRoute 166
4.98 FlightRestriction 166
4.99 <<enumeration>> FlightRules 167
4.100 <<enumeration>> FlightState 167
4.101 <<enumeration>> FlightTrend 168
4.102 <<enumeration>> FlightType 168
4.103 FourDimensionalTrajectory 169
4.104 FourDPoint 171
4.105 FourDTrajectoryPoint 171
4.106 typedef<string> FreezePoint 175
4.107 <<enumeration>> FrequencyOnAircraft 175
4.108 typedef<string> ICAOAircraftAddress 175
4.109 <<enumeration>> ICAOSTSIndicator 175
4.110 <<enumeration>> IFPIndicator 176
4.111 typedef<string> IFPLId 177
4.112 IFPSError 177
4.113 InvalidFiling 178
4.114 typedef<double> Latitude 178
4.115 <<enumeration>> LifeJacketEquipment 178
4.116 LoadStateAtReferenceLocation 179
4.117 typedef<double> Longitude 179
4.118 MessageOriginator 179
4.119 ModeSCapabilities 179
4.120 typedef<int> NumberOfDinghies_DataType 180
4.121 OtherAerodromeDesignation 180
4.122 typedef<string> OtherAircraftTypeDesignation_DataType 181
4.123 OtherInformation 181
4.124 <<enumeration>> PerformanceBasedNavigationCode 183
4.125 PointDAL 184
4.126 Position 184
4.127 ProfileTuningRestriction 185
4.128 ProfileValidity 185
4.129 <<enumeration>> ProfileValidityKind 185
4.130 ReadyStatus 186

4.131 <<enumeration>> ReasonForDPICancellation 186
4.132 ReclearanceInFlight 187
4.133 Relative4DPoint 187
4.134 RequestedFlightLevel 188
4.135 <<enumeration>> ReroutableStatus 188
4.136 ReroutingIndicator 188
4.137 <<enumeration>> ReroutingReason 189
4.138 ReroutingReference 189
4.139 <<enumeration>> ReroutingState 189
4.140 RevisionTimes 190
4.141 RouteInfo 190
4.142 RoutingAssistanceReplyDataResult 191
4.143 <<enumeration>> SegmentType 191
4.144 SlotImprovementProposal 192
4.145 SpecialHandlingIndicators 192
4.146 typedef<string> SSRCode 192
4.147 SSRInfo 192
4.148 <<enumeration>> SSRMode 193
4.149 typedef<string> StayInformation_DataType 193
4.150 SupplementaryInformation 193
4.151 SurveillanceEquipment 194
4.152 <<enumeration>> SurvivalEquipment 196
4.153 <<enumeration>> SuspensionStatus 196
4.154 TargetDPIRequest 197
4.155 typedef<string> TerminalName_DataType 197
4.156 TimeAndModel 197
4.157 typedef<int> TotalCapacity_DataType 198
4.158 <<enumeration>> TrafficType 198
4.159 TrajectoryPointRole 198
4.160 <<enumeration>> TrajectoryPointType 199
4.161 TrajectorySegment 200
4.162 <<abstract>> UpdateDPIRequest 201
4.163 <<enumeration>> WakeTurbulenceCategory 202
4.164 WindVector 203

5 PRE-OPS Testing 204

DOCUMENT FINAL PAGE 205

List of Figures

3.1 FlightPlanValidationRequest Class Diagram 17
3.2 FlightPlanValidationReply Class Diagram 18
3.3 ExtendedFlightPlanValidationRequest Class Diagram 19
3.4 ExtendedFlightPlanValidationReply Class Diagram 20
3.5 RoutingAssistanceRequest Class Diagram 21
3.6 RoutingAssistanceReply Class Diagram 24
3.7 FlightPlanCreationRequest Class Diagram 26
3.8 FlightPlanCreationReply Class Diagram 27
3.9 FlightPlanUpdateRequest Class Diagram 28
3.10 FlightPlanUpdateReply Class Diagram 29
3.11 FlightPlanCancellationRequest Class Diagram 30
3.12 FlightPlanCancellationReply Class Diagram 31
3.13 FlightDelayRequest Class Diagram 32
3.14 FlightDelayReply Class Diagram 33
3.15 FlightDepartureRequest Class Diagram 34
3.16 FlightDepartureReply Class Diagram 35
3.17 FlightArrivalRequest Class Diagram 36
3.18 FlightArrivalReply Class Diagram 37
3.19 FilingStatusRequest Class Diagram 38
3.20 FilingStatusReply Class Diagram 38
3.21 ExtendedFlightPlanCreationRequest Class Diagram 40
3.22 ExtendedFlightPlanCreationReply Class Diagram 41
3.23 ExtendedFlightPlanUpdateRequest Class Diagram 42
3.24 ExtendedFlightPlanUpdateReply Class Diagram 44
3.25 Departure Planning Information - General Overview 49
3.26 FlightPlanListRequest Class Diagram 50
3.27 FlightPlanListReply Class Diagram 54
3.28 FlightListByKeysRequest Class Diagram 55
3.29 FlightListByKeysReply Class Diagram 59
3.30 FlightListByAircraftOperatorRequest Class Diagram 60
3.31 FlightListByAircraftOperatorReply Class Diagram 61
3.32 FlightListByAerodromeRequest Class Diagram 62
3.33 FlightListByAerodromeReply Class Diagram 63
3.34 FlightListByAerodromeSetRequest Class Diagram 64
3.35 FlightListByAerodromeSetReply Class Diagram 65
3.36 FlightListByAirspaceRequest Class Diagram 66
3.37 FlightListByAirspaceReply Class Diagram 67
3.38 FlightListByPointRequest Class Diagram 68
3.39 FlightListByPointReply Class Diagram 69
3.40 FlightListByTrafficVolumeRequest Class Diagram 70
3.41 FlightListByTrafficVolumeReply Class Diagram 71
3.42 FlightListByMeasureRequest Class Diagram 72
3.43 FlightListByMeasureReply Class Diagram 73

3.44 FlightListByHotspotRequest Class Diagram 74
3.45 FlightListByHotspotReply Class Diagram 75
3.46 FlightRetrievalRequest Class Diagram 76
3.47 FlightRetrievalReply Class Diagram 79
3.48 EarlyDPIRequest Class Diagram 80
3.49 TargetDPITargetRequest Class Diagram 81
3.50 TargetDPISequencedRequest Class Diagram 82
3.51 ATCDPIRequest Class Diagram 83
3.52 CancelDPIRequest Class Diagram 84
3.53 ACC3AccreditationListReplacementRequest Class Diagram 85
3.54 ACC3AccreditationListReplacementReply Class Diagram 86
4.1 <<abstract>> AirportPlanningInformationRequest Class Diagram 94
4.2 <<abstract>> DeparturePlanningInformationRequest Class Diagram 107
4.3 ExtendedFlightPlan Class Diagram 116
4.4 ExtendedFlightPlanUpdate Class Diagram 117
4.5 <<abstract>> FilingReplyData Class Diagram 119
4.6 <<abstract>> FilingRequest Class Diagram 120
4.7 <<abstract>> FlightListByLocationReplyData Class Diagram 142
4.8 <<abstract>> FlightListByLocationRequest Class Diagram 143
4.9 <<abstract>> FlightListReplyData Class Diagram 145
4.10 <<abstract>> FlightListRequest Class Diagram 146
4.11 FlightPerformanceData Class Diagram 153
4.12 FlightPlan Class Diagram 154
4.13 FourDimensionalTrajectory Class Diagram 169
4.14 TargetDPIRequest Class Diagram 197
4.15 <<abstract>> UpdateDPIRequest Class Diagram 201
4.16 Wind Vector 203

List of Tables

4.1 <<enumeration>> AircraftPerformanceCategory 92
4.2 <<enumeration>> CfmuFlightType 102
4.3 <<enumeration>> DelayCharacteristics 105
4.4 <<enumeration>> DepartureAirportType 106
4.5 <<enumeration>> EURSTSIndicator 114
4.6 <<enumeration>> FlightDataset 131
4.7 <<enumeration>> FlightEventType 132
4.8 <<enumeration>> FlightField 136
4.9 FlightPlanHistoryInfoDetailed.checkPoints description 159
4.10 FlightPlanHistoryInfoDetailed.mode description 160
4.11 <<enumeration>> FlightPlanMessageStatus 161
4.12 <<enumeration>> FlightPlanMessageType 162
4.13 <<enumeration>> FlightRules 167
4.14 <<enumeration>> FlightType 168
4.15 <<enumeration>> FrequencyOnAircraft 175
4.16 <<enumeration>> ICAOSTSIndicator 176
4.17 <<enumeration>> IFPIndicator 176
4.18 <<enumeration>> LifeJacketEquipment 178
4.19 ModeSCapabilities.enhancedSurveillance description 180
4.20 <<enumeration>> PerformanceBasedNavigationCode 183
4.21 <<enumeration>> ReroutingReason 189
4.22 <<enumeration>> ReroutingState 190
4.23 <<enumeration>> SegmentType 191
4.24 <<enumeration>> TrafficType 198
4.25 <<enumeration>> WakeTurbulenceCategory 202

References

External

This document makes reference to the following external documents (an external document being defined as a document not produced by NM):

[1] Flight Object Interface Control Description for ICOG IOP Interface Specification

NM

[2] NOP/B2B Reference Manuals - CommonServices

[3] NOP/B2B Reference Manuals - AirspaceServices

[4] NOP/B2B Reference Manuals - FlowServices

[5] ANNEX to the IFPS USERS MANUAL; Generated Errors. IFPS USERS MANUAL

[6] DPI Implementation Guide. Edition 1.800 (March 2015)

[7] Flight Progress Messages. Edition 2.100 (March 2015)

[8] DPI and FUM Implementation Road Map. Edition 1.600 (March 2015)

[9] Site of European Airport CDM: http://www.euro-cdm.org

Terminology

Main Abbreviations and Acronyms

EFPM (Extended Flight Plan Message.)
: When referring to IFPS error class EFPM, NM means errors that are related to the semantic of the message (see [5]).

PROF (Error class Profile)
: When referring to IFPS error class PROF, NM means errors that are related to the four dimensional profile. They are typically errors on RAD and CDR availability (see [5]).

RA (Error class Routing Assistance)
: When referring to IFPS error class RA, NM means errors related to Routing Assistance. They are errors on the constraint that can be specified when requesting the generation of alternate route (e.g.: error on via point being unknown) (see [5]).

ROUTE (Error class ROUTE)
: When referring to IFPS error class ROUTE, NM means errors that are related to the semantic of the field 15 and to the two dimensional route description (see [5]).

SYN (Syntax Error)
: When referring to IFPS error class ROUTE, NM means errors that are related to the syntax of the message (see [5]).

ICAO2012 (ICAO 2012)
: Refers to the new ICAO flight plan content defined for implementation on the 15/11/2012.

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/Flight.

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

Chapter 2. Context

(1) NM NOP/B2B provides three flight services corresponding to the three stages of a flight:

a) Flight preparation: services used during the preparation of a flight plan, before filing it to NM

b) Flight filing: services related to the flight plan filing activity, including creation, update and cancellation

c) Flight management: services used to query and retrieve information on existing flight plans and flights

Chapter 3. Port Types

3.1. FlightPreparationService Port Type

3.1.1. Overview

(1) FlightPreparationService is intended to provide requests aimed at easing the preparation phase of the flight plan (prior to its filing to NM):

a) FlightPlanValidationRequest / FlightPlanValidationReply

b) ExtendedFlightPlanValidationRequest / ExtendedFlightPlanValidationReply

c) RoutingAssistanceRequest / RoutingAssistanceReply

3.1.2. Flight Plan Validation

3.1.2.1. SOAP

(1) The associated SOAP operation is:

```
FlightPlanValidationReply validateFlightPlan(
    FlightPlanValidationRequest request
    )
```

3.1.2.2. `FlightPlanValidationRequest`

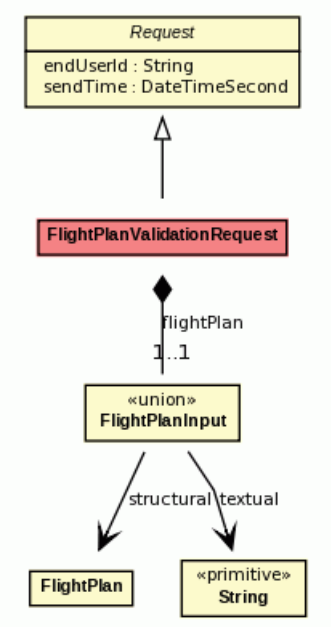

Figure 3.1. `FlightPlanValidationRequest` Class Diagram

(1) Request to query the validation of an FPL according to the NM/IFPS validation rules.

(2) The request provides the input flight plan information via a choice: either in string format or via a FlightPlan structure.

(3) Inherits from: `Request`

(4) Attributes:

a) **`FlightPlanInput flightPlan`** *(Mandatory)*
Flight plan to be validated.

3.1.2.3. `FlightPlanValidationReply`

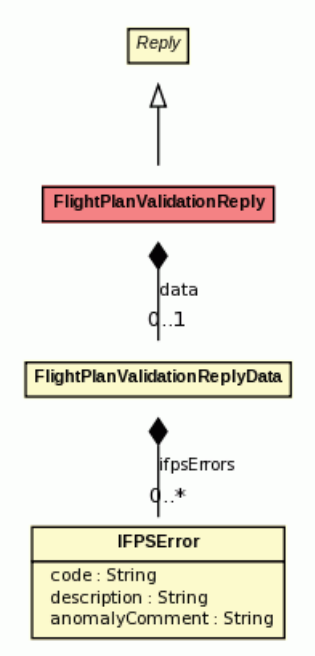

Figure 3.2. `FlightPlanValidationReply` Class Diagram

(1) Reply returned in response to FlightPlanValidationRequest.

(2) Inherits from: `Reply`

(3) Attributes:

a) **`IFPSError[] ifpsErrors`** *(Mandatory)*
Array of NM/IFPS errors in response to the flight plan validation.
Mandatory: the array is empty if there is no such error.
Constraint: Size must be comprised between 0 and ∞.

3.1.3. Extended Flight Plan Validation

3.1.3.1. SOAP

(1) The associated SOAP operation is:

```
ExtendedFlightPlanValidationReply validateExtendedFlightPlan(
    ExtendedFlightPlanValidationRequest request
    )
```

3.1.3.2. `ExtendedFlightPlanValidationRequest`

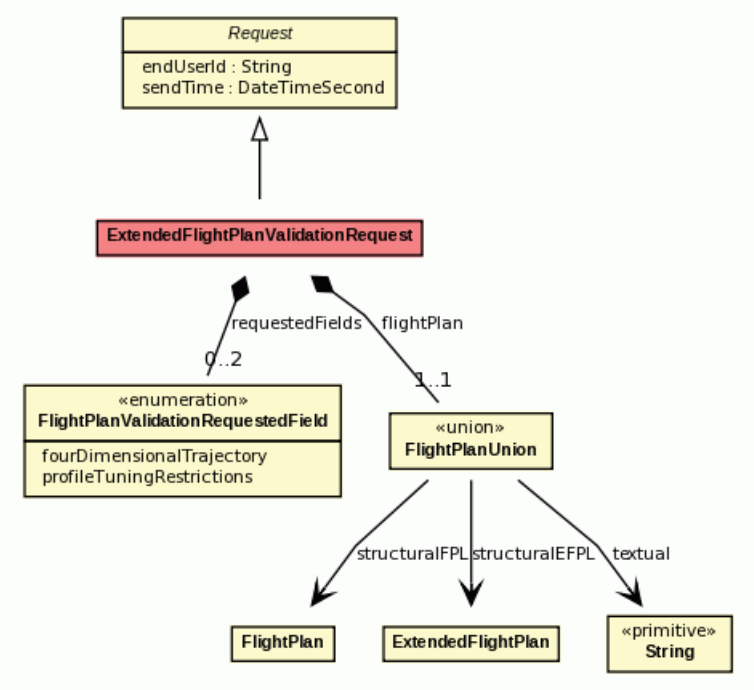

Figure 3.3. `ExtendedFlightPlanValidationRequest` Class Diagram

(1) Request to query the validation of an FPL according to the NM/IFPS validation rules.

(2) The request provides the input flight plan information via a choice: either in string format or via a FlightPlan or ExtendedFlightPlan structure.

(3) Inherits from: `Request`

(4) Attributes:

a) **`FlightPlanUnion flightPlan`** *(Mandatory)*
Flight plan to be validated.

b) **`FlightPlanValidationRequestedField[] requestedFields`** *(Optional)*
Additional information to be returned in the reply.
If the array is empty then no additional information will be returned.
Constraint: Size must be comprised between 0 and 2.

3.1.3.3. `ExtendedFlightPlanValidationReply`

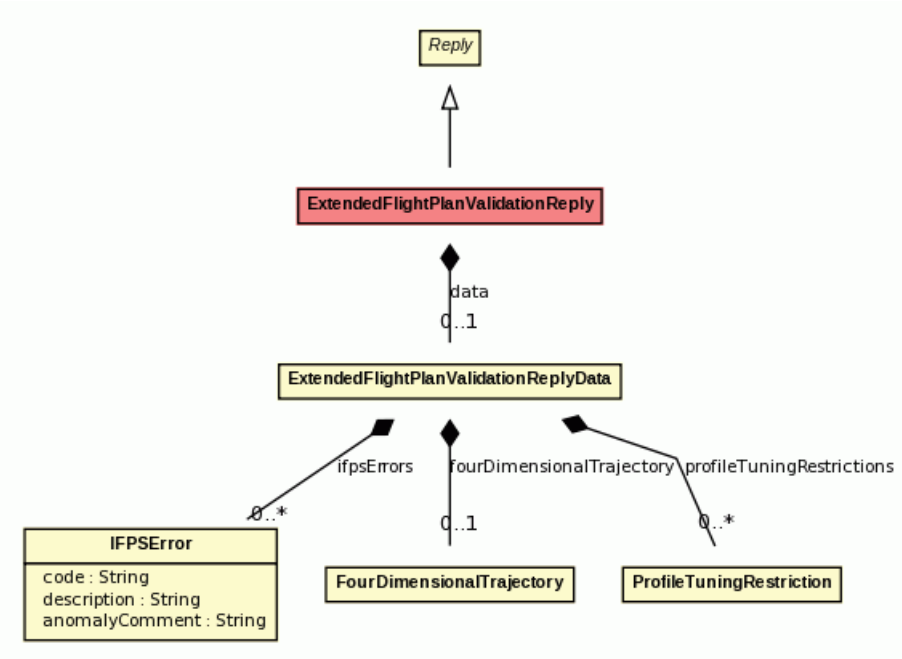

Figure 3.4. `ExtendedFlightPlanValidationReply` Class Diagram

(1) Reply returned in response to ExtendedFlightPlanValidationRequest.

(2) Inherits from: `Reply`

(3) Attributes:

a) **`IFPSError[] ifpsErrors`** *(Mandatory)*
Array of NM/IFPS errors in response to the flight plan validation.
Mandatory: the array is empty if there is no such error.
Constraint: Size must be comprised between 0 and ∞.

b) **`FourDimensionalTrajectory fourDimensionalTrajectory`** *(Optional)*
Four dimensional trajectory.

c) **`ProfileTuningRestriction[] profileTuningRestrictions`** *(Optional)*
List of profile tuning restrictions.
Constraint: Size must be comprised between 0 and ∞.

3.1.4. Routing Assistance

3.1.4.1. SOAP

(1) The associated SOAP operation is:

```
RoutingAssistanceReply proposeRoutes(
    RoutingAssistanceRequest request
    )
```

3.1.4.2. `RoutingAssistanceRequest`

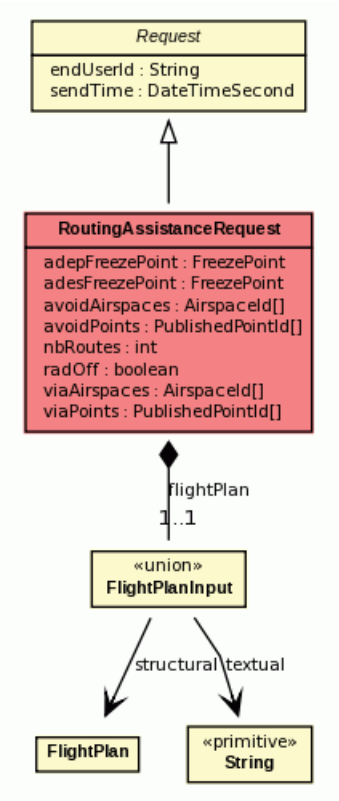

Figure 3.5. `RoutingAssistanceRequest` Class Diagram

(1) Request to query the generation of NM/IFPS-compliant routes for a given flight plan.

(2) The request provides the input flight plan information via a choice: either in string format or via a FlightPlan structure (FlightPlanInput`flightPlan` attribute).

(3) The request provides also a set of constraints apply to the route generation algorithm. These constraints are all applied: they are interpreted as combined via the AND logical operator.

(4) Inherits from: `Request`

(5) Attributes:

a) **`FreezePoint` `adepFreezePoint`** *(Optional)*
Field15 information must be provided via the `flightPlan` attribute. The referenced point is a point of the given field 15 from which the route generation will start.
The part of the route from the ADEP to that point is "frozen". The resulting routes will all start with this "frozen" part of the field15.

b) **FreezePoint adesFreezePoint** *(Optional)*
Field15 information must be provided via the `flightPlan` attribute. The referenced point is a point of the given field 15 where the route generation will stop. The part of the route from that point to the ADES is "frozen". The resulting routes will all end with this "frozen" part of the field15.

c) **AirspaceId[] avoidAirspaces** *(Optional)*
NM airspace identifiers. The 2D projection of the proposed alternative routes will avoid the 2D projection of all given airspaces.
NM rejects generation requests requiring more than 10 "via" and/or "avoid" constraints.
Constraints:

i) Size must be comprised between 0 and ∞.

ii) See TOO_MANY_VIA_AND_AVOID_CONSTRAINTS

d) **PublishedPointId[] avoidPoints** *(Optional)*
ICAO published point identifiers. The proposed alternative routes will avoid all given points.
Constraints:

i) Size must be comprised between 0 and ∞.

ii) See TOO_MANY_VIA_AND_AVOID_CONSTRAINTS

e) **FlightPlanInput flightPlan** *(Mandatory)*
Flight plan.

f) **int nbRoutes** *(Optional)*
Maximum number of routes to be generated.
Maximum value is 10. If greater than 10, the request is accepted and the service is realised with a value of 10.
Optional. When not set, defaults to 5.
Constraint: Range: `]- ∞,∞[`.

g) **boolean radOff** *(Optional)*
When set to true, the errors generated by the RAD validation are ignored. The proposed alternative routes may not be RAD-compliant. If this is the case, the corresponding errors will be given together with the route.
When omitted, defaults to false.

h) **AirspaceId[] viaAirspaces** *(Optional)*
NM airspace identifiers. The 2D projection of the proposed alternative routes will traverse the 2D projection of all given airspaces.
Constraints:

i) Size must be comprised between 0 and ∞.

ii) See TOO_MANY_VIA_AND_AVOID_CONSTRAINTS

i) **PublishedPointId[] viaPoints** *(Optional)*

ICAO published point identifiers. The proposed alternative routes will pass over all given points.
Constraints:

i) Size must be comprised between 0 and ∞.

ii) See TOO_MANY_VIA_AND_AVOID_CONSTRAINTS

(6) Constraint:

a)

Name	TOO_MANY_VIA_AND_AVOID_CONSTRAINTS
Attributes	avoidPoints, avoidAirspaces, viaPoints, viaAirspaces
Description	NM rejects generation requests requiring more than 10 "via" and/or "avoid" constraints.

3.1.4.3. `RoutingAssistanceReply`

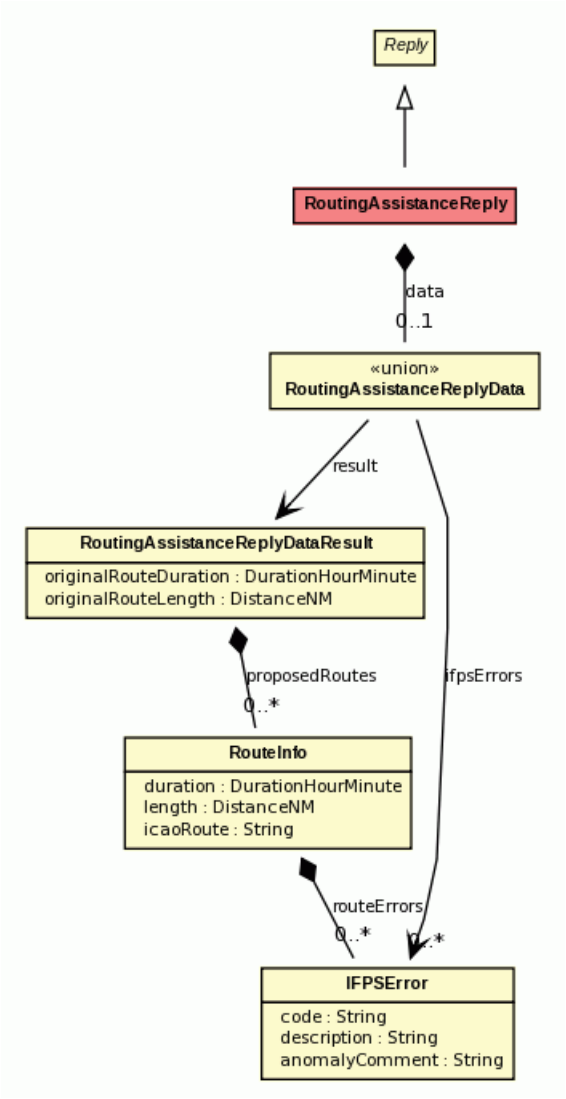

Figure 3.6. `RoutingAssistanceReply` Class Diagram

(1) Reply returned in response to RoutingAssistanceRequest.

(2) The proposed routes (if any) are expressed as an array of RouteInfo structures.

(3) Inherits from: `Reply`

(4) Choices:

a) **`RoutingAssistanceReplyDataResult result`**
Contains the generated alternate routes if no NM/IFPS errors occurred.

b) **`IFPSError[] ifpsErrors`**

Array of NM/IFPS errors in response to the routing assistance request. They are errors on the constraint that can be specified when requesting the generation of alternate route (e.g.: error on via point being unknown)
If not null, the array cannot be empty. The array cannot contain null or duplicate items.
Constraint: Size must be comprised between 0 and ∞.

3.2. FlightFilingService Port Type

3.2.1. Overview

(1) FlightFilingService is intended to provide requests aimed at filing flight plan messages:

a) ExtendedFlightPlanCreationRequest / ExtendedFlightPlanCreationReply

b) ExtendedFlightPlanUpdateRequest / ExtendedFlightPlanUpdateReply

c) FlightPlanCreationRequest / FlightPlanCreationReply

d) FlightPlanUpdateRequest / FlightPlanUpdateReply

e) FlightPlanCancellationRequest / FlightPlanCancellationReply

f) FlightDelayRequest / FlightDelayReply

g) FlightDepartureRequest / FlightDepartureReply

h) FlightArrivalRequest / FlightArrivalReply

i) FilingStatusRequest / FilingStatusReply

3.2.2. Flight Plan Creation

3.2.2.1. SOAP

(1) The associated SOAP operation is:

```
FlightPlanCreationReply fileNewFlightPlan(
    FlightPlanCreationRequest request
    )
```

3.2.2.2. `FlightPlanCreationRequest`

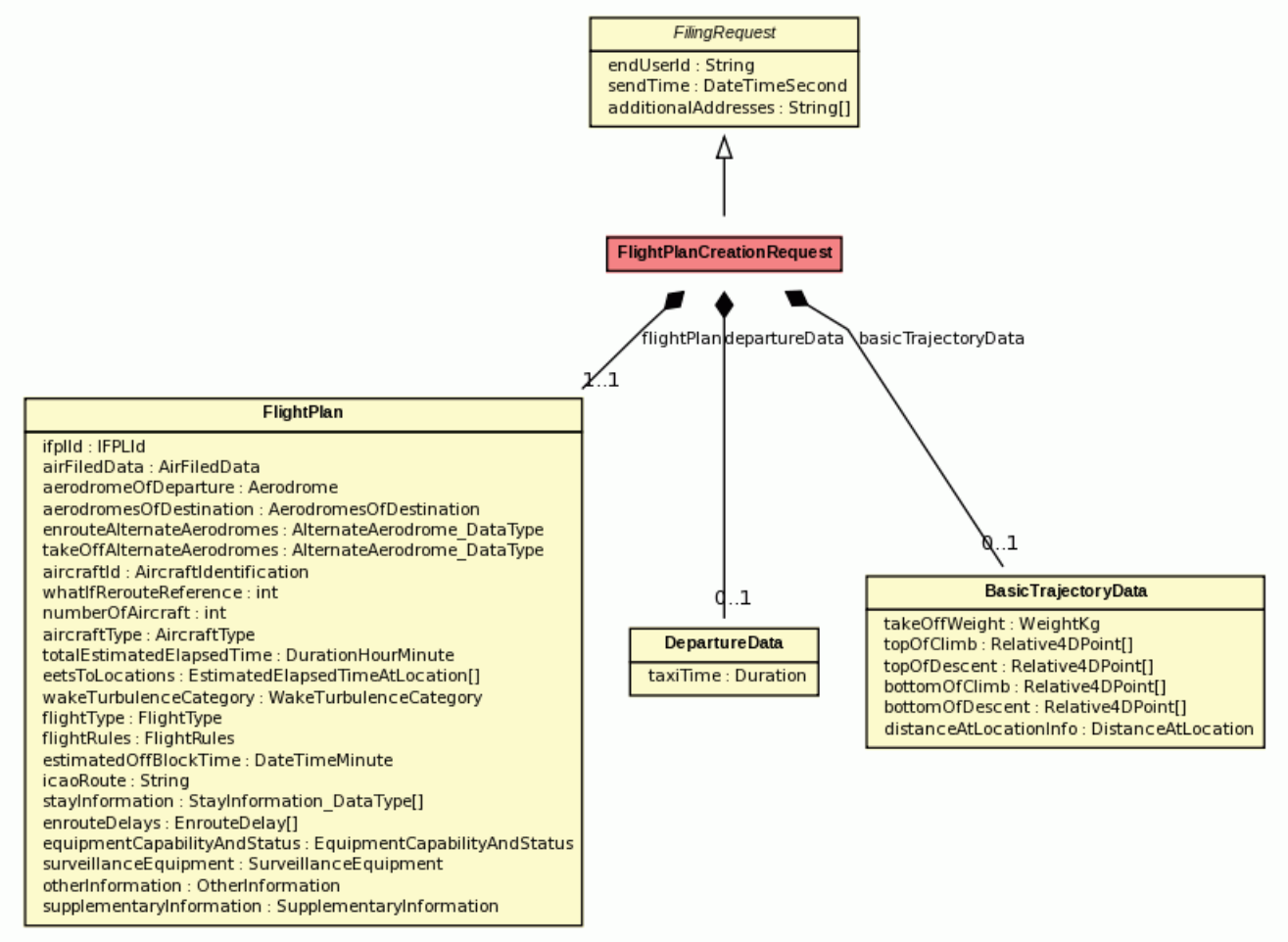

Figure 3.7. `FlightPlanCreationRequest` Class Diagram

(1) Request the submission (or filing) of a new flight plan to the NM.

(2) The same new flight plan cannot be created more than once: if NM detects that the submitted new flight plan has the same flight keys as an existing flight plan, the request is rejected with an error.

(3) Inherits from: `FilingRequest`

(4) Attributes:

a) **`FlightPlan` `flightPlan`** *(Mandatory)*
The submitted flight plan.

b) **`BasicTrajectoryData` `basicTrajectoryData`** *(Optional)*
Basic four dimensional data used by NM to calculate the trajectory.

c) **DepartureData departureData** *(Optional)*
Flight plan information related to the departure, before the aircraft takes-off.

3.2.2.3. FlightPlanCreationReply

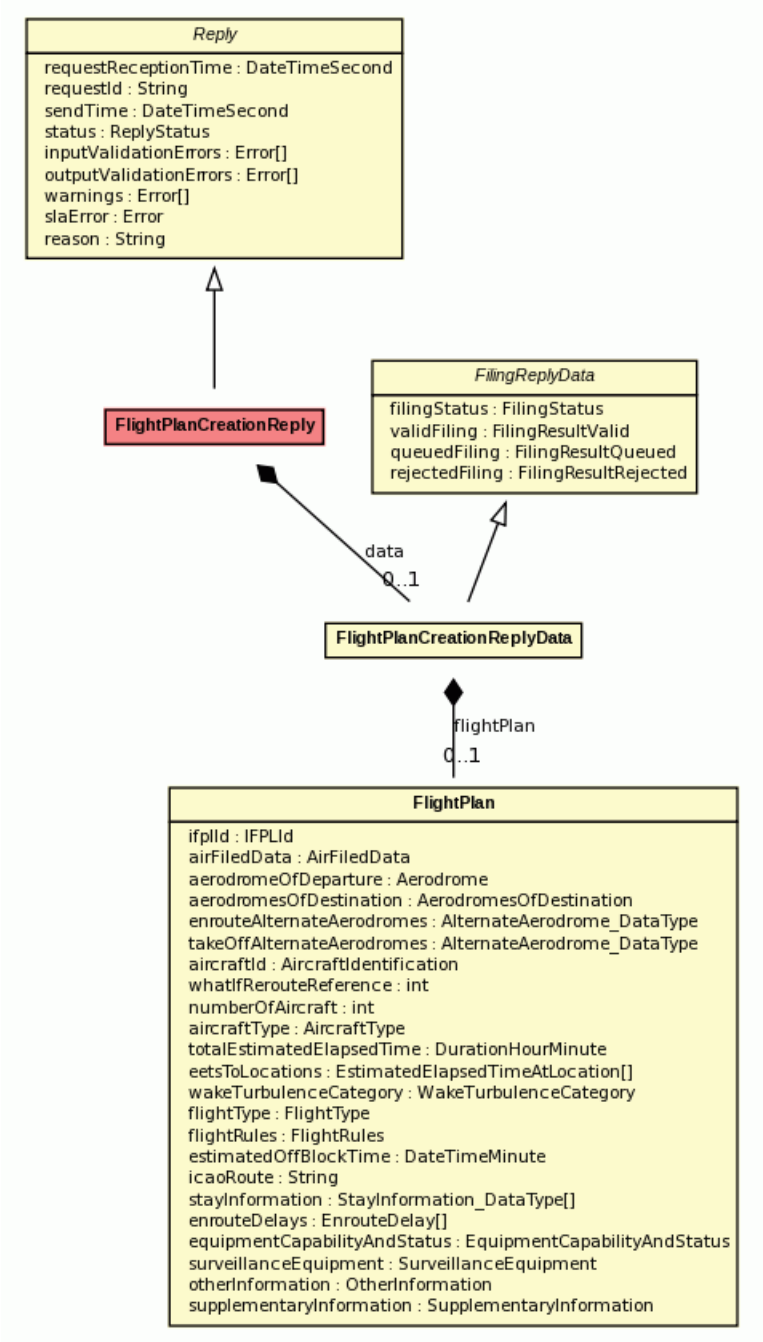

Figure 3.8. `FlightPlanCreationReply` Class Diagram

(1) Reply returned in response to FlightPlanCreationRequest.

(2) Inherits from: `Reply`

(3) Attributes:

a) **`FlightPlan flightPlan`** *(Optional)*
The flight plan as accepted by NM, may have been automatically or manually corrected. Cannot be null if `FilingReply.filingStatus` is `VALID`; must be null otherwise.

3.2.3. Flight Plan Update

3.2.3.1. SOAP

(1) The associated SOAP operation is:

```
FlightPlanUpdateReply fileFlightPlanUpdate(
    FlightPlanUpdateRequest request
    )
```

3.2.3.2. `FlightPlanUpdateRequest`

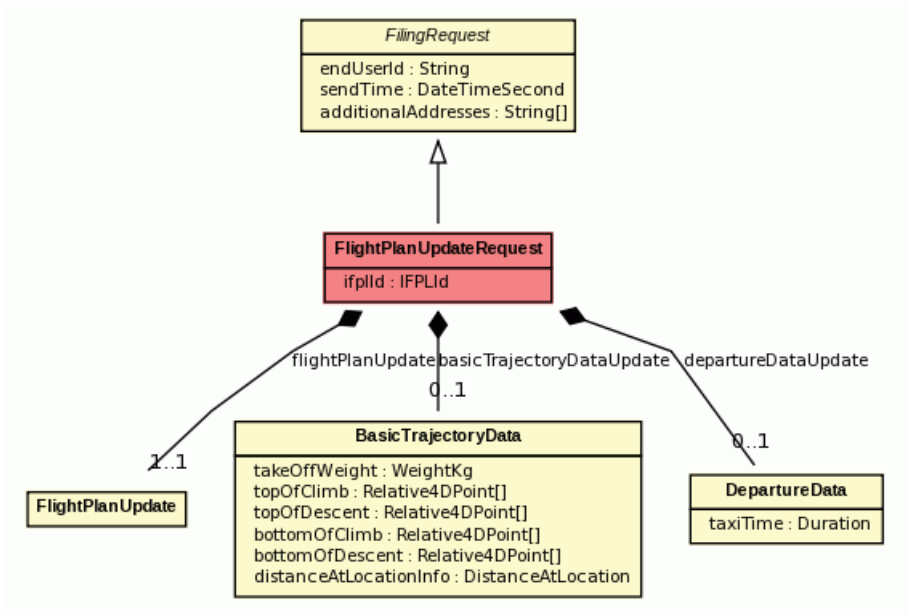

Figure 3.9. `FlightPlanUpdateRequest` Class Diagram

(1) Request for the update of a flight plan.

(2) The `FlightPlanUpdateRequest` supports the update of some selective fields which are updated in the existing flight plan while others are left unchanged.

(3) Inherits from: `FilingRequest`

(4) Attributes:

a) **IFPLId ifplId** *(Mandatory)*
Identification of the flight plan to be updated.

b) **FlightPlanUpdate flightPlanUpdate** *(Mandatory)*
Update of selective fields in the existing flight plan.

c) **BasicTrajectoryData basicTrajectoryDataUpdate** *(Optional)*
Basic four dimensional data used by NM to calculate the trajectory.

d) **DepartureData departureDataUpdate** *(Optional)*
Flight plan information related to the departure, before the aircraft takes-off.

3.2.3.3. FlightPlanUpdateReply

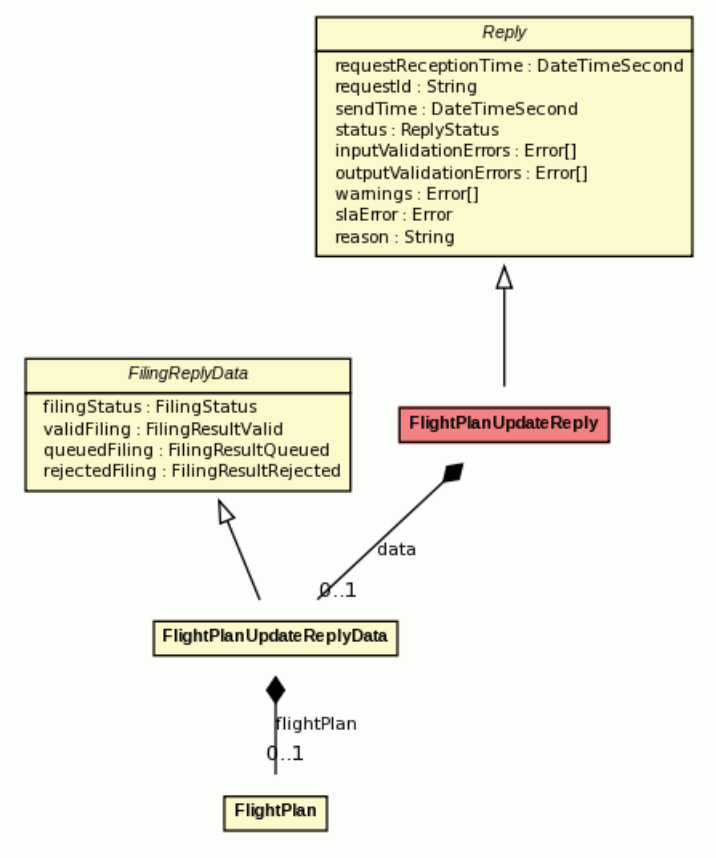

Figure 3.10. FlightPlanUpdateReply Class Diagram

(1) Reply returned in response to FlightPlanUpdateRequest.

(2) If the given flight identification is unknown or ambiguous, the error is reported as a generic input validation error (object not found), as described in NM References.

(3) Inherits from: Reply

(4) Attributes:

a) **`FlightPlan flightPlan`** *(Optional)*
The flight plan as updated by NM, may have been automatically or manually corrected. Cannot be null if `FilingReply.filingStatus` is `VALID`, meaning that the flight plan was indeed updated; must be null otherwise.

3.2.4. Flight Plan Cancellation

3.2.4.1. SOAP

(1) The associated SOAP operation is:

```
FlightPlanCancellationReply fileFlightPlanCancellation(
    FlightPlanCancellationRequest request
    )
```

3.2.4.2. `FlightPlanCancellationRequest`

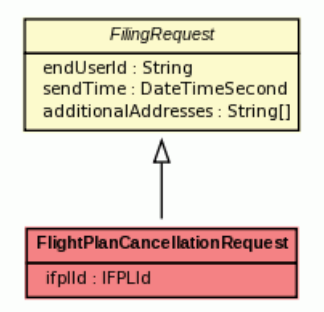

Figure 3.11. `FlightPlanCancellationRequest` Class Diagram

(1) Request for the cancellation of a flight plan.

(2) Inherits from: `FilingRequest`

(3) Attributes:

a) **`IFPLId ifplId`** *(Mandatory)*
Identification of the flight plan to be cancelled.

3.2.4.3. `FlightPlanCancellationReply`

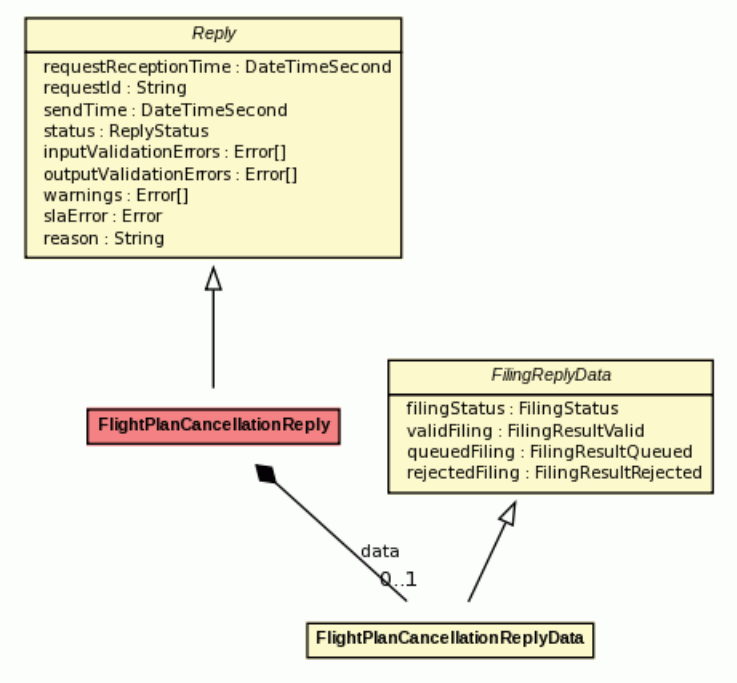

Figure 3.12. `FlightPlanCancellationReply` Class Diagram

(1) Reply returned in response to FlightPlanCancellationRequest.

(2) If the given flight identification is unknown or ambiguous, the error is reported as a generic input validation error (object not found), as described in the manual CommonServices .

(3) Inherits from: `Reply`

3.2.5. Flight Delay

3.2.5.1. SOAP

(1) The associated SOAP operation is:

```
FlightDelayReply fileFlightDelay(
    FlightDelayRequest request
    )
```

3.2.5.2. `FlightDelayRequest`

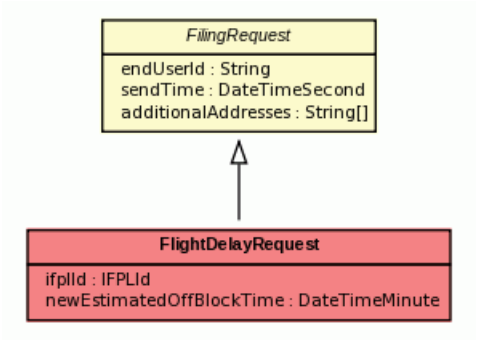

Figure 3.13. `FlightDelayRequest` Class Diagram

(1) Request to notify the delay of a flight.

(2) Inherits from: `FilingRequest`

(3) Attributes:

a) **`IFPLId` `ifplId`** *(Mandatory)*
Identification of the flight plan to be updated.

b) **`DateTimeMinute` `newEstimatedOffBlockTime`** *(Mandatory)*
New estimated off-block date/time.

3.2.5.3. `FlightDelayReply`

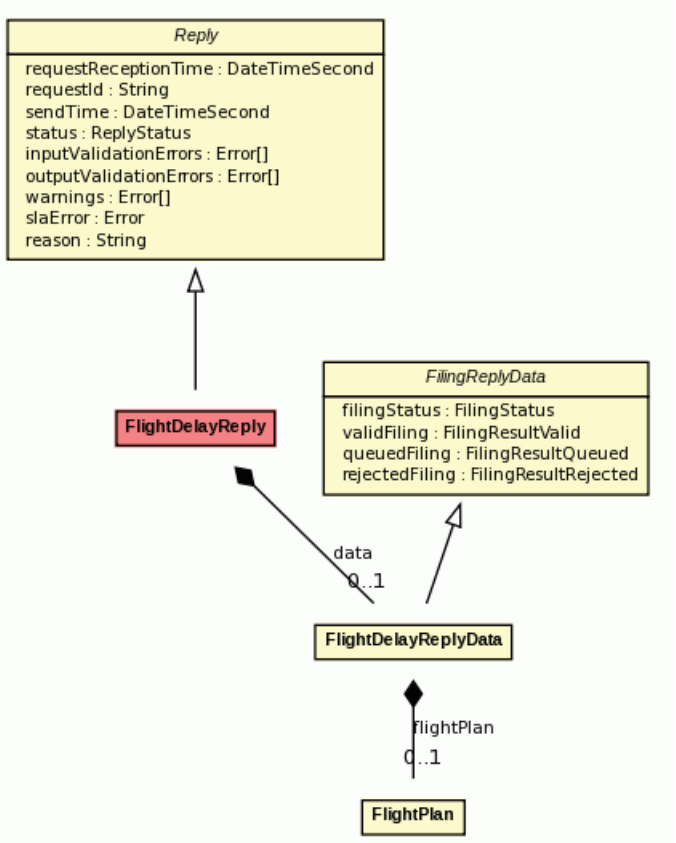

Figure 3.14. `FlightDelayReply` Class Diagram

(1) Reply returned in response to FlightDelayRequest.

(2) If the given flight identification is unknown or ambiguous, the error is reported as a generic input validation error (object not found), as described in the manual CommonServices .

(3) Inherits from: `Reply`

(4) Attributes:

a) **`FlightPlan flightPlan`** *(Optional)*
The updated flight plan.

3.2.6. Flight Departure

3.2.6.1. SOAP

(1) The associated SOAP operation is:

```
FlightDepartureReply fileFlightDeparture(
    FlightDepartureRequest request
    )
```

3.2.6.2. `FlightDepartureRequest`

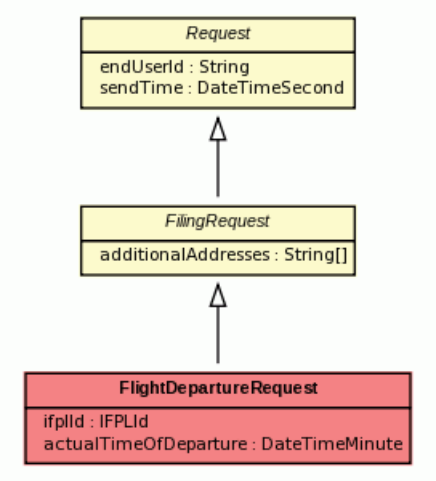

Figure 3.15. `FlightDepartureRequest` Class Diagram

(1) Request to notify the departure of a flight.

(2) Inherits from: `FilingRequest`

(3) Attributes:

a) **`IFPLId` `ifplId`** *(Mandatory)*
Identification of the flight to be updated.

b) **`DateTimeMinute` `actualTimeOfDeparture`** *(Mandatory)*
Actual date/time of departure.

3.2.6.3. `FlightDepartureReply`

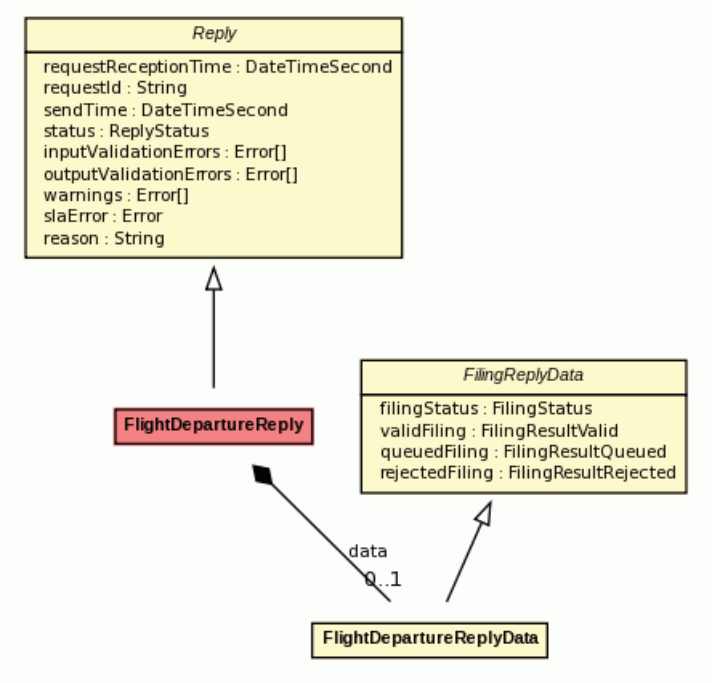

Figure 3.16. `FlightDepartureReply` Class Diagram

(1) Reply returned in response to FlightDepartureRequest.

(2) If the given flight identification is unknown or ambiguous, the error is reported as a generic input validation error (object not found), as described in the manual CommonServices .

(3) Note that flight departure filing is always either accepted or rejected; it never results in queuing for manual correction by an NM operator. In model terms, this means that the `filingStatus` associated to the returned `FilingReply` never takes the `INVALID_QUEUED_FOR_CORRECTION` value, and therefore its `queuedFiling` attribute is always null in the case of flight departure filing.

(4) Inherits from: `Reply`

3.2.7. Flight Arrival

3.2.7.1. SOAP

(1) The associated SOAP operation is:

```
FlightArrivalReply fileFlightArrival(
    FlightArrivalRequest request
    )
```

3.2.7.2. `FlightArrivalRequest`

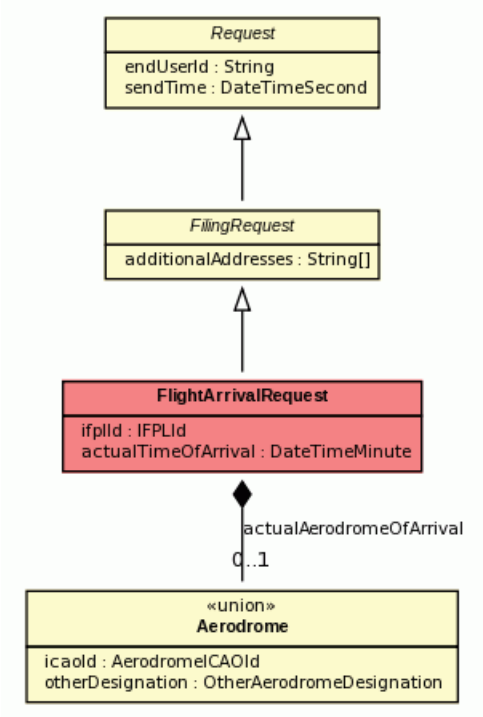

Figure 3.17. `FlightArrivalRequest` Class Diagram

(1) Request to notify the arrival of a flight.

(2) Inherits from: `FilingRequest`

(3) Attributes:

a) **`IFPLId` `ifplId`** *(Mandatory)*
Identification of the flight to be updated.

b) **`DateTimeMinute` `actualTimeOfArrival`** *(Mandatory)*
Actual date/time of arrival.

c) **`Aerodrome` `actualAerodromeOfArrival`** *(Optional)*
Actual aerodrome of arrival.

3.2.7.3. `FlightArrivalReply`

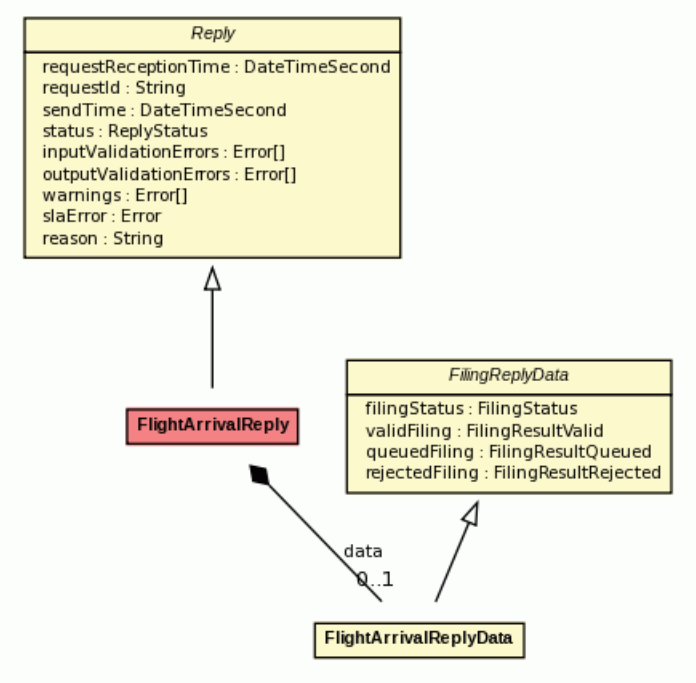

Figure 3.18. `FlightArrivalReply` Class Diagram

(1) Reply returned in response to FlightArrivalRequest.

(2) If the given flight identification is unknown or ambiguous, the error is reported as a generic input validation error (object not found), as described in the manual CommonServices .

(3) Note that flight arrival filing is always either accepted or rejected; it never results in queuing for manual correction by an NM operator. In model terms, this means that the `filingStatus` associated to the returned `FilingReply` never takes the `INVALID_QUEUED_FOR_CORRECTION` value, and therefore its `queuedFiling` attribute is always null in the case of flight arrival filing.

(4) Inherits from: `Reply`

3.2.8. Filing Status

3.2.8.1. SOAP

(1) The associated SOAP operation is:

```
FilingStatusReply retrieveFilingStatus(
    FilingStatusRequest request
    )
```

3.2.8.2. `FilingStatusRequest`

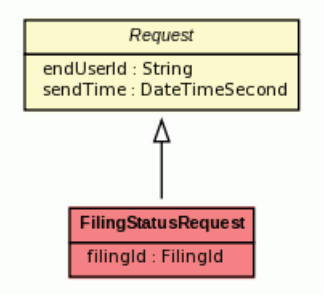

Figure 3.19. `FilingStatusRequest` Class Diagram

(1) Request for the current status of a previous filing request that resulted into queuing for manual correction by an NM operator, i.e. valid (after manual correction), still queued for manual correction, or rejected (after manual correction).

(2) Inherits from: `Request`

(3) Attributes:

a) **`FilingId` `filingId`** *(Mandatory)*
Filing id that was returned by the filing reply via `FilingResultQueued`.

3.2.8.3. `FilingStatusReply`

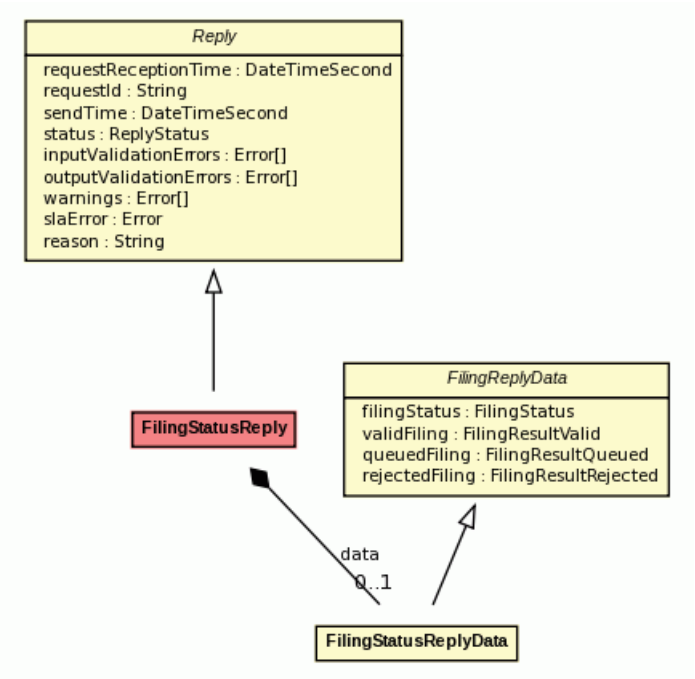

Figure 3.20. `FilingStatusReply` Class Diagram

(1) Reply returned in response to FilingStatusRequest.

(2) Note that this reply is a `FilingReply`, although the FilingStatusRequest is not a `FilingRequest`. This conveys the fact that requesting a filing status is typically achieved asynchronously (in polling) after a filing request, but still returns status information as if the filing reply has been returned synchronously when the original request was filed.

(3) Inherits from: `Reply`

3.2.9. Extended Flight Plan Creation

3.2.9.1. SOAP

(1) The associated SOAP operation is:

```
ExtendedFlightPlanCreationReply fileNewExtendedFlightPlan(
    ExtendedFlightPlanCreationRequest request
    )
```

3.2.9.2. `ExtendedFlightPlanCreationRequest`

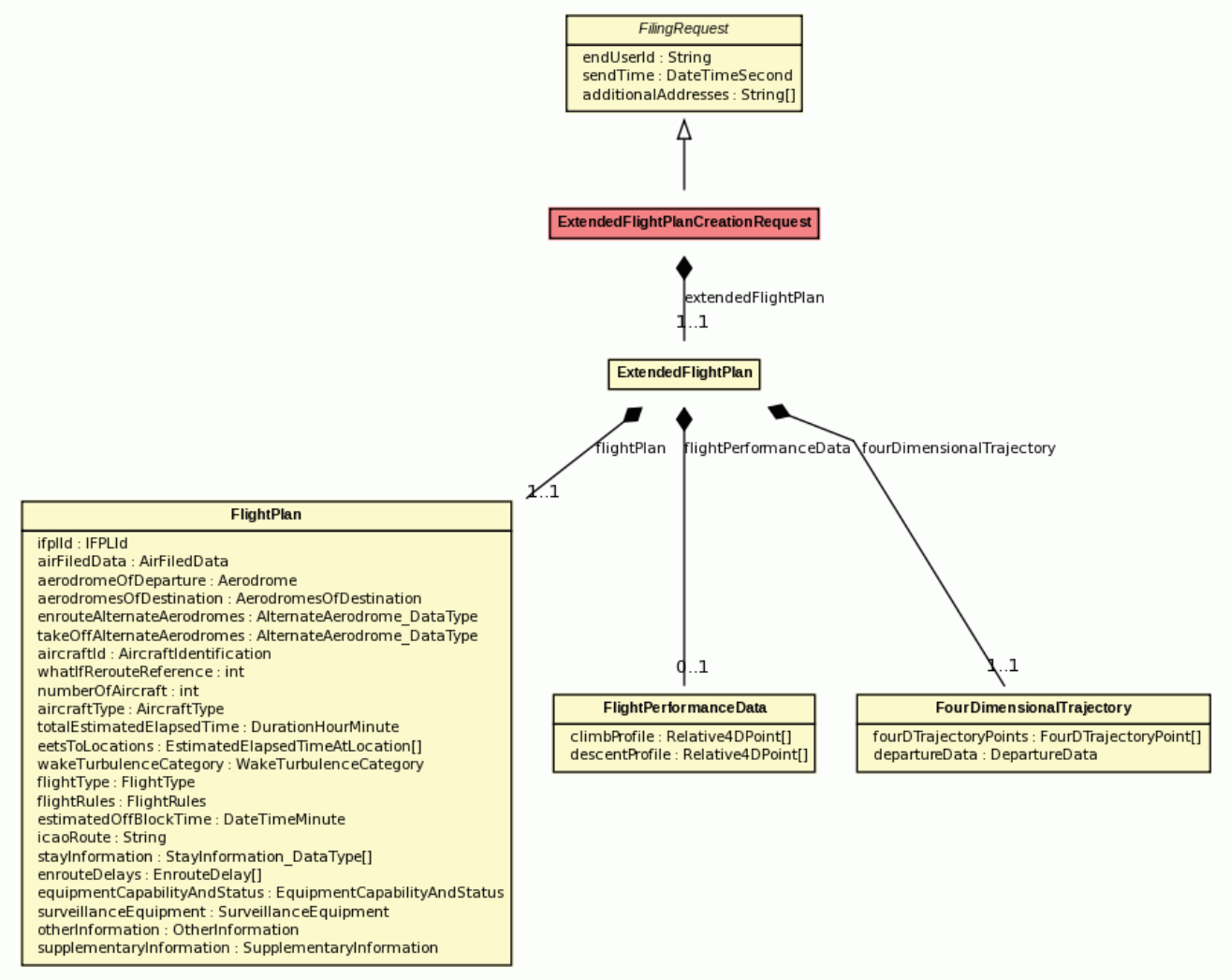

Figure 3.21. `ExtendedFlightPlanCreationRequest` Class Diagram

(1) Request the submission (or filing) of a new flight plan to the NM.

(2) The same new flight plan cannot be created more than once: if NM detects that the submitted new flight plan has the same flight keys as an existing flight plan, the request is rejected with an error.

(3) Inherits from: `FilingRequest`

(4) Attributes:

a) **`ExtendedFlightPlan` `extendedFlightPlan`** *(Mandatory)*
Flight plan message including an ICAO defined flight plan information, a 4D trajectory and Performance Data specific to the flight.

3.2.9.3. `ExtendedFlightPlanCreationReply`

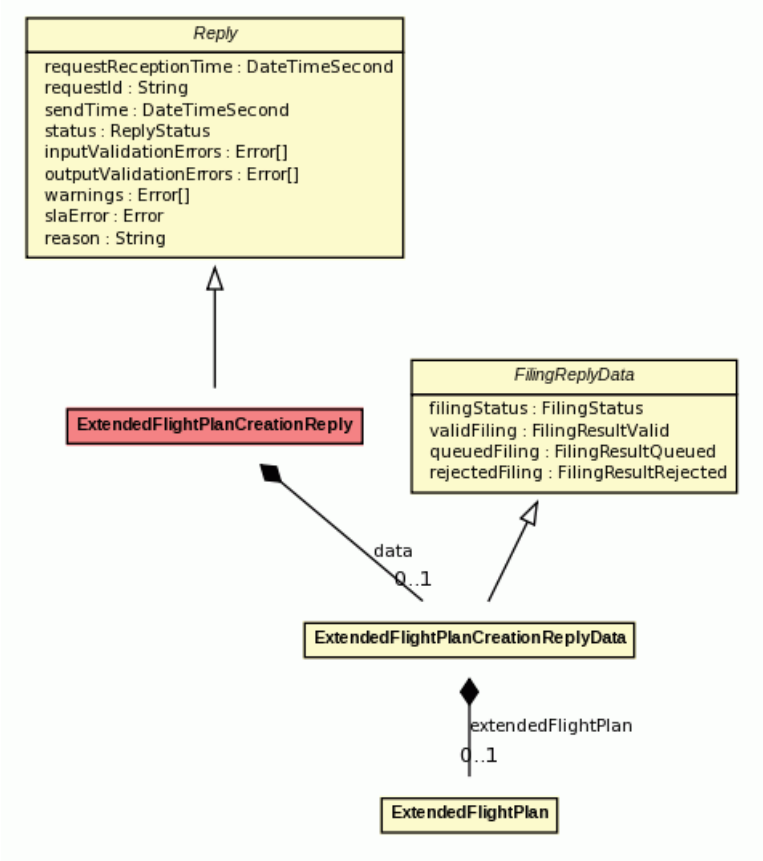

Figure 3.22. `ExtendedFlightPlanCreationReply` Class Diagram

(1) Reply returned in response to ExtendedFlightPlanCreationRequest.

(2) Inherits from: `Reply`

(3) Attributes:

a) **`ExtendedFlightPlan extendedFlightPlan`** *(Optional)*
The extended flight plan as accepted by NM, may have been automatically or manually corrected.
Cannot be null if FilingReply.filingStatus is VALID, must be null otherwise.

3.2.10. Extended Flight Plan Update

3.2.10.1. SOAP

(1) The associated SOAP operation is:

```
ExtendedFlightPlanUpdateReply fileExtendedFlightPlanUpdate(
    ExtendedFlightPlanUpdateRequest request
    )
```

3.2.10.2. `ExtendedFlightPlanUpdateRequest`

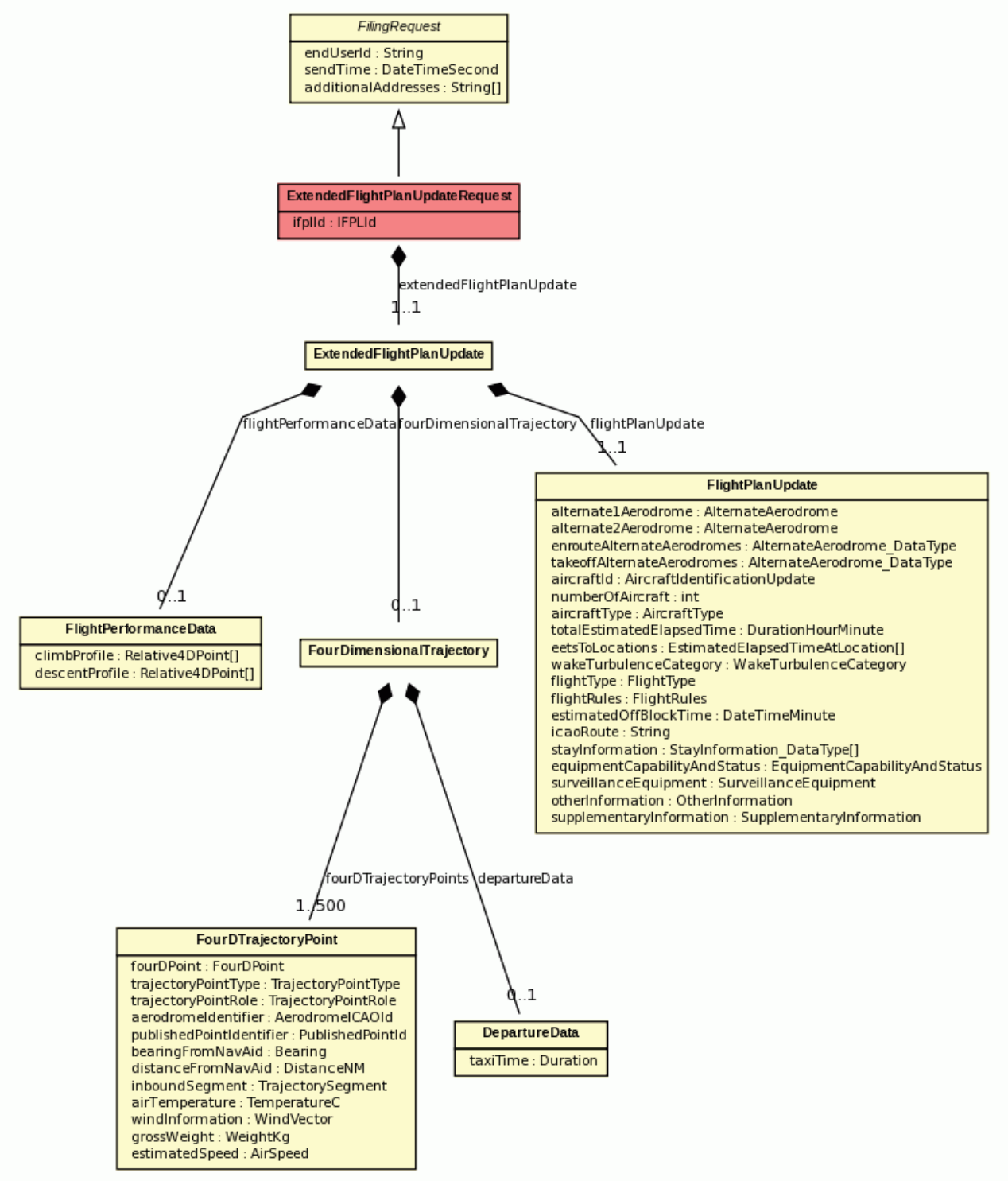

Figure 3.23. `ExtendedFlightPlanUpdateRequest` Class Diagram

(1) Request for the update of an extended flight plan.

(2) The `ExtendedFlightPlanUpdateRequest` supports the update of some selective fields which are updated in the existing flight plan while others are left unchanged.

(3) Inherits from: `FilingRequest`

(4) Attributes:

a) **`IFPLId` `ifplId`** *(Mandatory)*
Identification of the flight plan to be updated.

b) **`ExtendedFlightPlanUpdate` `extendedFlightPlanUpdate`** *(Mandatory)*
Update of selective fields in the existing flight plan.
Contains also the 4D Trajectory and Performance data when available from the AO.

3.2.10.3. `ExtendedFlightPlanUpdateReply`

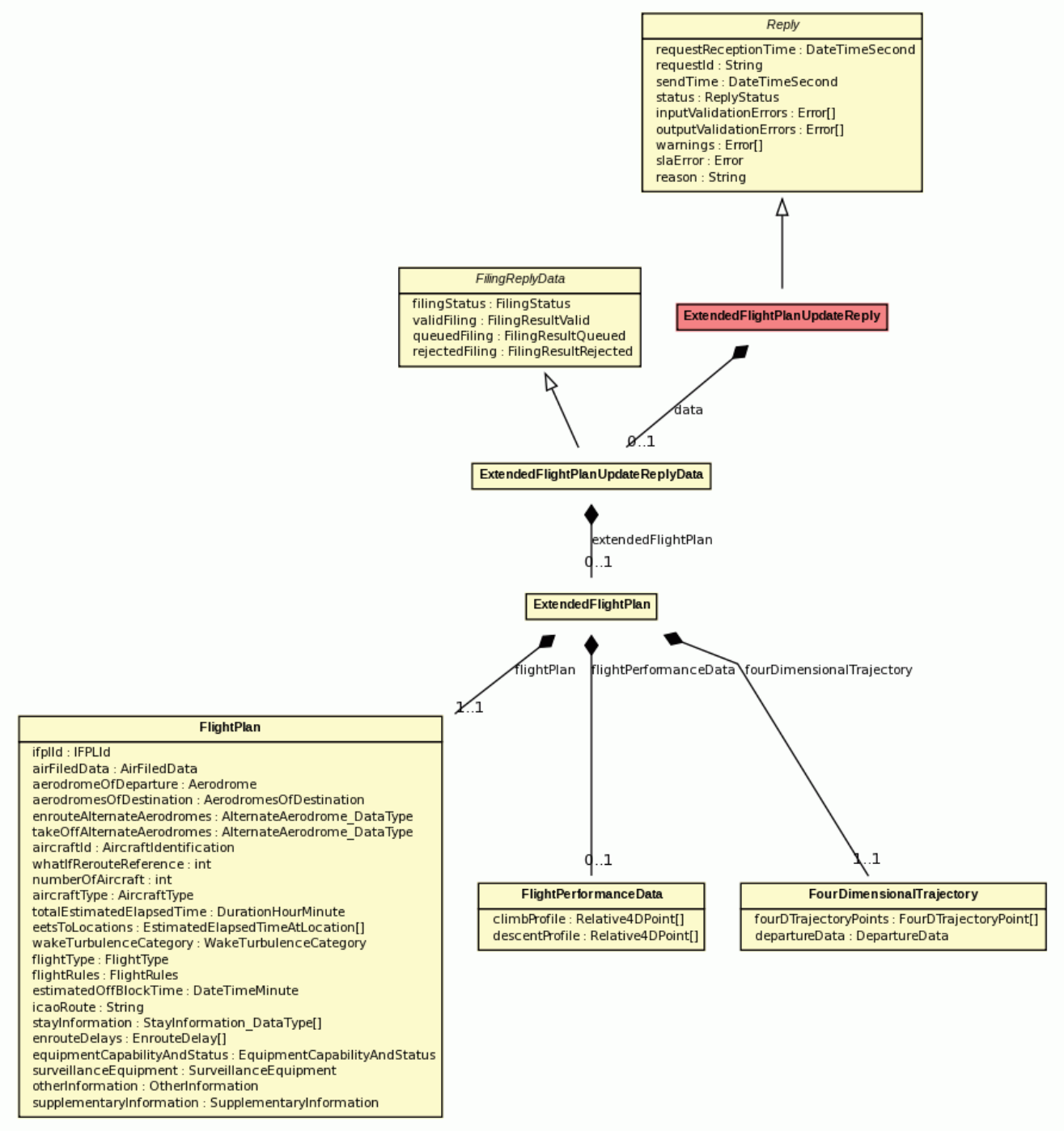

Figure 3.24. `ExtendedFlightPlanUpdateReply` Class Diagram

(1) Reply returned in response to ExtendedFlightPlanUpdateRequest.

(2) If the given flight identification is unknown or ambiguous, the error is reported as a generic input validation error (object not found), as described in section References.

(3) Inherits from: Reply

(4) Attributes:

a) **ExtendedFlightPlan extendedFlightPlan** *(Optional)*
The extended flight plan as updated by NM, may have been automatically or manually corrected.
Cannot be null if FilingReply.filingStatus is VALID, meaning that the extended flight plan was indeed updated, must be null otherwise.

3.3. FlightManagementService Port Type

3.3.1. Overview

3.3.1.1. Operations

(1) FlightManagementService provides requests aimed at managing flight plans and flights:

a) FlightPlanListRequest / FlightPlanListReply

b) FlightListByKeysRequest / FlightListByKeysReply

c) FlightListByAircraftOperatorRequest / FlightListByAircraftOperatorReply

d) FlightListByAerodromeRequest / FlightListByAerodromeReply

e) FlightListByAerodromeSetRequest / FlightListByAerodromeSetReply

f) FlightListByAirspaceRequest / FlightListByAirspaceReply

g) FlightListByPointRequest / FlightListByPointReply

h) FlightListByTrafficVolumeRequest / FlightListByTrafficVolumeReply

i) FlightListByMeasureRequest / FlightListByMeasureReply

j) FlightListByHotspotRequest / FlightListByHotspotReply

k) FlightRetrievalRequest / FlightRetrievalReply

l) EarlyDPIRequest / EarlyDPIReply

m) TargetDPITargetRequest / TargetDPITargetReply

n) TargetDPISequencedRequest / TargetDPISequencedReply

o) ATCDPIRequest / ATCDPIReply

p) CancelDPIRequest / CancelDPIReply

3.3.1.2. 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 (containing regulations/tactical updates) between D-6 (6 days in advance) and D-1 (1 day in advance) within the forecast dataset.

(2) To be able to do this accurately, also a traffic forecast is constructed/maintained.
So in this forecast dataset there are only predicted flights (CFMUFlightType: PREDICTED_FLIGHT & RPL). NM takes into account wind and NAT traffic predictions, airport slots, airline schedules, the traffic from a similar day in the past (typically the traffic from exactly one week earlier) to build up this forecast dataset.
The plan (including the prepared regulations and other tactical updates), is transferred to the operational dataset on D-1 around 16:00 UTC.

(3) The plan (and associated forecast traffic) 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.

(4) To fix the ideas:

a) the forecast dataset can be accessed : in [D-5 (5 days in the future), D 24:00 UTC]

b) the operational dataset can be accessed : at any point in time on D-1 and D via B2B,

(5) In parallel there exists the IFPS backend system that is handling the FlightFilingServices and flightplans.
The user can file his flightplan up to several days into advance.
These flightplans are then fed into the operational FlightManagement backend system (ETFMS) around 24 hours before off-block-time.
So wrt to FlightFilingServices and flightplans, the concept of operational/forecast does not apply.
There exists only one operational dataset : supporting flightplan filing multiple days in advance.
On the other hand, the forecast FlightManagemnt dataset is really a forecast containing predicted flights.
So even if a flightplan has been filed well in advance, in the forecast FlightManagement services, one might not find exactly the same flight back (as it can be adapted according to NAT predictions, closure of airspace predictions, etc).
As these flights are really predicted flights, they do not have an IFPL id.

(6) So this operational/forecast DataSet concept is not related to the FlightDataset type

(7) Note that access to forecast Dataset is subject to specific user authorisation: it is enabled/authorized on test platforms, otherwise when specifically requested by the user.

3.3.1.3. Simulation Datasets

(1) See Simulations.

3.3.1.4. Proposal Flights

(1) The NM systems (FlightManagement specifically) can have more than one version of a flight:

a) The normal flight (corresponding to what the airspace user has filed)

b) A proposal flight: a proposition from NM/ANSP to an airspace user/other ANSP.
There can be maximum one proposal flight for a given flight at a moment in time.
These proposal flights can be generated for example

   i) in the context of regulations (when NM systems have a proposal to improve)

   ii) in the context to avoid flight suspension

   iii) in the context of rerouting/level caps (RRP/RRN):
   Typically a rerouting proposal flight is used in the context of flight efficiency or ATFCM (a proposal to for example reroute around a zero rate regulation) or STAM trials (ANSP initiated) or Aircraft operator initiated (AO-What-if-Reroute (AOWIR))

c) A proposal flight also comes with a mechanism to try to commit to the proposed delay. So if, during the time the proposal exists (limted), a proposal flight is accepted (depending on the kind of proposal either by airspace users or by ANSP), then the proposed delay becomes the real delay (nominal case but there are exceptions).

d) In flight list/flight details (flight management services) and counts (flow services) the user can request to include proposals. If include proposal is requested, then if there exists a proposal flight, then the proposal flight is returned otherwise the normal flight is returned. This allows airspace users (AO) or ANSP to view/display/plot the proposal flight.

e) Note that access to proposal flights is subject to specific user authorisation: it is enabled/authorized on test platforms, otherwise when specifically requested by the user.

3.3.1.5. DPI - Departure Planning Information

3.3.1.5.1. IMPORTANT REMARK

(1) ATTENTION: The new service requests defined in the `FlightManagement` service allowing the DPI management are not yet operational. Note that they are available on the PREOPS platform but subject to Authorisation:

a) These new service requests are provided to start the development of external tools.

b) They cannot be used operationally.

c) The business behavior will work as the operational DPI using AFTN and ADEXP messages.

d) To make it operational, these service requests need to return in the reply proper error messages.

e) In the current implementation, error messages are still returned asynchroneously via the AFTN network.

3.3.1.5.2. Introduction

(1) The exchange of dynamic CDM information between the NMOC and the airport is a two-way process which consists in:

a) Sending DPI messages from the airport concerned to the NMOC. These messages contain the latest information on, for example, estimated or target times for the take-off of a particular flight, the aircraft type, taxi times, and the SID.

b) Sending Flight Update Messages (FUM) from the NMOC to the airports concerned, providing the Airport CDM platform with the flight status, the estimated landing times, etc.

The main information to be received via the DPI message is:

a) An accurate estimate of the take-off time;.

b) The taxi time.

c) The departure route (SID).

d) The aircraft type and registration.

This allows the proactive sharing of real time data with the NMOC, therefore optimising the ATFCM slot allocation process and achieving a more efficient use of the ATFCM network capacity.

3.3.1.5.3. General Overview

(1)

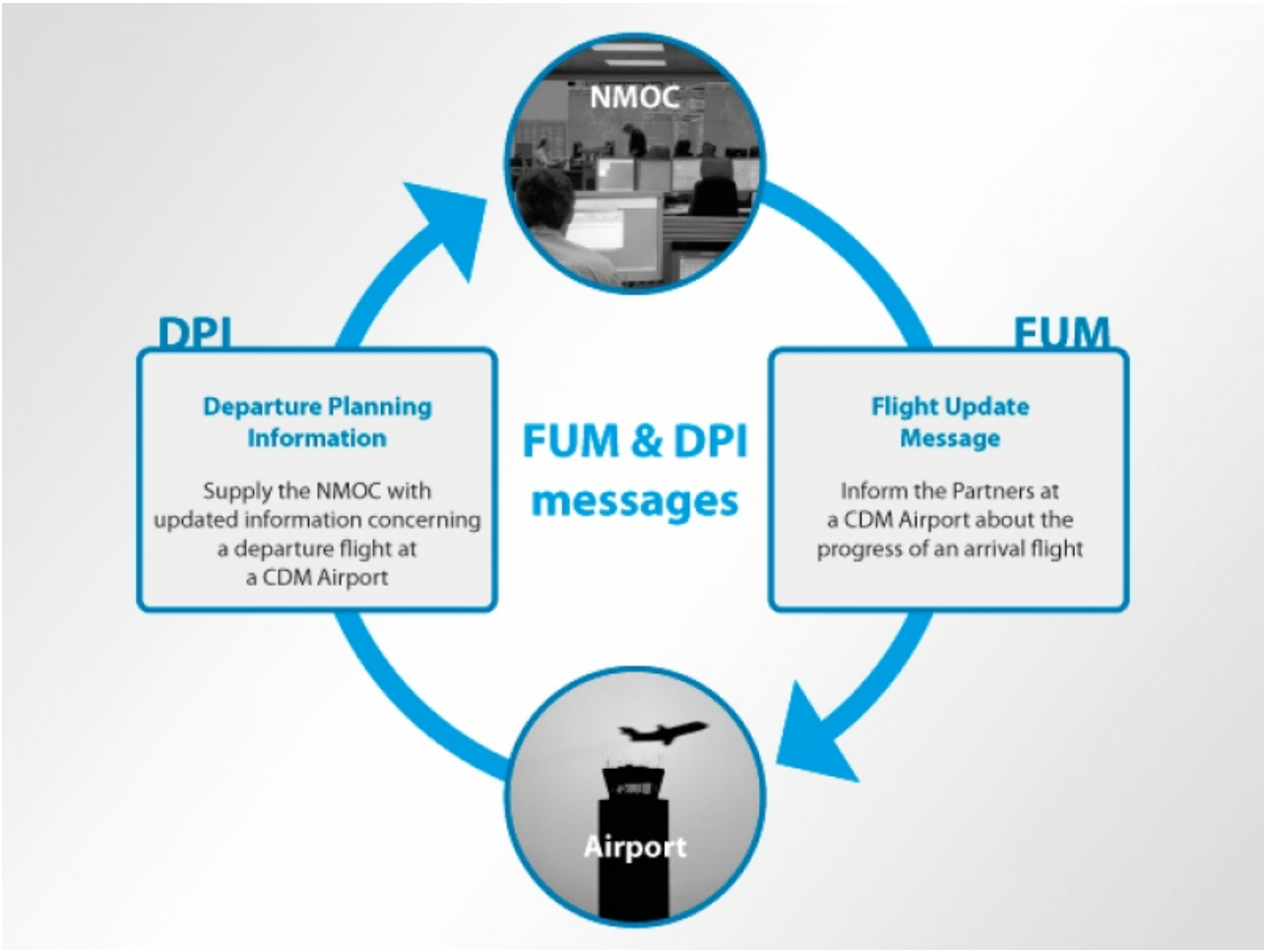

Figure 3.25. Departure Planning Information - General Overview

The airport situational information is collected direct from the Airport CDM systems in order to update the real-time flight situation, prior to take-off, in the Network Operations systems. Thanks to this improved accuracy of flight information, DPI ultimately serves to improve ATFCM traffic predictions and consequently, the effectiveness of the ATFM measures to be taken. Four phases have been identified which require coordination with ATFCM:

a) *Planning phase:* Airport schedule and flight plan estimates must be reconciled and consistent information must be sent to the Network Manager. Ghost flights and duplicated flights have to be deleted. A first evaluation of the realistic taxi-time and SID will be indicated to the Network Manager Operations Centre in order to facilitate a more realistic calculation of the ATFM slot.

b) *Turn-around phase:* Based on the flight connection, a more realistic estimate of the Off Block time will be available, based on the arrival time of the inbound flight and turn around time. It generally results in the creation and accurate maintenance of the Target Off-Block Time (TOBT) by AOs and handlers.

c) *Pre-sequencing:* 30-40mins before the TOBT, the flight is included in the ATC pre-departure sequence, which will result in a Target Start-up Approval Time (TSAT). For regulated flights, the TSAT takes the ATFM slot into account.

d) *ATC phase or pre-sequencing:* At delivery of engine start-up clearance delivery, the flight is handed over to the tower for push-back, taxiing and take-off. Local control units (ATC/Apron) will ensure that the flight goes off-blocks and takes-off as close as possible to the local target times (TSAT, TTOT).

At any time during these four phases a change in the Airport operating conditions may alter the taxi-time and/or SID.

3.3.1.5.4. Documents

Detailed information on DPI messages can be found in the following documents:

(1) DPI Implementation Guide - [6]

(2) Flight Progress Messages - [7]

(3) DPI and FUM Implementation Road Map - [8]

(4) Site of European Airport CDM - [9]

3.3.2. Flight Plan List

3.3.2.1. SOAP

(1) The associated SOAP operation is:

```
FlightPlanListReply queryFlightPlans(
    FlightPlanListRequest request
    )
```

3.3.2.2. `FlightPlanListRequest`

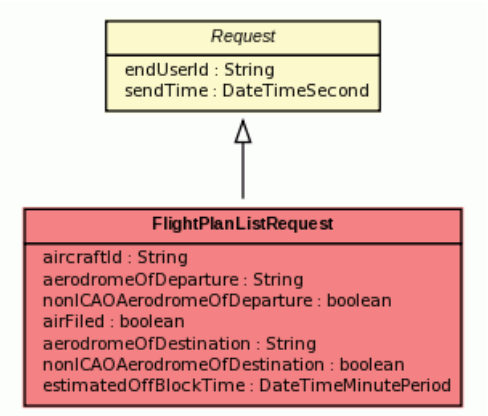

Figure 3.26. `FlightPlanListRequest` Class Diagram

(1) Request to query a flight plan list. Each item in the flight plan list is made of:

a) The summary of the last valid flight plan, if it exists.

b) The list of invalid filing summaries that are currently under manual correction by an NM operator.

(2) In order to get the full flight plan and/or the full flight plan history, the caller must use the FlightRetrievalRequest.

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

(4) The query supports wildcards, but is limited to some combinations of these wildcards in the sense that at least:

a) The aircraft id, or

b) The aerodrome of departure, or

c) The aerodrome of destination

must be fully specified. This is further detailed in the attribute definitions.

(5) Inherits from: `Request`

(6) Attributes:

a) **`string aircraftId`** *(Optional)*
ICAO aircraft id, with basic wildcard support ("*" is supported at the end of the field).
Optional: default value is "*".
(ALPHA|DIGIT){2,7} | (ALPHA|DIGIT){0,6}*
NM accepts a wildcard in this field if either:

i) `aerodromeOfDeparture` is fully specified, or

ii) `nonICAOAerodromeOfDeparture` is true, or

iii) `airFiled` is set to true, or

iv) `aerodromeOfDestination` is fully specified, or

v) `nonICAOAerodromeOfDestination` is true

Constraint: See `NONE_FULLY_SPECIFIED`

b) **`string aerodromeOfDeparture`** *(Optional)*
ICAO id of the aerodrome of departure, with basic wildcard support ("*" is supported at the end of the field).
Optional: default value is "*".
ALPHA{0,4} | ALPHA{0,3}*
This query field must be null if `nonICAOAerodromeOfDeparture` is set to true or if airFiled is set to true.
NM accepts a wildcard in this field if either:

i) `aircraftId` is fully specified, or

ii) `aerodromeOfDestination` is fully specified, or

iii) `nonICAOAerodromeOfDestination` is true

Constraints:

i) See ADEP_AIRFILED_NONICAOADEP_NOT_ALLOWED

ii) See AIRFILED_ICAOADEP_NOT_ALLOWED

iii) See NONE_FULLY_SPECIFIED

c) **`boolean nonICAOAerodromeOfDeparture`** *(Mandatory)*
True if the query concerns non ICAO aerodromes of departure; there is no way at the moment to specify what non ICAO aerodrome of departure is queried.
Cannot be true if `airFiled` is true.
Constraints:

i) See ADEP_AIRFILED_NONICAOADEP_NOT_ALLOWED

ii) See AIRFILED_NONICAOADEP_NOT_ALLOWED

d) **`boolean airFiled`** *(Mandatory)*
True if the query concerns flight plans that were filed airborne.
Cannot be true if `nonICAOAerodromeOfDeparture` is true.
Constraints:

i) See ADEP_AIRFILED_NONICAOADEP_NOT_ALLOWED

ii) See AIRFILED_ICAOADEP_NOT_ALLOWED

iii) See AIRFILED_NONICAOADEP_NOT_ALLOWED

e) **`string aerodromeOfDestination`** *(Optional)*
ICAO id of the aerodrome of destination, with basic wildcard support ("*" is supported at the end of the field).
Optional: default value is "*".
ALPHA{4} | ALPHA{0,3}*
This query field must be null if `nonICAOAerodromeOfDestination` is set to true.
NM accepts a wildcard in this field if either:

i) `aircraftId` is fully specified, or

ii) `aerodromeOfDeparture` is fully specified, or

iii) `nonICAOAerodromeOfDeparture` is true, or

iv) `airFiled` is true

Constraint: See NONE_FULLY_SPECIFIED

f) **boolean nonICAOAerodromeOfDestination** *(Mandatory)*
True if the query concerns non ICAO aerodromes of destination; there is no way at the moment to specify what non ICAO aerodrome of destination is queried.

g) **DateTimeMinutePeriod estimatedOffBlockTime** *(Mandatory)*
Period in which the estimated off-block date/time of the matching flight plans must belong.
Constraint: See PERIOD_EXTENSION_CANNOT_BE_GREATER_THAN_24_HOURS

(7) Constraints:

a)

Name	ADEP_AIRFILED_NONICAOADEP_NOT_ALLOWED
Attributes	airFiled, aerodromeOfDeparture, nonICAOAerodromeOfDeparture
Description	aerodromeOfDeparture must be null or * if airFiled is true and nonICAOAerodromeOfDeparture is true

b)

Name	NONE_FULLY_SPECIFIED
Attributes	aircraftId, aerodromeOfDeparture, aerodromeOfDestination
Description	At least one of aircraftId, aerodromeOfDeparture, aerodromeOfDestination should be fully specified

c)

Name	AIRFILED_NONICAOADEP_NOT_ALLOWED
Attributes	airFiled, nonICAOAerodromeOfDeparture
Description	nonICAOAerodromeOfDeparture and airFiled cannot both be true

d)

Name	AIRFILED_ICAOADEP_NOT_ALLOWED
Attributes	airFiled, aerodromeOfDeparture
Description	airFiled cannot be true if aerodromeOfDeparture has been specified

e)

Name	PERIOD_EXTENSION_CANNOT_BE_GREATER_THAN_24_HOURS
Attribute	estimatedOffBlockTime
Description	The period cannot extend more than 24 hours.

3.3.2.3. `FlightPlanListReply`

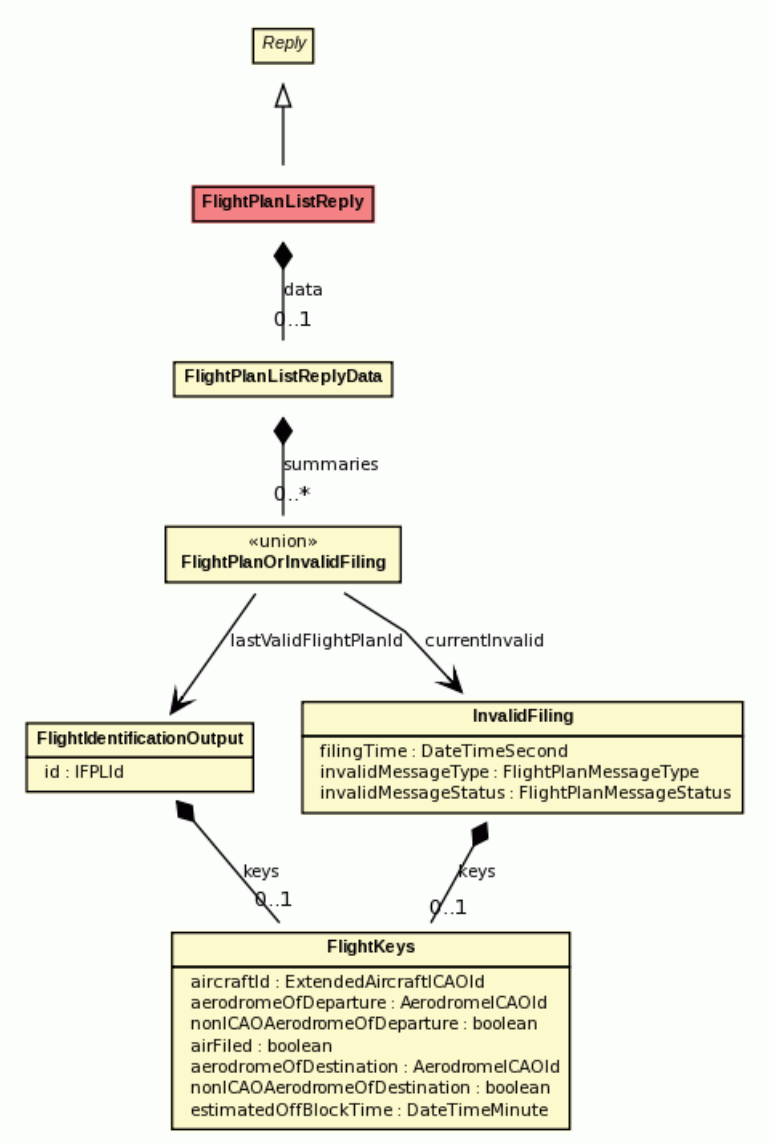

Figure 3.27. `FlightPlanListReply` Class Diagram

(1) Reply returned in response to FlightPlanListRequest.

(2) Inherits from: `Reply`

(3) Attributes:

a) **`FlightPlanOrInvalidFiling[] summaries`** *(Mandatory)*
The summaries of the valid flight plans and invalid filings matching the query fields.
Mandatory: if no match was found, the returned array is empty.
The array does not contain null or duplicate items.
Constraint: Size must be comprised between 0 and ∞.

3.3.3. Flight List by Keys

3.3.3.1. SOAP

(1) The associated SOAP operation is:

```
FlightListByKeysReply queryFlightsByKeys(
    FlightListByKeysRequest request
    )
```

3.3.3.2. `FlightListByKeysRequest`

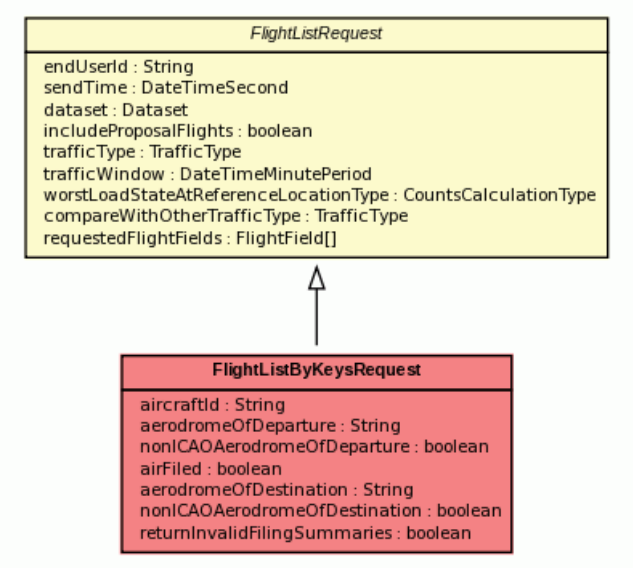

Figure 3.28. `FlightListByKeysRequest` Class Diagram

(1) Request to query a flight list from flight keys.

(2) If the request attribute `returnInvalidFilingSummaries` is true, the `flights` array contains flight plans or invalid filing summary information in addition to the `Flight` data.

(3) The logical AND operator applies between all the query fields described below and the query fields of its ancestor request.

(4) The query supports wildcards, but is limited to some combinations of these wildcards in the sense that at least:

a) The aircraft id, or

b) The aerodrome of departure, or

c) The aerodrome of destination

must be fully specified. This is further detailed in the attribute definitions.

(5) Inherits from: `FlightListRequest`

(6) Attributes:

a) **`string aircraftId`** *(Optional)*
ICAO aircraft id, with basic wildcard support ("*" is supported at the end of the field).
Optional: default value is "*".
The aircraft identifier may include special characters ("$", "#") for flights created by NM during prediction and simulation exercises.
NM accepts a wildcard in this field if either:

i) `aerodromeOfDeparture` is fully specified, or

ii) `aerodromeOfDestination` is fully specified
Constraints:

i) Pattern: `(ALPHA|DIGIT){2,7}|(ALPHA|DIGIT){0,6}*`

ii) See NONE_FULLY_SPECIFIED

b) **`string aerodromeOfDeparture`** *(Optional)*
ICAO id of the aerodrome of departure, with basic wildcard support ("*" is supported at the end of the field).
Optional: default value is "*".
This query field must be null if nonICAOAerodromeOfDeparture is set to true or if airFiled is set to true.
NM accepts a wildcard in this field if either:

i) `aircraftId` is fully specified, or

ii) `aerodromeOfDestination` is fully specified
Constraints:

i) Pattern: `ALPHA{4}|ALPHA{0,3}*`

ii) See ADEP_AIRFILED_NONICAOADEP_NOT_ALLOWED

iii) See AIRFILED_ICAOADEP_NOT_ALLOWED

iv) See NONE_FULLY_SPECIFIED

c) **`boolean nonICAOAerodromeOfDeparture`** *(Mandatory)*
True if the query concerns non ICAO aerodromes of departure; there is no way at the moment to specify what non ICAO aerodrome of departure is queried.
Cannot be true if `airFiled` is true.
Constraints:

i) See ADEP_AIRFILED_NONICAOADEP_NOT_ALLOWED

ii) See AIRFILED_NONICAOADEP_NOT_ALLOWED

d) **`boolean airFiled`** *(Mandatory)*

True if the query concerns flight plans that were filed airborne.
Cannot be true if `nonICAOAerodromeOfDeparture` is true.
Constraints:

i) See ADEP_AIRFILED_NONICAOADEP_NOT_ALLOWED

ii) See AIRFILED_ICAOADEP_NOT_ALLOWED

iii) See AIRFILED_NONICAOADEP_NOT_ALLOWED

e) **`string aerodromeOfDestination`** *(Optional)*
ICAO id of the aerodrome of destination, with basic wildcard support ("*" is supported at the end of the field).
Optional: default value is "*".
This query field must be null if `nonICAOAerodromeOfDestination` is set to true.
NM accepts a wildcard in this field if either:

i) `aircraftId` is fully specified, or

ii) `aerodromeOfDeparture` is fully specified
Constraints:

i) Pattern:`ALPHA{4}|ALPHA{0,3}*`

ii) See NONE_FULLY_SPECIFIED

f) **`boolean nonICAOAerodromeOfDestination`** *(Optional)*
True if the query concerns non ICAO aerodromes of destination; there is no way at the moment to specify what non ICAO aerodrome of destination is queried.
Constraint: See NONICAOADES_CANNOT_BE_NULL

g) **`boolean returnInvalidFilingSummaries`** *(Optional)*
Indicates that the flight plan filing summary is to be returned.
If true, the objects returned in the `flights` array contain a summary of the flight plan filing information.
Constraints:

i) See NONICAOADES_CANNOT_BE_NULL

ii) See RETURN_INVALID_FILING_SUMMARIES_NEEDS_TO_BE_SET_TO_FALSE

(7) Constraints:

a)

Name	ADEP_AIRFILED_NONICAOADEP_NOT_ALLOWED
Attributes	airFiled, aerodromeOfDeparture, nonICAOAerodromeOfDeparture
Description	`aerodromeOfDeparture` must be null or * if `airFiled` is true and `nonICAOAerodromeOfDeparture` is true

b)

Name	NONE_FULLY_SPECIFIED
Attributes	aircraftId, aerodromeOfDeparture, aerodromeOfDestination
Description	At least one of aircraftId, aerodromeOfDeparture, aerodromeOfDestination should be fully specified

c)

Name	AIRFILED_NONICAOADEP_NOT_ALLOWED
Attributes	airFiled, nonICAOAerodromeOfDeparture
Description	nonICAOAerodromeOfDeparture and airFiled cannot both be true

d)

Name	AIRFILED_ICAOADEP_NOT_ALLOWED
Attributes	airFiled, aerodromeOfDeparture
Description	airFiled cannot be true if aerodromeOfDeparture has been specified

e)

Name	RETURN_INVALID_FILING_SUMMARIES_NEEDS_TO_BE_SET_TO_FALSE
Attribute	returnInvalidFilingSummaries
Description	returnInvalidFilingSummaries needs to be false if FlightListRequest.dataset is FORECAST or SIMULATION

f)

Name	NONICAOADES_CANNOT_BE_NULL
Attributes	nonICAOAerodromeOfDestination, returnInvalidFilingSummaries
Description	nonICAOAerodromeOfDestination cannot be null if returnInvalidFilingSummaries is true

3.3.3.3. `FlightListByKeysReply`

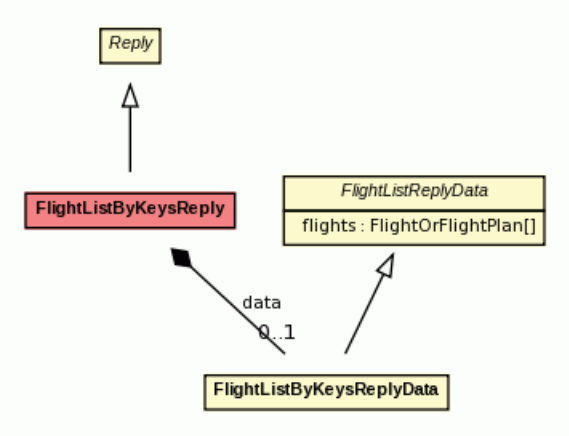

Figure 3.29. `FlightListByKeysReply` Class Diagram

(1) Reply returned in response to FlightListByKeysRequest

(2) See FlightListReplyData.

(3) Inherits from: `Reply`

3.3.4. Flight List by Aircraft Operator

3.3.4.1. SOAP

(1) The associated SOAP operation is:

```
FlightListByAircraftOperatorReply queryFlightsByAircraftOperator(
    FlightListByAircraftOperatorRequest request
    )
```

3.3.4.2. `FlightListByAircraftOperatorRequest`

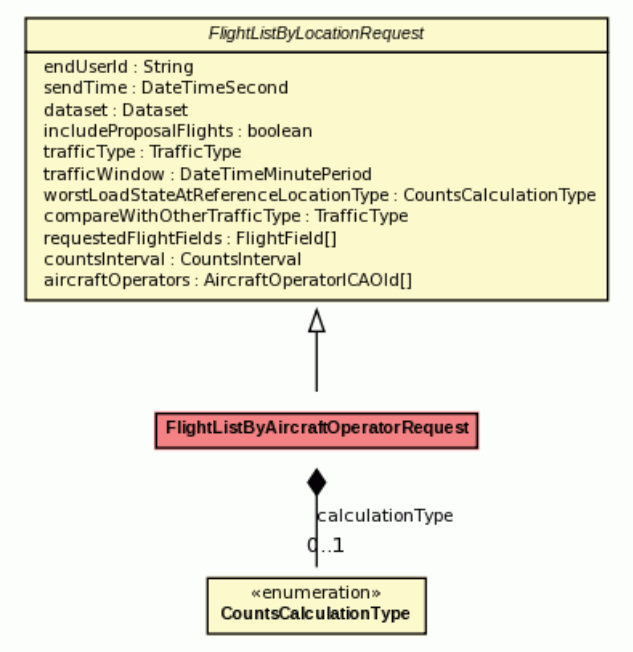

Figure 3.30. `FlightListByAircraftOperatorRequest` Class Diagram

(1) Request to query a flight list by aircraft operator only.

(2) Inherits from: `FlightListByLocationRequest`

(3) Attributes:

a) **`CountsCalculationType` `calculationType`** *(Optional)*
Indicates what is the calculation type of the count (entry or occupancy).
By default, `calculationType` is ENTRY.

3.3.4.3. `FlightListByAircraftOperatorReply`

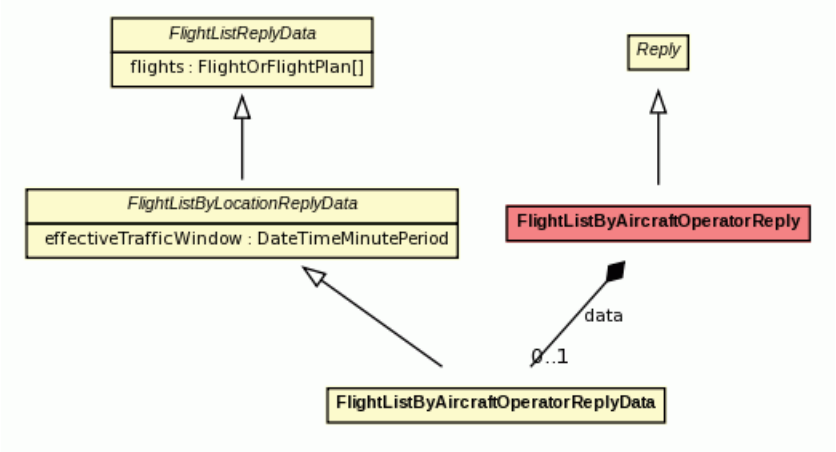

Figure 3.31. `FlightListByAircraftOperatorReply` Class Diagram

(1) Reply returned in response to FlightListByAircraftOperatorRequest.

(2) See FlightListReplyData.

(3) Inherits from: `Reply`

3.3.5. Flight List by Aerodrome

3.3.5.1. SOAP

(1) The associated SOAP operation is:

```
FlightListByAerodromeReply queryFlightsByAerodrome(
    FlightListByAerodromeRequest request
    )
```

3.3.5.2. `FlightListByAerodromeRequest`

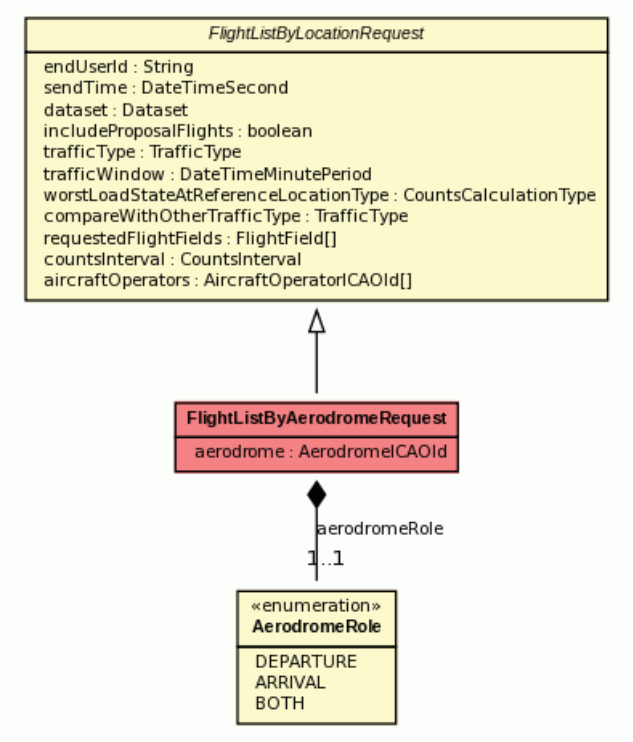

Figure 3.32. `FlightListByAerodromeRequest` Class Diagram

(1) Request to query a flight list by aerodrome.

(2) The logical AND operator applies between all the query fields described below and those inherited from FlightListByLocationRequest.

(3) Inherits from: `FlightListByLocationRequest`

(4) Attributes:

a) **`AerodromeICAOId` `aerodrome`** *(Mandatory)*
ICAO id of the aerodrome.

b) **`AerodromeRole` `aerodromeRole`** *(Mandatory)*
Specifies whether the aerodrome is meant to be departure, arrival or both.
If `aerodromeRole` is set to `AerodromeRole.DEPARTURE`, the traffic window specifies that only those flights taking off in the time window are requested.
If `aerodromeRole` is set to `AerodromeRole.ARRIVAL`, the traffic window specifies that only those flights arriving in the time window are requested.
If `aerodromeRole` is set to `AerodromeRole.BOTH`, the traffic window specifies that only those flights taking off or arriving in the time window are requested.

3.3.5.3. `FlightListByAerodromeReply`

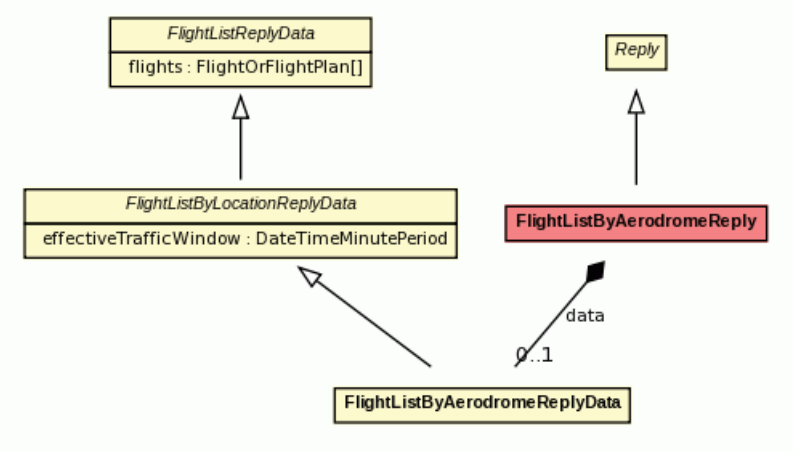

Figure 3.33. `FlightListByAerodromeReply` Class Diagram

(1) Reply returned in response to FlightListByAerodromeRequest.

(2) See FlightListReplyData.

(3) Inherits from: `Reply`

3.3.6. Flight List by Aerodrome Set

3.3.6.1. SOAP

(1) The associated SOAP operation is:

```
FlightListByAerodromeSetReply queryFlightsByAerodromeSet(
    FlightListByAerodromeSetRequest request
    )
```

3.3.6.2. `FlightListByAerodromeSetRequest`

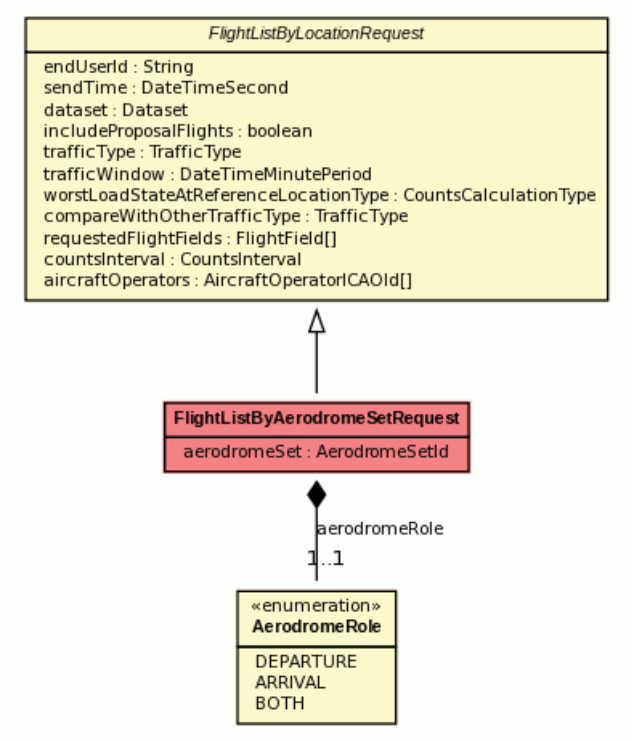

Figure 3.34. `FlightListByAerodromeSetRequest` Class Diagram

(1) Request to query a flight list by aerodrome set.

(2) The logical AND operator applies between all the query fields described below and those inherited from FlightListByLocationRequest.

(3) Inherits from: `FlightListByLocationRequest`

(4) Attributes:

a) **`AerodromeSetId` `aerodromeSet`** *(Mandatory)*
Id of the aerodrome set.

b) **`AerodromeRole` `aerodromeRole`** *(Mandatory)*
Specifies whether the aerodrome is meant to be departure, arrival or both.
If `aerodromeRole` is set to `AerodromeRole.DEPARTURE`, the traffic window specifies that only those flights taking off in the time window are requested.
If `aerodromeRole` is set to `AerodromeRole.ARRIVAL`, the traffic window specifies that only those flights arriving in the time window are requested.
If `aerodromeRole` is set to `AerodromeRole.BOTH`, the traffic window specifies that only those flights taking off or arriving in the time window are requested.

3.3.6.3. `FlightListByAerodromeSetReply`

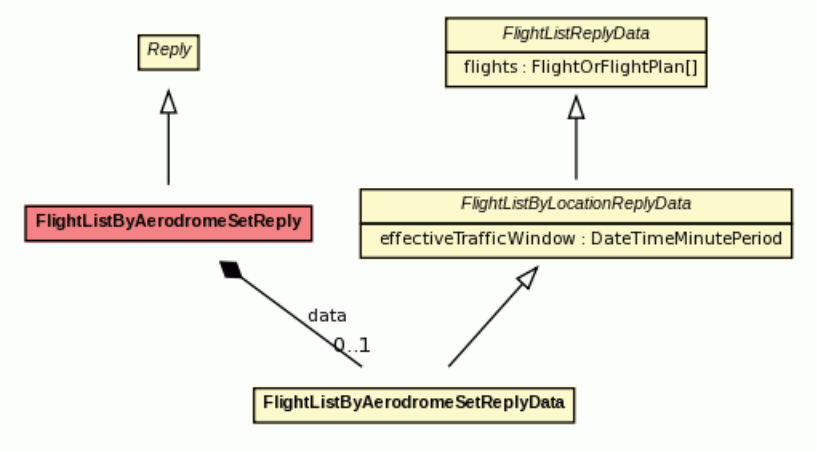

Figure 3.35. `FlightListByAerodromeSetReply` Class Diagram

(1) Reply returned in response to FlightListByAerodromeSetRequest.

(2) See FlightListReplyData.

(3) Inherits from: `Reply`

3.3.7. Flight List by Airspace

3.3.7.1. SOAP

(1) The associated SOAP operation is:

```
FlightListByAirspaceReply queryFlightsByAirspace(
    FlightListByAirspaceRequest request
    )
```

3.3.7.2. `FlightListByAirspaceRequest`

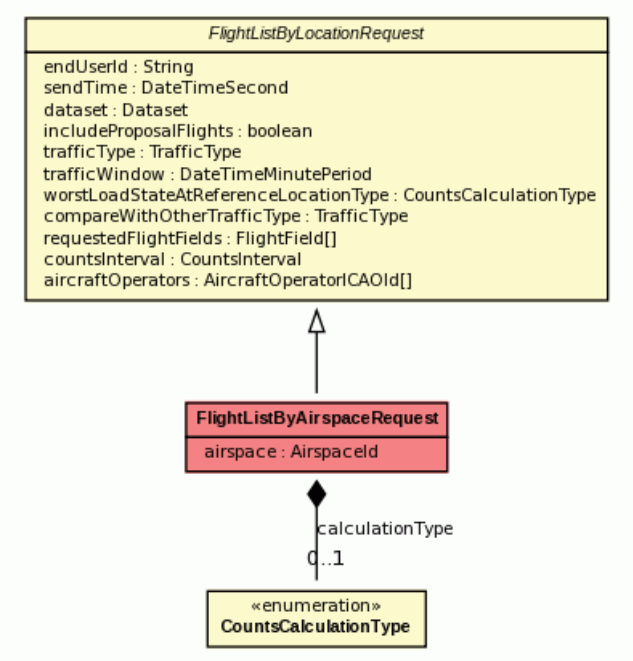

Figure 3.36. `FlightListByAirspaceRequest` Class Diagram

(1) Request to query a flight list by airspace, i.e. returns all flights occupying the given airspace during the given traffic window.

(2) The logical AND operator applies between all the query fields described below and those inherited from FlightListByLocationRequest.

(3) Inherits from: `FlightListByLocationRequest`

(4) Attributes:

a) **`CountsCalculationType` `calculationType`** *(Optional)*
Indicates what is the calculation type of the count (entry or occupancy).
By default, `calculationType` is ENTRY.

b) **`AirspaceId` `airspace`** *(Mandatory)*
Id of the airspace.

3.3.7.3. `FlightListByAirspaceReply`

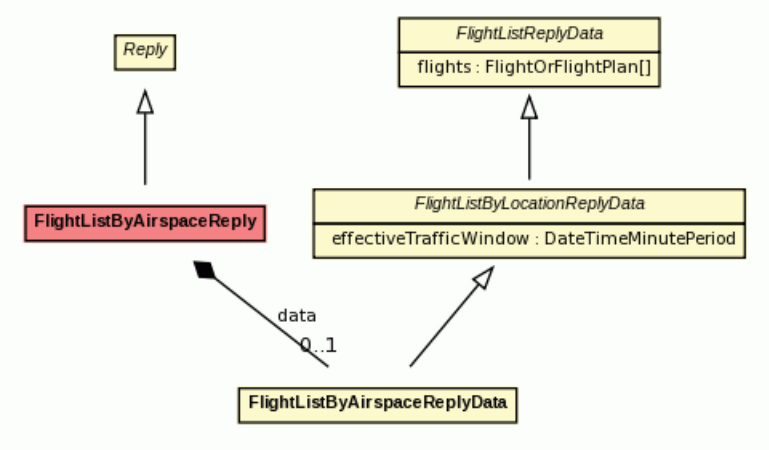

Figure 3.37. `FlightListByAirspaceReply` Class Diagram

(1) Reply returned in response to FlightListByAirspaceRequest.

(2) See FlightListReplyData.

(3) Inherits from: `Reply`

3.3.8. Flight List by Point

3.3.8.1. SOAP

(1) The associated SOAP operation is:

```
FlightListByPointReply queryFlightsByPoint(
    FlightListByPointRequest request
    )
```

3.3.8.2. `FlightListByPointRequest`

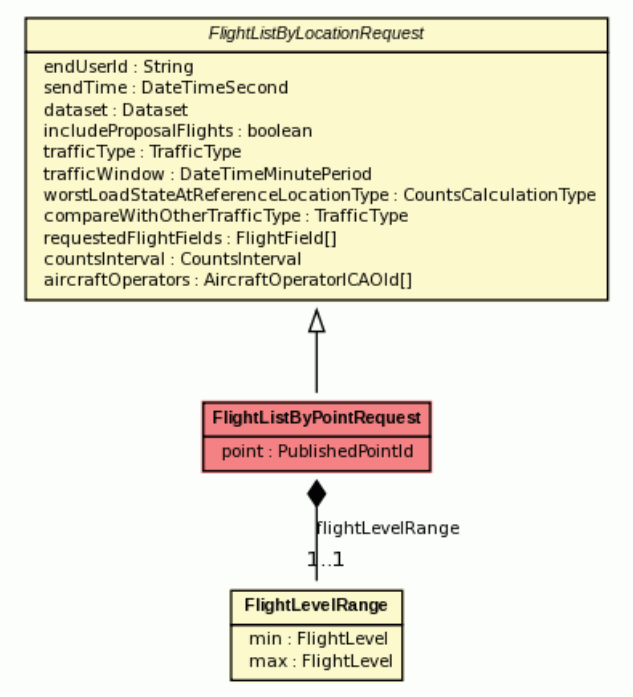

Figure 3.38. `FlightListByPointRequest` Class Diagram

(1) Request to query a flight list by point, i.e. returns flights being over that point at a time included in the given traffic window.

(2) The logical AND operator applies between all the query fields described below and those inherited from FlightListByLocationRequest.

(3) Inherits from: `FlightListByLocationRequest`

(4) Attributes:

a) **`PublishedPointId` `point`** *(Mandatory)*
Id of the published point.

b) **`FlightLevelRange` `flightLevelRange`** *(Mandatory)*
The range in which the flight level should be over the point.

3.3.8.3. `FlightListByPointReply`

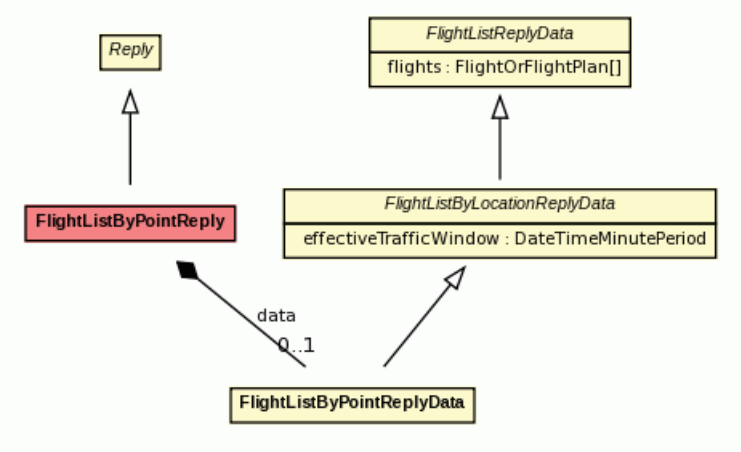

Figure 3.39. `FlightListByPointReply` Class Diagram

(1) Reply returned in response to FlightListByPointRequest.

(2) See FlightListReplyData.

(3) Inherits from: `Reply`

3.3.9. Flight List by Traffic Volume

3.3.9.1. SOAP

(1) The associated SOAP operation is:

```
FlightListByTrafficVolumeReply queryFlightsByTrafficVolume(
    FlightListByTrafficVolumeRequest request
    )
```

3.3.9.2. `FlightListByTrafficVolumeRequest`

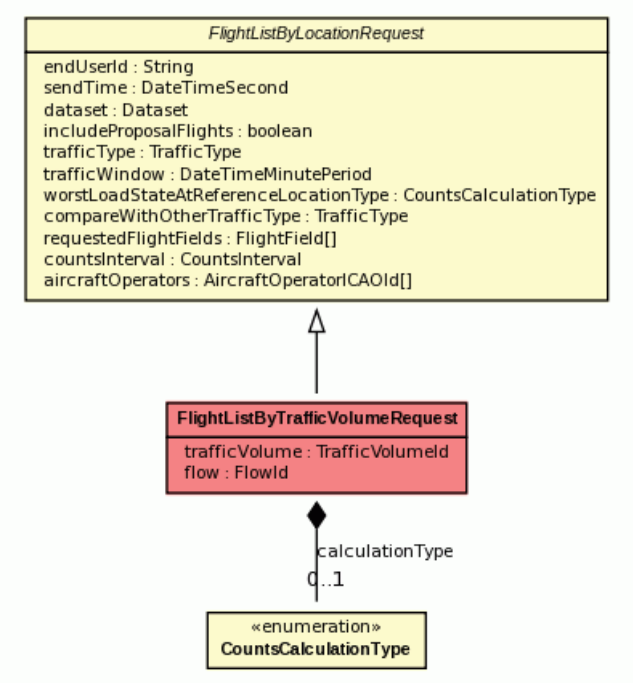

Figure 3.40. `FlightListByTrafficVolumeRequest` Class Diagram

(1) Request to query a flight list by traffic volume.

(2) The logical AND operator applies between all the query fields described below and those inherited from FlightListByLocationRequest.

(3) Inherits from: `FlightListByLocationRequest`

(4) Attributes:

a) **`CountsCalculationType` `calculationType`** *(Optional)*
Indicates what is the calculation type of the count (entry or occupancy).
Note: Occupancy counts for traffic volumes are only supported for traffic volumes defined on an airspace.

b) **`TrafficVolumeId` `trafficVolume`** *(Mandatory)*
Id of the traffic volume.

c) **`FlowId` `flow`** *(Optional)*
Id of the traffic flow.

3.3.9.3. `FlightListByTrafficVolumeReply`

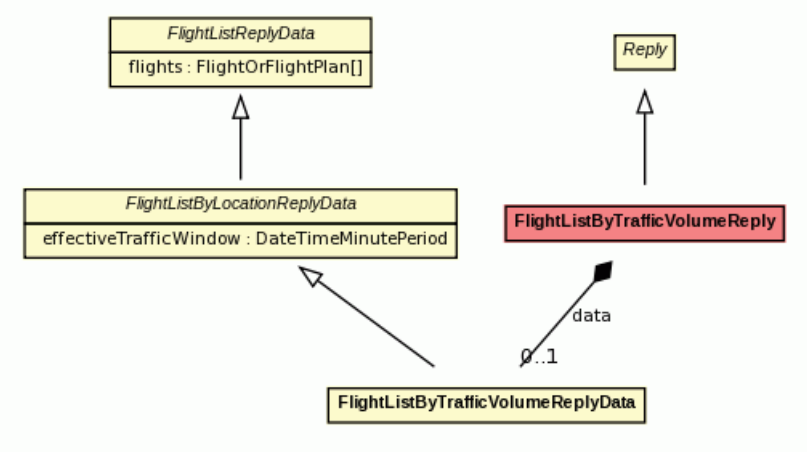

Figure 3.41. `FlightListByTrafficVolumeReply` Class Diagram

(1) Reply returned in response to FlightListByTrafficVolumeRequest.

(2) See FlightListReplyData.

(3) Inherits from: `Reply`

3.3.10. Flight List by Measure

3.3.10.1. SOAP

(1) The associated SOAP operation is:

```
FlightListByMeasureReply queryFlightsByMeasure(
    FlightListByMeasureRequest request
    )
```

3.3.10.2. `FlightListByMeasureRequest`

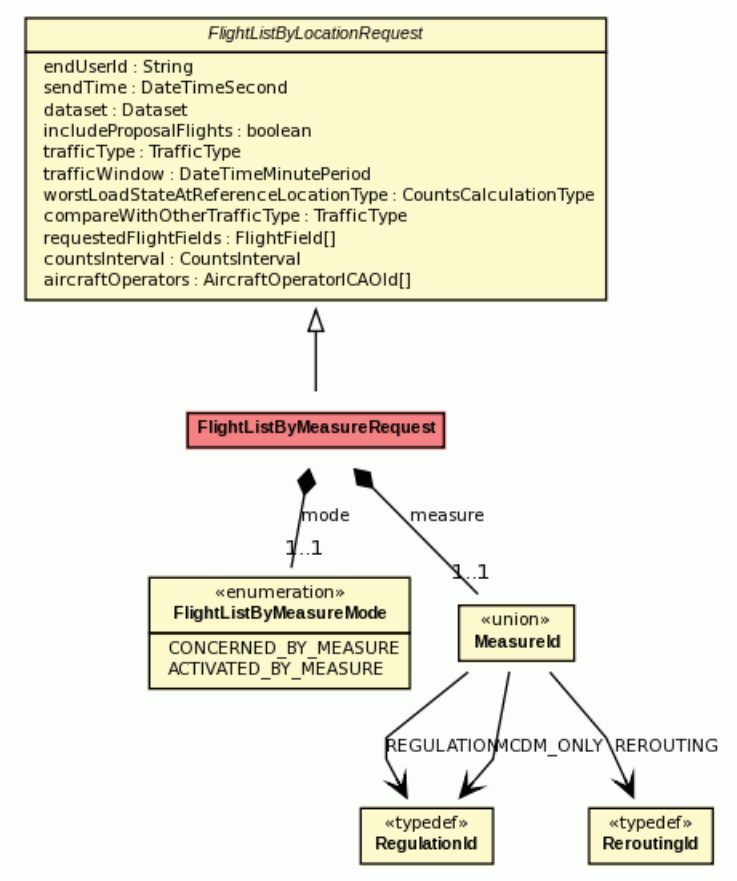

Figure 3.42. `FlightListByMeasureRequest` Class Diagram

(1) Request to query a flight list by measure (regulation/rerouting/MCDM-only measure). The reply only contains the flights that are concerned by that measure or alternatively only the flights that the measure has impacted. Note that the `FlightListByLocationRequest.traficWindow` is used to find the concerned regulation. Note that `FlightListByMeasureRequest` is subject to specific user authorization: it is enabled/authorized on test platforms, otherwise when specifically requested by the user.

(2) The logical AND operator applies between all the query fields described below and those inherited from FlightListByLocationRequest.

(3) Inherits from: `FlightListByLocationRequest`

(4) Attributes:

a) **`MeasureId` `measure`** *(Mandatory)*
Measure id.

b) **`FlightListByMeasureMode` `mode`** *(Mandatory)*

Indicates if the reply must contain the flights that are concerned by the given measure or the flights that the measure has impacted (measure activated).
For a regulation the concerned flights are those flights that use a regulation slot. However not all of them have an actual delay/have received a slot allocation message (typically exempted flights do not get regulated in a normal regulation (non-exceptional-conditions regulation). For a regulation, the flights that the measure has impacted (measure activated), are a subset of those flights: only those flights that did get a delay (can be 0 minutes) and have/will receive a SAM (Slot Allocation Message).
For a rerouting/MCDM-only measure, the concerned flights are those flights that cross the location/traffic volume during the period on the optional flow, while the the flights that the measure has impacted (measure activated), are a subset of those flights: only those flights that have been cherry picked for the rerouting/MCDM-only measure. Note that even if a flight has been cherry picked for a rerouting, it does not mean that the rerouting could find an alternate route/improvement (the result can be found inside the flight field: `FlightAtfcmMeasureLocation`).

3.3.10.3. `FlightListByMeasureReply`

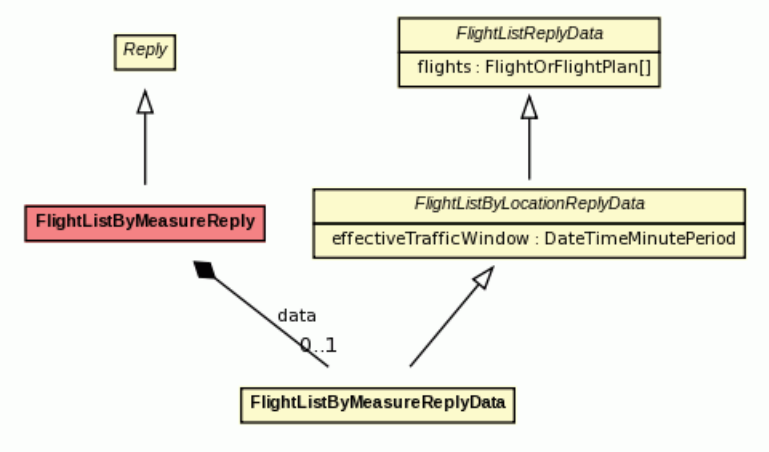

Figure 3.43. `FlightListByMeasureReply` Class Diagram

(1) Reply returned in response to FlightListByMeasureRequest.

(2) See FlightListReplyData.

(3) Inherits from: `Reply`

3.3.11. Flight List by Hotspot

3.3.11.1. SOAP

(1) The associated SOAP operation is:

```
FlightListByHotspotReply queryFlightsByHotspot(
    FlightListByHotspotRequest request
    )
```

3.3.11.2. `FlightListByHotspotRequest`

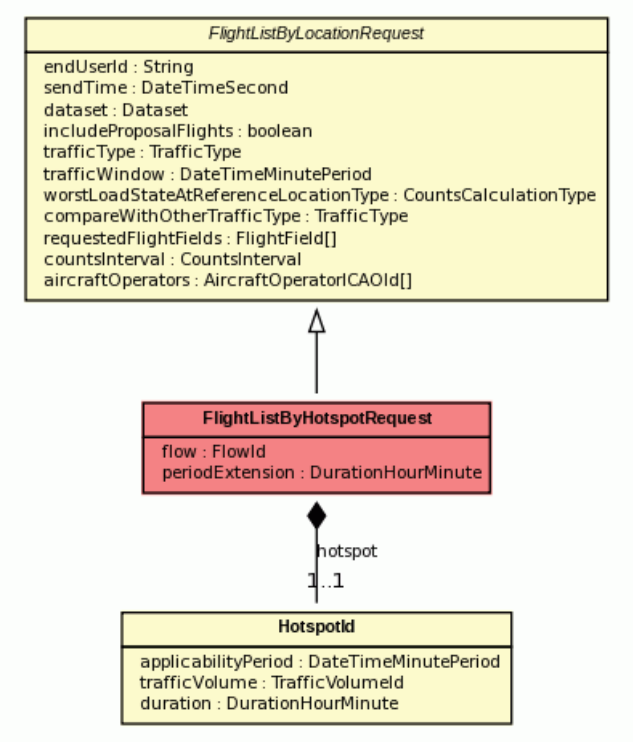

Figure 3.44. `FlightListByHotspotRequest` Class Diagram

(1) Request to query a flight list by hotspot. Note that hotspot related fields/services are trial related (STAM) fields : they are only accessible (authorized) during specific trials or on test platforms.

(2) A flight list by hotspot is always done on occupancy (as it is linked to a hotspot which is inherently linked to occupancy counts).

(3) The logical AND operator applies between all the query fields described below and those inherited from FlightListByLocationRequest.

(4) Inherits from: `FlightListByLocationRequest`

(5) Attributes:

a) **`HotspotId` `hotspot`** *(Mandatory)*
Hotspot id.

b) **`FlowId` `flow`** *(Optional)*
Id of the traffic flow.This allows to list only the flights for a specific flow.

c) **`DurationHourMinute` `periodExtension`** *(Mandatory)*
Period extension: For a hotspot flightlist, the FlightListRequest: trafficWindow is used to find the hotspot. However the HotspotId:applicabilityPeriod is used to query the flights. In some cases, the user wants to see a the flights around the real hotspot (typically to be able to

choose to what time over to cherry pick delay a flight). The periodExtension does exactly that: the `effectiveTrafficWindow` is the extended `HotspotId.applicabilityPeriod`(using the duration of the hotspot) and then extended some more (earlier and later) by the `periodExtension`.
Period extension: For a hotspot flightlist, the FlightListRequest: trafficWindow is not used. Instead the HotspotId:applicabilityPeriod is used to query the flights. In some cases, the user wants to see the flights around the real hotspot (typically to be able to choose to what time over to cherry pick delay a flight). The periodExtension does exactly that: the effectiveTrafficWindow is the extended `HotspotId.applicabilityPeriod` (using the duration of the hotspot) and then extended some more (earlier and later) by the periodExtension. For example: for a hotspot `applicabilityPeriod=[10:20, 10:50[` and hotspot duration=10 minutes and periodExtension=2 minutes (and for all OTMV/hotspot : step=1 minute), the `effectiveTrafficWindow` for the occupancy flightlist is `[10:18,11:01[`

3.3.11.3. `FlightListByHotspotReply`

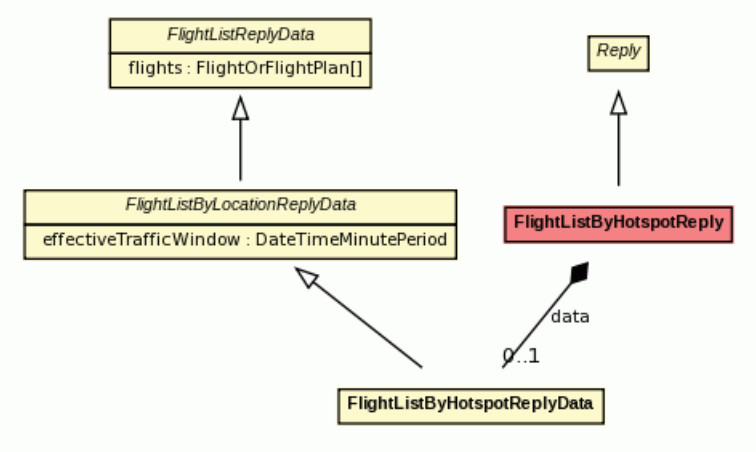

Figure 3.45. `FlightListByHotspotReply` Class Diagram

(1) Reply returned in response to FlightListByHotspotRequest.

(2) See FlightListReplyData.

(3) Inherits from: `Reply`

3.3.12. Flight Retrieval

3.3.12.1. SOAP

(1) The associated SOAP operation is:

```
FlightRetrievalReply retrieveFlight(
    FlightRetrievalRequest request
    )
```

3.3.12.2. `FlightRetrievalRequest`

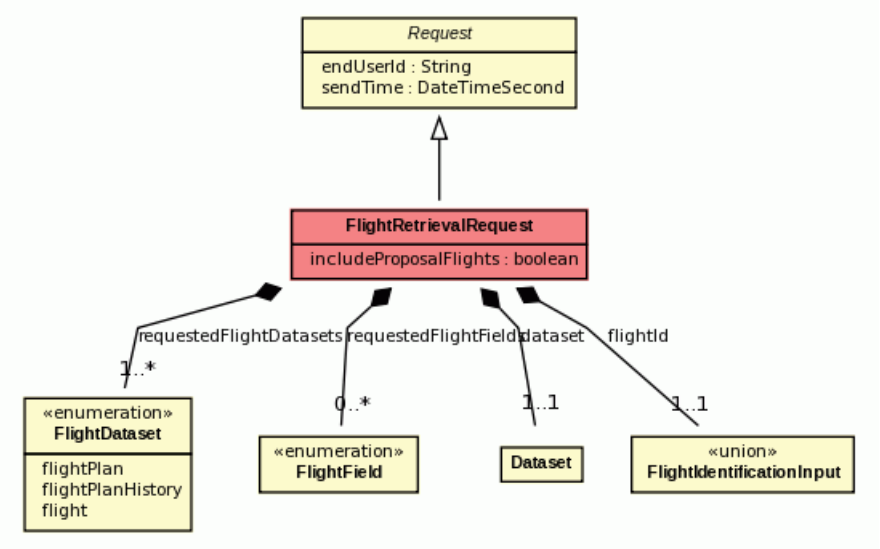

Figure 3.46. `FlightRetrievalRequest` Class Diagram

(1) Request to selectively retrieve all or part of the information regarding a single flight, i.e.:

a) The flight plan (FlightPlan), and/or

b) The flight plan history (FlightPlanHistory), and/or

c) The flight (Flight)

(2) These are specified using the `requestedFlightDatasets` request field.

(3) In addition, the flight fields (i.e. the attributes in Flight) are also selectively returned based on the caller's selection, expressed via `requestedFlightFields`, if they are available at NM.

(4) Note that the flight keys are always returned.

(5) REMARKS:

a) It is possible that according to the data set selected, a `FlightPlan` is returned and not a `Flight`. This can be for a temporary situation or for flight created with an EOBT in a far future (when EOBT-now > 22h).

b) For a CANCELLED `FlightPlan` NM returns:

i) A `FlightPlan` object when the `flightId.id` and a `FlightPlan` data set is specified and that until the `FlightPlan` object is archived.

ii) `OBJECT_NOT_FOUND` if the `flightId.keys` and a `FlightPlan` data set is specified.

iii) `OBJECT_NOT_FOUND` if the `flightId.keys` and a `Flight` data set is specified.

c) For a TERMINATED `Flight` NM returns:

i) A `FlightPlan` object when the `flightId.id` or `flightId.keys` and a `Flight-Plan` data set is specified and that until the `FlightPlan` object is archived.

ii) A `Flight` object if the `flightId.keys` and a `Flight` data set is specified and that until the `Flight` object is archived.

(6) Inherits from: `Request`

(7) Attributes:

a) **`Dataset` `dataset`** *(Mandatory)*
Dataset from which the flight list is requested.
Constraint: See `REQUESTED_FLIGHT_DATASETS_CAN_ONLY_CONTAIN_FLIGHT`

b) **`boolean` `includeProposalFlights`** *(Mandatory)*
If specified, the result will include also proposal flights.

c) **`FlightIdentificationInput` `flightId`** *(Mandatory)*
Identification of the requested flight plan.
The `flightId.id` or the `flightId.keys` can be used to retrieve `flightPlan` and `flightPlanHistory` data sets
Only `flightId.keys` can be used to retrieve `flight` data set.
Constraints:

i) See `ADES_OR_NONICAOADES_KEYS_MUST_BE_SET_FOR_FLIGHTPLAN_DATASET`

ii) See `KEYS_MUST_BE_PRESENT_IF_FLIGHT_IS_SPECIFIED_AS_REQUESTED_DATA-SET`

d) **`FlightDataset[]` `requestedFlightDatasets`** *(Mandatory)*
The reply returns only the requested datasets, and only if the requested datasets are available at NM. It can be for example that a flight plan is available but not the corresponding flight, or conversely.
Constraints:

i) Size must be comprised between 1 and ∞.

ii) See `ADES_OR_NONICAOADES_KEYS_MUST_BE_SET_FOR_FLIGHTPLAN_DATASET`

iii) See `CANNOT_CONTAIN_DUPLICATE_REQUESTED_DATASETS`

iv) See `KEYS_MUST_BE_PRESENT_IF_FLIGHT_IS_SPECIFIED_AS_REQUESTED_DATA-SET`

e) **`FlightField[]` `requestedFlightFields`** *(Optional)*
The reply returns only the requested attributes of the returned Flight, and only if the values of these requested fields are available at NM.

Cannot be null or empty or contain duplicates if the `flight` dataset is requested; must be null otherwise.
Constraints:

i) Size must be comprised between 0 and ∞.

ii) See CANNOT_CONTAIN_DUPLICATE_REQUESTED_FIELDS

iii) See REQUESTED_FIELD_NOT_ALLOWED_FOR_OPERATION

(8) Constraints:

a)

Name	KEYS_MUST_BE_PRESENT_IF_FLIGHT_IS_SPECIFIED_AS_REQUESTED_DATASET
Attributes	flightId, requestedFlightDatasets
Description	If 'FLIGHT' is specified in the requestedFlightDataSet - the flightId.keys cannot be null.

b)

Name	CANNOT_CONTAIN_DUPLICATE_REQUESTED_DATASETS
Attribute	requestedFlightDatasets
Description	Cannot contain duplicates.

c)

Name	CANNOT_CONTAIN_DUPLICATE_REQUESTED_FIELDS
Attribute	requestedFlightFields
Description	Cannot contain duplicates.

d)

Name	REQUESTED_FLIGHT_DATASETS_CAN_ONLY_CONTAIN_FLIGHT
Attribute	dataset
Description	if the dataset type is set as FORECAST or SIMULATION, then returnInvalidFilingSummaries needs to be set to false (otherwise an error is returned).

e)

Name	REQUESTED_FIELD_NOT_ALLOWED_FOR_OPERATION
Attribute	requestedFlightFields
Description	The fields worstLoadStateAtReferenceLocation, compareWithOtherTrafficType are not valid in the FlightRetrievalRequest.

f)

Name	ADES_OR_NONICAOADES_KEYS_MUST_BE_SET_FOR_FLIGHTPLAN_DATASET
Attributes	flightId, requestedFlightDatasets

Description	If `requestedFlightDatasets` contains `FlightDataset.flightPlan` or `FlightDataset.flightPlanHistory`, and if `flightId`'s choice is `keys`, then the attributes `flightId.keys.aerodromeOfDestination` and/or `flightId.keys.nonICAOAerodromeOfDestination` must be set.

3.3.12.3. `FlightRetrievalReply`

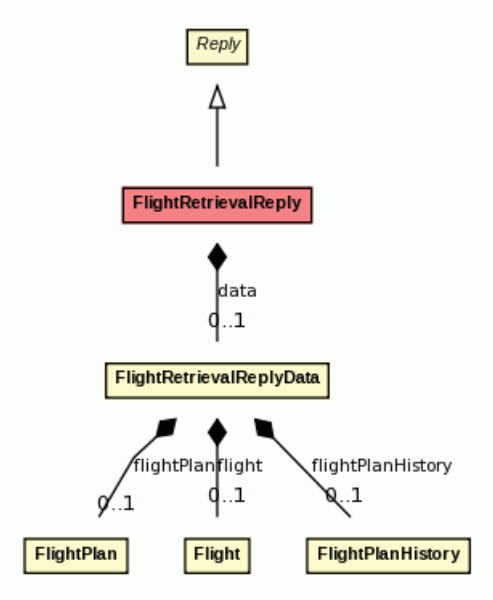

Figure 3.47. `FlightRetrievalReply` Class Diagram

(1) Reply returned in response to FlightRetrievalRequest.

(2) Inherits from: `Reply`

(3) Attributes:

a) **`FlightPlan` `flightPlan`** *(Optional)*
The flight plan, if requested.
Represents a flight plan with ICAO 2012 content.

b) **`FlightPlanHistory` `flightPlanHistory`** *(Optional)*
The flight plan history, if requested.

c) **`Flight` `flight`** *(Optional)*
The flight, if requested.
Represents a flight with ICAO 2012 content

3.3.13. Early DPI

3.3.13.1. SOAP

(1) The associated SOAP operation is:

```
EarlyDPIReply submitEarlyDPI(
    EarlyDPIRequest request
    )
```

3.3.13.2. `EarlyDPIRequest`

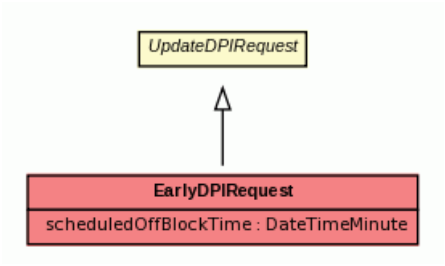

Figure 3.48. `EarlyDPIRequest` Class Diagram

(1) The Airport confirms to NMOC that an airport slot and flight plan for a particular flight have been correlated in accordance with local rules at the airport (A-CDM Mile stone 1).

(2) Detailed information regarding `Early DPI` messages can be found in the document DPI Implementation Guide section "E-DPI".

(3) Inherits from: `UpdateDPIRequest`

(4) Attributes:

a) **`DateTimeMinute` `scheduledOffBlockTime`** *(Optional)*
IATA schedule time of departure.
The prime originator is the Airline. It is the time that an aircraft is scheduled to depart.
For example: for passenger flights it is the time the passenger has on his ticket.
Acronym: SOBT.

3.3.13.3. `EarlyDPIReply`

(1) Reply returned in response to EarlyDPIRequest.

(2) Inherits from: `Reply`

3.3.14. Target DPI Target

3.3.14.1. SOAP

(1) The associated SOAP operation is:

```
TargetDPITargetReply submitTargetDPITarget(
    TargetDPITargetRequest request
    )
```

3.3.14.2. `TargetDPITargetRequest`

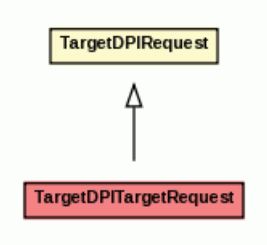

Figure 3.49. `TargetDPITargetRequest` Class Diagram

(1) <docbook:listitem>
`T-DPI-t` -- The T-DPI-t message must contain the Target Take-Off Time (TTOT) that takes into account all constraints from an AO and Handling Agent perspective.
</docbook:listitem>

(2) <docbook:listitem>
Detailed information regarding `Target DPI target` can be found in the document DPI Implementation Guide section "T-DPI-t.
</docbook:listitem>

(3) Inherits from: `TargetDPIRequest`

3.3.14.3. `TargetDPITargetReply`

(1) Reply returned in response to TargetDPITargetRequest.

(2) Inherits from: `Reply`

3.3.15. Target DPI Sequence

3.3.15.1. SOAP

(1) The associated SOAP operation is:

```
TargetDPISequencedReply submitTargetDPISequenced(
    TargetDPISequencedRequest request
    )
```

3.3.15.2. `TargetDPISequencedRequest`

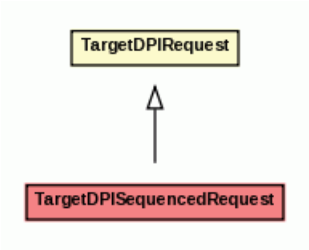

Figure 3.50. `TargetDPISequencedRequest` Class Diagram

(1) <docbook:listitem>
`T-DPI-s` -- The T-DPI-s contains the Take-Off-Time as calculated by the Pre-Departure Sequence. This Take-Off-Time (target take-off-time) is included in the TTOT-field.
</docbook:listitem>

(2) <docbook:listitem>
Detailed information regarding `Target DPI sequenced` messages can be found in the document DPI Implementation Guide section T-DPI-s.
</docbook:listitem>

(3) Inherits from: `TargetDPIRequest`

3.3.15.3. `TargetDPISequencedReply`

(1) Reply returned in response to TargetDPISequencedRequest.

(2) Inherits from: `Reply`

3.3.16. ATC DPI

3.3.16.1. SOAP

(1) The associated SOAP operation is:

```
ATCDPIReply submitAtcDPI(
    ATCDPIRequest request
    )
```

3.3.16.2. `ATCDPIRequest`

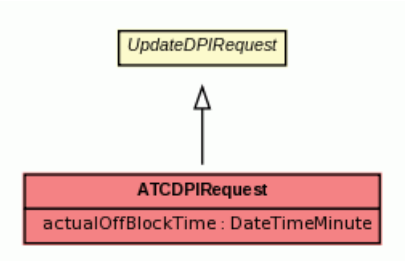

Figure 3.51. `ATCDPIRequest` Class Diagram

(1) The purpose of the A-DPI is to inform NM that the flight has off-blocked, i.e. the flight is "under ATC (or Apron) control" and taxiing to take-off.

(2) Detailed information regarding `ATC DPI` messages can be found in the document DPI Implementation Guide section "A-DPI".

(3) Inherits from: `UpdateDPIRequest`

(4) Attributes:

a) **`DateTimeMinute` `actualOffBlockTime`** *(Optional)*
Actual Off Block Time.
Prime originator is tower ATC. The actual date and time the aircraft has vacated the parking position (pushed back or on its own power).
Acronym: AOBT.

3.3.16.3. `ATCDPIReply`

(1) Reply returned in response to ATCDPIRequest.

(2) Inherits from: `Reply`

3.3.17. Cancel DPI

3.3.17.1. SOAP

(1) The associated SOAP operation is:

```
CancelDPIReply submitCancelDPI(
    CancelDPIRequest request
    )
```

3.3.17.2. `CancelDPIRequest`

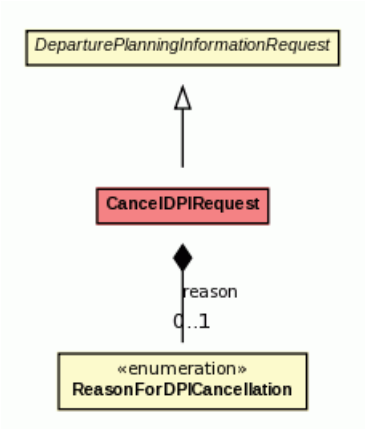

Figure 3.52. `CancelDPIRequest` Class Diagram

(1) The Airport informs NMOC that previously sent DPI information is no longer valid.

(2) Detailed information regarding `Cancel DPI` messages can be found in the document DPI Implementation Guide section "C-DPI".

(3) Inherits from: `DeparturePlanningInformationRequest`

(4) Attributes:

a) **`ReasonForDPICancellation` `reason`** *(Optional)*
Reason for requesting a `cancelDPI`.
Acronym: REASON.

3.3.17.3. `CancelDPIReply`

(1) Reply returned in response to CancelDPIRequest.

(2) Inherits from: `Reply`

3.4. FlightSafetyService Port Type

3.4.1. Overview

(1) The EC maintains a list of "green" third countries (understand, not in the EU territory). All aircraft operators that are SAFA-compliant may fly from all airports of these green countries. For airports that are not in the green country list neither in the EU territory, aircraft operators must get accreditations for their departure airports. So an ACC3 accreditation applies to a (AO, AD) pair. The EC requests Eurocontrol to operate an alerting service when a flight plan is submitted by a non-accredited (AO, AD) pair - also in case of diversion (airborne). In this context, the FlightSafety service is limited to setting the full accreditation list, typically sent to us once a day:

a) ACC3AccreditationListReplacementRequest / ACC3AccreditationListReplacementReply

3.4.2. Flight Safety

3.4.2.1. SOAP

(1) The associated SOAP operation is:

```
ACC3AccreditationListReplacementReply replaceACC3AccreditationList(
    ACC3AccreditationListReplacementRequest request
    )
```

3.4.2.2. `ACC3AccreditationListReplacementRequest`

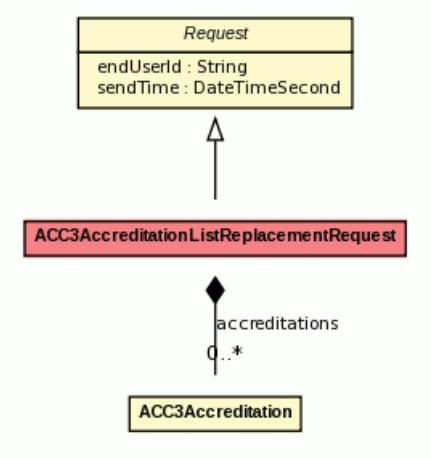

Figure 3.53. `ACC3AccreditationListReplacementRequest` Class Diagram

(1) An ACC3 accreditation list replacement request.

(2) Inherits from: `Request`

(3) Attributes:

a) **`ACC3Accreditation[] accreditations`** *(Mandatory)*
The accreditations.
Constraint: Size must be comprised between 0 and ∞.

3.4.2.3. `ACC3AccreditationListReplacementReply`

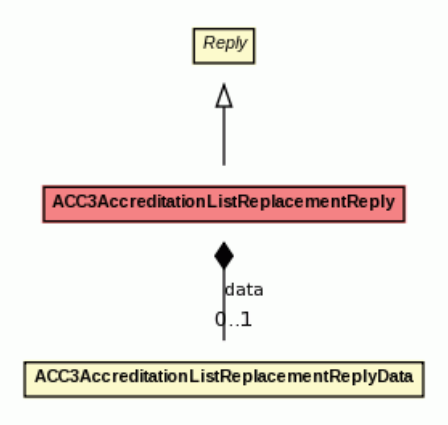

Figure 3.54. `ACC3AccreditationListReplacementReply` Class Diagram

(1) An ACC3 accreditation list replacement reply.

(2) Inherits from: `Reply`

Chapter 4. Data Types

4.1. ACC3Accreditation

(1) An ACC3 accreditation applies to an aircraft operator departing from an aerodrome. The whole accreditation list replacement is a single transaction: it fully succeeds or fully fails.

(2) Attributes:

a) **ACC3AccreditationId id** *(Mandatory)*
Unique id of the accreditation - unique within the accreditation list that applies at any point in time.

b) **AerodromeIATAOrICAOId adId** *(Mandatory)*
Contains either the IATA id or the ICAO id of the departure aerodrome to which the accreditation applies.

c) **AircraftOperatorIATAOrICAOId aoId** *(Mandatory)*
Contains either the IATA id or the ICAO id of the aircraft operator to which the accreditation applies.

(3) Used by: ACC3AccreditationListReplacementRequest.

4.2. typedef<string> ACC3AccreditationId

(1) Uniquely identifies an ACC3 accreditation.

(2) Pattern: `(ALPHA|DIGIT|/|_|*){0,100}`

(3) Used by: ACC3Accreditation.

4.3. Aerodrome

(1) Describes an aerodrome in the flight plan context.

(2) Choices:

a) **AerodromeICAOId icaoId**
The ICAO id of this aerodrome.

b) **OtherAerodromeDesignation otherDesignation**
Name and location of the aerodrome if the ICAO id is not provided for this aerodrome.

(3) Used by: FlightPlan, AerodromesOfDestination, FlightArrivalRequest.

4.4. AerodromeDAL

(1) Attributes:

a) **AerodromeICAOId aerodrome** *(Mandatory)*

b) **DistanceM cumulativeDistance** *(Mandatory)*

(2) Used by: DistanceAtLocation.

4.5. AerodromeIdAndRole

(1) Represents an aerodrome (e.g., EBBR) and a role which may be DEPARTURE, ARRIVAL or BOTH.

(2) Attributes:

a) **AerodromeIATAOrICAOId id** *(Mandatory)*
IATA (3 characters) or ICAO (4 characters) identifier of the aerodrome.

b) **AerodromeRole role** *(Mandatory)*
Role of the aerodrome.

4.6. typedef<string> AerodromeName_DataType

(1) The name of the aerodrome.

(2) Pattern: ANY{1,50}

(3) Used by: OtherAerodromeDesignation.

4.7. typedef<string> AerodromeNameLocationDescription_DataType

(1) Description of name and location of the aerodrome.

(2) Pattern: ANY{1,100}

(3) Used by: AlternateAerodrome.

4.8. <<enumeration>> AerodromeRole

(1) Used to specified if an aerodrome is meant as of departure, arrival or both.

(2) Values:

a) **ARRIVAL**

b) **BOTH**

c) **DEPARTURE**

(3) Used by: FlightListByAerodromeSetRequest, TrafficCountsByAerodromeSetRequest, FlightListByAerodromeRequest, AerodromeIdAndRole, TrafficCountsByAerodromeRequest.

4.9. AerodromesOfDestination

(1) Represents an aerodrome of destination in a flight plan, together with its alternates.

(2) Attributes:

a) **Aerodrome aerodromeOfDestination** *(Mandatory)*
Aerodrome of destination.

b) **AlternateAerodrome alternate1** *(Optional)*
First alternate aerodrome of destination.
Constraint: See ALTERNATE2_MUST_BE_NULL_IF_ALTERNATE1_IS_NULL

c) **AlternateAerodrome alternate2** *(Optional)*
Second alternate aerodrome of destination.
Constraint: See ALTERNATE2_MUST_BE_NULL_IF_ALTERNATE1_IS_NULL

(3) Constraint:

a)

Name	ALTERNATE2_MUST_BE_NULL_IF_ALTERNATE1_IS_NULL
Attributes	alternate1, alternate2
Description	alternate2 must be null if alternate1 is null.

(4) Used by: FlightPlan.

4.10. typedef<string> AircraftIATAId

(1) Concatenation of the carrierIdentification, the iataFlightNumber and optionally a suffix.

(2) a) carrierIdentification - Code of the Aircraft Operator of the identified flight as defined in the Schedule [AIDX, UFI].
Examples: - BA

b) iataFlightNumber - IATA flight number of the identified flight as defined in the Schedule [AIDX, UFI].
Examples: - 066

c) suffix - Suffix of the IATA flight number as defined in the Schedule [AIDX, UFI].
Examples: - Z

(3) Pattern: (UALPHA|DIGIT){2}(UALPHA|DIGIT|*){0,1}DIGIT{3,4}UALPHA{0,1}

(4) Used by: FlightIATAId.

4.11. typedef<string> AircraftICAOId

(1) ICAO aircraft identification as defined in ICAO doc 4444 Appendix 2, section 2.

(2) Pattern: `(ALPHA|DIGIT){2,7}`

(3) Used by: AirportPlanningInformationRequest, AircraftIdentification.

4.12. AircraftIdentification

(1) Aircraft identification: groups the ICAO aircraft id (designator of the aircraft operator followed by the flight identifier) possibly completed with other related data, e.g. registration mark and ICAO aircraft address.

(2) Attributes:

a) **AircraftICAOId aircraftId** *(Contextual)*
Aircraft ICAO id.
Presence:

i) Optional in ExtendedFlightPlanValidationRequest, FlightPlanValidationRequest, RoutingAssistanceRequest

ii) Mandatory otherwise.

b) **AircraftRegistrationMark registrationMark** *(Optional)*
Registration mark of the aircraft.

c) **ICAOAircraftAddress aircraftAddress** *(Contextual)*
ICAO address of the aircraft, formerly known as mode S address.
Presence:

i) Ignored in ExtendedFlightPlanValidationRequest, FlightPlanValidationRequest, RoutingAssistanceRequest

ii) Optional otherwise.

d) **SSRInfo ssrInfo** *(Contextual)*
SSR code assigned to the aircraft by the ATS and its transmission mode.
Presence:

i) Ignored in ExtendedFlightPlanValidationRequest, FlightPlanValidationRequest, RoutingAssistanceRequest

ii) Optional otherwise.

(3) Used by: FlightPlan.

4.13. AircraftIdentificationUpdate

(1) See AircraftIdentification.

(2) Attributes:

a) **AircraftRegistrationMark registrationMark** *(Optional)*

b) **ICAOAircraftAddress aircraftAddress** *(Optional)*

c) **SSRInfo ssrInfo** *(Optional)*

(3) Used by: FlightPlanUpdate.

4.14. typedef<string> AircraftOperatorIATAId

(1) IATA id of an aircraft operator.

(2) Examples:AF, BA, AAL, A3, 9P, 2V*,...

(3) Pattern: `(UALPHA|DIGIT){2}(UALPHA|DIGIT|*){0,1}`

4.15. typedef<string> AircraftOperatorIATAOrICAOId

(1) IATA or ICAO id of an aircraft operator.

(2) Pattern: `UALPHA{3}|(UALPHA|DIGIT){2}(UALPHA|DIGIT|*){0,1}`

(3) Used by: ACC3Accreditation.

4.16. typedef<string> AircraftOperatorICAOId

(1) Examples:AFR, BAW, TAP,...

(2) Pattern: `ALPHA{3}`

(3) Used by: Flight, FlightListByLocationRequest, TrafficCountsByAircraftOperatorRequest.

4.17. typedef<string> AircraftOperatorName_DataType

(1) Examples:XJC XCLUSIVE JET CHARTER AND MANAGEMENT 442380696992, ZENITH AVIATION, TURKISH AIR FORCE,...

(2) Pattern: `(UALPHA|DIGIT|'|\(|\)|+|,|=|?|.|/|:|WHITESPACE){0,50}`

(3) Used by: OtherInformation.

4.18. <<enumeration>> AircraftPerformanceCategory

(1) Aircraft performance categories as defined in the Procedures for Air Navigation Services - Aircraft Operations (PANS-OPS, Doc 8168).

(2) Values:

Value	Description
CAT_A	less than 169 km/h indicated airspeed (IAS)
CAT_B	169 km/h or more but less than 224 km/h IAS
CAT_C	224 km/h or more but less than 261 km/h IAS
CAT_D	261 km/h or more but less than 307 km/h IAS
CAT_E	307 km/h or more but less than 391 km/h IAS
CAT_H	Helicopters

Table 4.1. <<enumeration>> AircraftPerformanceCategory

(3) Used by: OtherInformation.

4.19. typedef<string> AircraftRegistrationMark

(1) Examples:011, 0216, GEUYC, Z BAM, YV2726, QH3023T, HT21A 1 ...

(2) Pattern: `(ALPHA|DIGIT|'|+|=|?|.|/|:| ){1,50}`

(3) Used by: CDMProvisionalInfo, UpdateDPIRequest, Flight, AircraftIdentificationUpdate, CDMInfo, AircraftIdentification.

4.20. AircraftType

(1) Describes an aircraft type in the flight plan context.

(2) Choices:

a) **AircraftTypeICAOId icaoId**
The ICAO id of this aircraft type.

b) **OtherAircraftTypeDesignation_DataType otherDesignation**
Name of the aircraft type if no ICAO id exists for this aircraft type (ICAO TYP/ field).

(3) Used by: CDMProvisionalInfo, FlightPlan, FlightPlanUpdate, CDMInfo.

4.21. typedef<string> AircraftTypeICAOId

(1) Examples:A7, A50, A310, A30B, AA1, AC5A,...

(2) Pattern: `ALPHA{1}(ALPHA|DIGIT){1,3}`

(3) Used by: UpdateDPIRequest, Flight, AircraftType.

4.22. AirFiledData

(1) Estimate data for an air-filed (AFIL) flight plan, i.e. a point, the joining flight level and the estimate date/time at this point. Note that the flight level indicated is the level at which the flight has been cleared to join controlled airspace over the point indicated: it does not have to be the same as the requested flight level.

(2) Attributes:

a) **AtsUnitId_DataType atsUnitId** *(Optional)*
ICAO id of the ATS unit from which supplementary flight plan data can be obtained.

b) **ICAOPoint startingPoint** *(Mandatory)*
Starting point.

c) **FlightLevel clearedLevel** *(Mandatory)*
Level at which the aircraft has been cleared to join controlled airspace over the given point.

d) **DateTimeSecond estimatedTimeOver** *(Mandatory)*
Estimated date/time over the given point.

(3) Used by: FlightPlan.

4.23. <<abstract>> AirportPlanningInformationRequest

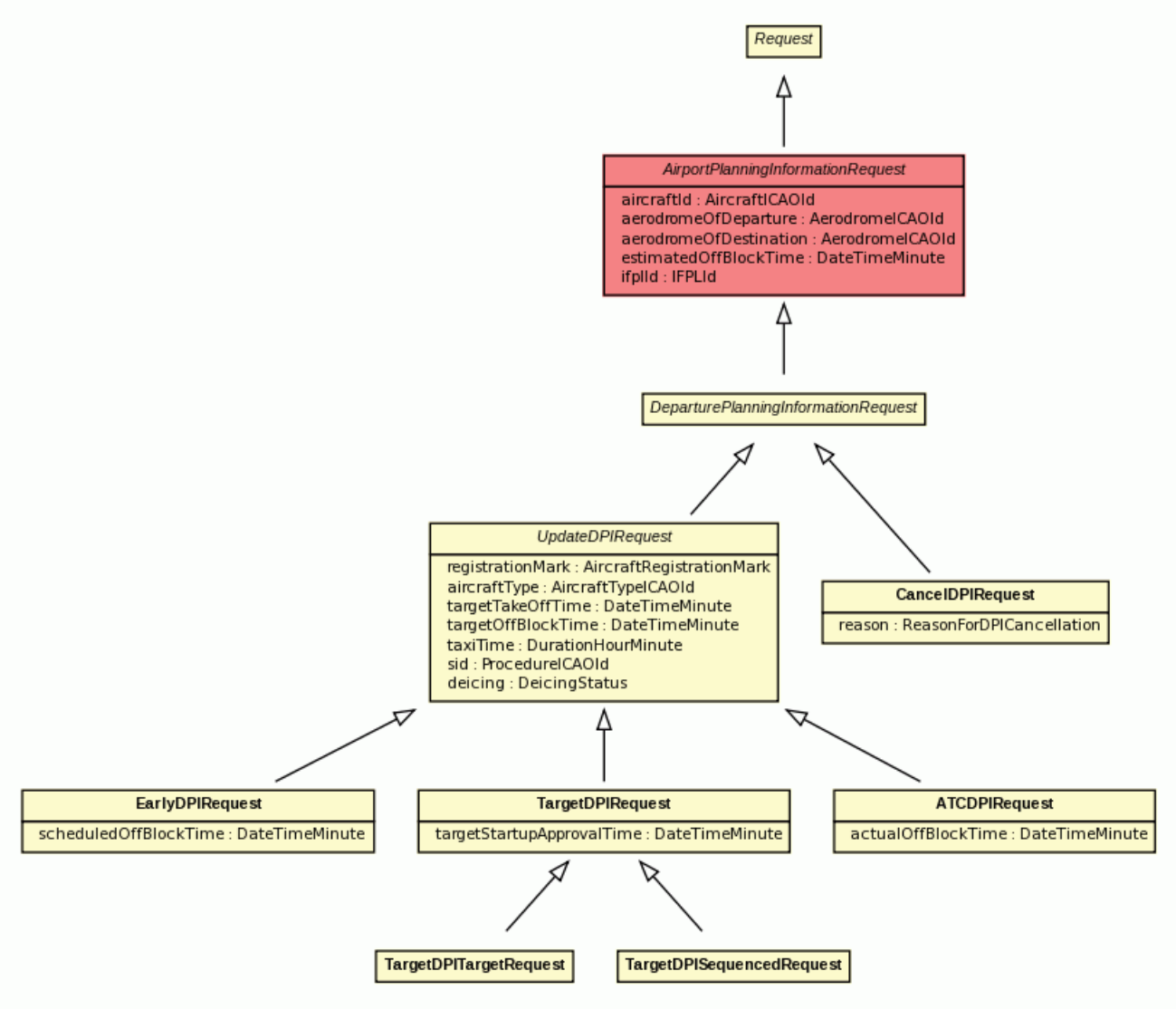

Figure 4.1. <<abstract>> AirportPlanningInformationRequest Class Diagram

(1) Base class of all Airport Planning Information requests.

(2) Inherits from: Request.

(3) Attributes:

a) **AircraftICAOId aircraftId** *(Mandatory)*
Aircraft ICAO id.

b) **AerodromeICAOId aerodromeOfDeparture** *(Mandatory)*
ICAO id of the aerodrome of departure.

c) **AerodromeICAOId aerodromeOfDestination** *(Mandatory)*
ICAO id of the aerodrome of destination.

d) **DateTimeMinute estimatedOffBlockTime** *(Mandatory)*
Estimated off-block date/time.

e) **IFPLId ifplId** *(Optional)*
Unique IFPL identifier as assigned by IFPS. Example: AA12345678.

(4) Extended by: DeparturePlanningInformationRequest.

4.24. AlternateAerodrome

(1) An alternate aerodrome. It is represented as either an ICAO aerodrome code or a free text description of the aerodrome if no ICAO code exist (ZZZZ used in ICAO message).

(2) Choices:

a) **AerodromeICAOId icaoId**
The ICAO id of this aerodrome.

b) **AerodromeNameLocationDescription_DataType nameLocationDescription**
Description of name and location of the aerodrome if the ICAO id is not provided for this aerodrome.

(3) Used by: FlightPlanUpdate, AerodromesOfDestination.

4.25. typedef<string> AlternateAerodrome_DataType

(1) Aerodromes where the aircraft may land in case of emergency along the route.

(2) Pattern: `ANY{1,100}`

(3) Used by: FlightPlan, FlightPlanUpdate.

4.26. <<enumeration>> ATFMMessageType

(1) Enumerates ATFM message types.

(2) Values:

a) **DES**
DE-Suspension message
NM sends a DES when a flight not subject to ATFCM restrictions is de-suspended.

b) **ERR**
ERRor message
NM sends an ERR message when a message is received where:

i) Its syntax is incorrect and, therefore, cannot be processed or

ii) The message or a part of the message is not relevant.

c) **FCM**
Flight Confirmation Message.
when:

i) An AO indicates to ETFMS the RVR capability of a flight with an EOBT in the future.Flight Confirmation Message.

ii) An AO indicates to ETFMS that a flight with an EOBT in the future is now confirmed for the regulation(s) provided in this FCM.

iii) An AO indicates to ETFMS that a flight with an EOBT in the future is now confirmed for the regulation(s) provided in this FCM.

d) **FLS**
FLight Suspension message
Flight suspension until further notice. In case of:

i) Aerodrome closure.

ii) Runway Visual Range (RVR) unknown.

iii) Reception of an SMM message.

iv) Not reported as airborne.

v) Suspended by departure airport (cancel DPI message).

e) **FUM**
Flight Update Message.
The FUM provides the airport of destination with the estimated landing time (ELDT). It also informs about the status of the flight (e.g. received a-DPI, Airborne,ATFM status...).

f) **REA**
REAdy message.
The REA message can only be sent by ATC following a request from AO.
AO may ask ATC to send REA in 2 situations:

i) The flight is ready to depart before the EOBT (maximum 30 minutes before).

ii) The flight is ready to depart before its CTOT.

g) **RFI**
Ready For Improvement message.
The RFI message can be sent by the AO in order to receive improvements directly with an SRM.

h) **RJT**
Rerouteing reJecTion (RJT) message
An RJT is a negative response to a Rerouteing Proposal (RRP) message.

i) **RRN**
ReRouteing Notification message
The RRN message is issued in case of an acceptance of the rerouteing with option 'CNL original FPL', book slot and flight plan refile by the AO via SITA/AFTN.

j) **RRP**
ReRouteing Proposal message
A sudden deterioration across the network would certainly be noticed when one of the ACCs reduces capacity resulting in excessive delays for example. ATFCM staff shall assess the situation before any decision is made. Assessment would include the best and worst case scenarios with alternatives to both. RRP will be one of the solutions to mitigate potential delays.ReRouteing Proposal message

k) **SAM**
Slot Allocation Message.
A SAM is sent to AOs/ATS any time a flight becomes regulated (new flight entering the system, new period of regulation in the system, in response to an FCM or CHG providing new RVR after a suspension) but at the earliest 2 hours before the last received EOBT.

l) **SIP**
Slot Improvement Proposal message.
A SIP message is sent to the AO by NM for a flight not being in an RFI status to propose a new take-off time if it is possible to improve the existing CTOT by a significant amount.

m) **SLC**
SLot requirement Cancellation message
An SLC is sent to AOs/ATS to advise that a flight which has received a CTOT is no longer subject to an ATFCM restriction. It may be due to the change in parameters of an existing restriction or its cancellation, or to the reception of a message from AOs such as DLA, CHG, and FCM.

n) **SMM**
Slot Missed Message.
An SMM is sent when the last received CTOT issued cannot be met and a new EOBT is NOT known.

o) **SPA**
Slot improvement Proposal Acceptance (SPA) message
An SPA is a positive response to a SIP which is received from NM. The AO will send an SPA if the proposed NEWCTOT in the SIP is acceptable.

p) **SRJ**
Slot improvement proposal ReJection (SRJ) message.
An SRJ is a negative response to a SIP received from NM. The AO will send an SRJ if they are unable to accept the proposed improvement.

q) **SRM**
Slot Revision Message.
An SRM may be sent by NM:

i) To notify all concerned of either a significant change (>5') to the original CTOT or a modification of the most penalising regulation or both.
Such changes are due to circumstances unrelated to the flight e.g. the introduction of a new restriction or a change to the parameters of an existing restriction.
By default, only flights in an RFI status or in a Ready (REA) status are considered for improvement but if the situation requires it, the NM Flow Controllers are able to let all flights, including those in SWM status, be considered for improvement.

ii) In response to a DLA or CHG when the current CTOT is no longer compliant with the new information.

iii) To notify all concerned of a routine improvement of the CTOT by the revision process for a flight in an RFI status or in a Ready (REA) situation.

iv) In response to a valid SPA to notify all concerned of the improvement of the CTOT.

r) **SWM**
SIP Wanted Message.
The SWM allows the flight to receive a SIP when there is a possibility to improve the flight.

s) **UNK**
Unknown message.

(3) Used by: Flight.

4.27. typedef<string> AtsUnitId_DataType

(1) ICAO id of the ATS unit from which supplementary flight plan data can be obtained.

(2) Pattern: `ANY{1,50}`

(3) Used by: AirFiledData.

4.28. BasicTrajectoryData

(1) The `BasicTrajectoryData` groups together information helping NM calculating the trajectory as close as possible to the trajectory calculated by the AO's. This is an alternative to the exchange of the full 4D Trajectory between AO's and NM. This full 4D trajectory exchange is currently (2012-2013) under validation within SESAR projects.

(2) Attributes:

a) **`WeightKg takeOffWeight`** *(Optional)*
The weight of the aircraft at take-off.

b) **`Relative4DPoint[] topOfClimb`** *(Optional)*
4D points where the requested level (RFL's) are estimated to be reached. These `topOfClimb` can only be given at RFL's being present in the flight plan route description to indicate the end of a climb to reach these RFL's.
Constraint: Size must be comprised between 0 and ∞.

c) **`Relative4DPoint[] topOfDescent`** *(Optional)*
4D points where the requested level (RFL's) are estimated to be left in order to descent either to the next RFL or to the destination. These `topOfDescent` can only be given at RFL's being present in the flight plan route description.
Constraint: Size must be comprised between 0 and ∞.

d) **`Relative4DPoint[] bottomOfClimb`** *(Optional)*
Transition 4D point from cruise phase to climb phase.
Constraint: Size must be comprised between 0 and ∞.

e) **`Relative4DPoint[] bottomOfDescent`** *(Optional)*
Transition 4D point from descent phase to cruise phase.
Constraint: Size must be comprised between 0 and ∞.

f) **`DistanceAtLocation distanceAtLocationInfo`** *(Optional)*

(3) Used by: FlightPlanCreationRequest, FlightPlanUpdateRequest.

4.29. CDM

(1) CDM (Collaborative Decision Making) systems located at airports provide DPI (Departure Planning Information) messages. Those DPI messages provide the NM system with more accurate information regarding the progression of the flights towards their take-off (taxi-time, target take-off time and departure procedure).

(2) Detailed information regarding CDM and DPI can be found in the documents DPI Implementation Guide and European Airport CDM .

(3) Attributes:

a) **`CDMStatus status`** *(Mandatory)*
Last known CDM status.

b) **`DepartureAirportType AirportType`** *(Mandatory)*
Departure Airport Type.

c) **`CDMProvisionalInfo provisionalInfo`** *(Optional)*
CDM provisional information, if available.

d) **`CDMInfo info`** *(Optional)*
CDM information, if available.

(4) Used by: Flight.

4.30. CDMInfo

(1) CDM information regarding the progression of the flights towards their take-off (taxi-time, target take-off time and departure procedure).

(2) Detailed information regarding CDM and DPI can be found in the documents DPI Implementation Guide and European Airport CDM .

(3) Attributes:

a) **DateTimeMinute earlyTargetTakeOffTime** *(Optional)*
Corresponds to the early target take-off time provided in an EARLY DPI message (E-DPI).

b) **DateTimeMinute aoTargetTakeOffTime** *(Optional)*
Corresponds to the aircraft operator target take-off time provided in a T-DPI message (T-DPI_t).

c) **DateTimeMinute atcTargetTakeOffTime** *(Optional)*
Corresponds to the ATC take-off time provided in an ATC DPI message (A-DPI).

d) **DateTimeMinute sequencedTargetTakeOffTime** *(Optional)*
Corresponds to the sequenced target take-off time provided in a sequenced DPI message (T-DPI_s).

e) **DurationHourMinuteSecond taxiTime** *(Optional)*
Corresponds to the latest taxi time received in a DPI message.
Note that this might not be the same as the taxi time of any of the FTFM and RTFM and CTFM flight profile.

f) **boolean offBlockTimeDiscrepancy** *(Mandatory)*
Indicates if there exists a significant difference between the filed off-block time and the off-block time that NM possibly received through DPI messages.

g) **ProcedureICAOId departureProcedure** *(Optional)*
Corresponds to the latest departure procedure received in a DPI message.

h) **boolean aircraftTypeDiscrepancy** *(Optional)*
Indicates if there exists a difference between the filed aircraft type and the aircraft type that NM possibly received through DPI messages.
Optional: not present if `aircraftType` is null.

i) **AircraftType aircraftType** *(Optional)*
Corresponds to the latest aircraft type received in a DPI message.
Cannot be null if `aircraftTypeDiscrepancy` is true.

j) **AircraftRegistrationMark registrationMark** *(Optional)*
Corresponds to the latest registration mark received in a DPI message. It will not exceed 7 characters length as only one registration mark is passed (max length of the type is 50).

k) **boolean registrationMarkDiscrepancy** *(Optional)*
Indicates if there exists a difference between the filed registration mark and the registration mark that NM possibly received through DPI messages.
Optional: not present if `registrationMark` is null.

l) **DurationHourMinute noSlotBefore** *(Optional)*
Minimum take-off time computed by the CDM system, taking into account the calculated take-off time sent by NM. This value is then the target take-off time sent to NM in a T-DPI message with DPI status "S". For subsequent slot re-computation, NM is requested to search a slot from this value.

m) **DepartureStatus departureStatus** *(Mandatory)*
Indicates the currently known departing status of the flight.

n) **DateTimeMinute targetOffBlockTime** *(Optional)*
Target Off-Block Time (TOBT) that is received from the CDM Airport.

o) **DateTimeMinute targetStartupApprovalTime** *(Optional)*
Target Start-up Approval Time (TSAT) that is received from the CDM Airport.

(4) Used by: CDM.

4.31. CDMProvisionalInfo

(1) Provides information received in a provisional target DPI message (T-DPI_p). This provisional information is a trial from the airport and is not applied to the flight. A what-if trial is performed to indicate what would be the CASA result if the T-DPI_p is confirmed.

(2) Detailed information regarding CDM and DPI can be found in the documents DPI Implementation Guide and European Airport CDM .

(3) Attributes:

a) **DateTimeMinute timestamp** *(Optional)*
Reception time of the latest T-DPI_p message.

b) **DateTimeMinute aoTargetTakeOffTime** *(Optional)*
Corresponds to the aircraft operator target take-off time provided in a T-DPI message (T-DPI_t).

c) **DurationHourMinuteSecond taxiTime** *(Optional)*
Taxi time provided in the T-DPI_p message.

d) **ProcedureICAOId departureProcedure** *(Optional)*
Departure procedure provided in the T-DPI_p message.

e) **AircraftType aircraftType** *(Optional)*
Aircraft type provided in the T-DPI_p message.

f) **AircraftRegistrationMark registrationMark** *(Optional)*
Registration mark provided in the T-DPI_p message. It will not exceed 7 characters length as only one registration mark is passed (max length of the type is 50).

g) **DepartureStatus departureStatus** *(Mandatory)*
De-icing mode provided in the T-DPI_p message.

h) **RegulationId mostPenalisingRegulation** *(Optional)*
Potential most penalising regulation found during the what-if trial with the provisional info.

i) **DateTimeMinute possibleCFMUTakeOffTime** *(Optional)*
Possible take-off time calculated by NM considering the most penalising regulation (if any) found during the what-if trial with the provisional info.

j) **SuspensionStatus suspensionStatus** *(Mandatory)*
Suspension status found during the what-if trial with the provisional info.

(4) Used by: CDM.

4.32. <<enumeration>> CDMStatus

(1) Enumerates the possible CDM status values.

(2) Values:

a) **ACTUAL_OFFBLOCK**
ATC DPI received from a CDM airport or from an advanced ATC/TWR.

b) **DEPARTING_FROM_CDM_AIRPORT**
no DPI received yet from a CDM airport or from an advanced ATC/TWR.

c) **DEPARTING_FROM_STANDARD_AIRPORT**

d) **ESTIMATED**
early DPI received from a CDM airport or from an advanced ATC/TWR.

e) **PRE_SEQUENCED**
target start-up approval DPI (T-DPI-s) received from a CDM airport and compliant with the departure tolerance window.

f) **TARGETED**
aircraft operator target DPI (T-DPI-t) received from a CDM airport.

(3) Used by: CDM.

4.33. <<enumeration>> CfmuFlightType

(1) Indicates state and/or origin of flight in the NM system.

(2) Values:

Value	Description
ACT	Flight is ATC activated.
IFPL	Flight created from a flight plan filed to IFPS.

Value	Description
MFD	Mini-flight created for the usage of CCAMS when the flight is unknown to NM.
PREDICTED_FLIGHT	Predicted flight data.
RPL	Flight created from a repetitive flight plan prior to the generation of the individual flight plan.
TACT_ACTIVATED	Flight activated by NM.
TERMINATED	Flight is terminated.

Table 4.2. <<enumeration>> CfmuFlightType

(3) Used by: Flight.

4.34. typedef<string> Colour_DataType

(1) Colour of the dinghies carried by the aircraft.

(2) Pattern: `(ALPHA|DIGIT){1,51}`

(3) Used by: Dinghies.

4.35. <<enumeration>> CTOTLimitReason

(1) Possible exceptional reasons that may affect the CTOT allocation of a flight.

(2) When a flight is regulated, its CTOT may delay the flight into one or more Flight Plan time dependent constraints (e.g. RAD restrictions, CDR2), therefore violating route and/or airspace restrictions.

(3) NM takes into account the route and airspace restrictions when a CTOT is allocated so that violations are avoided. The CASA algorithm takes into account the maximum delay to which the flight could be subject before it violates a route or an airspace restriction. When a slot is allocated by CASA `ctotLimitReason` attribute of the flight will be set to indicate if the delay was limited by the `Last Valid EOBT`. The `Last Valid EOBT` is the last valid EOBT acceptable for a flight before triggering Flight Plan processing errors.

(4) The following cases are foreseen:

(5) a) There is no `Last Valid EOBT` for the flight so the slot time has not been limited.

b) The flight's CTOT has been forced.

c) The delay of the flight is limited to the `Last Valid EOBT`.

d) The delay of the flight was limited firstly by the `Last Valid EOBT` but also by a yet more restrictive zero-rate or suspending regulation measure.

(6) Values:

a) **FORCED_BY_CHAMAN**
The CTOT has been forced by Chaman.

b) **FORCED_BY_NMOC**
The CTOT has been forced by NMOC.

c) **FORCED_BY_STAM_MEASURE**
The CTOT has been forced by a STAM measure.

d) **FORCED_BY_TOWER**
The CTOT has been forced by the tower.

e) **LIMITED_BY_VIOLATION**
CASA has based the CTOT on the `Last Valid EOBT` to avoid violations.

f) **LIMITED_BY_VIOLATION_THEN_ZERO_RATE_OR_RVR**
The same as for the `LIMITED_BY_VIOLATION` but, because this CTOT would give overlap with a (non-suspending) zero-rate or RVR subperiod, CASA has limited the CTOT further to the start time of the zero-rate or RVR subperiod.

g) **SLOT_TIME_NOT_LIMITED**
There is no forced CTOT neither limitations by the `Last Valid EOBT` for the flight so the slot time has not been limited.

h) **WAS_FORCED_BY_NMOC**
The CTOT was manually forced by NMOC but is no longer forced.

(7) Used by: Flight.

4.36. DatalinkCapabilities

(1) Describes the data-link applications or capabilities of an aircraft.

(2) Attributes:

a) **DataLinkCapabilities_DataType value** *(Optional)*

(3) Used by: OtherInformation.

4.37. typedef<string> DataLinkCapabilities_DataType

(1) Describes the data-link applications or capabilities of an aircraft.

(2) Pattern: `ANY{1,50}`

(3) Used by: DatalinkCapabilities.

4.38. <<enumeration>> DeicingStatus

(1) The de-icing process is planned,in progress or complete.

(2) Corresponding to a combination of `Expecting De-icing` flag and `flight status DEI` in OFA 05.01.01 appendix E.

(3) Values:

a) **`COMPLETED`**
Completed - We know that deicing has been done.

b) **`IN_PROGRESS`**
In progress - We know that deicing is taking place.

c) **`PLANNED`**
Planned - We know that deicing will take place, maybe taking place now or not sure.

(4) Used by: UpdateDPIRequest.

4.39. <<enumeration>> DelayCharacteristics

(1) Identifies the characteristics of the delay value associated to the flight.

(2) Values:

Value	Description
`ADJUSTED_TO_CLOCK`	Set when the delay value of the flight is adjusted to the clock.
`EXCEEDS_DELAY_CONFIRMATION`	Set when the delay value calculated for the flight is exceeding the delay confirmation threshold of a regulation affecting the flight.

Table 4.3. <<enumeration>> DelayCharacteristics

(3) Used by: Flight.

4.40. DeltaEntry

(1) Detects intruders by comparing the flight list for traffic type with demand or regulated demand or load.

(2) The result indicates if a flight is an intruder. The difference is returned between the times, flight levels, and geographical positions of the flight's entry point in the flight list and the flight list that is compared with.

(3) Attributes:

a) **`boolean isIntruder`** *(Mandatory)*
Indicates that a flight is an intruder or not. A flight is an intruder in a flight list if the other profile (according to the requested FlightListRequest.compareWithOtherTrafficType) is not crossing the reference location.

b) **AirspaceId originOfIntruder** *(Optional)*
The origin of an airspace intruder is defined as the active ATC sector in which the deviation that caused the intrusion was initiated.

c) **int deltaMinutes** *(Mandatory)*
The difference between the first entry time in the flight list (FlightListRequest.trafficType) and the flight list that is compared with (FlightListRequest.compareWithOtherTrafficType). `deltaMinutes` returned by NM are always truncated to 999 minutes.
Constraint: Range: `[-999,999]`.

d) **int deltaFlightLevel** *(Mandatory)*
The difference between the first level at first entry in the flight list (FlightListRequest.traffic-Type) and the flight list that is compared with (FlightListRequest.compareWithOtherTraffic-Type).
Constraint: Range: `[-999,999]`.

e) **DistanceNM deltaPosition** *(Mandatory)*
The distance in nautical miles between the geographical positions of the flight's entry point in the flight list (FlightListRequest.trafficType) and the flight list that is compared with (FlightListRequest.compareWithOtherTrafficType).
`deltaPosition` returned by NM are always truncated to 999 nautical miles.

(4) Used by: Flight.

4.41. <<enumeration>> DepartureAirportType

(1) Enumerates the possible Departure Airport Type values from which DPI messages could be received.

(2) Values:

Value	Description
ADVANCED_ATC_TWR	Advanced ATC TWR Airport.
CDM	CDM airport.
STANDARD	Standard airport.

Table 4.4. <<enumeration>> DepartureAirportType

(3) Used by: CDM.

4.42. DepartureData

(1) Provide information related to the departure of the aircraft before it takes-off, so from off block to take-off.

(2) Attributes:

a) **Duration taxiTime** *(Mandatory)*
The duration of the taxi time. This will be used by NM to calculate the take-off time by adding the `taxiTime` to the off-block time given for the flight plan. The `taxiTime` is provided in number of seconds that corresponds to what can be provided in an ICAO flight plan by using the `RMK/TAXI:mi` or in an ADEXP flight plan by using the field `-TAXI mi`; where `mi` is a number of minutes on 2 digits.
Constraint: See `TAXI_TIME_CANNOT_BE_GREATER_THAN_99_MIN`

(3) Constraint:

a)

Name	`TAXI_TIME_CANNOT_BE_GREATER_THAN_99_MIN`
Attribute	`taxiTime`
Description	The taxi time is limited to 99 minutes

(4) Used by: FlightPlanCreationRequest, FourDimensionalTrajectory, FlightPlanUpdateRequest.

4.43. <<abstract>> DeparturePlanningInformationRequest

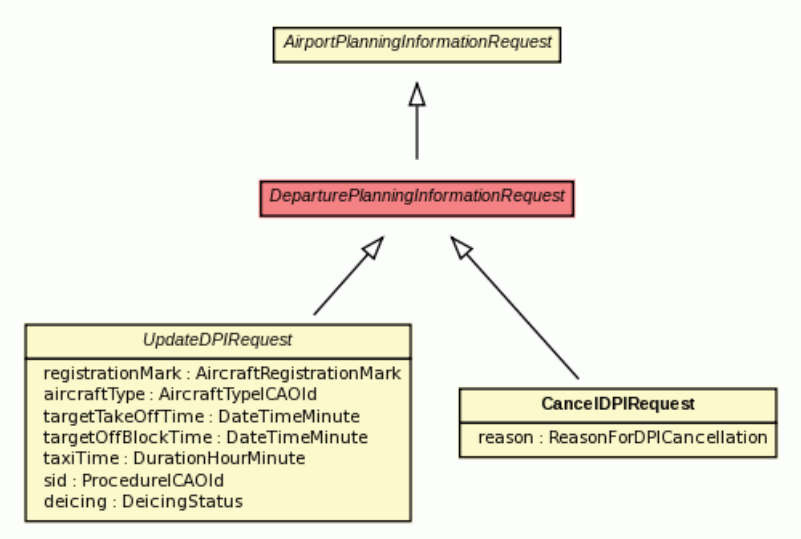

Figure 4.2. <<abstract>> DeparturePlanningInformationRequest Class Diagram

(1) Base class of all Departure Planning Information requests.

(2) Inherits from: `AirportPlanningInformationRequest`.

(3) Extended by: CancelDPIRequest, UpdateDPIRequest.

4.44. <<enumeration>> DepartureStatus

(1) Enumerates the possible departure status values.

(2) Values:

a) **DEICING**

b) **OK**

(3) Used by: CDMProvisionalInfo, CDMInfo.

4.45. DepartureTolerance

(1) Indicates the departure tolerance of a flight.

(2) Attributes:

a) **TimeHourMinutePeriod toleranceWindow** *(Mandatory)*
The departure tolerance window.

b) **boolean extended** *(Mandatory)*
True if this tolerance window differs from the default departure tolerance window.

(3) Used by: Flight.

4.46. Dinghies

(1) Information on the dinghies carried by an aircraft.

(2) Attributes:

a) **NumberOfDinghies_DataType numberOfDinghies** *(Optional)*
The number of dinghies carried by the aircraft.
If specified, must be in [0, 99].

b) **TotalCapacity_DataType totalCapacity** *(Optional)*
The total capacity, in persons, of all dinghies carried by the aircraft. If specified, must be in [0, 999].

c) **boolean areCovered** *(Optional)*
Specifies if the dinghies carried by the aircraft are covered.

d) **Colour_DataType colour** *(Optional)*
Colour of the dinghies carried by the aircraft.

(3) Used by: SupplementaryInformation.

4.47. DistanceAtLocation

(1) Attributes:

a) **AerodromeDAL adepDAL** *(Optional)*

b) **AerodromeDAL adesDAL** *(Optional)*

c) **PointDAL[] dalPoints** *(Mandatory)*
Constraint: Size must be comprised between 0 and ∞.

(2) Used by: BasicTrajectoryData.

4.48. EnrouteDelay

(1) Specify the point on the route where a delay is planned to occur together with the duration of the delay.

(2) Attributes:

a) **DurationHourMinute delay** *(Optional)*
The delay.

b) **ICAOPoint point** *(Optional)*
The point where the delay is planed to occur.

(3) Used by: FlightPlan.

4.49. <<enumeration>> EntryExit

(1) Indication on whether a flight is entering or exiting a volume.

(2) Values:

a) **ENTRY**
The flight is entering a volume (e.g. an airspace protected by a restriction).

b) **EXIT**
The flight is exiting a volume.

(3) Used by: FlightRestriction.

4.50. EquipmentCapabilityAndStatus

(1) Indicates the radio communication, navigation and approach aid equipment and capabilities of an aircraft.

(2) See also ICAO 4444 document field 10a.

(3) Attributes:

a) **EquipmentStatus gbas** *(Optional)*
GBAS Landing System.
ICAO code is "A".

b) **EquipmentStatus lpv** *(Optional)*

LPV.
ICAO code is "B".

c) **EquipmentStatus loranC** *(Optional)*
LORAN-C provides coverage for maritime navigation in U.S. coastal areas. It provides navigation, location, and timing services for both civil and military air, land and marine users. LORAN-C is approved as an en-route supplemental air navigation system for both Instrument Flight Rule (IFR) and Visual Flight Rule (VFR) operations.
ICAO code is "C".

d) **EquipmentStatus dme** *(Optional)*
Distance Measuring Equipment.
ICAO code is "D".

e) **EquipmentStatus fmcWprAcars** *(Optional)*
FMC WPR ACARS.
ICAO code is "E1".

f) **EquipmentStatus dFisAcars** *(Optional)*
D-FIS ACARS.
ICAO code is "E2".

g) **EquipmentStatus pdcAcars** *(Optional)*
PDC ACARS. ICAO code is "E3".

h) **EquipmentStatus adf** *(Optional)*
Automatic Direction Finder.
ICAO code is "F".

i) **EquipmentStatus gnss** *(Optional)*
Global Navigation Satellite Systems: a satellite assisted positioning system.
ICAO code is "G".

j) **EquipmentStatus hfRtf** *(Optional)*
High Frequency Radio Transmission Frequency.
ICAO code is "H".

k) **EquipmentStatus inertialNavigation** *(Optional)*
An inertial navigation system measures the position and altitude of a vehicle by measuring the accelerations and rotations applied to the system's inertial frame.
ICAO code is "I".

l) **EquipmentStatus cpdlcAtnVdlMode2** *(Optional)*
CPDLC ATN VHF Data Link Mode 2.
ICAO code is "J1".

m) **EquipmentStatus cpdlcFans1AHFDL** *(Optional)*
CPDLC FANS 1/A HF Data Link.
ICAO code is "J2".

n) **EquipmentStatus** **cpdlcFans1AVdlModeA** *(Optional)*
CPDLC FANS 1/A VHF Data Link Mode A.
ICAO code is "J3".

o) **EquipmentStatus** **cpdlcFans1AVdlMode2** *(Optional)*
CPDLC FANS 1/A VHF Data Link Mode 2.
ICAO code is "J4".

p) **EquipmentStatus** **cpdlcFans1ASatcomInmarsat** *(Optional)*
CPDLC FANS 1/A Satellite Communication INMARSAT.
ICAO code is "J5".

q) **EquipmentStatus** **cpdlcFans1ASatcomMtsat** *(Optional)*
CPDLC FANS 1/A Satellite Communication MTSAT.
ICAO code is "J6".

r) **EquipmentStatus** **cpdlcFans1ASatcomIridium** *(Optional)*
CPDLC FANS 1/A Satellite Communication Iridium.
ICAO code is "J7".

s) **EquipmentStatus** **mls** *(Optional)*
Microwave Landing System.
ICAO code is "K".

t) **EquipmentStatus** **ils** *(Optional)*
Instrument Landing System.
ICAO code is "L".

u) **EquipmentStatus** **atcRtfSatcomInmarsat** *(Optional)*
ATC Radio Telephony by Satellite Communication INMARSAT.
ICAO code is "M1".

v) **EquipmentStatus** **atcRtfSatcomMtsat** *(Optional)*
ATC Radio Telephony by Satellite Communication MTSAT.
ICAO code is "M2"

w) **EquipmentStatus** **atcRtfSatcomIridium** *(Optional)*
ATC Radio Telephony by Satellite Communication Iridium.
ICAO code is "M3".

x) **EquipmentStatus** **vor** *(Optional)*
VHF omni-directional radio range.
ICAO code is "O".

y) **EquipmentStatus** **rcp1** *(Optional)*
Reserved for RCP.
ICAO code is "P1".

z) **EquipmentStatus** **rcp2** *(Optional)*

Reserved for RCP.
ICAO code is "P2".

aa) **EquipmentStatus rcp3** *(Optional)*
Reserved for RCP.
ICAO code is "P3".

ab) **EquipmentStatus rcp4** *(Optional)*
Reserved for RCP.
ICAO code is "P4".

ac) **EquipmentStatus rcp5** *(Optional)*
Reserved for RCP.
ICAO code is "P5".

ad) **EquipmentStatus rcp6** *(Optional)*
Reserved for RCP.
ICAO code is "P6".

ae) **EquipmentStatus rcp7** *(Optional)*
Reserved for RCP.
ICAO code is "P7".

af) **EquipmentStatus rcp8** *(Optional)*
Reserved for RCP.
ICAO code is "P8".

ag) **EquipmentStatus rcp9** *(Optional)*
Reserved for RCP.
ICAO code is "P9".

ah) **EquipmentStatus pbnApproved** *(Optional)*
Indicates that the aircraft meets the Performance-based Navigation type.
ICAO code is "R".

ai) **EquipmentStatus standard** *(Optional)*
Standard equipment.
ICAO code is "S".

aj) **EquipmentStatus tacan** *(Optional)*
TACtical Air Navigation.
ICAO code is "T".

ak) **EquipmentStatus uhfRtf** *(Optional)*
Ultra High Frequency Radio Transmission Frequency.
ICAO code is "U".

al) **EquipmentStatus vhfRtf** *(Optional)*
VHF Radio Telephony.
ICAO code is "V".

am) **EquipmentStatus rvsm** *(Optional)*
Aircraft equipped to navigate in airspace where the "Reduced Vertical Separation Minima" is applicable.
ICAO code is "W".

an) **EquipmentStatus mnps** *(Optional)*
"Minimum Navigation Performance Specifications" approved aircraft.
"ICAO code is "X".

ao) **EquipmentStatus khz833** *(Optional)*
Equipped with VHF with a 8.33KHz channel spacing radio equipment.
ICAO code is "Y".

ap) **EquipmentStatus other** *(Optional)*
Other equipment as specified in OtherInformation (`communicationEquipment`, `navigationEquipment`, `datalinkCapabilities`).
ICAO code is "Z".

(4) Used by: Flight, FlightPlan, FlightPlanUpdate.

4.51. <<enumeration>> EquipmentStatus

(1) Describes the status of any kind of navigation equipment.

(2) Values:

a) **EQUIPPED**
The aircraft is equipped with the specified equipment.

b) **NOT_EQUIPPED**
The aircraft is not equipped with the specified equipment.

(3) Used by: SurveillanceEquipment, ModeSCapabilities, EquipmentCapabilityAndStatus.

4.52. EstimatedElapsedTimeAtLocation

(1) Association of a location and an elapsed time.

(2) Attributes:

a) **DurationHourMinute elapsedTime** *(Mandatory)*
The elapsed time.

b) **FIRICAOId fir** *(Optional)*
A FIR.
Constraints:

i) See AT_LEAST_ONE_LOCATION_SHOULD_BE_DEFINED

ii) See LOCATIONS_ARE_MUTUALLY_EXCLUSIVE

c) **ICAOPoint point** *(Optional)*
A point.
Constraints:

i) See AT_LEAST_ONE_LOCATION_SHOULD_BE_DEFINED

ii) See LOCATIONS_ARE_MUTUALLY_EXCLUSIVE

d) **Latitude latitude** *(Optional)*
A latitude.
Constraints:

i) See AT_LEAST_ONE_LOCATION_SHOULD_BE_DEFINED

ii) See LOCATIONS_ARE_MUTUALLY_EXCLUSIVE

e) **Longitude longitude** *(Optional)*
A longitude.
Constraints:

i) See AT_LEAST_ONE_LOCATION_SHOULD_BE_DEFINED

ii) See LOCATIONS_ARE_MUTUALLY_EXCLUSIVE

(3) Constraints:

a)

Name	AT_LEAST_ONE_LOCATION_SHOULD_BE_DEFINED
Attributes	fir, point, latitude, longitude
Description	Exactly one location (fir, point, latitude or longitude) must be not null.

b)

Name	LOCATIONS_ARE_MUTUALLY_EXCLUSIVE
Attributes	fir, point, latitude, longitude
Description	The locations (fir, point, latitude or longitude) are mutually exclusive.

(4) Used by: FlightPlan, FlightPlanUpdate.

4.53. <<enumeration>> EURSTSIndicator

(1) Enumerates the non-ICAO STS indicators in use in the EUR region.

(2) Values:

Value	Description
CPDLCX	Flights conducted wholly or partly in EUR CPDLC airspace, and not equipped with CPDLC capabilities but which have been granted an exemption.

Value	Description
EXM833	Exempted from the carriage of the 8.33kHz radios.
PROTECTED	Sensitive flights.
RNAVINOP	Failure or degradation results in aircraft being unable to meet B-RNAV functionality and accuracy requirements.
RNAVX	Aircraft not equipped with RNAV.

Table 4.5. <<enumeration>> EURSTSIndicator

(3) Used by: SpecialHandlingIndicators.

4.54. `ExclusionFromRegulations`

(1) Provides data regarding the possible exclusion of a flights from regulations.

(2) Attributes:

a) **`boolean onTrafficVolume`** *(Optional)*
Used only in flights that have been queried by traffic volume.
True if the flight is excluded from one or more regulations defined on the traffic volume.

b) **`int count`** *(Optional)*
The total count of regulations from which the flight is excluded.
Must be null if is `all` is true; cannot be null otherwise.
Constraint: Range: `[0,∞[`.

c) **`boolean all`** *(Optional)*
True if the flight is excluded from all regulations.
Must be null if is `count` is not null; cannot be null otherwise.

d) **`boolean hasBeenExcluded`** *(Mandatory)*
True if the flight has been excluded from one or more regulations in the past but is no longer.

(3) Used by: Flight.

4.55. `typedef<string> ExtendedAircraftICAOId`

(1) ICAO aircraft identification as defined in ICAO doc 4444 extended with characters '$' and '#'. These special characters are used by NM in the context of prediction and simulation exercises.

(2) Pattern: `(ALPHA|DIGIT|$|#){2,7}`

(3) Used by: FlightKeys.

4.56. ExtendedFlightPlan

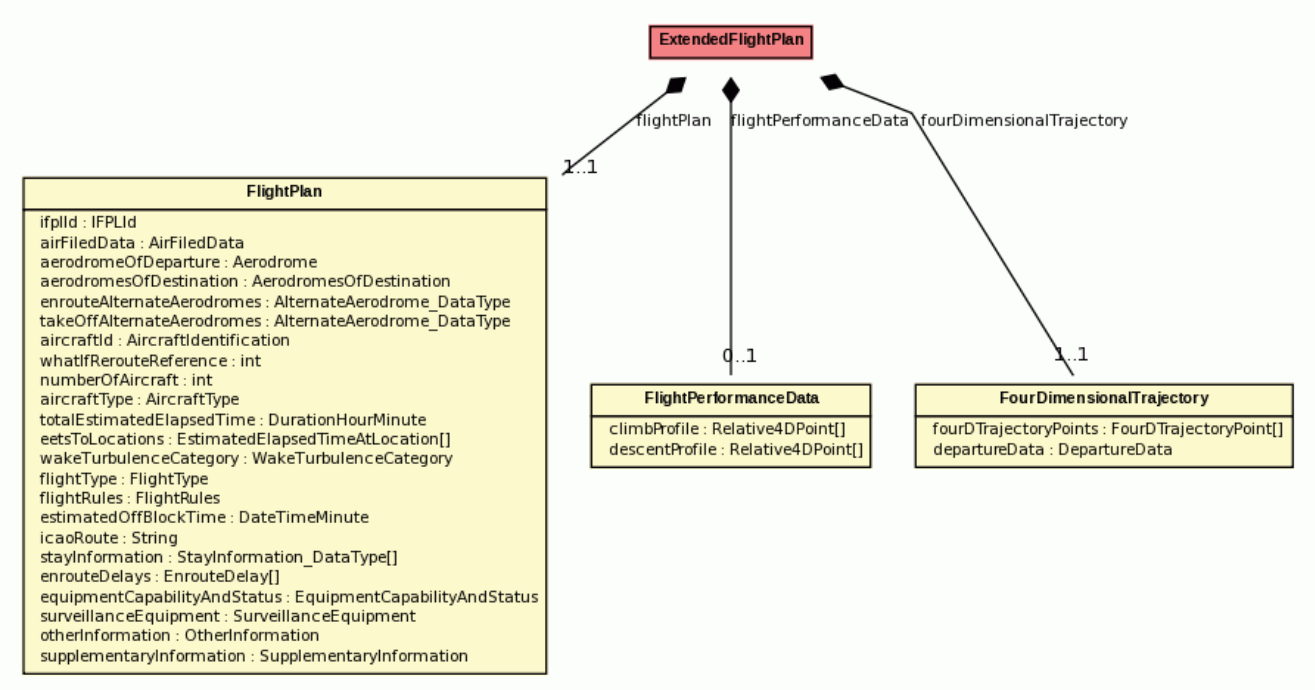

Figure 4.3. ExtendedFlightPlan Class Diagram

(1) An Extended Flight Plan Message (EFPLM) is a flight plan message which, in addition to the ICAO defined flight plan information, includes also flight trajectory information in the form of a 4D trajectory, as calculated by the operator of the flight, as well as Performance Data specific to the flight.

(2) Attributes:

a) **FourDimensionalTrajectory fourDimensionalTrajectory** *(Mandatory)*
Four dimensional profile calculated by the AO.

b) **FlightPlan flightPlan** *(Mandatory)*
All data to be provided in a filed flight plan as specified in the ICAO Doc 4444, including the Field 15 route information.

c) **FlightPerformanceData flightPerformanceData** *(Optional)*
Performance profiles instantiated per flight.
Constraint: See PERFORMANCE_DATA_CAN-NOT_BE_EMPTY_WHEN_NOT_ALL_POINTS_ARE_PROVIDED_WITH_A_WEIGHT

(3) Constraint:

a)

Name	PERFORMANCE_DATA_CAN-NOT_BE_EMPTY_WHEN_NOT_ALL_POINTS_ARE_PROVIDED_WITH_A_WEIGHT

Attribute	flightPerformanceData
Description	The flightPerformanceData can be empty if the totalWeight of the aircraft is provided for each 4DTrajectoryPoint. The flightPerformanceData cannot be empty if the totalWeight of the aircraft is not provided for each 4DTrajectoryPoint.

(4) Used by: ExtendedFlightPlanCreationRequest, ExtendedFlightPlanCreationReply, ExtendedFlightPlanUpdateReply, FlightPlanUnion.

4.57. ExtendedFlightPlanUpdate

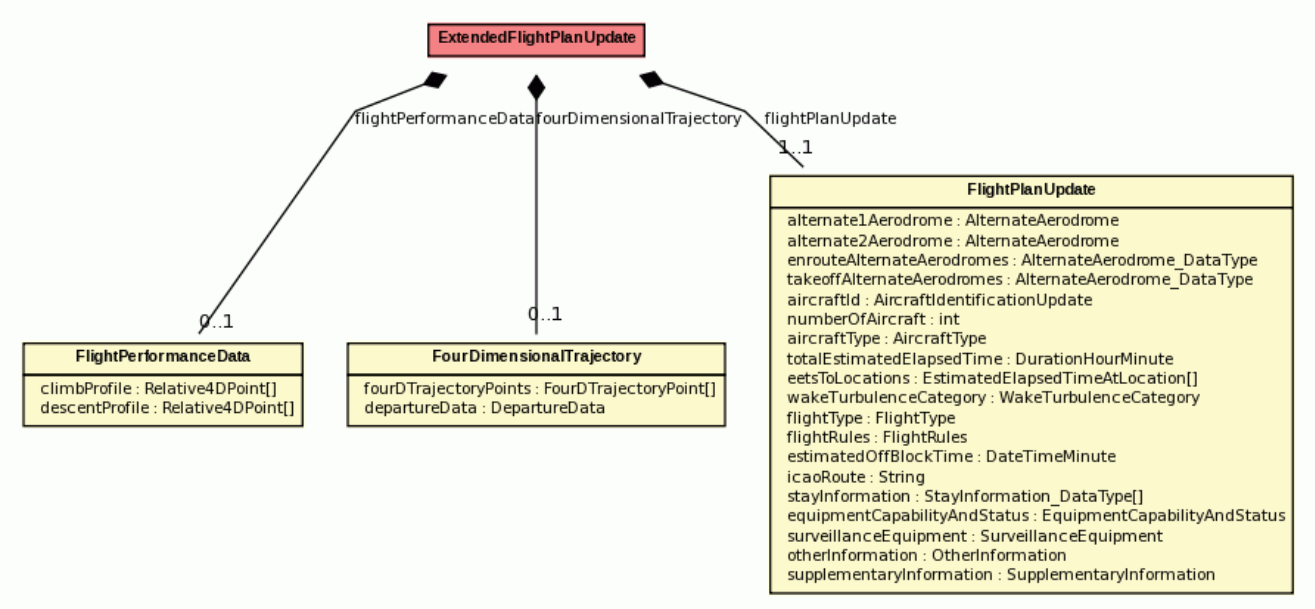

Figure 4.4. ExtendedFlightPlanUpdate Class Diagram

(1) An Extended Flight Plan Update message is a flight plan message which, in addition to the ICAO defined flight plan information, includes also flight trajectory information in the form of a 4D trajectory, as calculated by the operator of the flight, as well as Performance Data specific to the flight.

(2) Attributes:

a) **FourDimensionalTrajectory fourDimensionalTrajectory** *(Optional)*
Four dimensional profile calculated by the AO.

b) **FlightPlanUpdate flightPlanUpdate** *(Mandatory)*
Update of selective fields in the existing flight plan.

c) **FlightPerformanceData flightPerformanceData** *(Optional)*
Performance profiles instantiated per flight.
Constraint: See PERFORMANCE_DATA_CANNOT_BE_EMPTY_WHEN_NOT_ALL_POINTS_ARE_PROVIDED_WITH_A_WEIGHT

(3) Constraint:

a)

Name	PERFORMANCE_DATA_CAN-NOT_BE_EMPTY_WHEN_NOT_ALL_POINTS_ARE_PROVIDED_WITH_A_WEIGHT
Attribute	flightPerformanceData
Description	The flightPerformanceData can be empty if the totalWeight of the aircraft is provided for each 4DTrajectoryPoint. The flightPerformanceData cannot be empty if the totalWeight of the aircraft is not provided for each 4DTrajectoryPoint.

(4) Used by: ExtendedFlightPlanUpdateRequest.

4.58. <<enumeration>> FAMStatus

(1) Describes a flight FAM (Flight Activation Message) status.

(2) Values:

a) **AIRBORNE_WHEN_SHIFTED_BY_FAM**
was shifted by FAM, airborne data received when shifted.

b) **AIRBORNE_WHEN_SUSPENDED_BY_FAM**
was suspended by FAM, airborne data received when suspended.

c) **NOT_UNDER_FAM**
not under FAM yet or never under FAM.

d) **SHIFTED_BY_FAM**
currently shifted by FAM.

e) **SUBJECT_TO_FAM**
currently subject to FAM, if no data received, flight will be shifted soon.

f) **SUSPENDED_BY_FAM**
currently suspended by FAM.

g) **WAS_SHIFTED_BY_FAM**
was shifted by FAM, FPL data received when shifted.

h) **WAS_SUBJECT_TO_FAM**
was subject to FAM but airborne data received before first shift.

i) **WAS_SUSPENDED_BY_FAM**
was suspended by FAM, FPL data received when suspended.

(3) Used by: Flight.

4.59. typedef<string> FilingId

(1) Examples:AA00953172BB00956485

(2) Pattern: `(UALPHA{2}DIGIT{8}){1,2}`

(3) Used by: FilingResultQueued, FilingStatusRequest.

4.60. <<abstract>> FilingReplyData

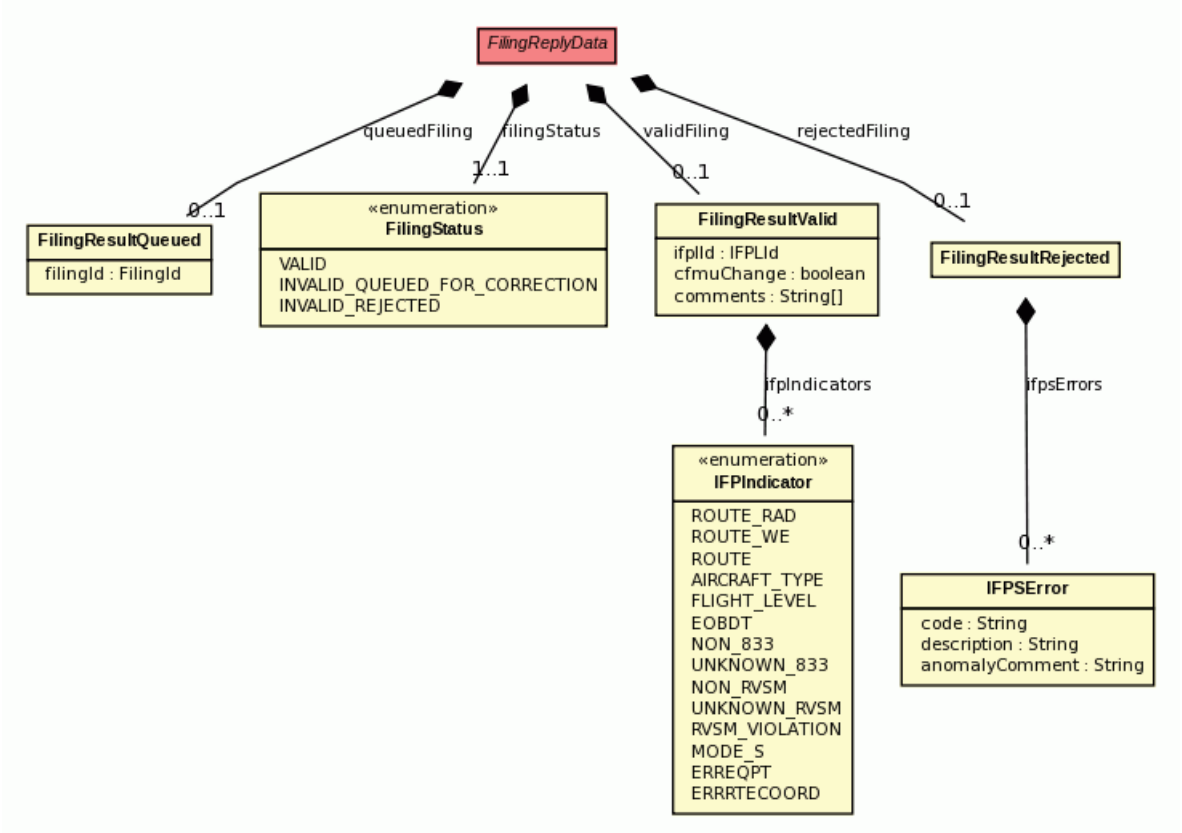

Figure 4.5. <<abstract>> FilingReplyData Class Diagram

(1) Abstract reply returned in response to an abstract FilingRequest.

(2) The content of the `FilingReplyData` structure depends on whether the filing request was evaluated by NM as:

a) `VALID`

b) `INVALID_QUEUED_FOR_CORRECTION`

c) `INVALID_REJECTED`

(3) This is expressed via a FilingStatus enumeration value that discriminates three attributes in the choice described below.

(4) Attributes:

a) **`FilingStatus` `filingStatus`** *(Mandatory)*
Indicates whether the filing request was evaluated by NM as valid, or was rejected, or queued for manual correction by an NM operator.

b) **`FilingResultValid` `validFiling`** *(Optional)*
Contains information regarding the processing of the filing request when it was valid.
Cannot be null if `filingStatus` is `VALID`; must be null otherwise.

c) **`FilingResultQueued` `queuedFiling`** *(Optional)*
Contains information regarding the processing of the filing request when it was queued for manual correction by an NM operator.
Cannot be null if `filingStatus` is `INVALID_QUEUED_FOR_CORRECTION`; must be null otherwise.

d) **`FilingResultRejected` `rejectedFiling`** *(Optional)*
Contains information regarding the processing of the filing request when it was rejected.
Cannot be null if `filingStatus` is `INVALID_REJECTED`; must be null otherwise.

(5) Extended by: FlightDelayReplyData, ExtendedFlightPlanUpdateReplyData, FlightPlanUpdateReplyData, ExtendedFlightPlanCreationReplyData, FlightArrivalReplyData, FlightPlanCreationReplyData, FlightDepartureReplyData, FlightPlanCancellationReplyData, FilingStatusReplyData.

4.61. <<abstract>> FilingRequest

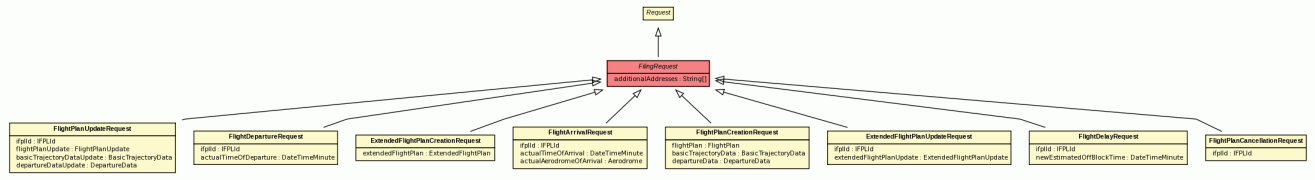

Figure 4.6. <<abstract>> FilingRequest Class Diagram

(1) Abstract ancestor request for all concrete requests that file flight plan related data to NM.

(2) The corresponding replies to `FilingRequest` requests all inherit from the abstract FilingReplyData.

(3) Inherits from: `Request`.

(4) Attributes:

a) **`string[]` `additionalAddresses`** *(Optional)*
AFTN addresses to which NM shall distribute the message after being accepted.
Constraint: Size must be comprised between 0 and ∞.

(5) Extended by: FlightPlanCancellationRequest, FlightDepartureRequest, FlightPlanCreationRequest, FlightArrivalRequest, FlightDelayRequest, ExtendedFlightPlanUpdateRequest, FlightPlanUpdate-Request, ExtendedFlightPlanCreationRequest.

4.62. FilingResultQueued

(1) Returned when the filing request is queued for manual correction by an NM operator.

(2) Attributes:

a) **FilingId filingId** *(Mandatory)*
Id of the received filing request when it results in queuing for manual correction by an NM operator. This id is to be used when subsequently retrieving the status of this filing.

(3) Used by: FilingReplyData.

4.63. FilingResultRejected

(1) Returned when the filed flight plan is rejected.

(2) Attributes:

a) **IFPSError[] ifpsErrors** *(Mandatory)*
The list of errors that caused the rejection.
Constraint: Size must be comprised between 0 and ∞.

(3) Used by: FilingReplyData.

4.64. FilingResultValid

(1) Returned when the filed flight plan is valid.

(2) Attributes:

a) **IFPLId ifplId** *(Optional)*
Cannot be null.

b) **boolean cfmuChange** *(Mandatory)*
True if NM has modified the filed data (routes); false otherwise.

c) **IFPIndicator[] ifpIndicators** *(Optional)*
Indications of errors that have been found in a flight plan and either ignored or automatically or manually corrected during the processing of the filing request.
Constraint: Size must be comprised between 0 and ∞.

d) **string[] comments** *(Optional)*
Comment produced by NM during the processing of the filing request.
Constraint: Size must be comprised between 0 and ∞.

(3) Used by: FilingReplyData.

4.65. <<enumeration>> FilingRule

(1) Describes the filing rule applying to a flight.

(2) Values:

a) **FILING_ALLOWED_BY_AO_CFMU**

b) **NOT_AUTHORISED**

c) **OPERATOR_MUST_REFILE**

(3) Used by: Flight.

4.66. <<enumeration>> FilingStatus

(1) Describes the status of a filing reply, resulting from the processing of a filing request.

(2) Values:

a) **INVALID_QUEUED_FOR_CORRECTION**
NM has evaluated that the filed flight plan is invalid but candidate for manual correction by an NM operator.

b) **INVALID_REJECTED**
NM has evaluated that the filed flight plan is invalid and is not candidate for manual correction by an NM operator.

c) **VALID**
NM has evaluated that the filed flight plan is valid.

(3) Used by: FilingReplyData.

4.67. Flight

(1) Description of a flight as seen from the NM perspective.

(2) Apart from the flight keys (that are always returned), all attributes are optional, depending on whether they are requested by the caller and available in the NM system.

(3) "Light" attributes (cf. FlightField regarding the weight of the `Flight` fields):

(4) Attributes:

a) **FlightIdentificationOutput flightId** *(Mandatory)*
IFPL id and flight keys associated to the flight. This attribute is always returned.

b) **AerodromeICAOId divertedAerodromeOfDestination** *(Optional)*
Diverted aerodrome of destination, if the flight was diverted. Null if the flight was not diverted.

c) **AircraftTypeICAOId** **aircraftType** *(Optional)*
ICAO id of the aircraft type.

d) **DateTimeMinute** **readyEstimatedOffBlockTime** *(Optional)*
Last flight plan related estimated off-block time, but amended by NM OPS room or READY message (filing time).
Present if different from the latest flight plan related estimated off-block time in flightId.keys; null otherwise.
Note that `readyEstimatedOffBlockTime` can be different than the FTFM flight profile off-block time (in case cdmEstimatedOffBlockTime is also set).

e) **DateTimeMinute** **cdmEstimatedOffBlockTime** *(Optional)*
Last ready estimated off-block time amended by E-DPI or T-DPI_t message (target take-off time).
Present if different from the latest flight plan related estimated off-block time in `flight-Id.keys`; null otherwise.
`cdmEstimatedOffBlockTime` corresponds to the FTFM flight profile off-block time. So in case DPI have been received (not cancelled by a C-DPI), then the `cdmEstimatedOff-BlockTime` = DPI TTOT - DPI taxi. Except if it concerns a DPI that creates a CTFM flight profile (ATC-DPI or T-DPI-sS that is conform with the slotwindow), then the TTOT is stored in the `actualTakeOffTime`.

f) **RevisionTimes** **revisionTimes** *(Optional)*
Revision times, i.e. times to insert an aircraft in sequence and remove an aircraft from sequence at the aerodrome of departure.

g) **DateTimeMinute** **estimatedTakeOffTime** *(Optional)*
Estimated take-off time: the take-off time corresponding to the FTFM flight profile.
The corresponding estimated off-block time of the FTFM flight profile is the `flight.estim-atedTakeOffTime` - `flight.taxiTime` (Note that this can be different than the latest flight plan related estimated off-block time in flightId.keys).

h) **DateTimeMinute** **calculatedTakeOffTime** *(Optional)*
Calculated take-off time: the take-off time corresponding to the RTFM flight profile.
The corresponding calculated off-block time is the `flight.calculatedTakeOffTime` - `flight.taxiTime` (except in some cases in FORECAST dataset).

i) **DateTimeMinute** **actualTakeOffTime** *(Optional)*
Estimated Actual take-off time: the take-off time corresponding to the CTFM flight profile.
The corresponding estimated actual off-block time is the `flight.actualTakeOffTime` - `flight.currentlyUsedTaxiTime`.

j) **ShiftHourMinute** **ctotShiftAlreadyAppliedByTower** *(Optional)*
Shift of calculated take-off time already applied by tower.

k) **FlightLevel** **requestedFlightLevel** *(Optional)*
This field is only returned in concrete FlightListByLocationReplyData types. It is ignored (and therefore not returned) if requested in other flight request types.

In FlightListByAerodromeReply, FlightListByAerodromeSetReply and FlightListByAircraft-OperatorReply, the field is the highest requested flight level.
In other concrete FlightListByLocationReplyData types, it represents the requested flight level applicable on the portion of the flight route that penetrates the location.

l) **DurationHourMinute taxiTime** *(Optional)*
Taxi time for the FTFM profile.

m) **DurationHourMinuteSecond currentlyUsedTaxiTime** *(Optional)*
Taxi time corresponding to the highest available TFM flight profile (CTFM if CTFM exists, otherwise RTFM, otherwise FTFM).
So basically if T-DPI-s/ATC-DPI have been received, `currentlyUsedTaxiTime` will contain the taxi time changes from that DPI and `Flight.taxiTime` will not contain that latest update (it can contain the previous taxi time from last Early-DPI or T-DPI-t update).

n) **DateTimeMinute estimatedTimeOfArrival** *(Optional)*
Estimated time of arrival: time of arrival according to the FTFM flight profile.

o) **DateTimeMinute calculatedTimeOfArrival** *(Optional)*
Calculated time of arrival: time of arrival accorting to the RTFM flight profile.

p) **DateTimeMinute actualTimeOfArrival** *(Optional)*
Estimated Actual time of arrival: time of arrival according to the CTFM flight profile.

q) **boolean lateFiler** *(Optional)*
True if the status of the flight is "Late Filer".

r) **boolean lateUpdater** *(Optional)*
True if the status of the flight is "Late Updater".

s) **SuspensionStatus suspensionStatus** *(Optional)*
Suspension status.

t) **DateTimeMinute suspensionResponseBy** *(Optional)*
The time limit for the confirmation message (FCM) after the issuance of a flight suspension, if present for the flight.

u) **FAMStatus famStatus** *(Optional)*
FAM (Flight Activation Monitoring) status.

v) **ReadyStatus readyStatus** *(Optional)*
Ready status.

w) **AircraftOperatorICAOId aircraftOperator** *(Optional)*
Aircraft operator.

x) **AircraftOperatorICAOId operatingAircraftOperator** *(Optional)*
Operating aircraft operator.

y) **ReroutingIndicator reroutingIndicator** *(Optional)*

Indicates if the flight was rerouted, why, and the resulting rerouting state.

z) **DurationHourMinute newRouteMinShiftDelayImprovement** *(Optional)*
Minimum improvement needed (by reducing either the shift or the delay of the flight) to allow an Aircraft Operator What-If-Reroute. In the current implementation, this value is a system parameter: the returned value is always the same for all flights.

aa) **ReroutableStatus reroutable** *(Optional)*
Indicates if the flight is reroutable and under what conditions.

ab) **boolean reroutingOpportunitiesExist** *(Optional)*
Indicates if there exist rerouting opportunities for this flight.

ac) **CDM cdm** *(Optional)*
CDM (Collaborative Design Making) information.

ad) **boolean slotIssued** *(Optional)*
Indicates that the flight is in state slot-issued or was in that state prior to activation/termination.

ae) **SlotImprovementProposal slotImprovementProposal** *(Optional)*
Proposal for slot improvement.

af) **TimeAndModel timeAtReferenceLocationEntry** *(Optional)*
Indicates the entry time at the reference location.

ag) **TimeAndModel timeAtReferenceLocationExit** *(Optional)*
Indicates the exit time at the reference location.

ah) **FlightLevel flightLevelAtReferenceLocationEntry** *(Optional)*
Indicates the entry time at the reference location.

ai) **FlightLevel flightLevelAtReferenceLocationExit** *(Optional)*
Indicates the exit time at the reference location.

aj) **FlightTrend trendAtReferenceLocationEntry** *(Optional)*
This field is only returned in FlightListByAirspaceReply and FlightListByTrafficVolumeReply. It is ignored (and therefore not returned) if requested in other flight request types.
Flight trend at the entry point of the location, i.e. cruising, climbing or descending.

ak) **FlightTrend trendAtReferenceLocationExit** *(Optional)*
This field is only returned in `FlightListByAirspaceReply` and `FlightListByTrafficVolumeReply`. It is ignored (and therefore not returned) if requested in other flight request types.
Flight trend at the exit point of the location, i.e. cruising, climbing or descending.

al) **FlightTrend trendAtReferenceLocationMiddle** *(Optional)*
Flight trend at the middle point of the location, i.e cruising, climbing or descending.

am) **boolean exemptedFromRegulations** *(Optional)*
True if the flight is exempted from regulations.

an) **DurationHourMinute delay** *(Optional)*
Delay applying to this flight (calculated take-off time minus estimated take-off time).

ao) **DelayCharacteristics delayCharacteristics** *(Optional)*
Identifies the characteristics of the delay value associated to the flight.

ap) **RegulationId mostPenalisingRegulation** *(Optional)*
Most penalising regulation impacting this flight.

aq) **boolean hasOtherRegulations** *(Optional)*
Indicates if this flight is impacted by other regulations than the most penalising one.

ar) **FlightRegulationLocation[] regulationLocations** *(Optional)*
Locations of the regulations impacting this flight.
Constraint: Size must be comprised between 0 and ∞.

as) **Set<FlightAtfcmMeasureLocation> atfcmMeasureLocations** *(Optional)*
Locations of the ATFCM measures impacting this flight.
Constraint: Size must be comprised between 0 and ∞.

at) **ATFMMessageType lastATFMMessageType** *(Optional)*
Indicates the type of the last exchanged ATFM message.

au) **ReceivedOrSent lastATFMMessageReceivedOrSent** *(Optional)*
Indicates whether the last ATFM message was received or sent by NM.

av) **DistanceM runwayVisualRange** *(Optional)*
Minimum visible range in meters for a flight to land.
Must be in [0, 999].

aw) **DistanceNM confirmedCTFM** *(Optional)*
Describes the distance on the CTFM (Current Tactical Flight Model) route that has been confirmed by CPR's.

ax) **ExclusionFromRegulations exclusionFromRegulations** *(Optional)*
Quantitative information regarding the regulations from which this flight is possibly excluded.

ay) **FlightLevel requestedInitialFlightLevel** *(Optional)*
The first flight level requested for this flight after departure.

az) **AirSpeed requestedInitialSpeed** *(Optional)*
The first true airspeed requested for this flight after departure.

ba) **DurationHourMinute estimatedElapsedTime** *(Optional)*
Estimated elapsed time.

bb) **FilingRule filingRule** *(Optional)*
Filing rule.

bc) **MessageOriginator flightPlanOriginator** *(Optional)*

Originator of the flight plan.

bd) **`string icaoRoute`** *(Optional)*
Complete ICAO field 15 information comprising of initial requested speed and flight level and route.
Contains corrected flight plan route information sent from NM to addressees outside NM. Note that the route is not always available, e.g. for flights that are full VFR.

be) **`DistanceNM routeLength`** *(Optional)*
Length of the route.

bf) **`ReroutingReference reroutingReference`** *(Optional)*
Rerouting reference (if any).

bg) **`FlightLevel defaultReroutingRequestedFlightLevel`** *(Optional)*
When rerouting, suggested flight level to be used for generating an alternate route.

bh) **`AirSpeed defaultReroutingRequestedSpeed`** *(Optional)*
When rerouting, suggested air speed to be used for generating an alternate route.

bi) **`DepartureTolerance departureTolerance`** *(Optional)*
Departure tolerance.

bj) **`RegulationCause mostPenalisingRegulationCause`** *(Optional)*
Cause of the most penalising regulation impacting this flight.

bk) **`MessageOriginator lastATFMMessageOriginator`** *(Optional)*
If the last ATFM message exchanged was received by NM, indicates its originator.

bl) **`FlightPoint[] ftfmPointProfile`** *(Optional)*
FTFM (Filed Tactical Flight Model) point profile.
The FTFM flight profile corresponds to the `trafficType` DEMAND. So in the operational dataset, it reflects the latest AO flightplan: i.e. the latest filed flightplan but updated (shifted) with the latest CDM related info and READY messages or amended by NM OPS room.
Note that the FTFM off-block time does not necessarily corresponds to the FlightKeys.estimatedOffBlockTime.
Constraint: Size must be comprised between 0 and ∞.

bm) **`FlightPoint[] rtfmPointProfile`** *(Optional)*
RTFM (Regulated Tactical Flight Model) point profile. If a flight has an RTFM, then it is the RTFM flight profile that is used for `trafficType` REGULATED_DEMAND.
Constraint: Size must be comprised between 0 and ∞.

bn) **`FlightPoint[] ctfmPointProfile`** *(Optional)*
CTFM (Current Tactical Flight Model) point profile.
If a flight has an CTFM, then it is the CTFM flight profile that is used for `trafficType` LOAD. Typically a flight has a CTFM point profile once it is off-block. However if the flight is involved in airport CDM, then the flight can have a CTFM point profile even if its CTFM off-block time is still relatively far in the future (e.g. 40 minutes) due to T-DPI-s.

Constraint: Size must be comprised between 0 and ∞.

bo) **FlightAirspace[] ftfmAirspaceProfile** *(Optional)*
FTFM airspace profile.
Constraint: Size must be comprised between 0 and ∞.

bp) **FlightAirspace[] rtfmAirspaceProfile** *(Optional)*
RTFM airspace profile.
Constraint: Size must be comprised between 0 and ∞.

bq) **FlightAirspace[] ctfmAirspaceProfile** *(Optional)*
CTFM airspace profile.
Constraint: Size must be comprised between 0 and ∞.

br) **RequestedFlightLevel[] ftfmRequestedFlightLevels** *(Optional)*
FTFM requested flight levels.
Constraint: Size must be comprised between 0 and ∞.

bs) **RequestedFlightLevel[] rtfmRequestedFlightLevels** *(Optional)*
RTFM requested flight levels.
Constraint: Size must be comprised between 0 and ∞.

bt) **RequestedFlightLevel[] ctfmRequestedFlightLevels** *(Optional)*
CTFM requested flight levels.
Constraint: Size must be comprised between 0 and ∞.

bu) **FlightEvent[] flightHistory** *(Optional)*
Ordered (time) list of events that make up the flight history.
Constraint: Size must be comprised between 0 and ∞.

bv) **FlightOperationalLogEntry[] operationalLog** *(Optional)*
This attribute can only be retrieved in a retrieval context, not in a list context (i.e. it can only be retrieved for a single flight via FlightRetrievalRequest). Ordered (time) list of entries that make up the flight operational log.
Constraint: Size must be comprised between 0 and ∞.

bw) **EquipmentCapabilityAndStatus equipmentCapabilityAndStatus** *(Optional)*
Indicates the radio communication, navigation and approach aid equipment and capabilities of an aircraft.

bx) **FlightRestriction[] ftfmRestrictionProfile** *(Optional)*
FTFM restriction profile.
Constraint: Size must be comprised between 0 and ∞.

by) **FlightRestriction[] rtfmRestrictionProfile** *(Optional)*
RTFM restriction profile.
Constraint: Size must be comprised between 0 and ∞.

bz) **FlightRestriction[] ctfmRestrictionProfile** *(Optional)*
CTFM restriction profile.

Constraint: Size must be comprised between 0 and ∞.

ca) **CfmuFlightType cfmuFlightType** *(Optional)*
Indicates state and/or origin of the flight in the NM system.

cb) **SSRCode ccamsSSRCode** *(Optional)*
SSR code as assigned by CCAMS.

cc) **AircraftRegistrationMark filedRegistrationMark** *(Optional)*
Aircraft registration mark as it was filed via flight plan messages.
Note that the registration mark can be provided or updated later, via DPI messages - see `Flight.cdm` attribute.

cd) **boolean isProposalFlight** *(Optional)*
Indicates if this flight is a proposal flight or not.
(See Proposal Flights).

ce) **boolean proposalExists** *(Optional)*
Indicates if a proposal flight exists or not.

cf) **boolean hasBeenForced** *(Optional)*
Indicates if for this flights, the delay is or has been forced.

cg) **int caughtInHotspots** *(Optional)*
Indicates this flight in how many hotspots has been caught.
Note that the hotspot related fields are trial related (STAM) fields : they are only accessible (authorized) during specific trials or on test platforms.
Constraint: Range: `[0,∞[`.

ch) **Set<FlightHotspotLocation> hotspots** *(Optional)*
Locations of all hotspots that are impacting this flight.
Note that the hotspot related fields are trial related (STAM) fields : they are only accessible (authorized) during specific trials or on test platforms.
Constraint: Size must be comprised between 0 and ∞.

ci) **FlightMCDMInfo mcdmInfo** *(Optional)*
Measure Collaborative Decision Making Info associated with to this flight: the most relevant M-CDM measure and its M-CDM state and indications if other M-CDM measures are impacting the flight.
Note that M-CDM related fields/services are trial related fields (a.o. STAM trials): they are only accessible (authorized) during specific trials or on test platforms.

cj) **LoadStateAtReferenceLocation worstLoadStateAtReferenceLocation** *(Optional)*
Indicates what is the worst monitored (entry or OTMV) load state in which this flight is involved.
Note that the worstLoadStateAtReferenceLocation flight field is only authorized for a user if he is also authorized to use the TrafficCounts.
See `FlightListRequest.worstLoadStateAtReferenceLocationType`

ck) **DeltaEntry compareWithOtherTrafficType** *(Optional)*
Comparison between this flight in FlightListRequest.trafficType and the same flight in FlightListRequest.compareWithOtherTrafficType. Remark: This attribute is subject to the validation rules `FlightListRequest.REQUESTED_FIELD_NOT_ALLOWED_FOR_OPERATION` and `FlightRetrievalRequest.REQUESTED_FIELD_NOT_ALLOWED_FOR_OPERATION`.

cl) **CTOTLimitReason ctotLimitReason** *(Optional)*
Possible exceptional reasons that may affect the CTOT allocation of a flight. See `CTOTLimitReason` class documentation for more details.

cm) **ProfileValidity profileValidity** *(Optional)*
Contains data relating to the validity of the RTFM, or else the FTFM with respect to Flight Plan violation errors.

(5) Used by: FlightOrFlightPlan, FlightRetrievalReply.

4.68. FlightAirspace

(1) Describes an airspace in a flight airspace profile.

(2) Attributes:

a) **AirspaceId airspaceId** *(Mandatory)*
Airspace id.

b) **AirspaceType airspaceType** *(Mandatory)*
Airspace type.

c) **DateTimeSecond firstEntryTime** *(Mandatory)*
Time of first entry of the flight in the airspace.

d) **FlightLevel firstEntryFlightLevel** *(Mandatory)*
The flight level at the first entry in the airspace.

e) **FlightLevel lastExitFlightLevel** *(Mandatory)*
The flight level at the last exit from the airspace.

f) **FlightTrend firstEntryTrend** *(Mandatory)*
The flight trend at the first entry in the airspace.

g) **FlightTrend middleTrend** *(Mandatory)*
The flight trend at the middle of the airspace.

h) **DistanceNM firstEntryDistance** *(Mandatory)*
Distance flown at the first entry of the flight in the airspace.

i) **DateTimeSecond lastExitTime** *(Mandatory)*
Time of last exit of the flight from the airspace.

j) **FlightTrend lastExitTrend** *(Mandatory)*
The flight trend at the last exit from the airspace.

k) **DistanceNM lastExitDistance** *(Mandatory)*
Distance flown at the last exit of the flight from the airspace.

l) **DurationHourMinuteSecond occupancyDuration** *(Mandatory)*
Elapsed time flown in the airspace.

m) **DistanceNM occupancyDistance** *(Mandatory)*
Distance flown in the airspace.

n) **boolean activated** *(Mandatory)*
True if the airspace is a sector and the sector is activated at the moment of the flight.

(3) Used by: Flight.

4.69. <<enumeration>> FlightDataset

(1) Describes the flight-related datasets that one can request when retrieving detailed flight data.

(2) Values:

Value	Description
flight	the reply will return the Flight.
flightPlan	the reply will return the FlightPlan.
flightPlanHistory	the reply will return the FlightPlanHistory.

Table 4.6. <<enumeration>> FlightDataset

(3) Used by: FlightRetrievalRequest.

4.70. FlightEvent

(1) Describes an event acting on a flight and corresponding to an input message or an output message. All other events are filtered out.

(2) Attributes:

a) **DateTimeSecond timestamp** *(Mandatory)*
Event occurrence timestamp.

b) **FlightEventType type** *(Mandatory)*
Type of the flight event.

c) **FlightState resultingState** *(Mandatory)*
Flight state resulting from the event.

d) **DateTimeMinute resultingOffBlockTime** *(Mandatory)*
Off-block time resulting from the event.

e) **boolean efdSent** *(Mandatory)*
Indicates if an EFD was sent.

f) **boolean fumSent** *(Mandatory)*
Indicates if a FUM was sent.

(3) Used by: Flight.

4.71. <<enumeration>> FlightEventType

(1) Describes the type of an event in a flight history.

(2) Values:

Value	Description
ACH	ATC flight plan CHange. The ATC flight plan change (ACH) is that change message type distributed by the IFPS upon receipt and successful processing of an FNM, MFS, and AFP for which a valid associated flight plan exists in the IFPS.
ADI	Anticipated DPI message
AFI	AIR-FILED flight plans. Air Filed Flight plans (AFIL) represent flight plans submitted by an ATS unit to the IFPS for processing on behalf of an aircraft already in flight.
APL	ATC flight PLan. The ATC flight plan (APL) is that flight plan message type distributed by the IFPS upon receipt and successful processing of an FNM, MFS, and AFP for which no valid associated flight plan exists in the IFPS.
APR	AO Position Reports message. For some flights departing from outside the ECAC area, AOs provide information on their estimated time of arrival.
ATT	Take-off from.
AXT	Taxi from.
CAL	Operator Runway Update.
CDC	CASA Delay Confirmation suspension with booking.
CDI	Cancel DPI message.
CEO	EOBT Change for simulation or predict reason.
CMC	CCAMS Monitoring: Predicted Conflict.
CMN	CCAMS Monitoring: No Conflict Predicted.
CNC	CASA slot allocation/update.
CPR	Correlated Position Report message. CPRs are extracted from surveillance data (radar derived positions).
CPT	Placed on time over by CASA.

Value	Description
CRL	CCAMS Code Release Message.
CRQ	CCAMS Code Request Message
CSC	CASA Slot Cancellation.
CSU	CCAMS Ssr Info assignment/update.
CTD	Target DPI Confirmation.
EDI	Early DPI message.
EMR	Error Message.
FCM	Flight Confirmation Message when: a) An AO indicates to ETFMS the RVR capability of a flight with an EOBT in the future.Flight Confirmation Message. b) An AO indicates to ETFMS that a flight with an EOBT in the future is now confirmed for the regulation(s) provided in this FCM. c) An AO indicates to ETFMS that a flight with an EOBT in the future is now confirmed for the regulation(s) provided in this FCM.
FDI	FAA ASDI Departure Information message.
FLS	FLight Suspension message. Flight suspension until further notice. In case of: a) Aerodrome closure. b) Runway Visual Range (RVR) unknown. c) Reception of an SMM message. d) Not reported as airborne. e) Suspended by departure airport (cancel DPI message).
FSA	First System Activation message. FSA is a message designed to enable ATC systems to automatically inform NM of significant events affecting a flight. The FSA message can only be sent by ATC and is normally generated automatically by an ATC system.
FUM	Flight Update Message. The flight status information sent to the IFPS by the ETFMS.
IAR	IFPS ARrival message. Indicates the arrival of a flight plan.
ICA	IFPS CAncel message. Indicates the cancel a flight plan by IFPS.
ICH	IFPS CHange message. Updates certain items of a flight plan by IFPS.
IDE	IFPS DEparture message. Indicates the departure of a flight plan by IFPS.
IDL	IFPS DeLay message. Indicates a delay for the departure of a flight plan by IFPS.
IFP	IFPS Flight Plan message. Indicates the creation of a flight plan by IFPS.

Value	Description
MET	Meteo update.
MSG	Output of ATFM message(s).
NEV	No event.
OAR	ATFM Rerouting.
OCA	Operator Cancellation.
OCM	Operator Change Manual suspension.
ODA	Operator De-Activation.
OEX	Exempt / de-exempt from regulation.
OIC	Operator Confirmation.
ORX	Exclude from / re-include in regulation
PTD	Provisional Target DPI.
PTX	EFD Message Periodic Transmission.
REA	Update Readiness message.
RFR	Re-route after reroute cancellation.
RJT	Rerouteing reJecTion (RJT) message. An RJT is a negative response to a Rerouteing Proposal (RRP) message.
RLR	Remove Loaded RPL.
RPL	RPL message.
RRM	Rerouting Proposal Creation.
RSI	CASA Revoke slot proposal.
RSU	Restriction Update.
RWI	CASA What if result.
SCA	Strat Cancel.
SCM	CCAMS Sync CCM.
SIP	CASA booking (SIP).
SIT	CASA SIT time out.
SMM	Slot Missed Message. An SMM is sent when the last received CTOT issued cannot be met and a new EOBT is NOT known.
SPA	Slot improvement Proposal Acceptance (SPA) message. An SPA is a positive response to a SIP which is received from NM. The AO will send an SPA if the proposed NEWCTOT in the SIP is acceptable.
SRJ	Slot improvement proposal ReJection (SRJ) message. An SRJ is a negative response to a SIP received from NM. The AO will send an SRJ if they are unable to accept the proposed improvement.
SSC	CCAMS Sync SLC.
SSM	CCAMS Sync SAM.

Value	Description
SSP	CASA Stam Proposal
SSR	CCAMS Sync SRM.
SUS	CASA RVR suspension without booking.
TAC	Assign Ssr Code time out.
TAM	Activation monitoring time out.
TDE	Departure time out.
TDI	Target DPI message.
TPF	PFD cancellation time out.
TRC	Release Ssr Code time out.
TRM	Resend Last Message time out.
TSA	Target Start-up Approval DPI message.
TSC	Simulate SSR Code time out.
TTE	Termination time out.
UCD	Update MCDM Data
UFA	Undo Flight Activation Monitoring.
UXC	Undo XCD Effect (suspension/shift).
XCR	Created for simulation or predict reason.

Table 4.7. <<enumeration>> FlightEventType

(3) Used by: FlightEvent.

4.72. <<enumeration>> FlightField

(1) Enumerates the fields that the caller may request to be returned in Flight objects.

(2) The NM system associates a weight to flight fields: a flight field is either "light" or "heavy", in the sense that heavy flight fields are significantly more demanding to return than light ones. NM kindly requests its customers to apply the following strategy:

(3) a) As a rule, client applications should never request flight fields that they do not need, regardless to their weight.

b) Client applications typically implement a query/retrieve pattern:

i) Query the small number of most relevant flight fields to display to the end user.

ii) Retrieve more details for a given flight (using the FlightRetrievalRequest) when the end user has selected a flight from the list.

c) In particular, the client application should avoid requesting heavy fields in a flight list if not strictly necessary.

d) However, in case one or more heavy fields are strictly necessary in the flight list, the customer is invited to request indeed the heavy fields in the flight list rather than querying first a "light" flight list and then iterating on the flight list and retrieving each individual flight to get the heavy ones.

(4) Values:

Value	Weight	Authorisation Level
actualTakeOffTime	Light	Service
actualTimeOfArrival	Light	Service
aircraftOperator	Light	Service
aircraftType	Light	Service
atfcmMeasureLocations	Heavy	Field
calculatedTakeOffTime	Light	Service
calculatedTimeOfArrival	Light	Service
caughtInHotspots	Light	Field
ccamsSSRCode	Light	Service
cdm	Light	Service
cdmEstimatedOffBlockTime	Light	Service
cfmuFlightType	Light	Service
compareWithOtherTrafficType	Light	Field
confirmedCTFM	Heavy	Service
ctfmAirspaceProfile	Heavy	Service
ctfmPointProfile	Heavy	Service
ctfmRequestedFlightLevels	Heavy	Service
ctfmRestrictionProfile	Heavy	Field
ctotLimitReason	Light	Field
ctotShiftAlreadyAppliedByTower	Light	Service
currentlyUsedTaxiTime	Light	Service
defaultReroutingRequestedFlightLevel	Heavy	Service
defaultReroutingRequestedSpeed	Heavy	Service
delay	Light	Service
delayCharacteristics	Light	Service
departureTolerance	Heavy	Field

Value	Weight	Authorisation Level
divertedAerodromeOfDestination	Light	Service
equipmentCapabilityAndStatus	Heavy	Service
estimatedElapsedTime	Heavy	Service
estimatedTakeOffTime	Light	Service
estimatedTimeOfArrival	Light	Service
exclusionFromRegulations	Light	Service
exemptedFromRegulations	Light	Service
famStatus	Light	Service
filedRegistrationMark	Light	Service
filingRule	Heavy	Service
flightHistory	Heavy	Service
flightLevelAtReferenceLocationEntry	Light	Field
flightLevelAtReferenceLocationExit	Light	Field
flightPlanOriginator	Heavy	Service
ftfmAirspaceProfile	Heavy	Service
ftfmPointProfile	Heavy	Service
ftfmRequestedFlightLevels	Heavy	Field
ftfmRestrictionProfile	Heavy	Field
hasBeenForced	Light	Field
hasOtherRegulations	Light	Service
hotspots	Heavy	Field
icaoRoute	Heavy	Service
isProposalFlight	Light	Field
lastATFMMessageOriginator	Heavy	Service
lastATFMMessageReceivedOrSent	Light	Service
lastATFMMessageType	Light	Service
lateFiler	Light	Service
lateUpdater	Light	Service
mcdmInfo	Light	Field
mostPenalisingRegulation	Light	Service
mostPenalisingRegulationCause	Heavy	Service
newRouteMinShiftDelayImprovement	Light	Service
operatingAircraftOperator	Light	Service

Value	Weight	Authorisation Level
operationalLog	Heavy	Service
profileValidity	Light	Field
proposalExists	Light	Field
readyEstimatedOffBlockTime	Light	Service
readyStatus	Light	Service
regulationLocations	Light	Service
requestedFlightLevel	Light	Service
requestedInitialFlightLevel	Heavy	Service
requestedInitialSpeed	Heavy	Service
reroutable	Light	Service
reroutingIndicator	Light	Service
reroutingOpportunitiesExist	Light	Service
reroutingReference	Heavy	Service
revisionTimes	Light	Service
routeLength	Heavy	Service
rtfmAirspaceProfile	Heavy	Service
rtfmPointProfile	Heavy	Service
rtfmRequestedFlightLevels	Heavy	Service
rtfmRestrictionProfile	Heavy	Field
runwayVisualRange	Heavy	Service
slotImprovementProposal	Light	Service
slotIssued	Light	Service
suspensionResponseBy	Light	Service
suspensionStatus	Light	Service
taxiTime	Light	Service
timeAtReferenceLocationEntry	Light	Field
timeAtReferenceLocationExit	Light	Field
trendAtReferenceLocationEntry	Light	Service
trendAtReferenceLocationExit	Light	Service
trendAtReferenceLocationMiddle	Light	Field
worstLoadStateAtReferenceLocation	Light	Field

Table 4.8. <<enumeration>> FlightField

(5) Used by: FlightListRequest, FlightRetrievalRequest.

4.73. FlightIATAId

(1) Flight identification structure as defined by IATA, also known as UFI (Unique Flight Identifier) as decribed in the AIDX (Aviation Information Data Exchange) standard.

(2) Examples:

(3)
a) carrierIdentification - BA

b) iataFlightNumber - 066

c) suffix - Z

d) departureAerodromeReference - PHL

e) destinationAerodromeReference - LHR

f) originDate - 2012-03-24

g) repeatNumber - 1

(4) Attributes:

a) **AircraftIATAId aircraftIATAId** *(Optional)*
Concatenation of the `carrierIdentification`, the `iataFlightNumber` and optionally a `suffix`.

b) **AerodromeIATAId departureAerodromeReference** *(Optional)*
Code of scheduled departure airport usually IATA but can be ICAO or other as defined in the Schedule [AIDX, UFI].

c) **AerodromeIATAId destinationAerodromeReference** *(Optional)*
Code of scheduled arrival airport usually IATA but can be ICAO or other as defined in the Schedule [AIDX, UFI].

d) **DateTimeMinute originDate** *(Optional)*
Scheduled flight origin date based on the flight as defined in the Schedule [AIDX, UFI].

e) **int repeatNumber** *(Optional)*
Repeat or departure attempt.
Constraint: Range: `[0,∞[`.

4.74. FlightIdentificationInput

(1) Identification of a flight, assumed to be unique.

(2) Choices:

a) **IFPLId id**
Unique IFPL id.

b) **FlightKeys keys**
Flight keys.

(3) Used by: FlightRetrievalRequest.

4.75. FlightIdentificationOutput

(1) Identification of a flight, assumed to be unique.

(2) Attributes:

a) **IFPLId id** *(Optional)*
Unique IFPL id.

b) **FlightKeys keys** *(Optional)*
Flight keys.

(3) Used by: Flight, FlightPlanOrInvalidFiling.

4.76. FlightKeys

(1) Represents the keys that uniquely identify a flight in the absence of an IFPL id.

(2) Attributes:

a) **ExtendedAircraftICAOId aircraftId** *(Mandatory)*
Aircraft id, can be an ICAO aircraft id or a special aircraft id containing '$' or '#' characters used by NM in the context of prediction and simulation exercises.

b) **AerodromeICAOId aerodromeOfDeparture** *(Optional)*
ICAO id of the aerodrome of departure.
Constraints:

i) See ADEP_NONICAOADEP_AND_AIRFILED_ARE_MUTUALLY_EXCLUSIVE

ii) See ONE_OF_ADEP_NONICAOADEP_OR_AIRFILED_IS_MANDATORY

c) **boolean nonICAOAerodromeOfDeparture** *(Optional)*
True if the aerodrome of departure is not an ICAO one.
Constraints:

i) See ADEP_NONICAOADEP_AND_AIRFILED_ARE_MUTUALLY_EXCLUSIVE

ii) See ONE_OF_ADEP_NONICAOADEP_OR_AIRFILED_IS_MANDATORY

d) **boolean airFiled** *(Mandatory)*
True if the flight plan was filed airborne.

Constraints:

i) See ADEP_NONICAOADEP_AND_AIRFILED_ARE_MUTUALLY_EXCLUSIVE

ii) See ONE_OF_ADEP_NONICAOADEP_OR_AIRFILED_IS_MANDATORY

e) **AerodromeICAOId aerodromeOfDestination** *(Optional)*
ICAO id of the aerodrome of destination.
Constraints:

i) See ADES_CANNOT_BE_NULL_IF_NOT_NONICAOADES

ii) See ADES_MUST_BE_NULL_IF_NONICAOADES

f) **boolean nonICAOAerodromeOfDestination** *(Optional)*
True if the aerodrome of destination is not an ICAO one.
Constraints:

i) See ADES_CANNOT_BE_NULL_IF_NOT_NONICAOADES

ii) See ADES_MUST_BE_NULL_IF_NONICAOADES

g) **DateTimeMinute estimatedOffBlockTime** *(Mandatory)*
Estimated off-block date/time according to the latest processed flightplan message (by IFPS). So after each flight plan, the `flightkeys.estimatedOffBlockTime` can change. Note that this off-block date/time does not necessarily corresponds to the off-block date/time of the FTFM flight profile (See : Flight.ftfmPointProfile).

(3) Constraints:

a)

Name	ONE_OF_ADEP_NONICAOADEP_OR_AIRFILED_IS_MANDATORY
Attributes	aerodromeOfDeparture, nonICAOAerodromeOfDeparture, airFiled
Description	One of the attributes aerodromeOfDeparture, nonICAOAerodromeOfDeparture or airFiled must be specified.

b)

Name	ADEP_NONICAOADEP_AND_AIRFILED_ARE_MUTUALLY_EXCLUSIVE
Attributes	aerodromeOfDeparture, nonICAOAerodromeOfDeparture, airFiled
Description	Attributes aerodromeOfDeparture, nonICAOAerodromeOfDeparture and airFiled are mutually exclusive. When aerodromeOfDeparture is specified, nonICAOAerodromeOfDeparture and airFiled must be set to false. When aerodromeOfDeparture is not specified, one and only one of nonICAOAerodromeOfDeparture and airFiled must be set to true

c)

Name	ADES_MUST_BE_NULL_IF_NONICAOADES

Attributes	aerodromeOfDestination, nonICAOAerodromeOfDestination
Description	Attribute aerodromeOfDestination must be null if nonICAOAerodromeOfDestination is true.

d)

Name	ADES_CANNOT_BE_NULL_IF_NOT_NONICAOADES
Attributes	aerodromeOfDestination, nonICAOAerodromeOfDestination
Description	Attribute aerodromeOfDestination cannot be null if nonICAOAerodromeOfDestination is false.

(4) Used by: RegulationForceDelayReply, InvalidFiling, AddFlightsToMeasureReply, FlightIdentificationOutput, RemoveFlightsFromMeasureReply, MCDMFlightTopic, FlightIdentificationInput, RemoveFlightsFromMeasureRequest, FlightInMeasure.

4.77. <<abstract>> FlightListByLocationReplyData

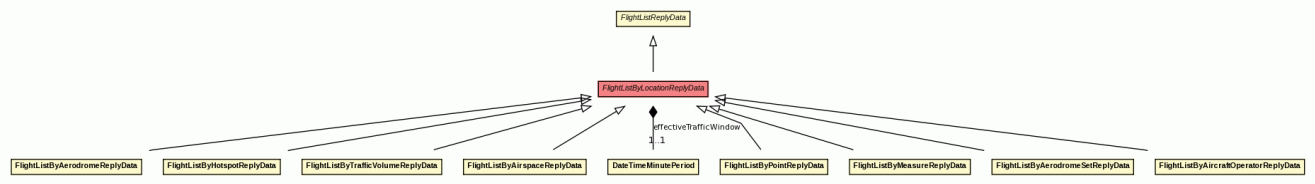

Figure 4.7. <<abstract>> FlightListByLocationReplyData Class Diagram

(1) Abstract reply returned in response to an abstract FlightListByLocationRequest.

(2) Inherits from: FlightListReplyData.

(3) Attributes:

a) **DateTimeMinutePeriod effectiveTrafficWindow** *(Mandatory)*
The effective period of time for which counts/flights were requested: flights from within this periods have been used in the flightlist/counts. This is the rounded and/or extended request trafficWindow (based on the countsInterval attribute). See also FlightListRequest.trafficWindow.

(4) Extended by: FlightListByHotspotReplyData, FlightListByMeasureReplyData, FlightListByTrafficVolumeReplyData, FlightListByPointReplyData, FlightListByAerodromeReplyData, FlightListByAerodromeSetReplyData, FlightListByAirspaceReplyData, FlightListByAircraftOperatorReplyData.

4.78. <<abstract>> FlightListByLocationRequest

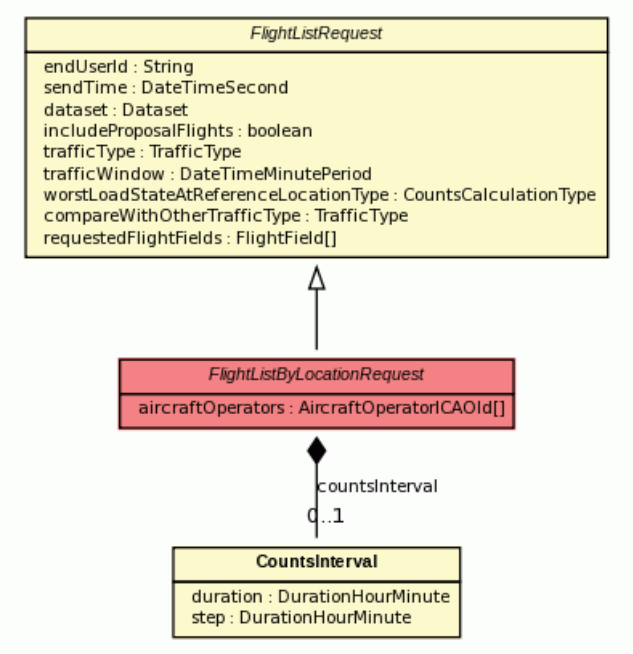

Figure 4.8. <<abstract>> FlightListByLocationRequest Class Diagram

(1) Abstract request to query a flight list by location.

(2) The logical AND operator applies between all the query fields described below and the query fields of its ancestor request.

(3) Inherits from: FlightListRequest.

(4) Attributes:

a) **CountsInterval countsInterval** *(Optional)*
Determines how the effectiveTrafficWindow needs to be derived from the FlightListRequest.trafficWindow.
By default, the countsInterval.duration is 1 minute and the countsInterval.step is 1 minute.
Constraint: See COUNTS_INTERVAL_MUST_BE_NULL

b) **AircraftOperatorICAOId[] aircraftOperators** *(Optional)*
Array of aircraft operator ICAO Id(s) for which flights are requested. Mandatory: the array is empty if there is no such aircraft operator ICAO id.
Constraint: Size must be comprised between 0 and ∞.

(5) Constraint:

a)

Name	COUNTS_INTERVAL_MUST_BE_NULL

Attribute	countsInterval
Description	The attribute countsInterval must be null when this request is a FlightListByHotspotRequest.

(6) Extended by: FlightListByHotspotRequest, FlightListByMeasureRequest, FlightListByTrafficVolumeRequest, FlightListByAerodromeRequest, FlightListByAircraftOperatorRequest, FlightListByPointRequest, FlightListByAirspaceRequest, FlightListByAerodromeSetRequest.

4.79. <<enumeration>> FlightListByMeasureMode

(1) Flight List By Measure Mode.

(2) For a regulation the concerned flights are those flights that use a regulation slot. However not all of them have an actual delay/have received a slot allocation message (typically exempted flights do not get regulated in a normal regulation (non-exceptional-conditions regulation). For a regulation, the flights that the measure has impacted (measure activated), are a subset of those flights: only those flights that did get a delay (can be 0 minutes) and have/will receive a SAM (Slot Allocation Message).

(3) For a rerouting/MCDM-only measure, the concerned flights are those flights that cross the location/traffic volume during the period on the optional flow, while the flights that the measure has impacted (measure activated: activated_by_measure), are a subset of those flights: only those flights that have been cherry picked for the rerouting/MCDM-only measure. Note that even if a flight has been cherry picked for a rerouting, it does not mean that the rerouting could find an alternate route/improvement (the result can be found inside the flight field: FlightAtfcmMeasureLocation).

(4) Values:

a) **ACTIVATED_BY_MEASURE**

b) **CONCERNED_BY_MEASURE**

(5) Used by: FlightListByMeasureRequest.

4.80. <<abstract>> FlightListReplyData

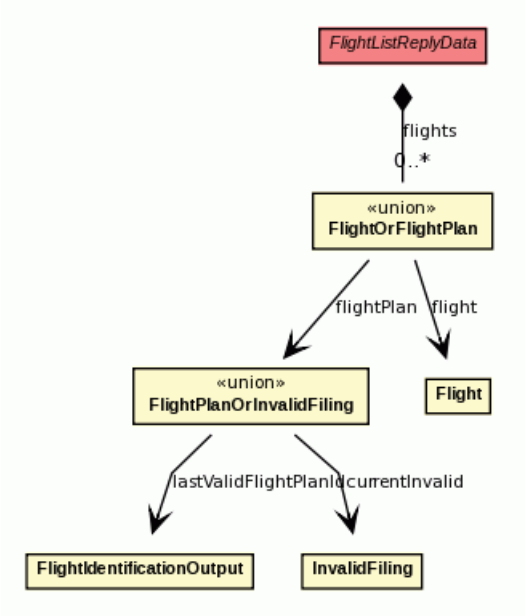

Figure 4.9. <<abstract>> FlightListReplyData Class Diagram

(1) Abstract reply returned in response to an abstract FlightListRequest.

(2) Attributes:

a) **FlightOrFlightPlan[] flights** *(Mandatory)*
The requested flights (together with their invalid filing summary if requested - this feature is only accessible via FlightListByKeysRequest).
Mandatory: if no match was found, the returned array is empty.
The array does not contain null or duplicate items.
Constraint: Size must be comprised between 0 and ∞.

(3) Extended by: FlightListByKeysReplyData, FlightListByLocationReplyData.

4.81. <<abstract>> FlightListRequest

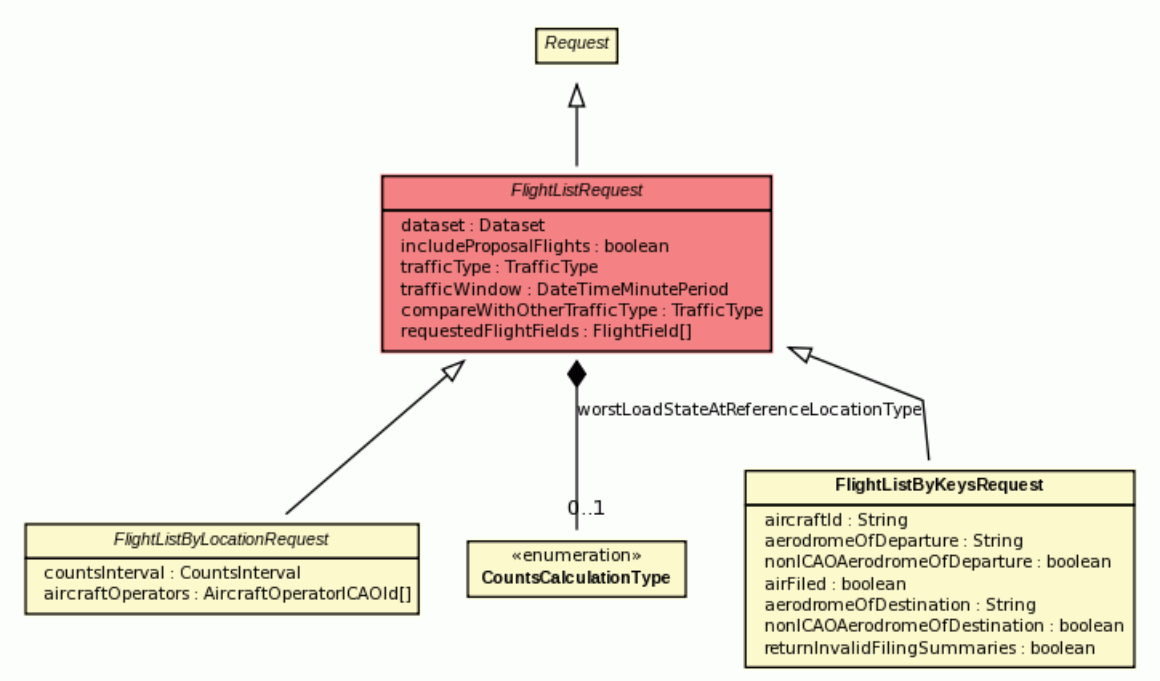

Figure 4.10. <<abstract>> FlightListRequest Class Diagram

(1) Abstract request to query an NM flight list, possibly together with the invalid filing messages.

(2) The logical AND operator applies between all the query fields described below.

(3) It is important to note that NM associates a weight to flight fields (see FlightField definition): a flight field is either "light" or "heavy", in the sense that heavy flight fields are significantly more demanding to return than light ones. NM kindly requests its customers to apply the following strategy:

(4)
a) As a rule, client applications should never request flight fields that they do not need, regardless of their weight

b) Client applications typically implement a query/retrieve pattern:

   i) Query the small number of most relevant flight fields to display to the end user.

   ii) Retrieve more details for a given flight (using the FlightRetrievalRequest) when the end user has selected a flight from the list

   The client application should not request flight fields in the flight list if these fields are not necessary for the end user to make his selection.

c) In particular, the client application should avoid requesting heavy fields in a flight list if not strictly necessary.

d) However, in case one or more heavy fields are strictly necessary in the flight list, the customer is invited to request indeed the heavy fields in the flight list rather than querying first a "light" flight list and then iterating on the flight list and retrieving each individual flight to get the heavy ones.

(5) Inherits from: `Request`.

(6) Attributes:

a) **`Dataset` `dataset`** *(Mandatory)*
Dataset from which the flight list is requested.
See Forecast and Operational Datasets and Simulation Datasets.
Constraint: See `INVALID_QUERY_PERIOD_RANGE`

b) **`boolean` `includeProposalFlights`** *(Mandatory)*
Determines if the selected traffic must include the proposal flights, or only the \"real\" flights. If the proposal flights are included, they replace their corresponding \"real\" flights.
See Proposal Flights.

c) **`TrafficType` `trafficType`** *(Mandatory)*
Traffic type for which flights are requested.
Returned flights are according to the "highest" available Tactical Flight Model:

i) If the requested traffic type is `TrafficType.LOAD`, returned flights are according to the "CTFM" (Current Tactical Flight Model) model if available; otherwise, according to the "RTFM" (Regulated Tactical Flight Model) model if available; otherwise, according to the "FTFM" (Filed Tactical Flight Model) model. Note however that suspended flights are never returned in the flight list obtained with `TrafficType.LOAD`.

ii) If the requested traffic type is `TrafficType.REGULATED_DEMAND`, returned flights are according to the "RTFM" (Regulated Tactical Flight Model) model if available; otherwise, according to the "FTFM" (Filed Tactical Flight Model) model. Note however that suspended flights are never returned in the flight list obtained with `TrafficType.REGULATED_DEMAND`.

iii) If the requested traffic type is `TrafficType.DEMAND`, returned flights are according to the "FTFM" (Filed Tactical Flight Model) model.

Constraint: See `COMPARE_WITH_OTHER_TRAFFIC_TYPE_INVALID_VALUE`

d) **`DateTimeMinutePeriod` `trafficWindow`** *(Contextual)*
The period of time for which flights are requested.
The meaning of the traffic window depends on the actual request type:

i) If the actual request is a FlightListByLocationRequest, and depending on the concrete request type, the traffic window specifies that only those flights taking off and/or landing, or being over a point, or entering a sector in the time window are returned. Alternatively if an `OCCUPANCY` calculationType is requested(applicability depending on the concrete request type), the traffic window specifies that only those flights oc-

cupying a sector or being airborne in the time window are returned (depending on the concrete request type).
Note that for a FlightListByLocationRequest, the trafficWindow is first (optionally) rounded and/or extended based on the countsInterval attribute.
The result is the `effectiveTrafficWindow` in the FlightListByLocationReplyData.
If a countsInterval is used with step 1 minute and duration 1 minute, then no rounding/extending takes place, and the `effectiveTrafficWindow` is the request trafficWindow.
The idea behind: if a user does a TrafficCountRequest with a trafficWindow from `[10:00,10:01[` with a `countsInterval` with step 1 and duration 10 minutes, the single count period returned has a duration of 10 minutes (`[10:00,10:10[`).
If the users want to list corresponding flights, then he can do a flightlist with trafficWindow `[10:00,10:01[` and a countsInterval with step 1 and duration 10 minutes. (i.e. the parameters that were used in the TrafficCounts request) to get exactly those flights corresponding to the counts.
So more general: the countsInterval attribute allows to round/extend the request trafficWindow to get the flights corresponding to the TrafficCounts (corresponding to the one or more count periods of the TrafficCountsReplyData).

ii) If the actual request is a FlightListByKeysRequest, the traffic window specifies that only those flights having an estimated off-block time in the period are returned.

Presence:

i) Must be null in `FlightListByHotspotRequest`

ii) Mandatory otherwise.

Constraints:

i) See `INVALID_QUERY_PERIOD_RANGE`

ii) See `PERIOD_EXTENSION_CANNOT_BE_GREATER_THAN_24_HOURS`

e) **`CountsCalculationType worstLoadStateAtReferenceLocationType`** *(Optional)*
When `requestedFlightFields.worstLoadStateAtReferenceLocation` is requested, then this CountsCalculationType indicates how the field needs to be computed :based on entry counts (with capacity values) or based on occupancy counts (with OTMV).
So basically, the worstLoadStateAtReferenceLocation field indicates if a flight contributes to an overload counts period(for an entry CountsCalculationType) or if the flight contributes to a peak/sustained flight overload (w.r.t. OTMV) occupancy count period (for an occupancy CountsCalculationType).
This info is useful when selecting flights to STAM.
Note that the `worstLoadStateAtReferenceLocation` flight field is only authorized for a user if he is also authorized to use the `TrafficCounts`.
Constraint: See `WORST_LOAD_STATE_AT_REFERENCE_LOCATION_TYPE_PRESENCE`

f) **`TrafficType compareWithOtherTrafficType`** *(Optional)*
When `requestedFlightFields.compareWithOtherTrafficType` is requested, then this `TrafficType` indicates how the field needs to be computed.

Basically, the compareWithOtherTrafficType field allows comparing two flight-lists: one with trafficType and the other with `compareWithOtherTrafficType`.
The results are shown in this `compareWithOtherTrafficType` field: it shows for a flight in trafficType where is the flight in `compareWithOtherTrafficType` w.r.t. timeOver, lateral deviation and vertical deviation and if it is an intruder or not for the queried location.
Constraints:

i) See COMPARE_WITH_OTHER_TRAFFIC_TYPE_INVALID_VALUE

ii) See COMPARE_WITH_OTHER_TRAFFIC_TYPE_PRESENCE

g) **FlightField[] requestedFlightFields** *(Optional)*
The reply returns only the requested flight fields in this array, and only if the values of these requested fields are available at NM. Note that the flight keys are always returned.
Optional: default is the empty array (used if only flight plan filing summary is requested).
Cannot be null or empty if the concrete request does not add any other data request.
Constraints:

i) Size must be comprised between 0 and ∞.

ii) See CANNOT_REQUEST_OPERATIONAL_LOG

iii) See COMPARE_WITH_OTHER_TRAFFIC_TYPE_PRESENCE

iv) See REQUESTED_FIELD_NOT_ALLOWED_FOR_OPERATION

v) See REQUESTED_FIELDS_CANNOT_CONTAIN_DUPLICATE

vi) See WORST_LOAD_STATE_AT_REFERENCE_LOCATION_TYPE_PRESENCE

(7) Constraints:

a)

Name	PERIOD_EXTENSION_CANNOT_BE_GREATER_THAN_24_HOURS
Attribute	trafficWindow
Description	The period extension must be smaller or equal to 24 hours.

b)

Name	REQUESTED_FIELDS_CANNOT_CONTAIN_DUPLICATE
Attribute	requestedFlightFields
Description	If specified, the array cannot contain duplicates.

c)

Name	CANNOT_REQUEST_OPERATIONAL_LOG
Attribute	requestedFlightFields
Description	The field `operationalLog` is not valid in the FlightListRequest.

d)

Name	WORST_LOAD_STATE_AT_REFERENCE_LOCATION_TYPE_PRESENCE
Attributes	worstLoadStateAtReferenceLocationType, requestedFlightFields
Description	The field worstLoadStateAtReferenceLocationType must be set if and only if the requestedFields.worstLoadStateAtReferenceLocation is requested.

e)

Name	COMPARE_WITH_OTHER_TRAFFIC_TYPE_PRESENCE
Attributes	compareWithOtherTrafficType, requestedFlightFields
Description	The field compareWithOtherTrafficType must be set if and only if the requestedFields.compareWithOtherTrafficType is requested.

f)

Name	COMPARE_WITH_OTHER_TRAFFIC_TYPE_INVALID_VALUE
Attributes	trafficType, compareWithOtherTrafficType
Description	The two attributes trafficType and compareWithOtherTrafficType cannot have the same value.

g)

Name	REQUESTED_FIELD_NOT_ALLOWED_FOR_OPERATION
Attribute	requestedFlightFields
Description	The fields worstLoadStateAtReferenceLocation, compareWithOtherTrafficType are not valid in the FlightListByAircraftOperatorRequest and in the FlightListByKeysRequest. The fields timeAtReferenceLocation, timeAtReferenceLocationEntry, timeAtReferenceLocationExit, flightLevelAtReferenceLocation, flightLevelAtReferenceLocationEntry, flightLevelAtReferenceLocationExit, trendAtReferenceLocationEntry, trendAtReferenceLocationExit, trendAtReferenceLocationMiddle, worstLoadStateAtReferenceLocation, compareWithOtherTrafficType are not valid in the FlightListByMeasureRequest when FlightListByMeasureRequest.mode equals FlightListByMeasureMode.ACTIVATED_BY_MEASURE.

h)

Name	INVALID_QUERY_PERIOD_RANGE
Attributes	trafficWindow, dataset
Description	The dataset.type from which the measures are requested and the trafficWindow must be set according to the following rules: i) if the DatasetType is equals to FORECAST the trafficWindow shall be defined within the range [D-5, D]

	ii)	if the DatasetType is equals to OPERATIONAL the `trafficWindow` shall be defined within the range [D-1, D]

(8) Extended by: FlightListByLocationRequest, FlightListByKeysRequest.

4.82. `FlightOperationalLogEntry`

(1) Describes an entry in a flight operational log.

(2) Attributes:

a) **`DateTimeSecond` `timestamp`** *(Optional)*
The date/time when the entry was recorded.

b) **`FlightOperationalLogEntryType` `type`** *(Optional)*
The type of the operational log entry.

c) **`int etfmsId`** *(Optional)*
The ETFMS unique id of the flight.
Constraint: Range:] - ∞,∞[.

d) **`IFPLId` `ifplId`** *(Optional)*
The IFPL id of the flight, if any in ETFMS.

e) **`string issuer`** *(Optional)*
The identification of the issuer of the message, if any.

f) **`string message`** *(Optional)*
The detailed text of the message (if the entry is a detailed entry).

g) **`string[] summaryFields`** *(Optional)*
The summary fields (if the entry is a summary entry).
Constraint: Size must be comprised between 0 and ∞.

(3) Used by: Flight, MeasureOpLogRetrievalReply.

4.83. `<<enumeration>> FlightOperationalLogEntryType`

(1) Describes a flight operational log entry type.

(2) Values:

a) **`ENVIRONMENT_MESSSAGE`**

b) **`ERRONEOUS_INCOMING_MESSAGE`**

c) **`ERROR_MESSAGE`**

d) **`HISTORY`**

e) **INCOMING_MESSAGE**

f) **OUTGOING_MESSAGE**

g) **PROCESS_ERROR**

h) **TEXT_MESSAGE**

i) **UNDEFINED**

j) **USER_COMMAND**

k) **VIOLATION**

l) **WARNING**

(3) Used by: FlightOperationalLogEntry.

4.84. FlightOrFlightPlan

(1) For a given IFPLId, represents the flight or flight plan.

(2) In the latter case, the flight plan may contains a valid flight plan or invalid filings messages.

(3) Depending on what has been received by NM, the flight follows the new ICAO2012 standard (flight).

(4) Choices:

a) **Flight flight**
The flight.

b) **FlightPlanOrInvalidFiling flightPlan**
The valid flight plan or invalid filings messages.

(5) Used by: FlightListReplyData.

4.85. `FlightPerformanceData`

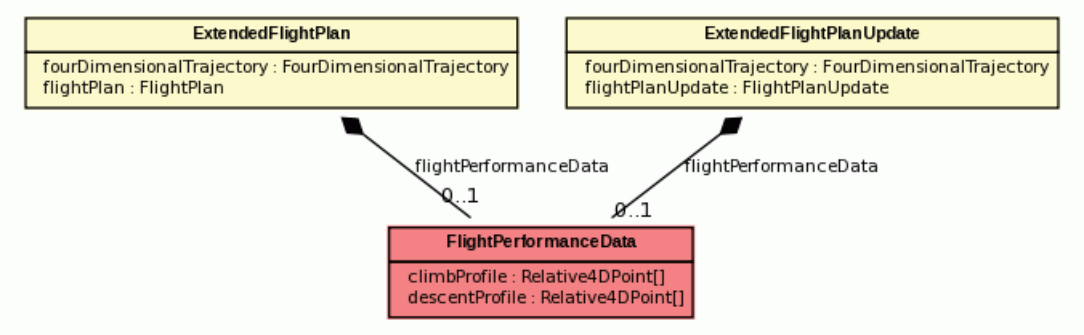

Figure 4.11. `FlightPerformanceData` Class Diagram

(1) It represents the climbing and descending capabilities of the aircraft specific to the flight, taking into account the performance of the airframe that is used to operate the flight as well as any other parameters that may influence it such as engine settings and status, cost factor applied by the operator.

(2) The climb and descent performance profiles are optimum and unconstrained climb and descent profiles instantiated per flight that satisfy the following conditions:

(3)
a) Are calculated without taking into account constraints regarding the vertical evolution of the flight such as route availability, RAD level restrictions, SID/STAR restrictions;

b) Are calculated without applying meteorological conditions (wind and temperature);

c) Are provided up to the maximum cruising level acceptable for the flight (even if not included in the flight plan). This would allow the recipient systems to generate accurate trajectories for vertical re-routings above the highest requested cruising level included in the filed flight plan. Performance profiles should be provided at least up to the highest requested cruising level given in the FPL

(4) Do not contain step-climbs and step-descents, i.e., if the aircraft is planned to do an initial climb to F350, then burn fuel during an hour of cruise, and then climb to F370, these two consecutive climbs shall be glued together.

(5) Attributes:

a) **`Relative4DPoint[] climbProfile`** *(Mandatory)*
The climb performance profile described as a sequence of points in which every point is defined by:

i) Cumulative Distance from the aerodrome of departure.

ii) Altitude above mean sea level (MSL) in feet (ft) or meters (m) or Flight level (FL).

iii) Cumulative Time elapsed from the aerodrome of departure.
Constraint: Size must be comprised between 2 and ∞.

b) **Relative4DPoint[] descentProfile** *(Mandatory)*
The descent performance profile described as a sequence of points, in reverse order starting from the aerodrome of destination, in which every point is defined by:

i) Cumulative Distance from the aerodrome of destination.

ii) Altitude above mean sea level (MSL) in feet (ft) or meters (m) or Flight level (FL).

iii) Cumulative Time elapsed from the aerodrome of destination.
Constraint: Size must be comprised between 2 and ∞.

(6) Used by: ExtendedFlightPlan, ExtendedFlightPlanUpdate.

4.86. FlightPlan

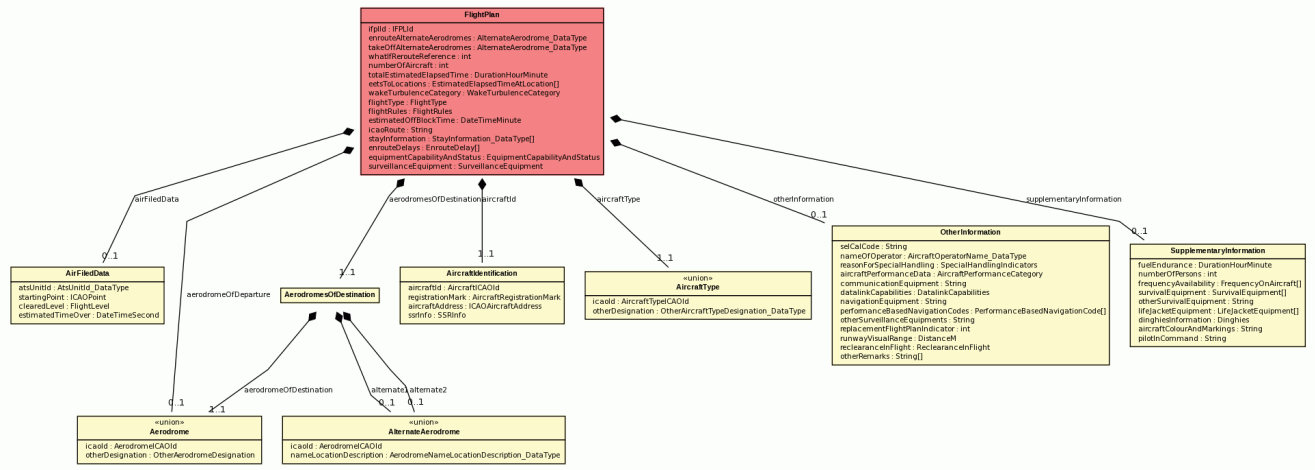

Figure 4.12. `FlightPlan` Class Diagram

(1) This set contains classes related to flight plan information.

(2) Note that the involved data types are highly inspired by the Flight Plan model proposed by the FO ICD for ICOG (see Flight Object ICD), where the CamelCase notation has replaced the original "_" notation.

(3) Attributes:

a) **IFPLId ifplId** *(Optional)*
Unique, opaque identifier assigned by the NM system to a submitted flight plan.
The attribute `FlightPlan.ifplId` is ignored in all request messages sent to NM.
When such `ifplId` information needs to be provided in a request message to NM, it will be done through a specific structure different from `FlightPlan` structure such as Flight-IdentificationInput.
The attribute `FlightPlan.ifplId` is never null when NM returns a submitted `FlightPlan` object.

b) **AirFiledData airFiledData** *(Contextual)*

Estimate data provided when the flight plan was filed airborne.
Presence:

i) Ignored in ExtendedFlightPlanValidationRequest, FlightPlanValidationRequest, RoutingAssistanceRequest

ii) Optional otherwise.
Constraint: See ADEP_AIRFILEDDATA_MUTUALLY_EXCLUSIVE

c) **Aerodrome aerodromeOfDeparture** *(Optional)*
Aerodrome of departure.
Constraint: See ADEP_AIRFILEDDATA_MUTUALLY_EXCLUSIVE

d) **AerodromesOfDestination aerodromesOfDestination** *(Contextual)*
Aerodromes of destination, including the alternates.
Presence:

i) Optional in ExtendedFlightPlanValidationRequest, FlightPlanValidationRequest, RoutingAssistanceRequest

ii) Mandatory otherwise.

e) **AlternateAerodrome_DataType enrouteAlternateAerodromes** *(Optional)*
Aerodromes where the aircraft may land in case of emergency along the route.

f) **AlternateAerodrome_DataType takeOffAlternateAerodromes** *(Optional)*
Aerodromes where the aircraft may land in case of emergency at take-off.

g) **AircraftIdentification aircraftId** *(Contextual)*
Information regarding the aircraft in this flight plan, i.e. the aircraft id but also other information like registration mark or SSR info.
Presence:

i) Optional in ExtendedFlightPlanValidationRequest, FlightPlanValidationRequest, RoutingAssistanceRequest

ii) Mandatory otherwise.

h) **int whatIfRerouteReference** *(Contextual)*
Indication of AO What-If rerouting reference in a flight plan.
Presence:

i) Ignored in ExtendedFlightPlanValidationRequest, FlightPlanValidationRequest, RoutingAssistanceRequest

ii) Optional otherwise.
Constraint: Range: `[1,9]`.

i) **int numberOfAircraft** *(Contextual)*
Number of aircraft in a formation flight.

Optional: default is 1.
Presence:

i) Ignored in RoutingAssistanceRequest

ii) Optional otherwise.
Constraint: Range: [1,99].

j) **AircraftType aircraftType** *(Mandatory)*
Aircraft type.

k) **DurationHourMinute totalEstimatedElapsedTime** *(Contextual)*
Total estimated elapsed time.
Presence:

i) Optional in RoutingAssistanceRequest

ii) Mandatory otherwise.

l) **EstimatedElapsedTimeAtLocation[] eetsToLocations** *(Contextual)*
Array of locations and the corresponding accumulated elapsed time to these locations. In case of FlightPlanUpdateRequest, the list of locations should be complete. Omitting a location will delete the location.
Presence:

i) Ignored in ExtendedFlightPlanValidationRequest, FlightPlanValidationRequest, RoutingAssistanceRequest

ii) Optional otherwise.
Constraint: Size must be comprised between 0 and ∞.

m) **WakeTurbulenceCategory wakeTurbulenceCategory** *(Contextual)*
Wake turbulence category.
Presence:

i) Optional in ExtendedFlightPlanValidationRequest, FlightPlanValidationRequest, RoutingAssistanceRequest

ii) Mandatory otherwise.

n) **FlightType flightType** *(Contextual)*
Type of the flight, e.g. scheduled, not scheduled, etc.
Presence:

i) Optional in ExtendedFlightPlanValidationRequest, FlightPlanValidationRequest, RoutingAssistanceRequest

ii) Mandatory otherwise.

o) **FlightRules flightRules** *(Contextual)*

Indicates if the rules applicable to the flight are visual (`FlightRules.VFR`), instrumented (`FlightRules.IFR`) or visual and then instrumented (`FlightRules.VFR_THEN_IFR`) or vice versa (`FlightRules.IFR_THEN_VFR`).
Presence:

i) Optional in `ExtendedFlightPlanValidationRequest`, `FlightPlanValidationRequest`, `RoutingAssistanceRequest`

ii) Mandatory otherwise.
Constraint: See `VFR_FLIGHT_RULES_NOT_SUPPORTED`

p) **`DateTimeMinute` `estimatedOffBlockTime`** *(Mandatory)*
Estimated off-block date/time.

q) **`string` `icaoRoute`** *(Mandatory)*
Represents the Flight Plan ICAO Route (Field15).

r) **`StayInformation_DataType[]` `stayInformation`** *(Contextual)*
Information concerning the type of activity (training, photographic mission, etc) to be performed during the stay periods mentioned in the route of the flight.
Item N in the array corresponds to the "STAY<N>" reference in the stay periods of the route, where N is in [1, 9].
The value of each item corresponds to the remark string in the ADEXP STAYINFO element.
In case of `FlightPlanUpdateRequest`, the list of activities should be complete. Omitting an activity will delete the activity.
Presence:

i) Ignored in `ExtendedFlightPlanValidationRequest`, `FlightPlanValidationRequest`, `RoutingAssistanceRequest`

ii) Optional otherwise.
Constraint: Size must be comprised between 0 and 9.

s) **`EnrouteDelay[]` `enrouteDelays`** *(Optional)*
Gives the list of delays or holdings planned at given points. Corresponds to the ICAO DLE/ field.
Note that in the EUR region the usage of the STAY within the route description is preferred to the DLE.
Constraint: Size must be comprised between 0 and ∞.

t) **`EquipmentCapabilityAndStatus` `equipmentCapabilityAndStatus`** *(Contextual)*
Represents the capability and status of the equipment of the aircraft of the flight.
For FlightPlanValidationRequest and RoutingAssistanceRequest, the default value corresponds to "-ADEXP SWY": this value is used if the attribute is null or if it does not contain any attribute (equipment type) marked as `EquipmentStatus.EQUIPPED`. All EquipmentStatus values that are not `EquipmentStatus.EQUIPPED` are ignored.
Presence:

i) Optional in ExtendedFlightPlanValidationRequest, FlightPlanValidationRequest, RoutingAssistanceRequest

ii) Mandatory otherwise.

u) **SurveillanceEquipment surveillanceEquipment** *(Contextual)*
Surveillance equipment of the aircraft of the flight.
For FlightPlanValidationRequest and RoutingAssistanceRequest, the default value of the surveillance equipment is "S".
Presence:

i) Optional in ExtendedFlightPlanValidationRequest, FlightPlanValidationRequest, RoutingAssistanceRequest

ii) Mandatory otherwise.

v) **OtherInformation otherInformation** *(Contextual)*
Any other flight data Items specified in the bilateral agreement.
Refer to ICAO 4444 field type 18 (Other information).
Presence:

i) Ignored in RoutingAssistanceRequest

ii) Optional otherwise.

w) **SupplementaryInformation supplementaryInformation** *(Contextual)*
Supplementary flight data. Refer to ICAO 4444 field type 19 (Supplementary information).
Presence:

i) Ignored in ExtendedFlightPlanValidationRequest, FlightPlanValidationRequest, RoutingAssistanceRequest

ii) Optional otherwise.

(4) Constraints:

a)

Name	ADEP_AIRFILEDDATA_MUTUALLY_EXCLUSIVE
Attributes	airFiledData, aerodromeOfDeparture
Description	airFiledData cannot be null if aerodromeOfDeparture is null; must be null otherwise. aerodromeOfDeparture is mandatory, except if airFiledData is not null, in which case it is optional. aerodromeOfDeparture is also optional in FlightPlanValidationRequest and RoutingAssistanceRequest.

b)

Name	VFR_FLIGHT_RULES_NOT_SUPPORTED
Attribute	flightRules

Context	FlightPlanCreationRequest, RoutingAssistanceRequest, ExtendedFlightPlanValidationRequest, FlightPlanValidationRequest
Description	The FlightRules.VFR is not supported.

(5) Used by: FlightPlanCreationRequest, FlightPlanUpdateReply, FlightPlanCreationReply, FlightPlanInput, ExtendedFlightPlan, FlightPlanUnion, FlightDelayReply, FlightRetrievalReply.

4.87. FlightPlanHistory

(1) History of a flight plan.

(2) Attributes:

a) **FlightPlanHistoryInfo[] infos** *(Optional)*
Description of the history
Constraint: Size must be comprised between 0 and ∞.

(3) Used by: FlightRetrievalReply.

4.88. FlightPlanHistoryInfo

(1) Information associated to the flight plan history.

(2) Attributes:

a) **DateTimeSecond timeStamp** *(Optional)*
Date and time of the checkpoint logging.

b) **string checkPoint** *(Optional)*

Checkpoint	Abbreviation	Possible mode	Description of the corresponding action
FUM Update	FU	A	NM Internal message.
Reprocess Suspended	RS	A, O	A flight plan is suspended by the automatic revalidation or by an operator.
Reprocess Advisory	RA	A	A flight plan is revalidated and has been invalidated in status REVAL_ADVISORY
Reprocess Compliant	RC	A	A flight plan becomes compliant as a result of automatic revalidation.
Force Compliant	FC	O	A flight plan is forced to the NORMAL (compliant) state by an operator action.
Create	CR	A,M,S	A message is successfully processed, and an flight plan is created.

Checkpoint	Abbreviation	Possible mode	Description of the corresponding action
Update	UP	A,M,S	A message is successfully processed, and the associated flight plan is updated.
Backup	BU	A,M,S	A message is received at the back-up IFPU.
Duplicate	DU	A	Exactly the same message was already received.
Invalid	IN	A,M	A message is processed and is invalid. It is passed to manual processing.
Reject	RE	A,M,S,O	A message is rejected.
Close FPD	CL	A,M,S,O	The flight plan is closed.
Multiple	MU	M	An invalid message contains more than one flight plan.
Delete Message	DE	O	An operator has deleted an invalid message.
Refer	RF	O	NM Internal event.
Escape	ES	O	NM Internal event.
Manual Transmit	MT	O	An operator has manually requested the transmission of a message.
Transmit	TO	A	Successful transmission of a message.
Transmit Fail	TF	A	Failed transmission of a message.
Edit	ED	O	Flight plan editor opened by an operator on an invalid message.
Associate	AS	O	Manual association done by an operator
Discard	DI	A	The processing of the message failed

Table 4.9. FlightPlanHistoryInfoDetailed.checkPoints description

c) **`string mode`** *(Optional)*

Mode	Abbreviation	Description
Auto	A	The checkpoint logged in *Auto* mode corresponds to an automatic action.
Man	M	The checkpoint logged in *Man* mode corresponds to an action performed by an NM operator when correcting an invalid flight plan.
Semi_Auto	S	The checkpoint logged in *Semi_Auto* mode corresponds to an action performed by an NM operator in the context of semi-automatic processing of invalid

Mode	Abbreviation	Description
		messages in the invalid group window after the correction of an initial message.

Table 4.10. FlightPlanHistoryInfoDetailed.mode description

d) **`string msgIn`** *(Optional)*
The type (FPL, CHG, ... of the received or processed message.

e) **`string msgOut`** *(Optional)*
The type of the transmitted message.

f) **`string[] addresses`** *(Optional)*
The address of the originator if `msgIn`; Addresses to which the `msgOut` is transmitted.
Constraint: Size must be comprised between 0 and ∞.

g) **`string detail`** *(Optional)*
The text of the message concerned by the action.

(3) Used by: FlightPlanHistory.

4.89. `FlightPlanInput`

(1) Flight plan information expressed in either a structural or a textual format.

(2) Choices:

a) **`FlightPlan structural`**
Object used when the flight plan data is input in a structured manner.

b) **`string textual`**
FPL message text used when the flight plan data is input via a string.
Two message formats are accepted: ICAO and ADEXP.

(3) Used by: FlightPlanValidationRequest, RoutingAssistanceRequest.

4.90. `<<enumeration>> FlightPlanMessageStatus`

(1) Enumerates possible flight plan message status.

(2) Values:

Value	Description
`DELETED`	the message has been deleted.
`DISCARD`	the processing of the message has failed (internal error).
`INVALID`	the message was queued for correction.

Value	Description
MUTIPLE	refers to an invalid message that contains more than one flight plan message (coming from AFTN/SITA).
REFERRED	the message was queued for correction (by supervisor).
REJECTED	the message was invalid and has been rejected.

Table 4.11. <<enumeration>> FlightPlanMessageStatus

(3) Used by: InvalidFiling.

4.91. <<enumeration>> FlightPlanMessageType

(1) Enumerates flight plan message types.

(2) Values:

Value	Description
ACH	Atc flight plan CHange
AFP	Airborne Flight Plan message
APL	Atc flight PLan
ARR	ARRival
CHG	CHanGe Message
CNL	CaNceLled flight plan
DEP	DEParture
DLA	DeLAyed flight plan
FNM	Flight Notification Message
FPL	Filed Flight PLan
MFS	Message From Shanwick
RQP	ReQuest flight Plan
RQS	ReQuest Supplementary flight plan

Table 4.12. <<enumeration>> FlightPlanMessageType

(3) Used by: InvalidFiling.

4.92. FlightPlanOrInvalidFiling

(1) For a given flight plan, container for the last valid flight keys or the current invalid filing summaries, not both.

(2) Choices:

a) **FlightIdentificationOutput lastValidFlightPlanId**
The last valid flight id.

b) **InvalidFiling currentInvalid**
The current invalid filing summaries.

(3) Used by: FlightOrFlightPlan, FlightPlanListReply.

4.93. FlightPlanUnion

(1) Flight plan information expressed in either a structural or a textual format.

(2) Choices:

a) **FlightPlan structuralFPL**
Object used when the flight plan data is input in a structured manner.

b) **ExtendedFlightPlan structuralEFPL**
Object used when the flight plan data is input in a extended structured manner.

c) **string textual**
FPL message text used when the flight plan data is input via a string.
Two message formats are accepted: ICAO and ADEXP.

(3) Used by: ExtendedFlightPlanValidationRequest.

4.94. FlightPlanUpdate

(1) This class is used to express updates on an existing flight plan.

(2) All attributes in this class are optional.

(3) The flight plan resulting from the application of the update to the existing flight plan must comply with the constraints expressed in FlightPlan.

(4) It is not possible to update separately one attribute of the `otherInformation` and `supplementaryInformation` attributes. This is a consequence of the new ICAO rules for the update of the field18 and field19. These new rules impose to provide the complete field in a change message where part of a field needs to be changed.

(5) Attributes:

a) **AlternateAerodrome alternate1Aerodrome** *(Optional)*

b) **AlternateAerodrome alternate2Aerodrome** *(Optional)*

c) **AlternateAerodrome_DataType enrouteAlternateAerodromes** *(Optional)*

d) **AlternateAerodrome_DataType takeoffAlternateAerodromes** *(Optional)*

e) **AircraftIdentificationUpdate aircraftId** *(Optional)*

f) **int numberOfAircraft** *(Optional)*
Constraint: Range: `[1,99]`.

g) **AircraftType aircraftType** *(Optional)*

h) **DurationHourMinute totalEstimatedElapsedTime** *(Optional)*

i) **EstimatedElapsedTimeAtLocation[] eetsToLocations** *(Optional)*
Constraint: Size must be comprised between 0 and ∞.

j) **WakeTurbulenceCategory wakeTurbulenceCategory** *(Optional)*

k) **FlightType flightType** *(Optional)*

l) **FlightRules flightRules** *(Optional)*

m) **DateTimeMinute estimatedOffBlockTime** *(Optional)*

n) **string icaoRoute** *(Optional)*

o) **StayInformation_DataType[] stayInformation** *(Optional)*
Constraint: Size must be comprised between 0 and 9.

p) **EquipmentCapabilityAndStatus equipmentCapabilityAndStatus** *(Optional)*

q) **SurveillanceEquipment surveillanceEquipment** *(Optional)*

r) **OtherInformation otherInformation** *(Optional)*

s) **SupplementaryInformation supplementaryInformation** *(Optional)*

(6) Used by: ExtendedFlightPlanUpdate, FlightPlanUpdateRequest.

4.95. <<enumeration>> FlightPlanValidationRequestedField

(1) Enumerates the additional fields that the caller may request to be returned in reply of FlightPlan-ValidationRequest.

(2) Values:

a) **fourDimensionalTrajectory**
Four Dimensional Trajectory

b) **profileTuningRestrictions**
Profile tuning restrictions

(3) Used by: ExtendedFlightPlanValidationRequest.

4.96. `FlightPoint`

(1) Describes a point in a flight point profile.

(2) Attributes:

a) **`DateTimeSecond` `timeOver`** *(Mandatory)*
Estimated time over the point.
If the point is a first point: take-off time.
If the point is a last point: landing time.

b) **`FlightLevel` `flightLevel`** *(Mandatory)*
Flight level over the point.

c) **`FlightTrend` `entryTrend`** *(Mandatory)*
Trend before overflying the point.

d) **`FlightTrend` `exitTrend`** *(Mandatory)*
Trend after overflying the point.

e) **`FlightPointRoute` `associatedRoute`** *(Optional)*
Route followed after overflying the point, unless the point is the last one in which case the route followed before the point (e.g. Standard Arrival Procedure).
Optional: null in case of DCT.

f) **`DistanceNM` `coveredDistance`** *(Mandatory)*
Distance from the first point in the profile measured on the 2D track of the point profile.

g) **`boolean` `isVisible`** *(Mandatory)*
True if the route segment following the point is "visible", i.e. GAT/IFR.

h) **`AerodromeICAOId` `aerodrome`** *(Optional)*
ICAO id of an aerodrome:

i) The aerodrome of departure if the point is the first in the profile

ii) The aerodrome of arrival if the point is the last in the profile

Optional, unless the point is first or last. Must be null if `aerodrome` is not null.

i) **`ICAOPoint` `point`** *(Optional)*
Point in the en-route point profile.
Note that for vector points (e.g. bottom of climb, top of descent), the coveredDistance is to be used to determine where the point is while `point` itself is set to null..
Must be null if `aerodrome` is not null. If null and `aerodrome` is also null, then it concerns a vector point.

j) **`boolean` `flightPlanPoint`** *(Optional)*
If `point` is not null, this attribute is set to true unless the point is not a flight plan point but was added by NM in order to provide a better approximation on a long DCT segment.

Cannot be null if `point` is not null; must be null otherwise.

(3) Used by: Flight.

4.97. `FlightPointRoute`

(1) Describes a route id associated to a point in a flight point profile.

(2) Attributes:

a) **`RouteId` `routeId`** *(Optional)*
Id of the route if the route segment is not a departure or an arrival procedure.
Cannot be null if `departureId` or `arrivalId` are not null; must be null otherwise.

b) **`ProcedureICAOId` `departureProcedureId`** *(Optional)*
Id of the procedure if the route segment is a departure procedure.
Cannot be null if `routeId` or `arrivalId` are not null; must be null otherwise.

c) **`AerodromeICAOId` `aerodromeOfDeparture`** *(Optional)*
ICAO id of the aerodrome of departure in case of departure procedure.
Cannot be null if `departureProcedureId` is not null; must be null otherwise.

d) **`ProcedureICAOId` `arrivalProcedureId`** *(Optional)*
Id of the procedure if the route segment is an arrival procedure.
Cannot be null if `routeId` or `departureId` are not null; must be null otherwise.

e) **`AerodromeICAOId` `aerodromeOfArrival`** *(Optional)*
ICAO id of the aerodrome of arrival in case of arrival procedure.
Cannot be null if `arrivalProcedureId` is not null; must be null otherwise.

(3) Used by: FlightPoint.

4.98. `FlightRestriction`

(1) Describes a Restriction event (entry or exit) over a geographical position.

(2) Attributes:

a) **`DateTimeSecond` `timeOver`** *(Mandatory)*
Estimated time over the point.

b) **`DistanceNM` `coveredDistance`** *(Mandatory)*
Distance from the first point in the profile measured on the 2D track of the point profile.

c) **`FlightPlanProcessing` `flightPlanProcessing`** *(Mandatory)*
Indicates the kind of restriction with regards to flight plan processing.

d) **`RestrictionId` `restrictionId`** *(Mandatory)*
Identification of the Restriction.

e) **EntryExit event** *(Mandatory)*
Indicates if the restriction is entered or exited at this position.

f) **Position position** *(Mandatory)*
Indicates the geographical position where the event occurs.

g) **FlightLevel flightLevel** *(Mandatory)*
Indicates the flight level where the event occurs.

(3) Used by: Flight.

4.99. <<enumeration>> FlightRules

(1) Applicable flight rules.

(2) Values:

Value	Description	ICAO value
IFR	Instrumental Flight Rules	I
IFR_THEN_VFR	First IFR, then VFR	Y
VFR	Visual Flight Rules	V
VFR_THEN_IFR	First VFR, then IFR	Z

Table 4.13. <<enumeration>> FlightRules

(3) Used by: FlightPlan, FlightPlanUpdate.

4.100. <<enumeration>> FlightState

(1) Describes the state of a flight resulting from an event in the flight history.

(2) Values:

a) **ATC_ACTIVATED**

b) **CANCELLED**

c) **FILED**

d) **FILED_SLOT_ALLOCATED**

e) **FILED_SLOT_ISSUED**

f) **PLANNED**

g) **PLANNED_REROUTED**

h) **PLANNED_SLOT_ALLOCATED**

i) **PLANNED_SLOT_ALLOCATED_REROUTED**

j) **TACT_ACTIVATED**

k) **TERMINATED**

(3) Used by: FlightEvent.

4.101. <<enumeration>> FlightTrend

(1) Describes a flight trend at some point, i.e. the flight vector that includes the point (as an end point in "trend in" and as start point in "trend out") is either a cruise vector, a climb vector or a descent vector.

(2) The NONE value means not applicable in the situation.

(3) Values:

a) **CLIMB**

b) **CRUISE**

c) **DESCENT**

d) **NONE**

(4) Used by: Flight, FlightAirspace, FlightPoint.

4.102. <<enumeration>> FlightType

(1) Type of the flight.

(2) Values:

Value	Description	ICAO value
GENERAL	general flight	G
MILITARY	military flight	M
NOT_SCHEDULED	not scheduled flight	N
OTHER	any flight type that is not of a type described above	X
SCHEDULED	scheduled flight	S

Table 4.14. <<enumeration>> FlightType

(3) Used by: FlightPlan, FlightPlanUpdate.

4.103. FourDimensionalTrajectory

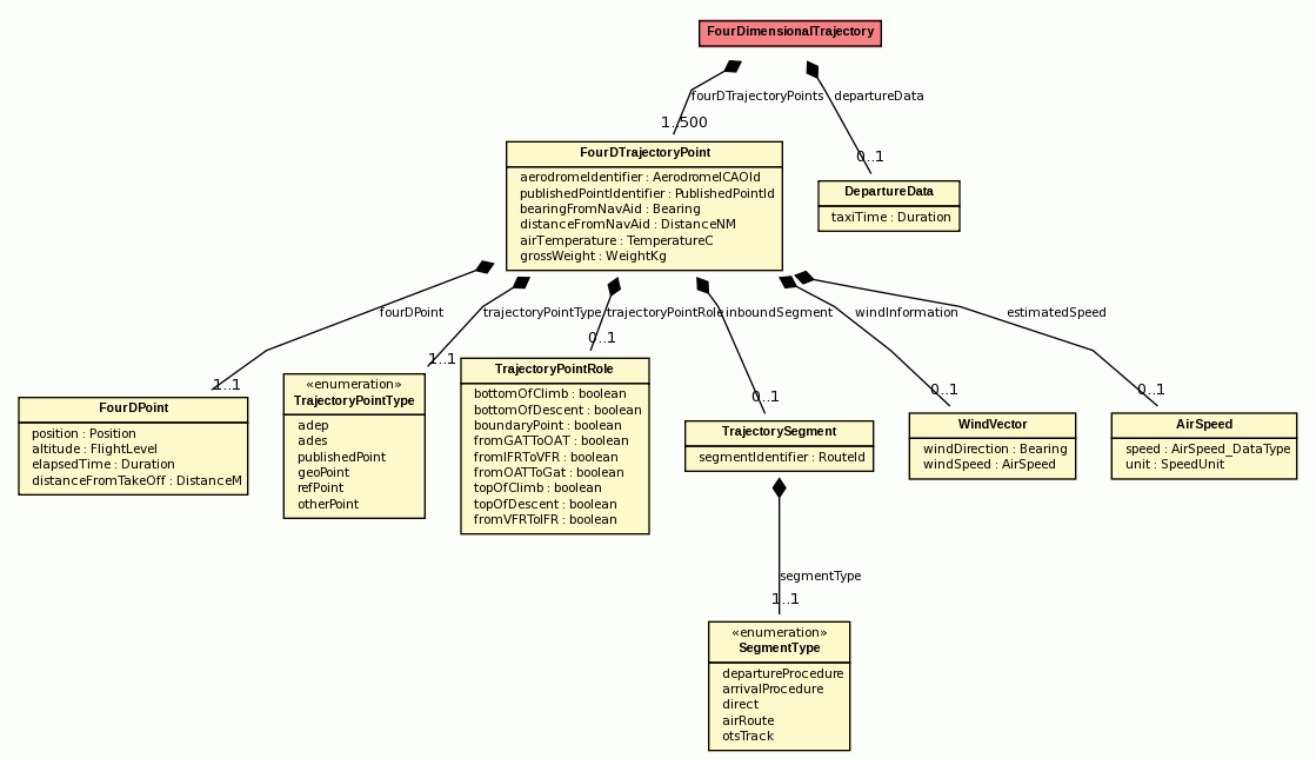

Figure 4.13. `FourDimensionalTrajectory` Class Diagram

(1) AO calculated flight trajectory taking into account constraints and meteorological information for its calculation.

(2) Attributes:

a) **`FourDTrajectoryPoint[] fourDTrajectoryPoints`** *(Mandatory)*
The list of 4D trajectory points.
Constraints:

i) Size must be comprised between 1 and 499.

ii) See AT_LEAST_ONE_ADEP_ONE_POINT_ONE_ADES

iii) See CUMULATIVE_DISTANCE_MUST_INCREASE_BY_MINIMUM_ONE_METER

iv) See ELAPSED_TIME_MUST_INCREASE_BY_MINIMUM_ONE_SECOND

v) See FIRST_POINT_MUST_HAVE_A_CUMULATIVE_DISTANCE

vi) See FIRST_POINT_MUST_HAVE_A_ZERO_TIME

vii) See ONE_AND_EXACTLY_ONE_ADEP

viii) See ONE_AND_EXACTLY_ONE_ADES

ix) See REFPOINT_OTHERPOINT_NOT_SUPPORTED

b) **DepartureData departureData** *(Optional)*

(3) Constraints:

a)

Name	AT_LEAST_ONE_ADEP_ONE_POINT_ONE_ADES
Attribute	fourDTrajectoryPoints
Description	The fourDTrajectoryPoints attribute should at least contain the aerodrome of departure, one point and the aerodrome of arrival.

b)

Name	ONE_AND_EXACTLY_ONE_ADEP
Attribute	fourDTrajectoryPoints
Description	The fourDTrajectoryPoints attribute cannot contain more than one aerodrome of departure.

c)

Name	ONE_AND_EXACTLY_ONE_ADES
Attribute	fourDTrajectoryPoints
Description	The fourDTrajectoryPoints attribute cannot contain more than one aerodrome of arrival.

d)

Name	FIRST_POINT_MUST_HAVE_A_ZERO_TIME
Attribute	fourDTrajectoryPoints
Description	The first FourDTrajectoryPoint of the fourDTrajectoryPoints list has a zero time. (Typically the Aerodrome of departure)

e)

Name	FIRST_POINT_MUST_HAVE_A_CUMULATIVE_DISTANCE
Attribute	fourDTrajectoryPoints
Description	The first FourDTrajectoryPoint of the fourDTrajectoryPoints list has a zero cumulative distance. (Typically the Aerodrome of departure)

f)

Name	CUMULATIVE_DISTANCE_MUST_INCREASE_BY_MINIMUM_ONE_METER
Attribute	fourDTrajectoryPoints
Description	FourDTrajectoryPoint must increase within the fourDTrajectoryPoints list by the minimum resolution of one meter.

g)

Name	ELAPSED_TIME_MUST_INCREASE_BY_MINIMUM_ONE_SECOND
Attribute	fourDTrajectoryPoints
Description	FourDTrajectoryPoint must increase within the fourDTrajectoryPoints list by the minimum resolution of one second.

h)

Name	REFPOINT_OTHERPOINT_NOT_SUPPORTED
Attribute	fourDTrajectoryPoints
Description	The values refPoint and otherPoint are not supported for trajectoryPointType.

(4) Used by: ExtendedFlightPlanValidationReply, ExtendedFlightPlan, ExtendedFlightPlanUpdate.

4.104. FourDPoint

(1) A representation of a 4 dimensional point.

(2) Attributes:

a) **Position position** *(Mandatory)*
The geographical position of the point.

b) **FlightLevel altitude** *(Mandatory)*
Elevation of the point.

c) **Duration elapsedTime** *(Mandatory)*
Time elapsed relative to the take-off time.

d) **DistanceM distanceFromTakeOff** *(Mandatory)*
Cumulative distance over the ground as a vertical projection on the ground from take-off up to the 4DPoint.

(3) Used by: FourDTrajectoryPoint.

4.105. FourDTrajectoryPoint

(1) Constraints: From the class FourDTrajectoryPoint we will not use the attributes:

(2) a) airTemperature

b) estimatedSpeed

c) windInformation

(3) If these attributes are provided in the B2B request, they will be ignored.

(4) Attributes:

a) **FourDPoint** **fourDPoint** *(Mandatory)*

b) **TrajectoryPointType** **trajectoryPointType** *(Mandatory)*
Indicate the type of point (e.g.: ADEP, geoPoint).
In case of refPoint, the Position inherited from FourDPoint is the geographical position of the trajectory point resulting from the calculation based on a NavAid, distance and bearing.
Constraints:

i) See AERODROME_IDENTIFIER_MANDATORY_IF_ADEP

ii) See AERODROME_IDENTIFIER_MANDATORY_IF_ADES

iii) See AERODROME_IDENTIFIER_NULL_IF_NOT_ADEP_OR_ADES

iv) See PUBLISHED_POINT_IDENTIFIER_MANDATORY_IF_PUBLISHED_POINT

v) See PUBLISHED_POINT_IDENTIFIER_MANDATORY_IF_REF_POINT

vi) See PUBLISHED_POINT_IDENTIFIER_NULL_IF_NOT_PUBLISHED_POINT_AND_NOT_REF_POINT

vii) See TRAJECTORY_POINT_ROLE_CONSTRAINT_IF_OTHER_POINT

c) **TrajectoryPointRole** **trajectoryPointRole** *(Optional)*
Indicate the role of the point in the trajectory, e.g.: bottomOfClimb, VFRToIFR.
A point can have multiple roles (e.g.: a publishedPoint can be the bottom of a climb and the point where the rules change from VFR To IFR).
Constraint: See TRAJECTORY_POINT_ROLE_CONSTRAINT_IF_OTHER_POINT

d) **AerodromeICAOId** **aerodromeIdentifier** *(Optional)*
Published coded designator of the trajectory point.
Constraints:

i) See AERODROME_IDENTIFIER_MANDATORY_IF_ADEP

ii) See AERODROME_IDENTIFIER_MANDATORY_IF_ADES

iii) See AERODROME_IDENTIFIER_NULL_IF_NOT_ADEP_OR_ADES

e) **PublishedPointId** **publishedPointIdentifier** *(Optional)*
Published coded designator of the trajectory point.
Constraints:

i) See PUBLISHED_POINT_IDENTIFIER_MANDATORY_IF_PUBLISHED_POINT

ii) See PUBLISHED_POINT_IDENTIFIER_MANDATORY_IF_REF_POINT

iii) See PUBLISHED_POINT_IDENTIFIER_NULL_IF_NOT_PUBLISHED_POINT_AND_NOT_REF_POINT

f) **Bearing bearingFromNavAid** *(Optional)*
Compulsory when trajectoryPointType is refPoint, is null in the other cases.
It is the bearing from a navaid used to define a reference point (Cf.: ICAO doc 4444).

g) **DistanceNM distanceFromNavAid** *(Optional)*
Compulsory when trajectoryPointType is refPoint, is null in the other cases.
It is the distance from a navaid used to define a reference point (Cf.: ICAO doc 4444).

h) **TrajectorySegment inboundSegment** *(Optional)*
The route segment that ends or continues at the 4DTrajectoryPoint. Is null for the first trajectoryPoint, is compulsory for all other 4DTrajectoryPoint.
For a point having the role:

i) `bottomOfClimb`

ii) `bottomOfDescent`

iii) `topOfClimb`

iv) `topOfDescent`

that needs to be inserted inside an Airway, the `segmentType` should be airRoute.
For example: WYPTA UL123 TOC UL123 WYPTB.
The same reasoning applies for points that need to be inserted inside `DCT`, `SID`, `STAR` or `OTS tracks`.
Examples:
for DCT: DCT JFK DCT HTO DCT *TOC* DCT ACK DCT WHALE.
for SID: DVR6J *TOC* DVR6J DVR UL9.
for STAR: REMBA REMBA1K *TOD* REMBA1K.
for OTS tracks: LESLU DCT LIMRI/M081F360 NATF 55N040W/M081F370 NATF *TOC* NATF STEAM/M081F400 DCT.

i) **TemperatureC airTemperature** *(Optional)*
The forecast static air temperature used to calculate the 4D Trajectory at the location and the corresponding estimated level included in the 4D Trajectory. It is only required when Speed is given as TAS.

j) **WindVector windInformation** *(Optional)*
The forecast direction and speed of the wind used to calculate the 4D Trajectory at the location and the corresponding estimated level included in the 4D trajectory.

k) **WeightKg grossWeight** *(Optional)*
Total weight of the aircraft at a location included in the 4D Trajectory, starting with the aerodrome of departure (ADEP). The total weight at the ADEP is the Take-Off Weight (TOW).

l) **AirSpeed estimatedSpeed** *(Optional)*
Estimated speed of the aircraft at the location expressed as Mach number or True Air Speed (TAS).

(5) Constraints:

a)

Name	TRAJECTORY_POINT_ROLE_CONSTRAINT_IF_OTHER_POINT
Attributes	trajectoryPointType, trajectoryPointRole
Description	When trajectoryPointType is otherPoint the trajectoryPointRole cannot be GATToOAT, IFRToVFR, OATToGAT, VFRToIFR.

b)

Name	PUBLISHED_POINT_IDENTIFIER_MANDATORY_IF_PUBLISHED_POINT
Attributes	trajectoryPointType, publishedPointIdentifier
Description	publishedPointIdentifier is compulsory when trajectoryPointType is publishedPoint.

c)

Name	PUBLISHED_POINT_IDENTIFIER_MANDATORY_IF_REF_POINT
Attributes	trajectoryPointType, publishedPointIdentifier
Description	publishedPointIdentifier is compulsory when trajectoryPointType is refPoint.

d)

Name	PUBLISHED_POINT_IDENTIFIER_NULL_IF_NOT_PUB-LISHED_POINT_AND_NOT_REF_POINT
Attributes	trajectoryPointType, publishedPointIdentifier
Description	publishedPointIdentifier must be null for trajectoryPointType other than refPoint or publishedPoint.

e)

Name	AERODROME_IDENTIFIER_MANDATORY_IF_ADEP
Attributes	trajectoryPointType, aerodromeIdentifier
Description	aerodromeIdentifier is compulsory when trajectoryPointType is adep.

f)

Name	AERODROME_IDENTIFIER_MANDATORY_IF_ADES
Attributes	trajectoryPointType, aerodromeIdentifier
Description	aerodromeIdentifier is compulsory when trajectoryPointType is ades.

g)

Name	AERODROME_IDENTIFIER_NULL_IF_NOT_ADEP_OR_ADES
Attributes	trajectoryPointType, aerodromeIdentifier

Description	`aerodromeIdentifier` must be null for `trajectoryPointType` other than `adep` or `ades`.

(6) Used by: FourDimensionalTrajectory.

4.106. typedef<string> FreezePoint

(1) If associated to ADEP, the route generation will start from this point. The part of the route from the ADEP to this point is "frozen"

(2) If associated to ADES, the route generation will end with this point. The part of the route from that point to the ADES is "frozen"

(3) Pattern: `ANY{0,15}`

(4) Used by: RoutingAssistanceRequest.

4.107. <<enumeration>> FrequencyOnAircraft

(1) Enumerates frequencies that can be available on an aircraft. (ICAO R/).

(2) Values:

Value	Description
ELT	Emergency Locator Transmitter.
UHF	Ultra-High Frequency: 243.0 MHz.
VHF	Very High Frequency: 121.5 MHz.

Table 4.15. <<enumeration>> FrequencyOnAircraft

(3) Used by: SupplementaryInformation.

4.108. typedef<string> ICAOAircraftAddress

(1) 24-bytes ICAO aircraft address, made of 6 hexadecimal digits expressed as ALPHANUM values constrained in ["0",..., "9", "A",..., "F"].

(2) Pattern: `HEXA{6}`

(3) Used by: AircraftIdentificationUpdate, AircraftIdentification.

4.109. <<enumeration>> ICAOSTSIndicator

(1) Enumerates the ICAO STS indicators.

(2) Values:

Value	Description
ALTRV	Flight operated in accordance with an altitude reservation.
ATFMX	flight approved for exemption from ATFM measures by the appropriate ATS authority.
FFR	Fire-Fighting.
FLTCK	Flight check for calibration of navaids.
HAZMAT	Flight carrying hazardous material.
HEAD	flight with Head of State status.
HOSP	medical flight declared by medical authorities.
HUM	flight operating on a humanitarian mission.
MARSA	Flight for which a military entity assumes responsibility for separation of military aircraft.
MEDEVAC	Life critical medical emergency evacuation.
NONRVSM	non-RVSM capable flight intending to operate in RVSM airspace.
SAR	flight engaged in a search and rescue mission.
STATE	flight engaged in military, customs or police services.

Table 4.16. <<enumeration>> ICAOSTSIndicator

(3) Used by: SpecialHandlingIndicators.

4.110. <<enumeration>> IFPIndicator

(1) Indication of known errors within a flight plan. This is an indication that some automatic or manual actions have been taken by NM to correct or ignore an error.

(2) Values:

Value	Description
AIRCRAFT_TYPE	An error that cannot be corrected has been found in the aircraft type.
EOBDT	An earlier estimated off-block time exists.
ERREQPT	Flight compliance with equipment.
ERRRTECOORD	Flight compliance with route coordinates.
FLIGHT_LEVEL	An error that cannot be corrected has been found in the requested flight level.
MODE_S	Flight compliance with mode S surveillance.
NON_833	Flight does not comply with 8.33 KHz requirements.
NON_RVSM	State flight is non-RVSM approved.
ROUTE	An error that cannot be corrected has been found in the route.

Value	Description
ROUTE_RAD	The route does not conform with the RAD rules.
ROUTE_WE	The route does not conform with the weekend routings.
RVSM_VIOLATION	Flight violates RVSM rules, flight plan originator cannot be contacted.
UNKNOWN_833	Flight compliance with 8.33 KHz requirements is unknown.
UNKNOWN_RVSM	RVSM approval status of the flight is unknown.

Table 4.17. <<enumeration>> IFPIndicator

(3) Used by: FilingResultValid.

4.111. typedef<string> IFPLId

(1) Examples:AA00953172, BB00956485,...

(2) Pattern: `UALPHA{2}DIGIT{8}`

(3) Used by: FlightDepartureRequest, FlightOperationalLogEntry, AirportPlanningInformationRequest, FlightIdentificationOutput, FlightPlan, FlightPlanCancellationRequest, ExtendedFlightPlanUpdate-Request, FlightArrivalRequest, FlightIdentificationInput, FlightDelayRequest, FilingResultValid, FlightPlanUpdateRequest.

4.112. IFPSError

(1) Represents an NM/IFPS error.

(2) An IFPS error is made of the IFPS error class (e.g.: "EFPM, PROF, ...) concatenated with the error identification number (e.g.: 052), e.g. "EFPM052". This is the error type id passed in the Error instances via the `code` attribute. The error description is the same as the one passed in the REJ message that NM sends to the FPL originator in case on invalid flight plan.

(3) Attributes:

a) **`string code`** *(Mandatory)*
An IFPS error code is made of the IFPS error class (e.g.: "EFPM", "PROF", ...) concatenated with the error identification number (e.g.: "052"), e.g. "EFPM052".

b) **`string description`** *(Optional)*
The error description is the same as the one passed in the REJ message that NM sends to the FPL originator in case on invalid flight plan.

c) **`string anomalyComment`** *(Optional)*

(4) Used by: FlightPlanValidationReply, ExtendedFlightPlanValidationReply, RoutingAssistanceReply, RouteInfo, FilingResultRejected.

4.113. InvalidFiling

(1) Summary of an invalid flight plan message.

(2) Attributes:

a) **DateTimeSecond** **filingTime** *(Mandatory)*

b) **FlightPlanMessageType** **invalidMessageType** *(Mandatory)*

c) **FlightPlanMessageStatus** **invalidMessageStatus** *(Mandatory)*

d) **FlightKeys** **keys** *(Optional)*

(3) Used by: FlightPlanOrInvalidFiling.

4.114. typedef<double> Latitude

(1) Represents latitude as an angle in degrees having the range [-90 .. 90].

(2) Positive values represent Northern latitudes where negative values represent Southern latitudes taking the place of the traditional "N" and "S" designators.

(3) Accuracy of the latitude coordinates: 0.1 seconds (~3 meters).

(4) Examples:50.78639940000001

(5) Range:] - ∞,∞[.

(6) Used by: Position.

4.115. <<enumeration>> LifeJacketEquipment

(1) Enumerates the possible equipment items that life jackets carried by an aircraft can have.

(2) ICAO J/ field.

(3) Values:

Value	Description
FLUORESCEIN	The life jackets are equipped with fluorescein.
LIGHTS	The life jackets are equipped with lights.
UHF	All life jackets are equipped with UHF on frequency 243.0 MHz.
VHF	All life jackets are equipped with VHF on frequency 121.5 MHz.

Table 4.18. <<enumeration>> LifeJacketEquipment

(4) Used by: SupplementaryInformation.

4.116. LoadStateAtReferenceLocation

(1) The ENTRY or OCCUPANCY load state at reference location.

(2) Choices:

a) **LoadState ENTRY**
Indicates the monitored entry load state in which the flight is involved.

b) **OtmvStatus OCCUPANCY**
Indicates the monitored occupancy OTMV load state in which the flight is involved.

(3) Used by: Flight.

4.117. typedef<double> Longitude

(1) Represents a longitude as an angle in degrees having the range [-180 .. 180].

(2) Positive values represent Eastern longitudes where negative values represent Western longitudes taking the place of the traditional "E" and "W" designators.

(3) Accuracy of the longitude coordinates: 0.1 seconds (~3 meters).

(4) Examples:4.247596199999975

(5) Range:]- ∞,∞[.

(6) Used by: Position.

4.118. MessageOriginator

(1) Originator of a flight message.

(2) Choices:

a) **AirNavigationUnitId airNavigationUnitId**
Id of the originating air navigation unit.

b) **NetworkAddress address**
Network address of the originating air navigation unit.

(3) Used by: Flight.

4.119. ModeSCapabilities

(1) Used in the surveillance equipment to describe the Mode S capabilities of the aircraft if equipped with Mode S transponder.

(2) Attributes:

a) **EquipmentStatus aircraftIdentification** *(Optional)*
Indicates if the Mode S transponder is transmitting the aircraft identification.

b) **EquipmentStatus pressureAltitude** *(Optional)*
Indicates if the Mode S transponder is transmitting the pressure and altitude.

c) **EquipmentStatus extendedSquitterADSB** *(Optional)*
Indicates if the Mode S transponder is equipped with an extended squitter (ADS-B).

d) **EquipmentStatus enhancedSurveillance** *(Optional)*
Indicates if the Mode S transponder is equipped with extended surveillance.
The accepted combinations and the correspondence with the ICAO Mode S equipment code I, P, X, E, H, L, S is defined by the following table:

ICAO code	aircraft Identification	pressure Altitude	extended SquitterADSB	enhanced Surveillance
I	equipped	not equipped	not equipped	not equipped
P	not equipped	equipped	not equipped	not equipped
X	not equipped	not equipped	not equipped	not equipped
E	equipped	equipped	equipped	not equipped
H	equipped	equipped	not equipped	equipped
L	equipped	equipped	equipped	equipped
S	equipped	equipped	not equipped	not equipped

Table 4.19. ModeSCapabilities.enhancedSurveillance description

Note: The ICAO description excludes the combination of I, P, X codes. It also excludes the combination of one or more of E, H, L, S with one of the I, P, X. But it does not exclude the combination of E, H, L, S codes, this means that when converting the ICAO field 10b into the 4 `ModeSCapabilities`, the "not equipped" status induced by one of E, H, L, S code shall be overwritten by the "equipped" status induced by an other E, H, L, S code.

(3) Used by: SurveillanceEquipment.

4.120. typedef<int> NumberOfDinghies_DataType

(1) The number of dinghies carried by the aircraft.

(2) Range: `[0,99]`.

(3) Used by: Dinghies.

4.121. OtherAerodromeDesignation

(1) Used to specify the name and location of an aerodrome for which no ICAO identification exist or the first or last point of the route when departing from or arriving to a place that is not an aerodrome.

(2) Attributes:

a) **AerodromeName_DataType aerodromeName** *(Optional)*
The name of the aerodrome.
Cannot be null if `firstLastRoutePoint` is null. Must be null if otherwise.

b) **NonPublishedPoint aerodromeLocation** *(Optional)*
The location of the aerodrome expressed as a reference point or a geographical position.
Must be null if `firstLastRoutePoint` is not null.

c) **ICAOPoint firstLastRoutePoint** *(Optional)*
The first or last point of the route: given only when the aircraft departs from or lands to a place that is not an aerodrome.
Cannot be null if `aerodromeName` is null. Must be null otherwise.

(3) Used by: Aerodrome.

4.122. typedef<string> OtherAircraftTypeDesignation_Data-Type

(1) Name of the aircraft type if no ICAO id exists for this aircraft type (ICAO TYP/ field).

(2) Pattern: `ANY{1,60}`

(3) Used by: AircraftType.

4.123. OtherInformation

(1) Any other flight data items specified in the bilateral agreement.

(2) See also ICAO 4444 document field 18 (other information).

(3) All attributes in this class are optional for all services using the FlightPlan structure.

(4) Attributes:

a) **string selCalCode** *(Optional)*
Selcal (Selective Calling) code made up of a four letter code. Included if so prescribed by the appropriate ATS authority.
ICAO field 18 SEL/. ADEXP -SEL.
Constraint: Pattern: `ALPHA{4,5}`

b) **AircraftOperatorName_DataType nameOfOperator** *(Optional)*
Name of the operator, if not obvious from the aircraft identification.
ICAO field 18 OPR/. ADEXP -OPR.

c) **SpecialHandlingIndicators reasonForSpecialHandling** *(Optional)*
Reasons for special handling by ATS.

d) **AircraftPerformanceCategory aircraftPerformanceData** *(Optional)*
Aircraft performance data, indicated by a single letter as specified in the ICAO Doc 8168, if so prescribed by the appropriate ATS authority.
ICAO field 18 PER/. ADEXP -PER.

e) **string communicationEquipment** *(Optional)*
Significant data related to communication equipment as required by the appropriate ATS authority, e.g. COM/UHF only.
ICAO field 18 COM/. ADEXP -COM.
Constraint: Pattern: `ANY{1,50}`

f) **DatalinkCapabilities datalinkCapabilities** *(Optional)*
Up to four different datalink capabilities.
ICAO field 18 DAT/. ADEXP -DAT.

g) **string navigationEquipment** *(Optional)*
Significant navigation equipment.
ICAO field 18 NAV/. ADEXP -NAV.
Constraint: Pattern: `ANY{1,50}`

h) **PerformanceBasedNavigationCode[] performanceBasedNavigationCodes** *(Optional)*
Indication of RNAV and/or RNP capabilities.
No duplicates are accepted.
ICAO field 18 PBN/. ADEXP -PBN.
Constraint: Size must be comprised between 0 and 8.

i) **string otherSurveillanceEquipments** *(Optional)*
Includes surveillance applications or capabilities not specified in the `surveillanceEquipment` of the flight plan (ICAO 10b).
ICAO field 18 SUR/. ADEXP -SUR.
Constraint: Pattern: `ANY{1,50}`

j) **int replacementFlightPlanIndicator** *(Contextual)*
External flight plan version possibly provided by the submitter when a flight plan is cancelled and re-submitted. This is opaque for NM.
ICAO field 18 RFP/. ADEXP -RFP
Presence:

i) Ignored in `ExtendedFlightPlanValidationRequest`, `FlightPlanValidationRequest`, `RoutingAssistanceRequest`

ii) Optional otherwise.

Constraint: Range: `[1,9]`.

k) **DistanceM runwayVisualRange** *(Contextual)*
Runway Visual Range (RVR). Operating minima when special meteorological conditions exist.
If specified, must be within [0, 999].

ICAO field 18 RVR/. ADEXP -RVR
Presence:

i) Ignored in ExtendedFlightPlanValidationRequest, FlightPlanValidationRequest, RoutingAssistanceRequest

ii) Optional otherwise.

l) **ReclearanceInFlight reclearanceInFlight** *(Contextual)*
Revised route subject to clearance in flight and terminating with the ICAO designator of the revised aerodrome of destination.
ICAO field 18 RIF/. ADEXP -RIF
Presence:

i) Ignored in ExtendedFlightPlanValidationRequest, FlightPlanValidationRequest, RoutingAssistanceRequest

ii) Optional otherwise.

m) **string[] otherRemarks** *(Optional)*
Any other plain language remarks when required by the appropriate ATS authority or deemed necessary by the pilot-in-command for the provision of air traffic services.
ICAO field 18 RMK/. ADEXP -RMK.
Constraint: Size must be comprised between 0 and ∞.

(5) Used by: FlightPlan, FlightPlanUpdate.

4.124. <<enumeration>> PerformanceBasedNavigationCode

(1) PBN code indicating the RNAV or RNP capability of the aircraft.

(2) Values:

Value	ICAO PBN Code
BASIC_RNP_1_ALL	O1
BASIC_RNP_1_DME_DME	O3
BASIC_RNP_1_DME_DME_IRU	O4
BASIC_RNP_1_GNSS	O2
RNAV_10	A1
RNAV_1_ALL	D1
RNAV_1_DME_DME	D3
RNAV_1_DME_DME_IRU	D4
RNAV_1_GNSS	D2
RNAV_2_ALL	C1
RNAV_2_DME_DME	C3

Value	ICAO PBN Code
RNAV_2_DME_DME_IRU	C4
RNAV_2_GNSS	C2
RNAV_5_ALL	B1
RNAV_5_DME_DME	B3
RNAV_5_GNSS	B2
RNAV_5_INS_OR_IRS	B5
RNAV_5_LORAN_C	B6
RNAV_5_VOR_DME	B4
RNP_4	L1
RNP_APCH	S1
RNP_APCH_BARO_VNAV	S2
RNP_AR_APCH_NO_RF	T2
RNP_AR_APCH_RF	T1

Table 4.20. <<enumeration>> PerformanceBasedNavigationCode

(3) Used by: OtherInformation.

4.125. PointDAL

(1) Attributes:

a) **ICAOPoint point** *(Mandatory)*

b) **DistanceM cumulativeDistance** *(Mandatory)*

(2) Used by: DistanceAtLocation.

4.126. Position

(1) Represents a position, i.e. a latitude and a longitude.

(2) Attributes:

a) **Latitude latitude** *(Mandatory)*

b) **Longitude longitude** *(Mandatory)*

(3) Used by: FourDPoint.

4.127. `ProfileTuningRestriction`

(1) Profile tuning restriction impacting the flight plan.

(2) Attributes:

a) **`string identifier`** *(Mandatory)*
Identifier of the profile restriction.

b) **`DateTimeSecond`** **`entryTime`** *(Mandatory)*
Entry time.

c) **`DateTimeSecond`** **`exitTime`** *(Mandatory)*
Exit time.

(3) Used by: ExtendedFlightPlanValidationReply.

4.128. `ProfileValidity`

(1) Contains data relating to the validity of the FTFM with respect to Flight Plan violations.

(2) Attributes:

a) **`ProfileValidityKind`** **`profileValidityKind`** *(Mandatory)*
Specifies if the FTFM profile validity:

i) has been evaluated and Flight Plan violations have been encountered before the maximum time limit has been reached.

ii) has been evaluated to the maximum time limit and no Flight Plan violations have been encountered.

iii) has not beeb evaluated.

b) **`DateTimeMinute`** **`lastValidEOBT`** *(Optional)*
The `last valid EOBT` represents the maximum EOBT to which the profile can be shifted into the future before any violation error(s) are encountered.

(3) Used by: Flight.

4.129. `<<enumeration>> ProfileValidityKind`

(1) According to the profile validity type value, the interpretation of the `lastValidEOBT` is different:

a) The FTFM profile validity has been evaluated and Flight Plan violations have been encountered before the maximum time limit has been reached. The optional field `lastValidEOBT` will be present and will represent the maximum EOBT to which the profile can be shifted into the future before Flight Plan violations have been encountered.

b) The FTFM profile validity has been evaluated to the maximum time limit and no Flight Plan violations have been encountered. The optional field `lastValidEOBT` will be present and will be set to the EOBT shifted by the maximum limit.

c) The FTFM profile validity has not been evaluated. The optional field `lastValidEOBT` will not be present.

(2) Values:

a) **`NO_VIOLATIONS`**
The FTFM profile validity has been evaluated to the maximum time limit and no Flight Plan violations have been encountered.

b) **`UNKNOWN`**
The FTFM profile validity has not been evaluated.

c) **`VIOLATIONS`**
The FTFM profile validity has been evaluated and Flight Plan violations have been encountered before the maximum time limit has been reached.

(3) Used by: ProfileValidity.

4.130. `ReadyStatus`

(1) Describes a flight readiness status.

(2) Attributes:

a) **`boolean readyForImprovement`** *(Optional)*
True if the flight is in Request For direct Improvement mode (RFI) state. False if it is not, i.e. when the SIP Wanted Message mode is on (SWM).
Null if `readyToDepart` is true; cannot be null otherwise.

b) **`boolean readyToDepart`** *(Mandatory)*
The flight is ready to depart (REA message received).

c) **`DurationHourMinute revisedTaxiTime`** *(Optional)*
Revised taxi time, if any: i.e. the minline-up of the last READY message.
Note that this might be different than the taxi time of the FTFM and RTFM and CTFM flight profiles.

(3) Used by: Flight.

4.131. `<<enumeration>> ReasonForDPICancellation`

(1) Improves the understanding of all operational users of the A-CDM events at the airports and helps the AOs and Handling Agents to take the best action for the flight concerned.

(2) Values:

a) **FLIGHT_PLAN_INVALID**
Flight plan invalid.
The discrepancy between TOBT and EOBT is larger then 15min (and needs to be resolved before startup will be issued).

b) **NO_AIRPORT_SLOT**
No Airport slot.
The airport does not have an airport slot for the departure.

c) **RETURN_TO_STAND**
Return to stand.
After having started taxiing, the flight returned back to stand/ramp.

d) **TOBT_UNKNOWN_OR_EXPIRED**
TOBT unknown or expired.
The TOBT was deleted, the pilot did not request startup or report ready in accordance with the procedures at the Airport.

e) **TSAT_EXPIRED**
TSAT expired.
The pilot did not request startup in accordance with the CDM procedures at the airport.

(3) Used by: CancelDPIRequest.

4.132. ReclearanceInFlight

(1) Describes a re-clearance in flight, i.e. the new route and destination aerodrome.

(2) Attributes:

a) **string icaoRoute** *(Mandatory)*
New route.

b) **AerodromeICAOId aerodrome** *(Mandatory)*
New aerodrome.

(3) Used by: OtherInformation.

4.133. Relative4DPoint

(1) Describes a four dimensional point relatively to the take-off.

(2) Attributes:

a) **DistanceM cumulativeDistance** *(Mandatory)*
The total ground distance in meters from the take-off up to the point.

b) **DistanceM altitude** *(Mandatory)*
Estimated level at the point expressed in meters above mean sea level (MSL).

c) **Duration elapsedTime** *(Mandatory)*
Estimated Time elapsed expressed in number of seconds, since take-off up to the point.

(3) Used by: FlightPerformanceData, BasicTrajectoryData.

4.134. RequestedFlightLevel

(1) Request flight level in a flight vertical profile.

(2) Attributes:

a) **FlightLevel flightLevel** *(Mandatory)*
Flight level.

b) **int segmentSequenceNumber** *(Mandatory)*
Identifies each segment of the profile.
Constraint: Range: `[0,9999]`.

c) **int relativeDistance** *(Mandatory)*
Relative distance (percentage) on the segment. If the requested flight level is on a segment point, the relative distance is zero.
Constraint: Range: `[0,100]`.

(3) Used by: Flight.

4.135. <<enumeration>> ReroutableStatus

(1) Describes whether a flight can be rerouted and under what conditions.

(2) Values:

a) **CANNOT_BE_REROUTED**

b) **TRY_ALLOWED**

c) **TRY_AND_APPLY_ALLOWED**

(3) Used by: Flight.

4.136. ReroutingIndicator

(1) This class describes both a rerouting reason and optionally a rerouting state.

(2) Attributes:

a) **boolean rerouted** *(Mandatory)*
True if the flight was rerouted.

b) **ReroutingReason reason** *(Optional)*
If the flight was rerouted, indicates why.

Cannot be null if `rerouted` is true; must be null otherwise.

c) **ReroutingState state** *(Optional)*
If the flight was rerouted, indicates its resulting rerouted state.
Cannot be null if `rerouted` is true; must be null otherwise.

(3) Used by: Flight.

4.137. <<enumeration>> ReroutingReason

(1) Describes whether a flight was rerouted, and if that is the case, why.

(2) Values:

Value	Description
AO	Aircraft Operator rerouting (AOWIR).
ATFM	ATFM rerouting.
AUTO	AUTOmatic rerouting (AWIR).
CEU	Central Executive Unit rerouting (CWIR).

Table 4.21. <<enumeration>> ReroutingReason

(3) Used by: ReroutingIndicator.

4.138. ReroutingReference

(1) Rerouting reference information.

(2) Attributes:

a) **string name** *(Optional)*
Name of the rerouting reference.
Constraint: Pattern: `ANY{1,14}`

b) **DateTimeMinute validTo** *(Optional)*
Time at which the RRP message was sent plus the maximum response time allowed.

(3) Used by: Flight.

4.139. <<enumeration>> ReroutingState

(1) If a flight was rerouted, describes the current rerouting state.

(2) Values:

Value	Description
EXECUTED	Rerouting has been executed.
NO_MATCH	Message received did not match the proposal; rerouting has been invalidated.
PRODUCED	There is a valid rerouting going on, waiting to be realised by either an FPL or a CHG.
REJECTED	Rerouting proposal has been rejected.
REVOKED	A booking was created for the proposal, but a manual operation that changed the regulation (typically deep rectify) has deleted the booking before the attempt to accept the proposal.
TIMED_OUT	No FPL/CHG received on time.

Table 4.22. <<enumeration>> ReroutingState

(3) Used by: ReroutingIndicator.

4.140. RevisionTimes

(1) Times to insert aircraft in sequence and remove aircraft from sequence.

(2) Attributes:

a) **DurationHourMinute timeToInsertInSequence** *(Optional)*
Time to insert in sequence.

b) **DurationHourMinute timeToRemoveFromSequence** *(Optional)*
Time to remove from sequence.

(3) Used by: Flight.

4.141. RouteInfo

(1) Description of an alternate route for a given flight plan.

(2) Attributes:

a) **DurationHourMinute duration** *(Mandatory)*
Duration of the alternate route.

b) **DistanceNM length** *(Mandatory)*
Length of the alternate route from ADEP to ADES.

c) **string icaoRoute** *(Mandatory)*
The complete field15 that can be used as an alternate to the route of the given flight plan.

d) **IFPSError[] routeErrors** *(Optional)*

Errors associated to a proposed alternate route. They can be of the following IFPS error classes: EFPM, PROF, ROUTE.
In some circumstances a route may be proposed with errors. If `radOff` is true, the proposed routes may contain errors.
The array is null if the proposed route has no error.
Constraint: Size must be comprised between 0 and ∞.

(3) Used by: RoutingAssistanceReplyDataResult.

4.142. RoutingAssistanceReplyDataResult

(1) Attributes:

a) **RouteInfo[] proposedRoutes** *(Mandatory)*
Contains the generated alternate routes.
If there is no NM/IFPS error but no alternate route was found, the array is not null but empty.
The array cannot contain null or duplicate items.
Constraint: Size must be comprised between 0 and ∞.

b) **DurationHourMinute originalRouteDuration** *(Mandatory)*
Cumulated duration of each route segments defining the two dimensional projections of the flight plan route from departure up to destination including flight portions being outside the IFPZ or being "invisible".

c) **DistanceNM originalRouteLength** *(Mandatory)*
Cumulated distance of each route segments defining the two dimensional projections of the flight plan route from departure up to destination including flight portions being outside the IFPZ or being "invisible".

(2) Used by: RoutingAssistanceReply.

4.143. <<enumeration>> SegmentType

(1) Route segment type. Can one on the following:

(2) Values:

Value	Description
airRoute	The segment belongs to a Air Route.
arrivalProcedure	The segment belongs to a STAR (Standard Terminal Arrival Route).
departureProced-ure	The segment belongs to a SID (Standard Instrument Departure).
direct	The segment belongs to a DIRECT Route.
otsTrack	The segment belongs to an OTS (Oceanic Track System) NAT (North Atlantic Region) Route.

Table 4.23. <<enumeration>> SegmentType

(3) Used by: TrajectorySegment.

4.144. SlotImprovementProposal

(1) Proposal for slot improvement, together with its associated response time.

(2) Attributes:

a) **DateTimeMinute proposedCalculatedTakeOffTime** *(Optional)*
Proposed take-off time.

b) **DateTimeMinute responseBy** *(Optional)*
Maximum time at which NM expects a response for this proposed slot improvement.

(3) Used by: Flight.

4.145. SpecialHandlingIndicators

(1) STS indicators, ICAO and exemptions (NM or other), used to indicate a reason for special handling of a flight.

(2) Attributes:

a) **ICAOSTSIndicator[] icaoSTSIndicators** *(Optional)*
List of ICAO STS indicators.
Optional. Cannot contain duplicates.
Constraint: Size must be comprised between 0 and ∞.

b) **EURSTSIndicator[] eurSTSIndicators** *(Optional)*
List of reasons for special handling used in the EUR region.
Optional. Cannot contain duplicates.
Constraint: Size must be comprised between 0 and ∞.

(3) Used by: OtherInformation.

4.146. typedef<string> SSRCode

(1) Examples:4567, 7683, 1352,...

(2) Pattern: `DIGIT{4}`

(3) Used by: Flight, SSRInfo.

4.147. SSRInfo

(1) SSR code assigned to an aircraft by the ATS and its transmission mode.

(2) Attributes:

a) **SSRCode code** *(Mandatory)*

SSR code transmitted via the mode described in `mode`.

b) **SSRMode mode** *(Mandatory)*
Transmission mode of the SSR code.

(3) Used by: AircraftIdentificationUpdate, AircraftIdentification.

4.148. <<enumeration>> SSRMode

(1) Transmission mode of the SSR code. Only SSRMode.A is supported.

(2) Values:

a) **A**

(3) Used by: SSRInfo.

4.149. typedef<string> StayInformation_DataType

(1) Examples:AIRWORK 10NM OF WUR BLOCK FL120 TO FL140, 0020 AIR REFUELING IN SPEEDY FL220B250 F16,...

(2) Pattern: `ANY{0,50}`

(3) Used by: FlightPlan, FlightPlanUpdate.

4.150. SupplementaryInformation

(1) See also ICAO 4444 document field 19 (Supplementary information).

(2) All attributes in this class are optional for all services using the FlightPlan structure.

(3) Attributes:

a) **DurationHourMinute fuelEndurance** *(Optional)*
Fuel endurance.

b) **int numberOfPersons** *(Optional)*
The total number of persons on board, when so prescribed by the appropriate ATS authority.
Constraint: Range: `[0,∞[`.

c) **FrequencyOnAircraft[] frequencyAvailability** *(Optional)*
Frequencies available on the aircraft.
The array does not accept null values or duplicates.
Constraint: Size must be comprised between 0 and ∞.

d) **SurvivalEquipment[] survivalEquipment** *(Optional)*
Survival equipment carried by the aircraft.
The array does not accept null values or duplicates.
Constraint: Size must be comprised between 0 and ∞.

e) **`string otherSurvivalEquipment`** *(Optional)*
Indicates any other survival equipment carried by the aircraft.
Constraint: Pattern: `ANY{1,50}`

f) **`LifeJacketEquipment[] lifeJacketEquipment`** *(Optional)*
Equipment of the life jackets carried by the aircraft.
The array does not accept null values or duplicates.
Constraint: Size must be comprised between 0 and ∞.

g) **`Dinghies dinghiesInformation`** *(Optional)*
Information on the dinghies carried by the aircraft.

h) **`string aircraftColourAndMarkings`** *(Optional)*
The colour and markings of the aircraft.
Constraint: Pattern: `ANY{1,50}`

i) **`string pilotInCommand`** *(Optional)*
The name of the pilot in command.
Constraint: Pattern: `ANY{1,50}`

(4) Used by: FlightPlan, FlightPlanUpdate.

4.151. SurveillanceEquipment

(1) Describes the serviceable surveillance equipment and capabilities of the aircraft.

(2) See also ICAO 4444 document field 10b.

(3) Attributes:

a) **`EquipmentStatus modeA`** *(Optional)*
Transponder Mode A (4 digits - 4096 codes).
ICAO code is "A".

b) **`EquipmentStatus modeAAndC`** *(Optional)*
Transponder Mode A (4 digits - 4096 codes) and Mode C.
ICAO code is "C".

c) **`EquipmentStatus modeS`** *(Optional)*
Indicates the presence of a Mode S transponder.
Constraints:

i) See `MODE_S_CAPABILITIES_MISSING_WHEN_MODE_S_EQUIPPED`

ii) See `MODE_S_CAPABILITIES_PRESENT_WHEN_MODE_S_NOT_EQUIPPED`

d) **`ModeSCapabilities modeSCapabilities`** *(Optional)*
Indicates the capabilities of the Mode S transponder.
Constraints:

i) See MODE_S_CAPABILITIES_MISSING_WHEN_MODE_S_EQUIPPED

ii) See MODE_S_CAPABILITIES_PRESENT_WHEN_MODE_S_NOT_EQUIPPED

e) **EquipmentStatus adsb1900Out** *(Optional)*
ADS-B with dedicated 1090 MHz ADS-B "out" capability.
ICAO code is "B1".

f) **EquipmentStatus adsb1900OutIn** *(Optional)*
ADS-B with dedicated 1090 MHz ADS-B "out" and "in" capability.
ICAO code is "B2".

g) **EquipmentStatus adsbOutUAT** *(Optional)*
ADS-B with "out" capability using Universal Access Transceiver (UAT).
CAO code is "U1".

h) **EquipmentStatus adsbOutInUAT** *(Optional)*
ADS-B with "out" capability using UAT.
ICAO code is "U2".

i) **EquipmentStatus adsbOutVDL4** *(Optional)*
ADS-B with "out" capability using VDL Mode 4.
ICAO code is "V1".

j) **EquipmentStatus adsbOutInVDL4** *(Optional)*
ADS-B with "out" and "in" capability using VDL Mode 4.
ICAO code is "V2".

k) **EquipmentStatus adscFans** *(Optional)*
ADS-C with FANS 1/A capability.
ICAO code is "D1".

l) **EquipmentStatus adscAtn** *(Optional)*
ADS-C with ATN capability.
ICAO code is "G1".

(4) Constraints:

a)

Name	MODE_S_CAPABILITIES_MISSING_WHEN_MODE_S_EQUIPPED
Attributes	modeS, modeSCapabilities
Description	When modeS is in status equipped the modeSCapabilities shall not be null

b)

Name	MODE_S_CAPABILITIES_PRESENT_WHEN_MODE_S_NOT_EQUIPPED
Attributes	modeS, modeSCapabilities

Description	`modeSCapabilities` shall be null when `modeS` is in status `not_equipped`. Must not be null otherwise

(5) Used by: FlightPlan, FlightPlanUpdate.

4.152. <<enumeration>> SurvivalEquipment

(1) Enumerates survival equipment items that an aircraft can carry.

(2) Values:

a) **DESERT**

b) **JUNGLE**

c) **MARITIME**

d) **POLAR**

(3) Used by: SupplementaryInformation.

4.153. <<enumeration>> SuspensionStatus

(1) Describes a flight suspension status.

(2) Values:

a) **AIRPORT_SUSPENSION**

b) **DELAY_CONFIRMATION**

c) **FLIGHT_PLAN_REVALIDATION**

d) **MANUAL_SUSPENSION**

e) **NOT_REPORTED_AS_AIRBORNE**

f) **NOT_SUSPENDED**

g) **REGULATION_CONFIRMATION**

h) **SIT_TIME_OUT**

i) **SLOT_MISSED**

j) **TRAFFIC_VOLUMES_CONDITION**

(3) Used by: CDMProvisionalInfo, Flight.

4.154. `TargetDPIRequest`

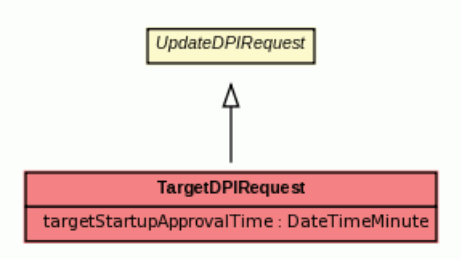

Figure 4.14. `TargetDPIRequest` Class Diagram

(1) There are two subtypes of `Target DPI` message:

(2) a) `T-DPI-t` -- The T-DPI-t message must contain the Target Take-Off Time (TTOT) that takes into account all constraints from an AO and Handling Agent perspective.

b) `T-DPI-s` -- The T-DPI-s contains the Take-Off-Time as calculated by the Pre-Departure Sequence. This Take-Off-Time (target take-off-time) is included in the TTOT-field.

(3) Detailed information regarding `Target DPI target` and `Target DPI sequenced` messages can be found in the document DPI Implementation Guide section "T-DPI-t" and "T-DPI-s".

(4) Inherits from: `UpdateDPIRequest`.

(5) Attributes:

a) **`DateTimeMinute` `targetStartupApprovalTime`** *(Optional)*
Target Start-up Approval Time.
The TSAT is the time at which the pilot can expect start-up approval from ATC.
Acronym: TSAT.

(6) Extended by: TargetDPISequencedRequest, TargetDPITargetRequest.

4.155. `typedef<string> TerminalName_DataType`

(1) Terminal name where the gate is located.

(2) The terminal is a building at an airport where passengers transfer between ground transportation and the facilities that allow them to board and disembark from aircraft.

(3) Examples:T1, 2G, T4S,...

(4) Pattern: `(ALPHA|DIGIT){1,2}`

4.156. `TimeAndModel`

(1) Time at reference location for a given flight model.

(2) Attributes:

a) **`TrafficType`** **`model`** *(Mandatory)*

b) **`DateTimeMinute`** **`time`** *(Mandatory)*

(3) Used by: Flight.

4.157. `typedef<int> TotalCapacity_DataType`

(1) The total capacity, in persons, of all dinghies carried by the aircraft.

(2) Range: `[0,999]`.

(3) Used by: Dinghies.

4.158. `<<enumeration>> TrafficType`

(1) Specifies a traffic type, i.e. if flights are involved in:

(2) Values:

Value	Description
`DEMAND`	Traffic demand ("FTFM": Filed Tactical Flight Model).
`LOAD`	Traffic load ("CTFM": Current Tactical Flight Model).
`REGULATED_DEMAND`	Regulated traffic demand ("RTFM": Regulated Tactical Flight Model).

Table 4.24. <<enumeration>> TrafficType

(3) Used by: TimeAndModel, TrafficCountsReplyData, FlightListRequest, TrafficCountsRequest.

4.159. `TrajectoryPointRole`

(1) Constraints: From the class `TrajectoryPointRole` we will only use the concepts:

a) `topOfClimb`

b) `topOfDescent`

c) `bottomOfClimb`

d) `bottomOfDescent`

(2) If any other `TrajectoryPointRole` value is provided in the B2B request, the `TrajectoryPointRole` value will be ignored.

(3) Attributes:

a) **boolean bottomOfClimb** *(Optional)*
Bottom of Climb (BOC) points for every transition from a cruise phase to a climb phase.

b) **boolean bottomOfDescent** *(Optional)*
Bottom of Descent (BOD) points for every a transition from a descent phase to a cruise phase

c) **boolean boundaryPoint** *(Optional)*
Points where the 4D Trajectory intersects the boundary of FIR/UIRs in whose airspace the flight is planned to fly.

d) **boolean fromGATToOAT** *(Optional)*
Points where the flight pass from GAT to OAT rules. Indicates the start of a military portion of the route.

e) **boolean fromIFRToVFR** *(Optional)*
Points where the flight pass from IFR to VFR rules. Indicates the start of VFR portion of the route.

f) **boolean fromOATToGat** *(Optional)*
Points where the flight pass from OAT to GAT rules. Indicates the end of a military portion of the route.

g) **boolean topOfClimb** *(Optional)*
Top of Climb (TOC) points for every transition from a climb phase to a cruise phase.

h) **boolean topOfDescent** *(Optional)*
Top of Descent (TOD) points for every transition from a cruise phase to a descent phase.

i) **boolean fromVFRToIFR** *(Optional)*
Points where the flight pass from VFR toIVFR rules. Indicates the end of VFR portion of the route.

(4) Used by: FourDTrajectoryPoint.

4.160. <<enumeration>> TrajectoryPointType

(1) Constraints: From the class `TrajectoryPointType` we will use:

a) `adep`

b) `ades`

c) `publishedPoint`

d) `geoPoint`

(2) If any other `TrajectoryPointType` value is provided in the B2B request, the B2B request will be rejected.. Typically for `TrajectoryPointType`:

a) `refPoint`

b) `otherPoint`

(3) Values:

a) **`adep`**
Aerodrome of departure.

b) **`ades`**
Aerodrome of destination.

c) **`geoPoint`**
An ICAO geographical point as it would appear in F15.
It separate 2 route segments.

d) **`otherPoint`**
A point being on a segment between two of points of the other types. Typically a point where performance changes (e.g. where climb rate changes), or where an FIR/UIR boundary is crossed.

e) **`publishedPoint`**
An ICAO point identifier.

f) **`refPoint`**
ICAO point defined in reference to a published point (a Navigation Aid) from which a bearing and a distance id given

(4) Used by: FourDTrajectoryPoint.

4.161. `TrajectorySegment`

(1) Attributes:

a) **`SegmentType segmentType`** *(Mandatory)*

b) **`RouteId segmentIdentifier`** *(Optional)*

(2) Used by: FourDTrajectoryPoint.

4.162. <<abstract>> UpdateDPIRequest

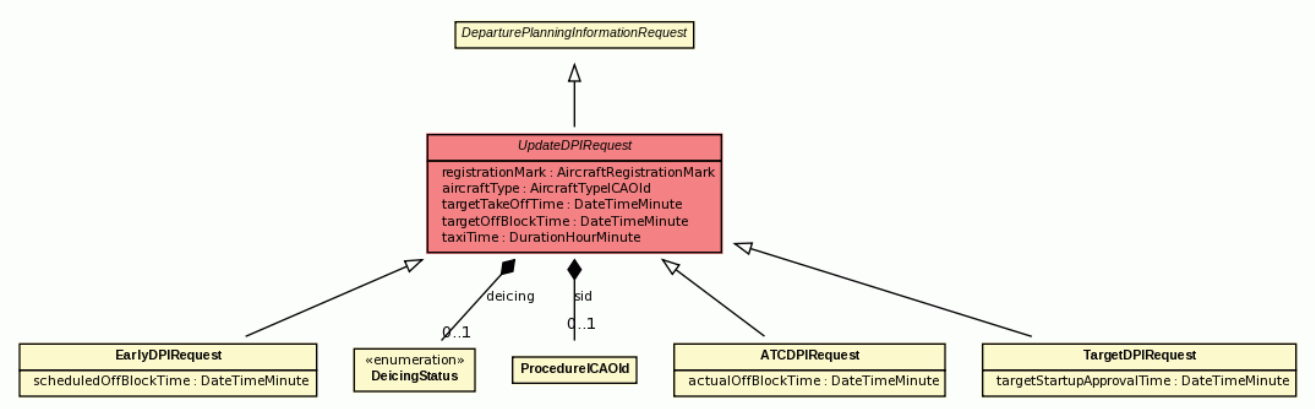

Figure 4.15. <<abstract>> UpdateDPIRequest Class Diagram

(1) Base class of all type of DPI update requests.

(2) Inherits from: DeparturePlanningInformationRequest.

(3) Attributes:

a) **AircraftRegistrationMark registrationMark** *(Optional)*
Aircraft registration mark.

b) **AircraftTypeICAOId aircraftType** *(Optional)*
ICAO Aircraft type.
ADEXP: -ARCTYP

c) **DateTimeMinute targetTakeOffTime** *(Optional)*
Target Take-Off Time.
The TTOT is the most accurate available take-off-time at airport at that moment in time. Time taking into account the TOBT (T-DPI-t message) or the TSAT (T-DPI-s message) plus the Estimated Taxi-Out Time (EXOT)
Acronym: TTOT.

d) **DateTimeMinute targetOffBlockTime** *(Contextual)*
Target Off Block Time.
The time that the aircraft operator or ground handler estimates that the aircraft is ready to push back.
Acronym: TOBT.
Presence:

i) Must be null in ATCDPIRequest

ii) Optional otherwise.

e) **DurationHourMinute taxiTime** *(Mandatory)*

Estimated Taxi out time.
See DPI Implementation Guide section "Use of DPIs during Special Circumstances at the airport" for detail information.

f) **ProcedureICAOId sid** *(Contextual)*
Standard Instrument Departure route identifier.
Acronym: SID.
Presence:

i) Mandatory in ATCDPIRequest

ii) Optional otherwise.

g) **DeicingStatus deicing** *(Optional)*
De-icing planned or in progress.

(4) Extended by: ATCDPIRequest, TargetDPIRequest, EarlyDPIRequest.

4.163. <<enumeration>> WakeTurbulenceCategory

(1) Wake turbulence category.

(2) Values:

Value	Description	ICAO value
HEAVY	Aircraft type with a maximum certificated take-off mass of 136.000 kg or more	H
LIGHT	Aircraft type with a maximum certificated take-off mass of 7.000 kg or less	L
MEDIUM	Aircraft type with a maximum certificated take-off mass of less than 136.000 kg but more than 7.000 kg	M
SUPER	Super heavy aircraft type in the order of 560.000 kg	J

Table 4.25. <<enumeration>> WakeTurbulenceCategory

(3) Used by: FlightPlan, FlightPlanUpdate.

4.164. `WindVector`

(1)

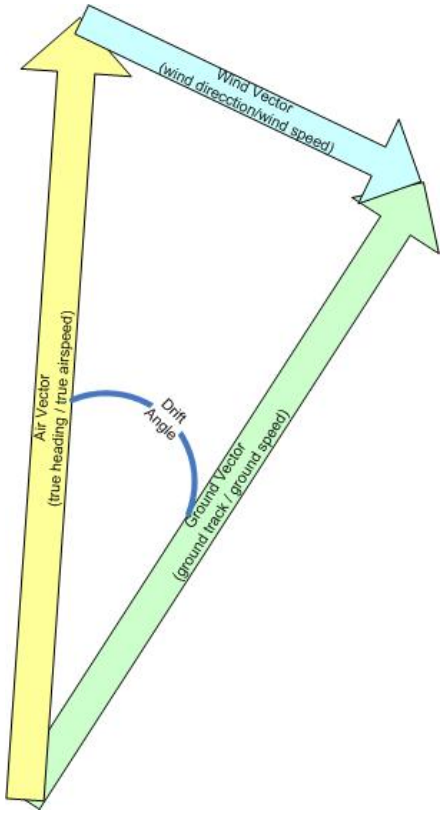

Figure 4.16. Wind Vector

(2) True direction and speed of the wind.

(3) Attributes:

a) **`Bearing windDirection`** *(Mandatory)*
Direction to which the wind is blowing.

b) **`AirSpeed windSpeed`** *(Mandatory)*
Wind velocity.

(4) Used by: FourDTrajectoryPoint.

Chapter 5. PRE-OPS Testing

(1) The airspace data in the PRE_OPS platform is currently synchronised from the OPS data.
The flight data used by the FlightManagementService in the PRE-OPS platform is composed of:

a) Flight data that the different B2B client applications input via NOP/B2B

b) Live feed of the operational flight data received by NM

(2) Flight data is cleaned up daily for terminated flights.

(3) The PRE-OPS platform is not fed with Meteo data, so the flight profiles may be different from the ones obtained from the OPS platform.

(4) The regulation data used by the FlightManagementService in the PRE-OPS platform is fed daily from the live systems.

(5) Regulation data is cleaned up daily.

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