NM 23.0.0 - NOP/B2B Reference Manuals - FlowServices

Edition No.	:	23.0.0.5.26	
Edition Issue Date	:	29/04/2019	
Author	:	NM NOP/B2B Team	
Reference	:	B2B/23.0.0/Flow	
Copy No.	:		← stamp here

Document Control

Copyright Notice

Document Identification

Full Title:	NM 23.0.0 - NOP/B2B Reference Manuals - FlowServices
Total Number of Pages:	279

Table of Contents

References **17**

Terminology **18**

1 Introduction **19**
1.1 Identification 19

2 Context **20**

3 Port Types **21**
3.1 TrafficCountsService Port Type 21
3.1.1 Overview 21
3.1.1.1 Introduction 21
3.1.2 Traffic Count List by Aircraft Operator 21
3.1.2.1 SOAP 21
3.1.2.2 `TrafficCountsByAircraftOperatorRequest` 22
3.1.2.3 `TrafficCountsByAircraftOperatorReply` 23
3.1.3 Traffic Count List by Aerodrome 23
3.1.3.1 SOAP 23
3.1.3.2 `TrafficCountsByAerodromeRequest` 24
3.1.3.3 `TrafficCountsByAerodromeReply` 25
3.1.4 Traffic Count List by Aerodrome Set 25
3.1.4.1 SOAP 25
3.1.4.2 `TrafficCountsByAerodromeSetRequest` 26
3.1.4.3 `TrafficCountsByAerodromeSetReply` 27
3.1.5 Traffic Count List by Airspace 27
3.1.5.1 SOAP 27
3.1.5.2 `TrafficCountsByAirspaceRequest` 28
3.1.5.3 `TrafficCountsByAirspaceReply` 29
3.1.6 Traffic Count List by Point 29
3.1.6.1 SOAP 29
3.1.6.2 `TrafficCountsByPointRequest` 30
3.1.6.3 `TrafficCountsByPointReply` 31
3.1.7 Traffic Count List by Traffic Volume 31
3.1.7.1 SOAP 31
3.1.7.2 `TrafficCountsByTrafficVolumeRequest` 32
3.1.7.3 `TrafficCountsByTrafficVolumeReply` 34
3.2 MeasuresService Port Type 34
3.2.1 Overview 34
3.2.1.1 Introduction 34
3.2.1.2 Update Pattern 35
3.2.1.3 Transaction and Errors 37
3.2.1.4 Allowed Times for Retrievals and Updates 38
3.2.1.4.1 Forecast and Operational Datasets 38
3.2.1.4.2 Past 38

3.2.1.5 Simulations 38
3.2.1.6 Overload protection 38
3.2.2 Regulation List 39
3.2.2.1 SOAP 39
3.2.2.2 RegulationListRequest 39
3.2.2.3 RegulationListReply 41
3.2.3 Regulation Creation 42
3.2.3.1 SOAP 42
3.2.3.2 RegulationCreationRequest 43
3.2.3.3 RegulationCreationReply 45
3.2.4 Regulation Update 45
3.2.4.1 SOAP 45
3.2.4.2 RegulationUpdateRequest 46
3.2.4.3 RegulationUpdateReply 48
3.2.5 Regulation Cancellation 48
3.2.5.1 SOAP 48
3.2.5.2 RegulationCancelRequest 49
3.2.5.3 RegulationCancelReply 50
3.2.6 Regulation Proposal List 50
3.2.6.1 SOAP 50
3.2.6.2 RegulationProposalListRequest 51
3.2.6.3 RegulationProposalListReply 53
3.2.7 Regulation Proposal Filing 54
3.2.7.1 SOAP 54
3.2.7.2 RegulationProposalFilingRequest 55
3.2.7.3 RegulationProposalFilingReply 58
3.2.8 Updating a filed Regulation Proposal 59
3.2.8.1 SOAP 59
3.2.8.2 RegulationProposalUpdateRequest 60
3.2.8.3 RegulationProposalUpdateReply 62
3.2.9 Regulation Proposal Revocation 63
3.2.9.1 SOAP 63
3.2.9.2 RegulationProposalRevocationRequest 63
3.2.9.3 RegulationProposalRevocationReply 65
3.2.10 Rerouting List 66
3.2.10.1 SOAP 66
3.2.10.2 ReroutingListRequest 66
3.2.10.3 ReroutingListReply 68
3.2.11 Rerouting Creation 69
3.2.11.1 SOAP 69
3.2.11.2 ReroutingCreationRequest 69
3.2.11.3 ReroutingCreationReply 71
3.2.12 Rerouting Update 71
3.2.12.1 SOAP 71
3.2.12.2 ReroutingUpdateRequest 72
3.2.12.3 ReroutingUpdateReply 73
3.2.13 Rerouting Cancellation 74

3.2.13.1 SOAP 74
3.2.13.2 ReroutingCancelRequest 74
3.2.13.3 ReroutingCancelReply 75
3.2.14 MCDMOnly List 76
3.2.14.1 SOAP 76
3.2.14.2 MCDMOnlyListRequest 76
3.2.14.3 MCDMOnlyListReply 78
3.2.15 MCDMOnly Creation 79
3.2.15.1 SOAP 79
3.2.15.2 MCDMOnlyCreationRequest 80
3.2.15.3 MCDMOnlyCreationReply 81
3.2.16 MCDMOnly Update 82
3.2.16.1 SOAP 82
3.2.16.2 MCDMOnlyUpdateRequest 82
3.2.16.3 MCDMOnlyUpdateReply 83
3.2.17 MCDMOnly Cancellation 84
3.2.17.1 SOAP 84
3.2.17.2 MCDMOnlyCancelRequest 84
3.2.17.3 MCDMOnlyCancelReply 85
3.2.18 Measure Operational Log Retrieval 86
3.2.18.1 SOAP 86
3.2.18.2 MeasureOpLogRetrievalRequest 86
3.2.18.3 MeasureOpLogRetrievalReply 87
3.2.19 Update Flights in Measure 88
3.2.19.1 SOAP 88
3.2.19.2 UpdateFlightsInMeasureRequest 88
3.2.19.3 UpdateFlightsInMeasureReply 90
3.2.20 ATFCM Situation Retrieval 90
3.2.20.1 SOAP 90
3.2.20.2 ATFCMSituationRequest 91
3.2.20.3 ATFCMSituationReply 92
3.2.21 Retrieve Network Impact Assessment 93
3.2.21.1 SOAP 93
3.2.21.2 NetworkImpactAssessmentRetrievalRequest 93
3.2.21.3 NetworkImpactAssessmentRetrievalReply 95
3.3 McdmService Port Type 96
3.3.1 Overview 96
3.3.1.1 Introduction 96
3.3.1.2 MCDM Operations 96
3.3.1.3 Update Pattern 97
3.3.2 MCDM Topic List 97
3.3.2.1 SOAP 97
3.3.2.2 MCDMTopicListRequest 98
3.3.2.3 MCDMTopicListReply 101
3.3.3 MCDM Topic Update 102
3.3.3.1 SOAP 102
3.3.3.2 MCDMTopicUpdateRequest 103

3.3.3.3 MCDMTopicUpdateReply 103
3.3.4 MCDM Topic Message Retrieval 104
3.3.4.1 SOAP 104
3.3.4.2 MCDMTopicMessageRetrievalRequest 104
3.3.4.3 MCDMTopicMessageRetrievalReply 105
3.3.5 MCDM State Update 105
3.3.5.1 SOAP 105
3.3.5.2 MCDMStateUpdateRequest 106
3.3.5.3 MCDMStateUpdateReply 107
3.4 TacticalUpdatesService Port Type 108
3.4.1 Overview 108
3.4.1.1 Introduction 108
3.4.1.2 CACD Retrievals vs. Tactical Situation 108
3.4.1.3 Daily Plan Update Pattern 109
3.4.1.4 Transactions and Errors 112
3.4.1.5 Allowed Times for Retrievals and Updates 113
3.4.1.5.1 Forecast and Operational Datasets 113
3.4.1.5.2 Past 114
3.4.2 Retrieve a Sector Configuration Plan 114
3.4.2.1 SOAP 114
3.4.2.2 SectorConfigurationPlanRetrievalRequest 115
3.4.2.3 SectorConfigurationPlanRetrievalReply 115
3.4.3 Update a Sector Configuration Plan 116
3.4.3.1 SOAP 116
3.4.3.2 SectorConfigurationPlanUpdateRequest 116
3.4.3.3 SectorConfigurationPlanUpdateReply 117
3.4.4 Retrieve a Capacity Plan 117
3.4.4.1 SOAP 117
3.4.4.2 CapacityPlanRetrievalRequest 118
3.4.4.3 CapacityPlanRetrievalReply 118
3.4.5 Update a Capacity Plan 119
3.4.5.1 SOAP 119
3.4.5.2 CapacityPlanUpdateRequest 119
3.4.5.3 CapacityPlanUpdateReply 120
3.4.6 Retrieve a Traffic Volume Activation Plan 120
3.4.6.1 SOAP 120
3.4.6.2 TrafficVolumeActivationPlanRetrievalRequest 121
3.4.6.3 TrafficVolumeActivationPlanRetrievalReply 121
3.4.7 Update a Traffic Volume Activation Plan 122
3.4.7.1 SOAP 122
3.4.7.2 TrafficVolumeActivationPlanUpdateRequest 122
3.4.7.3 TrafficVolumeActivationPlanUpdateReply 123
3.4.8 Retrieve an OTMV Plan 123
3.4.8.1 SOAP 123
3.4.8.2 OTMVPlanRetrievalRequest 124
3.4.8.3 OTMVPlanRetrievalReply 124
3.4.9 Update an OTMV Plan 125

3.4.9.1 SOAP 125
3.4.9.2 OTMVPlanUpdateRequest 125
3.4.9.3 OTMVPlanUpdateReply 126
3.4.10 Retrieve a runway configuration plan 127
3.4.10.1 SOAP 127
3.4.10.2 RunwayConfigurationPlanRetrievalRequest 127
3.4.10.3 RunwayConfigurationPlanRetrievalReply 127
3.4.11 Update a runway configuration plan 128
3.4.11.1 SOAP 128
3.4.11.2 RunwayConfigurationPlanUpdateRequest 128
3.4.11.3 RunwayConfigurationPlanUpdateReply 129
3.4.12 Hotspot List 129
3.4.12.1 SOAP 129
3.4.12.2 HotspotListRequest 130
3.4.12.3 HotspotListReply 132
3.4.13 Hotspot Plan Update 132
3.4.13.1 SOAP 132
3.4.13.2 HotspotPlanUpdateRequest 133
3.4.13.3 HotspotPlanUpdateReply 134
3.5 SimulationsService Port Type 134
3.5.1 Overview 134
3.5.1.1 Introduction 134
3.5.1.2 Simulations 134
3.5.2 Simulation List 136
3.5.2.1 SOAP 136
3.5.2.2 SimulationListRequest 136
3.5.2.3 SimulationListReply 138
3.5.3 Simulation Availability List 138
3.5.3.1 SOAP 138
3.5.3.2 SimulationAvailabilityRequest 139
3.5.3.3 SimulationAvailabilityReply 140
3.5.4 Simulation Start 140
3.5.4.1 SOAP 140
3.5.4.2 SimulationStartRequest 141
3.5.4.3 SimulationStartReply 142
3.5.5 Simulation Stop 142
3.5.5.1 SOAP 142
3.5.5.2 SimulationStopRequest 143
3.5.5.3 SimulationStopReply 144
3.6 Scenario Repository Port Type 144
3.6.1 Overview 144
3.6.1.1 Introduction 144
3.6.1.2 Scenario Repository 144
3.6.2 Retrieve list of Regulations from Scenario 146
3.6.2.1 SOAP 146
3.6.2.2 ScenarioRegulationRetrievalRequest 146
3.6.2.3 ScenarioRegulationRetrievalReply 147

3.6.3 Retrieve list of Reroutings from Scenario 147
3.6.3.1 SOAP 147
3.6.3.2 ScenarioReroutingRetrievalRequest 148
3.6.3.3 ScenarioReroutingRetrievalReply 148
3.6.4 Retrieve list of MCDMOnly from Scenario 149
3.6.4.1 SOAP 149
3.6.4.2 ScenarioMCDMOnlyRetrievalRequest 149
3.6.4.3 ScenarioMCDMOnlyRetrievalReply 150
3.6.5 List of Scenarios 150
3.6.5.1 SOAP 150
3.6.5.2 ScenarioListRequest 151
3.6.5.3 ScenarioListReply 152

4 Data Types 153
4.1 AerodromeLocation 153
4.2 AirspaceLocation 153
4.3 AllFlightsLocation 153
4.4 ANDedReroutingConstraints 153
4.5 AoLocation 153
4.6 ATFCMSituationCounts 153
4.7 ATFCMSituationDelays 154
4.8 ATFCMSituationRegulation 155
4.9 <<enumeration>> AvoidAirspaceReroutingKind 155
4.10 AvoidViaAirspaceReroutingConstraint 156
4.11 AvoidViaPointReroutingConstraint 156
4.12 <<enumeration>> AvoidViaReroutingType 156
4.13 typedef<int> Capacity 157
4.14 CapacityPlans 157
4.15 CherryPickedLocation 158
4.16 Counts 158
4.17 <<enumeration>> CountsCalculationType 158
4.18 CountsCalculationTypeAndInterval 159
4.19 CountsInterval 159
4.20 typedef<int> CountsValue 160
4.21 DatasetReference 160
4.22 <<strict enumeration>> DatasetReferenceType 161
4.23 DelayLocation 162
4.24 DeltaATFCMSituation 162
4.25 DeltaATFCMSituationCounts 162
4.26 DeltaATFCMSituationDelays 163
4.27 DeltaATFCMSituationRegulation 164
4.28 DeltaCounts 164
4.29 DeltaLevelReroutingConstraint 165
4.30 ExcludeReIncludeFlightInRegulation 165
4.31 FlightAtfcmMcdmOnlyLocation 165
4.32 <<abstract>> FlightAtfcmMeasureLocation 165

4.33 FlightAtfcmRegulationLocation 166
4.34 FlightAtfcmReroutingLocation 166
4.35 FlightHotspotLocation 167
4.36 FlightMCDMInfo 167
4.37 FlightRegulationLocation 168
4.38 Flow 168
4.39 typedef<string> FlowId 169
4.40 <<enumeration>> FlowRoleSelection 169
4.41 <<enumeration>> FlowType 170
4.42 ForceFlightInRegulation 171
4.43 <<enumeration>> FreezeTP 172
4.44 <<enumeration>> GroupReroutingIndicator 172
4.45 Hotspot 173
4.46 <<enumeration>> HotspotField 173
4.47 HotspotId 174
4.48 <<enumeration>> HotspotKind 174
4.49 HotspotPlans 175
4.50 <<enumeration>> HotspotSeverity 176
4.51 <<enumeration>> HotspotStatus 176
4.52 <<enumeration>> KindOfRestriction 177
4.53 LevelAndSpeedReroutingConstraint 177
4.54 <<enumeration>> LevelConstraintKind 177
4.55 LifeCycleEvent 178
4.56 <<enumeration>> LifeCycleEventType 178
4.57 <<abstract>> Location 179
4.58 <<enumeration>> MCDMApprovalState 179
4.59 <<enumeration>> MCDMCoordinationLevel 179
4.60 MCDMDeadlines 180
4.61 MCDMFlightTopic 181
4.62 MCDMHotspotTopic 182
4.63 MCDMMeasureTopic 183
4.64 MCDMMessage 185
4.65 <<enumeration>> MCDMMessageType 185
4.66 MCDMOnly 186
4.67 <<enumeration>> MCDMOnlyField 187
4.68 <<enumeration>> MCDMRole 188
4.69 MCDMRoleUserCategory 189
4.70 <<enumeration>> MCDMState 189
4.71 <<abstract>> MCDMStatefulTopic 192
4.72 <<abstract>> MCDMTopic 194
4.73 <<enumeration>> MCDMTopicField 195
4.74 typedef<string> MCDMTopicId 196
4.75 MCDMTopicListRequestSelector 196
4.76 MCDMUserAndRole 196
4.77 <<enumeration>> MCDMUserCategory 197

4.78 MCDMUserRoleAndApprovalState 197
4.79 <<abstract>> Measure 198
4.80 MeasureFromScenarioRepository 201
4.81 MeasureId 202
4.82 MeasureIdAndTV 202
4.83 MeasureListReplyData 202
4.84 <<abstract>> MeasureListRequest 203
4.85 <<enumeration>> MeasureSubType 205
4.86 NMOCManagedSimulation 205
4.87 OtherReroutingConstraint 206
4.88 OTMV 206
4.89 OtmvAlert 207
4.90 OTMVPeak 207
4.91 OTMVPlan 207
4.92 OTMVPlanForDuration 209
4.93 OTMVPlans 210
4.94 <<enumeration>> OtmvStatus 211
4.95 OTMVSustained 212
4.96 typedef<int> OTMVThreshold 212
4.97 OTMVWithDuration 212
4.98 <<strict enumeration>> PlanDataSource 213
4.99 PlannedCapacities 213
4.100 PlannedCapacity 214
4.101 PlannedOTMV 215
4.102 PlannedRunwayConfigurations 216
4.103 PlannedSectorConfigurationActivation 217
4.104 PlannedTrafficVolumeActivation 218
4.105 PlannedTrafficVolumeActivations 219
4.106 PointLocation 220
4.107 Regulation 220
4.108 RegulationCause 221
4.109 RegulationExceptionalConstraint 221
4.110 <<enumeration>> RegulationField 222
4.111 typedef<string> RegulationId 223
4.112 typedef<string> RegulationIdWildcard 224
4.113 RegulationInitialConstraint 224
4.114 <<enumeration>> RegulationLocationCategory 224
4.115 <<abstract>> RegulationOrMCDMOnly 225
4.116 RegulationOrMCDMOnlyListReplyData 229
4.117 <<abstract>> RegulationOrMCDMOnlyListRequest 230
4.118 RegulationProposal 231
4.119 <<enumeration>> RegulationProposalAction 233
4.120 <<enumeration>> RegulationProposalField 234
4.121 <<enumeration>> RegulationProposalKind 235
4.122 <<enumeration>> RegulationProposalState 235

4.123 <<enumeration>> RegulationReason 236
4.124 <<enumeration>> RegulationState 236
4.125 RegulationSupplementaryConstraint 237
4.126 <<enumeration>> RepositoryId 237
4.127 Rerouting 238
4.128 <<enumeration>> ReroutingApplyKind 240
4.129 <<abstract>> ReroutingConstraint 241
4.130 <<enumeration>> ReroutingField 241
4.131 ReroutingHorizontalConstraints 242
4.132 typedef<string> ReroutingId 243
4.133 typedef<string> ReroutingIdWildcard 243
4.134 ReroutingManualConstraints 243
4.135 <<enumeration>> ReroutingMeasureState 244
4.136 ReroutingSourcesAndConstraints 245
4.137 RestrictionLocation 246
4.138 RunwayConfiguration 246
4.139 RunwayConfigurationPlan 247
4.140 <<enumeration>> RunwayUsage 249
4.141 ScenarioAttributes 250
4.142 typedef<string> ScenarioEvent 252
4.143 typedef<string> ScenarioId 252
4.144 ScenarioImpact 253
4.145 ScenarioLevelConstraint 253
4.146 ScenarioMeasureRetrievalRequest 254
4.147 typedef<string> ScenarioName 255
4.148 <<enumeration>> ScenarioPublicationStatus 255
4.149 <<enumeration>> ScenarioTrafficVolumeMatchingKind 255
4.150 <<enumeration>> ScenarioType 256
4.151 typedef<string> SectorConfigurationId 257
4.152 SectorConfigurationPlan 257
4.153 SegmentLocation 259
4.154 SetOfAerodromesLocation 259
4.155 SignificantDeltaCount 259
4.156 SignificantDeltaDuration 259
4.157 <<abstract>> Simulation 260
4.158 SimulationAvailability 261
4.159 typedef<string> SimulationName 262
4.160 <<enumeration>> SimulationType 262
4.161 StandaloneSimex 262
4.162 <<enumeration>> SubTotalsTrafficCountsType 263
4.163 <<abstract>> TacticalConfigurationPlan 263
4.164 <<abstract>> TacticalConfigurationRetrievalRequest 265
4.165 <<abstract>> TrafficCountsReplyData 266
4.166 <<abstract>> TrafficCountsRequest 267
4.167 TrafficVolumeActivationPlan 269

4.168 TrafficVolumeActivationPlans 271
4.169 typedef<string> TrafficVolumeId 272
4.170 TrafficVolumeLocation 273
4.171 TrafficVolumeScenarios 274
4.172 UpdateFlightInMeasure 275
4.173 UpdateFlightInMeasureChoice 275
4.174 UserManagedSimulation 276

5 PRE-OPS Testing 278

DOCUMENT FINAL PAGE 279

List of Figures

3.1 TrafficCountsByAircraftOperatorRequest Class Diagram 22
3.2 TrafficCountsByAircraftOperatorReply Class Diagram 23
3.3 TrafficCountsByAerodromeRequest Class Diagram 24
3.4 TrafficCountsByAerodromeReply Class Diagram 25
3.5 TrafficCountsByAerodromeSetRequest Class Diagram 26
3.6 TrafficCountsByAerodromeSetReply Class Diagram 27
3.7 TrafficCountsByAirspaceRequest Class Diagram 28
3.8 TrafficCountsByAirspaceReply Class Diagram 29
3.9 TrafficCountsByPointRequest Class Diagram 30
3.10 TrafficCountsByPointReply Class Diagram 31
3.11 TrafficCountsByTrafficVolumeRequest Class Diagram 32
3.12 TrafficCountsByTrafficVolumeReply Class Diagram 34
3.13 RegulationListRequest Class Diagram 39
3.14 RegulationListReply Class Diagram 41
3.15 RegulationCreationRequest Class Diagram 43
3.16 RegulationCreationReply Class Diagram 45
3.17 RegulationUpdateRequest Class Diagram 46
3.18 RegulationUpdateReply Class Diagram 48
3.19 RegulationCancelRequest Class Diagram 49
3.20 RegulationCancelReply Class Diagram 50
3.21 RegulationProposalListRequest Class Diagram 51
3.22 RegulationProposalListReply Class Diagram 53
3.23 RegulationProposalFilingRequest Class Diagram 55
3.24 RegulationProposalFilingReply Class Diagram 58
3.25 RegulationProposalUpdateRequest Class Diagram 60
3.26 RegulationProposalUpdateReply Class Diagram 62
3.27 RegulationProposalRevocationRequest Class Diagram 63
3.28 RegulationProposalRevocationReply Class Diagram 65
3.29 ReroutingListRequest Class Diagram 66
3.30 ReroutingListReply Class Diagram 68
3.31 ReroutingCreationRequest Class Diagram 69
3.32 ReroutingCreationReply Class Diagram 71
3.33 ReroutingUpdateRequest Class Diagram 72
3.34 ReroutingUpdateReply Class Diagram 73
3.35 ReroutingCancelRequest Class Diagram 74
3.36 ReroutingCancelReply Class Diagram 75
3.37 MCDMOnlyListRequest Class Diagram 76
3.38 MCDMOnlyListReply Class Diagram 78
3.39 MCDMOnlyCreationRequest Class Diagram 80
3.40 MCDMOnlyCreationReply Class Diagram 81
3.41 MCDMOnlyUpdateRequest Class Diagram 82
3.42 MCDMOnlyUpdateReply Class Diagram 83
3.43 MCDMOnlyCancelRequest Class Diagram 84

3.44 MCDMOnlyCancelReply Class Diagram 85
3.45 MeasureOpLogRetrievalRequest Class Diagram 86
3.46 MeasureOpLogRetrievalReply Class Diagram 87
3.47 UpdateFlightsInMeasureRequest Class Diagram 88
3.48 UpdateFlightsInMeasureReply Class Diagram 90
3.49 ATFCMSituationRequest Class Diagram 91
3.50 ATFCMSituationReply Class Diagram 92
3.51 NetworkImpactAssessmentRetrievalRequest Class Diagram 93
3.52 NetworkImpactAssessmentRetrievalReply Class Diagram 95
3.53 MCDMTopicListRequest Class Diagram 98
3.54 MCDMTopicListReply Class Diagram 101
3.55 MCDMTopicUpdateRequest Class Diagram 103
3.56 MCDMTopicMessageRetrievalRequest Class Diagram 104
3.57 MCDMTopicMessageRetrievalReply Class Diagram 105
3.58 MCDMStateUpdateRequest Class Diagram 106
3.59 MCDMStateUpdateReply Class Diagram 107
3.60 SectorConfigurationPlanRetrievalRequest Class Diagram 115
3.61 SectorConfigurationPlanRetrievalReply Class Diagram 115
3.62 SectorConfigurationPlanUpdateRequest Class Diagram 116
3.63 SectorConfigurationPlanUpdateReply Class Diagram 117
3.64 CapacityPlanRetrievalRequest Class Diagram 118
3.65 CapacityPlanRetrievalReply Class Diagram 118
3.66 CapacityPlanUpdateRequest Class Diagram 119
3.67 CapacityPlanUpdateReply Class Diagram 120
3.68 TrafficVolumeActivationPlanRetrievalRequest Class Diagram 121
3.69 TrafficVolumeActivationPlanRetrievalReply Class Diagram 121
3.70 TrafficVolumeActivationPlanUpdateRequest Class Diagram 122
3.71 TrafficVolumeActivationPlanUpdateReply Class Diagram 123
3.72 OTMVPlanRetrievalRequest Class Diagram 124
3.73 OTMVPlanRetrievalReply Class Diagram 124
3.74 OTMVPlanUpdateRequest Class Diagram 125
3.75 OTMVPlanUpdateReply Class Diagram 126
3.76 RunwayConfigurationPlanRetrievalRequest Class Diagram 127
3.77 RunwayConfigurationPlanRetrievalReply Class Diagram 127
3.78 RunwayConfigurationPlanUpdateRequest Class Diagram 128
3.79 RunwayConfigurationPlanUpdateReply Class Diagram 129
3.80 HotspotListRequest Class Diagram 130
3.81 HotspotListReply Class Diagram 132
3.82 HotspotPlanUpdateRequest Class Diagram 133
3.83 SimulationListRequest Class Diagram 136
3.84 SimulationListReply Class Diagram 138
3.85 SimulationAvailabilityRequest Class Diagram 139
3.86 SimulationAvailabilityReply Class Diagram 140
3.87 SimulationStartRequest Class Diagram 141
3.88 SimulationStartReply Class Diagram 142

3.89 SimulationStopRequest Class Diagram 143
3.90 SimulationStopReply Class Diagram 144
3.91 ScenarioRegulationRetrievalRequest Class Diagram 146
3.92 ScenarioRegulationRetrievalReply Class Diagram 147
3.93 ScenarioReroutingRetrievalRequest Class Diagram 148
3.94 ScenarioReroutingRetrievalReply Class Diagram 148
3.95 ScenarioMCDMOnlyRetrievalRequest Class Diagram 149
3.96 ScenarioMCDMOnlyRetrievalReply Class Diagram 150
3.97 ScenarioListRequest Class Diagram 151
3.98 ScenarioListReply Class Diagram 152
4.1 DeltaATFCMSituation Class Diagram 162
4.2 <<abstract>> Location Class Diagram 179
4.3 MCDMFlightTopic Class Diagram 181
4.4 MCDMHotspotTopic Class Diagram 182
4.5 MCDMMeasureTopic Class Diagram 183
4.6 MCDMOnly Class Diagram 186
4.7 MCDM transition diagram 190
4.8 <<abstract>> MCDMStatefulTopic Class Diagram 192
4.9 <<abstract>> MCDMTopic Class Diagram 194
4.10 <<abstract>> Measure Class Diagram 198
4.11 MeasureFromScenarioRepository Class Diagram 201
4.12 <<abstract>> MeasureListRequest Class Diagram 203
4.13 Regulation Class Diagram 220
4.14 <<abstract>> RegulationOrMCDMOnly Class Diagram 225
4.15 <<abstract>> RegulationOrMCDMOnlyListRequest Class Diagram 230
4.16 RegulationProposal Class Diagram 231
4.17 Rerouting Class Diagram 238
4.18 <<abstract>> ReroutingConstraint Class Diagram 241
4.19 ReroutingSourcesAndConstraints Class Diagram 245
4.20 ScenarioAttributes Class Diagram 250
4.21 ScenarioLevelConstraint Class Diagram 253
4.22 ScenarioMeasureRetrievalRequest Class Diagram 254
4.23 <<enumeration>> ScenarioPublicationStatus Class Diagram 255
4.24 <<enumeration>> ScenarioTrafficVolumeMatchingKind Class Diagram 255
4.25 <<enumeration>> ScenarioType Class Diagram 256
4.26 <<abstract>> Simulation Class Diagram 260
4.27 <<abstract>> TacticalConfigurationRetrievalRequest Class Diagram 265
4.28 <<abstract>> TrafficCountsReplyData Class Diagram 266
4.29 <<abstract>> TrafficCountsRequest Class Diagram 267

List of Tables

4.1 <<enumeration>> GroupReroutingIndicator 173
4.2 <<enumeration>> MCDMTopicField 195
4.3 <<enumeration>> OtmvStatus 212
4.4 <<enumeration>> RegulationProposalAction 234
4.5 <<enumeration>> RegulationProposalState 236
4.6 <<enumeration>> RegulationState 237

References

NM

[1] NOP/B2B Reference Manuals - CommonServices

[2] NOP/B2B Reference Manuals - AirspaceServices

[3] NOP/B2B Reference Manuals - FlightServices

Terminology

Main Abbreviations and Acronyms

STAM (Short-Term ATFCM Measures)
: An approach to smooth sector workloads by reducing traffic peaks through short-term application of minor ground delays, appropriate flight level capping and exiguous rerouting to a limited number of flights.

Chapter 1. Introduction

1.1. Identification

(1) This document forms part of the set of the NM 23.0.0 - NOP/B2B Reference Manuals, which all together form the NM 23.0.0 - NOP/B2B Documentation.

(2) Its reference is B2B/23.0.0/Flow.

(3) Its title is NM 23.0.0 - NOP/B2B Reference Manuals - FlowServices.

Chapter 2. Context

(1) The FlowServices service group is intended to provide services in the flow and capacity management domain.

Chapter 3. Port Types

3.1. TrafficCountsService Port Type

3.1.1. Overview

3.1.1.1. Introduction

(1) This service is intended to provide querying of traffic counts. The requests currently available are:

a) TrafficCountsByAircraftOperatorRequest / TrafficCountsByAircraftOperatorReply

b) TrafficCountsByAerodromeRequest / TrafficCountsByAerodromeReply

c) TrafficCountsByAerodromeSetRequest / TrafficCountsByAerodromeSetReply

d) TrafficCountsByAirspaceRequest / TrafficCountsByAirspaceReply

e) TrafficCountsByPointRequest / TrafficCountsByPointReply

f) TrafficCountsByTrafficVolumeRequest / TrafficCountsByTrafficVolumeReply

3.1.2. Traffic Count List by Aircraft Operator

3.1.2.1. SOAP

(1) The associated SOAP operation is:

```
TrafficCountsByAircraftOperatorReply queryTrafficCountsByAircraftOperator(
    TrafficCountsByAircraftOperatorRequest request
    )
```

3.1.2.2. `TrafficCountsByAircraftOperatorRequest`

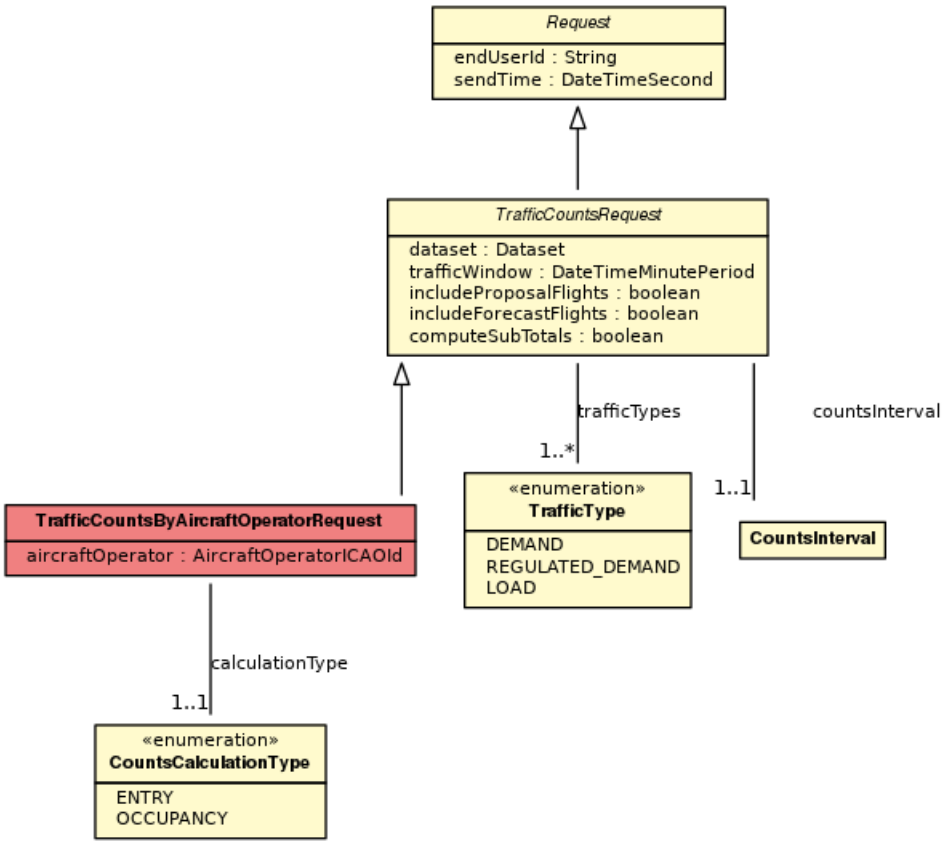

Figure 3.1. `TrafficCountsByAircraftOperatorRequest` Class Diagram

(1) Request to query the NM traffic counts for an aircraft operator.

(2) Inherits from: `TrafficCountsRequest`

(3) Attributes:

a) **`AircraftOperatorICAOId` `aircraftOperator`** *(Mandatory)*
The Aircraft operator ICAO Id for which traffic counts are requested.

b) **`CountsCalculationType` `calculationType`** *(Mandatory)*
Indicates what is the calculation type of the count (entry or occupancy).

3.1.2.3. `TrafficCountsByAircraftOperatorReply`

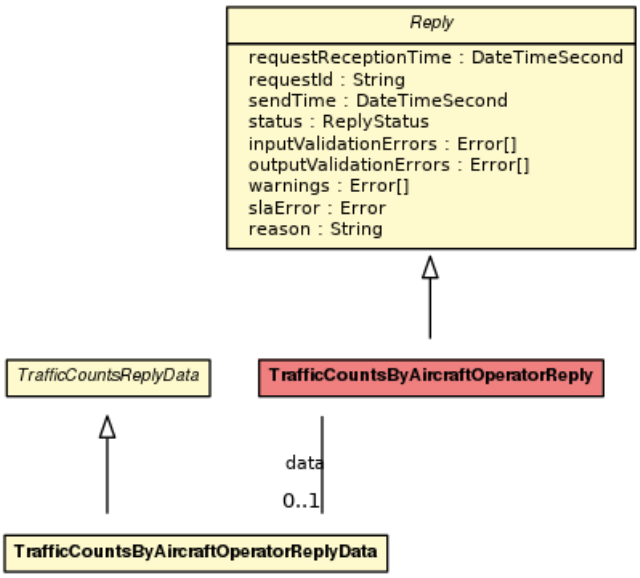

Figure 3.2. `TrafficCountsByAircraftOperatorReply` Class Diagram

(1) Reply returned in response to TrafficCountsByAircraftOperatorRequest.

(2) See TrafficCountsReplyData.

(3) Inherits from: `Reply`

3.1.3. Traffic Count List by Aerodrome

3.1.3.1. SOAP

(1) The associated SOAP operation is:

```
TrafficCountsByAerodromeReply queryTrafficCountsByAerodrome(
    TrafficCountsByAerodromeRequest request
    )
```

3.1.3.2. `TrafficCountsByAerodromeRequest`

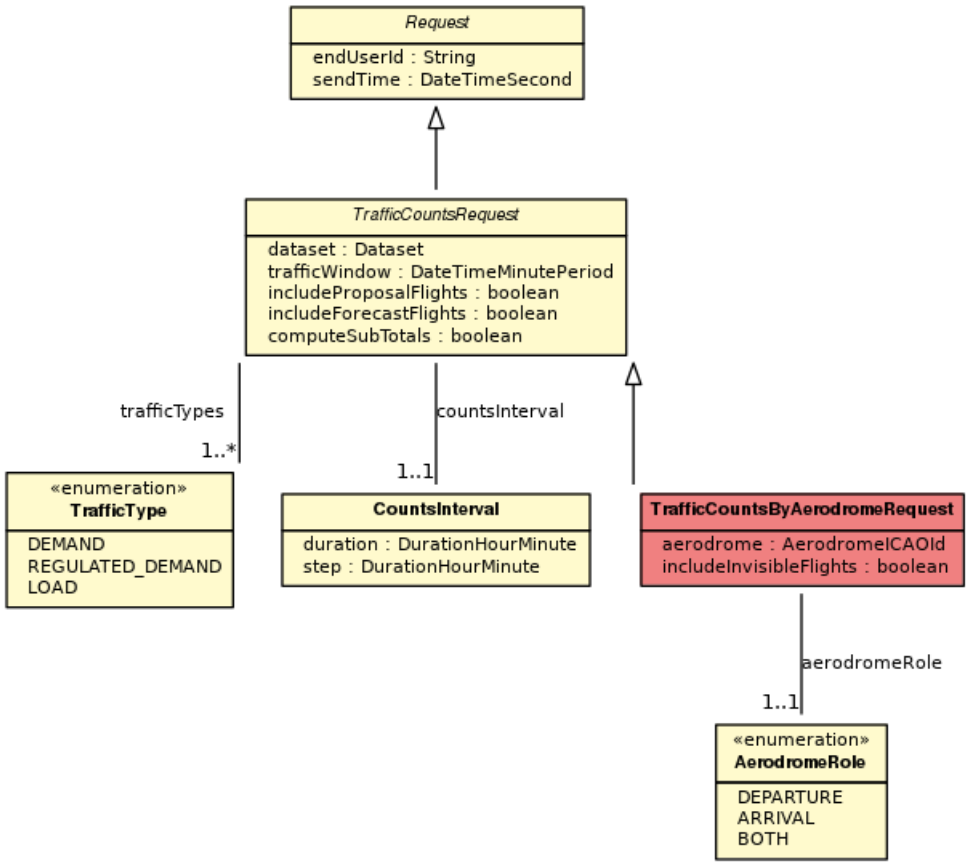

Figure 3.3. `TrafficCountsByAerodromeRequest` Class Diagram

(1) Request to query the NM traffic ENTRY counts for an aerodrome.

(2) Inherits from: `TrafficCountsRequest`

(3) 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.

c) **`boolean includeInvisibleFlights`** *(Optional)*
Indicates whether invisible flights (VFR, OAT, STAY, IFPSTOP) shall be included in the traffic counts.
Defaults to `false`.

3.1.3.3. `TrafficCountsByAerodromeReply`

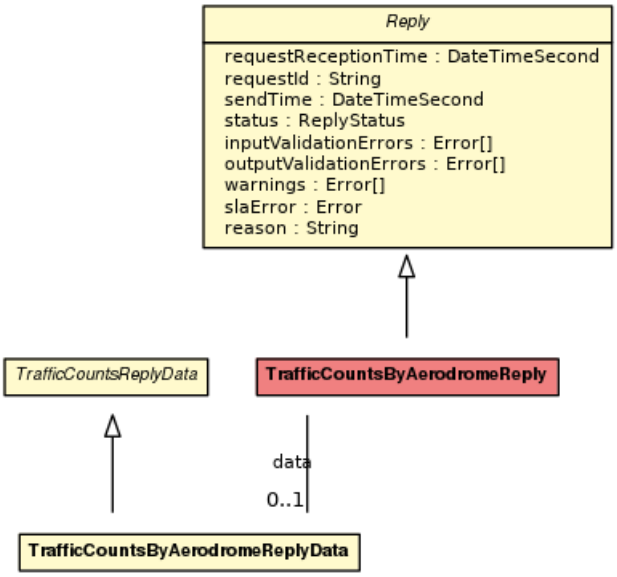

Figure 3.4. `TrafficCountsByAerodromeReply` Class Diagram

(1) Reply returned in response to TrafficCountsByAerodromeRequest.

(2) See TrafficCountsReplyData.

(3) Inherits from: `Reply`

3.1.4. Traffic Count List by Aerodrome Set

3.1.4.1. SOAP

(1) The associated SOAP operation is:

```
TrafficCountsByAerodromeSetReply queryTrafficCountsByAerodromeSet(
    TrafficCountsByAerodromeSetRequest request
    )
```

3.1.4.2. `TrafficCountsByAerodromeSetRequest`

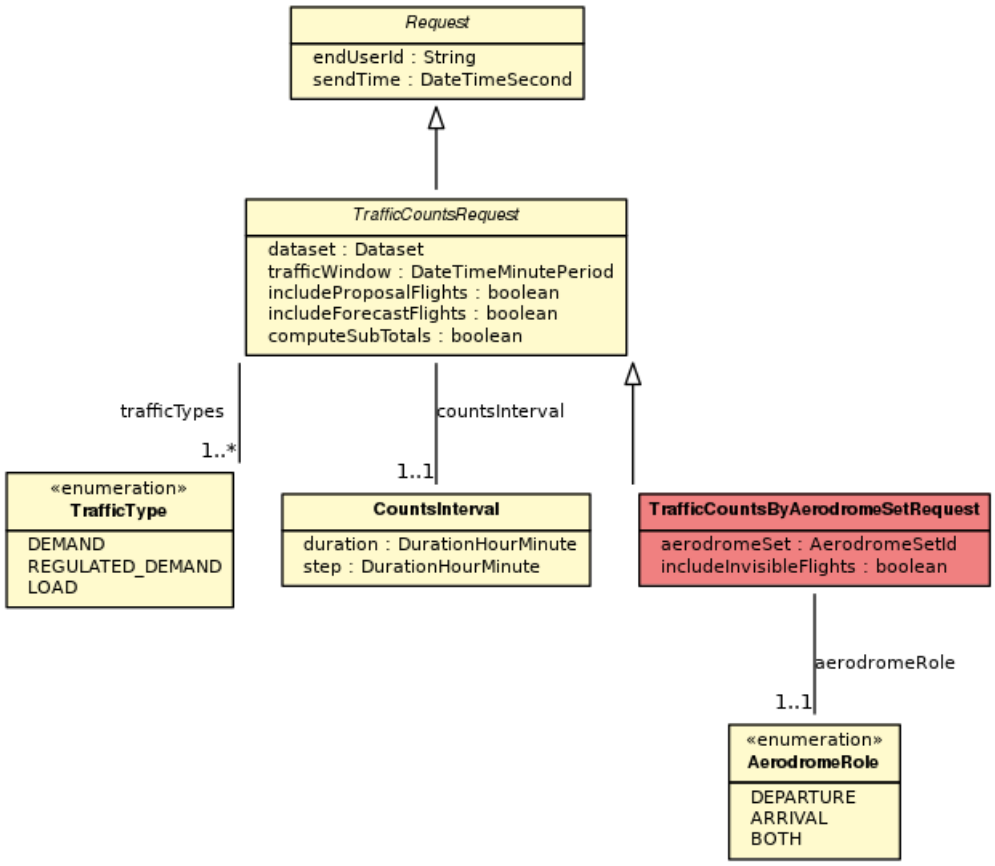

Figure 3.5. `TrafficCountsByAerodromeSetRequest` Class Diagram

(1) Request to query the NM traffic ENTRY counts for an aerodrome set.

(2) Inherits from: `TrafficCountsRequest`

(3) 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.

c) **`boolean includeInvisibleFlights`** *(Optional)*
Indicates whether invisible flights (VFR, OAT, STAY, IFPSTOP) shall be included in the traffic counts.
Defaults to `false`.

3.1.4.3. `TrafficCountsByAerodromeSetReply`

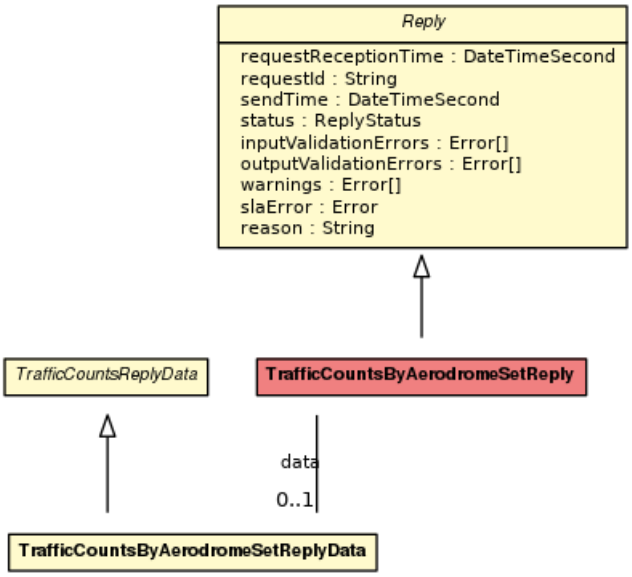

Figure 3.6. `TrafficCountsByAerodromeSetReply` Class Diagram

(1) Reply returned in response to TrafficCountsByAerodromeSetRequest.

(2) See TrafficCountsReplyData.

(3) Inherits from: `Reply`

3.1.5. Traffic Count List by Airspace

3.1.5.1. SOAP

(1) The associated SOAP operation is:

```
TrafficCountsByAirspaceReply queryTrafficCountsByAirspace(
    TrafficCountsByAirspaceRequest request
    )
```

3.1.5.2. `TrafficCountsByAirspaceRequest`

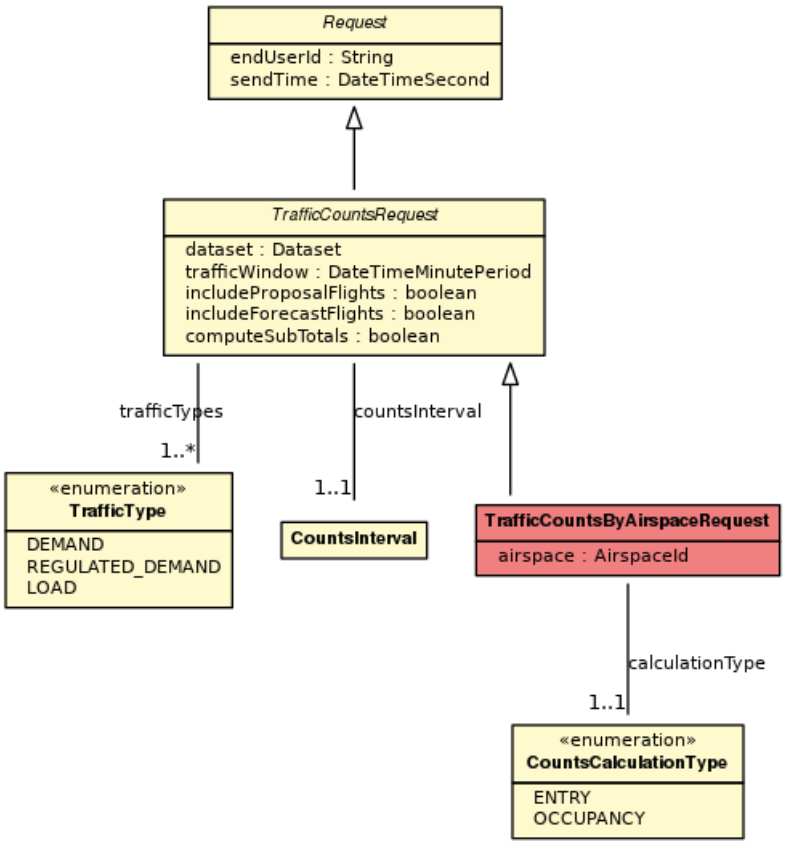

Figure 3.7. `TrafficCountsByAirspaceRequest` Class Diagram

(1) Request to query the NM traffic counts for an airspace.

(2) Inherits from: `TrafficCountsRequest`

(3) Attributes:

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

b) **`CountsCalculationType` `calculationType`** *(Mandatory)*
Indicates what is the calculation type of the count (entry or occupancy).

3.1.5.3. `TrafficCountsByAirspaceReply`

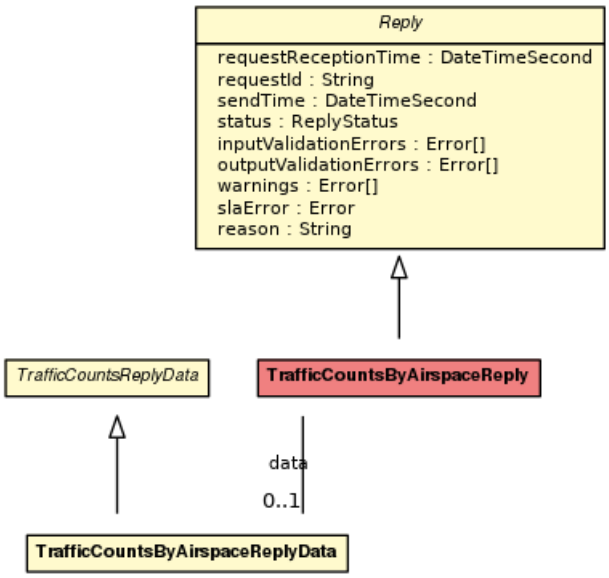

Figure 3.8. `TrafficCountsByAirspaceReply` Class Diagram

(1) Reply returned in response to TrafficCountsByAirspaceRequest.

(2) See TrafficCountsReplyData.

(3) Inherits from: `Reply`

3.1.6. Traffic Count List by Point

3.1.6.1. SOAP

(1) The associated SOAP operation is:

```
TrafficCountsByPointReply queryTrafficCountsByPoint(
    TrafficCountsByPointRequest request
    )
```

3.1.6.2. `TrafficCountsByPointRequest`

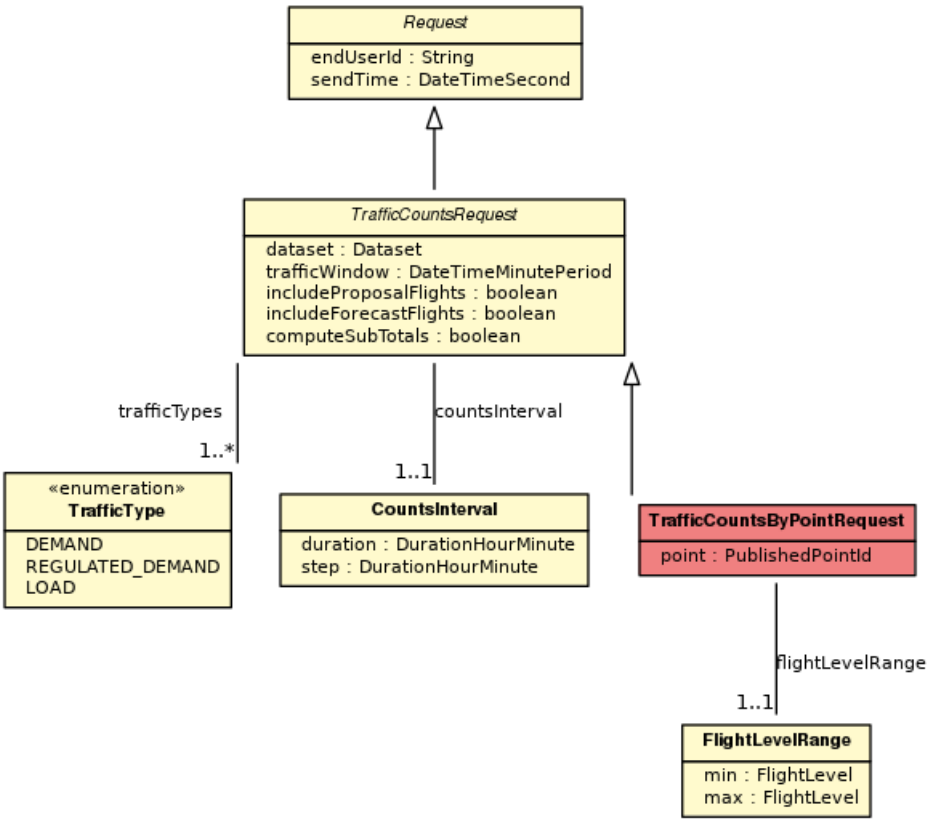

Figure 3.9. `TrafficCountsByPointRequest` Class Diagram

(1) Request to query the NM traffic ENTRY counts for a point.

(2) Inherits from: `TrafficCountsRequest`

(3) 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.

3.1.6.3. `TrafficCountsByPointReply`

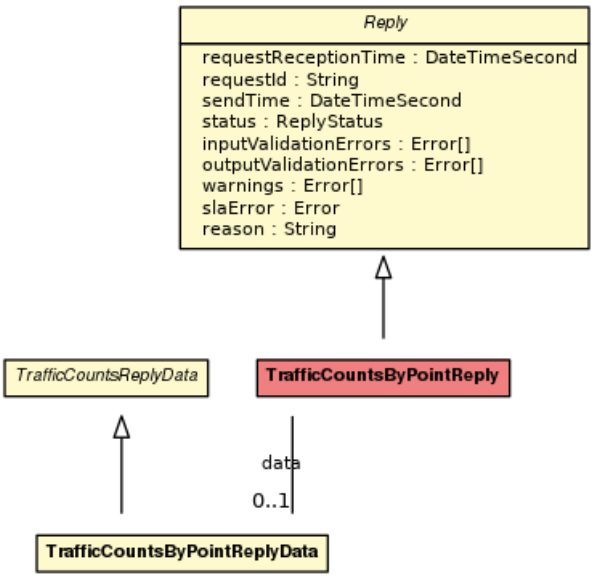

Figure 3.10. `TrafficCountsByPointReply` Class Diagram

(1) Reply returned in response to TrafficCountsByPointRequest.

(2) See TrafficCountsReplyData.

(3) Inherits from: `Reply`

3.1.7. Traffic Count List by Traffic Volume

3.1.7.1. SOAP

(1) The associated SOAP operation is:

```
TrafficCountsByTrafficVolumeReply queryTrafficCountsByTrafficVolume(
    TrafficCountsByTrafficVolumeRequest request
    )
```

3.1.7.2. `TrafficCountsByTrafficVolumeRequest`

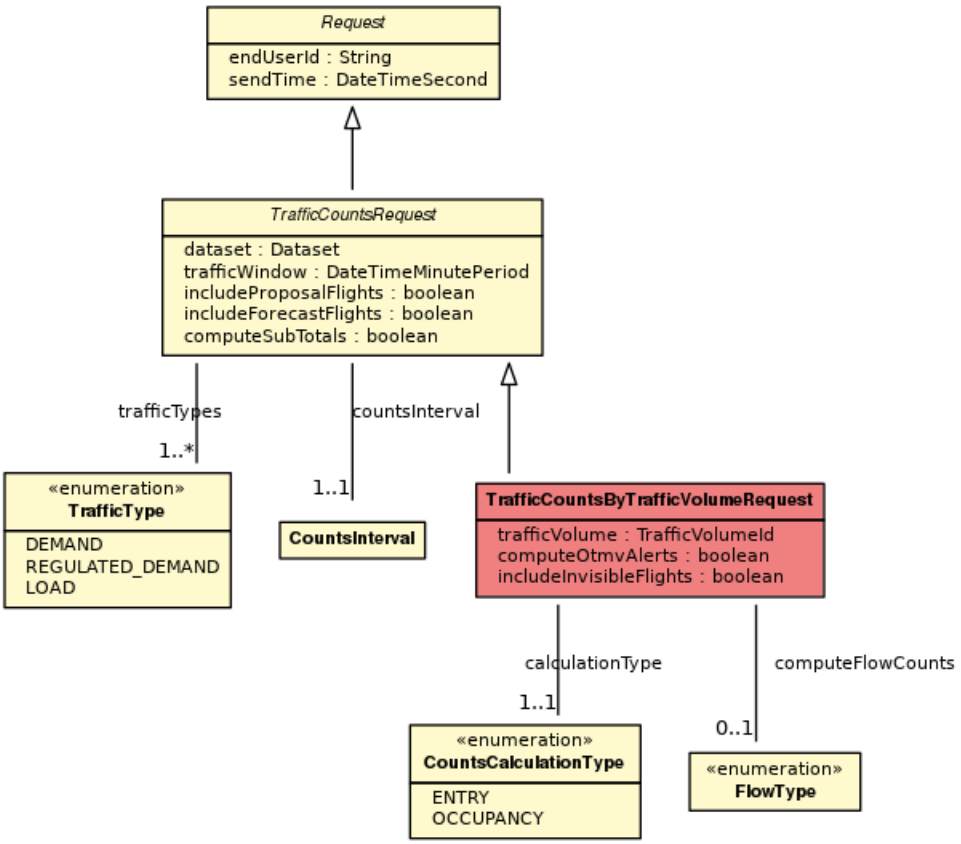

Figure 3.11. `TrafficCountsByTrafficVolumeRequest` Class Diagram

(1) Request to query the NM traffic counts for a traffic volume.

(2) Inherits from: `TrafficCountsRequest`

(3) Attributes:

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

b) **`CountsCalculationType` `calculationType`** *(Mandatory)*
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.
Constraint: See `INCONSISTENT_COUNTS_TYPE_AND_COMPUTE_OTMV_ALERTS`

c) **`boolean` `computeOtmvAlerts`** *(Optional)*
Indicates if OTMV alerts need to be computed (e.g., is the flight in an OTMV peak and during what count periods; see OtmvAlert) or not.
Default is `false`.

Constraints:

i) See INCONSISTENT_COUNTS_TYPE_AND_COMPUTE_OTMV_ALERTS

ii) See INCONSISTENT_FLOW_TYPE_AND_COMPUTE_OTMV_ALERTS

d) **FlowType computeFlowCounts** *(Optional)*
Indicates if traffic counts need to be computed by linked or associated or scenario flows. By default (i.e., if `computeFlowCounts` is not specified), traffic counts are not computed by flow.
Note that computing scenario flow counts is subject the special authorization (as it is a heavy field to compute).
Constraint: See INCONSISTENT_FLOW_TYPE_AND_COMPUTE_OTMV_ALERTS

e) **boolean includeInvisibleFlights** *(Optional)*
Indicates whether invisible flights (VFR, OAT, STAY, IFPSTOP) shall be included in the traffic counts.
Defaults to `false`.

(4) Constraints:

a)

Name	INCONSISTENT_COUNTS_TYPE_AND_COMPUTE_OTMV_ALERTS
Attributes	calculationType, computeOtmvAlerts
Description	If attribute `computeOtmvAlerts` is `true` then attribute `calculationType` must be `OCCUPANCY` and attribute `countsInterval.step` must be one minute (`0001`).

b)

Name	INCONSISTENT_FLOW_TYPE_AND_COMPUTE_OTMV_ALERTS
Attributes	computeFlowCounts, computeOtmvAlerts
Description	If attribute `computeFlowCounts` is `SCENARIO`then the attribute `computeOtmvAlerts` must be `false`).

3.1.7.3. `TrafficCountsByTrafficVolumeReply`

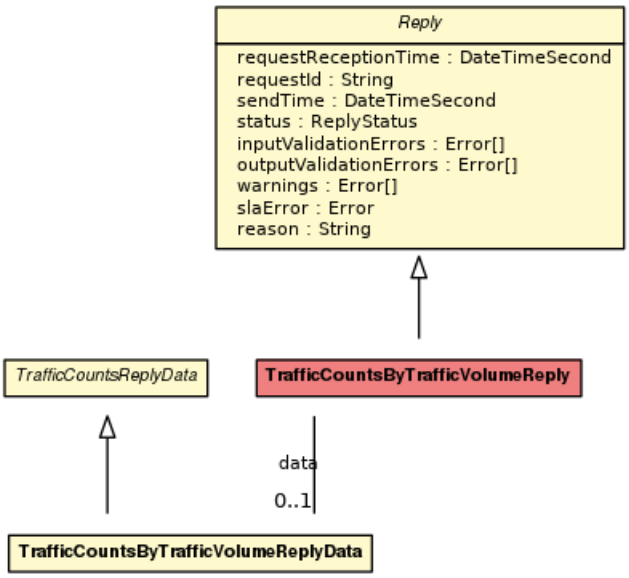

Figure 3.12. `TrafficCountsByTrafficVolumeReply` Class Diagram

(1) Reply returned in response to TrafficCountsByTrafficVolumeRequest.

(2) See TrafficCountsReplyData.

(3) Inherits from: `Reply`

3.2. MeasuresService Port Type

3.2.1. Overview

3.2.1.1. Introduction

(1) This service is intended to provide querying and update capabilities on ATFCM measures. The requests currently available are:

a) RegulationListRequest / RegulationListReply

b) RegulationCreationRequest / RegulationCreationReply

c) RegulationUpdateRequest / RegulationUpdateReply

d) RegulationCancelRequest / RegulationCancelReply

e) RegulationProposalListRequest / RegulationProposalListReply

f) RegulationProposalFilingRequest / RegulationProposalFilingReply

g) RegulationProposalUpdateRequest / RegulationProposalUpdateReply

h) RegulationProposalRevocationRequest / RegulationProposalRevocationReply

i) ReroutingListRequest / ReroutingListReply

j) ReroutingCreationRequest / ReroutingCreationReply

k) ReroutingUpdateRequest / ReroutingUpdateReply

l) ReroutingCancelRequest / ReroutingCancelReply

m) MCDMOnlyListRequest / MCDMOnlyListReply

n) MCDMOnlyCreationRequest / MCDMOnlyCreationReply

o) MCDMOnlyUpdateRequest / MCDMOnlyUpdateReply

p) MCDMOnlyCancelRequest / MCDMOnlyCancelReply

q) MeasureOpLogRetrievalRequest / MeasureOpLogRetrievalReply

r) UpdateFlightsInMeasureRequest / UpdateFlightsInMeasureReply

s) ATFCMSituationRequest / ATFCMSituationReply

t) NetworkImpactAssessmentRetrievalRequest / NetworkImpactAssessmentRetrievalReply

3.2.1.2. Update Pattern

(1) The pattern used for all measure updates is the update of one measure at the time. There can only exist 1 contiguous measure with the same id per day.

(2) When updating a measure, the B2B client can send a delta containing only those fields that need to be updated. Alternatively the B2B client can send the full object with all relevant data fields.

(3) When the user wants to update an existing (normal) measure via a proposal, then the user needs to send the full object with all relevant data fields. On the other hand, when the B2B client wants to update a proposal measure (before NM has started reviewing), then he can send a delta containing only those fields that need to be updated.

(4) Any of the measures and proposal measures may be updated via B2C and/or B2B and/or by NMOC, and by different operators. When an operator updates a measure via B2C, the next B2B retrieve operation will include these changes done via B2C. The pattern used on the backend side to deal with concurrent updates is the following:

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

b) Before updating a measure (via a proposal or directly), the updater must first get the measure and subsequently pass the associated data id when updating it. IMPORTANT:

note that this data id is also related to the dataset in use, i.e. a data id obtained from a dataset cannot be used with another dataset: doing so would result in an error.

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

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

ii) A NM client in a parallel modifies the same regulation (for the same measure id) via B2B or B2C -> latest version in NM systems : dataId I2.

iii) The B2B client end-user modifies some values of the regulation and tries to commit them as a proposal to modify the regulation (includes sending to NM; changes wrt I1).

iv) As the B2B client end-user started from dataId I1 but the measure was also conflictingly modified in parallel CONFLICTING_UPDATE ReplyStatus is returned.

d) From the concurrency perspective, a measure update or proposal to modify a measure is successful if:

i) There was no concurrent update, or

ii) There were identical concurrent updates.

e) IMPORTANT: NM insists that the B2B client only does a measure update in case something has changed for that measure.

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

(6) Unlike with capacity plan updates (See Tactical updates : Update Pattern) there is only 1 valid pattern to use the `dataId`:

a) B2C/NMOC (via phone coordination) can update the concerned measures. The `dataId` (in combination with the CONFLICTING_UPDATE error reply) must be used to detect conflicting parallel updates and report those to the end user so that he can decide what to do.

b) So the detailed steps for a regulation update (via proposal) would be :

i) On the create of a regulation R (via proposal regulation),

ii) B2B.fileRegulationProposal is used without dataId. The reply contains a dataId PR1

iii) Each next update for proposal R or to the MCDM state of R, would use the `dataId` returned by the previous update (PR1,PR2,...). Alternatively when updating a proposal R, the B2B client uses as `dataId`, the dataId from the regulation object as it was retrieved for showing to the client. In that case the dataId corresponds to the regulation version from which the client started to make changes.

iv) If the NM systems detect a conflicting parallel update between the time corresponding to the dataId PR1/PR2.. and the latest NM state wrt proposal regulation PR or MCDM of PR, the reply contains a CONFLICTING_UPDATE error. In that case the client system must warn the operator that a conflicting parallel update has occurred and would show the local data and the NM data to allow the operator to choose (and optionally update any local system as well).
When the B2b client, wants to create a proposal to update/cancel the normal regulation R, then B2B.fileRegulationProposal is used with normalRegulationDataId = NR2 and no dataId. The reply contains proposal regulation dataId PR5.
Alternatively the NM systems detect a conflicting parallel update between the time corresponding to the dataId NR2 and the latest NM state wrt normal regulation R or MCDM of R. In that case the reply contains a CONFLICTING_UPDATE error. In that case the client system must warn the operator that a conflicting parallel update has occurred and would show the local data and the NM data to allow the operator to choose (and optionally update any local system as well).

(7) If measure S is updated, then data Id of measure S (S1) needs to be used.

(8) Note that this pattern is also used by a.o. NM systems:

a) When an normal regulation needs to be updated (proposal for modification), first the NM data is shown and in the screen data the associated dataId NR1 is kept. (the user has seen the NM data corresponding to dataId NR1).
When the user applies his changes, this data Id NR1 is then used in the `RegulationProposalFilingRequest`. In case there were parallel conflicting updates, the user is notified and he needs to redo his update (including first looking at the latest NM data). The main reason behind: multiple operators at different terminals (including NMOC) can do conflicting updates and they need to be notified.

b) When a proposal regulation needs to be updated, first the NM proposal data is shown and in the screen data the associated dataId PR1 is kept. (the user has seen the NM proposal data corresponding to dataId PR1).
When the user applies his changes, this data Id PR1 is then used in the RegulationProposalRevocationRequest, `RegulationProposalUpdateRequest` or `MCDMStateUpdateRequest`. In case there were parallel conflicting updates, the user is notified and he needs to redo his update (including first looking at the latest NM data).

3.2.1.3. Transaction and Errors

(1) See Transactions and Errors.

3.2.1.4. Allowed Times for Retrievals and Updates

3.2.1.4.1. Forecast and Operational Datasets

(1) See Forecast and Operational Datasets.

3.2.1.4.2. Past

(1) Unlike with the tactical updates (See tactical updates : Past) , measure updates can modify the past:

(2) A typical scenario is: a runway got broken and since X minutes no flights were allowed to take-off. In such a case a regulation can be created that starts e.g. 30 minutes in the past and that gives the impacted flights a new CTOT in the future. In NM systems there is the slot change control that allows to control which flights must be changed and which flights cannot be changed or impacted when modifying regulations. By default only flights which have an EOBT of Y minutes or more in the future can be changed (See ATFCM reference manual). If the B2B client needs to change more/less flights, then this needs to be coordinated via phone with NMOC.

(3) So when the B2B client updates a regulation or submits a proposal to update a regulation that has already started and it is not the purpose to change the past, then the B2B client should not modify the past (a.o. archiving reasons). So the B2B clients should not change the start of the regulation but rather split the "current" period inside the initialConstraints in two parts: one for that past (unchanged) and one for the future part (somewhat similar to the remote updates that can not change the past but only change the future: (See tactical updates : Past)).

3.2.1.5. Simulations

(1) See Simulations.

3.2.1.6. Overload protection

(1) There are some technical limitations to the amount of outstanding proposals that can exist at the same time from a given FMP and the amount of proposals that may be sent in a day. The objective of these limits is to protect the NM systems from excessive or faulty demand (e.g. communication errors that triggers a loop of proposals being sent).

a) Classic Regulation Proposals:

   i) A maximum of 15 outstanding (not accepted, nor rejected) ATFM regulation proposals can exist at a time per FMP.

b) Cherry picked Regulation Proposals:

   i) A maximum of 1 CP regulation proposal in MCMState PROPOSED can exist at a time per FMP.

   ii) A maximum of 5 CP regulation proposals can be sent per day and per FMP.

iii) A maximum of 50 CP regulation proposals with regulationState="APPLIED" can exist at the same time (for all users combined).

(2) Note that if the required number exceeds the figures above the request shall be coordinated via telephone.

3.2.2. Regulation List

3.2.2.1. SOAP

(1) The associated SOAP operation is:

```
RegulationListReply queryRegulations(
    RegulationListRequest request
    )
```

3.2.2.2. `RegulationListRequest`

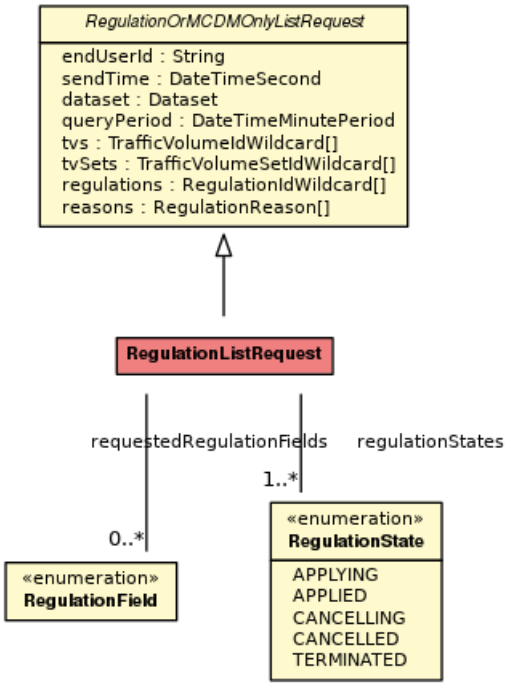

Figure 3.13. `RegulationListRequest` Class Diagram

(1) Request to query regulation.

(2) The reply will only contain "real" regulations (not any proposal regulations).

(3) Inherits from: `RegulationOrMCDMOnlyListRequest`

(4) Attributes:

a) **`Set<RegulationField> requestedRegulationFields`** *(Mandatory)*

The reply returns only the requested regulation fields in this set, and only if the values of these requested fields are available at NM. Note that the regulation id is always returned.
Constraint: Size must be comprised between 0 and 24.

b) **Set<RegulationState> regulationStates** *(Optional)*
Selects the regulations with a state that matches an entry in this set.
By default, regulations are selected regardless to their state.
Constraint: Size must be comprised between 1 and 5.

3.2.2.3. `RegulationListReply`

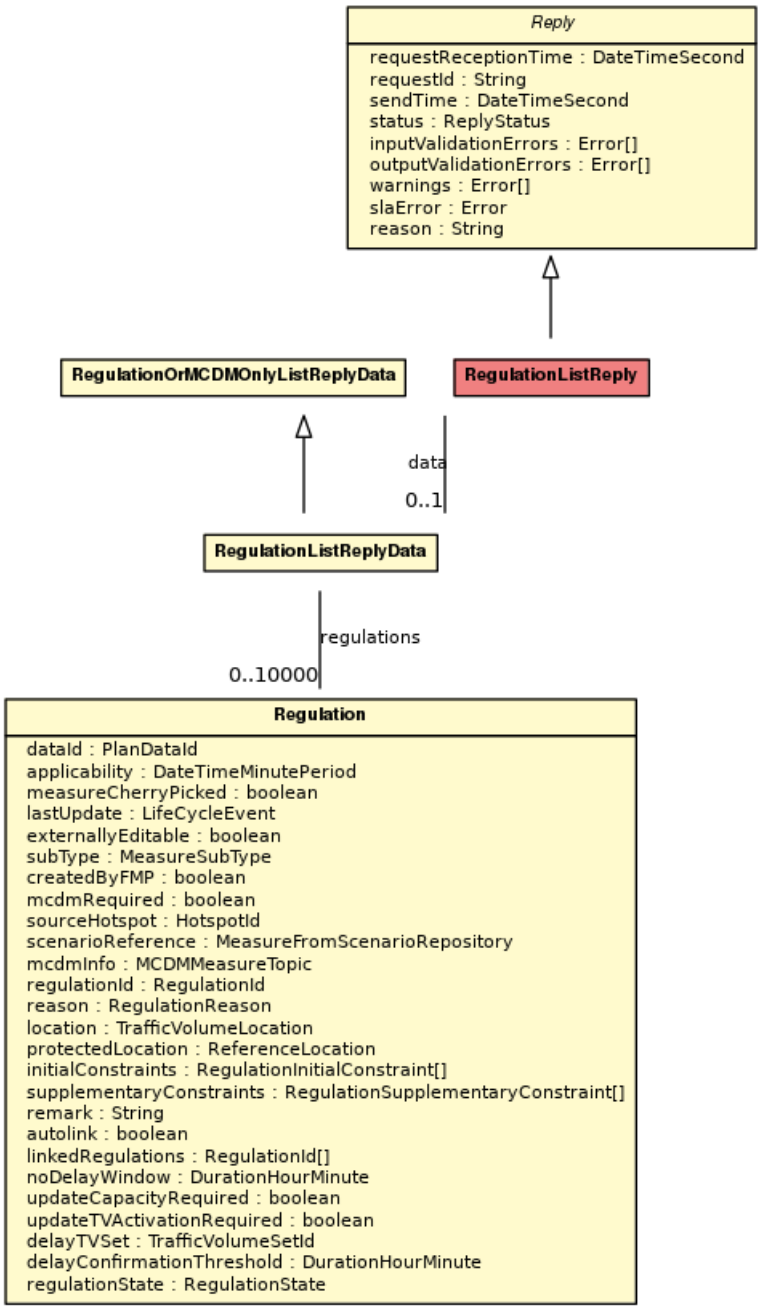

Figure 3.14. `RegulationListReply` Class Diagram

(1) Reply returned in response to RegulationListRequest.

(2) See RegulationOrMCDMOnlyListReplyData.

(3) Inherits from: `Reply`

(4) Attributes:

a) **Set<Regulation> regulations** *(Mandatory)*
Set of regulations that matched the RegulationListRequest criteria.
Can be empty (meaning that no regulation matched the criteria).
Constraint: Size must be comprised between 0 and 10000.

3.2.3. Regulation Creation

3.2.3.1. SOAP

(1) The associated SOAP operation is:

```
RegulationCreationReply createRegulation(
    RegulationCreationRequest request
    )
```

3.2.3.2. `RegulationCreationRequest`

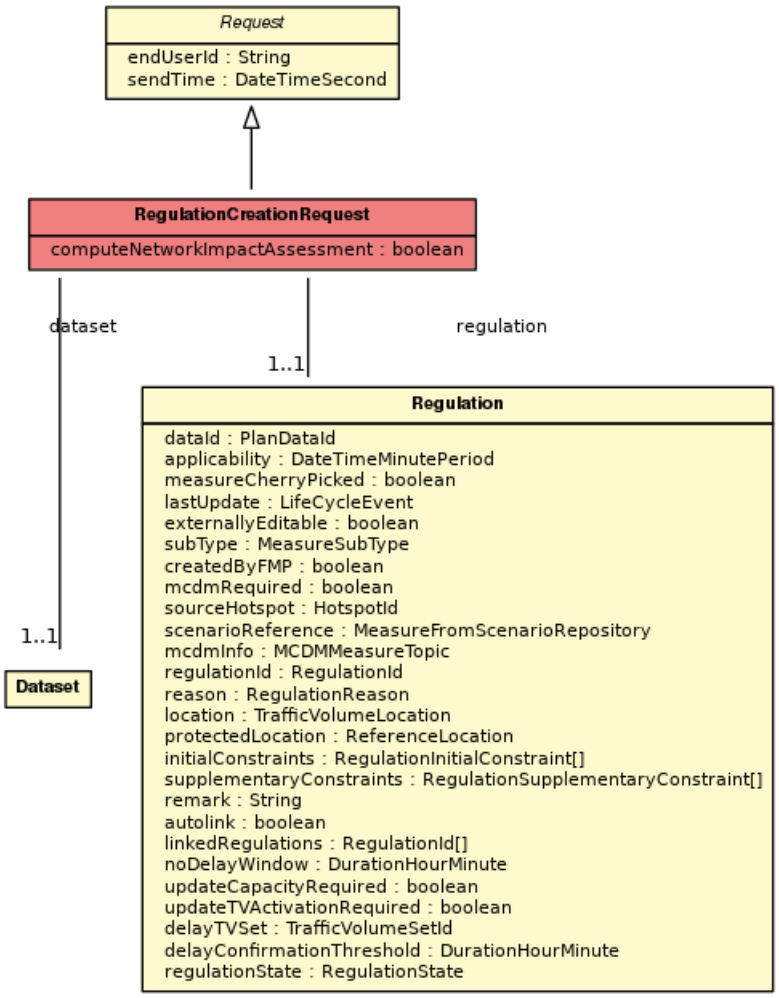

Figure 3.15. `RegulationCreationRequest` Class Diagram

(1) Request to create a regulation.

(2)

Note 1

The `RegulationCreationRequest` is trial related: it is only accessible (authorized) during specific trials or on specific test platforms or in simulations. In STAM trials context only the creation of cherry picked regulations are authorized.

(3)

Note 2

The creation of the regulation is synchronous but the activation is asynchronous. So before flights can be added, the regulation needs to have been activated in the system. A typical client will poll until the `regulationState` has become active, before forcing flights or requesting flightlists onMeasure. In simulation context the creation of the regulation is completely synchronous.

(4) Inherits from: `Request`

(5) Attributes:

a) **`Dataset` `dataset`** *(Mandatory)*
Dataset on which the regulation needs to be created.

b) **`Regulation` `regulation`** *(Mandatory)*
The filed regulation.
Constraint: See `INVALID_MEASURE_APPLICABILITY_PERIOD`

c) **`boolean` `computeNetworkImpactAssessment`** *(Optional)*
In simulation dataset, indicates if the network impact assesment needs to be computed (See `NetworkImpactAssessmentRetrievalRequest`)

(6) Constraint:

a)

Name	`INVALID_MEASURE_APPLICABILITY_PERIOD`
Attribute	`regulation`
Description	Invalid `regulation`(measure) applicability period should overlap Dataset period

3.2.3.3. `RegulationCreationReply`

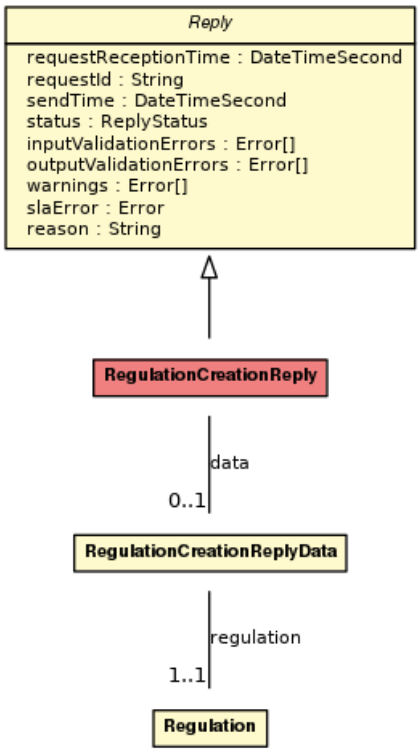

Figure 3.16. `RegulationCreationReply` Class Diagram

(1) Reply returned in response to RegulationCreationRequest.

(2) Inherits from: `Reply`

(3) Attributes:

a) **`Regulation` `regulation`** *(Mandatory)*
The regulation with fields set by the NM system.

3.2.4. Regulation Update

3.2.4.1. SOAP

(1) The associated SOAP operation is:

```
RegulationUpdateReply updateRegulation(
    RegulationUpdateRequest request
    )
```

3.2.4.2. `RegulationUpdateRequest`

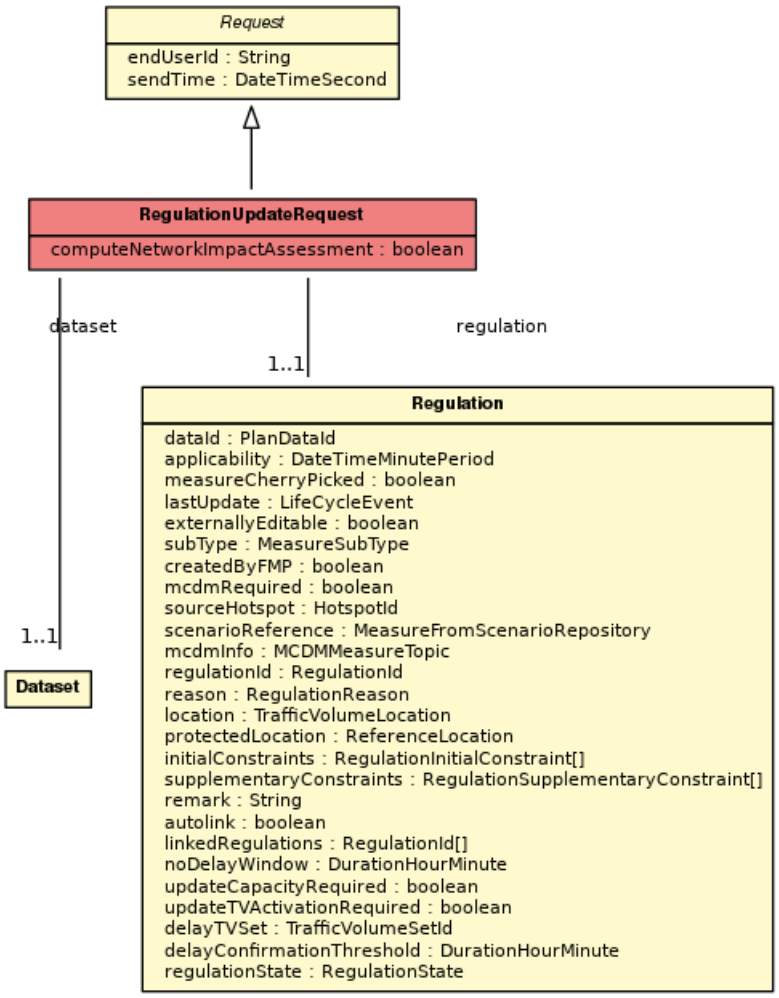

Figure 3.17. `RegulationUpdateRequest` Class Diagram

(1) Request to update (modify) an existing regulation. As a result the regulation will be modified and all concerned flights will be updated accordingly.

(2)

Note 1

The `RegulationUpdateRequest` is trial related: it is only accessible (authorized) during specific trials or on specific test platforms and only for externally editable regulations. In STAM trials context only the update of externally editable cherry picked regulations are authorized. In simulation context all regulations are considered externally editable.

(3)

Note 2

The activation of the updated regulation is asynchronous (See regulationCreationRequest).

(4) Inherits from: `Request`

(5) Attributes:

a) **`Dataset` `dataset`** *(Mandatory)*
Dataset on which the regulation needs to be updated.

b) **`Regulation` `regulation`** *(Mandatory)*
The regulation fields to be updated.
Note that only those fields that need to be updated, need to be non null. The other fields simply remain unchanged.
Constraint: See `INVALID_MEASURE_APPLICABILITY_PERIOD`

c) **`boolean` `computeNetworkImpactAssessment`** *(Optional)*
In simulation dataset, indicates if the network impact assesment needs to be computed (See `NetworkImpactAssessmentRetrievalRequest`)

(6) Constraint:

a)

Name	`INVALID_MEASURE_APPLICABILITY_PERIOD`
Attribute	`regulation`
Description	Invalid `regulation`(measure) applicability period should overlap Dataset period

3.2.4.3. `RegulationUpdateReply`

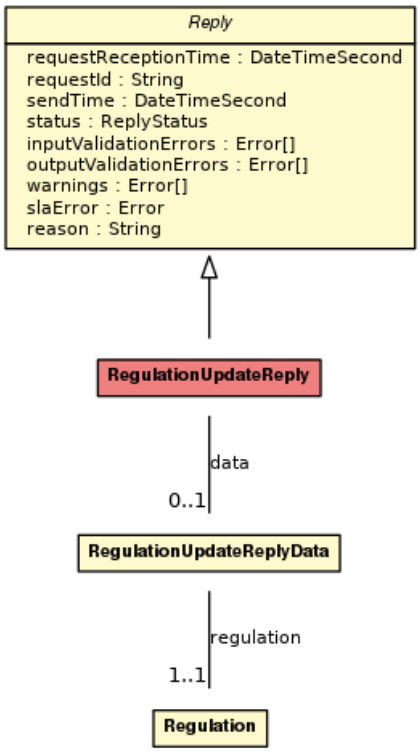

Figure 3.18. `RegulationUpdateReply` Class Diagram

(1) Reply returned in response to RegulationUpdateRequest.

(2) Inherits from: `Reply`

(3) Attributes:

a) **`Regulation regulation`** *(Mandatory)*
The updated regulation with all fields set by the NM system.

3.2.5. Regulation Cancellation

3.2.5.1. SOAP

(1) The associated SOAP operation is:

```
RegulationCancelReply cancelRegulation(
    RegulationCancelRequest request
    )
```

3.2.5.2. `RegulationCancelRequest`

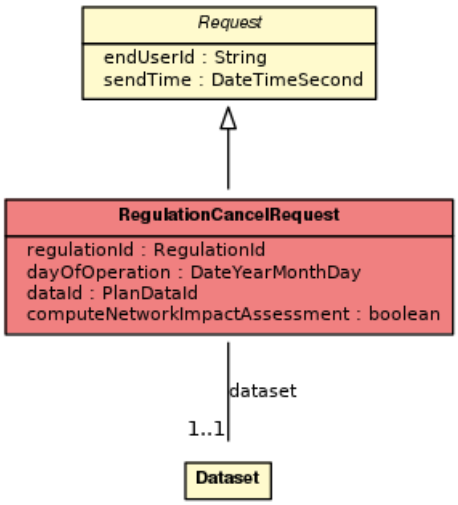

Figure 3.19. `RegulationCancelRequest` Class Diagram

(1) Request to cancel a regulation. As a result the regulation will be cancelled and all concerned flights will be updated and de-regulated if no other regulations impact the flight.

(2) Note that the `RegulationCancelRequest` is trial related: it is only accessible (authorized) during specific trials or on specific test platforms and only for externally editable regulations. In STAM trials context only the cancel of externally editable cherry picked regulations are authorized. In simulation context all regulations are considered externally editable.

(3) Inherits from: `Request`

(4) Attributes:

a) **`Dataset` `dataset`** *(Mandatory)*
Dataset on which the regulation has to be cancelled.

b) **`RegulationId` `regulationId`** *(Mandatory)*
The identifier of the regulation to be cancelled.

c) **`DateYearMonthDay` `dayOfOperation`** *(Mandatory)*
Day for which the regulation has to be cancelled.

d) **`PlanDataId` `dataId`** *(Mandatory)*
Opaque identifier representing the version of the regulation to revoke. The caller shall always keep this value unchanged.

e) **`boolean computeNetworkImpactAssessment`** *(Optional)*
In simulation dataset, indicates if the network impact assesment needs to be computed (See `NetworkImpactAssessmentRetrievalRequest`)

3.2.5.3. `RegulationCancelReply`

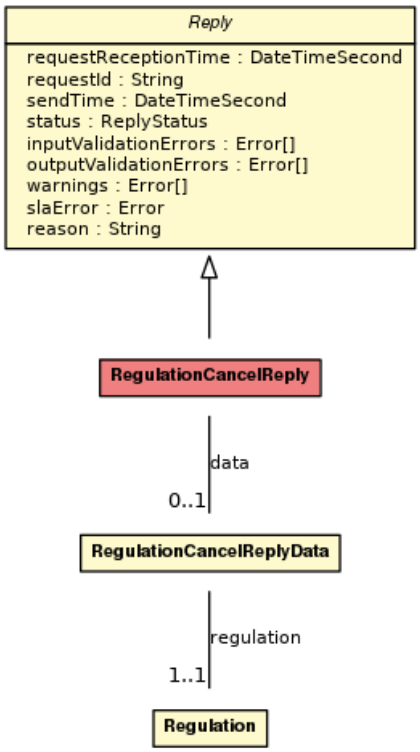

Figure 3.20. `RegulationCancelReply` Class Diagram

(1) Reply returned in response to RegulationCancelRequest.

(2) Inherits from: `Reply`

(3) Attributes:

a) **`Regulation regulation`** *(Mandatory)*
The regulation that has been cancelled.

3.2.6. Regulation Proposal List

3.2.6.1. SOAP

(1) The associated SOAP operation is:

```
RegulationProposalListReply queryRegulationProposals(
    RegulationProposalListRequest request
    )
```

3.2.6.2. `RegulationProposalListRequest`

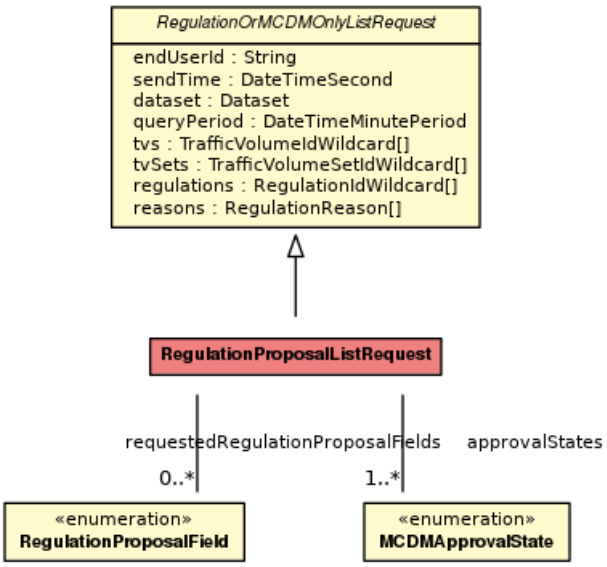

Figure 3.21. `RegulationProposalListRequest` Class Diagram

(1) Request to query regulation proposals.

(2) Regulation proposals are proposals to NM for creation/modification/cancellation of a regulation.

(3) Regulation proposals do not impact any flights directly.

(4) There are 2 types of regulation proposals : Those that support proposal flights and those that do not.

(5) Regulation proposals without proposal flights are basically filled in template for NM to accept/reject.

(6) Regulation proposals with proposal flights are cherry picked regulations (with initially no flights forced). When the users delays some flights, the real flight do not get delayed. Instead proposal flights are created that are reflected in the (with `includeProposalFlights`) flightlist and counts.

(7) NM can accept or reject a regulation proposal. If a proposal is rejected (MCDMApprovalState REJECTED and MCDMState ABANDONED), the user can refile a new proposal. When NM accepts a regulation creation proposal (optionally modifying some attributes:a.o. regulationId), then a real regulation is created (retrievable via the RegulationListRequest) and the regulation proposal has its MCDMApprovalState set to ACCEPTED and MCDMState set to IMPLEMENTED). When NM accepts a regulation modification/cancellation proposal (optionally modifying some attributes), then the real regulation is modified/cancelled.

(8) Note that the `RegulationProposalListRequest` is subject to authorization.

(9) Inherits from: `RegulationOrMCDMOnlyListRequest`

(10) Attributes:

a) **Set<RegulationProposalField> requestedRegulationProposalFields** *(Mandatory)*
The reply returns only the requested regulation proposal fields in this set, and only if the values of these requested fields are available at NM. Note that the regulation proposal id is always returned.
Constraint: Size must be comprised between 0 and 27.

b) **Set<MCDMApprovalState> approvalStates** *(Optional)*
Selects the regulation proposals with a NMOC approval state that matches an entry in this set.
By default, regulation proposals are selected regardless to their state.
Constraint: Size must be comprised between 1 and 4.

3.2.6.3. `RegulationProposalListReply`

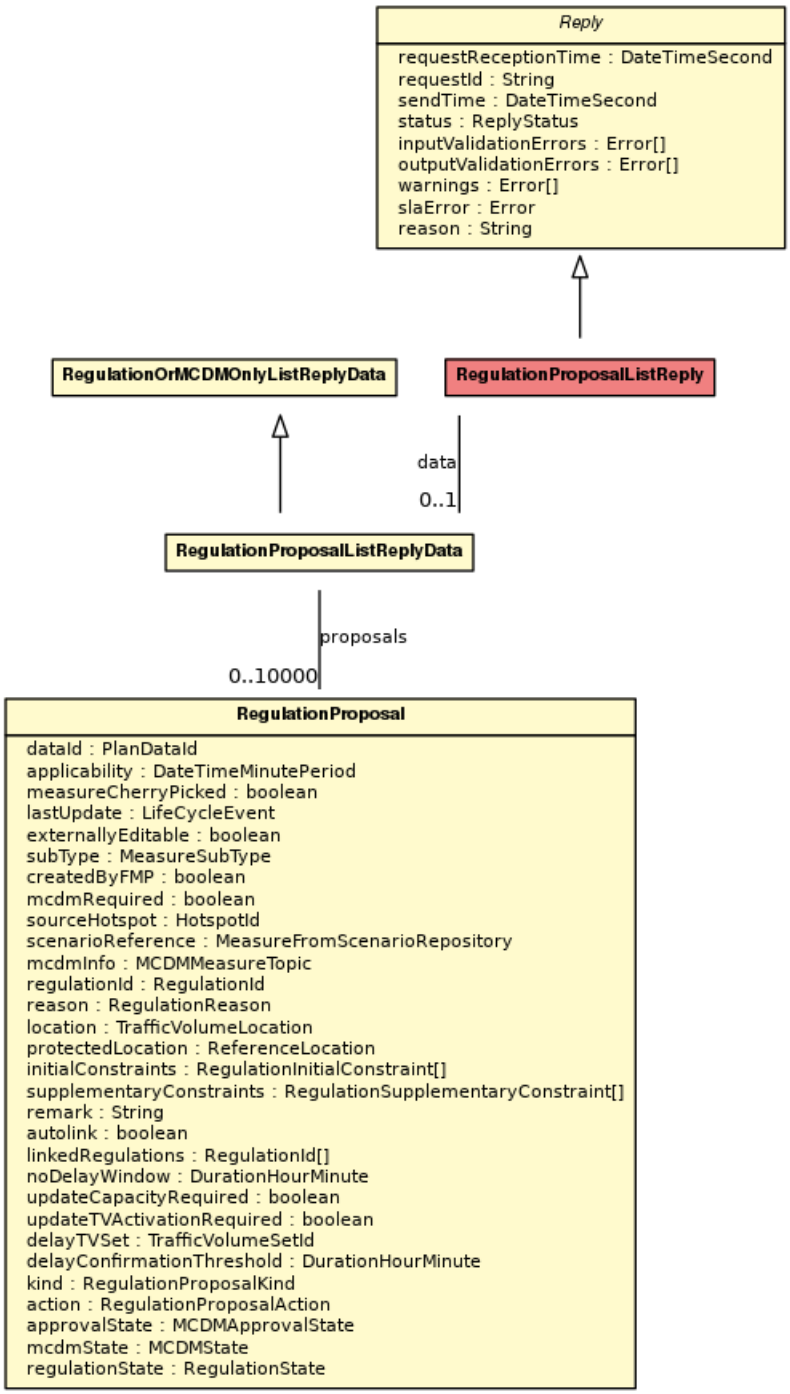

Figure 3.22. `RegulationProposalListReply` Class Diagram

(1) Reply returned in response to RegulationProposalListRequest.

(2) See RegulationOrMCDMOnlyListReplyData.

(3) Inherits from: `Reply`

(4) Attributes:

a) **`Set<RegulationProposal> proposals`** *(Mandatory)*
Set of regulation proposals that matched the RegulationProposalListRequest criteria.
Can be empty (meaning that no regulation proposal matched the criteria).
Constraint: Size must be comprised between 0 and 10000.

3.2.7. Regulation Proposal Filing

3.2.7.1. SOAP

(1) The associated SOAP operation is:

```
RegulationProposalFilingReply fileRegulationProposal(
    RegulationProposalFilingRequest request
    )
```

3.2.7.2. `RegulationProposalFilingRequest`

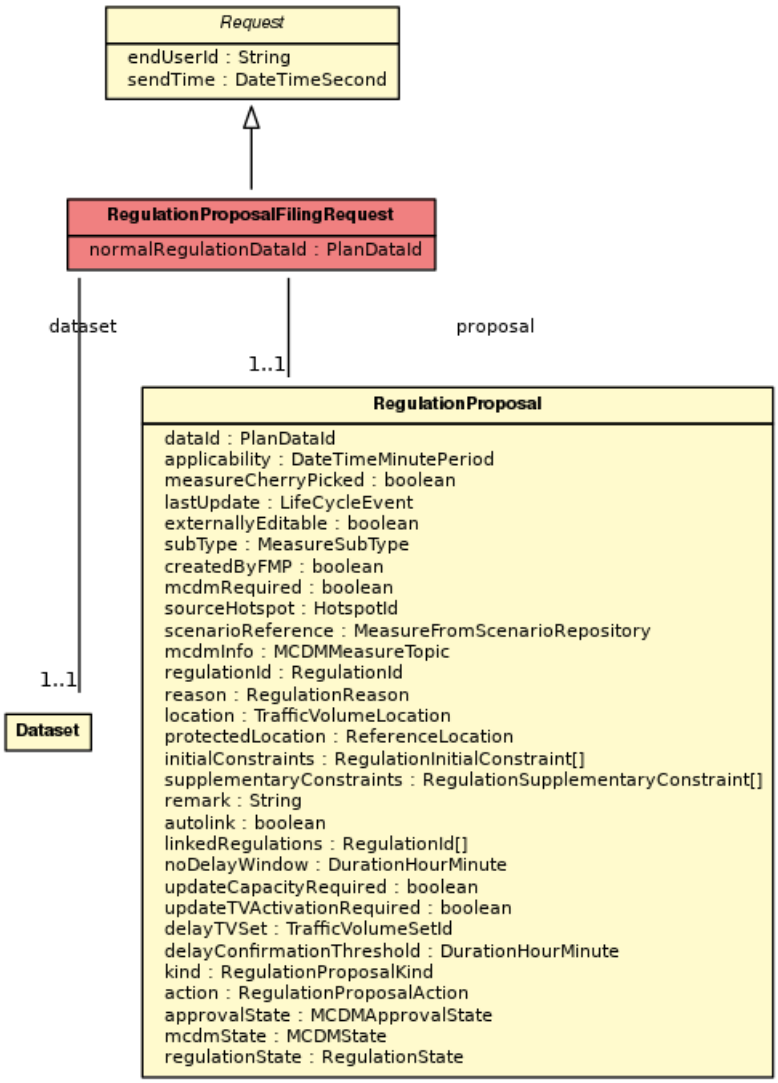

Figure 3.23. `RegulationProposalFilingRequest` Class Diagram

(1) Request to file a proposal to create a regulation or to file a proposal to update a regulation (non-proposal regulation from a `RegulationListRequest`) or to file a proposal to cancel a regulation (non-proposal regulation from a `RegulationListRequest`)

(2) When NMOC accepts the proposal, the end result corresponds to a:

(3) a) `RegulationCreationRequest`: when ACTION=CREATION. So concerned flights get a CTOT or a CTOT update (in case they already had a CTOT).

b) `RegulationUpdateRequest`: when ACTION=UPDATE. So concerned flights get a CTOT (in case the regulation period has been enlarged) or a CTOT update.

c) `RegulationCancelRequest` : when ACTION=CANCELLATION. So all concerned flights will be updated and de-regulated: no more CTOT (if there are no other regulations impacting the flight)

(4) See also RegulationProposalListRequest

(5)

Note 1

The RegulationProposalFilingRequest is subject to authorization.

(6)

Note 2

The creation of the cherry picked regulations is synchronous but the activation is asynchronous. So before flights can be added, the regulation needs to have been activated in the system. A typical client will poll until the regulationState has become active, before adding flights or requesting flightlists on Measure.

(7)

Note 3

If there is already a proposal regulation that has not been accepted, nor rejected by NMOC, then `RegulationProposalFilingRequest` is not to be used. Instead `RegulationProposalUpdateRequest` or `RegulationProposalRevocationRequest` should be used.

(8)

Note 4

There are limitation on the number of proposal regulations that can be requested per day: See Overload protections.

(9)

Note 5

Once a proposal regulation has been accepted by NMOC, it becomes visible in the `RegulationListReply`. Before it is only visible in the `ProposalListRequest`. When a proposal is filed to update a regulation, then this regulation remains unchanged and visible in the `RegulationListRequest`. The proposal itself becomes visible via the `ProposalListRequest` (or the previously accepted or rejected proposal is replaced with the new proposal for the `ProposalListRequest`).

(10)

Note 6

a) `ACTION=CREATION` is only allowed if no (non-proposal) regulation exists yet (via regulationListReply) with that regulationId.

b) `ACTION=UPDATE` is only allowed if a (non-proposal) regulation already exists (via regulationListReply) with that regulationId.

c) `ACTION=UPDATE` is not supported for CherryPick regulations via B2B (via B2B only adding/removing flights is allowed for accepted CherryPick regulations). Modyfing accepted cherry pick regulation data (e.g. remark) needs to be coordinated via phone.

d) `ACTION=CANCELLATION` is supported for CherryPicked regulations but only as a `RegulationProposalWithoutProposalFlights` (meaning there is no possibility to see the what-if proposal flights before submitting to NM for approval)

(11) **Note 7**

When creating a proposal regulation, the reason hotspot can be passed inside the `RegulationProposal` to indicate the original hotspot that will be solved via the proposal regulation. This does not imply that this hotspot needs to exist in NM systems as a hotspot object, nor that the hotspot will be created automatically by NM systems, nor that the B2B client should first create the hotspot nor that the B2B client needs to keep the hotspot field updated at all times. The hotspot field is only for informational awareness and can guide different actors in their decision process.

(12) Inherits from: `Request`

(13) Attributes:

a) **`Dataset` `dataset`** *(Mandatory)*
Dataset on which the regulation proposal is filed.
See Forecast and Operational Datasets.

b) **`RegulationProposal` `proposal`** *(Mandatory)*
The regulation proposal to be filled.
Note that only those fields that need to be updated, need to be non null. The other fields simply remain unchanged.
Constraints:

i) See `INVALID_ACTION`

ii) See `INVALID_MEASURE_APPLICABILITY_PERIOD`

c) **`PlanDataId` `normalRegulationDataId`** *(Optional)*
In case it concerns a proposal to update/cancel an existing regulation, then `normalRegulationDataId` must contain the `dataId` of that regulation (i.e. the `dataId` returned in the RegulationListRequest for the regulation with the same `proposal.regulationId`.
See Update Pattern.
Constraint: See `INVALID_NORMAL_REGULATION_DATA_ID_VALUE`

(14) Constraints:

a)

Name	`INVALID_ACTION`
Attribute	`proposal`
Description	The value for `proposal.action` should be set to `CREATION` when `proposal.kind` is `RegulationProposalWithProposalFlights`.

b)

Name	`INVALID_NORMAL_REGULATION_DATA_ID_VALUE`

Attribute	normalRegulationDataId
Description	From regulationProposal.action value the normalRegulationDataId field can take different values as follows: i) if the regulation proposal action is set as CREATION then the normalRegulationDataId must be null ii) if the regulation proposal action is set as UPDATE then the normalRegulationDataId cannot be null iii) if the regulation proposal action is set as CANCELLATION then the normalRegulationDataId cannot be null

c)

Name	INVALID_MEASURE_APPLICABILITY_PERIOD
Attribute	proposal
Description	Invalid proposal(measure) applicability period should overlap Dataset period

3.2.7.3. RegulationProposalFilingReply

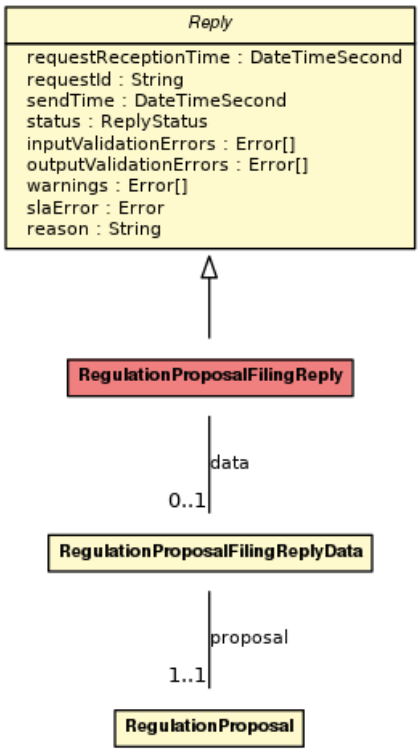

Figure 3.24. RegulationProposalFilingReply Class Diagram

(1) Reply returned in response to RegulationProposalFilingRequest.

(2) Special error conditions:

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

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

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

d) Note that there are also the INVALID_INPUT replyStatus errors, that cover the errors detected by the backend (See Transaction and Errors). For example : action=CREATE for a regulation proposal when there is already a normal (non-proposal) regulation with that regulationId.

(4) Inherits from: `Reply`

(5) Attributes:

a) **`RegulationProposal proposal`** *(Mandatory)*
The `regulation proposal` with fields set by the NM system.
For `RegulationProposalWithoutProposalFlights`, the resulting MCDM state will be set to PROPOSED.
For `RegulationProposalWithProposalFlights`, the resulting MCDM state will be set to DRAFT.

3.2.8. Updating a filed Regulation Proposal

3.2.8.1. SOAP

(1) The associated SOAP operation is:

```
RegulationProposalUpdateReply updateRegulationProposal(
    RegulationProposalUpdateRequest request
    )
```

3.2.8.2. `RegulationProposalUpdateRequest`

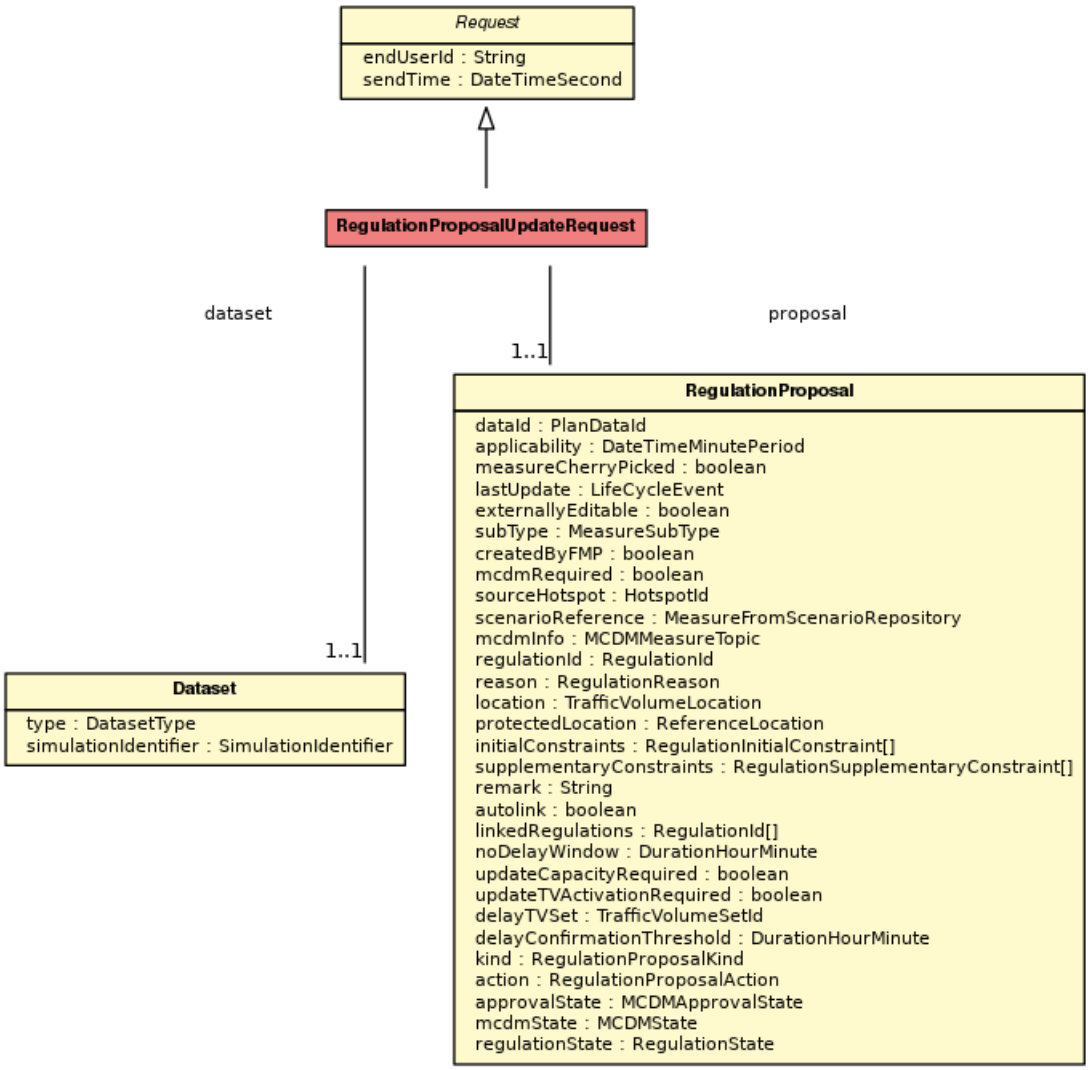

Figure 3.25. `RegulationProposalUpdateRequest` Class Diagram

(1) Request to update a filed regulation proposal before it has been accepted, nor rejected by NMOC.

(2) The B2B client has filed a regulation proposal but he changed his mind (or the solution was not sufficient enough after evaluating the proposal counts and flightlists) and the B2B client wants to make some changes before NMOC starts reviewing the proposal regulation.

(3)

Note 1

The RegulationProposalUpdateRequest is subject to authorization.

(4)

Note 2

The creation of the cherry picked regulations is synchronous but the activation is asynchronous. So before flights can be added, the regulation needs to have been activated in

the system. A typical client will poll until the regulationState has become active after a RegulationProposalUpdateRequest, before adding flights or requesting flightlists on Measure.

(5) **Note 3**

If there is no proposal regulation or the proposal regulation has already been acknowledged, accepted or rejected by NMOC, then `RegulaRegulationProposalUpdateRequest` is not to be used (so it is only allowed if `MCDMState` is DRAFT or PROPOSED). Instead `RegulationProposalFilingRequest` should be used. For `RegulationProposal-WithProposalFlights`, `RegulationProposalUpdateRequest` is only allowed if the `regulationState` = APPLIED.

(6) **Note 4**

In `RegulationProposalUpdateRequest`, `RegulationProposalKind` is immutable (w.r.t. `RegulationProposalKind` from the `RegulationProposalFilingRequest`).

(7) Inherits from: `Request`

(8) Attributes:

a) **`Dataset` `dataset`** *(Mandatory)*
Dataset on which the regulation proposal is updated.
See Forecast and Operational Datasets.

b) **`RegulationProposal` `proposal`** *(Mandatory)*
The regulation proposal fields to be updated.
Constraints:

i) See `INVALID_ACTION`

ii) See `INVALID_MEASURE_APPLICABILITY_PERIOD`

(9) Constraints:

a)

Name	`INVALID_ACTION`
Attribute	`proposal`
Description	The value for `proposal.action` should be set to `CREATION` when `proposal.kind` is `RegulationProposalWithProposalFlights`.

b)

Name	`INVALID_MEASURE_APPLICABILITY_PERIOD`
Attribute	`proposal`
Description	Invalid `proposal`(measure) applicability period should overlap Dataset period

3.2.8.3. `RegulationProposalUpdateReply`

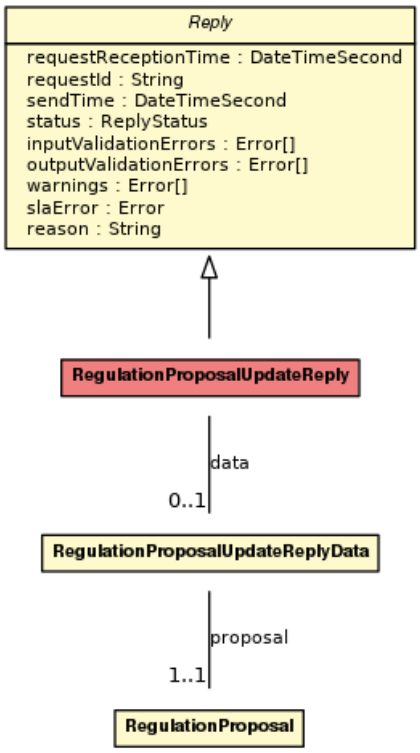

Figure 3.26. `RegulationProposalUpdateReply` Class Diagram

(1) Reply returned in response to RegulationProposalUpdateRequest.

(2) Special error conditions:

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

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

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

d) Note that there are also the INVALID_INPUT replyStatus errors, that cover the errors detected by the backend (See Transaction and Errors). For example : trying to change the regulation from cherry picked to non cherry picked or the other way around.

(4) Inherits from: `Reply`

(5) Attributes:

a) **`RegulationProposal`** **`proposal`** *(Mandatory)*

The updated regulation proposal.
For `RegulationProposalWithoutProposalFlights`, the resulting MCDM state will be set to proposed.
For `RegulationProposalWithProposalFlights`, the resulting MCDM state will be set to draft.

3.2.9. Regulation Proposal Revocation

3.2.9.1. SOAP

(1) The associated SOAP operation is:

```
RegulationProposalRevocationReply revokeRegulationProposal(
    RegulationProposalRevocationRequest request
    )
```

3.2.9.2. `RegulationProposalRevocationRequest`

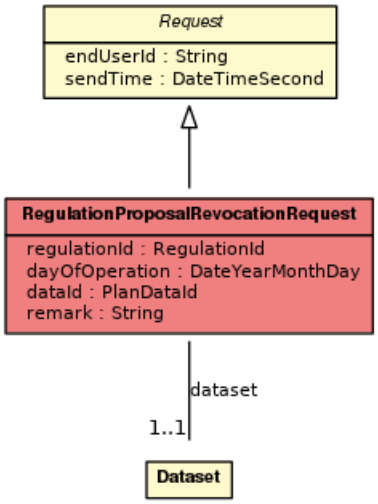

Figure 3.27. `RegulationProposalRevocationRequest` Class Diagram

(1) Request to revoke a regulation proposal.

(2) The B2B client has filed a regulation proposal but he changed his mind and wants to undo: remove the proposal.

(3) For example: if the B2B client has filed a regulation proposal and it has been accepted by NMOC, and afterwards the B2B client adds or changes some flights, then a revoke will simply remove this new proposal but leaves the already accepted regulation and flights unchanged. Note that `RegulationProposalFilingRequest` with ACTION=CANCELLATION is a proposal to cancel a regulation. So when the cancellation is accepted it will update and de-regulate the already accepted flights.

(4)

Note 1

The `RegulationProposalRevocationRequest` is subject to authorization.

(5)

Note 2

Revoking a never applied cherry pick regulation, results in cancelling the proposal regulation (so the regulationId can no longer be re-used in other proposal regulations). The cancellation of a cherry picked regulation is is asynchronous. So it can take some times (typically a few seconds) before all flights are de-regulated and the regulation state becomes cancelled.

(6)

Note 3

If there is no proposal regulation or the proposal regulation has already been acknowledged, accepted or rejected by NMOC, then `RegulaRegulationProposalRevocationRequest` is not to be used (so it is only allowed if `MCDMState` is DRAFT or PROPOSED). Instead `RegulationProposalFilingRequest` should be used. For `RegulationProposalWithProposalFlights`, `RegulationProposalRevocationRequest` is only allowed if the `regulationState` = APPLIED.

(7) Inherits from: `Request`

(8) Attributes:

a) **`Dataset` `dataset`** *(Mandatory)*
Dataset on which the regulation proposal has to be revoked.
See Forecast and Operational Datasets.

b) **`RegulationId` `regulationId`** *(Mandatory)*
The id of the regulation for which this proposal is to be revoked.

c) **`DateYearMonthDay` `dayOfOperation`** *(Mandatory)*
Day for which the proposal is to be revoked.
Constraint: See `INVALID_DAY_OF_OPERATION`

d) **`PlanDataId` `dataId`** *(Mandatory)*
Opaque identifier representing the version of the regulation proposal to revoke. The caller shall always keep this value unchanged.
See Update Pattern.

e) **`string` `remark`** *(Optional)*
Updated remark field of the proposal. The client can indicate the reason for the revocation to NM.

(9) Constraint:

a)

Name	`INVALID_DAY_OF_OPERATION`
Attribute	`dayOfOperation`

Description	Invalid `dayOfOperation` should be within Dataset period -1 day(yesterday including)

3.2.9.3. `RegulationProposalRevocationReply`

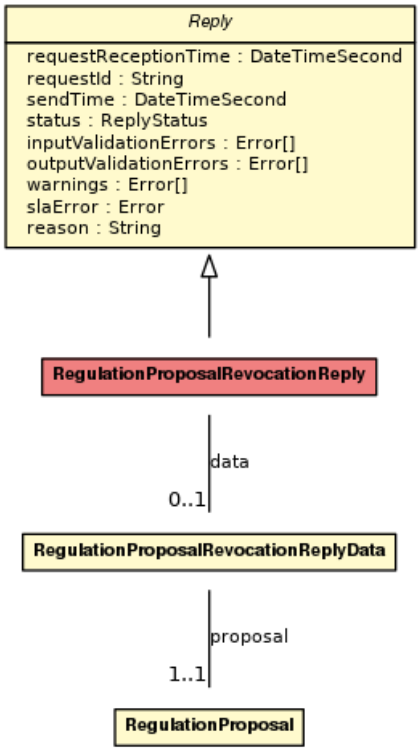

Figure 3.28. `RegulationProposalRevocationReply` Class Diagram

(1) Reply returned in response to RegulationProposalRevocationRequest.

(2) Special error conditions:

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

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

c) OBJECT_NOT_FOUND [Permanent error] If the proposal regulation is not known.

d) Note that there are also the INVALID_INPUT replyStatus errors, that cover the errors detected by the backend (See Transaction and Errors). For example : trying to revoke a proposal regulation that has already been acknowledged (MCDMState = COORDINATED).

(4) Inherits from: `Reply`

(5) Attributes:

a) **`RegulationProposal` `proposal`** *(Mandatory)*
The regulation proposal that has been revoked.
The resulting MCDM state will be set to ABANDONED.

3.2.10. Rerouting List

3.2.10.1. SOAP

(1) The associated SOAP operation is:

```
ReroutingListReply queryReroutings(
    ReroutingListRequest request
    )
```

3.2.10.2. `ReroutingListRequest`

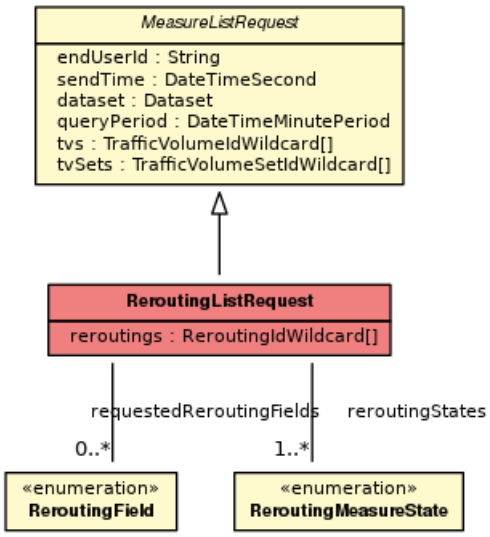

Figure 3.29. `ReroutingListRequest` Class Diagram

(1) Request to query reroutings.

(2) Reroutings are measures to level cap or reroute flights to avoid an airspace/point or to find shorter/cheaper routes. Typically they are used for ATFCM reasons (for example to avoid a zero rate regulation) or for STAM or for Flight Efficiency or to handle forecast expected flows (for example NAT traffic).

(3) Rerouting can either create a proposal flight (containing a proposed route) or they can modify the FTFM/RTFM point profile directly (used in forecast and simulations) or they can generate proposed routes in operational log messages.

(4) Inherits from: `MeasureListRequest`

(5) Attributes:

a) **Set<ReroutingIdWildcard> reroutings** *(Optional)*
The set of rerouting ids or wildcards
if specified, the reply returns only the requested reroutings.
The logical OR operator is meant between the items in the set.
Constraint: Size must be comprised between 1 and 100.

b) **Set<ReroutingField> requestedReroutingFields** *(Mandatory)*
The reply returns only the requested rerouting fields in this set, and only if the values of these requested fields are available at NM. Note that the rerouting id is always returned.
Constraint: Size must be comprised between 0 and 18.

c) **Set<ReroutingMeasureState> reroutingStates** *(Optional)*
Selects the reroutings with a state that matches an entry in this set.
By default, reroutings are selected regardless to their state.
Constraint: Size must be comprised between 1 and 4.

3.2.10.3. `ReroutingListReply`

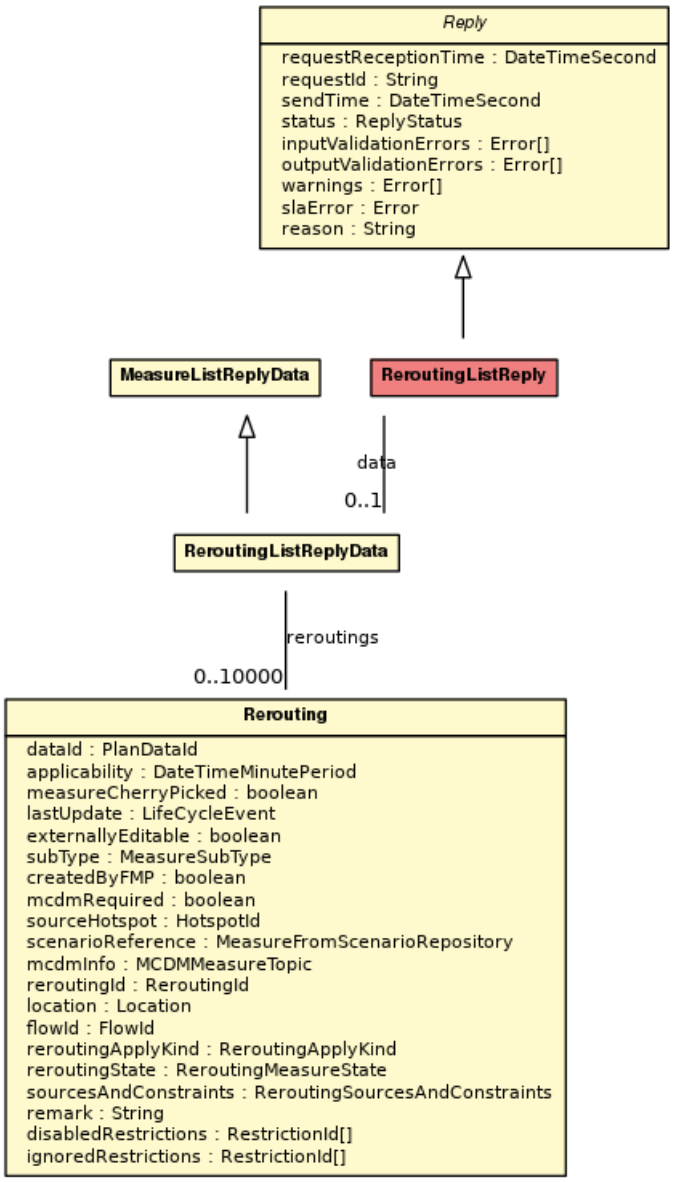

Figure 3.30. `ReroutingListReply` Class Diagram

(1) Reply returned in response to ReroutingListRequest.

(2) Inherits from: `Reply`

(3) Attributes:

a) **`Set<Rerouting> reroutings`** *(Mandatory)*
Set of reroutings that matched the ReroutingListRequest criteria.
Can be empty (meaning that no rerouting matched the criteria).

Constraint: Size must be comprised between 0 and 10000.

3.2.11. Rerouting Creation

3.2.11.1. SOAP

(1) The associated SOAP operation is:

```
ReroutingCreationReply createRerouting(
    ReroutingCreationRequest request
    )
```

3.2.11.2. `ReroutingCreationRequest`

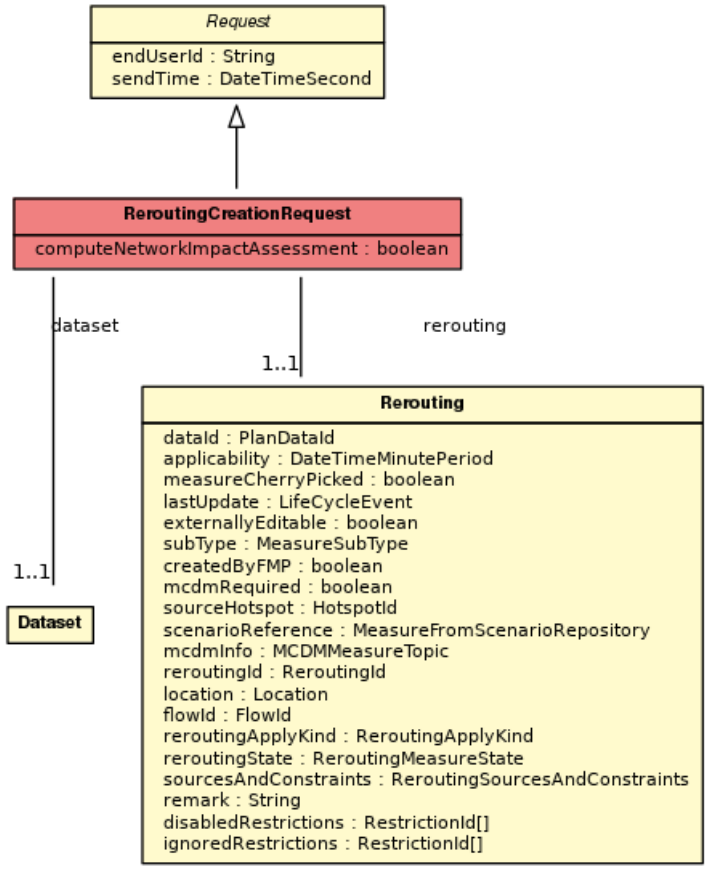

Figure 3.31. `ReroutingCreationRequest` Class Diagram

(1) Request to create a rerouting.

(2)

Note 1

The `ReroutingCreationRequest` is trial related: it is only accessible (authorized) during specific trials or on specific test platforms.

(3) **Note 2**

The creation of the rerouting is asynchronous. So if the service returns, the request is validated but the processing is not finished yet: in general it takes about 1-2 minutes to create and activate the rerouting. Before flights can be added, the rerouting needs to have been created and activated in the system. A typical client will poll until the `rerouting-State` has become active, before adding flights. Note that in simulation context, the creation of the rerouting is synchronous.

(4) Inherits from: `Request`

(5) Attributes:

a) **`Dataset` `dataset`** *(Mandatory)*
Dataset on which the rerouting needs to be created.

b) **`Rerouting` `rerouting`** *(Mandatory)*
The rerouting that is going to be created.
Constraint: See `INVALID_MEASURE_APPLICABILITY_PERIOD`

c) **`boolean` `computeNetworkImpactAssessment`** *(Optional)*
In simulation dataset, indicates if the networkImpactAssesment needs to be computed (See `NetworkImpactAssessmentRetrievalRequest`)

(6) Constraint:

a)

Name	`INVALID_MEASURE_APPLICABILITY_PERIOD`
Attribute	`rerouting`
Description	Invalid `rerouting`(measure) applicability period should overlap Dataset period

3.2.11.3. `ReroutingCreationReply`

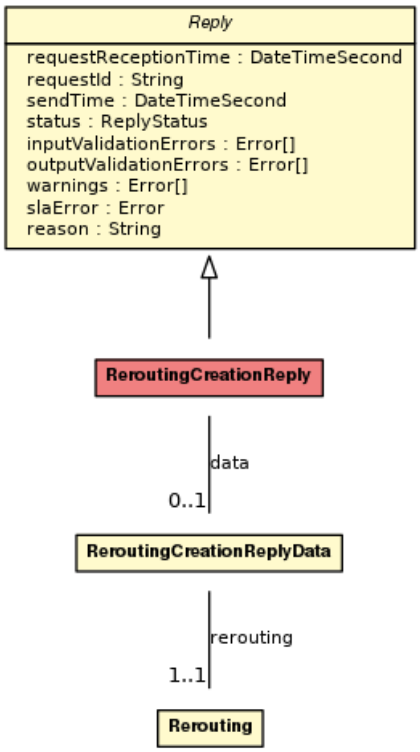

Figure 3.32. `ReroutingCreationReply` Class Diagram

(1) Reply returned in response to ReroutingCreationRequest.

(2) Inherits from: `Reply`

(3) Attributes:

a) **`Rerouting` `rerouting`** *(Mandatory)*
The rerouting with fields set by the NM system.

3.2.12. Rerouting Update

3.2.12.1. SOAP

(1) The associated SOAP operation is:

```
ReroutingUpdateReply updateRerouting(
    ReroutingUpdateRequest request
    )
```

3.2.12.2. `ReroutingUpdateRequest`

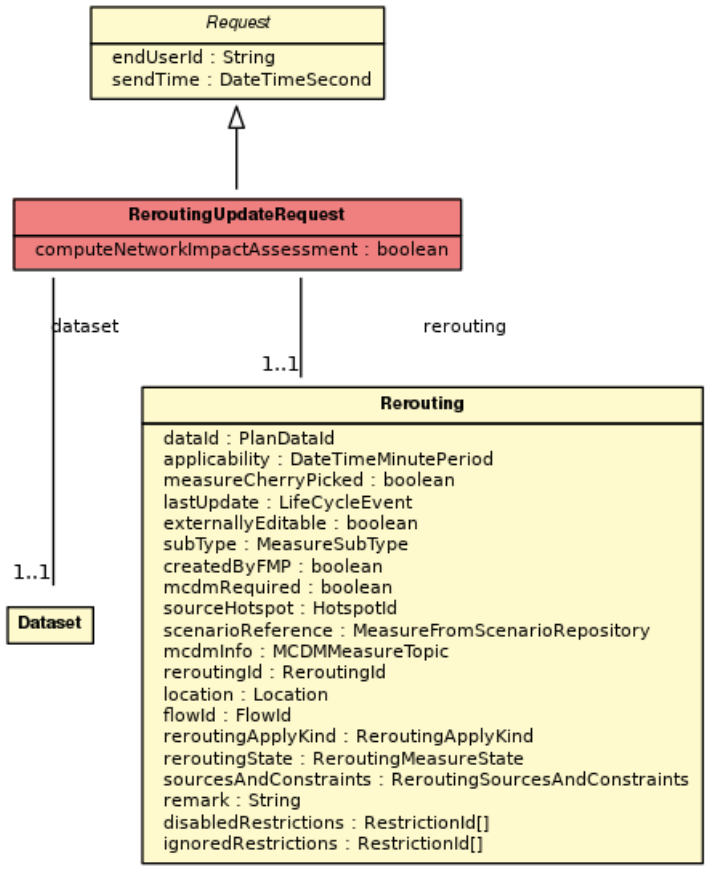

Figure 3.33. `ReroutingUpdateRequest` Class Diagram

(1) Request to update (modify) an existing rerouting.

(2)

Note 1

The `ReroutingUpdateRequest` is trial related: it is only accessible (authorized) during specific trials or on specific test platforms and only for externally editable reroutings.

(3)

Note 2

The update of a rerouting is asynchronous. (See rerouting creation request). A typical client will poll until the `reroutingState` has become active and the `dataId` of the existing rerouting has changed, before adding flights (except when in simulation context).

(4) Inherits from: `Request`

(5) Attributes:

a) **`Dataset` `dataset`** *(Mandatory)*
Dataset on which the rerouting needs to be updated.

b) **`Rerouting rerouting`** *(Mandatory)*
The rerouting fields to be updated.
Note that only those fields that need to be updated, need to be non null. The other fields simply remain unchanged.
Constraint: See `INVALID_MEASURE_APPLICABILITY_PERIOD`

c) **`boolean computeNetworkImpactAssessment`** *(Optional)*
In simulation dataset, indicates if the network impact assesment needs to be computed (See `NetworkImpactAssessmentRetrievalRequest`)

(6) Constraint:

a)

Name	`INVALID_MEASURE_APPLICABILITY_PERIOD`
Attribute	`rerouting`
Description	Invalid `rerouting`(measure) applicability period should overlap Dataset period

3.2.12.3. `ReroutingUpdateReply`

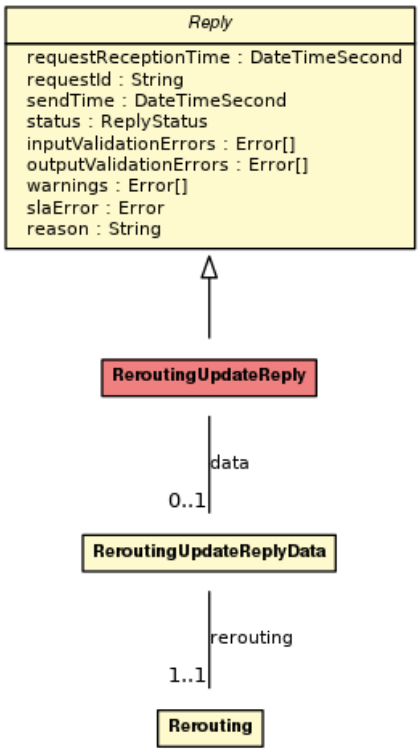

Figure 3.34. `ReroutingUpdateReply` Class Diagram

(1) Reply returned in response to ReroutingUpdateRequest.

(2) Inherits from: `Reply`

(3) Attributes:

a) **`Rerouting` `rerouting`** *(Mandatory)*
The updated rerouting with all fields set by the NM system.

3.2.13. Rerouting Cancellation

3.2.13.1. SOAP

(1) The associated SOAP operation is:

```
ReroutingCancelReply cancelRerouting(
    ReroutingCancelRequest request
    )
```

3.2.13.2. `ReroutingCancelRequest`

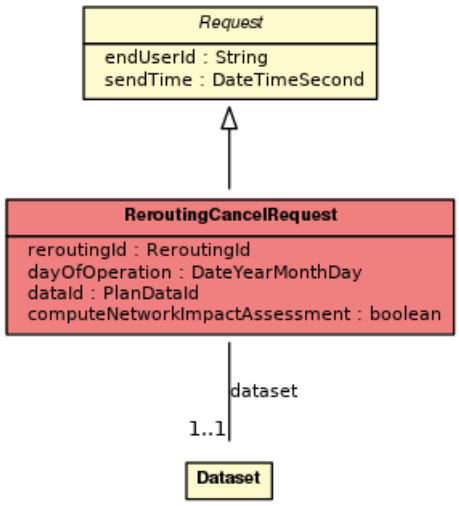

Figure 3.35. `ReroutingCancelRequest` Class Diagram

(1) Request to cancel a rerouting. Undo the effect on the flights.

(2)

Note

The `ReroutingCancelRequest` is trial related: it is only accessible (authorized) during specific trials or on specific test platforms and only for externally editable reroutings. In STAM trials context only the cancel of externally editable cherry picked reroutings are authorized. In simualtion context all rerouting are considered externally editable.

(3) Inherits from: `Request`

(4) Attributes:

a) **Dataset dataset** *(Mandatory)*
Dataset on which the rerouting proposal has to be cancelled.

b) **ReroutingId reroutingId** *(Mandatory)*
The identifier of the rerouting to be cancelled.

c) **DateYearMonthDay dayOfOperation** *(Mandatory)*
Day for which the rerouting has to be cancelled.

d) **PlanDataId dataId** *(Mandatory)*
Opaque identifier representing the version of the rerouting to revoke. The caller shall always keep this value unchanged.

e) **boolean computeNetworkImpactAssessment** *(Optional)*
In simulation dataset, indicates if the network impact assesment needs to be computed (See NetworkImpactAssessmentRetrievalRequest)

3.2.13.3. ReroutingCancelReply

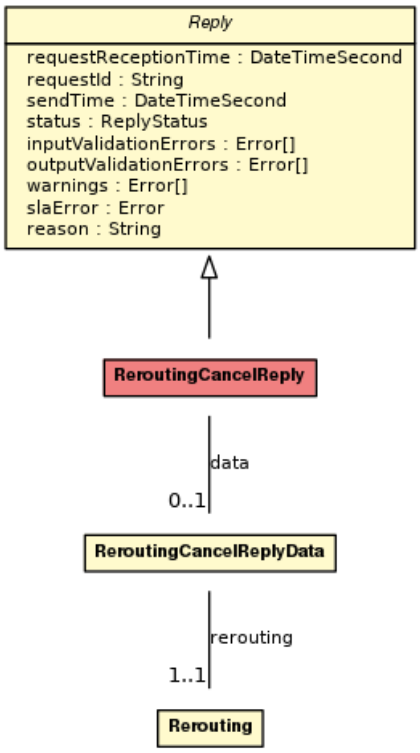

Figure 3.36. ReroutingCancelReply Class Diagram

(1) Reply returned in response to ReroutingCancelRequest.

(2) Inherits from: Reply

(3) Attributes:

a) **`Rerouting` rerouting** *(Mandatory)*
The rerouting that has been cancelled.

3.2.14. MCDMOnly List

3.2.14.1. SOAP

(1) The associated SOAP operation is:

```
MCDMOnlyListReply queryMCDMOnly(
    MCDMOnlyListRequest request
    )
```

3.2.14.2. `MCDMOnlyListRequest`

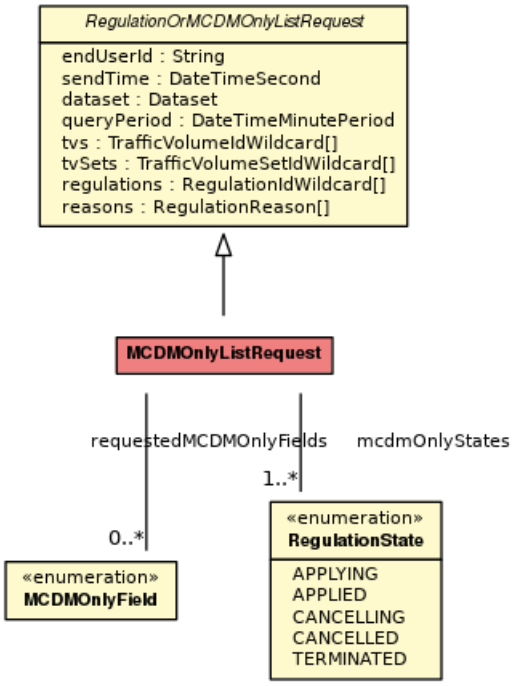

Figure 3.37. `MCDMOnlyListRequest` Class Diagram

(1) Request to query `mcdmOnly` measures. `mcdmOnly` measures are pure text coordination measures. The remark field actually describes what kind of measure this is and what is expected. They allow to associate flights to `mcdmOnly` measures and follow the MCDM process. However there are no proposal flights associated to `mcdmOnly` measures, so no what-if counts and flightlists can be done to evaluate any potential impact.

(2) Inherits from: `RegulationOrMCDMOnlyListRequest`

(3) Attributes:

a) **`Set<MCDMOnlyField> requestedMCDMOnlyFields`** *(Mandatory)*

The reply returns only the requested mcdmOnly fields in this set, and only if the values of these requested fields are available at NM. Note that the mcdm only id is always returned.
Constraint: Size must be comprised between 0 and 24.

b) **Set<RegulationState> mcdmOnlyStates** *(Optional)*
Selects the mcdmOnlys with a state that matches an entry in this set.
By default, mcdmOnlys are selected regardless to their state.
Constraint: Size must be comprised between 1 and 5.

3.2.14.3. `MCDMOnlyListReply`

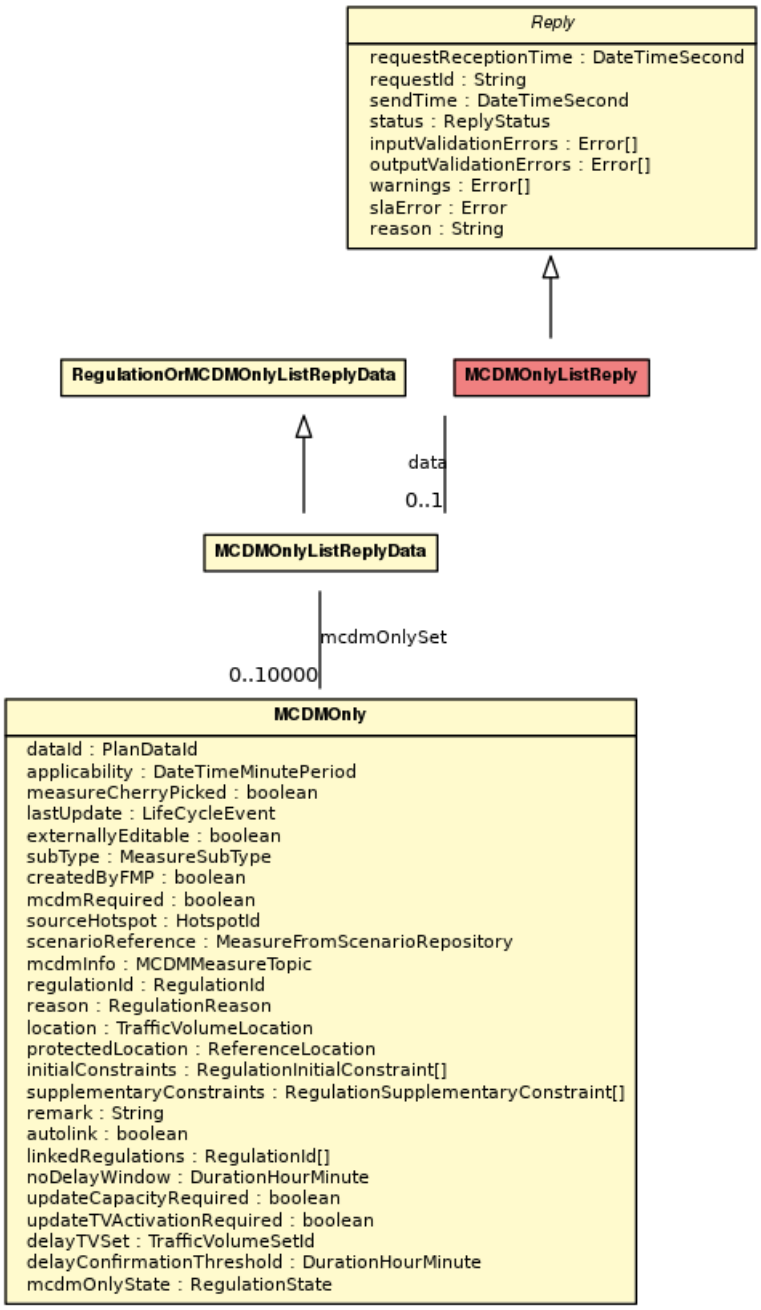

Figure 3.38. `MCDMOnlyListReply` Class Diagram

(1) Reply returned in response to MCDMOnlyListRequest.

(2) See RegulationOrMCDMOnlyListReplyData.

(3) Inherits from: `Reply`

(4) Attributes:

a) **Set<MCDMOnly> mcdmOnlySet** *(Mandatory)*
Set of mcdm only that matched the MCDMOnlyListRequest criteria.
Can be empty (meaning that no mcdm only matched the criteria).
Constraint: Size must be comprised between 0 and 10000.

3.2.15. MCDMOnly Creation

3.2.15.1. SOAP

(1) The associated SOAP operation is:

```
MCDMOnlyCreationReply createMCDMOnly(
    MCDMOnlyCreationRequest request
    )
```

3.2.15.2. `MCDMOnlyCreationRequest`

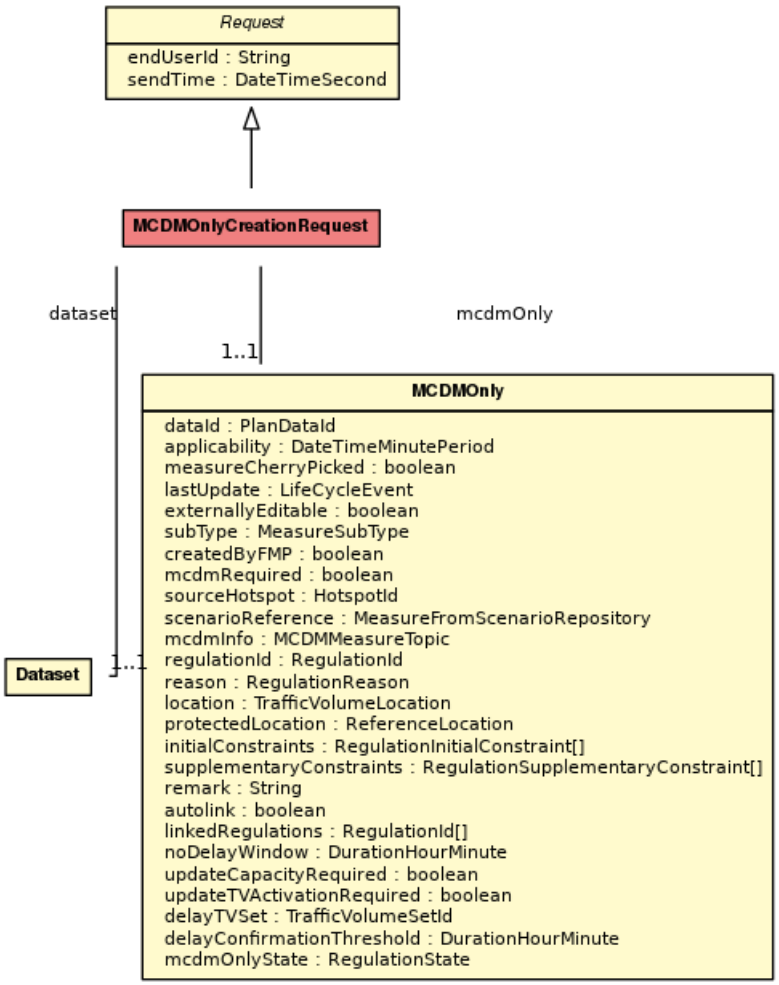

Figure 3.39. `MCDMOnlyCreationRequest` Class Diagram

(1) Request to create a mcdmOnly.

(2) Note that the `MCDMOnlyCreationRequest` is trial related: it is only accessible (authorized) during specific trials or on specific test platforms. In STAM trials context only the creation of cherry picked mcdmOnlys are authorized.

(3) Inherits from: `Request`

(4) Attributes:

a) **`Dataset` `dataset`** *(Mandatory)*
Dataset on which the mcdmOnly needs to be created.

b) **`MCDMOnly` `mcdmOnly`** *(Mandatory)*
The filed mcdmOnly.

Constraint: See INVALID_MEASURE_APPLICABILITY_PERIOD

(5) Constraint:

a)

Name	INVALID_MEASURE_APPLICABILITY_PERIOD
Attribute	mcdmOnly
Description	Invalid mcdmOnly(measure) applicability period should overlap Dataset period

3.2.15.3. MCDMOnlyCreationReply

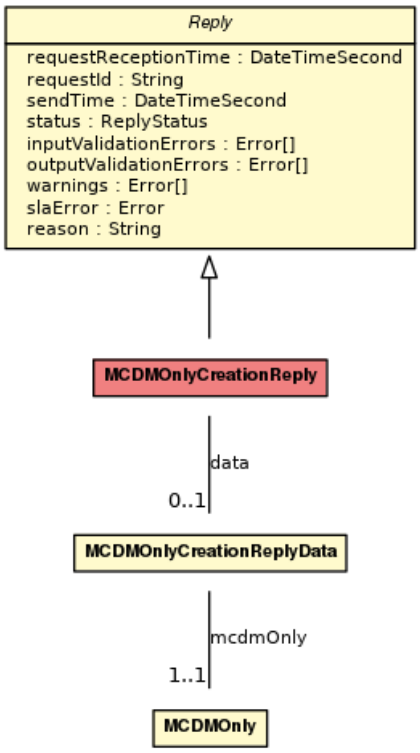

Figure 3.40. MCDMOnlyCreationReply Class Diagram

(1) Reply returned in response to MCDMOnlyCreationRequest.

(2) Inherits from: Reply

(3) Attributes:

a) **MCDMOnly mcdmOnly** *(Mandatory)*
The mcdm only with fields set by the NM system.

3.2.16. MCDMOnly Update

3.2.16.1. SOAP

(1) The associated SOAP operation is:

```
MCDMOnlyUpdateReply updateMCDMOnly(
    MCDMOnlyUpdateRequest request
    )
```

3.2.16.2. `MCDMOnlyUpdateRequest`

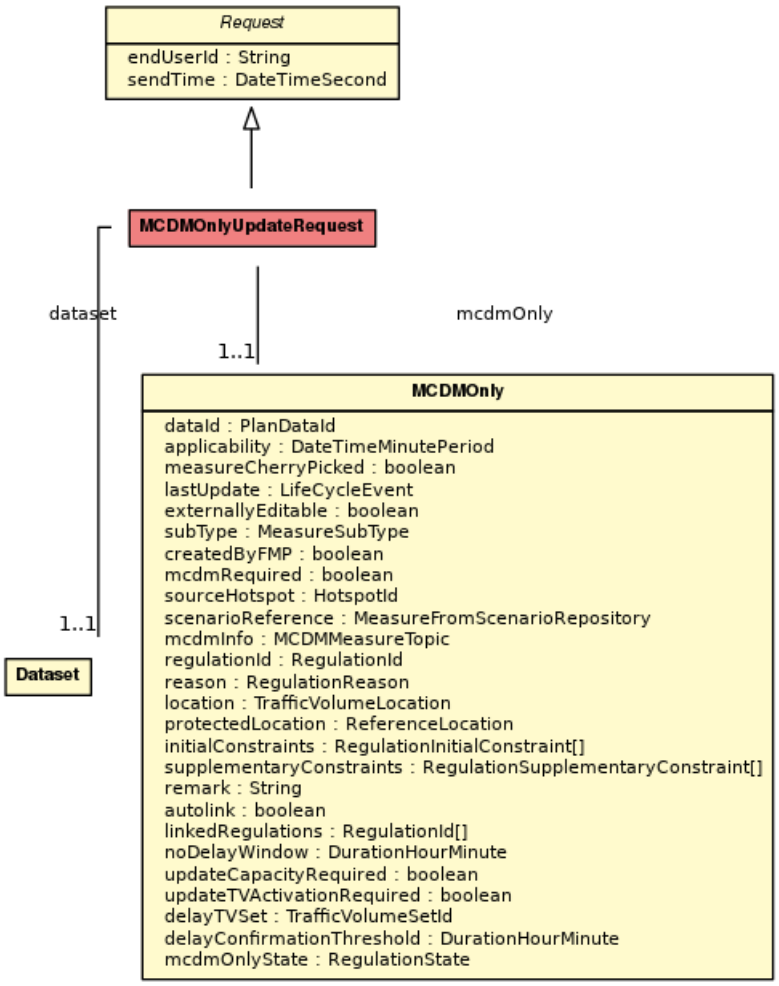

Figure 3.41. `MCDMOnlyUpdateRequest` Class Diagram

(1) Request to update a mcdmOnly.

(2) Note that the `MCDMOnlyUpdateRequest` is trial related: it is only accessible (authorized) during specific trials or on specific test platforms and only for externally editable mcdmOnlys. In STAM trials context only the update of externally editable cherry picked mcdmOnlys are authorized.

(3) Inherits from: Request

(4) Attributes:

a) **Dataset dataset** *(Mandatory)*
Dataset on which the mcdmOnly needs to be updated.

b) **MCDMOnly mcdmOnly** *(Mandatory)*
The mcdmOnly fields to be updated.
Note that only those fields that need to be updated, need to be non null. The other fields simply remain unchanged.
Constraint: See INVALID_MEASURE_APPLICABILITY_PERIOD

(5) Constraint:

a)

Name	INVALID_MEASURE_APPLICABILITY_PERIOD
Attribute	mcdmOnly
Description	Invalid mcdmOnly(measure) applicability period should overlap Dataset period

3.2.16.3. MCDMOnlyUpdateReply

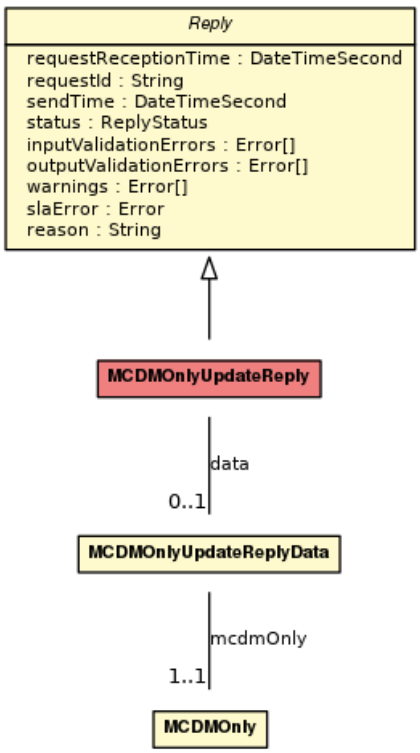

Figure 3.42. MCDMOnlyUpdateReply Class Diagram

(1) Reply returned in response to MCDMOnlyUpdateRequest.

(2) Inherits from: `Reply`

(3) Attributes:

a) **`MCDMOnly` `mcdmOnly`** *(Mandatory)*
The updated mcdmOnly with all fields set by the NM system.

3.2.17. MCDMOnly Cancellation

3.2.17.1. SOAP

(1) The associated SOAP operation is:

```
MCDMOnlyCancelReply cancelMCDMOnly(
    MCDMOnlyCancelRequest request
    )
```

3.2.17.2. `MCDMOnlyCancelRequest`

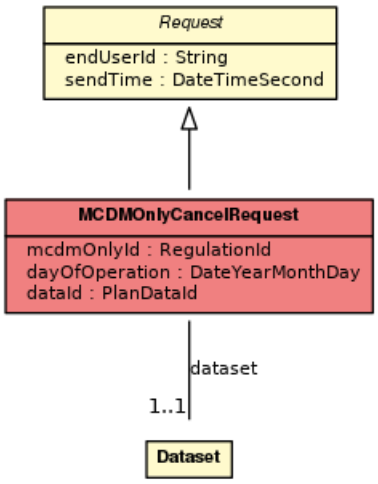

Figure 3.43. `MCDMOnlyCancelRequest` Class Diagram

(1) Request to cancel a mcdmOnly.

(2) Note that the `MCDMOnlyCancelRequest` is trial related: it is only accessible (authorized) during specific trials or on specific test platforms and only for externally editable mcdmOnlys. In STAM trials context only the cancel of externally editable cherry picked mcdmOnlys are authorized.

(3) Inherits from: `Request`

(4) Attributes:

a) **`Dataset` `dataset`** *(Mandatory)*
Dataset on which the mcdmOnly has to be cancelled.

b) **RegulationId mcdmOnlyId** *(Mandatory)*
The identifier of the mcdmOnly to be cancelled.

c) **DateYearMonthDay dayOfOperation** *(Mandatory)*
Day for which the mcdmOnly has to be cancelled.

d) **PlanDataId dataId** *(Mandatory)*
Opaque identifier representing the version of the mcdmOnly to revoke. The caller shall always keep this value unchanged.

3.2.17.3. MCDMOnlyCancelReply

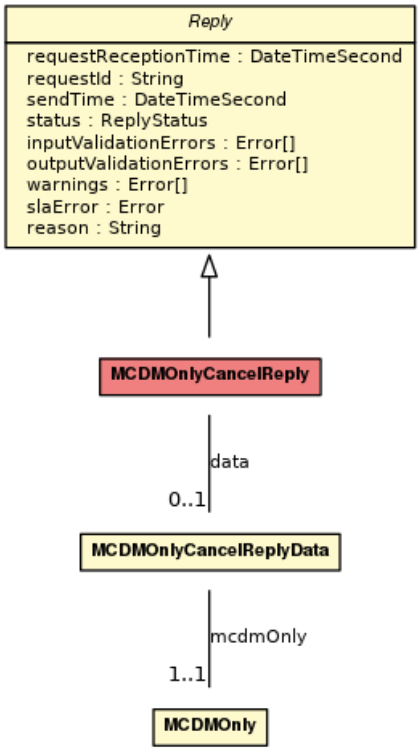

Figure 3.44. MCDMOnlyCancelReply Class Diagram

(1) Reply returned in response to MCDMOnlyCancelRequest.

(2) Inherits from: Reply

(3) Attributes:

a) **MCDMOnly mcdmOnly** *(Mandatory)*
The mcdmOnly that has been cancelled.

3.2.18. Measure Operational Log Retrieval

3.2.18.1. SOAP

(1) The associated SOAP operation is:

```
MeasureOpLogRetrievalReply retrieveMeasureOpLog(
    MeasureOpLogRetrievalRequest request
    )
```

3.2.18.2. `MeasureOpLogRetrievalRequest`

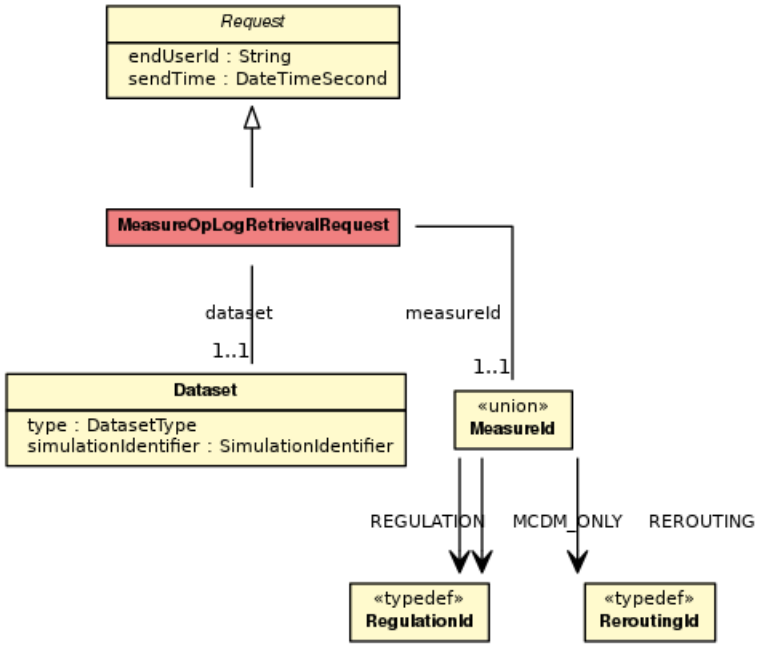

Figure 3.45. `MeasureOpLogRetrievalRequest` Class Diagram

(1) Request to retrieve operational logs of a particular measure.

(2) For a rerouting, the returned operational logs contain the global results of the rerouting (considering all rerouted and non rerouted flights):

a) The network impact: additional or avoided regulations, violated restrictions if no good solution could be found, the important traffic volumes where there is a change (i.e. those traffic volumes that are overloaded/have a peak/sustained traffic counts alert)

b) The complete set of impacted On-Loaded/Off-loaded sectors

c) For the unsuccessfully rerouted flights: the reason why they could not be rerouted

d) For the successfully rerouted flights: the additional /gained distance/fuel/time

e) Per rerouted flight: the rerouting result, a.o., did the flight reroute according to the proposed route or not

f) For a cherry picked regulation: it contains per flight the network impact of the delay

(3) Inherits from: `Request`

(4) Attributes:

a) **`Dataset` `dataset`** *(Mandatory)*
Dataset on which the measure operational logs are requested.

b) **`MeasureId` `measureId`** *(Mandatory)*
Measure id for which the operational logs must be retrieved.

3.2.18.3. `MeasureOpLogRetrievalReply`

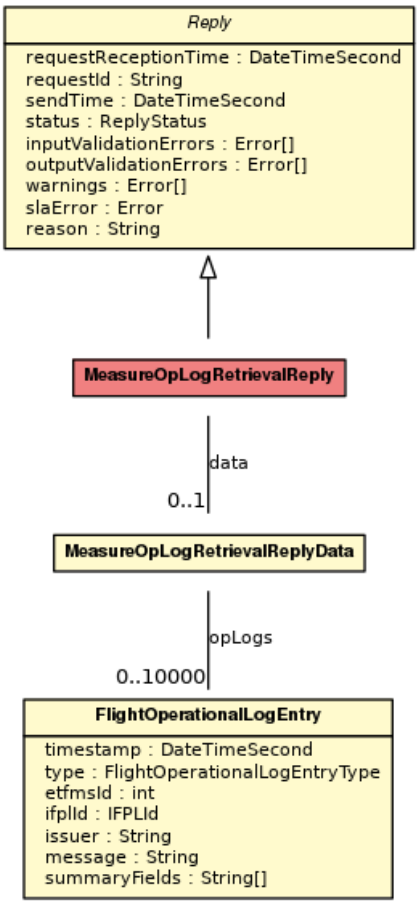

Figure 3.46. `MeasureOpLogRetrievalReply` Class Diagram

(1) Reply returned in response to MeasureOpLogRetrievalRequest.

(2) Inherits from: `Reply`

(3) Attributes:

a) **`Set<FlightOperationalLogEntry> opLogs`** *(Mandatory)*
A set of operational log entries that matched the MeasureOpLogRetrievalRequest criteria. Can be empty (meaning that no opLog matched the criteria).
The result can contain a mix of operational logs concerning the measure as a whole (`etfmsId` = 0) or concerning individual flights (`etfmsId` > 0). In addition, for the flight related operational logs, the IFPL id is not always set.
Constraint: Size must be comprised between 0 and 10000.

3.2.19. Update Flights in Measure

3.2.19.1. SOAP

(1) The associated SOAP operation is:

```
UpdateFlightsInMeasureReply updateFlightsInMeasure(
    UpdateFlightsInMeasureRequest request
    )
```

3.2.19.2. `UpdateFlightsInMeasureRequest`

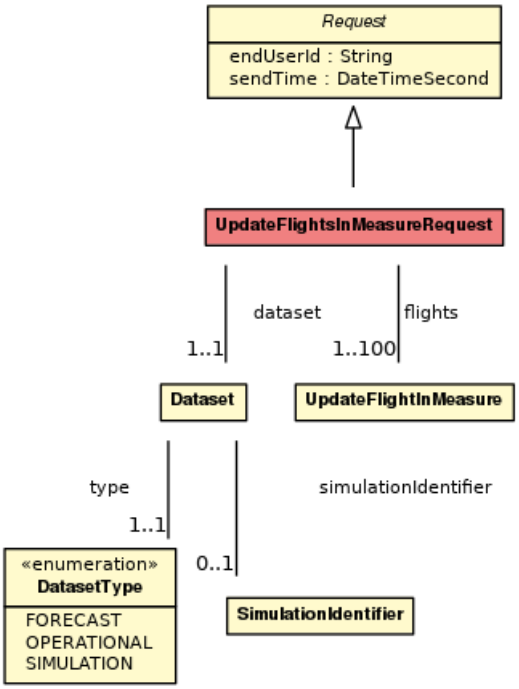

Figure 3.47. `UpdateFlightsInMeasureRequest` Class Diagram

(1) Request to add/remove/modify flights in a measure (for example: change the CTOT of a flight for a regulation).

(2) For a cherry picked measure, `forceFlightInRegulation` allows to initially associate the flight to a measure (and give or change the CTOT) or afterwards add additional flights or modify already added flights. For a cherry picked measure `unforceFlightInRegulation` allows to unforce ("remove") a flight from a regulation (becomes exempted again).

(3) If it is not a cherry picked measure, the service can be used to create a proposal (to e.g. exclude a flight from a regulation) for NMOC to review and accept/reject. (MCDM services can be used to check the status of the request).

(4)

Regulation proposal context:

The update kinds `forceFlightInRegulation`/`unforceFlightInRegulation` are accessible (authorized) when the user has access to cherry pick proposalRegulationFiling. In that case it is limited to adding proposal flights to a regulation or creating a proposal flight to remove (re-exempt) a flight from the cherry picked regulation.
Adding/removing flights results in proposal flights that are forced. When NMOC accepts the regulation proposal, the proposal flight is removed and the normal flights gets a CTOT or CTOT cancellation.
Adding/removing flights (`forceFlightInRegulation`/`unforceFlightInRegulation`) is only allowed if the `MCDMState` of the cherry picked proposal regulation is (re-)set to `DRAFT` and the regulationActivity is applied. For `forceFlightInRegulation`, the Flights need to have their CTOT in the future and CTOT can not be earlier than ETOT.
Flights should not be `ATC_ACTIVATED` yet. Flights can be only be added if hey have the ETO (expected time over according to FTFM) or ATO (actual time over according to CTFM: e.g. for `TACT_ACTIVATED` flights due to DPI) inside the regulation period.
`forceFlightInRegulation`/`unforceFlightInRegulation`, removes any previous proposal flight and creates a new proposal flight.
The `forceFlightInRegulation` results in an updated regulation proposal applicability period if the regulation period needs to be extended or reduced (See ATFCM reference manual).
In that case the B2B client needs to wait until the regulationActivity is `APPLIED` again before doing any subsequent `UpdateFlightsInMeasuresRequests`.

(5)

Note 1

The full service (adding/removing flights to rerouting, etc.) is trial related (STAM): it is only accessible (authorized) during specific trials or on specific test platforms.

(6) Inherits from: `Request`

(7) Attributes:

a) **`Dataset` `dataset`** *(Mandatory)*
Dataset in which the flights need to be updated in one or more measures.
See Forecast and Operational Datasets.

b) **`Set<UpdateFlightInMeasure>` `flights`** *(Mandatory)*
Flights to be updated in one or more measures.
Constraint: Size must be comprised between 1 and 100.

3.2.19.3. `UpdateFlightsInMeasureReply`

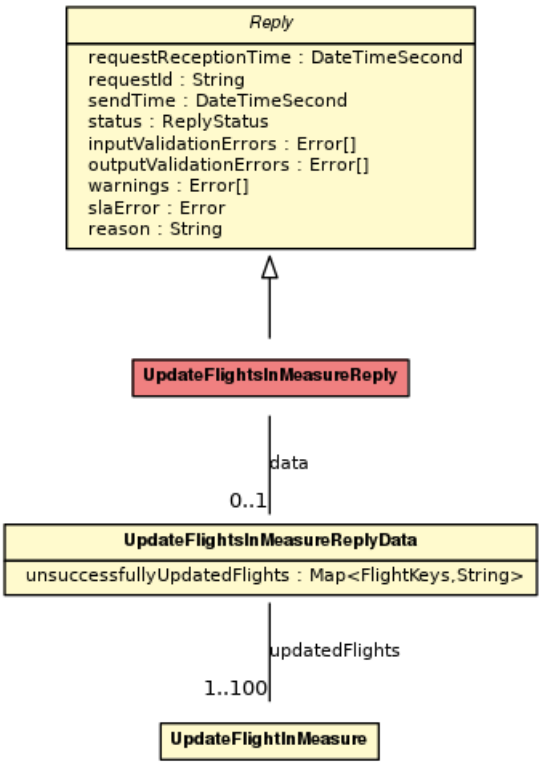

Figure 3.48. `UpdateFlightsInMeasureReply` Class Diagram

(1) Reply returned in response to UpdateFlightsInMeasureRequest.

(2) Inherits from: `Reply`

(3) Attributes:

a) **`Set<UpdateFlightInMeasure> updatedFlights`** *(Optional)*
Flights added successfully to the measure.
Constraint: Size must be comprised between 1 and 100.

b) **`Map<FlightKeys,string> unsuccessfullyUpdatedFlights`** *(Optional)*
A reason description is given for every flights that was not updated successfully.
Constraint: Size must be comprised between 1 and 100.

3.2.20. ATFCM Situation Retrieval

3.2.20.1. SOAP

(1) The associated SOAP operation is:

```
ATFCMSituationReply retrieveATFCMSituation(
    ATFCMSituationRequest request
    )
```

3.2.20.2. `ATFCMSituationRequest`

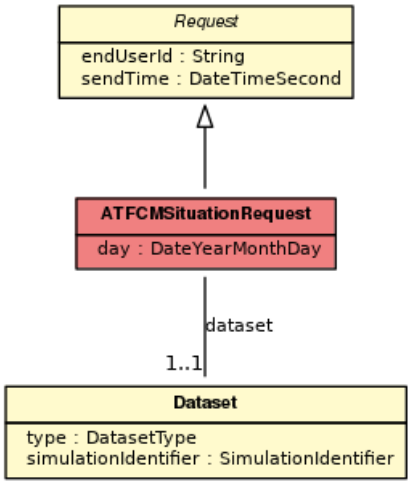

Figure 3.49. `ATFCMSituationRequest` Class Diagram

(1) Request to retrieve the ATFCM situation of a given day.

(2) Inherits from: `Request`

(3) Attributes:

a) **`Dataset` `dataset`** *(Mandatory)*
Dataset from which the ATFCM situation must be retrieved.
Constraint: See `INCONSISTENT_DAY_AND_DATASET_TYPE`

b) **`DateYearMonthDay` `day`** *(Mandatory)*
Day for which the ATFCM situation is requested.
Constraint: See `INCONSISTENT_DAY_AND_DATASET_TYPE`

(4) Constraint:

a)

Name	`INCONSISTENT_DAY_AND_DATASET_TYPE`
Attributes	`day`, `dataset`
Description	The `day` must be in [D-5, D] depending on `dataset` type i.e.: i) `day` must be in [D-5, D] in an `FORECAST` context. ii) `day` must be in [D-1, D] in an `OPERATIONAL` context. iii) `day` has no range constraint in a `SIMULATION` context.

3.2.20.3. `ATFCMSituationReply`

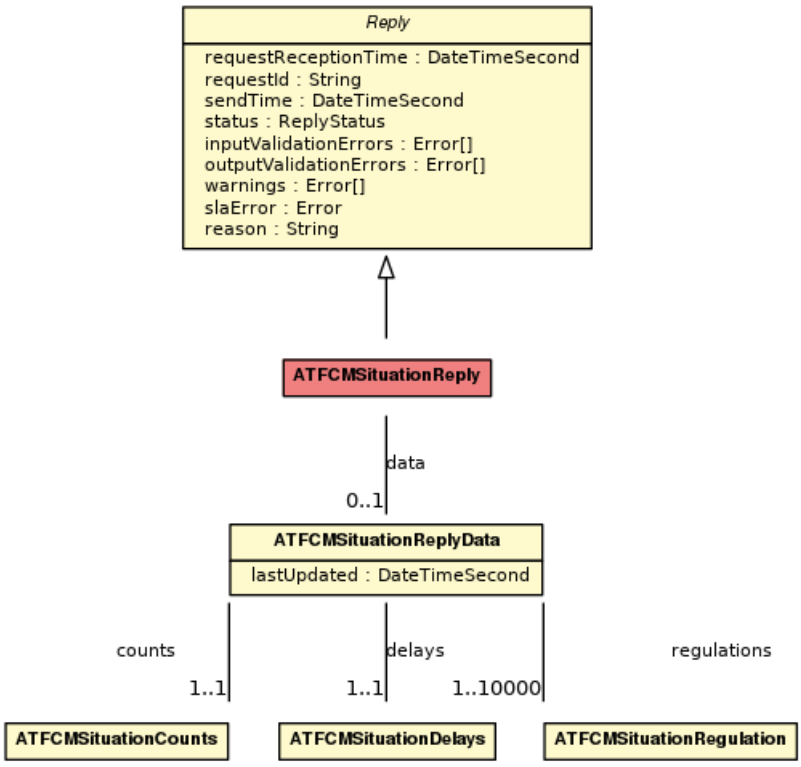

Figure 3.50. `ATFCMSituationReply` Class Diagram

(1) Reply returned in response to ATFCMSituationRequest.

(2) Inherits from: `Reply`

(3) Attributes:

a) **`DateTimeSecond` `lastUpdated`** *(Mandatory)*
Indicates when the ATFCM situation was last updated.

b) **`ATFCMSituationCounts` `counts`** *(Mandatory)*
ATFCM situation counts.

c) **`ATFCMSituationDelays` `delays`** *(Mandatory)*
ATFCM situation delays.

d) **`Set<ATFCMSituationRegulation>` `regulations`** *(Optional)*
ATFCM situation regulations.
Constraint: Size must be comprised between 1 and 10000.

3.2.21. Retrieve Network Impact Assessment

3.2.21.1. SOAP

(1) The associated SOAP operation is:

```
NetworkImpactAssessmentRetrievalReply retrieveNetworkImpactAssessment(
    NetworkImpactAssessmentRetrievalRequest request
    )
```

3.2.21.2. `NetworkImpactAssessmentRetrievalRequest`

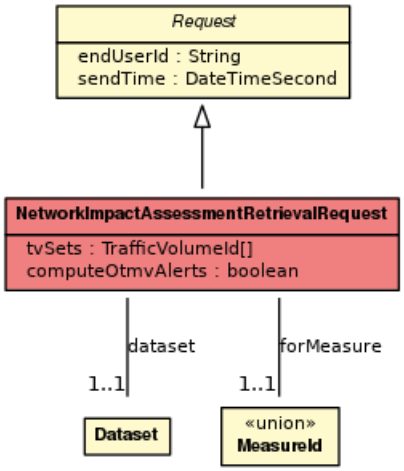

Figure 3.51. `NetworkImpactAssessmentRetrievalRequest` Class Diagram

(1) Request to retrieve the NetworkImpactAssessment for a regulation or rerouting creation/modification/cancellation.

(2) When the user e.g. creates a regulation in a simulation the `NetworkImpactAssessmentRetrievalRequest` will return a summary of the impact:

(3) a) In terms of delay changes in the directly or indirectly impacted regulations (via a Delta ATFCMSituation) (See also `ATFCMSituationRequest`)

b) In terms of count changes in the directly or indirectly impacted traffic volumes (via delta counts).

(4) The user would then typically request to see the detailed counts/flightlists on those impacted traffic volumes where the B2B client could e.g. show the before and after situation (by doing a flightlist/count query and showing the delta as returned by the `NetworkImpactAssessmentRetrievalRequest`).

(5) The `NetworkImpactAssessmentRetrievalRequest` returns which traffic volumes are impacted and gives a summary of the impact. The summary includes the impacted active traffic volumes and the non impacted active traffic volumes: i.e. traffic volumes/counts that had or have

otmv alerts or that were or are in overload. (if the user is only interrested in the changed counts, it is up to the B2B client to filter out unnecessary counts/traffic volumes).

(6) In simulation dataset context:

(7) a) Returns the Delta ATFCMSituation (See also `ATFCMSituationRequest`) comparing the before and after situation.

b) Returns the relevant active traffic volumes and the delta counts and otmv alerts.

(8) In forecast/operational dataset context:

(9) a) Returns no Delta ATFCMSituation.

b) Returns the relevant count changes in the active traffic volumes comparing normal vs proposal flights: supporting rerouting with proposal flights and proposal cherry picked regulations (both with proposal flights).

(10) Note that in simulation context the user needs to have asked explicitely to `computeNetworkImpactAssessment` in the regulation/rerouting request. (see e.g. `RegulationCreationRequest`)

(11) Note that in simulation context, this operation can take some time (depending on the number of flights directly and indirectly impacted). It could take ~ 1 minute to compute.

(12) Inherits from: `Request`

(13) Attributes:

a) **`Dataset` `dataset`** *(Mandatory)*
Dataset on which the regulation needs to be created.

b) **`MeasureId` `forMeasure`** *(Mandatory)*
The regulation or rerouting for which to retrieve the NetworkImpactAssessment of the last creation or modification or cancellation.

c) **`Set<TrafficVolumeId>` `tvSets`** *(Optional)*
In the NetworkImpactAssessmentReply for the counts, only returns the traffic volumes belonging to at least one of the given traffic volume sets.
Constraint: Size must be comprised between 0 and ∞.

d) **`boolean` `computeOtmvAlerts`** *(Optional)*
Indicates if OTMV alerts need to be computed (e.g., is the flight in an OTMV peak and during what count periods; see OtmvAlert) or not.

3.2.21.3. NetworkImpactAssessmentRetrievalReply

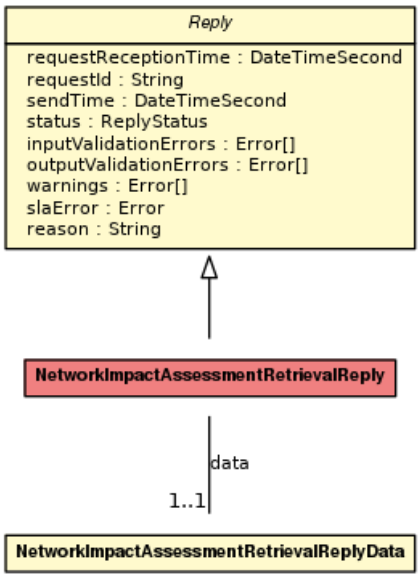

Figure 3.52. `NetworkImpactAssessmentRetrievalReply` Class Diagram

(1) Reply to the NetworkImpactAssessmentRetrievalRequest for a regulation or rerouting creation/modification/cancellation.

(2) Inherits from: `Reply`

(3) Attributes:

a) **`DateTimeSecond` `lastUpdated`** *(Mandatory)*
Last updated.

b) **`DeltaATFCMSituation` `atfcmSituationChanges`** *(Optional)*

c) **`Map<TrafficVolumeId,Map<CountsCalculationTypeAndInterval,Map<DateTimeMinutePeriodWithUFN,DeltaCounts>>>` `countsChanges`** *(Optional)*
Describes the delta counts, grouped by concerned traffic volume (See also `NetworkImpactAssessmentRetrievalRequest`) for which counts/traffic volumes are returned) and `CountsCalculationTypeAndInterval` (basically entry or occupancy) and counts period.
The countsChanges contains the OTMV alerts before/after and a list of before/after counts. Note that in case there is only 1 single e.g. entry count impacted, the `DateTimeMinutePeriodWithUFN` is not a period but the unt time is null. Otherwise e.g. for a period [10:00, 11:20[with entry counts (step is 20 minutes, duration is 60 minutes : See `CountsInterval`) then there are 5 delta counts : the first one representing the count [10:00 , 11:00] the second one : [10:20, 11:20[.. and the last one [11:20, 12:20[
Constraints:

i) Size must be comprised between 0 and ∞.

ii) Item size must be comprised between 0 and ∞.

iii) Item size must be comprised between 0 and ∞.

3.3. McdmService Port Type

3.3.1. Overview

3.3.1.1. Introduction

(1) MCDM operations provide requests that aim at managing MCDM coordination:

a) MCDMTopicListRequest / MCDMTopicListReply

b) MCDMTopicUpdateRequest / MCDMTopicUpdateReply

c) MCDMTopicMessageRetrievalRequest / MCDMTopicMessageRetrievalReply

d) MCDMStateUpdateRequest / MCDMStateUpdateReply

3.3.1.2. MCDM Operations

(1) The MCDM operations are providing a collaboration facility, which allows coordination between ATFM actors for the implementation of a measure or the resolution of a hotspot.

(2) It provides facilities to each actor to:

a) Monitor and update the state of coordination.

b) Organise their work according to deadlines with the help of ordered tasks and visual timelines.

c) Communicate with each other using an instant messaging tool for coordination topics.

(3) The MCDM operations allows coordination on 3 levels:

a) On the hotspot: Actors can use the coordination tool to comment on the hotspot and give some more context and describe alternatives.

b) On measures: Actors can comment/approve/reject individual measures and manage the actors.

c) On flights linked to a measure: Actors can comment/approve/reject on individual flights and manage the actors.

(4) In the context of regulation proposals a subset of the full MCDM process is used:

a) MCDM messaging or MCDM deadlines are not used. Actors can not be changed or set: only the initiator and NMOC are involved.

b) There is no MCDM on hotspots. There is only MCDM for measures and in case of cherry pick regulations there is also MCDM for the flights.

c) Measure MCDM allows B2B clients to see when NMOC has acknowledged the reception of a proposal regulation and when NMOC has accepted or rejected the proposal regulation or when afterwards the regulation got cancelled (due to other reasons).

d) Flight MCDM allows B2B clients to track for cherry picked regulation which flights got accepted and which flights got rejected and which flights got unforced or deregulated afterwards due to other reasons.

e) The B2B client can update the MCDM state for RegulationProposalWithProposalFlights (via MCDMStateUpdateRequest) : limited to (re-)setting the MCDM state of the proposal regulation to DRAFT or to PROPOSED. In addition the B2B client can query the MCDM data (measures and flights) via MCDMTopicListRequest (based on a measureId) or via the flightlist (atfcm_measure_locations field) or via the RegulationProposalListRequest. The regulation proposal MCDM services are accessible (authorized) when the user has access to cherry pick proposalRegulationFiling.

(5) Note that the full set of MCDM related services are trial related (a.o. STAM): it is only accessible (authorized) during specific trials or on specific test platforms.

3.3.1.3. Update Pattern

(1) Similar to: See Update Pattern.

(2) However in the context of regulation proposals, the `dataId` to be used in `MCDMStateUpdateRequest`, can be either the `dataId` retrieved via the `MCDMTopicListRequest` or retrieved via the `RegulationProposalListRequest`.

3.3.2. MCDM Topic List

3.3.2.1. SOAP

(1) The associated SOAP operation is:

```
MCDMTopicListReply queryMCDM(
    MCDMTopicListRequest request
    )
```

3.3.2.2. `MCDMTopicListRequest`

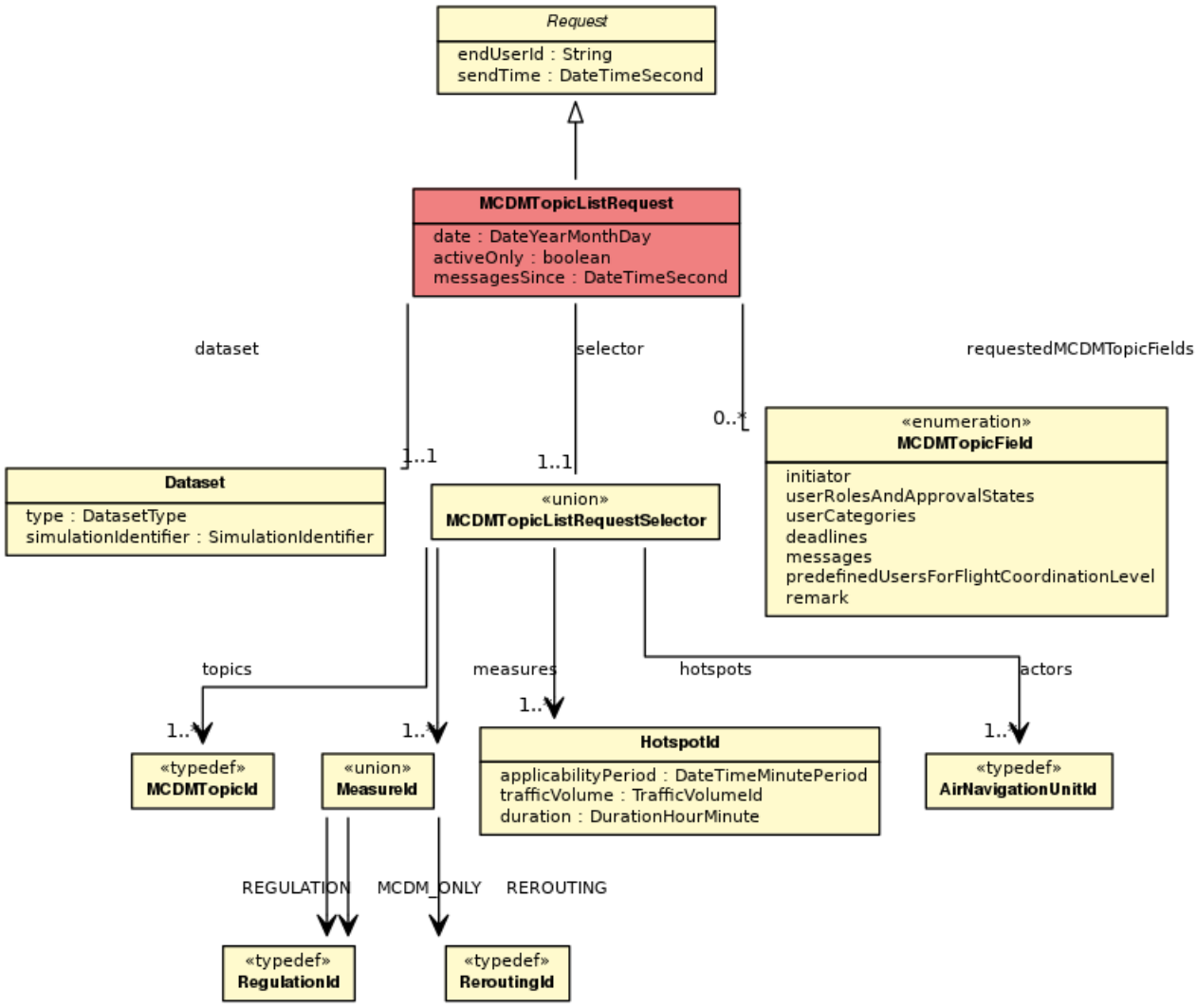

Figure 3.53. `MCDMTopicListRequest` Class Diagram

(1) Request to query a list of MCDM related summary information, as well as to retrieve the MCDM topic details. See also MCDM service Port type for more info. This query method allows the caller to select the topic fields requested in the reply (see `requestedTopicFields`). NM kindly requests its customers to apply the following strategy:

a) As a rule, client applications should never request topic fields that they do not need

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

i) Query the small number of most relevant topic fields to display to the end user (using this `MCDMTopicListRequest`)

ii) Retrieve more details for a given regulation when the end user has selected a topic from the list (also using this `MCDMTopicListRequest`, but with other requested fields)

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

(3) **Note:**
Note that this service is subject to authorization: it can only be used in regulation proposal related context or trail related context (e.g. STAM)

(4) Inherits from: `Request`

(5) Attributes:

a) **`Dataset` dataset** *(Mandatory)*
Dataset on which the MCDM summary is requested
See Forecast and Operational Datasets

b) **`DateYearMonthDay` date** *(Mandatory)*
Day of operation for which the MCDM summary is requested

c) **`MCDMTopicListRequestSelector` selector** *(Mandatory)*
Selects topics either by topic ids or by measure ids or by hotstpot ids or by actor ids.

d) **`boolean activeOnly`** *(Mandatory)*
If `true`, then the reply returns only the active MCDM topics (i.e. those that have an MCDM State that is not yet abandoned or finshed).

e) **`DateTimeSecond` messagesSince** *(Optional)*
Indicates since when the MCDM messages have to be retrieved.
Typically the client would give the previous Reply.requestReceptionTime to get all the messages that were posted since the previous request. (Using the previous Reply.sendTime can lead to messages being lost due to race conditions). However this can lead to receiving messages twice, so the client should filter out duplicates (especially for those that have a timestamp between `[lastReply.requestReceptionTime, lastReply.sendTime]`)
Constraints:

i) See `MESSAGES_SINCE_CANNOT_BE_NULL_IF_REQUESTED`

ii) See `MESSAGES_SINCE_CANNOT_BE_OLDER_THAN_FOUR_HOURS`

iii) See `MESSAGES_SINCE_MUST_BE_NULL_IF_NOT_REQUESTED`

f) **`Set<MCDMTopicField>` requestedMCDMTopicFields** *(Mandatory)*
The reply returns only the requested topic fields in this set, and only if the values of these requested fields are available at NM. Note that the topic id is always returned.
Constraints:

i) Size must be comprised between 0 and 7.

ii) See MESSAGES_SINCE_CANNOT_BE_NULL_IF_REQUESTED

iii) See MESSAGES_SINCE_MUST_BE_NULL_IF_NOT_REQUESTED

(6) Constraints:

a)

Name	MESSAGES_SINCE_MUST_BE_NULL_IF_NOT_REQUESTED
Attributes	messagesSince, requestedMCDMTopicFields
Description	The attribute messagesSince must be null if requestedMCDMTopicFields.messages is not requested.

b)

Name	MESSAGES_SINCE_CANNOT_BE_NULL_IF_REQUESTED
Attributes	messagesSince, requestedMCDMTopicFields
Description	The attribute messagesSince is mandatory if requestedMCDMTopicFields.messages is requested.

c)

Name	MESSAGES_SINCE_CANNOT_BE_OLDER_THAN_FOUR_HOURS
Attribute	messagesSince
Description	The date time messagesSince cannot be before the requested time minus 4 hours.

3.3.2.3. `MCDMTopicListReply`

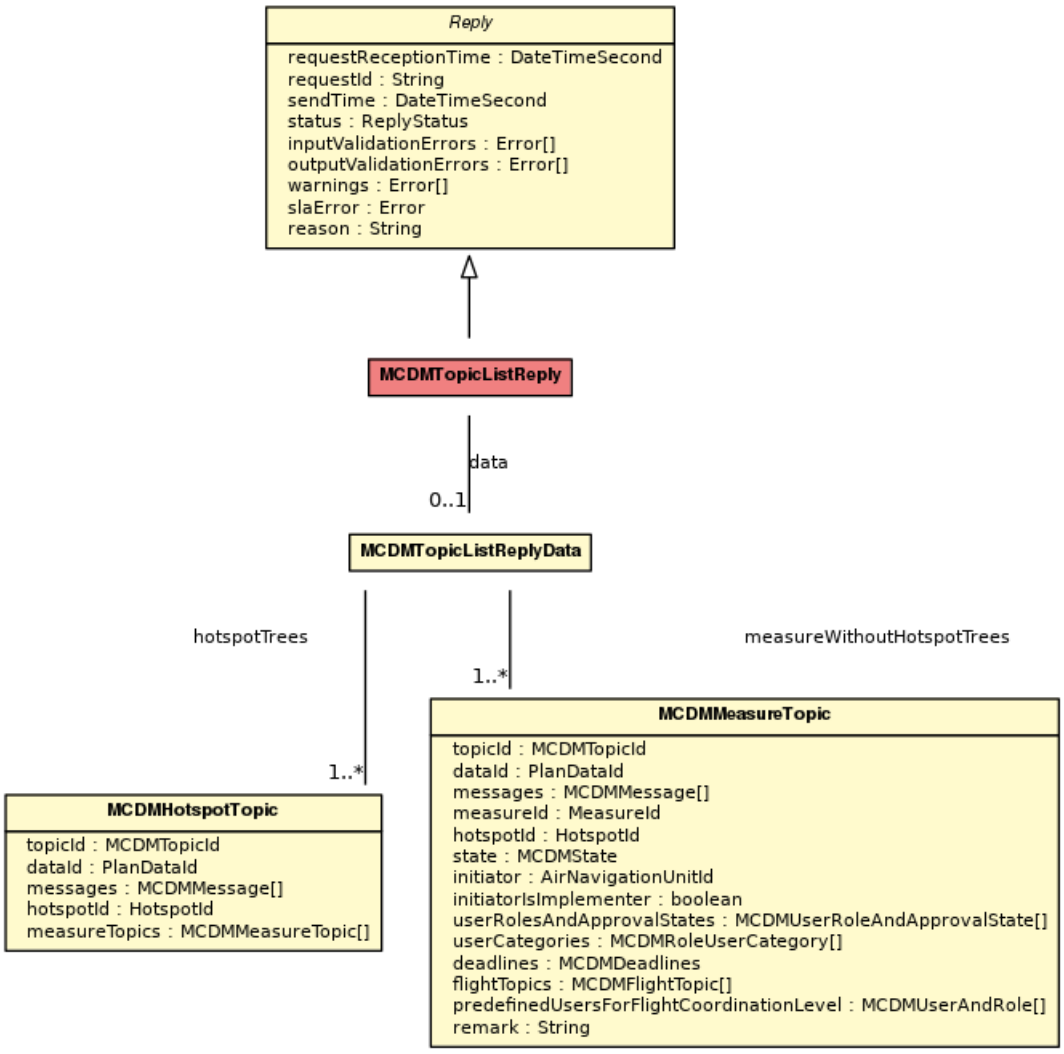

Figure 3.54. `MCDMTopicListReply` Class Diagram

(1) Reply returned in response to MCDMTopicListRequest.

(2) Inherits from: `Reply`

(3) Attributes:

a) **`Set<MCDMHotspotTopic> hotspotTrees`** *(Optional)*
A set of hotspot topic details that matched the MCDMTopicListRequest criteria. Can be empty (meaning that no hotspot topics matched the criteria).

Note:
Note that `hotspotTrees` will only be present if the measures that have an associated hotspot in the measure data (for example regulation proposals can have a

hotspot inside the regulation proposal data). So when querying on for example a `measureId` or a `topicId`, the results can be returned inside `hotspotTrees.measureTopics` (which only has 1 measure when quering on measureId) and inside this measureTopic , one would find the `MCDMState` of all flights concerned.

Constraint: Size must be comprised between 1 and ∞.

b) **`Set<MCDMMeasureTopic> measureWithoutHotspotTrees`** *(Optional)*
A set of measure topic details which are not linked to a hotspot, that matched the MCDMTopicListRequest criteria.
Can be empty (meaning that no such measure topics matched the criteria).
Constraints:

i) Size must be comprised between 1 and ∞.

ii) See HOTSPOT_ID_CANNOT_BE_SET_IN_MEASURE_TOPICS

(4) Constraint:

a)

Name	HOTSPOT_ID_CANNOT_BE_SET_IN_MEASURE_TOPICS
Attribute	measureWithoutHotspotTrees
Description	For every measure topics from the `measureWithoutHotspotTrees` set, the `hotspotId` attribute must be null.

3.3.3. MCDM Topic Update

3.3.3.1. SOAP

(1) The associated SOAP operation is:

```
MCDMTopicUpdateReply updateMCDM(
    MCDMTopicUpdateRequest request
    )
```

3.3.3.2. `MCDMTopicUpdateRequest`

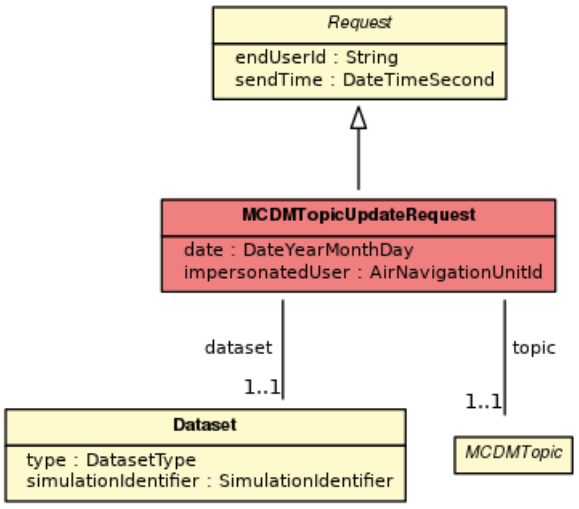

Figure 3.55. `MCDMTopicUpdateRequest` Class Diagram

(1) Request to update a MCDM topic (e.g., change a vote or change the roles and approvals for users).

(2) **Note:**
Note that this sevice is trial related (e.g. STAM). So it is only available during specific trials or on specific test platforms.

(3) Inherits from: `Request`

(4) Attributes:

a) **`Dataset` `dataset`** *(Mandatory)*
Dataset on which the MCDM topic update is requested

b) **`DateYearMonthDay` `date`** *(Mandatory)*
Day of operation for which the MCDM topic update is requested

c) **`AirNavigationUnitId` `impersonatedUser`** *(Mandatory)*
The user on behalf the update is done or the message is sent.

d) **`MCDMTopic` `topic`** *(Mandatory)*
The topic to be updated

3.3.3.3. `MCDMTopicUpdateReply`

(1) Reply returned in response to MCDMTopicUpdateRequest.

(2) Inherits from: `Reply`

(3) Attributes:

a) **`MCDMTopic topic`** *(Mandatory)*
The updated MCDM topic details.

3.3.4. MCDM Topic Message Retrieval

3.3.4.1. SOAP

(1) The associated SOAP operation is:

```
MCDMTopicMessageRetrievalReply retrieveMCDMMessages(
    MCDMTopicMessageRetrievalRequest request
    )
```

3.3.4.2. `MCDMTopicMessageRetrievalRequest`

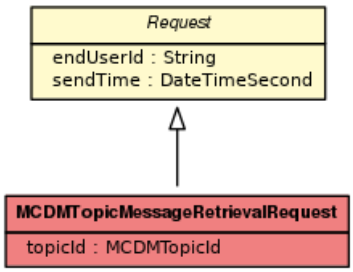

Figure 3.56. `MCDMTopicMessageRetrievalRequest` Class Diagram

(1) Request to retrieve all messages to/from all actors for a given topic.

(2) **Note:**
Note that this sevice is trial related (e.g. STAM). So it is only available during specific trials or on specific test platforms.

(3) Inherits from: `Request`

(4) Attributes:

a) **`MCDMTopicId topicId`** *(Mandatory)*
The MCDM topic identifier for which we want to retrieve all the MCDM messages.

3.3.4.3. `MCDMTopicMessageRetrievalReply`

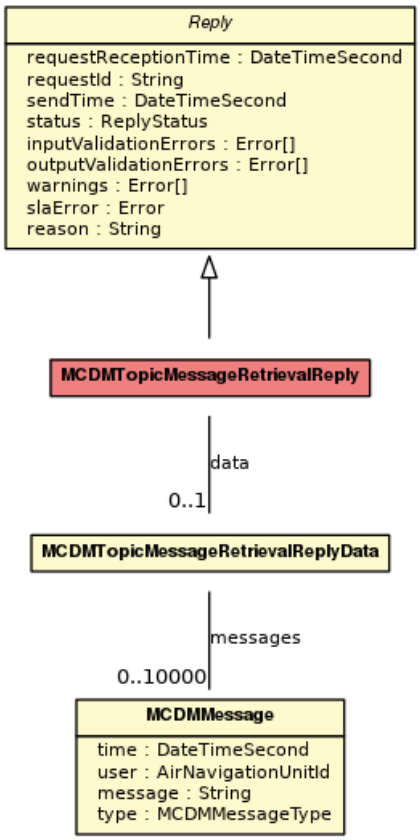

Figure 3.57. `MCDMTopicMessageRetrievalReply` Class Diagram

(1) Reply returned in response to MCDMTopicMessageRetrievalRequest.

(2) Inherits from: `Reply`

(3) Attributes:

a) **`Set<MCDMMessage> messages`** *(Mandatory)*
A set of MCDM messages that matched the MCDMTopicMessageRetrievalRequest criteria. Can be empty (meaning that no messages matched the criteria).
Constraint: Size must be comprised between 0 and 10000.

3.3.5. MCDM State Update

3.3.5.1. SOAP

(1) The associated SOAP operation is:

```
MCDMStateUpdateReply updateMCDMState(
    MCDMStateUpdateRequest request
    )
```

3.3.5.2. `MCDMStateUpdateRequest`

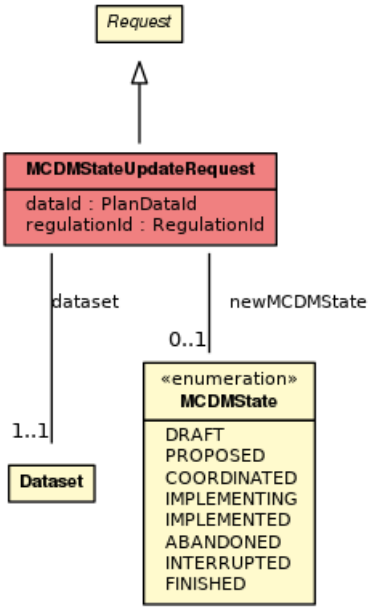

Figure 3.58. `MCDMStateUpdateRequest` Class Diagram

(1) Request to update the MCDMState of a regulation (typically used in the context of regulation proposals).

(2) **Note 1:**
Note that this service is subject to authorization: it can only be used in regulation proposal related context or trial related context (e.g. STAM).

(3) **Note 2:**
Note that in regulation proposal context, it is only allowed to change he MCDM state of the regulation when NMOC is not yet reviewing the regulation or when the review has been completed (so then `MCDMState` is DRAFT, PROPOSED, IMPLEMENTED, ABANDONED or INTERUPTED)
Except when the `regulationState` is CANCELLED or TERMINATED, then it is no longer possible to update the `MCDMState`. The MCDM state can only be changed to PROPOSED from the DRAFT `MCDMState`.

(4) Inherits from: `Request`

(5) Attributes:

a) **`PlanDataId` `dataId`** *(Mandatory)*
Opaque identifier representing the version of the MCDM state to update. See Update Pattern.

b) **`Dataset` `dataset`** *(Mandatory)*
Dataset to which the regulation belongs. See Forecast and Operational Datasets.

c) **RegulationId regulationId** *(Mandatory)*
The unique id of the regulation or regulation proposal.

d) **MCDMState newMCDMState** *(Optional)*
The new MCDM state.
In the context of regulation proposals, only DRAFT or PROPOSED are allowed (DRAFT when the B2B client wants to update a regulation or a regulation proposal and PROPOSED when the B2B client wants NMOC to review): See MCDM service Port type.

3.3.5.3. MCDMStateUpdateReply

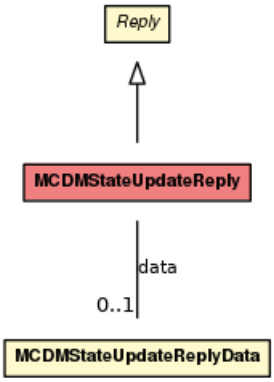

Figure 3.59. MCDMStateUpdateReply Class Diagram

(1) Reply to a MCDMStateUpdateRequest.

(2)

Special error conditions:

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

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

c) OBJECT_NOT_FOUND [Permanent error] If the measure is not known or if it has no MCDM.

d) Note that there are also the INVALID_INPUT replyStatus errors, that cover the errors detected by the backend (See Transaction and Errors). For example : trying to change the MCDMState to FINISHED in the context of regulation proposal).

(3) Inherits from: Reply

(4) Attributes:

a) **MCDMState newMCDMState** *(Mandatory)*

The resulting MCDM state.

3.4. TacticalUpdatesService Port Type

3.4.1. Overview

3.4.1.1. Introduction

(1) This service is intended to provide querying and update capabilities on Tactical updates. The requests currently available are:

a) SectorConfigurationPlanRetrievalRequest / SectorConfigurationPlanRetrievalReply

b) SectorConfigurationPlanUpdateRequest / SectorConfigurationPlanUpdateReply

c) CapacityPlanRetrievalRequest / CapacityPlanRetrievalReply

d) CapacityPlanUpdateRequest / CapacityPlanUpdateReply

e) TrafficVolumeActivationPlanRetrievalRequest / TrafficVolumeActivationPlanRetrievalReply

f) TrafficVolumeActivationPlanUpdateRequest / TrafficVolumeActivationPlanUpdateReply

g) OTMVPlanRetrievalRequest / OTMVPlanRetrievalReply

h) OTMVPlanUpdateRequest / OTMVPlanUpdateReply

i) RunwayConfigurationPlanRetrievalRequest / RunwayConfigurationPlanRetrievalReply

j) RunwayConfigurationPlanUpdateRequest / RunwayConfigurationPlanUpdateReply

k) HotspotListRequest / HotspotListReply

l) HotspotPlanUpdateRequest / HotspotPlanUpdateReply

3.4.1.2. CACD Retrievals vs. Tactical Situation

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

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

b) NM flow system for tactical updates

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

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

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

c) To retrieve a Daily Plan:

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

3.4.1.3. Daily Plan Update Pattern

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

i) There was no concurrent update, or

ii) There were identical concurrent updates, and/or

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

e) IMPORTANT: NM insists that the B2B client only does a tactical update to a plan in case something has changed for that plan. Updates that do not change anything are logged. In case too many of such non-updates are logged, then access can be restricted. This ensures that the backend responsible for the (pre-) tactical updates, does not get overloaded with non-updates.

(7) The `data id` is an opaque identifier of the version of the global state of the backend system related to CACD or tactical updatable related data (not pure flight data). Whenever `dataId` is passed in an update request, the system verifies if there have been conflicting updates between what the B2B client tried to update (wrt the state of the system linked to the dataid) and the latest state. Note that the `dataId` represent the global state of the backend system (not linked to specific

locations). It changes continuously (between subsequent retrievePlan requests). However the fact that it changes continuously does not impact the B2B client, as it is only used to detect if there have been conflicting parallel updates between the latest state and what the B2B client changed in the update request.

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

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

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

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

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

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

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

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

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

iv) If the NM systems detect a conflicting parallel update between the time corresponding to the dataId A1/A2.. and the latest NM state wrt AUA A and day X, the reply contains a CONFLICTING_UPDATE error. In that case the client system would warn the oper-

ator that a conflicting parallel update has occurred and would show the local data and the NM data to allow the operator to choose (and optionally update any local system as well)

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

3.4.1.4. Transactions and Errors

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

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

(3) As seen below, some error conditions are permanent in the sense that retrying the transaction later will never solve the problem (typically because the data in the request violates some validation rule), and a few others are temporary, e.g. the transaction failed because the NM system is itself in some temporar y state that prevents it from processing the request (e.g. the daily plan is being transferred see below). In any case, the temporary error conditions are always w ell identified within the reply so that the B2B client knows when it is worth retrying a few minutes later NM insists that the updates that failed due to a tempora ry condition are only retried a few minutes later, not a few seconds later and even less a few milliseconds later.

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

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

3.4.1.5. Allowed Times for Retrievals and Updates

3.4.1.5.1. Forecast and Operational Datasets

(1) See also Forecast and Operational Datasets.

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

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

(4) To summarize:

a) The plan can be updated:

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

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

b) The plan can be retrieved:

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

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

(5) Regarding the plan transfer, the updates to the forecast dataset are rejected from a cut-off time on D-1 that is 16:00 UTC; the NM OPS Room may then fine tune the plan for a little while, and finally transfers it to the operational dataset. The time elapsed between the cut-off time of the forecast dataset and the time at which the plan has been actually transferred to the operational dataset is typically around 10 minutes. This elapsed time may exceptionally be longer though, e.g. in case of unexpected change of the network situation (like unexpected strike or weather conditions), so that the plan can exceptionally be transferred a few hours after the cut-off time. The B2B client designer is invited to take this variability into consideration.
Updates to the plan, in the forecast or operational datasets, are rejected during this elapsed time.

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

3.4.1.5.2. Past

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

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

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

3.4.2. Retrieve a Sector Configuration Plan

3.4.2.1. SOAP

(1) The associated SOAP operation is:

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

3.4.2.2. `SectorConfigurationPlanRetrievalRequest`

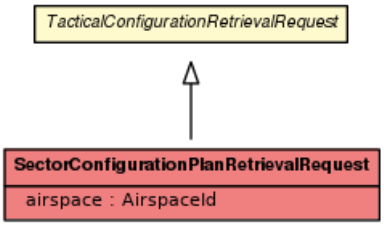

Figure 3.60. `SectorConfigurationPlanRetrievalRequest` Class Diagram

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

(2) Inherits from: `TacticalConfigurationRetrievalRequest`

(3) Attributes:

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

3.4.2.3. `SectorConfigurationPlanRetrievalReply`

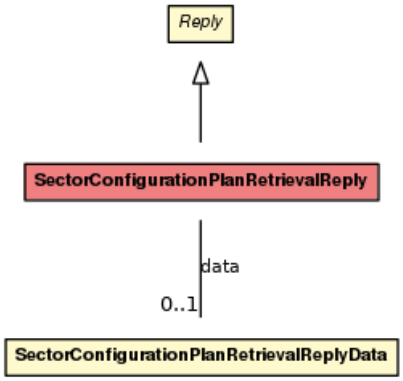

Figure 3.61. `SectorConfigurationPlanRetrievalReply` Class Diagram

(1) Reply to a SectorConfigurationPlanRetrievalRequest.

(2) Special error conditions:

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

(4) Inherits from: `Reply`

(5) Attributes:

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

3.4.3. Update a Sector Configuration Plan

3.4.3.1. SOAP

(1) The associated SOAP operation is:

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

3.4.3.2. `SectorConfigurationPlanUpdateRequest`

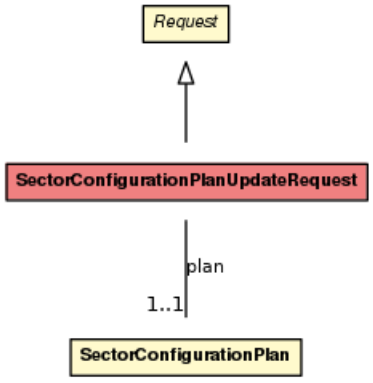

Figure 3.62. `SectorConfigurationPlanUpdateRequest` Class Diagram

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

(2) Inherits from: `Request`

(3) Attributes:

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

3.4.3.3. `SectorConfigurationPlanUpdateReply`

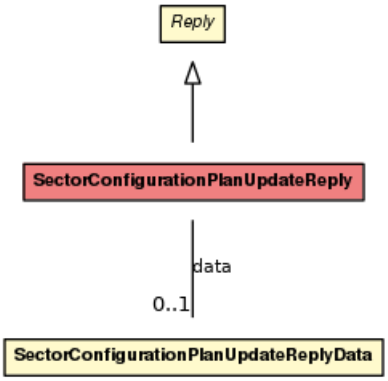

Figure 3.63. `SectorConfigurationPlanUpdateReply` Class Diagram

(1) Reply to a SectorConfigurationPlanUpdateRequest.

(2) Special error conditions:

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

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

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

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

(4) Inherits from: `Reply`

(5) Attributes:

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

3.4.4. Retrieve a Capacity Plan

3.4.4.1. SOAP

(1) The associated SOAP operation is:

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

3.4.4.2. `CapacityPlanRetrievalRequest`

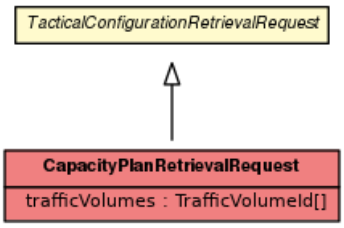

Figure 3.64. `CapacityPlanRetrievalRequest` Class Diagram

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

(2) Inherits from: `TacticalConfigurationRetrievalRequest`

(3) Attributes:

a) **`Set<TrafficVolumeId> trafficVolumes`** *(Mandatory)*
The traffic volume for which the capacity plan is requested.
Constraint: Size must be comprised between 0 and ∞.

3.4.4.3. `CapacityPlanRetrievalReply`

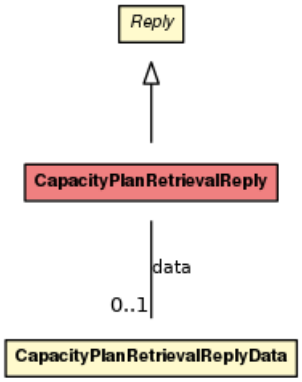

Figure 3.65. `CapacityPlanRetrievalReply` Class Diagram

(1) Reply to a CapacityPlanRetrievalRequest.

(2) Inherits from: `Reply`

(3) Attributes:

a) **`CapacityPlans` plans** *(Mandatory)*
The complete capacity plan for a given traffic volume on a given day.

3.4.5. Update a Capacity Plan

3.4.5.1. SOAP

(1) The associated SOAP operation is:

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

3.4.5.2. `CapacityPlanUpdateRequest`

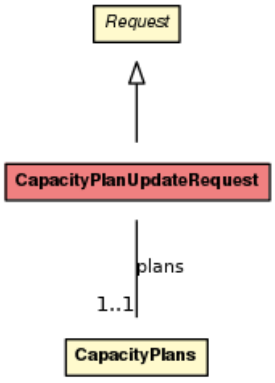

Figure 3.66. `CapacityPlanUpdateRequest` Class Diagram

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

(2) The update of several traffic volumes is possible in the same B2B request.

(3) Inherits from: `Request`

(4) Attributes:

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

3.4.5.3. `CapacityPlanUpdateReply`

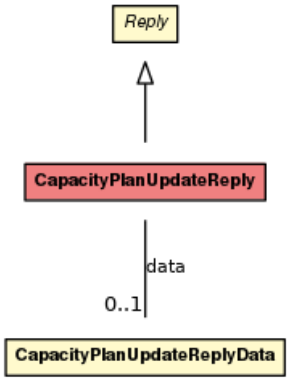

Figure 3.67. `CapacityPlanUpdateReply` Class Diagram

(1) Reply to a CapacityPlanUpdateRequest.

(2) Special error conditions:

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

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

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

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

(4) Inherits from: `Reply`

(5) Attributes:

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

3.4.6. Retrieve a Traffic Volume Activation Plan

3.4.6.1. SOAP

(1) The associated SOAP operation is:

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

3.4.6.2. `TrafficVolumeActivationPlanRetrievalRequest`

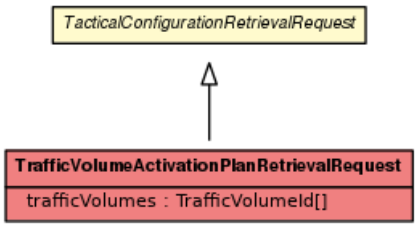

Figure 3.68. `TrafficVolumeActivationPlanRetrievalRequest` Class Diagram

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

(2) Inherits from: `TacticalConfigurationRetrievalRequest`

(3) Attributes:

a) **`Set<TrafficVolumeId> trafficVolumes`** *(Mandatory)*
The traffic volumes for which according activation plans are requested.
Constraint: Size must be comprised between 0 and ∞.

3.4.6.3. `TrafficVolumeActivationPlanRetrievalReply`

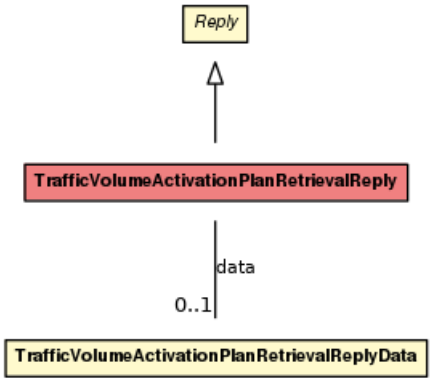

Figure 3.69. `TrafficVolumeActivationPlanRetrievalReply` Class Diagram

(1) Reply to a TrafficVolumeActivationPlanRetrievalRequest.

(2) Inherits from: `Reply`

(3) Attributes:

a) **`TrafficVolumeActivationPlans plans`** *(Mandatory)*
The complete activation plans for a given traffic volumes on a given day.

3.4.7. Update a Traffic Volume Activation Plan

3.4.7.1. SOAP

(1) The associated SOAP operation is:

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

3.4.7.2. `TrafficVolumeActivationPlanUpdateRequest`

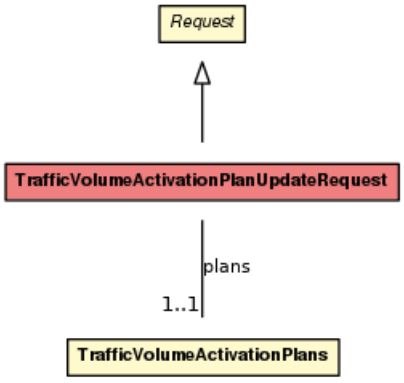

Figure 3.70. `TrafficVolumeActivationPlanUpdateRequest` Class Diagram

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

(2) The update of several traffic volumes is possible in the same B2B request.

(3) Inherits from: `Request`

(4) Attributes:

a) **`TrafficVolumeActivationPlans plans`** *(Mandatory)*
Traffic volumes activation plans reflecting the requested updates.

3.4.7.3. `TrafficVolumeActivationPlanUpdateReply`

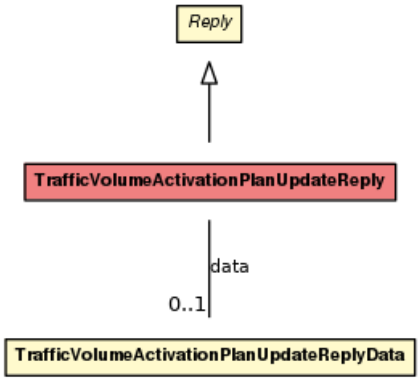

Figure 3.71. `TrafficVolumeActivationPlanUpdateReply` Class Diagram

(1) Reply to a TrafficVolumeActivationPlanUpdateRequest.

(2) Special error conditions:

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

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

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

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

(4) Inherits from: `Reply`

(5) Attributes:

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

3.4.8. Retrieve an OTMV Plan

3.4.8.1. SOAP

(1) The associated SOAP operation is:

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

3.4.8.2. `OTMVPlanRetrievalRequest`

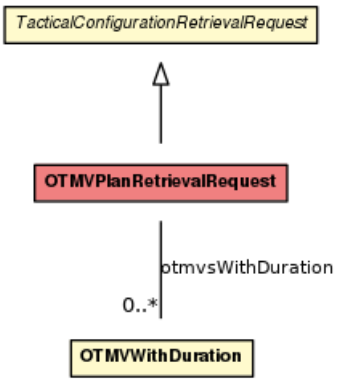

Figure 3.72. `OTMVPlanRetrievalRequest` Class Diagram

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

(2) Inherits from: `TacticalConfigurationRetrievalRequest`

(3) Attributes:

a) **`Set<OTMVWithDuration> otmvsWithDuration`** *(Mandatory)*
The Set of OTMVWithDuration objects, which contain traffic volume and the OTMV plan is requested.
Constraint: Size must be comprised between 0 and ∞.

3.4.8.3. `OTMVPlanRetrievalReply`

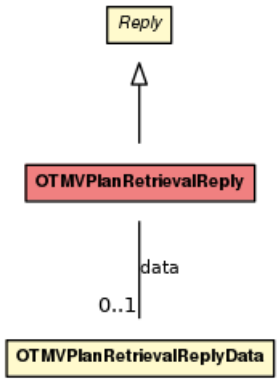

Figure 3.73. `OTMVPlanRetrievalReply` Class Diagram

(1) Reply to a OTMVPlanRetrievalRequest.

(2) Special error conditions:

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

(4) Inherits from: `Reply`

(5) Attributes:

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

3.4.9. Update an OTMV Plan

3.4.9.1. SOAP

(1) The associated SOAP operation is:

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

3.4.9.2. `OTMVPlanUpdateRequest`

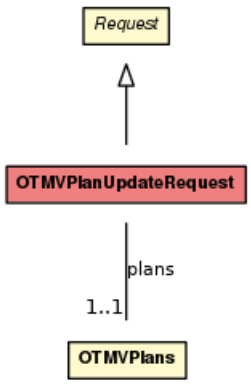

Figure 3.74. `OTMVPlanUpdateRequest` Class Diagram

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

(2) The update of several traffic volumes is possible in the same B2B request.

(3) Inherits from: `Request`

(4) Attributes:

a) **`OTMVPlans plans`** *(Mandatory)*
OTMV plans reflecting the requested updates.

3.4.9.3. `OTMVPlanUpdateReply`

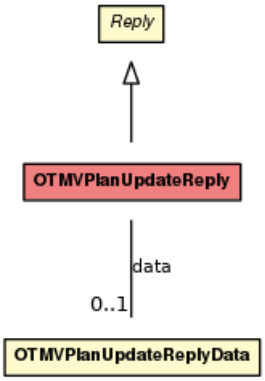

Figure 3.75. `OTMVPlanUpdateReply` Class Diagram

(1) Reply to a OTMVPlanUpdateRequest.

(2) Special error conditions:

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

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

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

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

(4) Inherits from: `Reply`

(5) Attributes:

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

3.4.10. Retrieve a runway configuration plan

3.4.10.1. SOAP

(1) The associated SOAP operation is:

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

3.4.10.2. `RunwayConfigurationPlanRetrievalRequest`

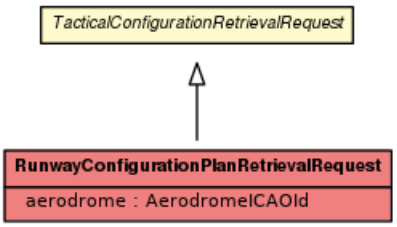

Figure 3.76. `RunwayConfigurationPlanRetrievalRequest` Class Diagram

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

(2) Inherits from: `TacticalConfigurationRetrievalRequest`

(3) Attributes:

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

3.4.10.3. `RunwayConfigurationPlanRetrievalReply`

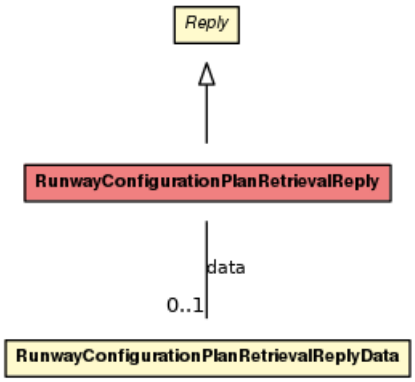

Figure 3.77. `RunwayConfigurationPlanRetrievalReply` Class Diagram

(1) Reply to a RunwayConfigurationPlanRetrievalRequest.

(2) Inherits from: `Reply`

(3) Attributes:

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

3.4.11. Update a runway configuration plan

3.4.11.1. SOAP

(1) The associated SOAP operation is:

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

3.4.11.2. `RunwayConfigurationPlanUpdateRequest`

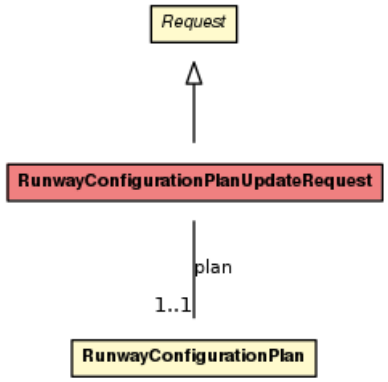

Figure 3.78. `RunwayConfigurationPlanUpdateRequest` Class Diagram

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

(2) Inherits from: `Request`

(3) Attributes:

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

3.4.11.3. `RunwayConfigurationPlanUpdateReply`

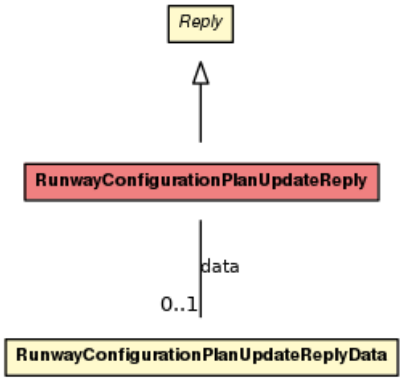

Figure 3.79. `RunwayConfigurationPlanUpdateReply` Class Diagram

(1) Reply to a RunwayConfigurationPlanUpdateRequest.

(2) Special error conditions:

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

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

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

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

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

(4) Inherits from: `Reply`

(5) Attributes:

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

3.4.12. Hotspot List

3.4.12.1. SOAP

(1) The associated SOAP operation is:

```
HotspotListReply queryHotspots(
    HotspotListRequest request
    )
```

3.4.12.2. `HotspotListRequest`

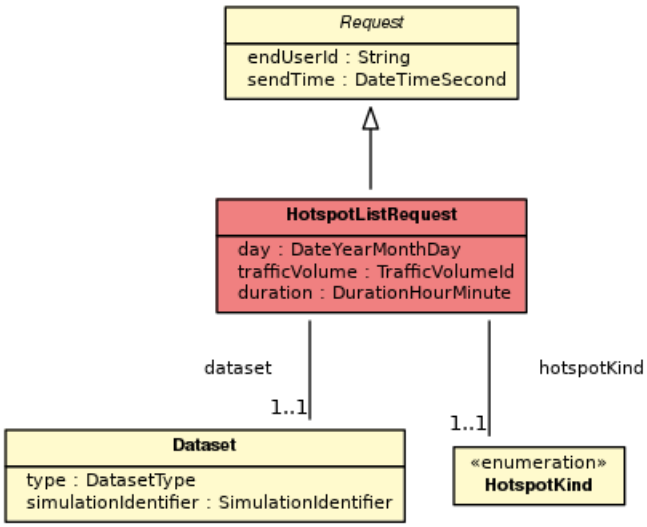

Figure 3.80. `HotspotListRequest` Class Diagram

(1) Request to retrieve for a given day and a hotspotKind, the hotspots for all traffic volumes and for all applicable durations or the hotspots for a given traffic volume for all applicable durations, or to retrieve the hotspots for a specific (traffic volume, duration) pair.

(2) There exists 2 types of hotspots : LOCATION_OF_INTEREST hotspots and PROBLEM hotspots (see `HotspotKind`). Basically : Location of interest hotspots are used operationally to raise awarness about potential hotspots or problems.

(3) LOCATION_OF_INTEREST hotspots hotspot lists are subject to authorization. PROBLEM hotspots are STAM related hotspots and are linked to a demand-capacity inbalance. PROBLEM hotspots are only accessible (authorized) during specific trials or on test platforms. Only PROBLEM hotspots are visible in flightlist and regulations.

(4) Specific error conditions:

a) Permanent error: `OBJECT_NOT_FOUND` reply status if the traffic volume is not known or if it is a traffic volume that does not have as reference location an airspace.

(5) Inherits from: `Request`

(6) Attributes:

a) **`Dataset` `dataset`** *(Mandatory)*
Dataset for which the hotspot list is requested. See Forecast and Operational Datasets.

b) **DateYearMonthDay day** *(Mandatory)*
Day for which the hotspot list is requested.

c) **TrafficVolumeId trafficVolume** *(Contextual)*
The traffic volume for which the hotspot list is requested. If not present then all hotspots of hotspotKind are returned.
Presence:

i) Optional in HotspotListRequest

ii) Mandatory otherwise.

d) **DurationHourMinute duration** *(Optional)*
Selects the hotspots applying to the given traffic volume according to their duration. When not specified, all duration are considered.
Constraint: See DURATION_MUST_BE_0001_WITH_LOCATION_OF_INTEREST

e) **HotspotKind hotspotKind** *(Mandatory)*
The kind of hotspot.
Constraint: See DURATION_MUST_BE_0001_WITH_LOCATION_OF_INTEREST

(7) Constraint:

a)

Name	DURATION_MUST_BE_0001_WITH_LOCATION_OF_INTEREST
Attributes	duration, hotspotKind
Description	The duration must be 0001 when hotspotKind is LOCATION_OF_INTEREST.

3.4.12.3. `HotspotListReply`

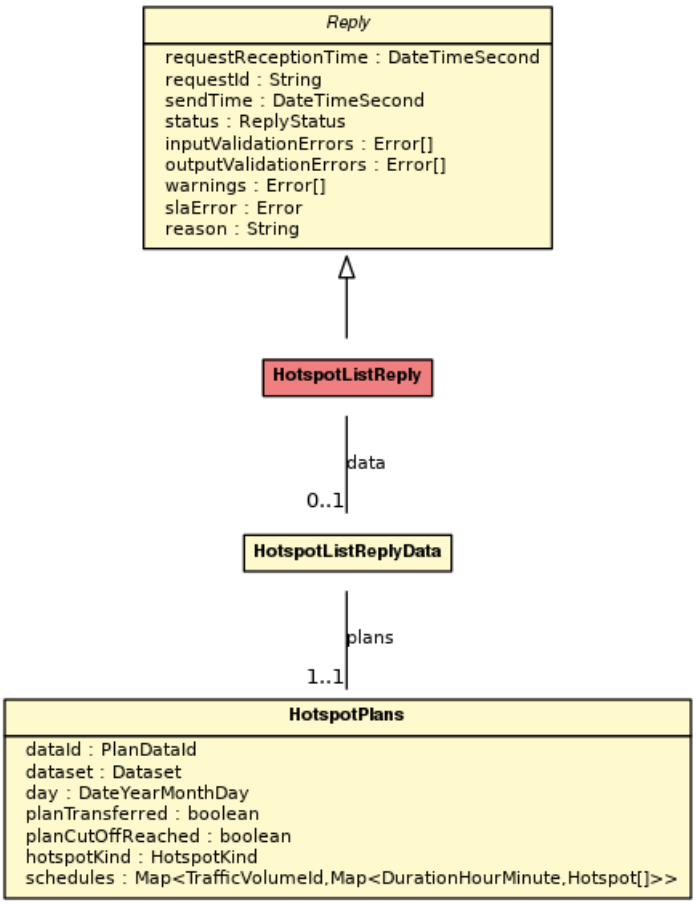

Figure 3.81. `HotspotListReply` Class Diagram

(1) Reply returned in response to HotspotListRequest.

(2) Inherits from: `Reply`

(3) Attributes:

a) **`HotspotPlans` `plans`** *(Mandatory)*
All hotspots for a given (traffic volume, duration) pair on a given day.

3.4.13. Hotspot Plan Update

3.4.13.1. SOAP

(1) The associated SOAP operation is:

```
HotspotPlanUpdateReply updateHotspots(
    HotspotPlanUpdateRequest request
    )
```

3.4.13.2. `HotspotPlanUpdateRequest`

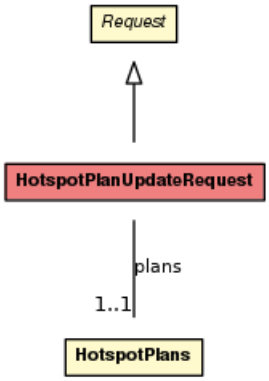

Figure 3.82. `HotspotPlanUpdateRequest` Class Diagram

(1) Request to update the hotspot plans for a set of (traffic volume, duration) pairs on a given day.

(2) Note that the service is trial related (STAM): it is only accessible (authorized) during specific trials or on test platforms.

(3) Note that the service includes Hotspot creation and Hotspot cancellation.

(4) Specific error conditions:

a) Temporary error: `INVALID_DATASET` reply status if the `FORECAST` update was rejected due to D-1 forecast update cut-off or if the `OPERATIONAL` update was rejected due to D-1 plan not transferred yet.

b) Permanent error: `CONFLICTING_UPDATE` reply status if the update failed due to incompatible concurrent changes.

c) Permanent error: `OBJECT_NOT_FOUND` reply status if one of the traffic volumes is not known or if there is PROBLEM hotspot with a traffic volume that does not have as reference location an airspace.

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

e) If errors are detected with at least one of the hotspot plans, then none of the hotspotplans are updated in NM systems.

(5) Inherits from: `Request`

(6) Attributes:

a) **`HotspotPlans` `plans`** *(Mandatory)*

Hotspot plans reflecting the requested updates.

3.4.13.3. `HotspotPlanUpdateReply`

(1) Reply returned in response to HotspotPlanUpdateRequest.

(2) Inherits from: `Reply`

(3) Attributes:

a) **`HotspotPlans` `plans`** *(Mandatory)*
The complete hotspot plans for the given (traffic volume, duration) pairs on a given day, resulting from the update.

3.5. SimulationsService Port Type

3.5.1. Overview

3.5.1.1. Introduction

(1) This service is intended to provide querying and management capabilities of Simulations. The requests currently available are:

a) SimulationListRequest / SimulationListReply

b) SimulationAvailabilityRequest / SimulationAvailabilityReply

c) SimulationStartRequest / SimulationStartReply

d) SimulationStopRequest / SimulationStopReply

3.5.1.2. Simulations

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

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

(3) There are 4 main type of simulations:

- Simulations on OPERATIONAL: Such simulations are started with a snapshot of the data contained in OPERATIONAL. Flights and regulations are automatically loaded from the OPERA-

TIONAL dataset when such a simulation is started for the simulationPeriod (i.e. The period that is being simulated). The user can then modify regulations and reroutings and tactical updates and query the results via flightlist, counts,regulations list etc. Optionally the B2B user can compare the results (e.g. counts) of the simulation with the results on the reference dataset (i.e. the operational dataset itself).

- Simulations on FORECAST: Such simulations are started with a snapshot of the data contained in FORECAST. The user can then modify regulations and reroutings and tactical updates and query the results via flightlist, counts,regulations list etc. Optionally the B2B user can compare the results (e.g. counts) of the simulation with the results on the reference dataset (i.e. the forecast dataset itself).

- STANDALONE_SIMEX: The simulation is a standalone SIMulation Experiment (SIMEX). Special future events are typically prepared and simulated on SIMEX with specially modified environment (CACD) data and forecasted traffic for a date in the future. These type of simulation are managed (start/stop) and prepared by NMOC for the other users (B2B and B2C) to have a look at the results. Optionally the B2B user can make changes to this STANDALONE_SIMEX. There can be more than one STANDALONE_SIMEX: NMOC can prepare multiple independent future events at the same time.

- Simulations on a STANDALONE_SIMEX: Such simulations are started with a snapshot of the data contained in a STANDALONE_SIMEX reference dataset. The user can then modify regulations and reroutings and tactical updates and query the results via flightlist, counts,regulations list etc. Optionally the B2B user can compare the results (e.g. counts) of the simulation with the results on the reference dataset (i.e. the STANDALONE_SIMEX dataset itself).

(4) A simulation can be managed by NMOC or by B2B (or NOP) users. For a NMOC managed simulation, the typical workflow is:

- NMOC first starts and prepares a simulation.

- The B2B user then queries the simulations currently running in NM systems (via the `querySimulations` service request). The result contains the details of the different simulations and the datasets of those simulations.

- Then the B2B user can do counts, flightlist, regulation list, create or modify regulations or reroutings etc by specifying the dataset on which to do the requested operation.

(5) For a simulation managed by the B2B client:

- The B2B client first queries the available simulations that can be used to start a simulation. (via `queryAvailableSimulations`). There is only a limited pool of user managed simulations available.

- If there are still simulations available, the B2B user Starts a new simulation on the desired reference : so either a simulation on OPERATIONAL or on FORECAST or or a STANDALONE_SIMEX.

- Once the user is done with the simulation, the simulation needs to be stopped (via `stopSimulation`) to make it available again for the next `startSimulation` request.

(6) If for a simulation, the `simulationPeriod` (i.e. The period that is being simulated) is in the past, then counts, flightlist,etc can no longer retrieve any data (as e.g. flightlist and count querying expects the queried date to be inside of the reference data set period). So the simulation needs to be stopped. See also `SimulationListRequest`.

3.5.2. Simulation List

3.5.2.1. SOAP

(1) The associated SOAP operation is:

```
SimulationListReply querySimulations(
    SimulationListRequest request
    )
```

3.5.2.2. `SimulationListRequest`

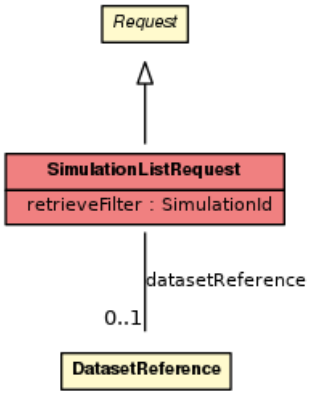

Figure 3.83. `SimulationListRequest` Class Diagram

(1) Request to query the published simulations.

(2) A simulation is basically a sandboxed environment containing some environment data, some traffic and a plan. It allows different actors to look at effects of a specific user scenario that is simulated and allows the different actors to interact with the simulation (for example: updating runway activations/sector configurations, regulations). In addition (selected) changes can be pushed back to the server.

(3) Simulations can be started via B2C and/or B2B and/or by NMOC.

(4) When an NMOC user starts a simulation, the NMOC user can decide to publish the simulation (i.e. make the simulation accessable via B2C and B2B). The simulation remains accessible via B2C and B2B until the simulation is unpublished or until the simulation is stopped.

(5) When a simulation is started via B2C and/or B2B, then that simulation is considered always published (accessible via B2C and B2B) until the simulation is stopped via B2C or B2B.

(6) A simulation typically is based on a reference: starting the simulation implies copying all the flight, regulation, environment, and tactical-updates related data into the simulation from a reference dataset: typically operational or forecast. This is the reference from which the data has been loaded. The user can then, after having created or modified some regulations or reroutings, compare the results with the data in the reference dataset.

(7) There is a limited number of simulation engines available for the users to start and stop. The user needs to stop a simulation after having started it, in order to make the simulation engine available again for the users.

(8) A simulation only support processing one request in parallel. So the B2B user should serialise the simulation requests (e.g. counts and flightlist and regulation creations and...).

(9) See also general text on simulations.

(10) Inherits from: `Request`

(11) Attributes:

a) **`SimulationId` `retrieveFilter`** *(Optional)*
Retrieves the simulation corresponding to a specific simulation. By default all simulations are returned.

b) **`DatasetReference` `datasetReference`** *(Optional)*
Retrieves the simulation corresponding to a specific reference data set. By default all simulations are returned.

3.5.2.3. `SimulationListReply`

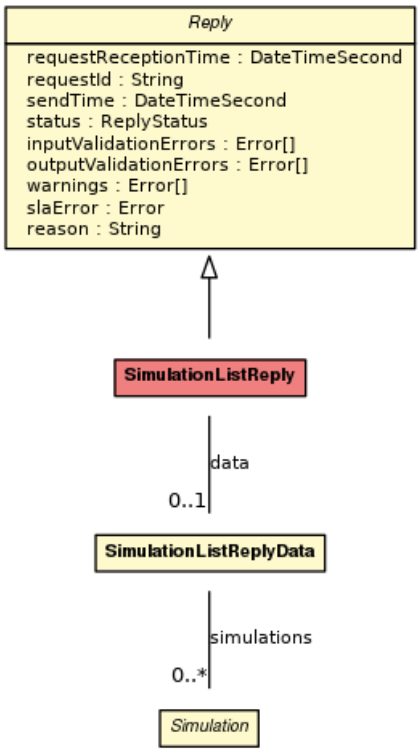

Figure 3.84. `SimulationListReply` Class Diagram

(1) Reply returned in response to SimulationListRequest.

(2) Inherits from: `Reply`

(3) Attributes:

a) **`Set<Simulation> simulations`** *(Optional)*
Set of simulations that matched the SimulationListRequest criteria.
Can be empty (meaning that no simulation matched the criteria).
Constraint: Size must be comprised between 0 and ∞.

3.5.3. Simulation Availability List

3.5.3.1. SOAP

(1) The associated SOAP operation is:

```
SimulationAvailabilityReply queryAvailableSimulations(
    SimulationAvailabilityRequest request
    )
```

3.5.3.2. `SimulationAvailabilityRequest`

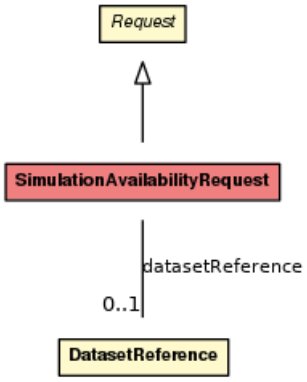

Figure 3.85. `SimulationAvailabilityRequest` Class Diagram

(1) Request to query the available (user managed) simulation engines that can be used to start a simulation.

(2) Inherits from: `Request`

(3) Attributes:

a) **`DatasetReference` `datasetReference`** *(Optional)*
Retrieves the available user managed simulations corresponding to a specific reference data set.
By default all available user_managed simulations are returned

3.5.3.3. `SimulationAvailabilityReply`

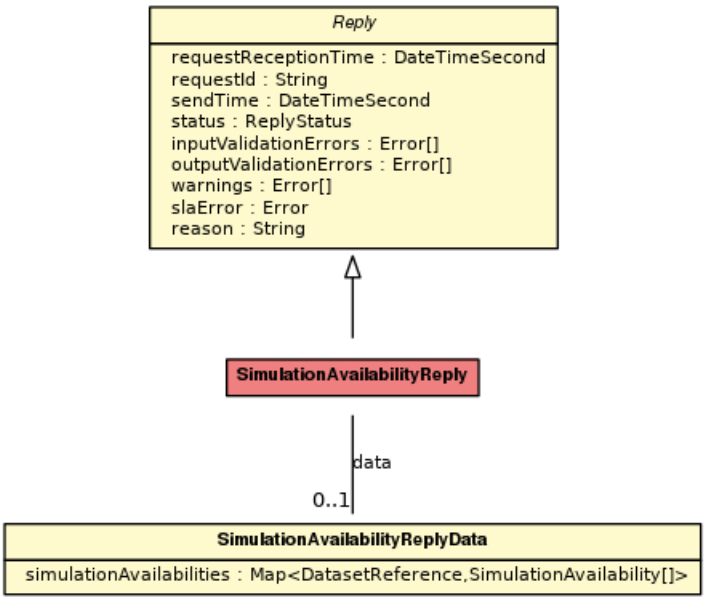

Figure 3.86. `SimulationAvailabilityReply` Class Diagram

(1) Reply returned in response to SimulationAvailabilityRequest.

(2) Inherits from: `Reply`

(3) Attributes:

a) **`Map<DatasetReference,Set<SimulationAvailability>> simulationAvailabilities`** *(Optional)*
The simulation availability per reference dataset.
Constraints:

i) Size must be comprised between 0 and ∞.

ii) Item size must be comprised between 0 and ∞.

3.5.4. Simulation Start

3.5.4.1. SOAP

(1) The associated SOAP operation is:

```
SimulationStartReply startSimulation(
    SimulationStartRequest request
    )
```

3.5.4.2. `SimulationStartRequest`

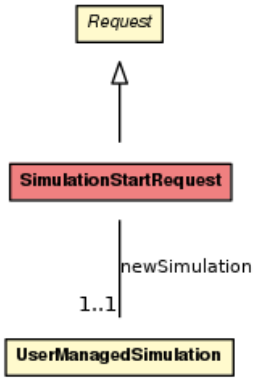

Figure 3.87. `SimulationStartRequest` Class Diagram

(1) Request to start a UserManagedSimulation based on a reference.

(2) All environment (CACD) data, tactical updates, flights and measures are initially loaded from the reference.

(3) Note that, once the flights have been loaded into the simulation, the fligh‌s are not updated , except for timers related events (like terminating the flights when they should have landed) and actions done in the simlation(like creating or modifying regulations or rerouting). So new radar info or flightUpdate related info is not processed in the simulation.

(4) Note that starting the simulation and loading the flights can take some time. (typically around 30 seconds to 1 minute)

(5) Note that when the user is done with the simulation, the user needs to use the StopSimulation service to free the simulation again.

(6) Inherits from: `Request`

(7) Attributes:

a) **`UserManagedSimulation` `newSimulation`** *(Mandatory)*
The simulation to start.
The reference dataset shall indicate the reference for the new simulation.

3.5.4.3. `SimulationStartReply`

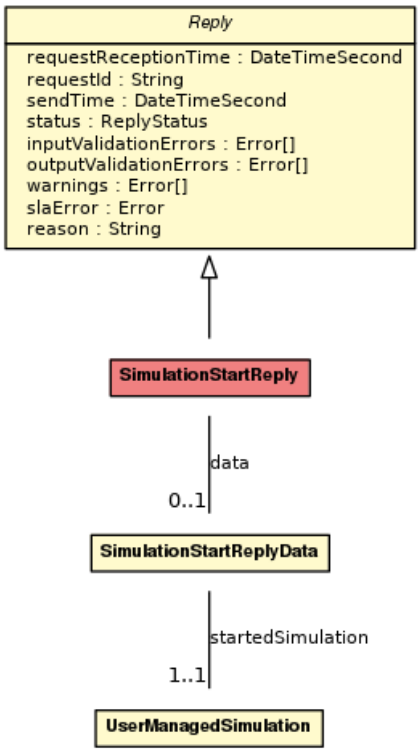

Figure 3.88. `SimulationStartReply` Class Diagram

(1) Reply returned in response to SimulationStartRequest.

(2) Inherits from: `Reply`

(3) Attributes:

a) **`UserManagedSimulation` `startedSimulation`** *(Mandatory)*

3.5.5. Simulation Stop

3.5.5.1. SOAP

(1) The associated SOAP operation is:

```
SimulationStopReply stopSimulation(
    SimulationStopRequest request
    )
```

3.5.5.2. `SimulationStopRequest`

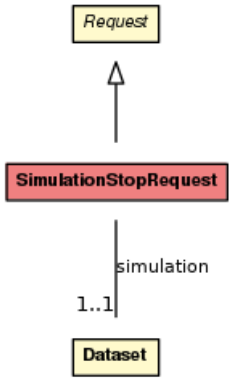

Figure 3.89. `SimulationStopRequest` Class Diagram

(1) Request to stop a user_managed_simulation. This will free the simulation and make it available to start again.

(2) Note that stopping a simulation can take some time (typically around 1 or 2 minutes).

(3) Inherits from: `Request`

(4) Attributes:

a) **`Dataset` `simulation`** *(Mandatory)*
The dataset of the UserManagedSimulation that needs to be stopped.
Constraint: See `INVALID_SIMULATION_TYPE`

(5) Constraint:

a)

Name	`INVALID_SIMULATION_TYPE`
Attribute	`simulation`
Description	The `simulationType` needs to be set to `UserManagedSimulation` in stop Simulation.

3.5.5.3. `SimulationStopReply`

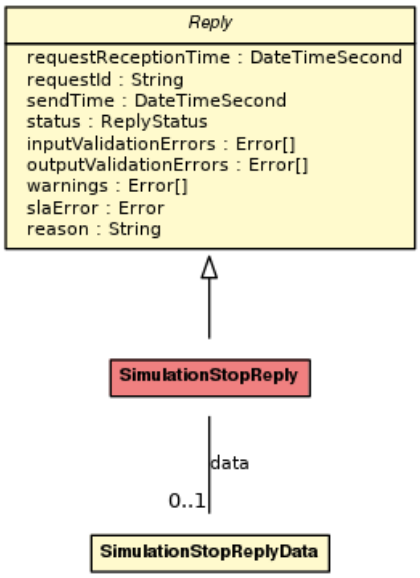

Figure 3.90. `SimulationStopReply` Class Diagram

(1) Reply returned in response to SimulationStopRequest.

(2) Inherits from: `Reply`

3.6. Scenario Repository Port Type

3.6.1. Overview

3.6.1.1. Introduction

(1) This service is intended to provide querying and update capabilities on the ScenarioRepository. The requests currently available are:

a) ScenarioRegulationRetrievalRequest / ScenarioRegulationRetrievalReply

b) ScenarioReroutingRetrievalRequest / ScenarioReroutingRetrievalReply

c) ScenarioMCDMOnlyRetrievalRequest / ScenarioMCDMOnlyRetrievalReply

d) ScenarioListRequest / ScenarioListReply

3.6.1.2. Scenario Repository

(1) The scenario repository contains pre-defined (and pre-agreed) solutions to handle an overload: scenarios.

(2) Typically the solutions make the flight reroute horizontally or vertically to avoid a congested airspace.

(3) To handle an overload an FMP can decide to either regulate the concerned traffic volume(s) or the FMP can implement one or more FL or RR type scenarios.

(4) Those scenarios contain a zero rate suspending regulation "forcing" the flights to reroute outside of the congested airspace.
Those scenarios also contain a rerouting. The rerouting allows to evaluate the impact of applying the 0 rate suspending regulation: applying the rerouting shows how the flights are expected to fly.

(5) Note it is the Airspace user that decides how he will file to avoid the congested airspace.
The rerouting gives a possible way to reroute: it represents the most likely or most efficient way for the flights to reroute.

(6) STAM scenario typically contain only cherry picked rerouting. When applied, the Airspace User receives from NM systems an RRN message indicating that airspace user should reroute. In this STAM case, the flights are not suspended.

(7) Note that a scenario can have multiple measures : for example a contingency scenario contains multiple measures (on different traffic volumes) that can be used to handle the contingency. When such a scenario is applied one or more measures of the scenario are applied (but not necessarily all measures of the scenario).

(8) The scenario from the scenario repository are published to the external users (AO, FMP,..) via the public portal and via B2B.

(9) The scenario allow airspace users to have an idea how to refile and it allows FMP to know where the typical on-load and off-load areas are in case he scenario is applied.

(10) The B2B user can query, via flightlist and traffic count, the applicableScenarios for a problem traffic volume (e.g. `"overloaded"`).
The result will include all the different scenarios (on different traffic volumes) that can off-load the problem traffic volume.

(11) The `queryScenarioRepository` service request can be used to retrieve the details about the scenarios.

(12) The `retrieveRegulationsFromScenario` and `retrieveReroutingsFromScenario` service requests can then be used to retrieve the applicable measures themselves from these scenarios.

(13) The scenario regulations can then be applied via B2B in:

- A NM simulation (to evaluate the impact) via the `createRegulation` service request for a simulation dataset
- or can be requested for implementation in FORECAST or in OPERATIONAL via the `fileRegulationProposal` service request.

(14) The scenario rerouting can then be applied via B2B in:

- A NM simulation (to evaluate the impact) via the `createRerouting` service request for a simulation dataset.
- FORECAST or in OPERATIONAL via `createRerouting` service: only supported for cherry picked rerouting (e.g. STAM type scenario) with a scenario reference pointing to an existing scenario measure Note that in that case, the traffic volume of the rerouting needs to match the traffic volume of the scenario measure.

(15) Note that in specific trial contexts, NM test systems can be configured to also accept creating any type of rerouting in FORECAST or OPERATIONAL.

(16) Note that the scenario repository is shared between FORECAST and OPERATIONAL. So when querying the scenario repository, it will return the same result for FORECAST or OPERATIONAL or any simulation that has as reference forecast or operational.

3.6.2. Retrieve list of Regulations from Scenario

3.6.2.1. SOAP

(1) The associated SOAP operation is:

```
ScenarioRegulationRetrievalReply retrieveRegulationsFromScenario(
    ScenarioRegulationRetrievalRequest request
    )
```

3.6.2.2. `ScenarioRegulationRetrievalRequest`

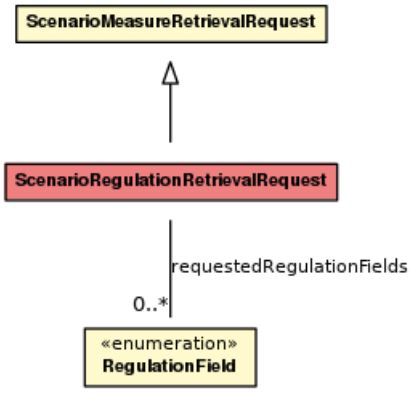

Figure 3.91. `ScenarioRegulationRetrievalRequest` Class Diagram

(1) Request to retrieve the regulation(s) contained in scenario from the scenario repository.

(2) Inherits from: `ScenarioMeasureRetrievalRequest`

(3) Attributes:

a) **Set<RegulationField> requestedRegulationFields** *(Mandatory)*
The reply returns only the requested regulation fields in this set, and only if the values of these requested fields are available at NM. Note that the regulation id is always returned.
Constraint: Size must be comprised between 0 and 24.

3.6.2.3. ScenarioRegulationRetrievalReply

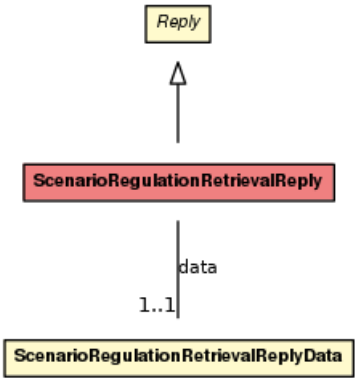

Figure 3.92. ScenarioRegulationRetrievalReply Class Diagram

(1) Reply returned in response to ScenarioRegulationRetrievalRequest.

(2) Inherits from: Reply

(3) Attributes:

a) **Set<Regulation> regulations** *(Mandatory)*
Constraint: Size must be comprised between 0 and ∞.

3.6.3. Retrieve list of Reroutings from Scenario

3.6.3.1. SOAP

(1) The associated SOAP operation is:

```
ScenarioReroutingRetrievalReply retrieveReroutingsFromScenario(
    ScenarioReroutingRetrievalRequest request
    )
```

3.6.3.2. `ScenarioReroutingRetrievalRequest`

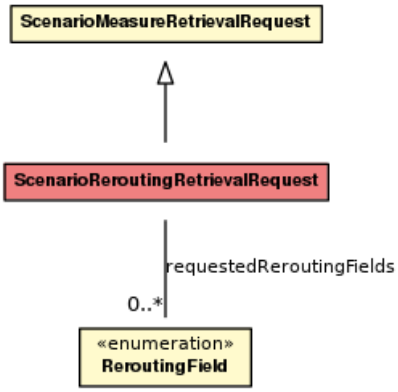

Figure 3.93. `ScenarioReroutingRetrievalRequest` Class Diagram

(1) Request to retrieve rerouting contained in scenario from the scenario repository.

(2) Inherits from: `ScenarioMeasureRetrievalRequest`

(3) Attributes:

a) **`Set<ReroutingField>` `requestedReroutingFields`** *(Mandatory)*
The reply returns only the requested rerouting fields in this set, and only if the values of these requested fields are available at NM. Note that the regulation id is always returned.
Constraint: Size must be comprised between 0 and 18.

3.6.3.3. `ScenarioReroutingRetrievalReply`

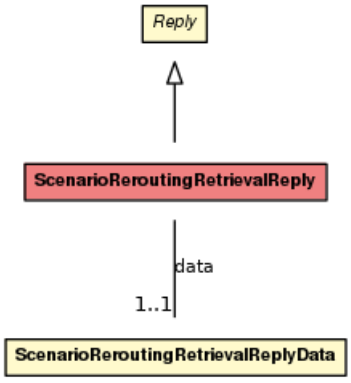

Figure 3.94. `ScenarioReroutingRetrievalReply` Class Diagram

(1) Reply to a ScenarioReroutingRetrievalRequest.

(2) Inherits from: `Reply`

(3) Attributes:

a) **Set<Rerouting> reroutings** *(Mandatory)*
Constraint: Size must be comprised between 0 and ∞.

3.6.4. Retrieve list of MCDMOnly from Scenario

3.6.4.1. SOAP

(1) The associated SOAP operation is:

```
ScenarioMCDMOnlyRetrievalReply retrieveMCDMOnlyFromScenario(
    ScenarioMCDMOnlyRetrievalRequest request
    )
```

3.6.4.2. `ScenarioMCDMOnlyRetrievalRequest`

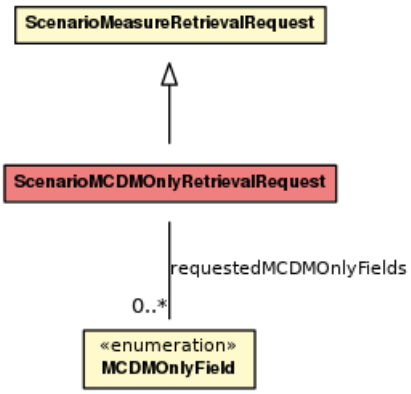

Figure 3.95. `ScenarioMCDMOnlyRetrievalRequest` Class Diagram

(1) Request to retrieve the MCDM-only measures contained in scenario from the scenario repository (See also MCDMOnlyRetrievalRequest).

(2) Inherits from: `ScenarioMeasureRetrievalRequest`

(3) Attributes:

a) **Set<MCDMOnlyField> requestedMCDMOnlyFields** *(Mandatory)*
The reply returns only the requested MCDM only fields in this set, and only if the values of these requested fields are available at NM.
Constraint: Size must be comprised between 0 and 24.

3.6.4.3. `ScenarioMCDMOnlyRetrievalReply`

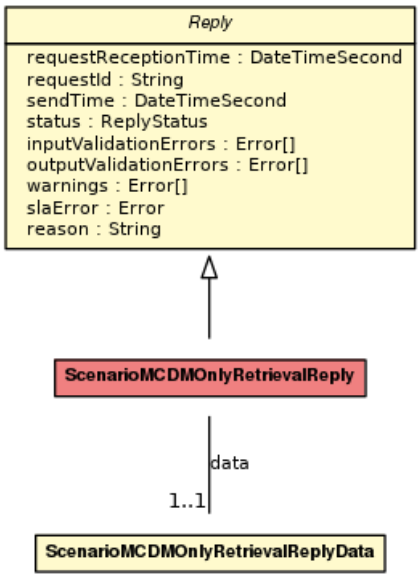

Figure 3.96. `ScenarioMCDMOnlyRetrievalReply` Class Diagram

(1) Reply to the scenario MCDM only retrieval.

(2) Inherits from: `Reply`

(3) Attributes:

a) **`Set<MCDMOnly> mcdmOnlySet`** *(Mandatory)*
Constraint: Size must be comprised between 0 and ∞.

3.6.5. List of Scenarios

3.6.5.1. SOAP

(1) The associated SOAP operation is:

```
ScenarioListReply queryScenarioRepository(
    ScenarioListRequest request
    )
```

3.6.5.2. `ScenarioListRequest`

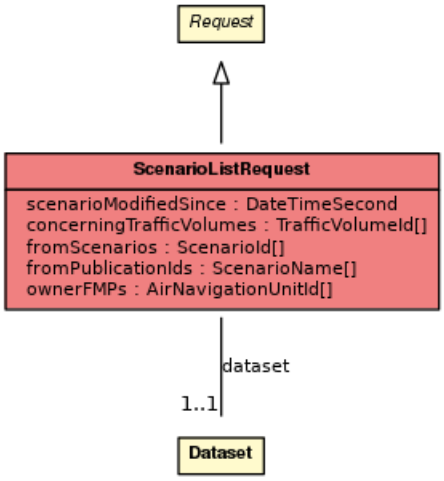

Figure 3.97. `ScenarioListRequest` Class Diagram

(1) Request to query the scenario and the scenario attributes from the scenario repository.

(2) Inherits from: `Request`

(3) Attributes:

a) **`Dataset` `dataset`** *(Mandatory)*
Dataset for which the scenario list is requested. See Forecast and Operational Datasets.

b) **`DateTimeSecond` `scenarioModifiedSince`** *(Optional)*
Selects scenario only that have been modified since this time.

c) **`Set<TrafficVolumeId>` `concerningTrafficVolumes`** *(Optional)*
Selects scenario only with a traffic volume that matches an entry in this set.
Constraint: Size must be comprised between 0 and ∞.

d) **`Set<ScenarioId>` `fromScenarios`** *(Optional)*
Selects only scenario that matches an entry in this set.
Constraint: Size must be comprised between 0 and ∞.

e) **`Set<ScenarioName>` `fromPublicationIds`** *(Optional)*
Selects only scenario that matches an entry in this set.
Constraint: Size must be comprised between 0 and ∞.

f) **`Set<AirNavigationUnitId>` `ownerFMPs`** *(Optional)*
Selects scenario only with a owner (FMP) that matches an entry in this set.
Constraint: Size must be comprised between 0 and ∞.

3.6.5.3. `ScenarioListReply`

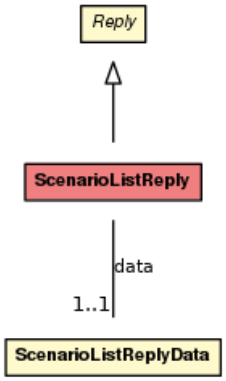

Figure 3.98. `ScenarioListReply` Class Diagram

(1) Reply to the query scenario attributes from the scenario repository.

(2) Inherits from: `Reply`

(3) Attributes:

a) **`Set<ScenarioAttributes> attributes`** *(Optional)*
List of scenario attributes.
Constraint: Size must be comprised between 0 and ∞.

Chapter 4. Data Types

4.1. AerodromeLocation

(1) Aerodrome location.

(2) Inherits from: Location.

4.2. AirspaceLocation

(1) Airspace location.

(2) Inherits from: Location.

4.3. AllFlightsLocation

(1) All flights location.

(2) Inherits from: Location.

4.4. ANDedReroutingConstraints

(1) Describes a set of rerouting constraints that are ANDed together.

(2) Attributes:

a) **Set<ReroutingConstraint> constraints** *(Mandatory)*
A list of ordered rerouting constraints that apply to the reroutings (can be empty).
Constraint: Size must be comprised between 0 and ∞.

(3) Used by: ReroutingSourcesAndConstraints.

4.5. AoLocation

(1) Aircraft operator location.

(2) Inherits from: Location.

4.6. ATFCMSituationCounts

(1) ATFCM situation counts.

(2) Attributes:

a) **CountsValue nrLandedTraffic** *(Mandatory)*
Number of flights which are landed.
Includes all flights whose status is either ATC terminated, TACT terminated, expecting FSA, or TACT terminated without expecting FSA.

b) **CountsValue nrAirborneTraffic** *(Mandatory)*
Number of flights which are airborne.
Includes all flights whose status is either ATC activated, TACT activated expecting FSA, or TACT activated without expecting FSA.

c) **CountsValue nrExpectedTraffic** *(Mandatory)*
Number of flights that have not taken off as of ATFCMSituationReply.lastUpdated.

d) **CountsValue nrFlightsUndefinedSlotCompliance** *(Mandatory)*
Includes all regulated and not suspended flights for which slot compliance information is not known.

e) **CountsValue nrFlightsDepBeforeSlot** *(Mandatory)*
Includes all regulated and not suspended flights whose ATOT is before (CTOT-5) minutes.

f) **CountsValue nrFlightsCompliedWithSlot** *(Mandatory)*
Includes all regulated and not suspended flights whose ATOT is either at or between (CTOT-5) and (CTOT+10) minutes.

g) **CountsValue nrFlightsDepAfterSlot** *(Mandatory)*
Includes all regulated and not suspended flights whose ATOT is after (CTOT+10) minutes.

h) **CountsValue suspendedFlightsDueToATFMMeasure** *(Mandatory)*
Number of flights which are suspended due to an ATFM measure.
Includes all suspended flights whose suspension status is either REGULATION_CONFIRMATION or TRAFFIC_VOLUMES_CONDITION.

i) **CountsValue suspendedFlightsDueToFAM** *(Mandatory)*
Number of flights which are suspended due to FAM.
Includes all suspended flights whose suspension status is NOT_REPORTED_AS_AIRBORNE.

j) **CountsValue flightsDelayedByMoreThan30Min** *(Mandatory)*
Number of flights which are delayed by more than 30 minutes.

(3) Used by: ATFCMSituationReply.

4.7. ATFCMSituationDelays

(1) ATFCM situation delays.

(2) Attributes:

a) **LongDurationHourMinute enRouteDelay** *(Mandatory)*
Includes all regulated and not suspended flights which are subject to a most penalising regulation whose reference location is not an airport or a set of airports.

b) **LongDurationHourMinute airportDelay** *(Mandatory)*
Includes all regulated and not suspended flights which are subject to a most penalising regulation whose reference location is an airport or a set of airports.

c) **Map<RegulationReason,LongDurationHourMinute> delaysPerRegulationReason** *(Mandatory)*
Total delays per regulation reasons.
Constraint: Size must be comprised between 1 and 15.

(3) Used by: ATFCMSituationReply.

4.8. ATFCMSituationRegulation

(1) Regulation in ATFCM situation.

(2) Attributes:

a) **RegulationId regulationId** *(Mandatory)*
Regulation unique identifier.

b) **DateTimeMinutePeriod period** *(Mandatory)*
Period.

c) **TrafficVolumeId trafficVolumeId** *(Mandatory)*
Traffic volume id.

d) **RegulationState regulationState** *(Mandatory)*
Regulation state.

e) **RegulationReason regulationReason** *(Mandatory)*
Regulation reason.

f) **LongDurationHourMinute delay** *(Mandatory)*
Total delay due to the regulation.

g) **CountsValue nrImpactedFlights** *(Mandatory)*
Number of flights which are impacted by the regulation.

h) **AerodromeICAOId protectedAerodrome** *(Optional)*
The protected aerodrome of the regulation.

(3) Used by: DeltaATFCMSituationRegulation, ATFCMSituationReply.

4.9. <<enumeration>> AvoidAirspaceReroutingKind

(1) Enumeration of possible Avoid Airspace Kinds.

(2) Values:

a) **HORIZONTAL**

b) **VERTICAL**

(3) Used by: AvoidViaAirspaceReroutingConstraint.

4.10. AvoidViaAirspaceReroutingConstraint

(1) Describes a manual rerouting constraint to avoid an airspace or to force the flights to cross an airspace. The AvoidViaAirspaceReroutingConstraint applies to both horizontal and vertical rerouting constraints.

(2) Inherits from: ReroutingConstraint.

(3) Attributes:

a) **AvoidViaReroutingType type** *(Mandatory)*
The kind of restriction.

b) **AirspaceId location** *(Mandatory)*
The airspace identifier.

c) **AvoidAirspaceReroutingKind avoidKind** *(Optional)*
If it concerns an avoid airspace rerouting constraint, indicates how the airspace is to be avoided (horizontally or vertically) If not present, then the default is used: if the vertical (or speed) rerouting source is used, then defaults to Vertical otherwise horizontal. Note the typical use case is: a vertical roerouting that sometimes has to change horizontally.

4.11. AvoidViaPointReroutingConstraint

(1) Describes a manual rerouting constraint to avoid a point or to force the flights to cross point. This type of constraint only applies for horizontal rerouting constraints.

(2) Inherits from: ReroutingConstraint.

(3) Attributes:

a) **AvoidViaReroutingType type** *(Mandatory)*
The kind of restriction.

b) **DBEOrPublishedPointId location** *(Mandatory)*
The point identifier.

4.12. <<enumeration>> AvoidViaReroutingType

(1) Enumerates the two possible AvoidVia rerouting constraint types: AVOID and VIA.

(2) Values:

a) **AVOID**

b) **VIA**

(3) Used by: AvoidViaPointReroutingConstraint, AvoidViaAirspaceReroutingConstraint.

4.13. typedef<int> Capacity

(1) Capacity on a traffic volume, expressed as a number of flights per hour.

(2) Range: `[0,998[`.

(3) Used by: PlannedCapacity.

4.14. CapacityPlans

(1) Capacity plans for one or more traffic volumes on a given day.

(2) A capacity plan is a special plan in the sense that, there can be cases where no capacity is defined in CACD. So there exist traffic volumes for which the capacity is not known at all or not known for some periods.

(3) In addition, regulation measures can overrule the capacity values defined (either in CACD or by the B2B update capacity service). This (optional) overruling regulation info can be found back in the `nmSchedule` attribute. However in the client schedule the non regulated capacities are maintained (just in case the regulation can be cancelled before the end of its applicability period).

(4) In a retrieval context, the plan is said to be 'complete' in the sense that it contains all the plan entries from all involved data sources (including NO_DATA data source in case no info is known).

(5) In an update context, the plan can be complete (if the B2B client designer decided to overwrite the default CACD values) or limited to the (full list) of (pre-)tactical updates with the gaps marked as AIRSPACE datasource (to obtain a complete time partition).

(6) In any case, periods in the time partition marked as AIRSPACE datasource correspond to removing any potential (pre-)tactical update and hence reset the corresponding values to the CACD definition for that period.

(7) Inherits from: `TacticalConfigurationPlan`.

(8) Attributes:

a) **Map<TrafficVolumeId,PlannedCapacities> tvCapacities** *(Mandatory)*
Constraints:

i) Size must be comprised between 0 and ∞.

ii) See INCOMPLETE_SCHEDULE

(9) Constraint:

a)

Name	INCOMPLETE_SCHEDULE
Attribute	tvCapacities
Description	tvCapacities has gaps and/or overlaps in the time partition or is not covering exactly one day.

(10) Used by: CapacityPlanRetrievalReply, CapacityPlanUpdateReply, CapacityPlanUpdateRequest.

4.15. CherryPickedLocation

(1) Cherry picked location.

(2) Inherits from: `Location`.

4.16. Counts

(1) Container for traffic counts values, for a given DateTimeMinutePeriod and TrafficType.

(2) Attributes:

a) **`CountsValue` totalCounts** *(Mandatory)*
Total traffic counts.

b) **Map<`FlowId`,`CountsValue`> flowCounts** *(Optional)*
Traffic counts by flow.
This attribute is set only when requested (i.e., `TrafficCountsByTrafficVolumeRequest.computeFlowCounts` has been set to `true`)
Note that if there are more than 50 flows (e.g. in case of scenario flow counts), then only the first 50 are shown here while TrafficCountsReply.flows does show them all.
Constraint: Size must be comprised between 0 and 50.

c) **Map<`SubTotalsTrafficCountsType`,`CountsValue`> subTotalsCounts** *(Optional)*
Sub total traffic counts.
This attribute is set only when requested (i.e., `TrafficCountsRequest.computeSubTotals` has been set to `true`)
Constraint: Size must be comprised between 7 and 7.

(3) Used by: TrafficCountsReplyData.

4.17. <<enumeration>> CountsCalculationType

(1) Enumeration of possible counts calculation type.

(2) Values:

a) **ENTRY**
Only flights taking off and/orlanding, or being over a point, or entering a sector in the `trafficWindow` are considered.

b) **OCCUPANCY**
Only flights occupying a sector or being airborne in the trafficWindow are considered.

(3) Used by: FlightListByAircraftOperatorRequest, LoadStateAtReferenceLocation, CountsCalculationTypeAndInterval, TrafficCountsByAircraftOperatorRequest, TrafficCountsByAirspaceRequest,

FlightListRequest, FlightListByTrafficVolumeRequest, FlightListByAirspaceRequest, TrafficCounts-ByTrafficVolumeRequest.

4.18. CountsCalculationTypeAndInterval

(1) .

(2) Attributes:

a) **CountsCalculationType calculationType** *(Mandatory)*
Indicates what is the calculation type of the count (entry or occupancy).

b) **DurationHourMinute duration** *(Mandatory)*
Specifies the duration used to calculate the counts (or the duration used to calculate the flightlist corresponding to those counts) : each count period, concerns a period of duration.

c) **DurationHourMinute step** *(Mandatory)*
Specifies the every x minutes a count need to be done: determines how many counts are returned for a requested period. It also determines the rounding of the trafficWindow.

(3) Used by: NetworkImpactAssessmentRetrievalReply.

4.19. CountsInterval

(1) Description what is the interval that is used to compute the counts / flight list.

(2) For ENTRY counts, the typical value is: show counts every 20 minutes and each count period has a duration of 60 minutes.

(3) For OCCUPANCY counts, the step is typically 1 minute while the duration can vary (depending on the location)

(4) Attributes:

a) **DurationHourMinute duration** *(Mandatory)*
Specifies the duration used to calculate the counts (or the duration used to calculate the flightlist corresponding to those counts) : each count period, concerns a period of duration. So if a user does a TrafficCounts request 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[`).
A duration of 1 minute, means that no extension of the trafficWindow is done.
Constraints:

i) See INVALID_COUNTS_INTERVAL

ii) See INVALID_DURATION_RANGE

b) **DurationHourMinute step** *(Mandatory)*
Specifies the every x minutes a count need to be done: determines how many counts are returned for a requested period. It also determines the rounding of the trafficWindow.

So if a user does a TrafficCounts request with a trafficWindow from `[09:59,11:20[` with a countsInterval with step 20 and duration 60 minutes, the 5 count periods returned are (`[09:40,10:40[`,`[10:00,11:00[`,`[10:20,11:20[`,`[10:40,11:40[`,`[11:00,12:00[`). This corresponds to an `effectiveTrafficWindow` of `[09:40, 12:00[`: flights from within this periods have been considered in the counts.
Note that the start of the count periods is rounded to a multiple of step.
Constraints:

i) See `INVALID_COUNTS_INTERVAL`

ii) See `INVALID_STEP_RANGE`

(5) Constraints:

a)

Name	`INVALID_DURATION_RANGE`
Attribute	`duration`
Description	The allowed `duration` value must be between 1 minute (`0001`) and 24 hours (`2400`).

b)

Name	`INVALID_STEP_RANGE`
Attribute	`step`
Description	The allowed `step` value must be between 1 minute (`0001`) and 24 hours (`2400`) and must be a divisor of 24 hours.

c)

Name	`INVALID_COUNTS_INTERVAL`
Attributes	`duration`, `step`
Description	The `duration` must be a multiple of `step`.

(6) Used by: FlightListByLocationRequest, TrafficCountsRequest.

4.20. typedef<int> CountsValue

(1) Range: `[0,100000]`.

(2) Used by: DeltaATFCMSituationRegulation, ScenarioImpact, SignificantDeltaCount, ATFCMSituationCounts, Counts, ATFCMSituationRegulation, DeltaCounts.

4.21. DatasetReference

(1) A simulation can be based on a reference (i.e. from which environment data has been copied and where initially the flights and measures have been copied from). This DataSetReference class represents these references.

(2) For a simulation based on a STANDALONE_SIMEX reference, then `type` must be STANDALONE_SIMEX and the `simulationId` must be value of `SimulationIdentifier.SimulationId`.

(3) For a simulation based on a OPERATIONAL/FORECAST reference, then `type` must be OPERATIONAL/FORECAST and the `simulationId` is not allowed.

(4) Attributes:

a) **`DatasetReferenceType` `type`** *(Mandatory)*
The type of reference dataset.

b) **`SimulationId` `simulationId`** *(Optional)*
The StandaloneSimex simulation identifier on which a UserManagedSimulation or a NMOCManagedSimulation is based
For a simulation based on a STANDALONE_SIMEX reference, the `simulationId` is value of `SimulationIdentifier.SimulationId`.
For a simulation based on a OPERATIONAL/FORECAST reference, the `simulationId` is not allowed.
Constraint: See `INVALID_SIMULATION_ID`

(5) Constraint:

a)

Name	`INVALID_SIMULATION_ID`
Attribute	`simulationId`
Description	Invalid Simulation Id. If the `type` is STANDALONE_SIMEX then `simulationId` should not be `null`.

(6) Used by: NMOCManagedSimulation, SimulationAvailabilityRequest, SimulationListRequest, SimulationAvailabilityReply, UserManagedSimulation.

4.22. <<strict enumeration>> DatasetReferenceType

(1) Possible reference dataset types for simulations.

(2) Values:

a) **`FORECAST`**

b) **`OPERATIONAL`**

c) **`STANDALONE_SIMEX`**

(3) Used by: DatasetReference.

4.23. DelayLocation

(1) Delay location.

(2) Inherits from: Location.

4.24. DeltaATFCMSituation

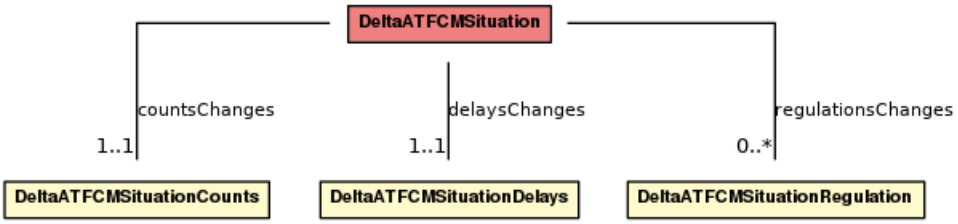

Figure 4.1. DeltaATFCMSituation Class Diagram

(1) .

(2) Attributes:

a) **DeltaATFCMSituationCounts countsChanges** *(Mandatory)*
ATFCM situation counts.

b) **DeltaATFCMSituationDelays delaysChanges** *(Mandatory)*
ATFCM situation delays.

c) **Set<DeltaATFCMSituationRegulation> regulationsChanges** *(Optional)*
ATFCM situation regulations.
Constraint: Size must be comprised between 0 and ∞.

(3) Used by: NetworkImpactAssessmentRetrievalReply.

4.25. DeltaATFCMSituationCounts

(1) Delta ATFCM situation counts.

(2) Attributes:

a) **SignificantDeltaCount nrLandedTraffic** *(Mandatory)*
Number of flights which are landed.
Includes all flights whose status is either ATC terminated, TACT terminated, expecting FSA, or TACT terminated without expecting FSA.

b) **SignificantDeltaCount nrAirborneTraffic** *(Mandatory)*
Number of flights which are airborne.

Includes all flights whose status is either ATC activated, TACT activated expecting FSA, or TACT activated without expecting FSA.

c) **SignificantDeltaCount nrExpectedTraffic** *(Mandatory)*
Number of flights that have not taken off as of ATFCMSituationReply.lastUpdated.

d) **SignificantDeltaCount nrFlightsUndefinedSlotCompliance** *(Mandatory)*
Includes all regulated and not suspended flights for which slot compliance information is not known.

e) **SignificantDeltaCount nrFlightsDepBeforeSlot** *(Mandatory)*
Includes all regulated and not suspended flights whose ATOT is before (CTOT-5) minutes.

f) **SignificantDeltaCount nrFlightsCompliedWithSlot** *(Mandatory)*
Includes all regulated and not suspended flights whose ATOT is either at or between (CTOT-5) and (CTOT+10) minutes.

g) **SignificantDeltaCount nrFlightsDepAfterSlot** *(Mandatory)*
Includes all regulated and not suspended flights whose ATOT is after (CTOT+10) minutes.

h) **SignificantDeltaCount suspendedFlightsDueToATFMMeasure** *(Mandatory)*
Number of flights which are suspended due to an ATFM measure.
Includes all suspended flights whose suspension status is either REGULATION_CONFIRMATION or TRAFFIC_VOLUMES_CONDITION.

i) **SignificantDeltaCount suspendedFlightsDueToFAM** *(Mandatory)*
Number of flights which are suspended due to FAM.
Includes all suspended flights whose suspension status is NOT_REPORTED_AS_AIRBORNE.

j) **SignificantDeltaCount flightsDelayedByMoreThan30Min** *(Mandatory)*
Number of flights which are delayed by more than 30 minutes.

(3) Used by: DeltaATFCMSituation.

4.26. DeltaATFCMSituationDelays

(1) Delta ATFCM situation delays.

(2) Attributes:

a) **SignificantDeltaDuration enRouteDelay** *(Mandatory)*
Includes all regulated and not suspended flights which are subject to a most penalising regulation whose reference location is not an airport or a set of airports.

b) **SignificantDeltaDuration airportDelay** *(Mandatory)*
Includes all regulated and not suspended flights which are subject to a most penalising regulation whose reference location is an airport or a set of airports.

c) **Map<RegulationReason,SignificantDeltaDuration> delaysPerRegulationReason** *(Mandatory)*
Total delays per regulation reasons.
Constraint: Size must be comprised between 1 and 15.

(3) Used by: DeltaATFCMSituation.

4.27. DeltaATFCMSituationRegulation

(1) Compares a before and after situation and indicates if the change is significant. An insignificant change is a change that is considered not relevant.

(2) Attributes:

a) **ATFCMSituationRegulation afterSituation** *(Mandatory)*

b) **LongDurationHourMinute delayBefore** *(Mandatory)*
Total delay due to the regulation before the change.

c) **CountsValue numberOfImpactedFlightsBefore** *(Mandatory)*
Number of flights which were impacted by the regulation before the change.

d) **boolean isSignificantChange** *(Mandatory)*

(3) Used by: DeltaATFCMSituation.

4.28. DeltaCounts

(1) Compares a before and after situation and represent the differences (counts and OTMV alerts) between the before and after situation for a certain TV with a CountsCalculationTypeAndInterval for a certain count period.

(2) Attributes:

a) **Set<OtmvAlert> otmvAlertsBefore** *(Mandatory)*
Constraint: Size must be comprised between 0 and ∞.

b) **Set<OtmvAlert> otmvAlertsAfter** *(Mandatory)*
Constraint: Size must be comprised between 0 and ∞.

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

d) **CountsValue[] countsAfter** *(Mandatory)*
Constraint: Size must be comprised between 0 and ∞.

(3) Used by: NetworkImpactAssessmentRetrievalReply.

4.29. DeltaLevelReroutingConstraint

(1) Describes a delta level rerouting constraint.

(2) Inherits from: RoutingConstraint.

(3) Attributes:

a) **DBEOrPublishedPointId pointId** *(Mandatory)*

b) **int deltaFlightLevel** *(Mandatory)*
Constraint: Range: [-998,998[.

4.30. ExcludeReIncludeFlightInRegulation

(1) Allows to exclude or re-include a flight in 1 or more regulations.

(2) Attributes:

a) **Set<RegulationId> excludeFromRegulations** *(Mandatory)*
Regulations to additionally exclude or re-include.
Constraint: Size must be comprised between 1 and 20.

b) **boolean flightProposal** *(Mandatory)*
Determines if the flight is to be excluded/re-included as a proposal or not.
Typically in simulations, one would exclude directly while in operational dataset context, one needs to submit a proposal for NMOC to review.

(3) Used by: UpdateFlightInMeasureChoice.

4.31. FlightAtfcmMcdmOnlyLocation

(1) Flight ATFCM MCDM only location.

(2) Inherits from: FlightAtfcmMeasureLocation.

(3) Attributes:

a) **RegulationId mcdmOnlyMeasureId** *(Mandatory)*
The id of the MCDM only measure.

4.32. <<abstract>> FlightAtfcmMeasureLocation

(1) Abstract ancestor for all flight atfcm measure locations.

(2) This type is equivalent to the FlightRegulationLocation but extended to cope with the different types of measure (regulation/rerouting/...)

(3) Attributes:

a) **ReferenceLocation referenceLocation** *(Optional)*
Reference location on which the measure applies.
Mandatory for regulation
Optional for rerouting : not present if there is no such reference location.

b) **MeasureSubType measureSubType** *(Mandatory)*

c) **HotspotId hotspotId** *(Optional)*
The Id of the problem hotspot associated to the measure.
Note that problem hotspot related fields are trial related (STAM) fields : they are only accessible/filled during specific trials or on test platforms.
In addition, the hotspot id is not immutable as it contains the applicability period and the applicability period of a hotspot can change depending on traffic evolution.

d) **MCDMState mcdmState** *(Optional)*
The mcdm state of the measure (null if this measure is not under MCDM)

(4) Extended by: FlightAtfcmRegulationLocation, FlightAtfcmReroutingLocation, FlightAtfcmMcdmOnlyLocation.

(5) Used by: Flight.

4.33. FlightAtfcmRegulationLocation

(1) Flight ATFCM regulation location.

(2) Inherits from: FlightAtfcmMeasureLocation.

(3) Attributes:

a) **RegulationId regulationId** *(Mandatory)*
The regulation identifier.

b) **boolean toConfirm** *(Mandatory)*
True if the regulation must still be confirmed.

4.34. FlightAtfcmReroutingLocation

(1) Flight ATFCM rerouting location. It contains info about what the type of rerouting and about the outcome of the rerouting (basically success or no alternative/better route found).

(2) Inherits from: FlightAtfcmMeasureLocation.

(3) Attributes:

a) **ReroutingId reroutingId** *(Mandatory)*
The rerouting unique identifier.

b) **ReroutingApplyKind reroutingApplyKind** *(Mandatory)*

c) **GroupReroutingIndicator groupReroutingIndicator** *(Mandatory)*

4.35. FlightHotspotLocation

(1) The hotspot is used to indicate a period when there is too much traffic in a traffic volume according to occupancy counts or complexity analysis for a specific occupancy traffic count duration.

(2) Attributes:

a) **Hotspot hotspot** *(Mandatory)*
A hotspot by which the flight is or was concerned.

b) **ReferenceLocation referenceLocation** *(Mandatory)*
Reference location of the traffic volume on which the hotspot applies.

(3) Used by: Flight.

4.36. FlightMCDMInfo

(1) Flight M-CDM (Measure Collaborative Decision Making)information: a flight can be concerned by 0 or more measures (regulation/rerouting/M-CDM only) involved in M-CDM.

(2) Attributes:

a) **MeasureId firstAssociatedMCDMMeasure** *(Mandatory)*
The first/most relevant measure that is impacting/penalizing the flight (somewhat equivalent to the most penalizing regulation but generalized to different types of measure)

b) **int nrAssociatedMCDMRegulations** *(Mandatory)*
Constraint: Range: `[0,∞[`.

c) **int nrAssociatedMCDMReroutings** *(Mandatory)*
Constraint: Range: `[0,∞[`.

d) **int nrAssociatedMCDMOnlyMeasures** *(Mandatory)*
Constraint: Range: `[0,∞[`.

e) **MCDMState worstMCDMState** *(Mandatory)*
The worst flight MCDM state of the different measures affecting the flight, if any.
Following values are possible and listed from the most worst state to the less worst state:

i) DRAFT

ii) PROPOSED

iii) INTERRUPTED

iv) COORDINATED

v) IMPLEMENTING

vi) IMPLEMENTED

vii) FINISHED
Basically the `worstMCDMState` gives the MCDM state of the least advanced MCDM measure impacting the flight.
`ABANDONED` MCDM states are not considered (except for the queryFlightsByMeasure service).

(3) Used by: Flight.

4.37. `FlightRegulationLocation`

(1) Describes the location of a regulation based on the reference location of the traffic volume to which the regulation applies.

(2) Attributes:

a) **`RegulationId` `regulationId`** *(Mandatory)*
Id of the regulation.

b) **`ReferenceLocation` `referenceLocation`** *(Mandatory)*
Reference location in the traffic volume on which the regulation applies.

c) **`boolean toConfirm`** *(Mandatory)*
True if the regulation must still be confirmed.

(3) Used by: Flight.

4.38. `Flow`

(1) Flow identification.

(2) Attributes:

a) **`FlowId` `id`** *(Mandatory)*
Flow id or the scenario traffic volume id (in case of scenario flow counts).

b) **`FlowType` `type`** *(Mandatory)*
Flow type.
Constraints:

i) See `FLOW_ROLE_MUST_BE_NULL`

ii) See `SCENARIO_IMPACT_AND_APPLICABLE_SCENARIO_CANNOT_BE_NULL`

c) **`FlowRoleSelection` `role`** *(Optional)*
Flow role selection.
Constraint: See `FLOW_ROLE_MUST_BE_NULL`

d) **`TrafficVolumeScenarios` `applicableScenarios`** *(Optional)*

Describes the applicable scenarios that have measures on the traffic volume corresponding to `id`.
Note that even scenario are returned that have no scenarioImpact (e.g. no flights for the selected period captured by the traffic volume).
This allows the B2B client to also use this field to query more generally the scenario repository: what are all the scenario impacting a traffic volume (not looking at specific dates or flights).
Constraint: See SCENARIO_IMPACT_AND_APPLICABLE_SCENARIO_CANNOT_BE_NULL

e) **ScenarioImpact scenarioImpact** *(Optional)*
Scenario impact.
Constraint: See SCENARIO_IMPACT_AND_APPLICABLE_SCENARIO_CANNOT_BE_NULL

(3) Constraints:

a)

Name	FLOW_ROLE_MUST_BE_NULL
Attributes	role, type
Description	The role must be null if type is ASSOCIATED.

b)

Name	SCENARIO_IMPACT_AND_APPLICABLE_SCENARIO_CANNOT_BE_NULL
Attributes	applicableScenarios, scenarioImpact, type
Description	The scenarioImpact and applicableScenario can not be null if type is SCENARIO and must be null otherwise.

(4) Used by: TrafficCountsReplyData.

4.39. typedef<string> FlowId

(1) Pattern: TEXT{1,8}

(2) Used by: FlightListByHotspotRequest, FlightListByTrafficVolumeRequest, Rerouting, Counts, Flow.

4.40. <<enumeration>> FlowRoleSelection

(1) The selection flow role (for FlowType.LINKED flows of traffic volume).

(2) Values:

a) **EXCLUDED**
Flights on EXCLUDED flows are not considered part of the traffic volume (so they are not taken into account in flight lists or counts or regulations on that traffic volume).

b) **EXEMPTED**

Flights on EXEMPTED flows are considered part of the traffic volume if there are no INCLUDED / INCLUDED_AND_EXEMPTED flows (so they are taken into account in flight lists or counts on that traffic volume). However, w.r.t. regulations and delay: the flights are exempted. This means only exceptional constraint regulations can impact the flight.

c) **INCLUDED**
Flights on INCLUDED flows are considered part of the traffic volume. If there are INCLUDED or INCLUDED_AND_EXEMPTED flows, then any flight that is not on an INCLUDED or INCLUDED_AND_EXEMPTED flow, is considered not part of the traffic volume.

d) **INCLUDED_AND_EXEMPTED**
Flights on an INCLUDED_AND_EXEMPTED flow, are considered part of the traffic volume. However hey are also considered as an INCLUDED flow (w.r.t. other flights that are not on an INCLUDED flow) and as an EXEMPTED flow (w.r.t. regulations).

(3) Used by: Flow.

4.41. <<enumeration>> FlowType

(1) Enumerates the possible flow types.

(2) Values:

a) **ASSOCIATED**
The counts correspond to the traffic of each one of the ASSOCIATED flows and the OHERS flow specified on the reference location that intersects the traffic inside the traffic volume excluding the traffic of the EXCLUDED flows. The returned counts is per ASSOCIATED and OTHERS flow.
Following possibilities exist:

i) Traffic volume without flows: the returned counts contain traffic counts for the traffic of each ASSOCIATED flow.

ii) Traffic volume with included flows: the returned counts contain traffic counts for the traffic of each ASSOCIATED flow intersecting with the traffic of the INCLUDED flows. The traffic counts are equal to the counts of the traffic intersection (common traffic) of each ASSOCIATED flow with the INCLUDED flows.

iii) Traffic volume with excluded flows: the returned counts contain traffic counts for the traffic of each ASSOCIATED flow and the OTHERS flow, excluding the traffic of the excluded flows. The traffic counts are equal to the counts of the traffic of each ASSOCIATED and OTHERS flow minus the traffic of the excluded flows.

iv) Traffic volume with included and excluded flows: the returned counts contain traffic counts for the traffic of each ASSOCIATED flow with traffic intersecting the traffic of the INCLUDED flows, excluding the traffic of the excluded flows. The traffic counts are equal to the counts of the traffic intersection (common traffic) of each ASSOCIATED flow with the INCLUDED flows minus the traffic of the excluded flows.

Note that it is possible that a traffic volume has no associated nor linked flows.

b) **LINKED**
The counts correspond to the traffic inside each one of the INCLUDED, EXEMPTED and INCLUDED_AND_EXEMPTED flows that are specified on the traffic volume. The counts are per INCLUDED, EXEMPTED and INCLUDED_AND_EXEMPTED flow.

c) **SCENARIO**
When querying flowcounts for scenario type flows, the flow counts show one flow per scenario traffic volume. This allows to show for a trafficvolume, which scenarios can be applied to off-load that traffic volume and by how much (inside the scenario flow count numbers)
In this case, the flow counts correspond to the common traffic (intersection) between

i) the traffic inside the traffic volume (excluding the traffic of the EXCLUDED flows)

ii) the traffic on the traffic volumes of the applicable scenarios and their measures
See also Class TrafficVolumeScenarios and general into inside scenario repository.
Querying the scenario flow counts is a heavy field for the NM systems.

(3) Used by: Flow, TrafficCountsByTrafficVolumeRequest.

4.42. ForceFlightInRegulation

(1) Flight in regulation.

(2) Attributes:

a) **RegulationId regulationId** *(Mandatory)*
The regulation unique identifier.

b) **DateTimeMinute newCto** *(Optional)*
The new calculated time over the regulation.
Constraints:

i) See BOTH_NEW_CTO_AND_CTOT_MUST_BE_SET

ii) See EITHER_NEW_CTO_OR_CTOT_MUST_BE_SET

c) **DateTimeMinute newCtot** *(Optional)*
The new calculated take-off time.
Constraints:

i) See BOTH_NEW_CTO_AND_CTOT_MUST_BE_SET

ii) See EITHER_NEW_CTO_OR_CTOT_MUST_BE_SET

d) **boolean flightProposal** *(Mandatory)*
Determines if this Force action needs to be done on the normal flight or if a proposal flight needs to be created.

Typically in simulations, one would force directly while in operational dataset context, one needs to submit a proposal for NMOC to review: in the context of a proposal cherry picked regulation (See `fileRegulationProposal` service) or in the context of modifying the CTOT of a flight in a normal (non-cherry-picked) regulation.

e) **`boolean immediatelyUnforce`** *(Mandatory)*
Determines if this flight needs to be unforced after the forcing (i.e. to allow improvements/deteriorations).

(3) Constraints:

a)

Name	`EITHER_NEW_CTO_OR_CTOT_MUST_BE_SET`
Attributes	`newCto`, `newCtot`
Context	`UpdateFlightsInMeasureRequest`
Description	Only one of either `newCto` or `newCtot` attribute must be set.

b)

Name	`BOTH_NEW_CTO_AND_CTOT_MUST_BE_SET`
Attributes	`newCto`, `newCtot`
Context	`UpdateFlightsInMeasureReply`
Description	Both `newCto` and `newCtot` attributes must be set.

(4) Used by: UpdateFlightInMeasureChoice.

4.43. <<enumeration>> FreezeTP

(1) Enumeration of possible freeze terminal procedure.

(2) Values:

a) **`FREEZE_SID_STAR`**

b) **`FREEZE_SID_STAR_AND_CONNECTING_POINTS`**

c) **`NO`**

(3) Used by: ReroutingHorizontalConstraints.

4.44. <<enumeration>> GroupReroutingIndicator

(1) Group rerouting indicator.

(2) Values:

Value	Description
EXECUTED	Rerouting executed and new route stored in the RTFM of the flight
INTERESTING	Group rerouting for indication attempted and an alternative route found
NO_REROUTING	No rerouting attempted or no rerouting executed
OPPORTUNITY	Group rerouting for opportunity attempted and an opportunity found
UNINTERESTING	No rerouting proposal

Table 4.1. <<enumeration>> GroupReroutingIndicator

(3) Used by: FlightAtfcmReroutingLocation.

4.45. Hotspot

(1) The hotspot is used to indicate a period when there is too much traffic in a traffic volume according to occupancy counts or complexity analysis for a specific occupancy traffic count duration.

(2) Attributes:

a) **HotspotId hotspotId** *(Mandatory)*
The identifier of the hotspot.
Note that hotspot id is not immutable as it contains the applicability period and the applicability period of a hotspot can change depending on traffic evolution.

b) **HotspotSeverity severity** *(Mandatory)*
The severity of the hotspot.

c) **HotspotStatus status** *(Mandatory)*
The status of the hotspot.

d) **string remark** *(Optional)*
Free text used for remarks and notes
Constraint: Pattern: `(ALPHA|DIGIT|'|\(|\)|+|,|=|?|.|/|:| ){0,255}`

e) **string trafficVolumeDescription** *(Contextual)*
The traffic volume description to which the hotspot applies.
Presence:

i) Must be null in HotspotPlanUpdateRequest

ii) Optional otherwise.

(3) Used by: HotspotPlans, FlightHotspotLocation.

4.46. <<enumeration>> HotspotField

(1) Enumeration of possible severity levels for a hotspot.

(2) Values:

a) **DIARY**

b) **MEASURE**

c) **SEVERITY**

d) **STATUS**

4.47. HotspotId

(1) The identifier of an hotspot.

(2) Attributes:

a) **DateTimeMinutePeriod applicabilityPeriod** *(Mandatory)*
The applicability period of the hotspot.
The applicability period of a hotspot indicates that in the corresponding occupancy traffic counts (`step` = 1 minute, `duration` of the `HotspotId`), for the count periods/intervals that start in the `applicabilityPeriod` there is a capacity/complexity problem.

b) **TrafficVolumeId trafficVolume** *(Mandatory)*
Traffic volume to which this hotspot applies.

c) **DurationHourMinute duration** *(Mandatory)*
The duration of this hotspot.

(3) Used by: MCDMTopicListRequestSelector, FlightListByHotspotRequest, Measure, Hotspot, FlightAtfcmMeasureLocation, MCDMHotspotTopic, MCDMStatefulTopic.

4.48. <<enumeration>> HotspotKind

(1) Identifies the kind of hotspot.

(2) Values:

a) **LOCATION_OF_INTEREST**
Location of interest hotspots are used raise awarness about potential hotspots or problems in the context of the daily plan: For example weather or special events (a.o. Soyuz rocket launches that imply a closure of some airspaces for a period of time).

b) **PROBLEM**
Problem hotspots are STAM related hotspots and are linked to a demand-capacity inbalance. Typically a Location of interest hotspot can evolve into a problem hotspot if the potential risk metrializes and introduces a real demand-capacity inbalance for a specific period.

(3) Used by: HotspotPlans, HotspotListRequest.

4.49. HotspotPlans

(1) Hotspot plan for a given traffic volume and day.

(2) Attributes:

a) **PlanDataId dataId** *(Mandatory)*
Opaque identifier representing the version of this hotspot plan.

b) **Dataset dataset** *(Mandatory)*
Dataset to which the hotspot plan belongs.

c) **DateYearMonthDay day** *(Mandatory)*
Day for which this hotspot plan is valid.

d) **boolean planTransferred** *(Optional)*
Indicates if the plan has been transferred to the OPERATIONAL dataset. When false, it means that the most up-to-date data can be found in the FORECAST dataset.

e) **boolean planCutOffReached** *(Optional)*
Indicates if the plan can still be updated in the FORECAST dataset, i.e. if the forecast cut-off time has been reached or not.

f) **HotspotKind hotspotKind** *(Mandatory)*
The kind of hotspot.

g) **Map<TrafficVolumeId,Map<DurationHourMinute,Set<Hotspot>>> schedules** *(Mandatory)*
(Pre-)tactical hotspots associated to their applicability period for a set of specific traffic volumes and duration. The schedule exposes the complete time partition of the hotspots for the day. Missing periods in the time partition correspond to not having any hotspot for that period.
Constraints:

i) Size must be comprised between 0 and 1000.

ii) Item size must be comprised between 0 and 1000.

iii) Item size must be comprised between 0 and 1000.

iv) See INVALID_SCHEDULE

v) See ONLY_ONE_ENTRY_CAN_BE_UPDATED_IN_SCHEDULE

(3) Constraints:

a)

Name	INVALID_SCHEDULE
Attribute	schedules

Description	The duration key used in each of the `schedules` map attribute has to be equal to the duration of all hotspots linked to that duration key.

b)

Name	`ONLY_ONE_ENTRY_CAN_BE_UPDATED_IN_SCHEDULE`
Attribute	`schedules`
Context	`HotspotPlanUpdateRequest`
Description	Only one entry in each of the `schedules` map attribute can be updated (i.e., for one duration).

(4) Used by: HotspotPlanUpdateRequest, HotspotListReply, HotspotPlanUpdateReply.

4.50. <<enumeration>> HotspotSeverity

(1) Enumeration of possible severity levels for a hotspot.

(2) Values:

a) **HIGH**

b) **LOW**

c) **MEDIUM**

(3) Used by: Hotspot.

4.51. <<enumeration>> HotspotStatus

(1) Enumeration of possible severity levels for a hotspot.

(2) Values:

a) **ACCEPTABLE**
After analysis it appeared that the problem was not a real problem and does not require action to be solved. Note that if a Hotspot is really no longer required it is simply removed completely.
The ACCEPTABLE state is solely intended to indicate that a Hotspot has been accepted as a problem that can be tolerated (e.g. no resolution action such as a STAM needs to be applied).

b) **ACTIVE**
The problem is confirmed and must be solved (e.g. by a STAM)

c) **DRAFT**
A potential problem (peak) has been identified and it is still needed to verify if it is a real problem or not.

d) **SOLVED**
The problem has been solved (e.g. by a STAM)

(3) Used by: Hotspot.

4.52. <<enumeration>> KindOfRestriction

(1) Enumeration of possible restriction location.

(2) Values:

a) **AC**

b) **CF**

c) **DCT**

d) **RAD**

e) **SC**

(3) Used by: RestrictionLocation.

4.53. LevelAndSpeedReroutingConstraint

(1) Describes a level and speed rerouting constraint.

(2) Inherits from: `ReroutingConstraint`.

(3) Attributes:

a) **DBEOrPublishedPointId fromPoint** *(Optional)*
From this point onwards the level and/or speed needs to be adapted.
If not present, the level and/or speed is updated from the departure aerodrome.

b) **DBEOrPublishedPointId untilPoint** *(Optional)*
The level and/or speed needs to be adapted until untilPoint.
If not present, the level and/or speed is updated until the arrival aerodrome or until the next flightlevel constraint.

c) **FlightLevel level** *(Mandatory)*

d) **AirSpeed airSpeed** *(Optional)*
If not present, then the speed of the flights is not updated.

4.54. <<enumeration>> LevelConstraintKind

(1)

(2) Values:

a) **ABOVE**

b) **BELOW**

(3) Used by: ScenarioLevelConstraint.

4.55. LifeCycleEvent

(1) Describes when and by whom an object has been last updated.

(2) Attributes:

a) **DateTimeSecond eventTime** *(Mandatory)*
The time at which the last update was done (by a user explicitly or by the system).
If this is the first event for this measure, then it is the time when the measure was created

b) **DateTimeSecond userUpdateEventTime** *(Optional)*
If the `eventTime` is the event time of a system triggered event, then the `userUpdateEventTime` indicates the time at which the last user triggered update was done.
If this is the first event for this measure, then it is the time when the measure was created.

c) **LifeCycleEventType userUpdateType** *(Mandatory)*
The update type of the last user update.

d) **UserId userId** *(Mandatory)*
Id of the user who created, last updated or deleted the object.

(3) Used by: Measure.

4.56. <<enumeration>> LifeCycleEventType

(1) Lifecycle event type.

(2) Values:

a) **CREATION**

b) **DELETION**
Note that `DELETION` for a regulation in fact means that the regulation was cancelled.

c) **UPDATE**

(3) Used by: LifeCycleEvent.

4.57. <<abstract>> Location

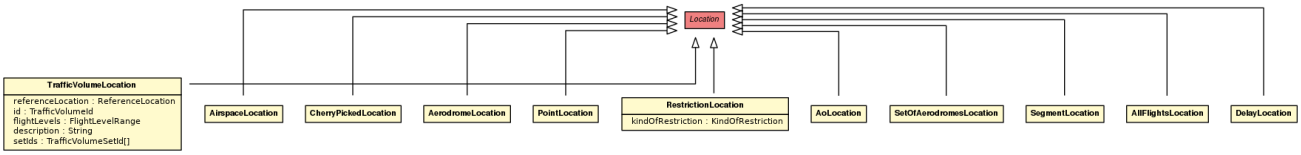

Figure 4.2. <<abstract>> Location Class Diagram

(1) Ancestor structure for all locations.

(2) Extended by: AirspaceLocation, SegmentLocation, AerodromeLocation, DelayLocation, AllFlights-Location, SetOfAerodromesLocation, RestrictionLocation, PointLocation, TrafficVolumeLocation, CherryPickedLocation, AoLocation.

(3) Used by: Rerouting.

4.58. <<enumeration>> MCDMApprovalState

(1) Enumeration of possible types for a MCDM approval state (e.g. after casting a vote).

(2) In regulation proposal context, the NMOC actor can have any of the 4 different states. The initiator can only be in UNKNOWN (default) or REJECTED (when the B2B client revokes a proposal)

(3) Values:

a) **ACKNOWLEDGED**
Actor acknowledges the request and starts thinking about the response.

b) **APPROVED**
Actor approves the measure.

c) **REJECTED**
Actor disapproves the measure.

d) **UNKNOWN**
Default state, no response has been given by the actor.

(4) Used by: RegulationProposalListRequest, MCDMUserRoleAndApprovalState, RegulationProposal.

4.59. <<enumeration>> MCDMCoordinationLevel

(1) The MCDM coordination level.

(2) Values:

a) **FLIGHT**

b) **MEASURE**

(3) Used by: MCDMRoleUserCategory.

4.60. MCDMDeadlines

(1) All measure deadlines.

(2) Attributes:

a) **DateTimeMinute timeToCoordinate** *(Contextual)*
The deadline to finish the coordination activities.
Presence:

i) Optional in MCDMTopicUpdateRequest

ii) Optional otherwise.
Constraint: See AT_LEAST_ONE_DEADLINE_MUST_BE_SET

b) **DateTimeMinute timeToStartImplement** *(Contextual)*
The deadline to start the implementation activities.
Presence:

i) Optional in MCDMTopicUpdateRequest

ii) Optional otherwise.
Constraint: See AT_LEAST_ONE_DEADLINE_MUST_BE_SET

c) **DateTimeMinute timeToImplement** *(Contextual)*
The deadline to finish the implementation activities.
Presence:

i) Optional in MCDMTopicUpdateRequest

ii) Optional otherwise.
Constraint: See AT_LEAST_ONE_DEADLINE_MUST_BE_SET

(3) Constraint:

a)

Name	AT_LEAST_ONE_DEADLINE_MUST_BE_SET
Attributes	timeToCoordinate, timeToStartImplement, timeToImplement
Context	MCDMTopicUpdateRequest
Description	At least one deadline must be set.

(4) Used by: MCDMMeasureTopic.

4.61. `MCDMFlightTopic`

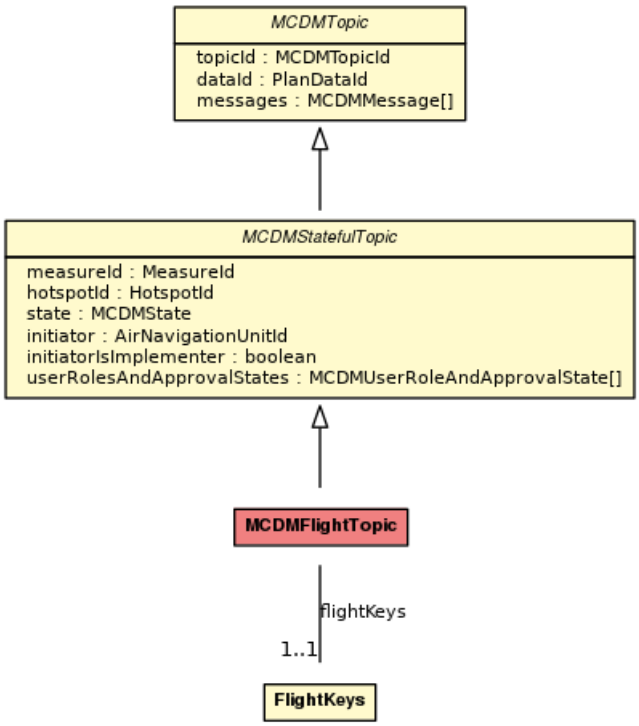

Figure 4.3. `MCDMFlightTopic` Class Diagram

(1) Represents a flight MCDM topic.

(2) **Note:**
Note that in regulation proposal context, only cherry picked flights have `MCDMFlightTopic`.

(3) Inherits from: `MCDMStatefulTopic`.

(4) Attributes:

a) **`FlightKeys` `flightKeys`** *(Mandatory)*
The flight keys.
Note: The keys are not unique and have to be used together with the `topicId`.

(5) Used by: MCDMMeasureTopic.

4.62. MCDMHotspotTopic

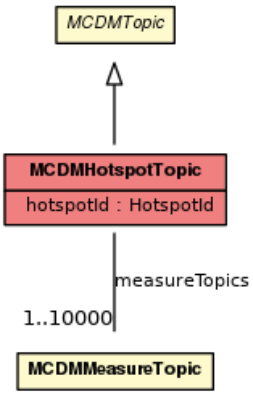

Figure 4.4. MCDMHotspotTopic Class Diagram

(1) Represents a hotspot MCDM topic.

(2) Inherits from: MCDMTopic.

(3) Attributes:

a) **HotspotId hotspotId** *(Contextual)*
The related hotspot.
Presence:

i) Must be null in MCDMTopicUpdateRequest

ii) Mandatory otherwise.
Constraint: See SAME_HOTSPOT_ID_MUST_BE_SET_IN_MEASURE_TOPICS

b) **Set<MCDMMeasureTopic> measureTopics** *(Contextual)*
A set of MCDM measure topics related to this MCDM hotspot topic.
In MCDMTopicUpdateRequest context, only one MCDM measure topic can be updated.
Presence:

i) Optional in MCDMTopicUpdateReply, MCDMTopicUpdateRequest

ii) Mandatory otherwise.
Constraints:

i) Size must be comprised between 1 and 10000.

ii) See AT_LEAST_MESSAGES_OR_MEASURE_TOPICS_MUST_BE_SET

iii) See ONLY_ONE_MEASURE_TOPIC_CAN_BE_UPDATED

iv) See SAME_HOTSPOT_ID_MUST_BE_SET_IN_MEASURE_TOPICS

(4) Constraints:

a)

Name	SAME_HOTSPOT_ID_MUST_BE_SET_IN_MEASURE_TOPICS
Attributes	hotspotId, measureTopics
Description	The hotspotId of all MCDM measure topic from measureTopics must be the same as the hotspotId of this MCDM hotspot topic.

b)

Name	AT_LEAST_MESSAGES_OR_MEASURE_TOPICS_MUST_BE_SET
Attribute	measureTopics
Context	MCDMTopicUpdateRequest
Description	At least messages or measureTopics must be set.

c)

Name	ONLY_ONE_MEASURE_TOPIC_CAN_BE_UPDATED
Attribute	measureTopics
Context	MCDMTopicUpdateRequest
Description	Only one measure topic of this hotspot topic can be updated.

(5) Used by: MCDMTopicListReply.

4.63. MCDMMeasureTopic

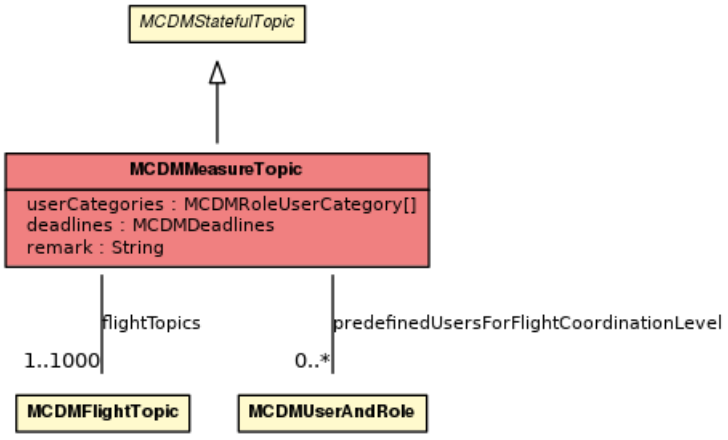

Figure 4.5. MCDMMeasureTopic Class Diagram

(1) Represents a measure MCDM topic.

(2) Inherits from: MCDMStatefulTopic.

(3) Attributes:

a) **Set<MCDMRoleUserCategory> userCategories** *(Optional)*
A set of user categories (actor and role) of the given topic.
Present if MCDMTopicField.userCategories has been requested. Note that it can be an empty set when in transient state i.e. when the measure has been created but the topic has not been updated yet via MCDMTopicUpdateRequest.
Constraint: Size must be comprised between 0 and 1000.

b) **MCDMDeadlines deadlines** *(Optional)*
The measure's deadlines.
Present if MCDMTopicField.deadlines has been requested.

c) **Set<MCDMFlightTopic> flightTopics** *(Optional)*
An optional set of MCDM flight topics related to this MCDM measure topic.
Constraints:

i) Size must be comprised between 1 and 1000.

ii) See SAME_HOTSPOT_ID_MUST_BE_SET_IN_FLIGHT_TOPICS

iii) See SAME_MEASURE_ID_MUST_BE_SET_IN_FLIGHT_TOPICS

d) **Set<MCDMUserAndRole> predefinedUsersForFlightCoordinationLevel** *(Optional)*
The pre-defined actors (and their role) for the per-flight coordination. Normally the flight actors (and their role) are computed based on the given userCategories and the profiles of the flights.
However if the user wants to control explicitly which actors should be involved in the flight coordination process, then he can define PredefinedUsersForFlightCoordinationLevel. If userCategories indicates that no flight coordination is needed, then NM systems will compute flight-by-flight the involved flight-coordination actors (and their role) based on the PredefinedUsersForFlightCoordinationLevel.
Constraint: Size must be comprised between 0 and ∞.

e) **string remark** *(Optional)*
Constraint: Pattern: `MULTILINE_TEXT{1,1000}`

(4) Constraints:

a)

Name	SAME_HOTSPOT_ID_MUST_BE_SET_IN_FLIGHT_TOPICS
Attribute	flightTopics
Description	The hotspotId of all MCDM flight topic from flightTopics must be the same of the hotspotId as this MCDM measure topic.

b)

Name	SAME_MEASURE_ID_MUST_BE_SET_IN_FLIGHT_TOPICS
Attribute	flightTopics

Description	The `measureId` of all MCDM flight topic from `flightTopics` must be the same as the `measureId` of this MCDM measure topic.

(5) Used by: Measure, MCDMTopicListReply, MCDMHotspotTopic.

4.64. MCDMMessage

(1) Free text used for message in MCDM collaboration environment.

(2) Attributes:

a) **DateTimeSecond time** *(Contextual)*
The time when NM received the message.
Presence:

i) Must be null in MCDMTopicUpdateRequest

ii) Mandatory otherwise.

b) **AirNavigationUnitId user** *(Contextual)*
The user's agency.
Not set when system message.
Presence:

i) Must be null in MCDMTopicUpdateRequest

ii) Optional otherwise.

c) **string message** *(Mandatory)*
The message.
Constraint: Pattern: `ANY{0,1024}`

d) **MCDMMessageType type** *(Contextual)*
Type of the message.
Presence:

i) Must be null in MCDMTopicUpdateRequest

ii) Mandatory otherwise.

(3) Used by: MCDMTopic, MCDMTopicMessageRetrievalReply.

4.65. <<enumeration>> MCDMMessageType

(1) Enumeration of possible types for a MCDM topic message.

(2) Values:

a) **INCOMING**

Messages that are incoming w.r.t. the current user: messages that other user have sent.

b) **OUTGOING**
Messages that are outgoing w.r.t. the current user: messages that this user has sent.

c) **SYSTEM**
Messages that are generated by the system (typically when the MCDM state of a measure is changed, then the system will update the MCDM states of the concerned flights correspondingly).

(3) Used by: MCDMMessage.

4.66. MCDMOnly

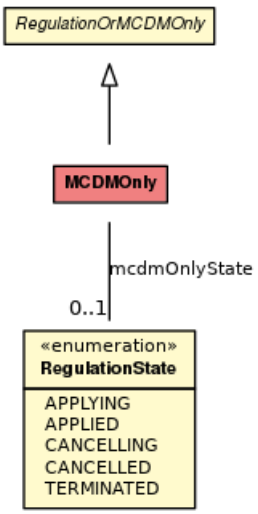

Figure 4.6. MCDMOnly Class Diagram

(1) An `mcdmOnly` measure. `MCDMOnly` measures are pure text coordination measures. The `remark` field actually describes what kind of measure this is and what is expected. They allow to associate flights to `mcdmOnly` measures and follow the MCDM process. However there are no proposal flights associated to mcdmOnly measures, so no what-if counts and flightlists can be done to evaluate any potential impact.

(2) Inherits from: `RegulationOrMCDMOnly`.

(3) Attributes:

a) **`RegulationState` `mcdmOnlyState`** *(Contextual)*
Current state of the mcdm only.
When requested, this attribute is never left null.
Presence:

i) Must be null in `MCDMOnlyCreationRequest`, `MCDMOnlyUpdateRequest`

ii) Optional otherwise.

(4) Used by: MCDMOnlyCreationReply, MCDMOnlyCreationRequest, ScenarioMCDMOnlyRetrievalReply, MCDMOnlyUpdateReply, MCDMOnlyCancelReply, MCDMOnlyUpdateRequest, MCDMOnlyListReply.

4.67. <<enumeration>> MCDMOnlyField

(1) Enumerates the fields that the caller may request to be returned in MCDMOnly objects when returned by MCDMOnlyListRequest.

(2) As a rule, client applications should never request mcdmOnly fields that they do not need. Client applications typically implement a query/retrieve pattern:

a) Query the small number of most relevant mcdmOnly fields to display to the end user

b) Retrieve more details for a given mcdmOnly when the end user has selected a mcdmOnly from the list

(3) Values:

a) **applicability**

b) **autolink**

c) **createdByFMP**

d) **dataId**

e) **delayConfirmationThreshold**

f) **delayTVSet**

g) **externallyEditable**

h) **initialConstraints**

i) **lastUpdate**

j) **linkedRegulations**

k) **location**

l) **mcdmOnlyState**

m) **mcdmRequired**

n) **measureCherryPicked**

o) **noDelayWindow**

p) **protectedLocation**

q) **reason**

r) **remark**

s) **scenarioReference**

t) **sourceHotspot**

u) **subType**

v) **supplementaryConstraints**

w) **updateCapacityRequired**

x) **updateTVActivationRequired**

(4) Used by: ScenarioMCDMOnlyRetrievalRequest, MCDMOnlyListRequest.

4.68. <<enumeration>> MCDMRole

(1) Enumeration of possible roles for an actor working with MCDM topic.

(2) In regulation proposal context, only a subset of the roles are used: APPROVAL, INITIATOR and NOT_INVOLVED are the only ones that can occur. NMOC is implicitely considered the implementer.

(3) Values:

a) **APPROVAL**
The user needs to approve.

b) **IMPLEMENTER**
The user needs to implement the measure/action on a flight.

c) **INFO**
The user is informed about the MCDM.

d) **INITIATOR**
The user is the initiator of the measure/action on a flight.

e) **NOT_INVOLVED**
The user is not involved in the measure/action on a flight.

f) **ROLE_INFO**
There are multiple ANUs found for a specific IMPLEMTENTER / APPROVAL (AO/tower). The first one was selected as for approval or for implementation. The others have the ROLE_INFO. They need to manually check that the correct ANU gets the APPROVAL or IMPLEMENTER role.

(4) Used by: MCDMUserRoleAndApprovalState, MCDMRoleUserCategory, MCDMUserAndRole.

4.69. MCDMRoleUserCategory

(1) Structure that contains mapping of a MCDM user category and its associated role.

(2) In the MCDM process, actor categories also have roles. They pilot how NM systems computes the default actors and their role. For example having ALL_FMP wih role for INFO for the measure means that all the FMP concerned (by the flights of the measure) will be inside the `MCDMUserRoleAndApprovalState` for the measure with a default role for INFO.

(3) Attributes:

a) **`MCDMUserCategory` `category`** *(Mandatory)*
The user category.

b) **`MCDMCoordinationLevel` `coordinationLevel`** *(Mandatory)*
The coordination level (flight or measure).

c) **`MCDMRole` `role`** *(Mandatory)*
The role for a category.
Constraint: See `NOT_INVOLVED_ROLE_NOT_ALLOWED`

(4) Constraint:

a)

Name	`NOT_INVOLVED_ROLE_NOT_ALLOWED`
Attribute	`role`
Context	`MCDMTopicListRequest`
Description	The `role` cannot be `NOT_INVOLVED`.

(5) Used by: MCDMMeasureTopic.

4.70. <<enumeration>> MCDMState

(1) Enumeration of possible types for a MCDM state.

(2) In regulation proposal context, FINISHED does not occur.

(3) The nominal finite state transaction flows for the trial related MCDM process (not the regulation proposal subset) are pictured below.

(4)

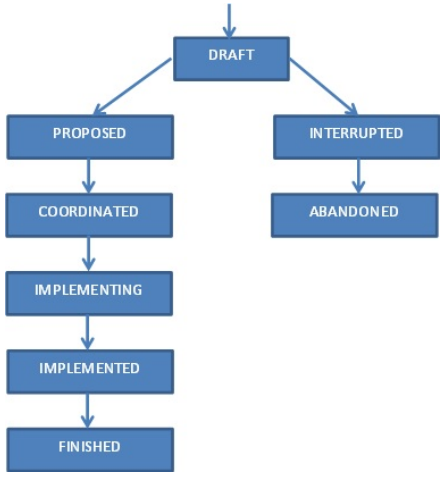

Figure 4.7. MCDM transition diagram

(5) Values:

a) **ABANDONED**
The measure will not occur or the flight will not be included in the measure. The measure was forced to abandoned by a manual user action by the initiator (e.g. a revoke of a proposal regulation by the initiator) or in regulation proposal context: NMOC has rejected the measure. Alternatively the MCDMState of a flight is set to abandoned because this flight specifically has been rejected by NMOC (a.o. due to RAD violations)

b) **COORDINATED**
In full MCDM context : All the concerned actors have agreed that:

i) the measure should/could occur

ii) or that a flight should be included in the measure and that the proposed action is accepted.

However the initiator is waiting for the for the expected capacity problem to materialize.
In regulation proposal context : COORDINATED means that NMOC has acknowledged the regulation proposal ans started handling the request.

c) **DRAFT**
A measure is being created to solve a hotspot/problem area. It is too early to start MCDM approval: the solution is in the process of being evaluated by the initiator. The measure/flight is draft even while it is incomplete, for example the FMP has selected that there will be cherry picking but has not yet selected the flights.
The measure/flight may be reset from subsequent states to Draft when the proposed measure/proposed action on a flight cannot is not sufficient and it needs to be changed (for example in regulation proposal context : when additional flights need to be added).
If required, the measure/flight may also be forced by the Initiator to Abandoned state at this stage (for example: in regulation proposal context when revoking a proposal).

d) **FINISHED**

This state is a simple combination of Implemented and the end time of the hotspot being in the past.
Note that in regulation proposal context, this state does not occur.

e) **IMPLEMENTED**
The actor who should implement the measure/flight action has agreed to implement it at the relevant time - that is at or before the implementation time limit.
Implemented may mean that an action will be taken, for example ATC may do something when the flight appears, or implemented may mean that the action has already been taken, for example the aircraft operator may have submitted a change to his flight plan.
In the context of regulation proposals, it means that the proposal has been accepted and that the "real" flight (i.e. non-proposal flight) has received a CTOT.

f) **IMPLEMENTING**
When the Initiator decides the measure/flight action is really needed, the measure/flight is moved from Coordinated to For Implementation and the job of implementation can begin.
In the context of regulation proposals, it is a temporary state required by NM systems to indicate that the transformation of proposal flight CTOT into nomal flight CTOT has started, but not all flights have been processed yet.

g) **INTERRUPTED**
The measure will not occur or the flight became excluded from the measure.
The measure was forced to abandoned by the system or by NMOC after it had been acceted (for example: a measure could get interrupted after NMOC cancelled the measure ater having accepted the proposal to create it).
Alternativley the aircraft operator refiled and avoided a cherry pick regulation.

h) **PROPOSED**
The measure/flight is being shown via B2 or NOP to all concerned actors in order to get their agreement that the measure should/could occur or that the flight should be included in the measure.
Note that in regulation proposal context, it means that the B2B client wants NMOC to review the proposal but NMOC has not started looking at it yet.

(6) Used by: FlightMCDMInfo, RegulationProposal, MCDMStateUpdateRequest, MCDMStateUpdateReply, FlightAtfcmMeasureLocation, MCDMStatefulTopic.

4.71. <<abstract>> MCDMStatefulTopic

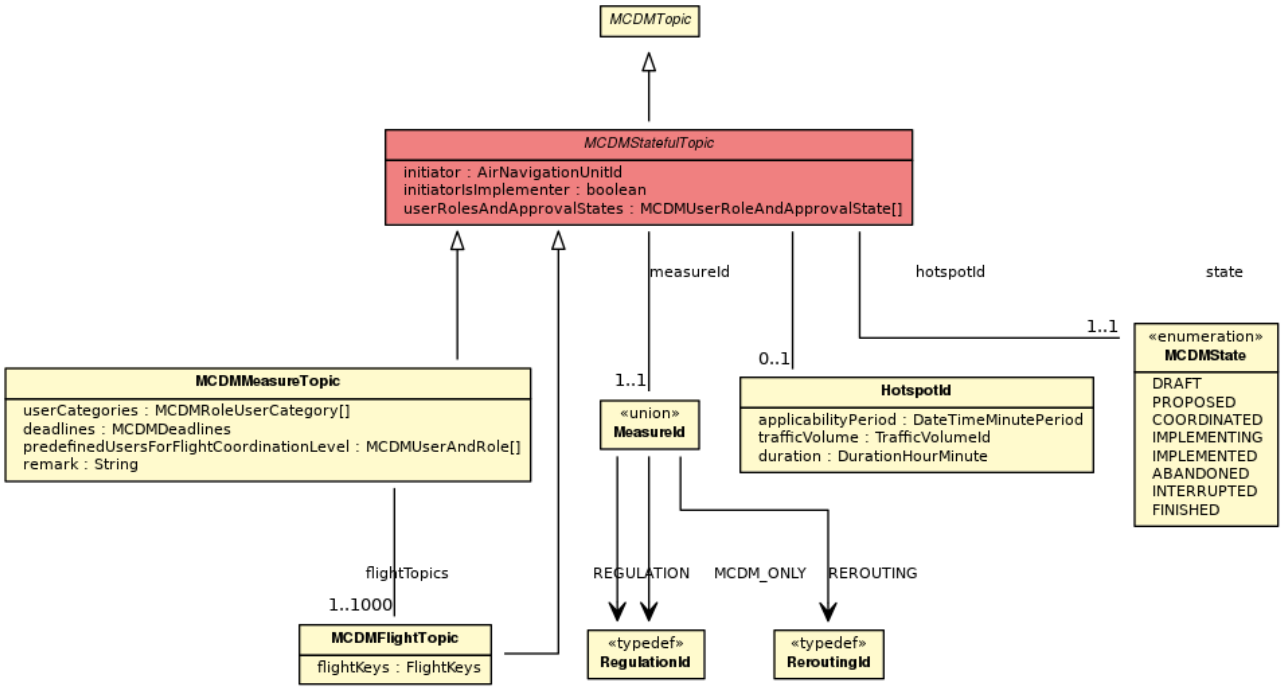

Figure 4.8. <<abstract>> MCDMStatefulTopic Class Diagram

(1) Defines the common attributes of MCDM topics that have state (measure and flight).

(2) Inherits from: MCDMTopic.

(3) Attributes:

a) **MeasureId measureId** *(Contextual)*
The related measure.
Presence:

i) Must be null in MCDMTopicUpdateRequest

ii) Mandatory otherwise.

b) **HotspotId hotspotId** *(Contextual)*
The related hotspot.
Presence:

i) Must be null in MCDMTopicUpdateRequest

ii) Optional otherwise.

c) **MCDMState state** *(Contextual)*
The MCDM Topic state.

Presence:

i) Optional in MCDMTopicUpdateReply, MCDMTopicUpdateRequest

ii) Mandatory otherwise.
Constraint: See INTERRUPTED_MCDM_STATE_NOT_ALLOWED

d) **AirNavigationUnitId initiator** *(Optional)*
The initiator.
Present if MCDMTopicField.initiator was requested.

e) **boolean initiatorIsImplementer** *(Optional)*
Flag indicating whether the initiator is the implementer or not.
Present if MCDMTopicField.initiator was requested.
Constraint: See INITIATOR_IS_IMPLEMENTER_MUST_BE_NULL

f) **Set<MCDMUserRoleAndApprovalState> userRolesAndApprovalStates** *(Optional)*
A set of user with their role and optionally an approval state.
In regulation proposal context, the NM systems maintain this: there is the intiator and NMOC actor and in case of cherry picked regulations, also all other potentially impacts FMP are present (with NOT_INVOLVED role).
When NMOC accepts/rejects a measure or flight, the corresponding NMOC approval state becomes accepted or rejected.
When the FMP afterwards changes the MCDMState of the regulation back to DRAFT, the NMOC actor approvalState is reset to UNKNOWN.
Present if MCDMTopicField.userRolesAndApprovalStates has been requested.
Can be null if no actors are defined yet.
Constraint: Size must be comprised between 1 and ∞.

(4) Constraints:

a)

Name	INTERRUPTED_MCDM_STATE_NOT_ALLOWED
Attribute	state
Context	MCDMTopicUpdateRequest
Description	The attribute state cannot be INTERRUPTED.

b)

Name	INITIATOR_IS_IMPLEMENTER_MUST_BE_NULL
Attribute	initiatorIsImplementer
Context	MCDMTopicUpdateRequest
Description	The attribute initiatorIsImplementer must be null when the topic to be updated is a MCDMFlightTopic.

(5) Extended by: MCDMFlightTopic, MCDMMeasureTopic.

4.72. <<abstract>> MCDMTopic

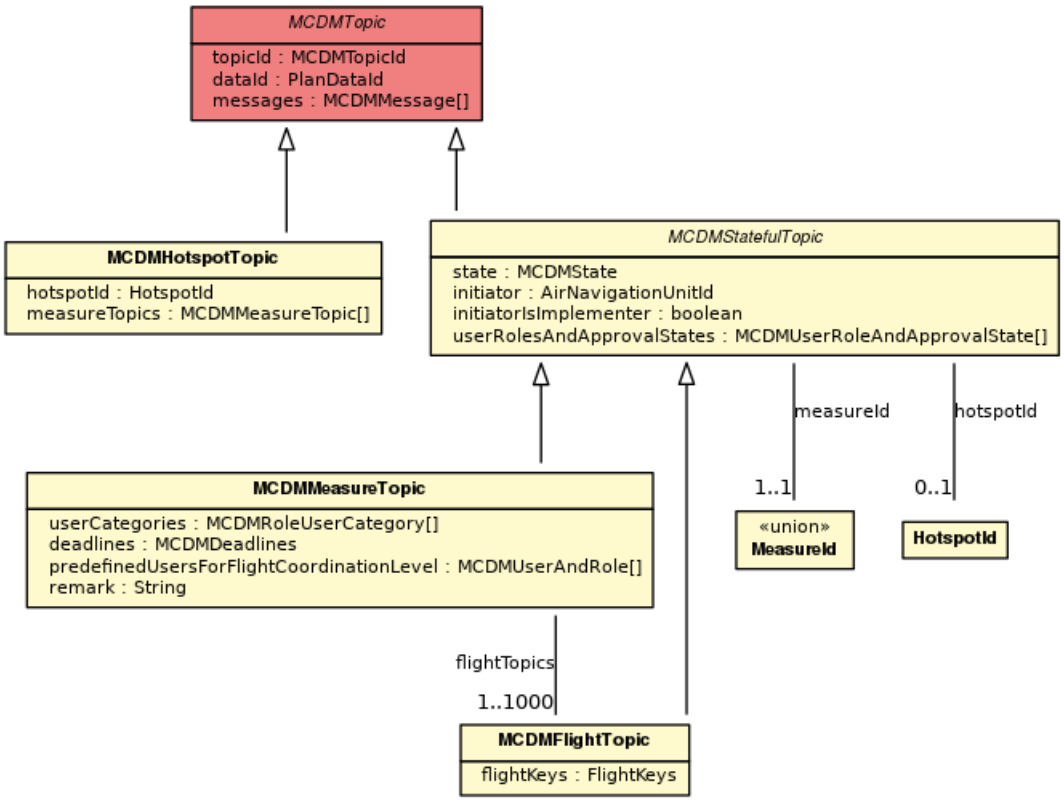

Figure 4.9. <<abstract>> MCDMTopic Class Diagram

(1) An abstract object that defines the common attributes of MCDM topics (hotspot, measure, flight).

(2) Attributes:

a) **MCDMTopicId topicId** *(Mandatory)*
The unique identifier of the topic.

b) **PlanDataId dataId** *(Mandatory)*
An opaque identifier, representing the version of this MCDM topic.
See MCDM Update pattern.

c) **Set<MCDMMessage> messages** *(Optional)*
Conversation history, usualy comments as part of collaboration between different actors.
Only present if MCDMTopicField.messages is requested.
Constraints:

i) Size must be comprised between 1 and 10000.

ii) See ONLY_ONE_MESSAGE_CAN_BE_SENT

(3) Constraint:

a)

Name	ONLY_ONE_MESSAGE_CAN_BE_SENT
Attribute	messages
Context	MCDMTopicUpdateRequest
Description	Only one message can be sent.

(4) Extended by: MCDMHotspotTopic, MCDMStatefulTopic.

(5) Used by: MCDMTopicUpdateReply, MCDMTopicUpdateRequest.

4.73. <<enumeration>> MCDMTopicField

(1) Enumerates the fields that the caller may request to be returned in MCDMTopic objects when returned by MCDMTopicListRequest.

(2) As a rule, client applications should never request topic fields that they do not need. Client applications typically implement a query/retrieve pattern:

a) Query the small number of most relevant topic fields to display to the end user

b) Retrieve more details for a given topic when the end user has selected a topic from the list

(3) Note that the MCDM deadlines field is trial related (STAM): it is only accessible (authorized) during specific trials or on specific test platforms.

(4) Note that the MCDM messages field is trial related (STAM): it is only accessible (authorized) during specific trials or on specific test platforms.

(5) Values:

Value	Weight
deadlines	Light
initiator	Light
messages	Light
predefinedUsersForFlightCoordinationLevel	Light
remark	Light
userCategories	Light
userRolesAndApprovalStates	Light

Table 4.2. <<enumeration>> MCDMTopicField

(6) Used by: MCDMTopicListRequest.

4.74. typedef<string> MCDMTopicId

(1) A unique identifier of a MCDM topic.

(2) Pattern: `(H|M|F)_(ALPHA|DIGIT|_){1,300}`

(3) Used by: MCDMTopicListRequestSelector, MCDMTopicMessageRetrievalRequest, MCDMTopic.

4.75. MCDMTopicListRequestSelector

(1) Selector used in MCDMTopicListRequest.

(2) Choices:

a) **Set<MCDMTopicId> topics**
Selects the topics that match one of the topic identifiers in this set.

b) **Set<MeasureId> measures**
Selects the topics with a measure that matches an entry in this set.

c) **Set<HotspotId> hotspots**
Selects the topics with a hotspot that matches an entry in this set.

> **Note:**
> Note that querying MCDM based on hotspots is trial related (STAM): it is only accessible (authorized) during specific trials or on specific test platforms.

d) **Set<AirNavigationUnitId> actors**
Selects the topics with an actor that matches an entry in this set.

> **Note:**
> Note that querying MCDM based on actors is trial related (STAM): it is only accessible (authorized) during specific trials or on specific test platforms.

(3) Used by: MCDMTopicListRequest.

4.76. MCDMUserAndRole

(1) A user and its role.

(2) Attributes:

a) **AirNavigationUnitId user** *(Mandatory)*
The user.

b) **MCDMRole role** *(Contextual)*
The role.
Presence:

i) Optional in MCDMTopicUpdateRequest

ii) Mandatory otherwise.
Constraint: See MCDM_ROLE_IS_MANDATORY

(3) Constraint:

a)

Name	MCDM_ROLE_IS_MANDATORY
Attribute	role
Context	MCDMTopicUpdateRequest
Description	MCDMRole is mandatory.

(4) Used by: MCDMMeasureTopic.

4.77. <<enumeration>> MCDMUserCategory

(1) Enumeration of possible possible categories of actors working with MCDM topics.

(2) Values:

a) **ADJACENT_FMP**

b) **AIRCRAFT_OPERATOR**

c) **ALL_FMP**

d) **NMOC**

e) **TOWER**

(3) Used by: MCDMRoleUserCategory.

4.78. MCDMUserRoleAndApprovalState

(1) A user, its role and its optional approval state.

(2) Attributes:

a) **AirNavigationUnitId user** *(Mandatory)*
The user.

b) **MCDMRole role** *(Contextual)*
The role.
Presence:

i) Optional in MCDMTopicUpdateRequest

ii) Mandatory otherwise.

c) **`MCDMApprovalState approvalState`** *(Contextual)*
The approval state.
Presence:

i) Optional in `MCDMTopicUpdateRequest`

ii) Mandatory otherwise.

(3) Used by: MCDMStatefulTopic.

4.79. `<<abstract>> Measure`

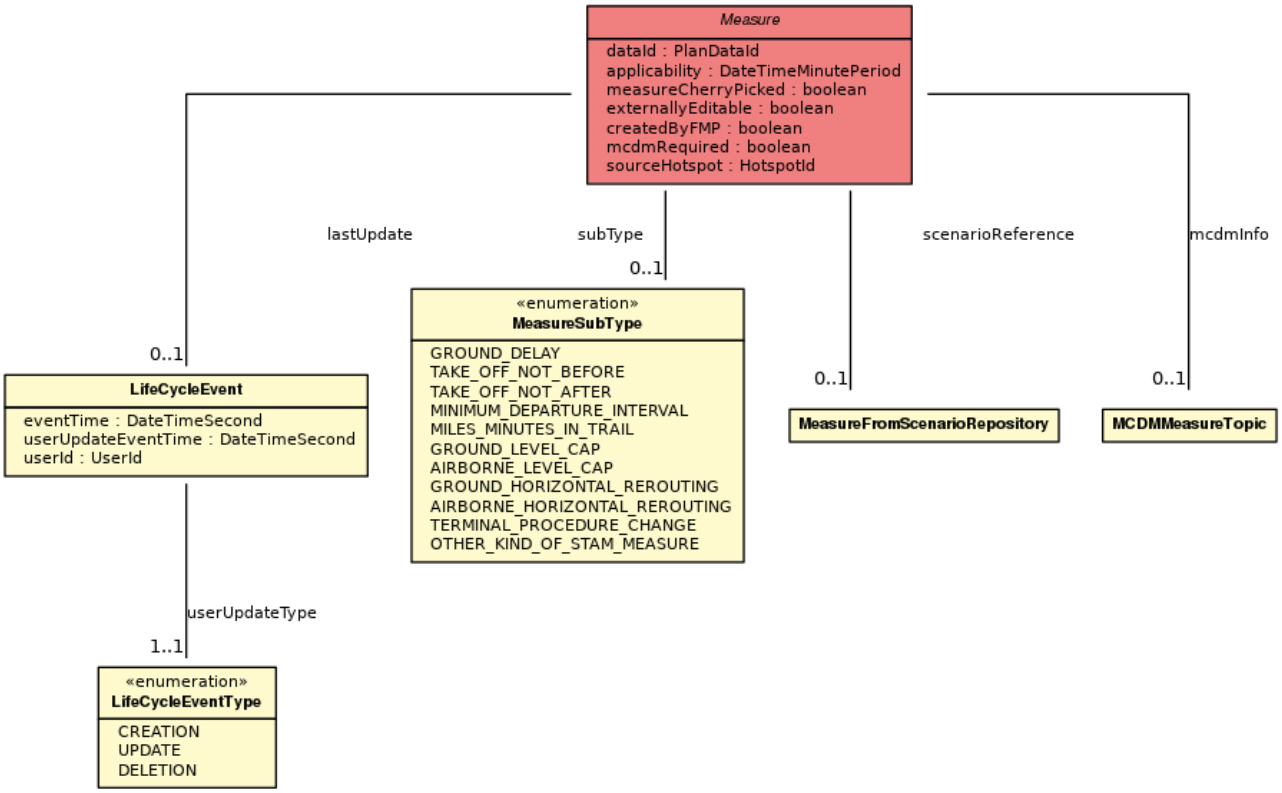

Figure 4.10. `<<abstract>> Measure` Class Diagram

(1) Abstract ancestor of a measure.

(2) Attributes:

a) **`PlanDataId dataId`** *(Contextual)*
Opaque identifier representing the version of this measure.
The caller shall always keep this value unchanged.
See Update Pattern.
Presence:

i) Must be null in MCDMOnlyCreationRequest, RegulationCreationRequest, RegulationProposalFilingRequest, ReroutingCreationRequest

ii) Optional otherwise.

b) **DateTimeMinutePeriod applicability** *(Contextual)*
The period of time during which the measure affects the flights entering the traffic volumes of this regulation.
This period extends to maximum 24 hours.
When requested, this attribute is never left null.

> **NOTE:**
> For cherry picked regulation, NM adapts the applicability based on the flights in the regulation.

Presence:

i) Mandatory in MCDMOnlyCreationRequest, MCDMOnlyUpdateRequest, RegulationCreationRequest, RegulationProposalFilingRequest, RegulationProposalUpdateRequest, RegulationUpdateRequest, ReroutingCreationRequest, ReroutingUpdateRequest

ii) Optional otherwise.

Constraint: See APPLICABILITY_PERIOD_CANNOT_BE_GREATER_THAN_24_HOURS

c) **boolean measureCherryPicked** *(Contextual)*
Indicates if this regulation was created for flight cherry picking (only the selected cherry picked flights are subject to the measure, e.g., will have a delay).
When requested, this attribute is never left null.
Presence:

i) Mandatory in MCDMOnlyCreationRequest, RegulationCreationRequest, RegulationProposalFilingRequest, ReroutingCreationRequest

ii) Must be null in MCDMOnlyUpdateRequest, RegulationProposalUpdateRequest, RegulationUpdateRequest, ReroutingUpdateRequest

iii) Optional otherwise.

d) **LifeCycleEvent lastUpdate** *(Contextual)*
Provides the last update life cycle event.
When requested, this attribute is never left null.
Presence:

i) Must be null in MCDMOnlyCreationRequest, MCDMOnlyUpdateRequest, RegulationCreationRequest, RegulationProposalFilingRequest, RegulationProposalUpdateRequest, RegulationUpdateRequest, ReroutingCreationRequest, ReroutingUpdateRequest

ii) Optional otherwise.

e) **`boolean externallyEditable`** *(Contextual)*
Indicates that the FMP is entitled to modify the regulation, either because the regulation is part of a regulation proposal, or because NMOC flagged this regulation as being externally editable.
When requested, this attribute is never left null.
Presence:

i) Must be null in `MCDMOnlyCreationRequest`, `MCDMOnlyUpdateRequest`, `RegulationCreationRequest`, `RegulationProposalFilingRequest`, `RegulationProposalUpdateRequest`, `RegulationUpdateRequest`, `ReroutingCreationRequest`, `ReroutingUpdateRequest`

ii) Optional otherwise.

f) **`MeasureSubType subType`** *(Contextual)*
The specific type of this measure.
When requested, this attribute is never left null.
Presence:

i) Mandatory in `MCDMOnlyCreationRequest`, `RegulationCreationRequest`, `RegulationProposalFilingRequest`, `ReroutingCreationRequest`

ii) Must be null in `RegulationProposalUpdateRequest`

iii) Optional otherwise.
Constraint: See `INVALID_SUBTYPE`

g) **`boolean createdByFMP`** *(Contextual)*
Indicates if the measure has been created by an FMP himself or not.
When requested, this attribute is never left null.
Presence:

i) Must be null in `MCDMOnlyCreationRequest`, `MCDMOnlyUpdateRequest`, `RegulationCreationRequest`, `RegulationProposalFilingRequest`, `RegulationProposalUpdateRequest`, `RegulationUpdateRequest`, `ReroutingCreationRequest`, `ReroutingUpdateRequest`

ii) Optional otherwise.

h) **`boolean mcdmRequired`** *(Contextual)*
Indicates if full MCDM is required for this regulation.
`mcdmRequired` is False in case of proposal regulation MCDM workflow.
When requested, this attribute is never left null.
Presence:

i) Mandatory in `RegulationCreationRequest`, `ReroutingCreationRequest`

ii) Must be null in `MCDMOnlyCreationRequest`, `RegulationProposalFilingRequest`, `RegulationProposalUpdateRequest`

iii) Optional otherwise.

i) **HotspotId sourceHotspot** *(Optional)*
If this regulation was created from a hotspot resolution, id of that hotspot.
When requested, this attribute is never left null.

j) **MeasureFromScenarioRepository scenarioReference** *(Optional)*
In measure list context, indicates if this measure is a measure from the scenario repository and if so this field indicates from which scenario and which measure inside that scenario.
In create/update context, indicates on which scenario and measure to base the new/updated measure on (for example: to get the rerouting cost criteria to guide profile generation.)

k) **MCDMMeasureTopic mcdmInfo** *(Optional)*
When requesting scenario measures, (via ScenarioRegulationRetrievalRequest or ScenarioReroutingRetrievalRequest or ScenarioMCDMOnlyRetrievalRequest , the MCDMInfo can optionally be present, (if the measure needs to be coordinated via MCDM on application).

(3) Constraints:

a)

Name	APPLICABILITY_PERIOD_CANNOT_BE_GREATER_THAN_24_HOURS
Attribute	applicability
Description	The applicability period must be smaller or equal to 24 hours.

b)

Name	INVALID_SUBTYPE
Attribute	subType
Description	Attribute subType must be equals to GROUND_DELAY when filing a regulation proposal.

(4) Extended by: RegulationOrMCDMOnly, Rerouting.

4.80. MeasureFromScenarioRepository

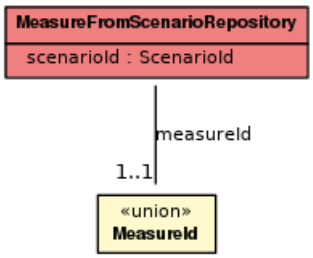

Figure 4.11. MeasureFromScenarioRepository Class Diagram

(1)

(2) Attributes:

a) **ScenarioId scenarioId** *(Mandatory)*

b) **MeasureId measureId** *(Mandatory)*

(3) Used by: Measure.

4.81. MeasureId

(1) Represents a regulation or rerouting identifier.

(2) Choices:

a) **RegulationId REGULATION**
The regulation unique identifier.

b) **ReroutingId REROUTING**
The rerouting unique identifier.

c) **RegulationId MCDM_ONLY**
The mcdm only unique identifier.

(3) Used by: FlightMCDMInfo, MeasureOpLogRetrievalRequest, FlightListByMeasureRequest, MCDMTopicListRequestSelector, NetworkImpactAssessmentRetrievalRequest, MeasureIdAndTV, MeasureFromScenarioRepository, MCDMStatefulTopic.

4.82. MeasureIdAndTV

(1) Represents a summary of a measures inside a scenario: it gives the measureId and the traffic volume id on which the measure is. In addition it describes for a rerouting measure, the (indireclty) off-loaded traffic volumes.

(2) Attributes:

a) **MeasureId measureId** *(Mandatory)*

b) **TrafficVolumeId tvId** *(Optional)*

c) **Set<AirspaceId> offLoadAirspaces** *(Mandatory)*
Constraint: Size must be comprised between 0 and ∞.

(3) Used by: ScenarioAttributes.

4.83. MeasureListReplyData

(1) Abstract reply data associated to the reply of the abstract RegulationOrMCDMOnlyListRequest or ReroutingListRequest.

(2) Attributes:

a) **`boolean planTransferred`** *(Optional)*
Indicates if the plan has been transferred to the `OPERATIONAL` dataset.
When `false`, it means that the most up-to-date data can be found in the `FORECAST` dataset.
(See also Forecast and Operational Datasets)
Must be present if the dataset is `OPERATIONAL`, must be null otherwise.

b) **`boolean planCutOffReached`** *(Optional)*
Indicates if the plan can still be updated in the FORECAST dataset, i.e. if the forecast cut-off time has been reached or not.
Must be present if the dataset is `FORECAST`, must be null otherwise.

c) **`Dataset dataset`** *(Mandatory)*
Dataset on which the regulations are requested.

(3) Extended by: RegulationOrMCDMOnlyListReplyData, ReroutingListReplyData.

4.84. <<abstract>> MeasureListRequest

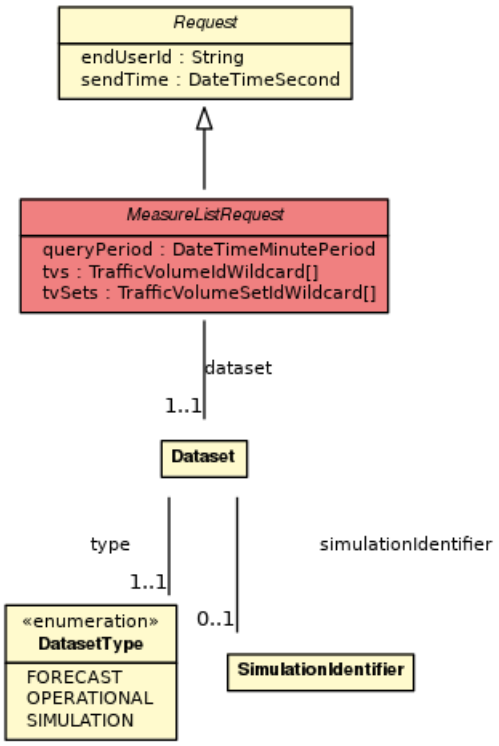

Figure 4.12. <<abstract>> MeasureListRequest Class Diagram

(1) Abstract request to query an NM measure list, as well as to retrieve the measure details.

(2) Inherits from: `Request`.

(3) Attributes:

a) **`Dataset` `dataset`** *(Mandatory)*
Dataset on which the measures are requested.
See Forecast and Operational Datasets and Simulation Datasets
Constraint: See `INVALID_QUERY_PERIOD_RANGE`

b) **`DateTimeMinutePeriod` `queryPeriod`** *(Mandatory)*
Selects the measures of which the applicability period overlaps this `queryPeriod`. The `MeasureListRequest` provides the measure list for:

i) The tactical day: up-to-date view of the set of measures that exist for today in the NM system, at the time of the request

ii) One pre-tactical day (tomorrow): the request returns the up-to-date view of the set of measures applying tomorrow in the NM system (hence including cross-midnight measures), at the time of the request.

iii) 21 post-operations days (yesterday down to 21 days in the past): the request returns the set of measures in the system for that day, as they existed at the end of the day: consequently, a request for a post-ops day always returns the same result regardless of the time at which it is issued in the day.

Constraints:

i) See `INVALID_QUERY_PERIOD_RANGE`

ii) See `MAX_QUERY_PERIOD_DURATION_IS_2_DAYS`

c) **`Set<TrafficVolumeIdWildcard>` `tvs`** *(Optional)*
Selects the regulations applying to the given traffic volumes.
The logical OR operator is meant between the items in the set.
By default, all traffic volumes are considered.
Constraint: Size must be comprised between 1 and 100.

d) **`Set<TrafficVolumeSetIdWildcard>` `tvSets`** *(Optional)*
Selects the regulations applying to a traffic volume belonging to at least one of the given traffic volume sets.
The logical OR operator is meant between the items in the set.
By default, all traffic volumes of all traffic volume sets are considered.
Constraint: Size must be comprised between 1 and 100.

(4) Constraints:

a)

Name	`MAX_QUERY_PERIOD_DURATION_IS_2_DAYS`
Attribute	`queryPeriod`
Description	The maximum duration of `queryPeriod` is 2 days

b)

Name	`INVALID_QUERY_PERIOD_RANGE`
Attributes	`queryPeriod`, `dataset`

Description	The `dataset.type` on which the measures are requested and the `queryPeriod` must be set according to the following rules: i) if the `DatasetType` is equals to `FORECAST` the `queryPeriod` shall be defined within the range [today .. today+5d] ii) if the `DatasetType` is equals to `OPERATIONAL` the `queryPeriod` shall be defined within the range [today .. tomorrow]

(5) Extended by: RegulationOrMCDMOnlyListRequest, ReroutingListRequest.

4.85. <<enumeration>> MeasureSubType

(1) The sub type of the measure.

(2) Values:

a) **AIRBORNE_HORIZONTAL_REROUTING**

b) **AIRBORNE_LEVEL_CAP**

c) **GROUND_DELAY**

d) **GROUND_HORIZONTAL_REROUTING**

e) **GROUND_LEVEL_CAP**

f) **MILES_MINUTES_IN_TRAIL**

g) **MINIMUM_DEPARTURE_INTERVAL**

h) **OTHER_KIND_OF_STAM_MEASURE**

i) **TAKE_OFF_NOT_AFTER**

j) **TAKE_OFF_NOT_BEFORE**

k) **TERMINAL_PROCEDURE_CHANGE**

(3) Used by: Measure, FlightAtfcmMeasureLocation.

4.86. NMOCManagedSimulation

(1) The simulation is managed (start/stop) and prepared by NMOC for the other users (B2B & B2C) to have a look at the results.

(2) Optionally the users can modify the measures and tactical updates to evaluate the effect.

(3) The dataset corresponding to a `NMOCManagedSimulation` simulation has as type NMOC_MANAGED_SIMULATION.

(4) Inherits from: `Simulation`.

(5) Attributes:

a) **`DatasetReference` `datasetReference`** *(Optional)*
The reference on which this simulation is based (i.e. from which environment data has been copied and where initially the flights and measures have been copied from).
Note that if the `datasetReference` is none, then it concerns a standalone simulation (typically for a future date with specifically prepared environment and flight data) without a reference.
Constraint: See `INVALID_DATASET_REFERENCE_TYPE`

(6) Constraints:

a)

Name	`INVALID_DATASET_REFERENCE_TYPE`
Attribute	`datasetReference`
Description	DatasetReference.type must be `OPERATIONAL` or `FORECAST`.

b)

Name	`INVALID_SIMULATION_IDENTIFIER_TYPE`
Attribute	`dataset`
Description	SimulationIdentifier.simulationType must be `NMOC_MANAGED_SIMULATION`.

4.87. `OtherReroutingConstraint`

(1) Describes a rerouting constraint not already listed.

(2) Inherits from: `ReroutingConstraint`.

4.88. `OTMV`

(1) Definition of an OTMV.

(2) Attributes:

a) **`TrafficVolumeId` `trafficVolume`** *(Mandatory)*
Traffic volume to which this OTMV applies.

b) **`DurationHourMinute` `otmvDuration`** *(Mandatory)*
The duration of this OTMV.

c) **`OTMVPeak` `peak`** *(Optional)*
The peak configuration of this OTMV.

d) **OTMVSustained sustained** *(Optional)*
The sustained configuration of this OTMV.

e) **string remark** *(Optional)*
Constraint: Pattern: TEXT{1,255}

(3) Used by: PlannedOTMV.

4.89. OtmvAlert

(1) OTMV alert for a date time period. This indicates which count periods/intervals are involved in an OTMV peak or an OTMV sustained alert: all count periods that start in `OtmvAlert.period` are considered to have an OTMV alert with OtmvStatus.

(2) Attributes:

a) **DateTimeMinutePeriod period** *(Mandatory)*
Period of the OTMV alert.

b) **OtmvStatus status** *(Mandatory)*
Status of the OTMV alert.

(3) Used by: TrafficCountsReplyData, DeltaCounts.

4.90. OTMVPeak

(1) OTMV peak data.

(2) Attributes:

a) **OTMVThreshold threshold** *(Mandatory)*
Peak threshold of an OTMV.

(3) Used by: OTMV.

4.91. OTMVPlan

(1) OTMV plan for a given traffic volume on a given day.

(2) An OTMV plan is a special plan in the sense that for a given traffic volume there can be multiple OTMV durations. For each of these durations there exists a plan covering the full day (completely independent of any other duration). In update mode, only one duration can be updated in a single request. For a specific (traffic volume, OTMV duration) plan, there can be cases where there is no data known. So there are (traffic volume, OTMV duration) pairs where there is NO_DATA at all or only for some periods.

(3) In a retrieval context, the plan for a (traffic volume, OTMV duration) is said to be 'complete' in the sense that it contains all the plan entries from all involved data sources (including NO_DATA data source in case no info is known).

(4) In an update context, the plan can be complete (if the B2B client designer decided to overwrite the default CACD values) or limited to the (full list) of (pre-)tactical updates with the gaps marked as AIRSPACE (meaning in update context: either NO_DATA or CACD) datasource (to obtain a complete time partition).

(5) In any case, periods in the time partition marked with datasource AIRSPACE correspond to removing any potential (pre-)tactical update and hence reset the corresponding values to the CACD definition for that period.

(6) Inherits from: TacticalConfigurationPlan.

(7) Attributes:

a) **TrafficVolumeId trafficVolume** *(Mandatory)*
Traffic volume to which this OTMV plan applies.

b) **Map<DurationHourMinute,OTMVPlanForDuration> otmvPlans** *(Mandatory)*
The set of durations for which there are OTMV updates and for each duration, the OTMV plan.
If only one specific duration was requested, then the map will only contain the OTMV plan for that duration.
Constraints:

i) Size must be comprised between 0 and ∞.

ii) See INCOMPLETE_SCHEDULE

iii) See INVALID_SCHEDULE

iv) See ONLY_ONE_ENTRY_CAN_BE_UPDATED_IN_PLAN

(8) Constraints:

a)

Name	INCOMPLETE_SCHEDULE
Attribute	otmvPlans
Context	CapacityPlanUpdateRequest, OTMVPlanUpdateRequest, RunwayConfigurationPlanUpdateRequest, SectorConfigurationPlanUpdateRequest, TrafficVolumeActivationPlanUpdateRequest
Description	clientSchedule has gaps and/or overlaps in the time partition or is not covering exactly one day.

b)

Name	ONLY_ONE_ENTRY_CAN_BE_UPDATED_IN_PLAN
Attribute	otmvPlans
Context	CapacityPlanUpdateRequest, OTMVPlanUpdateRequest, RunwayConfigurationPlanUpdateRequest, SectorConfigurationPlanUpdateRequest, TrafficVolumeActivationPlanUpdateRequest

Description	Only one entry in the `otmvPlans` map attribute can be updated (i.e., for one duration).

c)

Name	`INVALID_SCHEDULE`
Attribute	`otmvPlans`
Context	`CapacityPlanUpdateRequest`, `OTMVPlanUpdateRequest`, `RunwayConfigurationPlanUpdateRequest`, `SectorConfigurationPlanUpdateRequest`, `TrafficVolumeActivationPlanUpdateRequest`
Description	The duration key used in the `otmvPlans` map attribute has to be equal to the duration of all OTMVs linked to that duration key.

4.92. `OTMVPlanForDuration`

(1) The OTMV plan for a specific duration.

(2) Attributes:

a) **`Set<PlannedOTMV> nmSchedule`** *(Contextual)*
The plan resulting from the superimposition of all constraints from all data sources, which the NM system is using as plan for its own calculations.
The NM schedule exposes the complete time partition of the configurations for the day, i.e. the data coming from the various data sources contributing to the NM view of the plan: these data sources are defined in PlanDataSource.
The NM schedule cannot be updated directly by the caller; it is the outcome of updates via the various data sources.
The possible values of dataSource are limited to `NO_DATA`, `AIRSPACE` and `TACTICAL`.
Presence:

i) Must be null in `CapacityPlanUpdateRequest`, `OTMVPlanUpdateRequest`, `RunwayConfigurationPlanUpdateRequest`, `SectorConfigurationPlanUpdateRequest`, `TrafficVolumeActivationPlanUpdateRequest`

ii) Mandatory otherwise.
Constraint: Size must be comprised between 0 and ∞.

b) **`Set<PlannedOTMV> clientSchedule`** *(Mandatory)*
(Pre-)tactical OTMVs associated to their applicability period, as maintained by the client. This plan contains only the updated configurations together with an indication that the default CACD values (AIRSPACE datasource) apply when not updated (cf. PlanDataSource). The actual CACD values for these CACD periods can be found in the `nmSchedule`
In an update context, the `clientSchedule` can be complete (if the B2B client designer decided to overwrite the default CACD values) or limited to the actual differences with regards to the CACD defaults. This is a B2B client designer's decision and depends on how CACD wants to be used in combination with the B2B. If the `clientSchedule` only contains the actual differences with regards to the CACD defaults, then the `clientschedule` still needs

to contain the full list of (pre-) tactical updates and for the non (pre-) tactically updated periods, an explicit indication that the CACD values need to be used (but without repeating the CACD values themselves). So in any case, the `clientschedule` needs to be a complete time partition for the full day.
In any case, it is of drastic importance to understand that an entry in the schedule (i.e. a period) overwrites all CACD values in that period.
The possible values of dataSource are limited to `AIRSPACE` and `TACTICAL` - the `AIRSPACE` value meaning that the value associated to the applicabilityPeriod is the CACD one (i.e. CACD or that there is no data defined in the CACD), and the `TACTICAL` value meaning that this plan entry corresponds to the explicit tactical update expressed via otmv.
Constraint: Size must be comprised between 0 and ∞.

(3) Used by: OTMVPlans, OTMVPlan.

4.93. `OTMVPlans`

(1) OTMV plans for one or more traffic volumes on a given day.

(2) An OTMV plan is a special plan in the sense that for a given traffic volume there can be multiple OTMV durations. For each of these durations there exists a plan covering the full day (completely independent of any other duration). In update mode, only one duration can be updated in a single request. For a specific (traffic volume, OTMV duration) plan, there can be cases where there is no data known. So there are (traffic volume, OTMV duration) pairs where there is NO_DATA at all or only for some periods.

(3) In a retrieval context, the plan for a (traffic volume, OTMV duration) is said to be 'complete' in the sense that it contains all the plan entries from all involved data sources (including NO_DATA data source in case no info is known).

(4) In an update context, the plan can be complete (if the B2B client designer decided to overwrite the default CACD values) or limited to the (full list) of (pre-)tactical updates with the gaps marked as AIRSPACE (meaning in update context: either NO_DATA or CACD) datasource (to obtain a complete time partition).

(5) In any case, periods in the time partition marked with datasource AIRSPACE correspond to removing any potential (pre-)tactical update and hence reset the corresponding values to the CACD definition for that period.

(6) Inherits from: `TacticalConfigurationPlan`.

(7) Attributes:

a) **`Map<TrafficVolumeId,Map<DurationHourMinute,OTMVPlanForDuration>>`**
`tvsOTMVs` *(Mandatory)*
Constraints:

i) Size must be comprised between 0 and ∞.

ii) Item size must be comprised between 0 and ∞.

iii) See INCOMPLETE_SCHEDULE

iv) See INVALID_SCHEDULE

v) See ONLY_ONE_ENTRY_CAN_BE_UPDATED_IN_PLAN

(8) Constraints:

a)

Name	INCOMPLETE_SCHEDULE
Attribute	tvsOTMVs
Context	CapacityPlanUpdateRequest, OTMVPlanUpdateRequest, RunwayConfigurationPlanUpdateRequest, SectorConfigurationPlanUpdateRequest, TrafficVolumeActivationPlanUpdateRequest
Description	clientSchedule has gaps and/or overlaps in the time partition or is not covering exactly one day.

b)

Name	ONLY_ONE_ENTRY_CAN_BE_UPDATED_IN_PLAN
Attribute	tvsOTMVs
Context	CapacityPlanUpdateRequest, OTMVPlanUpdateRequest, RunwayConfigurationPlanUpdateRequest, SectorConfigurationPlanUpdateRequest, TrafficVolumeActivationPlanUpdateRequest
Description	Only one entry in the otmvPlans map attribute can be updated (i.e., for one duration).

c)

Name	INVALID_SCHEDULE
Attribute	tvsOTMVs
Context	CapacityPlanUpdateRequest, OTMVPlanUpdateRequest, RunwayConfigurationPlanUpdateRequest, SectorConfigurationPlanUpdateRequest, TrafficVolumeActivationPlanUpdateRequest
Description	The duration key used in the otmvPlans map attribute has to be equal to the duration of all OTMVs linked to that duration key.

(9) Used by: OTMVPlanRetrievalReply, OTMVPlanUpdateRequest, OTMVPlanUpdateReply.

4.94. <<enumeration>> OtmvStatus

(1) Status of a OTMV alert.

(2) Values:

Value	Description
PEAK	Above the peak value
SUSTAINED	"Above" the sustained value (taking into account `OTMVSustained.crossingOccurrences` and `OTMVSustained.elapsed`.

Table 4.3. <<enumeration>> OtmvStatus

(3) Used by: LoadStateAtReferenceLocation, OtmvAlert.

4.95. OTMVSustained

(1) OTMV sustained data.

(2) Attributes:

a) **OTMVThreshold threshold** *(Mandatory)*
Sustained threshold of an OTMV.

b) **int crossingOccurrences** *(Mandatory)*
Number of crossing occurrences of the sustained threshold within `elapsed`, before this OTMV triggers an alert.
Constraint: Range: `[1,9999[`.

c) **DurationHourMinute elapsed** *(Mandatory)*
Duration of the time window on which `crossingOccurrences` are counted.

(3) Used by: OTMV.

4.96. typedef<int> OTMVThreshold

(1) Threshold type used in OTMVs, in flights per user-defined OTMV duration.

(2) Range: `[0,9998[`.

(3) Used by: OTMVSustained, OTMVPeak.

4.97. OTMVWithDuration

(1) Composite object which contains trafficVolume and duration .

(2) Attributes:

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

b) **DurationHourMinute otmvDuration** *(Optional)*
Selects the OTMVs applying for the given traffic volume according to their OTMV duration.

When not specified, all duration are considered.

(3) Used by: OTMVPlanRetrievalRequest.

4.98. <<strict enumeration>> PlanDataSource

(1) A source of data for a plan, within or outside NM.

(2) Values:

a) **AIRSPACE**
Data from the NM Airspace system (CACD), the data is either baselined with the AIRAC or results from a live update.

b) **MEASURE**
Data resulting from the application of a measure in the NM system.

c) **NO_DATA**
There is no data defined in NM.

d) **TACTICAL**
Following a tactical update, typically, from the NOP user (B2B or B2C).

(3) Used by: PlannedOTMV, PlannedCapacity, RunwayConfiguration, PlannedTrafficVolumeActivation, PlannedSectorConfigurationActivation, PlannedRunwayConfigurations.

4.99. PlannedCapacities

(1) Planned Capacities for to be mapped on TrafficVolumeId element in Map .

(2) Attributes:

a) **Set<PlannedCapacity> nmSchedule** *(Contextual)*
The plan resulting from the superimposition of all constraints from all data sources, which the NM system is using as plan for its own calculations.
The NM schedule exposes the complete time partition of the configurations for the day, i.e. the data coming from the various data sources contributing to the NM view of the plan: these data sources are defined in PlanDataSource.
The NM schedule cannot be updated directly by the caller; it is the outcome of updates via the various data sources.
The possible values of dataSource are limited to NO_DATA, AIRSPACE, TACTICAL and MEASURE - the MEASURE value being used to express that the capacity is derived from a regulation.
Presence:

i) Must be null in CapacityPlanUpdateRequest, OTMVPlanUpdateRequest, RunwayConfigurationPlanUpdateRequest, SectorConfigurationPlanUpdateRequest, TrafficVolumeActivationPlanUpdateRequest

ii) Mandatory otherwise.
Constraint: Size must be comprised between 0 and ∞.

b) **Set<PlannedCapacity> clientSchedule** *(Mandatory)*
(Pre-)tactical capacities associated to their applicability period, as maintained by the client. This plan contains only the updated configurations together with an indication that the default CACD values apply when not updated (cf. PlanDataSource). The actual CACD values for these CACD periods can be found in the `nmSchedule`
In an update context, the clientSchedule can be complete (if the B2B client designer decided to overwrite the default CACD values) or limited to the actual differences with regards to the CACD defaults. This is a B2B client designer's decision and depends on how CACD wants to be used in combination with the B2B. If the clientSchedule only contains the actual differences with regards to the CACD defaults, then the clientschedule still needs to contain the full list of (pre-) tactical updates and for the non (pre-) tactically updated periods, an explicit indication that the CACD values need to be used (but without repeating the CACD values themselves). So in any case, the clientschedule needs to be a complete time partition for the full day.
In any case, it is of drastic importance to understand that an entry in the schedule (i.e. a period) overwrites all CACD values in that period.
The possible values of dataSource are limited to `AIRSPACE` and `TACTICAL` - the `AIRSPACE` value meaning that the value associated to the applicabilityPeriod is the CACD one (i.e. CACD or that there is no data defined in the CACD), and the `TACTICAL` value meaning that this plan entry corresponds to the explicit tactical update expressed via capacity.
Constraint: Size must be comprised between 0 and ∞.

(3) Used by: CapacityPlans.

4.100. PlannedCapacity

(1) An entry within a capacity plan.

(2) Attributes:

a) **DateTimeMinutePeriod applicabilityPeriod** *(Mandatory)*
The time period in the plan to which this entry applies.

b) **PlanDataSource dataSource** *(Mandatory)*
The data source of this entry in the plan.
Constraints:

i) See INVALID_DATASOURCE

ii) See VALUE_CANNOT_BE_NULL

iii) See VALUE_MUST_BE_NULL

c) **Capacity capacity** *(Optional)*
Capacity
Constraints:

i) See VALUE_CANNOT_BE_NULL

ii) See VALUE_MUST_BE_NULL

(3) Constraints:

a)

Name	VALUE_CANNOT_BE_NULL
Attributes	dataSource, capacity
Description	capacity cannot be null if dataSource is TACTICAL. For nmSchedule: capacity cannot be null if dataSource is different from NO_DATA.

b)

Name	VALUE_MUST_BE_NULL
Attributes	dataSource, capacity
Description	capacity must be null if dataSource is NO_DATA. For clientSchedule: capacity must be null if dataSource is not TACTICAL.

c)

Name	INVALID_DATASOURCE
Attribute	dataSource
Description	NO_DATA is not a valid dataSource in an update context. MEASURE is not a valid dataSource in an update context.

(4) Used by: PlannedCapacities.

4.101. PlannedOTMV

(1) An entry within an OTMV plan.

(2) Attributes:

a) **DateTimeMinutePeriod applicabilityPeriod** *(Mandatory)*
The time period in the plan to which this entry applies.

b) **PlanDataSource dataSource** *(Mandatory)*
The data source of this entry in the plan.
Constraints:

i) See INVALID_DATASOURCE

ii) See VALUE_CANNOT_BE_NULL

iii) See VALUE_MUST_BE_NULL

c) **OTMV otmv** *(Optional)*
OTMV
Constraints:

i) See VALUE_CANNOT_BE_NULL

ii) See VALUE_MUST_BE_NULL

(3) Constraints:

a)

Name	VALUE_CANNOT_BE_NULL
Attributes	dataSource, otmv
Description	otmv cannot be null if dataSource is TACTICAL. For nmSchedule: otmv cannot be null if dataSource is different from NO_DATA.

b)

Name	VALUE_MUST_BE_NULL
Attributes	dataSource, otmv
Description	otmv must be null if dataSource is NO_DATA. For clientSchedule: otmv must be null if dataSource is not TACTICAL.

c)

Name	INVALID_DATASOURCE
Attribute	dataSource
Description	NO_DATA is not a valid dataSource in an update context. MEASURE is not a valid dataSource in an update context.

(4) Used by: OTMVPlanForDuration.

4.102. PlannedRunwayConfigurations

(1) An entry within an aerodrome runway configuration plan.

(2) Attributes:

a) **DateTimeMinutePeriod applicabilityPeriod** *(Mandatory)*
The time period in the plan to which this entry applies.

b) **PlanDataSource dataSource** *(Mandatory)*
The data source of this entry in the plan.

c) **Set<RunwayConfiguration> runwayConfigurations** *(Optional)*
Full set of runway configurations for an aerodrome within applicabilityPeriod
Constraint: Size must be comprised between 0 and ∞.

(3) Used by: RunwayConfigurationPlan.

4.103. PlannedSectorConfigurationActivation

(1) An entry within a sector configuration plan - its presence in the plan denotes the activation of the associated sector configuration.

(2) Attributes:

a) **DateTimeMinutePeriod applicabilityPeriod** *(Mandatory)*
The time period in the plan to which this entry applies.

b) **PlanDataSource dataSource** *(Mandatory)*
The data source of this entry in the plan.
Constraints:

i) See INVALID_DATASOURCE

ii) See VALUE_CANNOT_BE_NULL

iii) See VALUE_MUST_BE_NULL

c) **SectorConfigurationId sectorConfigurationId** *(Optional)*
Unique Id of the sector configuration, as known in the NM system.
Constraints:

i) See VALUE_CANNOT_BE_NULL

ii) See VALUE_MUST_BE_NULL

(3) Constraints:

a)

Name	VALUE_CANNOT_BE_NULL
Attributes	dataSource, sectorConfigurationId
Description	sectorConfigurationId cannot be null if dataSource is TACTICAL. This is applicable for both nmSchedule and clientSchedule.

b)

Name	VALUE_MUST_BE_NULL
Attributes	dataSource, sectorConfigurationId
Description	sectorConfigurationId must be null if dataSource is not TACTICAL. This is only applicable for the clientSchedule.

c)

Name	INVALID_DATASOURCE
Attribute	dataSource
Description	NO_DATA is not a valid dataSource in an update context.

	MEASURE is not a valid dataSource in an update context.

(4) Used by: SectorConfigurationPlan.

4.104. PlannedTrafficVolumeActivation

(1) Denotes whether a traffic volume is active or not during a period.

(2) Attributes:

a) **DateTimeMinutePeriod applicabilityPeriod** *(Mandatory)*
The time period in the plan to which this entry applies.

b) **PlanDataSource dataSource** *(Mandatory)*
The data source of this entry in the plan.
Constraints:

i) See INVALID_DATASOURCE

ii) See VALUE_CANNOT_BE_NULL

iii) See VALUE_MUST_BE_NULL

c) **boolean active** *(Optional)*
Indicates if the traffic volume is active or not during applicabilityPeriod
Constraints:

i) See VALUE_CANNOT_BE_NULL

ii) See VALUE_MUST_BE_NULL

(3) Constraints:

a)

Name	VALUE_CANNOT_BE_NULL
Attributes	dataSource, active
Description	active cannot be null if dataSource is TACTICAL. For nmSchedule: active must be true if dataSource is AIRSPACE.

b)

Name	VALUE_MUST_BE_NULL
Attributes	dataSource, active
Description	active must be null if dataSource is NO_DATA. For clientSchedule: active must be null if dataSource is not TACTICAL.

c)

Name	INVALID_DATASOURCE

Attribute	dataSource
Description	NO_DATA is not a valid dataSource in an update context. MEASURE is not a valid dataSource in an update context.

(4) Used by: PlannedTrafficVolumeActivations, TrafficVolumeActivationPlan.

4.105. PlannedTrafficVolumeActivations

(1) PlannedTrafficVolumeActivations contains two sets of PlannedTrafficVolumeActivation called client and nm schedules.

(2) Attributes:

a) **Set<PlannedTrafficVolumeActivation> nmSchedule** *(Contextual)*
The plan resulting from the superimposition of all constraints from all data sources, which the NM system is using as plan for its own calculations.
The NM schedule exposes the complete time partition of the configurations for the day, i.e. the data coming from the various data sources contributing to the NM view of the plan: these data sources are defined in PlanDataSource.
The NM schedule cannot be updated directly by the caller; it is the outcome of updates via the various data sources.
In nmSchedule the possible values of dataSource are limited to NO_DATA, AIRSPACE, TACTICAL and MEASURE.
Note that NO_DATA in nmClientSchedule means either inactive or that no data has been specified.
Presence:

i) Must be null in CapacityPlanUpdateRequest, OTMVPlanUpdateRequest, RunwayConfigurationPlanUpdateRequest, SectorConfigurationPlanUpdateRequest, TrafficVolumeActivationPlanUpdateRequest

ii) Mandatory otherwise.

Constraint: Size must be comprised between 0 and ∞.

b) **Set<PlannedTrafficVolumeActivation> clientSchedule** *(Mandatory)*
(Pre-)tactical traffic volume activations associated to their applicability period, as maintained by the client. This plan contains only the updated configurations together with an indication (AIRSPACE datasource) that the default CACD/sector config derived values apply when not updated (cf. PlanDataSource). The actual activity for these airspace datasource periods can be found in the nmSchedule
In an update context, the clientSchedule can be complete (if the B2B client designer decided to overwrite the default CACD values) or limited to the actual differences with regards to the AIRSPACE defaults. This is a B2B client designer's decision and depends on how CACD wants to be used in combination with the B2B. If the clientSchedule only contains the actual differences with regards to the AIRSPACE defaults, then the clientschedule still needs to contain the full list of (pre-) tactical updates and for the non (pre-) tactically updated periods, an explicit indication that the AIRSPACE values need to be used (but without repeating the

NO_DATA/CACD/sector config derived values themselves). So in any case, the clientschedule needs to be a complete time partition for the full day.
In any case, it is of drastic importance to understand that an entry in the schedule (i.e. a period) overwrites all CACD/sector config derived values in that period.
The possible values of dataSource are limited to `AIRSPACE` and `TACTICAL` - the `AIRSPACE` value meaning that the value associated to the applicabilityPeriod is the CACD one (i.e. CACD or the TV activation derived from a sector configuration or that there is no data defined in CACD), and the `TACTICAL` value meaning that this plan entry corresponds to the explicit tactical update expressed via active.
Constraint: Size must be comprised between 0 and ∞.

(3) Used by: TrafficVolumeActivationPlans.

4.106. `PointLocation`

(1) Point location.

(2) Inherits from: `Location`.

4.107. `Regulation`

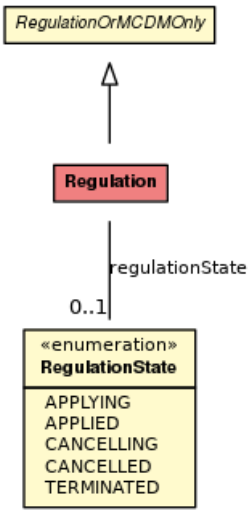

Figure 4.13. `Regulation` Class Diagram

(1) Regulation.

(2) Inherits from: `RegulationOrMCDMOnly`.

(3) Attributes:

a) **`RegulationState regulationState`** *(Contextual)*
Current state of the regulation.
When requested, this attribute is never left null.

Presence:

i) Must be null in RegulationCreationRequest, RegulationUpdateRequest

ii) Optional otherwise.

(4) Used by: RegulationListReply, RegulationMessage, RegulationUpdateRequest, RegulationCreationReply, ScenarioRegulationRetrievalReply, RegulationCancelReply, RegulationCreationRequest, RegulationUpdateReply.

4.108. RegulationCause

(1) Describes the cause of a regulation.

(2) Attributes:

a) **RegulationReason reason** *(Mandatory)*
The reason of the regulations due to this regulation cause.

b) **RegulationLocationCategory locationCategory** *(Mandatory)*
The location category of the regulations due to this regulation cause.

c) **int iataDelayCode** *(Mandatory)*
The IATA delay code of the regulations due to this regulation cause.
Constraint: Range: `[0,∞[`.

(3) Used by: Flight.

4.109. RegulationExceptionalConstraint

(1) Groups all exceptional constraints expressed by a regulation for a specific initial constraint period.

(2) In an initial constraint period, flights can either:

a) Be suspended if the flight is not confirmed (and FCM is mandatory) or if the flight has an insufficient minimum runway visible range

b) Be shifted

c) Use a slot inside the initial constraint period (corresponding to the rates of that period)

(3) Attributes:

a) **DistanceM runwayVisualRange** *(Optional)*
For departure/arrival regulations: minimum visible range in meters for a flight to use slots in the corresponding constraint period. If the flight has an insufficient runway visual range, the flight is either shifted at the end of the constraint period (if shift is true and the flight has a minimum runwayVisualRange), or is suspended (if shift is false or if the flight has no minimum runwayVisualRange)
Constraint:Range: `[0,999[`.

Constraint: See INCONSISTENT_RUNWAY_VISUAL_RANGE_AND_FCM_MANDATORY

b) **boolean fcmMandatory** *(Mandatory)*
Indicates if the flight must be confirmed in this regulation before trying to find a slot for this flight in the associated constraint period. If the flight is not confirmed in this exceptional constraint, the flight is suspended.
Constraint: See INCONSISTENT_RUNWAY_VISUAL_RANGE_AND_FCM_MANDATORY

c) **boolean shift** *(Mandatory)*
If the flight is not suspended due to `fcmMandatory` and no minimum visible range is required, flights affected by this constraint must be shifted to the end of the constraint period when trying to find a slot (and possibly further, depending on next constraint periods). If the flight is not suspended due to `fcmMandatory` but a minimum visible range is required, the flight is shifted if it has an insufficient `runwayVisualRange`.

(4) Constraint:

a)

Name	INCONSISTENT_RUNWAY_VISUAL_RANGE_AND_FCM_MANDATORY
Attributes	fcmMandatory, runwayVisualRange
Context	RegulationCreationRequest, RegulationProposalFilingRequest, RegulationUpdateRequest, RegulationProposalUpdateRequest
Description	If fcmMandatory is true then runwayVisualRange must be null

(5) Used by: RegulationInitialConstraint.

4.110. <<enumeration>> RegulationField

(1) Enumerates the fields that the caller may request to be returned in Regulation objects when returned by RegulationListRequest.

(2) As a rule, client applications should never request regulation fields that they do not need. Client applications typically implement a query/retrieve pattern:

a) Query the small number of most relevant regulation fields to display to the end user

b) Retrieve more details for a given regulation when the end user has selected a regulation from the list

(3) Values:

a) **applicability**

b) **autolink**

c) **createdByFMP**

d) **dataId**

e) **delayConfirmationThreshold**

f) **delayTVSet**

g) **externallyEditable**

h) **initialConstraints**

i) **lastUpdate**

j) **linkedRegulations**

k) **location**

l) **mcdmRequired**

m) **measureCherryPicked**

n) **noDelayWindow**

o) **protectedLocation**

p) **reason**

q) **regulationState**

r) **remark**

s) **scenarioReference**

t) **sourceHotspot**

u) **subType**

v) **supplementaryConstraints**

w) **updateCapacityRequired**

x) **updateTVActivationRequired**

(4) Used by: ScenarioRegulationRetrievalRequest, RegulationListRequest, RegulationPayloadConfiguration.

4.111. typedef<string> RegulationId

(1) Unique id of a regulation measure (inside a given month), allocated by NM or via the user.

(2) Note that there can be 2 regulations with the same RegulationId over a 2 month period.

(3) Pattern: `UALPHA(UALPHA|DIGIT){0,5}DIGIT{2}UALPHA{0,1}`

(4) Used by: UpdateFlightInMeasureChoice, ArrivalInformation, MCDMOnlyCancelRequest, ExcludeReIncludeFlightInRegulation, MCDMStateUpdateRequest, TargetTakeOffAPIRequest, ATFCMSituationRegulation, ForceFlightInRegulation, RegulationCancelRequest, RegulationOrMCDMOnly, MeasureId, Flight, FlightRegulationLocation, TargetTime, FlightAtfcmMcdmOnlyLocation, FlightAtfcmRegulationLocation, RegulationProposalRevocationRequest.

4.112. typedef<string> RegulationIdWildcard

(1) Either a full regulation id, or a simple wildcard for regulation ids.

(2) Pattern: `(UALPHA|DIGIT){1,8}|(UALPHA|DIGIT){0,7}*`

(3) Used by: RegulationOrMCDMOnlyListRequest.

4.113. RegulationInitialConstraint

(1) Groups all the rate information and exceptional constraints expressed by a regulation for a specific initial constraint period.

(2) All rate values are expressed as number of slots per hour.

(3) Attributes:

a) **DateTimeMinutePeriod constraintPeriod** *(Mandatory)*
Applicability period of this constraint.

b) **int normalRate** *(Mandatory)*
Normal rate.
Constraint: Range: `]- ∞,∞[`.

c) **int pendingRate** *(Mandatory)*
Pending rate.
Constraint: Range: `]- ∞,∞[`.

d) **int equipmentRate** *(Mandatory)*
Equipment rate.
Constraint: Range: `]- ∞,∞[`.

e) **RegulationExceptionalConstraint exceptionalConstraint** *(Optional)*
Exceptional constraints expressed by a regulation for a specific initial constraint period.

(4) Used by: RegulationOrMCDMOnly.

4.114. <<enumeration>> RegulationLocationCategory

(1) Enumeration of possible location categories for a regulation.

(2) Values:

a) **ARRIVAL**

b) **DEPARTURE**

c) **ENROUTE**

(3) Used by: RegulationCause.

4.115. <<abstract>> RegulationOrMCDMOnly

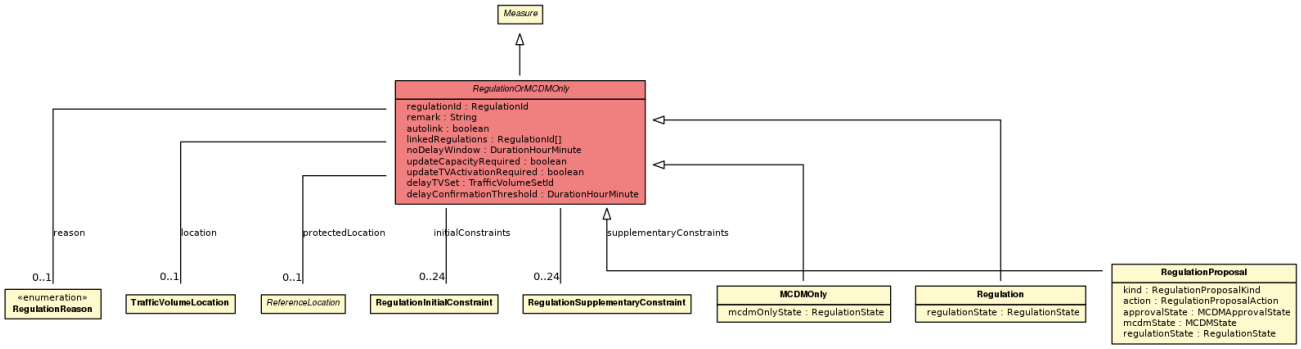

Figure 4.14. <<abstract>> RegulationOrMCDMOnly Class Diagram

(1) Common ancestor of Regulation, RegulationProposal and MCDMOnly.

(2) The `initialConstraints` and `supplementaryConstraints` attributes group all the rate information and exceptional constraints expressed by a regulation.

(3) A regulation rate describes how many slots per hour the regulation can accept, i.e. the maximal number of flights per hour accepted in the regulation. The regulation applicability period is partitioned into one or more regulation "initial constraint periods": different rates and exceptional constraints apply in each initial constraint period. The superset of initial constraint periods must cover the full applicability period, and cannot overlap - it is indeed a time partition.

(4) A regulation can also have different supplementary rates, each defined on a "supplementary period" (i.e. the period associated to a supplementary rate). The supplementary period can overlap partly or fully with one or more initial constraint periods, and with zero or more other supplementary periods. Within the NM system, the supplementary rate is added to the normal rates for the duration of the supplementary period.

(5) There are different kinds of regulation rates:

a) Normal rates: nominal rate

b) Pending rate: rate reserved for late updater, i.e. when a flight changes significantly its profile close to the departure time, the flight has access to pending rate slots (to limit excessive delay deteriorations). Note that pending rate slots are transformed into normal rate slots at some time before the start of the applicability period of the regulation.

c) Equipment rate: rate reserved for flights containing the required MLS equipment in their aircraft equipment set.

(6) Inherits from: Measure.

(7) Attributes:

a) **RegulationId regulationId** *(Mandatory)*
The unique id of the regulation. Regulation ids are unique within an AIRAC cycle and contain at least the day (of the month) of the start date of the regulation. Note that the regulation id is immutable.

b) **RegulationReason reason** *(Contextual)*
Reason that triggered the creation of the regulation.
When requested, this attribute is never left null.
Presence:

i) Mandatory in MCDMOnlyCreationRequest, RegulationCreationRequest, RegulationProposalFilingRequest

ii) Optional otherwise.
Constraint: See INVALID_REASON_DEICING

c) **TrafficVolumeLocation location** *(Contextual)*
Information related to the location to which this regulation applies.
Presence:

i) Mandatory in MCDMOnlyCreationRequest, RegulationCreationRequest, RegulationProposalFilingRequest

ii) Optional otherwise.

d) **ReferenceLocation protectedLocation** *(Contextual)*
Specifies the reference location that the regulation is meant to protect, when the protected reference location is not the reference location specified in location. The `protectedLocation` value is left null if it is the regulation reference specified in location.
Presence:

i) Must be null in MCDMOnlyCreationRequest, MCDMOnlyUpdateRequest, RegulationCreationRequest, RegulationProposalFilingRequest, RegulationProposalUpdateRequest, RegulationUpdateRequest

ii) Optional otherwise.

e) **RegulationInitialConstraint[] initialConstraints** *(Optional)*
Initial constraints.
When requested, this attribute is never left null.
Constraints:

i) Size must be comprised between 0 and 24.

ii) See CHERRY_PICKED_MUST_BE_TRUE

iii) See INCONSISTENT_CHERRY_PICKED_CONSTRAINTS

iv) See INCONSISTENT_CHERRY_PICKED_INITIAL_CONSTRAINTS

f) **RegulationSupplementaryConstraint[] supplementaryConstraints** *(Optional)*
Supplementary constraints.
When requested, this attribute is never left null.
Constraints:

i) Size must be comprised between 0 and 24.

ii) See INCONSISTENT_CHERRY_PICKED_CONSTRAINTS

g) **string remark** *(Contextual)*
Remark made by the NM operations, as provided in the Initial Network Plan. This remark typically provides some more details about the reason of the regulation and refers to any NOTAM if applicable.
Presence:

i) Mandatory in MCDMOnlyCreationRequest, RegulationCreationRequest, RegulationProposalFilingRequest

ii) Optional otherwise.
Constraint: Pattern: `MULTILINE_TEXT{0,128}`

h) **boolean autolink** *(Contextual)*
Indicates if the local delay given by this regulation is taken into account to compute the delay in other regulations.
When requested, this attribute is never left null.
Presence:

i) Mandatory in MCDMOnlyCreationRequest, RegulationCreationRequest

ii) Must be null in RegulationProposalFilingRequest, RegulationProposalUpdateRequest

iii) Optional otherwise.

i) **Set<RegulationId> linkedRegulations** *(Contextual)*
The set of regulations which are linked to this regulation.
When requested, this attribute is never left null. Can be empty.
Presence:

i) Must be null in RegulationProposalFilingRequest, RegulationProposalUpdateRequest

ii) Optional otherwise.
Constraint: Size must be comprised between 0 and 10.

j) **DurationHourMinute noDelayWindow** *(Optional)*

Defines a time window (a.k.a. "window width") around the ETO within which the regulation does not delay the flight.
In certain cases, the window is applied around the minimum CTO.
Constraint: See INVALID_WINDOW_WIDTH

k) **boolean updateCapacityRequired** *(Contextual)*
Indicates that a capacity update will be automatically applied according to the rates of the regulation.
When requested, this attribute is never left null.
Presence:

i) Mandatory in MCDMOnlyCreationRequest, RegulationCreationRequest, RegulationProposalFilingRequest

ii) Optional otherwise.

l) **boolean updateTVActivationRequired** *(Contextual)*
Indicates that a traffic volume activation will be achieved according to the applicability period of the regulation.
When requested, this attribute is never left null.
Presence:

i) Mandatory in MCDMOnlyCreationRequest, RegulationCreationRequest, RegulationProposalFilingRequest

ii) Optional otherwise.

m) **TrafficVolumeSetId delayTVSet** *(Contextual)*
The traffic volume set to which the delay of this regulation is associated.
When requested, this attribute is never left null.
Presence:

i) Must be null in MCDMOnlyCreationRequest, MCDMOnlyUpdateRequest, RegulationCreationRequest, RegulationProposalFilingRequest, RegulationProposalUpdateRequest, RegulationUpdateRequest

ii) Optional otherwise.

n) **DurationHourMinute delayConfirmationThreshold** *(Optional)*
If present, then the regulation has a delayConfirmationThreshold : all flights that have more delay than the `delayConfirmationThreshold` get suspended at EOBT - 2H.

(8) Constraints:

a)

Name	INCONSISTENT_CHERRY_PICKED_CONSTRAINTS
Attributes	measureCherryPicked, initialConstraints, supplementaryConstraints

Context	RegulationProposalFilingRequest, RegulationCreationRequest, RegulationProposalUpdateRequest, RegulationUpdateRequest
Description	If Measure.measureCherryPicked is true then supplementaryConstraints must be null If Measure.measureCherryPicked is false then initialConstraints cannot be null.

b)

Name	INCONSISTENT_CHERRY_PICKED_INITIAL_CONSTRAINTS
Attributes	measureCherryPicked, initialConstraints
Context	RegulationProposalFilingRequest, RegulationCreationRequest, RegulationProposalUpdateRequest, RegulationUpdateRequest
Description	If Measure.measureCherryPicked is true then, for each sub-period of initialConstraints, pendingRate and equipmentRate must be zero, and exceptionalConstraint must be null.

c)

Name	CHERRY_PICKED_MUST_BE_TRUE
Attribute	initialConstraints
Context	MCDMOnlyCreationRequest, MCDMOnlyUpdateRequest
Description	The Measure.measureCherryPicked value must be set to true.

d)

Name	INVALID_REASON_DEICING
Attribute	reason
Context	RegulationCreationRequest, RegulationProposalFilingRequest, RegulationUpdateRequest, RegulationProposalUpdateRequest
Description	The Measure.reason value cannot be DEICING.

e)

Name	INVALID_WINDOW_WIDTH
Attribute	noDelayWindow
Context	RegulationCreationRequest, RegulationProposalFilingRequest, RegulationUpdateRequest, RegulationProposalUpdateRequest
Description	The Measure.noDelayWindow value cannot be 0000.

(9) Extended by: RegulationProposal, Regulation, MCDMOnly.

4.116. RegulationOrMCDMOnlyListReplyData

(1) Abstract reply data associated to the reply of the abstract RegulationOrMCDMOnlyListRequest.

(2) Inherits from: `MeasureListReplyData`.

(3) Extended by: RegulationProposalListReplyData, MCDMOnlyListReplyData, RegulationListReply-Data.

4.117. <<abstract>> RegulationOrMCDMOnlyListRequest

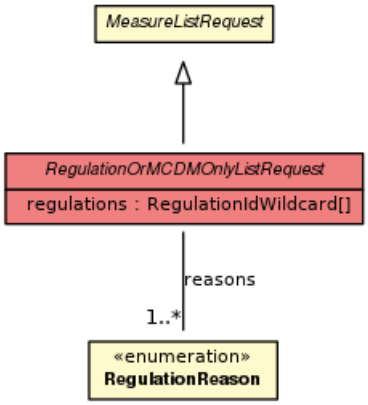

Figure 4.15. <<abstract>> RegulationOrMCDMOnlyListRequest Class Diagram

(1) Abstract request to query an NM regulation list, as well as to retrieve the regulation details. This query method allows the caller to select the regulation fields requested in the reply (see `requestedRegulationFields`). NM kindly requests its customers to apply the following strategy:

a) As a rule, client applications should never request regulation fields that they do not need

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

i) Query the small number of most relevant regulation fields to display to the end user (using this `RegulationListRequest`)

ii) Retrieve more details for a given regulation when the end user has selected a regulation from the list (also using this `RegulationListRequest`, but with other requested fields)

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

(3) Inherits from: `MeasureListRequest`.

(4) Attributes:

a) **`Set<RegulationIdWildcard> regulations`** *(Optional)*
The set of regulation ids or wildcards.
If specified, the reply returns only the requested regulations.
The logical OR operator is meant between the items in the set.
Constraint: Size must be comprised between 1 and 100.

b) **Set<RegulationReason> reasons** *(Optional)*
Selects the regulations with a reason that matches an entry in this set.
By default, regulations are selected regardless to their reason.
Constraint: Size must be comprised between 1 and 14.

(5) Extended by: RegulationListRequest, RegulationProposalListRequest, MCDMOnlyListRequest.

4.118. RegulationProposal

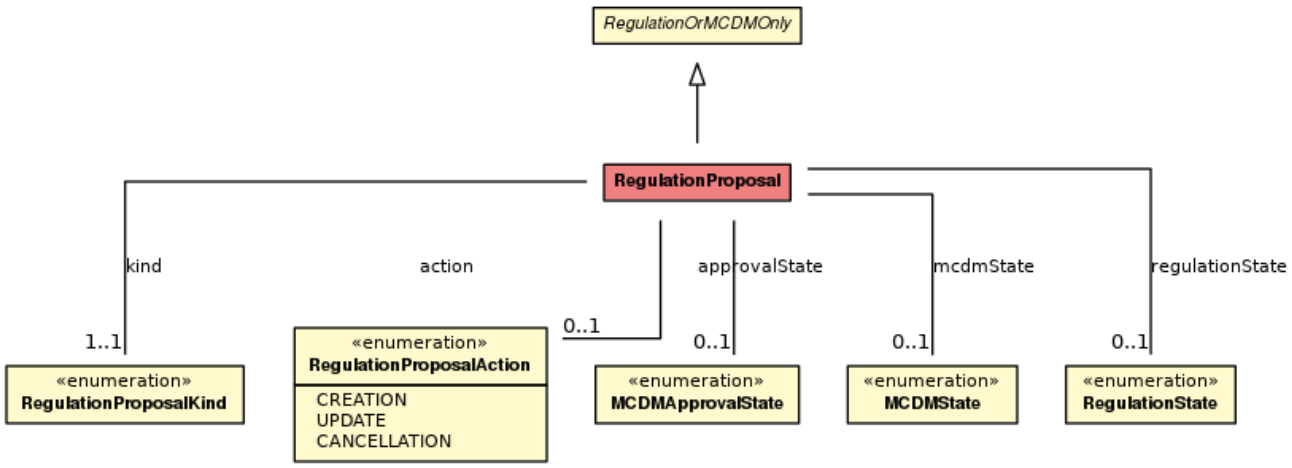

Figure 4.16. `RegulationProposal` Class Diagram

(1) Regulation proposals are proposals to NM for creation/modification/cancellation of a regulation (see RegulationProposalListRequest).

(2) A tailored MCDM process (a subset of the full MCDM process) is used in the proposal regulation and corresponding coordination process.

(3) There are 2 types of proposal regulations:

a) With proposal flights (only supported for cherry picked regulations where the user selects the flights and the delay to attribute to flights): the user proposes a regulation, the MCDM state will be automatically set to draft. Once the regulation has been set to applied (asynchronously by NM systems) , the user can add flights to the regulation (includes giving the desired `CalculatedTakeOffTime` or `CalculatedTimeOver`).
Once the flights have been added (synchronously), he user can inspect the desired result in the flightlist and counts (with `includeProposalFlights`).
If the user is not satisfied with the result, he can remove some flights or add some more flights or modify/revoke the proposal (via `updateRegulationProposal` or `revokeRegulationProposal`).
Once the user is happy with the results, he should modify the MCDM state to proposed so that it becomes visible to NM.
Once NMOC starts assessing the proposal, the NM actor `approvalState` is set to standby (i.e. acknowledged) and the MCDMState is set to COORDINATED..

When NM accepts (and has implemented the regulation), the NM actor `approvalState` is set to approved. In addition the MCDM state is set to IMPLEMENTED.
If NM rejects the proposal, the NM actor state is set to rejected and the MCDM state is set to ABANDONED. In this case the user can file a new regulation proposal (resetting the MCDM state to draft).

b) Without proposal flights:
When the user proposes a regulation, the MCDM state will be automatically set to PROPOSED (visible to NM). Once NMOC starts assessing the proposal, the NM actor `approvalState` is set to STANDBY (i.e. ACKNOWLEDGED).
When NM accepts (and has implemented the regulation), the NM actor `approvalState` is set to APPROVED. In addition the MCDM state is set to IMPLEMENTED.
If NM rejects the proposal, the NM actor state is set to rejected and the MCDM state is set to ABANDONED. In this case the user can file a new regulation proposal (resetting the MCDM state to PROPOSED).

(4) Inherits from: `RegulationOrMCDMOnly`.

(5) Attributes:

a) **`RegulationProposalKind`** **`kind`** *(Mandatory)*

b) **`RegulationProposalAction`** **`action`** *(Contextual)*
Describes whether the proposal is to create a new regulation, or to update or cancel an existing one.
Once created, the action of the regulation proposal is immutable.
In order to replace an update by a cancellation, the proposal must be revoked and a new proposal must be filed or alternatively NMOC needs to first accept or reject the proposal.
Note that the revoked proposal remains in the NM system (as revoked), until a new proposal is made for the same regulation, in which case the revoked proposal is replaced.
When requested, this attribute is never left null.
Presence:

i) Mandatory in `RegulationProposalFilingRequest`, `RegulationProposalUpdateRequest`

ii) Optional otherwise.

c) **`MCDMApprovalState`** **`approvalState`** *(Contextual)*
Describes the measure approval state (not to be confused with the regulation state) of the MCDM NMOC actor: a.o did NM accept (and implement) the regulation proposal.
When requested, this attribute is never left null.
Presence:

i) Must be null in `RegulationProposalFilingRequest`, `RegulationProposalUpdateRequest`

ii) Optional otherwise.

d) **MCDMState mcdmState** *(Contextual)*
Describes the regulation proposal MCDM state: a.o is the measure in draft (not visible to NM yet) or proposed or implemented or interrupted (a.o. the measure was cancelled after it had been implemented).
When requested, this attribute is never left null.
When a regulation proposal is filed, the MCDM state is (re-)set by the system.
When the client has filed the request (and optionally added flights to the cherry picked regulation) and the client is happy with the results, then he should set the `mcdmState` to proposed (to make the proposal visible to NM)): for `RegulationProposalWithProposalFlights` only.
NM will then accept or reject the `regulationProposal` and the MCDM state will go to implemented (in case of accept)/interrupted (in case of reject).
Note that even though the regulation proposal can be accepted by NM, it does not mean that all of the individual flights have been accepted (in case of cherry picked regulation proposal). In order to retrieve for each individual flight, its MCDM state and the NMOC actor approval state, the `queryMCDM` service can be used.
In addition, interrupted or abandoned MCDM state for a flight (i.e. in a cherry picked regulation) means that the flight got "un-picked" from that regulation (e.g. due to changes in the flight such that it's forced slot could not be kept).
Presence:

i) Must be null in `RegulationProposalFilingRequest`, `RegulationProposalUpdateRequest`

ii) Optional otherwise.

e) **RegulationState regulationState** *(Contextual)*
Current state of the regulation.
When requested, this attribute is null when `kind = ProposalRegulationWithoutProposalFlights`.
Presence:

i) Must be null in `RegulationProposalFilingRequest`, `RegulationProposalUpdateRequest`

ii) Optional otherwise.

(6) Used by: RegulationProposalRevocationReply, RegulationProposalUpdateReply, RegulationProposalListReply, RegulationProposalFilingRequest, RegulationProposalUpdateRequest, RegulationProposalFilingReply.

4.119. <<enumeration>> RegulationProposalAction

(1) Action requested by a regulation proposal.

(2) Values:

Value	Description
CANCELLATION	The regulation proposal requests NMOC to cancel a regulation. When NMOC accepts this will result in a cancelled regulation and all the concerned flights will be unforced or de-regulated.
CREATION	The regulation proposal requests NMOC to create a regulation
UPDATE	The regulation proposal requests NMOC to update a regulation

Table 4.4. <<enumeration>> RegulationProposalAction

(3) Used by: RegulationProposal.

4.120. <<enumeration>> RegulationProposalField

(1) Enumerates the fields that the caller may request to be returned in RegulationProposal objects when returned by RegulationProposalListRequest.

(2) As a rule, client applications should never request regulation proposal fields that they do not need. Client applications typically implement a query/retrieve pattern:

a) Query the small number of most relevant regulation proposal fields to display to the end user

b) Retrieve more details for a given regulation proposal when the end user has selected a regulation proposal from the list

(3) Values:

a) **action**

b) **applicability**

c) **approvalState**

d) **autolink**

e) **createdByFMP**

f) **dataId**

g) **delayConfirmationThreshold**

h) **delayTVSet**

i) **externallyEditable**

j) **initialConstraints**

k) **lastUpdate**

l) **linkedRegulations**

m) **location**

n) **mcdmRequired**

o) **mcdmState**

p) **measureCherryPicked**

q) **noDelayWindow**

r) **protectedLocation**

s) **reason**

t) **regulationState**

u) **remark**

v) **scenarioReference**

w) **sourceHotspot**

x) **subType**

y) **supplementaryConstraints**

z) **updateCapacityRequired**

aa) **updateTVActivationRequired**

(4) Used by: RegulationProposalListRequest.

4.121. <<enumeration>> RegulationProposalKind

(1) Values:

a) **RegulationProposalWithProposalFlights**

b) **RegulationProposalWithoutProposalFlights**

(2) Used by: RegulationProposal.

4.122. <<enumeration>> RegulationProposalState

(1) State of a regulation proposal.

(2) Values:

Value	Description
ACCEPTED	The regulation proposal has been accepted by NMOC
REJECTED	The regulation proposal has been rejected by NMOC
WAITING	The regulation proposal has not been processed by NMOC yet

Table 4.5. <<enumeration>> RegulationProposalState

4.123. <<enumeration>> RegulationReason

(1) Enumeration of possible reasons for a regulation.

(2) Values:

a) **ACCIDENT_INCIDENT**

b) **AERODROME_CAPACITY**

c) **AERODROME_SERVICES**

d) **AIRSPACE_MANAGEMENT**

e) **ATC_CAPACITY**

f) **ATC_EQUIPMENT**

g) **ATC_INDUSTRIAL_ACTION**

h) **ATC_ROUTINGS**

i) **ATC_STAFFING**

j) **ENVIRONMENTAL_ISSUES**

k) **NON_ATC_INDUSTRIAL_ACTION**

l) **OTHERS**

m) **SPECIAL_EVENT**

n) **WEATHER**

(3) Used by: ATFCMSituationDelays, RegulationOrMCDMOnly, DeltaATFCMSituationDelays, RegulationOrMCDMOnlyListRequest, RegulationCause, ATFCMSituationRegulation.

4.124. <<enumeration>> RegulationState

(1) Enumerates the possible states of a regulation in the NM system.

(2) When created, a regulation starts in the APPLYING state. Once all the flights have been successfully updated, the regulation goes to the APPLIED state. Any subsequent modification to the regulation will put it back to the APPLYING state. Again, once all the flights have been successfully updated, the regulation will go again to the APPLIED state. Once the applicability period is passed, the regulation goes to the TERMINATED state. If the regulation is no longer needed while the applicability period is still in the future (partly or completely), then the regulation state is set to CANCELLING. Once all the flights have been successfully updated, the regulation is set to CANCELLED. CANCELLED and TERMINATED are final regulation states.

(3) Values:

Value	Description
APPLIED	The regulation is activated and the subsequent flight recalculation is finished
APPLYING	The regulation is activated but the subsequent flight recalculation is not finished yet
CANCELLED	The regulation is cancelled and the subsequent flight recalculation is finished
CANCELLING	The regulation is cancelled but the subsequent flight recalculation is not finished yet
TERMINATED	The regulation is terminated

Table 4.6. <<enumeration>> RegulationState

(4) Used by: MCDMOnly, RegulationProposal, MCDMOnlyListRequest, Regulation, RegulationListRequest, ATFCMSituationRegulation.

4.125. RegulationSupplementaryConstraint

(1) Supplementary rate information expressed by a regulation for a specific supplementary period, expressed as number of slots per hour.

(2) Attributes:

a) **DateTimeMinutePeriod constraintPeriod** *(Mandatory)*
Applicability period of this constraint.

b) **int supplementaryRate** *(Mandatory)*
Supplementary rate.
Constraint: Range:]- ∞,∞[.

(3) Used by: RegulationOrMCDMOnly.

4.126. <<enumeration>> RepositoryId

(1) Repository Id.

(2) Values:

a) **PREDICT**

b) **SCENARIO**

c) **TACT**

4.127. Rerouting

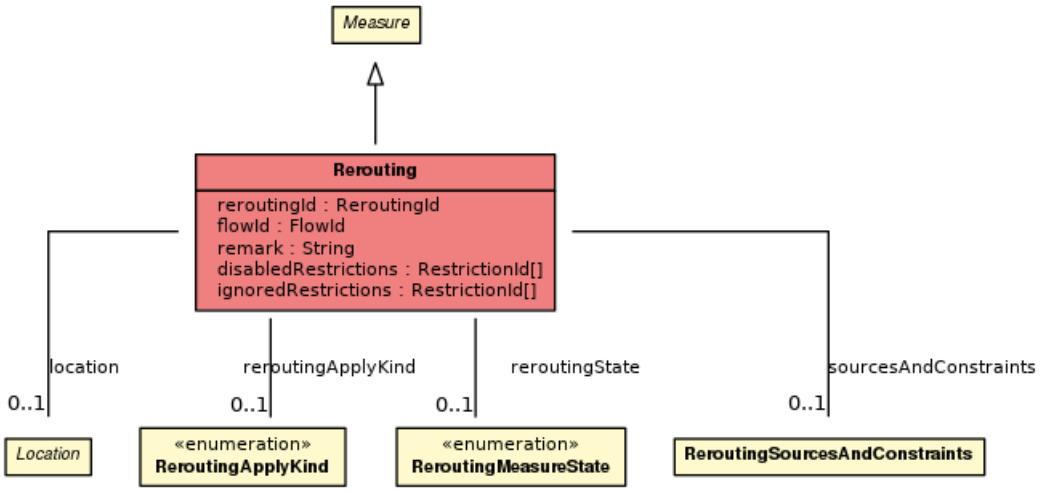

Figure 4.17. `Rerouting` Class Diagram

(1) Reroutings are measures to level cap or reroute flights to avoid an airspace/point or to find shorter/cheaper routes. Typically they are used for ATFCM reasons (for example to avoid a zero rate regulation) or for STAM or for Flight Efficiency (to find more efficient routes) or to handle forecast expected flows (for example NAT traffic). Rerouting can either create a proposal flight (containing a proposed route) or they can modify the FTFM/RTFM point profile directly (used in forecast and simulations) or they can generate proposed routes in operational log messages.

(2) Inherits from: `Measure`.

(3) Attributes:

a) **`ReroutingId` `reroutingId`** *(Mandatory)*
The unique id of the rerouting.

b) **`Location` `location`** *(Contextual)*
The type of location used to select the flights.
When requested, this attribute is never left null.
Presence:

i) Mandatory in `ReroutingCreationRequest`

ii) Optional otherwise.
Constraint: See `ONLY_TRAFFIC_VOLUME_LOCATION_TYPE_PERMITTED`

c) **FlowId flowId** *(Optional)*
The traffic flow to/from the CountLocation to wich this rerouting applies.
if null it applies to all the traffic for the CountLocation.
Event if Flow is requested, it can be that no filtering flow is used.
When requested, this attribute is never left null.

d) **ReroutingApplyKind reroutingApplyKind** *(Contextual)*
The apply kind of the rerouting.
When requested, this attribute is never left null.
Presence:

i) Mandatory in ReroutingCreationRequest

ii) Optional otherwise.

e) **ReroutingMeasureState reroutingState** *(Contextual)*
Enumerates the states of the rerouting.
When requested, this attribute is never left null.
Presence:

i) Must be null in ReroutingCreationRequest, ReroutingUpdateRequest

ii) Optional otherwise.

f) **ReroutingSourcesAndConstraints sourcesAndConstraints** *(Contextual)*
Describes the rerouting constraints.
When requested, this attribute is never left null.
Presence:

i) Mandatory in ReroutingCreationRequest

ii) Optional otherwise.

g) **string remark** *(Optional)*
Remark made by the NM or FMPs . This remark typically provides some more details about the reason and the description of the rerouting.
Constraint: Pattern: TEXT{1,128}((WHITESPACE)(TEXT{1,128})){0,1}

h) **Set<RestrictionId> disabledRestrictions** *(Optional)*
RAD restrictions that need to be disabled (ENV) before the scenario can be applied.
Constraint: Size must be comprised between 0 and ∞.

i) **Set<RestrictionId> ignoredRestrictions** *(Optional)*
RAD restrictions that need to be ignored (IFPS before the scenario can be applied.
Constraint: Size must be comprised between 0 and ∞.

(4) Constraint:

a)

Name	ONLY_TRAFFIC_VOLUME_LOCATION_TYPE_PERMITTED

Attribute	location
Description	Only the TrafficVolumeLocation type is permitted in ReroutingCreationRequest and ReroutingUpdateRequest. Note that when retrieving reroutings, there can be other types of location.

(5) Used by: ReroutingUpdateRequest, ReroutingCreationRequest, ReroutingListReply, ReroutingCreationReply, ScenarioReroutingRetrievalReply, ReroutingUpdateReply, ReroutingCancelReply.

4.128. <<enumeration>> ReroutingApplyKind

(1) Rerouting apply kind. Indicates if it concerns a rerouting for indication or not.

(2) If it concerns a rerouting for indication, the generated proposed routes can be found in the operational log messages. In addition, in the flight list, the field reroutingOpportunitiesExist can be used to determine if the rerouting succeed to find a new (interesting) route or not.

(3) Values:

a) **EXECUTE**
Indicates it concerns an execute rerouting.
In this case either the FTFM or RTFM are replaced by the rerouting.
No proposal flights are created.
This rerouting applies to flights in a simulation or flights in status planned (typically all flights on the pre-tactical dataset)

b) **FOR_INDICATION_WITHOUT_AUTOMATIC_PROPOSAL_FLIGHT**
Indicates it concerns a rerouting for indication.
No proposal flights are created.

c) **FOR_INDICATION_WITH_AUTOMATIC_RRN**
Indicates it concerns a rerouting for indication.
In addition proposal flights are created and the corresponding RRN messages are or will be sent.

d) **FOR_INDICATION_WITH_AUTOMATIC_RRP**
Indicates it concerns a rerouting for indication.
In addition proposal flights are created and the corresponding RRP messages are or will be sent.

(4) Used by: FlightAtfcmReroutingLocation, Rerouting.

4.129. <<abstract>> ReroutingConstraint

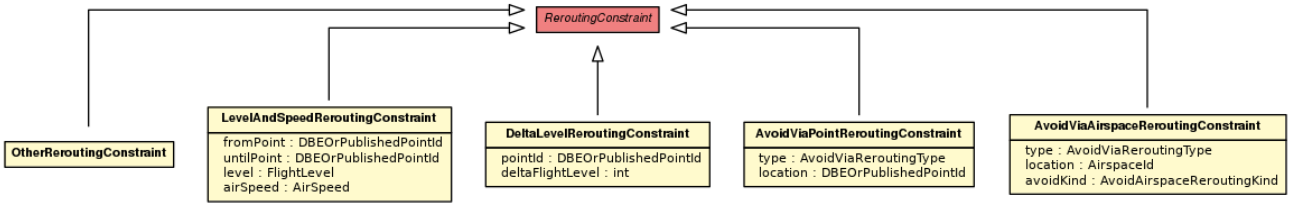

Figure 4.18. <<abstract>> ReroutingConstraint Class Diagram

(1) Ancestor structure for all rerouting constraints.

(2) Extended by: OtherReroutingConstraint, DeltaLevelReroutingConstraint, AvoidViaAirspaceReroutingConstraint, LevelAndSpeedReroutingConstraint, AvoidViaPointReroutingConstraint.

(3) Used by: ANDedReroutingConstraints.

4.130. <<enumeration>> ReroutingField

(1) Enumerates the fields that the caller may request to be returned in Rerouting objects when returned by ReroutingListRequest.

(2) As a rule, client applications should never request rerouting fields that they do not need. Client applications typically implement a query/retrieve pattern:

a) Query the small number of most relevant rerouting fields to display to the end user

b) Retrieve more details for a given rerouting when the end user has selected a rerouting from the list

(3) Values:

a) **applicability**

b) **constraints**

c) **createdByFMP**

d) **dataId**

e) **disabledRestrictions**

f) **externallyEditable**

g) **flowId**

h) **ignoredRestrictions**

i) **lastUpdate**

j) **location**

k) **mcdmRequired**

l) **measureCherryPicked**

m) **remark**

n) **reroutingApplyKind**

o) **reroutingState**

p) **scenarioReference**

q) **sourceHotspot**

r) **subType**

(4) Used by: ReroutingListRequest, ScenarioReroutingRetrievalRequest.

4.131. ReroutingHorizontalConstraints

(1) Describes the constraints for a horizontal/lateral rerouting.

(2) Attributes:

a) **boolean freezeOutsideZone** *(Mandatory)*
Indicates whether the Freeze Outside IFPZ option is triggered.

b) **FreezeTP freezeTPs** *(Mandatory)*
Indicates the Freeze Terminal procedures kind.

c) **boolean useRouteGenerator** *(Mandatory)*
Generate horizontal route alternatives based on the route generator.
Constraints:

i) See AT_LEAST_ONE_USE_SOURCE_MUST_BE_SET_TO_TRUE

ii) See USE_PATH_FINDER_INCONSISTENCY_PAIR

d) **boolean forRouteGeneratorUseDctAsReference** *(Optional)*
If yes then replace the F15 of the reference by DCT.
Constraint: See USE_PATH_FINDER_INCONSISTENCY_PAIR

e) **boolean useRouteCatalogue** *(Mandatory)*
Generate horizontal route alternatives based on the route catalogue.
Constraint: See AT_LEAST_ONE_USE_SOURCE_MUST_BE_SET_TO_TRUE

f) **`boolean useCurrentFlightsInSystem`** *(Optional)*
Generate horizontal route alternatives based on the flights currently present in NM systems.
Constraint: See AT_LEAST_ONE_USE_SOURCE_MUST_BE_SET_TO_TRUE

(3) Constraints:

a)

Name	AT_LEAST_ONE_USE_SOURCE_MUST_BE_SET_TO_TRUE
Attributes	useRouteGenerator, useRouteCatalogue, useCurrentFlightsIn-System
Description	At least one of 'useRouteGenerator', 'useRouteCatalogue', 'useCurrentFlightsIn-System' needs to be set to true.

b)

Name	USE_PATH_FINDER_INCONSISTENCY_PAIR
Attributes	forRouteGeneratorUseDctAsReference, useRouteGenerator
Description	if 'forRouteGeneratorUseDctAsReference' is set either to false or to true then 'useRouteGenerator' must be set to true. if 'forRouteGeneratorUseDctAsReference' is not set then 'useRouteGenerator' must be set to false.

(4) Used by: ReroutingSourcesAndConstraints.

4.132. `typedef<string> ReroutingId`

(1) Unique id of a rerouting measure (for a given day), allocated by NM or via the user.

(2) Note that there can be 2 reroutings with the same ReroutingId over a 2 day period.

(3) Pattern: `(UALPHA|DIGIT|*){1,8}`

(4) Used by: UpdateFlightInMeasureChoice, FlightAtfcmReroutingLocation, MeasureId, Rerouting, ReroutingCancelRequest.

4.133. `typedef<string> ReroutingIdWildcard`

(1) Either a full rerouting id, or a simple wildcard for rerouting ids.

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

(3) Used by: ReroutingListRequest.

4.134. `ReroutingManualConstraints`

(1) Describes a manual only rerouting constraint.

(2) Attributes:

a) **`Set<string> manualField15OrField15Portions`** *(Mandatory)*
Describes the `Field15`s or Partial `Field15`s that are ORed together with the other rerouting constraints: for each concerned flight, each of different `Field15` are evaluated together with the other applicable rerouting constraints and the best solution is selected (according to cost criteria)
The manual routes are in ICAO field 15 syntax. Either a complete field15abc can be entered which will replace the original flight route or a (partial) field15c only can be entered. In the latter case the new entered portion will replace the matching portion of the original flight route. The begin and end point (reconnection points) of this portion must be present in the correct order in the original flight route otherwise an error is shown for a flight if none of the given F15 reconnection points match this specific flight. The system will use the RFLs and RSPs specified in the input ICAO field 15, during the profile calculation for the alternative (both for complete field15abc and partial field15c.
When for a partial field15, the reconnection point is either ADEP or ADES, then the special point indicator "ADEP" or "ADES" can be used. If the reconnection point is both ADEP and ADES, then a complete field15abc needs to be entered. When for a partial field15, the speed of the flight does not need to be updated, then the special point indicator "NOSPEED" can be used to indicate that the speeds indicators of the manual F15 do not need to be taken into account. The NOSPEED indicator needs to be the first word of the partial F15. If the ADEP indicator is specified, then the NOSPEED needs to be the second keyword of the F15.
Constraint: Size must be comprised between 0 and ∞.

(3) Used by: ReroutingSourcesAndConstraints.

4.135. <<enumeration>> ReroutingMeasureState

(1) Enumerates the possible state of a rerouting in the NM system.

(2) Values:

a) **`ACTIVATING`**

b) **`APPLIED`**

c) **`CANCELLED`**

d) **`CANCELLING`**

(3) Used by: Rerouting, ReroutingListRequest.

4.136. `ReroutingSourcesAndConstraints`

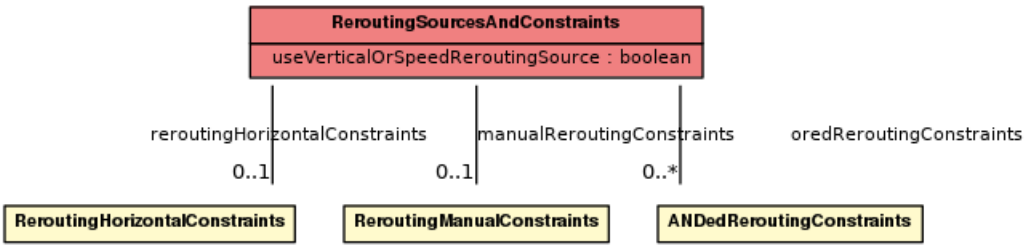

Figure 4.19. `ReroutingSourcesAndConstraints` Class Diagram

(1) Describes the constraints for a all kind of rerouting.

(2) Attributes:

a) **`ReroutingHorizontalConstraints` `reroutingHorizontalConstraints`** *(Optional)*
Describes the constraints for a horizontal/lateral rerouting.
Constraint: See `AT_LEAST_ONE_CONSTRAINT_TYPE_MUST_BE_SET`

b) **`boolean useVerticalOrSpeedReroutingSource`** *(Mandatory)*
Indicates weather the constraints is for a non horizontal rerouting (i.e. vertical/or speed adaptations).
Constraint: See `AT_LEAST_ONE_CONSTRAINT_TYPE_MUST_BE_SET`

c) **`ReroutingManualConstraints` `manualReroutingConstraints`** *(Optional)*
manual only rerouting constraints
Constraint: See `AT_LEAST_ONE_CONSTRAINT_TYPE_MUST_BE_SET`

d) **`Set<ANDedReroutingConstraints>` `oredReroutingConstraints`** *(Optional)*
A set of rerouting constraints between which the OR operator applies.
Constraint: Size must be comprised between 0 and ∞.

(3) Constraint:

a)

Name	`AT_LEAST_ONE_CONSTRAINT_TYPE_MUST_BE_SET`
Attributes	`reroutingHorizontalConstraints`, `useVerticalOrSpeedReroutingSource`, `manualReroutingConstraints`
Description	At least one of 'reroutingHorizontalConstraints', 'useVerticalOrSpeedReroutingSource', 'manualReroutingConstraints' needs to be set.

(4) Used by: Rerouting.

4.137. RestrictionLocation

(1) Restriction location.

(2) Inherits from: `Location`.

(3) Attributes:

a) **`KindOfRestriction` `kindOfRestriction`** *(Mandatory)*
The kind of restriction.

4.138. RunwayConfiguration

(1) A runway configuration at some point in time.

(2) Attributes:

a) **`RunwayId` `runway`** *(Mandatory)*
Runway to which this configuration applies.

b) **`RunwayUsage` `usage`** *(Optional)*
Usage of the runway in this configuration.

c) **`PlanDataSource` `runwayUsageDataSource`** *(Mandatory)*
The data source of the `runwayUsage` for this entry in the plan.
Note that NM prefers not to have any CACD (runwayUsageDataSource = AIRSPACE) runway usage when tactically updating runway configurations. If an AIRSPACE runwayUsageDataSource applies, then this means that in NM profile computations, the RunwayUsage is considered unknown (as it is difficult to predict accurately up to one month in advance the wind/runway activation schedule).

d) **`DurationHourMinute` `departureTaxiTime`** *(Optional)*
Departure taxi time.
Note that the departure taxi time must be specified even if the `usage` is `ARRIVAL` or `INACTIVE`, so that in the exceptional case where the runway would be used in another way (indicated via `DPI`) the NM system could still use this taxi time for computing the flight.

e) **`PlanDataSource` `departureTaxiTimeDataSource`** *(Mandatory)*
The data source of the `departureTaxiTime` for this entry in the plan.

f) **`DurationHourMinute` `timeToInsertInSequence`** *(Optional)*
Time to insert aircraft in sequence at the aerodrome of departure.
Note that the `timeToInsertInSequence` must be specified even if the `usage` is `ARRIVAL` or `INACTIVE`, so that in the exceptional case where the runway would be used in another way (indicated via `DPI`) the NM system could still use this taxi time for computing the flight.

g) **`PlanDataSource` `timeToInsertInSequenceDataSource`** *(Mandatory)*
The data source of the `timeToInsertInSequence` for this entry in the plan.

h) **DurationHourMinute timeToRemoveFromSequence** *(Optional)*
Time to remove aircraft from sequence at the aerodrome of departure.
Note that the `timeToRemoveFromSequence` must be specified even if the `usage` is `ARRIVAL` or `INACTIVE`, so that in the exceptional case where the runway would be used in another way (indicated via `DPI`) the NM system could still use this taxi time for computing the flight.

i) **PlanDataSource timeToRemoveFromSequenceDataSource** *(Mandatory)*
The data source of the `timeToRemoveFromSequence` for this entry in the plan.

j) **DurationHourMinute arrivalTaxiTime** *(Optional)*
Arrival taxi time.
Note that the departure taxi time must be specified even if the `usage` is `DEPARTURE` or `INACTIVE`.

k) **PlanDataSource arrivalTaxiTimeDataSource** *(Mandatory)*
The data source of the `arrivalTaxiTime` for this entry in the plan.

(3) Used by: PlannedRunwayConfigurations.

4.139. RunwayConfigurationPlan

(1) Runway configuration plan for a given aerodrome on a given day.

(2) A RunwayConfiguration plan is a special plan in the sense that all known runways need to be specified for a given period (PlannedRunwayConfiguration) or else the entire period needs to be marked CACD (AIRSPACE PlanDataSource). In addition, for a (pre-)tactically updated period it is possible to (pre-)tactically update selected runway attributes for selected runways. So each runway attribute (usage, departureTaxTime,...) has an associated dataSource attribute allowing to indicate if that attribute needs to be updated or if CACD data is to be used for that attribute, for that runway and for that specific period. The only constraint is: if a period (PlannedRunwayConfiguration) is marked as TACTICAL updated, then at least one of the attributes of one of the runways needs to be TACTICAL updated (otherwise INVALID_INPUT replyStatus).

(3) In addition the RunwayUsage attribute is special in the sense that all runways for a specific TACTICAL updated period need to be either all CACD, or else a mix of DEPARTURE/ARRIVAL/DEPARTURE_ARRIVALI/INACTIVE).

(4) Note that NM prefers not to have any CACD (AIRSPACE datasource) RunwayUsage when tactically updating runway configurations. If a CACD RunwayUsage applies, then this means that in NM profile computations, the RunwayUsage is considered unknown (as it is difficult to predict accurately up to one month in advance the wind/runway activation schedule).

(5) In a retrieval context, the plan is said to be 'complete' in the sense that it contains all the plan entries from all involved data sources.

(6) In an update context, the plan can be complete (if the B2B client designer decided to overwrite the default CACD values) or limited to the (full list) of (pre-)tactical updates with the gaps marked as AIRSPACE (meaning in update context: CACD) datasource (to obtain a complete time partition).

(7) In any case, periods in the time partition marked with datasource AIRSPACE, correspond to removing any potential (pre-)tactical update and hence reset the corresponding values to the CACD definition for that period.

(8) Inherits from: TacticalConfigurationPlan.

(9) Attributes:

a) **AerodromeICAOId aerodrome** *(Mandatory)*
Aerodrome to which this runway configuration plan belongs.

b) **Set<RunwayId> knownRunwayIds** *(Mandatory)*
The list of runways that NM knows for the aerodrome, regardless to their usage, e.g. an inactive runway is part of this set.
Constraint: Size must be comprised between 0 and ∞.

c) **Set<PlannedRunwayConfigurations> nmSchedule** *(Contextual)*
The plan resulting from the superimposition of all constraints from all data sources, which the NM system is using as plan for its own calculations.
The NM schedule exposes the complete time partition of the configurations for the day, i.e. the data coming from the various data sources contributing to the NM view of the plan: these data sources are defined in PlanDataSource.
The NM schedule cannot be updated directly by the caller; it is the outcome of updates via the various data sources.
Presence:

i) Must be null in CapacityPlanUpdateRequest, OTMVPlanUpdateRequest, RunwayConfigurationPlanUpdateRequest, SectorConfigurationPlanUpdateRequest, TrafficVolumeActivationPlanUpdateRequest

ii) Mandatory otherwise.

Constraint: Size must be comprised between 0 and ∞.

d) **Set<PlannedRunwayConfigurations> clientSchedule** *(Mandatory)*
(Pre-)tactical runway configuration of the aerodrome associated to their applicability period, as maintained by the client. This plan contains only the updated configurations together with an indication that the default CACD values apply when not updated (cf. PlanDataSource). The actual CACD values for these CACD periods can be found in the `nmSchedule`.
In an update context, the clientSchedule can be complete (if the B2B client designer decided to overwrite the default CACD values) or limited to the actual differences with regards to the CACD defaults. This is a B2B client designer's decision and depends on how CACD wants to be used in combination with the B2B. If the clientSchedule only contains the actual differences with regards to the CACD defaults, then the clientschedule still needs to contain the full list of (pre-) tactical updates and for the non (pre-) tactically updated periods, an explicit indication that the CACD values need to be used (but without repeating the CACD values themselves). So in any case, the clientschedule needs to be a complete time partition for the full day.
In any case, it is of drastic importance to understand that an entry in the schedule (i.e. a period):

i) Must be complete in the sense that all the aerodrome runways must be present in the entry

ii) The runway configuration values provided for that period overwrite all CACD values for that same period

Constraints:

i) Size must be comprised between 0 and ∞.

ii) See INCOMPLETE_SCHEDULE

(10) Constraint:

a)

Name	INCOMPLETE_SCHEDULE
Attribute	clientSchedule
Description	clientSchedule has gaps and/or overlaps in the time partition or is not covering exactly one day.

(11) Used by: RunwayConfigurationPlanUpdateRequest, RunwayConfigurationPlanUpdateReply, RunwayConfigurationPlanRetrievalReply.

4.140. <<enumeration>> RunwayUsage

(1) Possible usages of a runway.

(2) Values:

a) **ARRIVAL**
The runway is used for arrivals only.
Note that an arrival runway is not considered by the NM system for departure when processing flights, but DPI (Departure Planning Information) messages are still able to use an arrival runway.

b) **DEPARTURE**
The runway is used for departures only.

c) **DEPARTURE_ARRIVAL**
The runway is used for both departures and arrivals.

d) **INACTIVE**
The runway is neither used for depatures or arrivals.
Note that an inactive runway is not considered by the NM system for departure when processing flights, but DPI (Departure Planning Information) messages are still able to use an inactive runway.

(3) Used by: RunwayConfiguration.

4.141. ScenarioAttributes

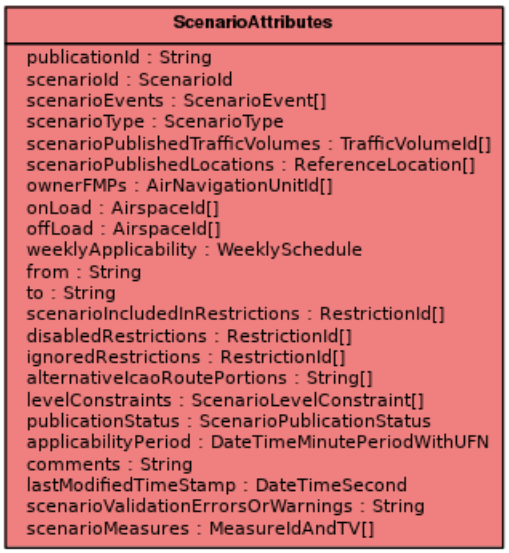

Figure 4.20. `ScenarioAttributes` Class Diagram

(1) Scenario attributes.

(2) Attributes:

a) **`string publicationId`** *(Mandatory)*
The published scenarioId. For AR,RR,FL type of scenario it is typically the traffic volume name of the scenario.
Constraint: Pattern: `TEXT{1,10000}`

b) **`ScenarioId scenarioId`** *(Mandatory)*

c) **`Set<ScenarioEvent> scenarioEvents`** *(Optional)*
The events associated by the user to this scenario: typically Strike, SKI, ConingencyMUAC
Constraint: Size must be comprised between 0 and ∞.

d) **`ScenarioType scenarioType`** *(Mandatory)*

e) **`Set<TrafficVolumeId> scenarioPublishedTrafficVolumes`** *(Optional)*
The trafic volumes(s) of the scenario: i.e. the traffic volume(s) to avoid. For RR/FL type of scenario, it is typically the traffic volume of the scenario measures (but not always: in some cases the scenario measures use a larger traffic volume to avoid/protect some other traffic volume).
Constraint: Size must be comprised between 0 and ∞.

f) **`Set<ReferenceLocation> scenarioPublishedLocations`** *(Optional)*

The reference location(s) of the scenario: i.e. the location(s) to avoid. For RR/FL type of scenario, it is typically the reference location of the traffic volume of the scenario (but not always: in some cases the scenario intends to avoid/protect some other reference location).
Constraint: Size must be comprised between 0 and ∞.

g) **Set<AirNavigationUnitId> ownerFMPs** *(Optional)*
The FMP(s) that are the owner of the scenario (e.g. it could happen that the location of the scenario overlaps with multiple ACC/FMP).
Constraint: Size must be comprised between 0 and ∞.

h) **Set<AirspaceId> onLoad** *(Optional)*
The list of published airspaces that will be on-loaded when the scenario is applied.
Constraint: Size must be comprised between 0 and ∞.

i) **Set<AirspaceId> offLoad** *(Optional)*
The list of published airspace that will be off-loaded when the scenario is applied.
Constraint: Size must be comprised between 0 and ∞.

j) **WeeklySchedule weeklyApplicability** *(Optional)*
If the scenario can only be applied during certain periods, this weekly schedule represents this.
Tyical usage is: scenario that are already included in certain RAD restrictions that are always applied during the weekend. So the scenario should be applied during the periods when this RAD is dis-abled (in CACD)

k) **string from** *(Optional)*
The textual representation of the flows (upstream) of the traffic volume of the scenario.
Constraint: Pattern: `MULTILINE_TEXT{1,10000}`

l) **string to** *(Optional)*
The textual representation of the flows (downsteam) of the traffic volume of the scenario.
Constraint: Pattern: `MULTILINE_TEXT{1,10000}`

m) **Set<RestrictionId> scenarioIncludedInRestrictions** *(Optional)*
RAD restrictions (if enabled) that incorporates this scenario: e.g. some RR/FL scenario are included in some RAD.
Constraint: Size must be comprised between 0 and ∞.

n) **Set<RestrictionId> disabledRestrictions** *(Optional)*
RAD restrictions that need to be disabled (CACD) before the scenario can be applied.
Constraint: Size must be comprised between 0 and ∞.

o) **Set<RestrictionId> ignoredRestrictions** *(Optional)*
RAD restrictions that need to be ignored (IFPS) before the scenario can be applied.
Constraint: Size must be comprised between 0 and ∞.

p) **Set<string> alternativeIcaoRoutePortions** *(Optional)*
Each suggestedAlternativeRoute represents one possible alternative (partial F15) AR/RR/FL scenario contains examples on how an AO could fly to get around the location to avoid.

Constraint: Size must be comprised between 0 and ∞.

q) **Set<ScenarioLevelConstraint> levelConstraints** *(Optional)*
Each levelConstraints represents one possible level constraint. Typically it states something like: file below FL240 until clear of sector.
Constraint: Size must be comprised between 0 and ∞.

r) **ScenarioPublicationStatus publicationStatus** *(Mandatory)*
The textual representation of the flows (upstream) of the traffic volume of the scenario.

s) **DateTimeMinutePeriodWithUFN applicabilityPeriod** *(Mandatory)*
Describes when the scenario is considered applicable: when `applicabilityPeriod` is in the past, the scenario is considered an archived scenario no longer applicable.

t) **string comments** *(Optional)*
Constraint: Pattern: `MULTILINE_TEXT{0,10000}`

u) **DateTimeSecond lastModifiedTimeStamp** *(Mandatory)*
The timestamp when the scenario was last modified/saved.

v) **string scenarioValidationErrorsOrWarnings** *(Optional)*
If there are validation errors or warnings with a scenario, the scenarioValidationErrorsOr-Warnings will be present.
Constraint: Pattern: `MULTILINE_TEXT{1,10000}`

w) **Set<MeasureIdAndTV> scenarioMeasures** *(Optional)*
Gives a summary of all the measures inside the scenario and their traffic volume id on which they are based. In addition it describes per rerouting measure, the off-loaded traffic volumes.
Constraint: Size must be comprised between 0 and ∞.

(3) Used by: ScenarioListReply.

4.142. typedef<string> ScenarioEvent

(1) Pattern: `TEXT{1,10000}`

(2) Used by: ScenarioAttributes.

4.143. typedef<string> ScenarioId

(1) Pattern: `TEXT{1,255}`

(2) Used by: ScenarioMeasureRetrievalRequest, MeasureFromScenarioRepository, ScenarioL-istRequest, TrafficVolumeScenarios, ScenarioAttributes.

4.144. ScenarioImpact

(1) Describes the scenario impact: if a scenario measure is applied to solve a problem on a traffic volume, then this scenario impacts the traffic in that problem traffic volume. This class describes the impact (See also TrafficVolumeScenarios).

(2) Attributes:

a) **CountsValue totalCommonFlights** *(Mandatory)*
The total number of common flight between the problem traffic volume and the solution (scenario) traffic volume.
This corresponds to the total number of flights inside the flow counts for this scenario traffic volume flow.

b) **CountsValue totalOtherFlights** *(Mandatory)*
The total number of flights captured by the solution (scenario) traffic volume that are not captured by the queried ("problem") TrafficVolume. i.e. when the user would apply a relevant scenario measure, then this measure would correctly offload numberCommonFlights from the 'problem' Traffic volume, but additionally it would "penalise" numberOtherFlights that are not even crossing the "problem" Traffic volume.

c) **DateTimeMinutePeriod scenarioTrafficVolumeEntryPeriod** *(Optional)*
The (smallest entry count based) period that would be needed for the measure on scenario-TrafficVolume to capture all the common flights. In addition when the measure is applied (with a scenarioTrafficVolumeEntryPeriod), then it will also capture additionally totalOther-Flights.

(3) Used by: Flow.

4.145. ScenarioLevelConstraint

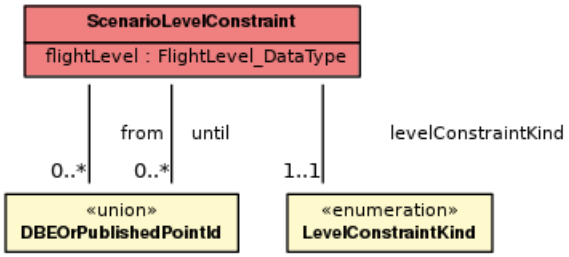

Figure 4.21. ScenarioLevelConstraint Class Diagram

(1)

(2) Attributes:

a) **Set<DBEOrPublishedPointId> from** *(Mandatory)*
Constraint: Size must be comprised between 0 and ∞.

b) **LevelConstraintKind levelConstraintKind** *(Mandatory)*

c) **FlightLevel_DataType flightLevel** *(Mandatory)*

d) **Set<DBEOrPublishedPointId> until** *(Mandatory)*
Constraint: Size must be comprised between 0 and ∞.

(3) Used by: ScenarioAttributes.

4.146. ScenarioMeasureRetrievalRequest

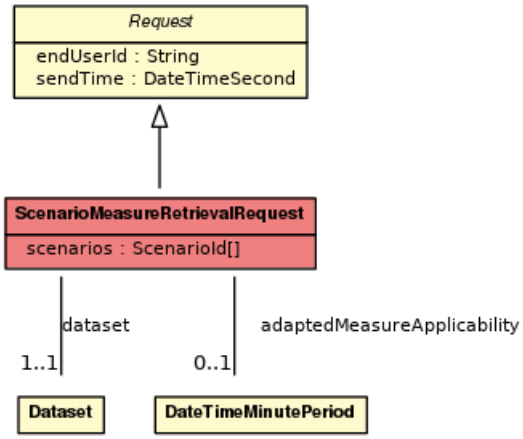

Figure 4.22. ScenarioMeasureRetrievalRequest Class Diagram

(1) Abstract request to query an NM measure list for a scenario, as well as to retrieve the measure details.

(2) Inherits from: Request.

(3) Attributes:

a) **Dataset dataset** *(Mandatory)*
Dataset for which the retrieve scenario measure is requested. See Forecast and Operational Datasets.

b) **Set<ScenarioId> scenarios** *(Mandatory)*
Selects the measures contained in the given list of scenarioId.
Constraint: Size must be comprised between 0 and ∞.

c) **DateTimeMinutePeriod adaptedMeasureApplicability** *(Optional)*
When specified, the measures applicabliy period is adapted to match AdaptedMeasure-Applicablity.
When not specified, the measures returned have their applicability un-changed wrt what is in the scenario repository (this could for example be many years in the past).

(4) Extended by: ScenarioMCDMOnlyRetrievalRequest, ScenarioReroutingRetrievalRequest, ScenarioRegulationRetrievalRequest.

4.147. typedef<string> ScenarioName

(1) Pattern: TEXT{1,255}

(2) Used by: ScenarioListRequest.

4.148. <<enumeration>> ScenarioPublicationStatus

Figure 4.23. <<enumeration>> ScenarioPublicationStatus Class Diagram

(1) Scenario Publication status

(2) Values:

a) **DRAFT**

b) **PUBLISHED**

(3) Used by: ScenarioAttributes.

4.149. <<enumeration>> ScenarioTrafficVolumeMatchingKind

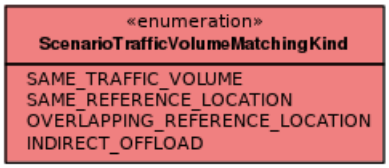

Figure 4.24. <<enumeration>> ScenarioTrafficVolumeMatchingKind Class Diagram

(1) Typically when the end-user has an overload on a "problem" traffic volume (i.e. the queried traffic volume to off-load), then he wants to see which scenarios (solutions) can be applied to off-load that "problem" traffic volume.

(2) Values:

a) **INDIRECT_OFFLOAD**
Solution Traffic Volume is indirectly off-loading the "problem" traffic volume.
For example when applying a level cap on a sector, often some of the upstream sectors are also impacted (and indirectly off-loaded).

b) **OVERLAPPING_REFERENCE_LOCATION**
Solution Traffic Volume is overlapping with the "problem" traffic volume

c) **SAME_REFERENCE_LOCATION**
Solution Traffic Volume is based on the same reference location as the problem traffic volume.

d) **SAME_TRAFFIC_VOLUME**
Solution Traffic Volume is equal to the "problem" traffic volume.

(3) Used by: TrafficVolumeScenarios.

4.150. <<enumeration>> ScenarioType

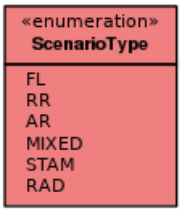

Figure 4.25. <<enumeration>> ScenarioType Class Diagram

(1)

(2) Values:

a) **AR**
The scenario type AR stands for an alternate route scenario: i.e. a scenario with 2 measures in it: a (low rate) regulation for a "non-standard" route and normally a horizontal rerouting to reroute flights into the new "route".

b) **FL**
The scenario type FL stands for Flight Level Capping scenario: i.e. a scenario with 2 measures in it: a (zero rate) regulation and normally a vertical rerouting.

c) **MIXED**
The scenario type Mixed represents all other scenario: for example contingency scenario containing a set of regulations. When such a scenario is aplied, one or more measures of the scenario are applied (but not necessarily all of them).

d) **RAD**
The scenario type Restriction stands for a scenario describing what a flight should do if it gets caught by a restriction (typically EU RAD restrictions).The scenario contains 1 rerouting measure.

e) **RR**
The scenario type RR stands for Flight Level Capping scenario: i.e. a scenario with 2 measures in it: a (zero rate) regulation and normally a horizontal rerouting.

f) **STAM**
The scenario type STAM stands for STAM scenario : i.e. typically a scenario with 1 measure in it: either a cherry picked regulation or a cherry picked level-cap or horizontal rerouting or an MCDM-only measure.
Note that in some cases we could have multiple cherry picked measures inside 1 STAM level cap scenario to describe the different options of where to decent and where to climb up again.

(3) Used by: ScenarioAttributes.

4.151. typedef<string> SectorConfigurationId

(1) Unique id of a sector configuration.

(2) Pattern: `(UALPHA|DIGIT|.){1,6}`

(3) Used by: SectorConfigurationPlan, PlannedSectorConfigurationActivation.

4.152. SectorConfigurationPlan

(1) Sector configuration plan for a given AUA or sector cluster on a given day.

(2) In a retrieval context, the plan is said to be 'complete' in the sense that it contains all the plan entries from all involved data sources.

(3) In an update context, the plan can be complete (if the B2B client designer decided to overwrite the default CACD values) or limited to the (full list) of (pre-)tactical updates with the gaps marked as AIRSPACE datasource (to obtain a complete time partition).

(4) In any case, periods in the time partition marked with datasource AIRSPACE correspond to removing any potential (pre-)tactical update and hence reset the corresponding values to the CACD definition for that period.

(5) Inherits from: `TacticalConfigurationPlan`.

(6) Attributes:

a) **`AirspaceId` airspace** *(Mandatory)*
AUA or sector cluster to which these sector configurations belong.

b) **Map<SectorConfigurationId,Set<AirspaceId>> knownConfigurations** *(Contextual)*
The set of sector configuration ids that NM knows for the AUA or sector cluster, and for each sector configuration, the set of sectors that compose it.
Presence:

i) Must be null in `CapacityPlanUpdateRequest`, `OTMVPlanUpdateRequest`, `RunwayConfigurationPlanUpdateRequest`, `SectorConfigurationPlanUpdateRequest`, `TrafficVolumeActivationPlanUpdateRequest`

ii) Mandatory otherwise.

Constraints:

i) Size must be comprised between 0 and ∞.

ii) Item size must be comprised between 0 and ∞.

c) **Set<PlannedSectorConfigurationActivation> nmSchedule** *(Contextual)*

The plan resulting from the superimposition of all constraints from all data sources, which the NM system is using as plan for its own calculations.

The NM schedule exposes the complete time partition of the configurations for the day, i.e. the data coming from the various data sources contributing to the NM view of the plan: these data sources are defined in PlanDataSource.

The NM schedule cannot be updated directly by the caller; it is the outcome of updates via the various data sources.

The possible values of dataSource are limited to AIRSPACE and TACTICAL - the AIRSPACE value meaning that the value associated to the applicabilityPeriod is the CACD one, and the TACTICAL value meaning that this plan entry corresponds to the explicit tactical update expressed via sectorConfigurationId.

Presence:

i) Must be null in CapacityPlanUpdateRequest, OTMVPlanUpdateRequest, RunwayConfigurationPlanUpdateRequest, SectorConfigurationPlanUpdateRequest, TrafficVolumeActivationPlanUpdateRequest

ii) Mandatory otherwise.

Constraint: Size must be comprised between 0 and ∞.

d) **Set<PlannedSectorConfigurationActivation> clientSchedule** *(Mandatory)*

(Pre-)tactical sector configuration activations of the AUA or sector cluster associated to their applicability period, as maintained by the client. This plan contains only the updated configurations together with an indication that the default CACD values apply when not updated (cf. PlanDataSource). The actual CACD values for these CACD periods can be found in the nmSchedule

In an update context, the clientSchedule can be complete (if the B2B client designer decided to overwrite the default CACD values) or limited to the actual differences with regards to the CACD defaults. This is a B2B client designer's decision and depends on how CACD wants to be used in combination with the B2B. If the clientSchedule only contains the actual differences with regards to the CACD defaults, then the clientschedule still needs to contain the full list of (pre-) tactical updates and for the non (pre-) tactically updated periods, an explicit indication that the CACD values need to be used (but without repeating the CACD values themselves). So in any case, the clientschedule needs to be a complete time partition for the full day.

In any case, it is of drastic importance to understand that an entry in the schedule (i.e. a period) overwrites all CACD values in that period.

The possible values of dataSource are limited to AIRSPACE and TACTICAL - the AIRSPACE value meaning that the value associated to the applicabilityPeriod is the CACD one, and the TACTICAL value meaning that this plan entry corresponds to the explicit tactical update expressed via sectorConfigurationId.

Constraints:

i) Size must be comprised between 0 and ∞.

ii) See INCOMPLETE_SCHEDULE

(7) Constraint:

a)

Name	INCOMPLETE_SCHEDULE
Attribute	clientSchedule
Description	clientSchedule has gaps and/or overlaps in the time partition or is not covering exactly one day.

(8) Used by: SectorConfigurationPlanRetrievalReply, SectorConfigurationPlanUpdateReply, SectorConfigurationPlanUpdateRequest.

4.153. SegmentLocation

(1) Segment location: i.e. a part of a route or of a direct route.

(2) Inherits from: Location.

4.154. SetOfAerodromesLocation

(1) Set of aerodromes location.

(2) Inherits from: Location.

4.155. SignificantDeltaCount

(1) Compares a before and after situation and indicates if the change is significant. An insignificant change is a change that is considered not relevant.

(2) Attributes:

a) **CountsValue beforeCount** *(Mandatory)*

b) **CountsValue afterCount** *(Mandatory)*

c) **boolean isSignificantChange** *(Mandatory)*

(3) Used by: DeltaATFCMSituationCounts.

4.156. SignificantDeltaDuration

(1) Compares a before and after situation and indicates if the change is significant. An insignificant change is a change that is considered not relevant.

(2) Attributes:

a) **LongDurationHourMinute beforeCount** *(Mandatory)*

b) **LongDurationHourMinute afterCount** *(Mandatory)*

c) **boolean isSignificantChange** *(Mandatory)*

(3) Used by: DeltaATFCMSituationDelays.

4.157. <<abstract>> Simulation

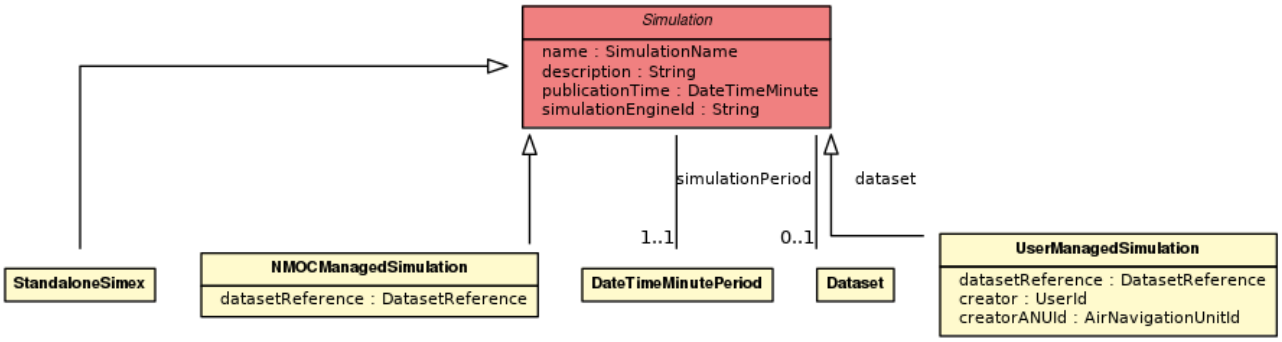

Figure 4.26. <<abstract>> Simulation Class Diagram

(1) Common ancestor of the different types of simulation.

(2) Attributes:

a) **SimulationName name** *(Mandatory)*
The given name of the simulation.

b) **string description** *(Mandatory)*
The description of the simulation.

c) **DateTimeMinute publicationTime** *(Contextual)*
The time when the simulation became published. For a USER_MANAGED_SIMULATION, it is the time when the simulation was started.
Presence:

i) Mandatory in SimulationListReply, SimulationStartReply

ii) Must be null in SimulationStartRequest

iii) Optional otherwise.

d) **string simulationEngineId** *(Contextual)*
The Id of the simulation engine running the simulation (also known as simulation_ID in B2C). It is mainly needed by NMOC to find back the simulation in case of questions.

Presence:

i) Mandatory in SimulationListReply, SimulationStartReply

ii) Must be null in SimulationStartRequest

iii) Optional otherwise.

e) **DateTimeMinutePeriod simulationPeriod** *(Mandatory)*
The period that is being simulated (e.g. flights have been loaded that are departing/arriving/flying within that period).
When starting a USER_MANAGED_SIMULATION, the simulationPeriod needs to be exactly 1 day and it needs to be contained inside the simulation period of the reference. For example, a simulation for an operational datasetReference can either have as period [D, D-1 00:00[or [D-1 00:00, D-2 00:00[(See also OPERATIONAL and FORECAST datasets, and SIMULATION datasets.)

f) **Dataset dataset** *(Contextual)*
The dataset of the simulation. This dataset needs to be used in any query on that simulation (for example in the flightlist and regulationlist)
Presence:

i) Mandatory in SimulationListReply, SimulationStartReply

ii) Must be null in SimulationStartRequest

iii) Optional otherwise.
Constraint: See DATASET_TYPE_MUST_BE_SIMULATION

(3) Constraint:

a)

Name	DATASET_TYPE_MUST_BE_SIMULATION
Attribute	dataset
Description	Dataset must be SIMULATION type.

(4) Extended by: UserManagedSimulation, NMOCManagedSimulation, StandaloneSimex.

(5) Used by: SimulationListReply.

4.158. SimulationAvailability

(1) Shows the available (free) user managed simulations and the used user managed simulation for a reference dataset

(2) Attributes:

a) **DateTimeMinutePeriod simulatablePeriod** *(Mandatory)*
The period within a simulation period needs to be.

For STANDALONE_SIMEX simulations, it is typically a 7 day period relatively far in the future.

b) **`int availableSimulations`** *(Mandatory)*
Constraint: Range:]- ∞,∞[.

c) **`int simulationsInUse`** *(Mandatory)*
Constraint: Range:]- ∞,∞[.

(3) Used by: SimulationAvailabilityReply.

4.159. `typedef<string> SimulationName`

(1) Pattern: `TEXT{1,40}`

(2) Used by: Simulation.

4.160. `<<enumeration>> SimulationType`

(1) Describes the different types of simulations.

(2) Values:

a) **`NMOC_MANAGED_SIMULATION`**
The simulation is managed (start/stop) and prepared by NMOC for the other users (B2B & B2C) to have a look at the results. Optionally the users can create or modify the measures and tactical updates to evaluate the what-if effects.

b) **`STANDALONE_SIMEX`**
The simulation is a standalone SIMulation Experiment (SIMEX). Special future events are typically prepared and simulated on SIMEX with specially modified environment (CACD) data and forecasted traffic. It can be used as a reference for other simulations to evaluate different alternatives and compare with the `STANDALONE_SIMEX` as reference.
The simulation is managed (start/stop) and prepared by NMOC for the other users (B2B & B2C) to have a look at the results. Optionally the users can create or modify the measures and tactical updates.

c) **`USER_MANAGED_SIMULATION`**
The simulation is managed (start/stop) and prepared by the users (via B2B & B2C) to simulate some measures and evaluate their effect. In addition the user can request other users or NMOC to have a look at proposed solution or to contribute to the simulation.

4.161. `StandaloneSimex`

(1) The simulation is a standalone SIMulation Experiment (SIMEX).

(2) It can be used as a reference for other simulations.

(3) The dataset corresponding to a standalone SIMEX simulation has as type STANDALONE_SIMEX.

(4) Inherits from: `Simulation`.

4.162. <<enumeration>> SubTotalsTrafficCountsType

(1) Enumerates the possible sub totals counts types.

(2) Values:

a) **`ATC_ACTIVATED`**
ATC activated flights. Note that this also includes terminated flights that were ATC activated.

b) **`IFPL`**
Flights created from a flight plan filed to IFPS that are not suspended, nor `ATC_ACTIVATED`, nor `TACT_ACTIVATED_WITH_FSA`, nor `TACT_ACTIVATED_WITHOUT_FSA`.

c) **`PFD`**
Predicted flights (See Forecast and Operational Datasets) that are not suspended

d) **`RPL`**
Flights created from a repetitive flight plan that are not suspended and for which no flight plan has been filed yet (by the Airspace user or by IFPS).

e) **`SUSPENDED`**
Suspended Flights. Note that suspended flights are not considered part of the Traffic-Type.LOAD.

f) **`TACT_ACTIVATED_WITHOUT_FSA`**
TACT activated with no FSA message expected. Note that this also includes terminated flights that were `TACT_ACTIVATED_WITHOUT_FSA`.

g) **`TACT_ACTIVATED_WITH_FSA`**
TACT activated with FSA message expected (but not yet received). Note that this also includes terminated flights that were `TACT_ACTIVATED_WITH_FSA`.

(3) Used by: Counts.

4.163. <<abstract>> TacticalConfigurationPlan

(1) Common information for all configuration plans on a given day.

(2) Attributes:

a) **`PlanDataId` `dataId`** *(Mandatory)*
Opaque identifier representing the version of this plan.
Constraint: See `INCONSISTENT_DATAID_AND_DATASET_TYPE`

b) **`Dataset` `dataset`** *(Mandatory)*
Dataset to which the plan belongs.
Constraints:

i) See INCONSISTENT_DATAID_AND_DATASET_TYPE

ii) See INCONSISTENT_DAY_AND_DATASET_TYPE

c) **DateYearMonthDay day** *(Mandatory)*
Day for which this plan is valid.
Constraint: See INCONSISTENT_DAY_AND_DATASET_TYPE

d) **boolean planTransferred** *(Optional)*
Indicates if the plan has been transferred to the OPERATIONAL dataset. When false, it means that the most up-to-date data can be found in the FORECAST dataset.

e) **boolean planCutOffReached** *(Optional)*
Indicates if the plan can still be updated in the FORECAST dataset, i.e. if the forecast cut-off time has been reached or not.

(3) Constraints:

a)

Name	INCONSISTENT_DATAID_AND_DATASET_TYPE
Attributes	dataId, dataset
Description	The dataId must match the one returned by the corresponding retrieval request for the given dataset.

b)

Name	INCONSISTENT_DAY_AND_DATASET_TYPE
Attributes	day, dataset
Description	The day must be in [today .. today+5d] depending on dataset type i.e.: i) day must be in [today .. today+5d] in an FORECAST context. ii) day must be in [today .. tomorrow] in an OPERATIONAL context. iii) day has no range constraint in a SIMULATION context.

(4) Extended by: TrafficVolumeActivationPlans, OTMVPlan, SectorConfigurationPlan, TrafficVolumeActivationPlan, CapacityPlans, RunwayConfigurationPlan, OTMVPlans.

4.164. <<abstract>> TacticalConfigurationRetrievalRequest

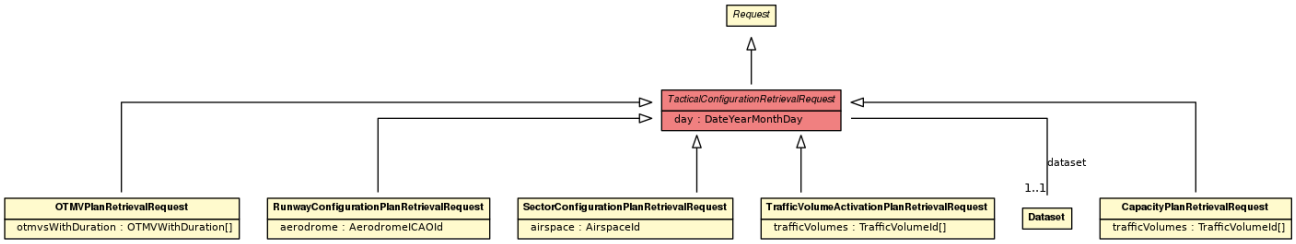

Figure 4.27. <<abstract>> TacticalConfigurationRetrievalRequest Class Diagram

(1) Abstract request to retrieve a tactical configuration (runway, sector, capacity, TV, OTMV) on a given day.

(2) Inherits from: Request.

(3) Attributes:

a) **Dataset dataset** *(Mandatory)*
Dataset for which the tactical configuration is requested.
Constraint: See INCONSISTENT_DAY_AND_DATASET_TYPE

b) **DateYearMonthDay day** *(Mandatory)*
Day for which the tactical configuration is requested.
Constraint: See INCONSISTENT_DAY_AND_DATASET_TYPE

(4) Constraint:

a)

Name	INCONSISTENT_DAY_AND_DATASET_TYPE
Attributes	day, dataset
Description	The day must be in [D-5, D] depending on dataset type i.e.: i) day must be in [D-5, D] in an FORECAST context. ii) day must be in [D-1, D] in an OPERATIONAL context. iii) day has no range constraint in a SIMULATION context.

(5) Extended by: CapacityPlanRetrievalRequest, RunwayConfigurationPlanRetrievalRequest, TrafficVolumeActivationPlanRetrievalRequest, SectorConfigurationPlanRetrievalRequest, OTMVPlanRetrievalRequest.

4.165. <<abstract>> TrafficCountsReplyData

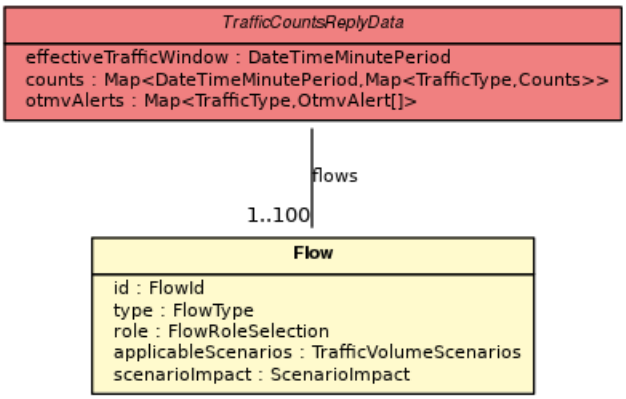

Figure 4.28. <<abstract>> TrafficCountsReplyData Class Diagram

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

(2) Attributes:

a) **DateTimeMinutePeriod effectiveTrafficWindow** *(Mandatory)*
The effective period of time for which counts were requested.
Flights from within this period have been used in the counts. See also TrafficCounts-Request.trafficWindow.

b) **Set<Flow> flows** *(Optional)*
The flows for which the traffic counts are requested.
This attribute is set only when requested (i.e., TrafficCountsByTrafficVolumeRe-quest.computeFlowCounts has been set to `true`)
Constraint: Size must be comprised between 1 and 100.

c) **Map<DateTimeMinutePeriod,Map<TrafficType,Counts>> counts** *(Optional)*
The traffic counts by period (interval) and by requested traffic type.
Constraints:

i) Size must be comprised between 0 and 1440.

ii) Item size must be comprised between 1 and 3.

d) **Map<TrafficType,Set<OtmvAlert>> otmvAlerts** *(Optional)*
The OTMV alerts by traffic type for OTMV duration TrafficCountsRequest.countsIn-terval.duration.
All count periods(intervals) that start in OtmvAlert.period are considered to have an OTMV alert with OtmvStatus.
This attribute is set only when requested (i.e., TrafficCountsByTrafficVolumeRe-quest.computeOtmvAlerts has been set to `true`)
Constraints:

i) Size must be comprised between 0 and 3.

ii) Item size must be comprised between 1 and 1440.

(3) Extended by: TrafficCountsByPointReplyData, TrafficCountsByTrafficVolumeReplyData, TrafficCountsByAirspaceReplyData, TrafficCountsByAircraftOperatorReplyData, TrafficCountsByAerodromeReplyData, TrafficCountsByAerodromeSetReplyData.

4.166. <<abstract>> TrafficCountsRequest

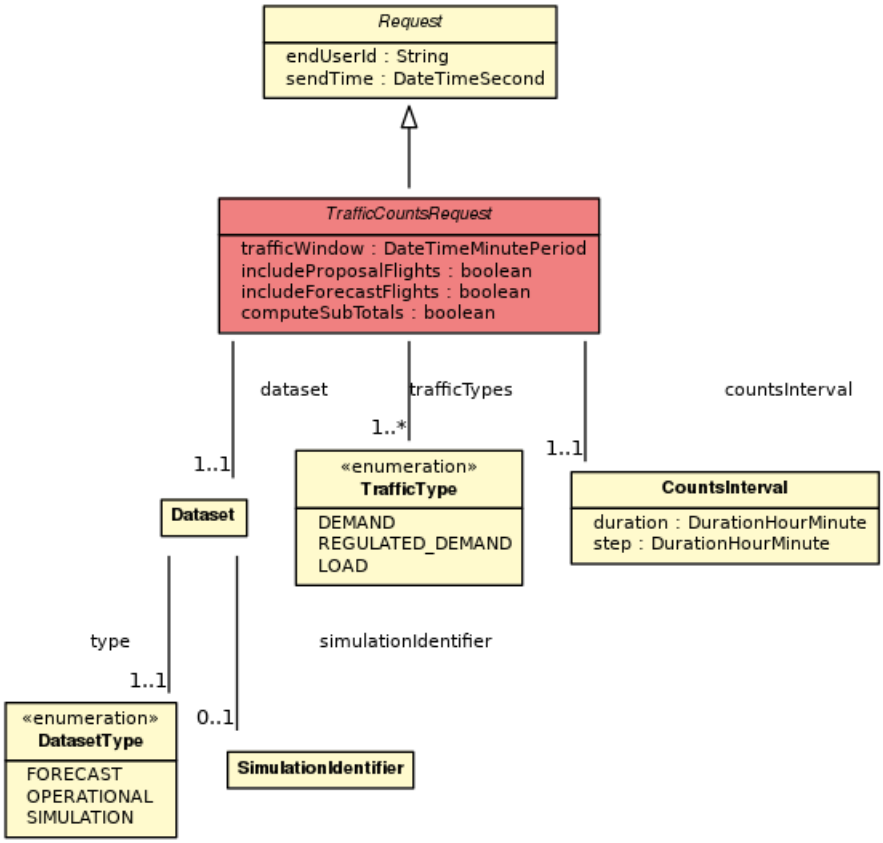

Figure 4.29. <<abstract>> TrafficCountsRequest Class Diagram

(1) Abstract request to query traffic counts.

(2) Inherits from: Request.

(3) Attributes:

a) **Dataset dataset** *(Mandatory)*
Dataset for which the runway configuration plan is requested.
See Forecast and Operational Datasets and Simulation Datasets.
Constraint: See INVALID_QUERY_PERIOD_RANGE

b) **DateTimeMinutePeriod trafficWindow** *(Mandatory)*
The period of time for which traffic counts are requested.
Typically in a traffic counts reply, more than one traffic count period/interval is returned. For each such period, count values are returned. Each such count period has a duration of `countsInterval.duration`. The start of the count periods is are separated by `countsInterval.step`.
The last count period (for which counts are returned) is the count period that starts before `trafficWindow.Unt` (but the end of that last count period can extend beyond the `trafficWindow.Unt`, as the duration of that count period is `countsInterval.duration`).
Note that the first count period can start before `trafficWindow.Wef` as the count period start is rounded down to a multiple of `countsInterval.step`.
The `effectiveTrafficWindow` indicates the real traffic window for which flights have been taken into account to compute the different count periods and count values.
See CountsInterval.
Constraints:

i) See INVALID_QUERY_PERIOD_RANGE

ii) See PERIOD_EXTENSION_CANNOT_BE_GREATER_THAN_24_HOURS

c) **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.

d) **boolean includeForecastFlights** *(Mandatory)*
Determines if the selected traffic must include the "forecast" flights.
In the OPERATIONAL dataset (or in a simulation on OPERATIONAL), a "forecast" flight is a predicted flight (cfmuFlightType = PREDICTED_FLIGHT).
In the FORECAST dataset (or in a simulation on FORECAST), a "forecast" flight is a predicted flight with no intention data.

e) **Set<TrafficType> trafficTypes** *(Mandatory)*
Requested traffic types. For each traffic type, the count values will be returned.
Constraints:

i) Size must be comprised between 1 and 3.

ii) See ONLY_ONE_TRAFFIC_TYPE_IF_FLOW_COUNTS

f) **boolean computeSubTotals** *(Mandatory)*
If `false` then for each traffic type selected a single total is returned. If `true` then for each traffic type selected detailed sub-totals are returned.
The sub-totals traffic counts types are defined in SubTotalsTrafficCountsType.

g) **CountsInterval countsInterval** *(Mandatory)*
Determines how many count periods need to be returned in the `trafficWindow` and determines the duration of each period.

(4) 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	ONLY_ONE_TRAFFIC_TYPE_IF_FLOW_COUNTS
Attribute	trafficTypes
Description	Only one traffic type can be requested in trafficTypes when TrafficCounts-ByTrafficVolumeRequest.computeFlowCounts is set to true.

c)

Name	INVALID_QUERY_PERIOD_RANGE
Attributes	trafficWindow, dataset
Description	The dataset.type from which the traffic counts 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 [today .. today+5d] ii) if the DatasetType is equals to OPERATIONAL the trafficWindow shall be defined within the range [yesterday 21:00 UTC .. tomorrow]

(5) Extended by: TrafficCountsByPointRequest, TrafficCountsByAerodromeRequest, TrafficCounts-ByTrafficVolumeRequest, TrafficCountsByAircraftOperatorRequest, TrafficCountsByAirspaceRe-quest, TrafficCountsByAerodromeSetRequest.

4.167. TrafficVolumeActivationPlan

(1) Activation plan for a given traffic volume on a given day.

(2) A TrafficVolumeActivation plan is a special plan in the sense that, there can be cases where there is no data known. So there exist traffic volumes for which there is no data at all or only for some periods. In addition, non-activation is not defined in CACD. Therefore the absence of CACD data (NO_DATA datasource) means either no data known or in-active. Either way, the absence of data is considered by NM systems as an in-active traffic volume. In addition, sector configuration activ-ations (tactically updated or CACD defined) also over-rule CACD activations of a traffic volume. They can activate or de-activate a traffic volume (also marked by AIRSPACE datasource). The above activations can be over-ruled (set active or in-active) by the tactical updates (B2B or HMI). In addition regulation measures can activate a traffic volume (over-ruling either CACD or B2B/HMI updates). The consolidated info can be found back in the nmSchedule attribute.

(3) So the (de-)activation of a traffic volume is determined hierarchically by NO_DATA (least priority), CACD (including derived from sector config), tactical updates (a.o. via B2B) , regulation measures

(highest priority). In the client schedule the non regulated traffic volumes activations are maintained (just in case the regulation can be cancelled before the end of its applicability period).

(4) In a retrieval context, the plan is said to be 'complete' in the sense that it contains all the plan entries from all involved data sources (including NO_DATA data source in case no info is known).

(5) In an update context, the plan can be complete (if the B2B client designer decided to overwrite the default CACD values/sector configuration derived activation) or limited to the (full list) of (pre-)tactical updates with the gaps marked as AIRSPACE (meaning in update context: NO_DATA or CACD or derived from sector config) datasource (to obtain a complete time partition).

(6) In any case, periods marked with datasource AIRSPACE in the time partition correspond to removing any potential (pre-)tactical update and hence reset the corresponding values to the NO_DATA/CACD/"derived from sector config" definition for that period.

(7) Inherits from: `TacticalConfigurationPlan`.

(8) Attributes:

a) **`TrafficVolumeId trafficVolume`** *(Mandatory)*
Traffic volume to which this activation plan applies.

b) **`Set<PlannedTrafficVolumeActivation> nmSchedule`** *(Contextual)*
The plan resulting from the superimposition of all constraints from all data sources, which the NM system is using as plan for its own calculations.
The NM schedule exposes the complete time partition of the configurations for the day, i.e. the data coming from the various data sources contributing to the NM view of the plan: these data sources are defined in PlanDataSource.
The NM schedule cannot be updated directly by the caller; it is the outcome of updates via the various data sources.
In `nmSchedule` the possible values of dataSource are limited to `NO_DATA`, `AIRSPACE`, `TACTICAL` and `MEASURE`.
Note that NO_DATA in nmClientSchedule means either inactive or that no data has been specified.
Presence:

i) Must be null in `CapacityPlanUpdateRequest`, `OTMVPlanUpdateRequest`, `RunwayConfigurationPlanUpdateRequest`, `SectorConfigurationPlanUpdateRequest`, `TrafficVolumeActivationPlanUpdateRequest`

ii) Mandatory otherwise.

Constraint: Size must be comprised between 0 and ∞.

c) **`Set<PlannedTrafficVolumeActivation> clientSchedule`** *(Mandatory)*
(Pre-)tactical traffic volume activations associated to their applicability period, as maintained by the client. This plan contains only the updated configurations together with an indication (AIRSPACE datasource) that the default CACD/sector config derived values apply when not updated (cf. PlanDataSource). The actual activity for these airspace datasource periods can be found in the `nmSchedule`

In an update context, the clientSchedule can be complete (if the B2B client designer decided to overwrite the default CACD values) or limited to the actual differences with regards to the AIRSPACE defaults. This is a B2B client designer's decision and depends on how CACD wants to be used in combination with the B2B. If the clientSchedule only contains the actual differences with regards to the AIRSPACE defaults, then the clientschedule still needs to contain the full list of (pre-) tactical updates and for the non (pre-) tactically updated periods, an explicit indication that the AIRSPACE values need to be used (but without repeating the NO_DATA/CACD/sector config derived values themselves). So in any case, the clientschedule needs to be a complete time partition for the full day.
In any case, it is of drastic importance to understand that an entry in the schedule (i.e. a period) overwrites all CACD/sector config derived values in that period.
The possible values of dataSource are limited to `AIRSPACE` and `TACTICAL` - the `AIRSPACE` value meaning that the value associated to the applicabilityPeriod is the CACD one (i.e. CACD or the TV activation derived from a sector configuration or that there is no data defined in CACD), and the `TACTICAL` value meaning that this plan entry corresponds to the explicit tactical update expressed via active.
Constraints:

i) Size must be comprised between 0 and ∞.

ii) See `INCOMPLETE_SCHEDULE`

(9) Constraint:

a)

Name	`INCOMPLETE_SCHEDULE`
Attribute	`clientSchedule`
Description	`clientSchedule` has gaps and/or overlaps in the time partition or is not covering exactly one day.

4.168. `TrafficVolumeActivationPlans`

(1) Activation plans for one or more traffic volumes on a given day.

(2) A TrafficVolumeActivation plans contains a map of a special plan in the sense that, there can be cases where there is no data known. So there exist traffic volumes for which there is no data at all or only for some periods. In addition, non-activation is not defined in CACD. Therefore the absence of CACD data (NO_DATA datasource) means either no data known or in-active. Either way, the absence of data is considered by NM systems as an in-active traffic volume. In addition, sector configuration activations (tactically updated or CACD defined) also over-rule CACD activations of a traffic volume. They can activate or de-activate a traffic volume (also marked by AIRSPACE datasource). The above activations can be over-ruled (set active or in-active) by the tactical updates (B2B or HMI). In addition regulation measures can activate a traffic volume (over-ruling either CACD or B2B/HMI updates). The consolidated info can be found back in the `nmSchedule` attribute.

(3) So the (de-)activation of a traffic volume is determined hierarchically by NO_DATA (least priority), CACD (including derived from sector config), tactical updates (a.o. via B2B) , regulation measures

(highest priority). In the client schedule the non regulated traffic volumes activations are maintained (just in case the regulation can be cancelled before the end of its applicability period).

(4) In a retrieval context, the plan is said to be 'complete' in the sense that it contains all the plan entries from all involved data sources (including NO_DATA data source in case no info is known).

(5) In an update context, the plan can be complete (if the B2B client designer decided to overwrite the default CACD values/sector configuration derived activation) or limited to the (full list) of (pre-)tactical updates with the gaps marked as AIRSPACE (meaning in update context: NO_DATA or CACD or derived from sector config) datasource (to obtain a complete time partition).

(6) In any case, periods marked with datasource AIRSPACE in the time partition correspond to removing any potential (pre-)tactical update and hence reset the corresponding values to the NO_DATA/CACD/"derived from sector config" definition for that period.

(7) Inherits from: `TacticalConfigurationPlan`.

(8) Attributes:

a) **`Map<TrafficVolumeId,PlannedTrafficVolumeActivations> tvActivations`**
(Mandatory)
A map of Traffic volumes as keys mapped to its capacities.
Constraints:

i) Size must be comprised between 0 and ∞.

ii) See `INCOMPLETE_SCHEDULE`

(9) Constraint:

a)

Name	`INCOMPLETE_SCHEDULE`
Attribute	`tvActivations`
Description	`tvCapacities.clientSchedule` has gaps and/or overlaps in the time partition or is not covering exactly one day.

(10) Used by: TrafficVolumeActivationPlanRetrievalReply, TrafficVolumeActivationPlanUpdateReply, TrafficVolumeActivationPlanUpdateRequest.

4.169. `typedef<string> TrafficVolumeId`

(1) NM unique identifier for a traffic volume.

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

(3) Used by: OTMVPlan, OTMV, NetworkImpactAssessmentRetrievalReply, TrafficVolumeActivationPlanRetrievalRequest, NetworkImpactAssessmentRetrievalRequest, TrafficVolumeActivationPlans, MeasureIdAndTV, CapacityPlans, ScenarioListRequest, CapacityPlanRetrievalRequest, OTMVWithDuration, TrafficVolumeActivationPlan, TrafficVolumeScenarios, ScenarioAttributes.

4.170. TrafficVolumeLocation

(1) Details the location at which a measure applies.

(2) Inherits from: Location.

(3) Attributes:

a) **ReferenceLocation referenceLocation** *(Contextual)*
The reference location of the traffic volume to which the measure applies.
Presence:

i) Must be null in MCDMOnlyCreationRequest, MCDMOnlyUpdateRequest, RegulationCreationRequest, RegulationProposalFilingRequest, RegulationProposalUpdateRequest, RegulationUpdateRequest, ReroutingCreationRequest, ReroutingUpdateRequest

ii) Optional otherwise.

b) **TrafficVolumeId id** *(Mandatory)*
The traffic volume to which the measure applies.

c) **FlightLevelRange flightLevels** *(Contextual)*
Flight level range of the traffic volume to which this measure applies.
Presence:

i) Must be null in MCDMOnlyCreationRequest, MCDMOnlyUpdateRequest, RegulationCreationRequest, RegulationProposalFilingRequest, RegulationProposalUpdateRequest, RegulationUpdateRequest, ReroutingCreationRequest, ReroutingUpdateRequest

ii) Optional otherwise.

d) **string description** *(Contextual)*
The traffic volume description to which the measure applies.
Presence:

i) Must be null in MCDMOnlyCreationRequest, MCDMOnlyUpdateRequest, RegulationCreationRequest, RegulationProposalFilingRequest, RegulationProposalUpdateRequest, RegulationUpdateRequest, ReroutingCreationRequest, ReroutingUpdateRequest

ii) Optional otherwise.

e) **Set<TrafficVolumeSetId> setIds** *(Contextual)*
The set of traffic volume sets containing the traffic volume to which this measure applies.
Presence:

i) Must be null in MCDMOnlyCreationRequest, MCDMOnlyUpdateRequest, RegulationCreationRequest, RegulationProposalFilingRequest, RegulationProposalUpdateRequest, RegulationUpdateRequest, ReroutingCreationRequest, ReroutingUpdateRequest

ii) Optional otherwise.
Constraint: Size must be comprised between 1 and 200.

(4) Used by: RegulationOrMCDMOnly.

4.171. TrafficVolumeScenarios

(1) For a traffic volume, represents the scenario's (from the NM scenario repository) that contain measures (regulation or rerouting) based on that traffic volume.

(2) Typically when the end-user has an overload on a "problem" traffic volume, then he wants to see which scenarios (solutions) can be applied to off-load that "problem" traffic volume. Those scenarios might be based on different traffic volumes that e.g. overlap with the "problem" traffic volume (e.g. in case the current sector configuration has grouped some sectors together).

(3) This class represents the different scenarios that have measures on one such scenario traffic volume (i.e. solution traffic volume).

(4) Note that a scenario can have multiple measures : for example a contingency scenario contains multiple measures (on different traffic volumes) that can be used to handle the contingency. When such a scenario is applied one or more measures of the scenario are applied (but not necessarily all measures of the scenario).

(5) The retrieveRegulationsFromScenario and retrieveReroutingsFromScenario services (flow) can then be used to retrieve the applicable measures themselves from these scenarios.

(6) The applicable measures are those measures that have as `trafficVolumeId`, the `solutionTrafficVolumeId`. Those measures can then be applied via B2B in a NM simulation (to evaluate the impact) or can be requested for implementation via the `fileRegulationProposal` , `createRegulation` and `createRerouting` service requests. (See general text on scenario repository in flow (below))

(7) The `queryScenarioRepository` can be used to retrieve the details about the scenarios.

(8) The `trafficVolumeMatchingKind` describes how the solution traffic volume relates to the problem traffic volume (e.g. they are both traffic volumes on the same reference location or they are overlapping or ... (See for more info: trafficVolumeMatchingKind)

(9) Attributes:

a) **TrafficVolumeId solutionTrafficVolumeId** *(Mandatory)*
The (solution) traffic volume of the concerned scenario measures.

b) **ScenarioTrafficVolumeMatchingKind trafficVolumeMatchingKind** *(Mandatory)*

Describes how the `solutionTrafficVolumeId` relates to the problem traffic volume (e.g. they are both traffic volumes on the same reference location or they are overlapping or ...(See for more info : ScenarioTrafficVolumeMatchingKind)).

c) **Set<ScenarioId> scenarios** *(Mandatory)*
The list of scenarios that contain measures on `solutionTrafficVolumeId`.
Constraint: Size must be comprised between 1 and ∞.

(10) Used by: Flight, Flow.

4.172. UpdateFlightInMeasure

(1) Add, remove or update a flight in a measure. `flightUpdate` describes the different kind of operations supported on a flight.

(2) Attributes:

a) **FlightKeys flightKeys** *(Mandatory)*
The flight keys.

b) **UpdateFlightInMeasureChoice flightUpdate** *(Mandatory)*
Either:

- force/unforce the CTOT, or
- include/exclude the flight from one or more regulations, or
- add/remove a flight to/from a measure.

(3) Used by: UpdateFlightsInMeasureReply, UpdateFlightsInMeasureRequest.

4.173. UpdateFlightInMeasureChoice

(1) Describes the different kind of operations supported on a flight in the context of an `UpdateFlightInMeasure`.

(2) Choices:

a) **ReroutingId addFlightToRerouting**
Add a flight to a cherry picked rerouting.

b) **ReroutingId removeFlightFromRerouting**
Remove a flight from a rerouting.

c) **RegulationId addFlightToMCDMOnlyMeasure**
Add a flight to the MCDM_only measure with this measure identifier.

d) **RegulationId removeFlightFromMCDMOnlyMeasure**
Remove a flight from the MCDM_only measure with this measure identifier.

e) **ForceFlightInRegulation forceFlightInRegulation**
force a flight in a regulation or add a flight to a cherry picked regulation (by forcing).

f) **RegulationId unforceFlightInRegulation**
Unforce a forced flight in a regulation or remove a flight from a cherry picked regulation (by unforcing it will become exempted again).
If the normal flight is forced, then this will result in a proposal to NMOC to unforce the flight from the regulation (in simulations it directly unforces the flight).
Typically in operational dataset context, one needs to submit a proposal for NMOC to review: in the context of a proposal cherry picked regulation (See `fileRegulationProposal` service) or in the context of modifying the CTOT of a flight in a normal (non-cherry-picked) regulation.

g) **ExcludeReIncludeFlightInRegulation excludeFlightFromRegulation**
Exclude a flight from 1 or more regulations.

h) **ExcludeReIncludeFlightInRegulation reIncludeFlightInRegulation**
Re-include a flight in one or more regulations from which the flight is excluded.

i) **void removeProposalFlight**
Allows to remove/undo a proposal created by `forceFlightInRegulation`, `unforceFlightInRegulation`, `excludeFlightFromRegulation` or `reIncludeFlightInRegulation`.
Note that `removeFlightFromRerouting` needs to be used to remove a proposal created by `addFlightToRerouting`.

(3) Used by: UpdateFlightInMeasure.

4.174. UserManagedSimulation

(1) The simulation is managed (start/stop) and prepared by the users (via B2B & B2C) to simulate some measures and evaluate their effect.

(2) In addition the user can request other users or NMOC to have a look at proposed solution or to contribute to the simulation.

(3) The dataset corresponding to a `UserManagedSimulation` simulation has as type USER_MANAGED_SIMULATION

(4) Inherits from: `Simulation`.

(5) Attributes:

a) **DatasetReference datasetReference** *(Mandatory)*
The reference on which this simulation is based (i.e. from which environment data has been copied and where initially the flights and measures have been copied from).
Constraint: See INVALID_DATASET_REFERENCE_TYPE

b) **UserId creator** *(Optional)*

The B2B or B2C user that created (started) the simulation.

c) **AirNavigationUnitId creatorANUId** *(Optional)*
The ANU identifier of the creator.

(6) Constraints:

a)

Name	INVALID_DATASET_REFERENCE_TYPE
Attribute	datasetReference
Description	DatasetReference.type must be OPERATIONAL or FORECAST.

b)

Name	INVALID_SIMULATION_IDENTIFIER_TYPE
Attribute	dataset
Description	SimulationIdentifier.simulationType must be USER_MANAGED_SIMULATION.

(7) Used by: SimulationStartRequest, SimulationStartReply.

Chapter 5. PRE-OPS Testing

(1) The data available in the PRE-OPS platform for the current MeasuresService is fed daily from the OPS systems.

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