

Open SDV API 仕様

Open SDV Initiative

202609β

目次

1. Open SDV API仕様の位置付け . . . 19
   1.1. オープンSDVとOpen SDV Initiativeの取り組み . . . 19
      1.1.1. SDVとオープンSDV . . . 19
      1.1.2. ビークルAPIと標準化 . . . 19
      1.1.3. Open SDV Initiativeの取り組み . . . 20
      1.1.4. 本仕様の策定状況 . . . 20
   1.2. APIの位置付けと策定方針 . . . 20
      1.2.1. 策定するビークルAPIの位置付け . . . 20
      1.2.2. ビークルAPIの利用者と適用範囲 . . . 21
      1.2.3. ビークルAPIに対する要件 . . . 21
      1.2.4. 要件を満たすためのアプローチ . . . 22
      1.2.5. リスクのあるAPIの扱い . . . 23
   1.3. 想定するシステム階層 . . . 24
      1.3.1. ソフトウェアのモジュール化/階層化の目的 . . . 24
      1.3.2. システム全体の階層構造 . . . 24
      1.3.3. ビークルOSの内部構造 . . . 25
   1.4. 想定する実装構造 . . . 26
2. 主要な概念 . . . 29
   2.1. APIに関する用語とデータ型 . . . 29
      2.1.1. 用語の定義 . . . 29
      2.1.2. データ型 . . . 29
   2.2. 車両の構成記述 . . . 37
      2.2.1. 構成記述に対する要求 . . . 37
      2.2.2. 車両構成の階層構造 . . . 38
      2.2.3. 車両の構成情報の取得方法 . . . 40
      2.2.4. 車両の構成情報の多言語対応 . . . 41
      2.2.5. 構成情報の変化 . . . 41
   2.3. APIの要素と規定内容 . . . 42
      2.3.1. 名称と略称 . . . 42
      2.3.2. 位置付けと機能 . . . 42
      2.3.3. 機能クラス . . . 43
      2.3.4. リスククラス . . . 43
      2.3.5. 構成情報 . . . 43
      2.3.6. 状態 . . . 43
      2.3.7. サービスコール . . . 44
      2.3.8. イベント . . . 44
   2.4. APIの設計方針 . . . 46
      2.4.1. サービスコールの設計方針 . . . 46
      2.4.2. OSに対する想定 . . . 47

2.5. 制御の調停とリスク制御 47
2.5.1. 制御の調停機能 47
2.5.2. リスク制御機能 48
3. 共通規定 52
3.1. 共通データ型 52
3.2. すべてのオブジェクトが持つサービスコール 55
3.3. 共通のイベント 58
3.4. LockableObject 59
3.4.1. LockableObjectの機能 59
3.4.2. LockableObjectの状態 59
3.4.3. LockableObjectのデータ型 60
3.4.4. LockableObjectのイベント 60
3.4.5. LockableObjectのサービスコール 61
3.5. MovableObject 62
3.5.1. MovableObjectの機能 62
3.5.2. MovableObjectの状態 63
3.5.3. MovableObjectのイベント 64
3.5.4. MovableObjectのデータ型 65
3.5.5. MovableObjectのサービスコール 67
4. コアビークルAPI 70
4.1. Vehicle（略称：Vehicle） 70
4.2. Vehicle.VersionOSDVI（略称：VersionOSDVI） 70
4.2.1. 位置付けと機能 70
4.2.2. 構成情報 70
4.2.3. サービスコール一覧 70
4.2.4. データ型の定義 70
4.2.5. サービスコールの詳細規定 70
4.3. Vehicle.VehicleIdentification（略称：VehicleId） 71
4.3.1. 位置付けと機能 71
4.3.2. 構成情報 71
4.3.3. サービスコール一覧 72
4.3.4. データ型の定義 72
4.3.5. サービスコールの詳細規定 73
4.4. Vehicle.Specification（略称：VehicleSpec） 73
4.4.1. 位置付けと機能 73
4.4.2. 構成情報 74
4.4.3. サービスコール一覧 74
4.4.4. データ型の定義 74
4.4.5. サービスコールの詳細規定 75
4.5. Vehicle.Cabin（略称：Cabin） 75
4.6. Vehicle.Cabin.Door（略称：Door） 75

4.6.1. 位置付けと機能 75
4.6.2. 機能クラス 76
4.6.3. リスククラス 76
4.6.4. 構成情報 76
4.6.5. 状態 77
4.6.6. サービスコール一覧 77
4.6.7. イベント 78
4.6.8. データ型の定義 78
4.6.9. サービスコールの詳細規定 80
4.7. Vehicle.Cabin.DoorSwitch（略称：DoorSwitch） 85
4.7.1. 位置付けと機能 85
4.7.2. 機能クラス 86
4.7.3. 構成情報 86
4.7.4. 状態 87
4.7.5. サービスコール一覧 87
4.7.6. イベント 87
4.7.7. データ型の定義 87
4.7.8. サービスコールの詳細規定 89
4.8. Vehicle.Cabin.InteriorLight（略称：InteriorLight） 91
4.8.1. 位置付けと機能 91
4.8.2. 機能クラス 93
4.8.3. リスククラス 94
4.8.4. 構成情報 94
4.8.5. 状態 95
4.8.6. サービスコール一覧 96
4.8.7. イベント 96
4.8.8. データ型の定義 97
4.8.9. サービスコールの詳細規定 100
4.9. Vehicle.Cabin.InteriorLightSwitch（略称：InteriorLightSwitch） 106
4.9.1. 位置付けと機能 106
4.9.2. 機能クラス 107
4.9.3. 構成情報 108
4.9.4. 状態 109
4.9.5. サービスコール一覧 109
4.9.6. イベント 109
4.9.7. データ型の定義 110
4.9.8. サービスコールの詳細規定 112
4.10. Vehicle.Cabin.InteriorMirror（略称：InteriorMirror） 115
4.10.1. 位置付けと機能 115
4.10.2. 機能クラス 116
4.10.3. リスククラス 116

4.10.4. 構成情報 116
4.10.5. 状態 117
4.10.6. サービスコール一覧 117
4.10.7. イベント 118
4.10.8. データ型の定義 118
4.10.9. サービスコールの詳細規定 120
4.11. Vehicle.Cabin.InteriorMirrorSwitch（略称：InteriorMirrorSwitch） 125
4.11.1. 位置付けと機能 125
4.11.2. 機能クラス 125
4.11.3. 構成情報 126
4.11.4. 状態 126
4.11.5. サービスコール一覧 126
4.11.6. イベント 126
4.11.7. データ型の定義 127
4.11.8. サービスコールの詳細規定 128
4.12. Vehicle.Cabin.Seat（略称：Seat） 131
4.12.1. 位置付けと機能 131
4.12.2. 機能クラス 131
4.12.3. リスククラス 131
4.12.4. 構成情報 132
4.12.5. 状態 132
4.12.6. サービスコール一覧 133
4.12.7. イベント 133
4.12.8. データ型の定義 134
4.12.9. サービスコールの詳細規定 136
4.13. Vehicle.Cabin.SeatSwitch（略称：SeatSwitch） 142
4.13.1. 位置付けと機能 142
4.13.2. 機能クラス 143
4.13.3. 構成情報 143
4.13.4. 状態 143
4.13.5. サービスコール一覧 143
4.13.6. イベント 144
4.13.7. データ型の定義 144
4.13.8. サービスコールの詳細規定 146
4.14. Vehicle.Cabin.Shade（略称：Shade） 148
4.14.1. 位置付けと機能 148
4.14.2. 機能クラス 149
4.14.3. リスククラス 149
4.14.4. 構成情報 149
4.14.5. 状態 150
4.14.6. サービスコール一覧 150

4.14.7. イベント . . . 151
4.14.8. データ型の定義 . . . 151
4.14.9. サービスコールの詳細規定 . . . 153
4.15. Vehicle.Cabin.ShadeSwitch（略称：ShadeSwitch） . . . 157
4.15.1. 位置付けと機能 . . . 157
4.15.2. 機能クラス . . . 157
4.15.3. 構成情報 . . . 158
4.15.4. 状態 . . . 158
4.15.5. サービスコール一覧 . . . 158
4.15.6. イベント . . . 158
4.15.7. データ型の定義 . . . 159
4.15.8. サービスコールの詳細規定 . . . 160
4.16. Vehicle.Cabin.Window（略称：Window） . . . 163
4.16.1. 位置付けと機能 . . . 163
4.16.2. 機能クラス . . . 164
4.16.3. リスククラス . . . 164
4.16.4. 構成情報 . . . 164
4.16.5. 状態 . . . 165
4.16.6. サービスコール一覧 . . . 165
4.16.7. イベント . . . 166
4.16.8. データ型の定義 . . . 167
4.16.9. サービスコールの詳細規定 . . . 169
4.17. Vehicle.Cabin.WindowSwitch（略称：WindowSwitch） . . . 175
4.17.1. 位置付けと機能 . . . 175
4.17.2. 機能クラス . . . 175
4.17.3. 構成情報 . . . 176
4.17.4. 状態 . . . 176
4.17.5. サービスコール一覧 . . . 176
4.17.6. イベント . . . 176
4.17.7. データ型の定義 . . . 177
4.17.8. サービスコールの詳細規定 . . . 179
4.18. Vehicle.Cabin.Hvac（略称：HVAC） . . . 181
4.18.1. 位置付けと機能 . . . 181
4.18.2. 機能クラス . . . 182
4.18.3. リスククラス . . . 182
4.18.4. 構成情報 . . . 182
4.18.5. 状態 . . . 183
4.18.6. サービスコール一覧 . . . 183
4.18.7. イベント . . . 184
4.18.8. データ型の定義 . . . 184
4.18.9. サービスコールの詳細規定 . . . 186

4.19. Vehicle.Cabin.HvacSwitch（略称：HVACSwitch） . . . **194**
4.19.1. 位置付けと機能 . . . **194**
4.19.2. 機能クラス . . . **195**
4.19.3. 構成情報 . . . **195**
4.19.4. 状態 . . . **196**
4.19.5. サービスコール一覧 . . . **196**
4.19.6. イベント . . . **197**
4.19.7. データ型の定義 . . . **197**
4.19.8. サービスコールの詳細規定 . . . **199**
4.20. Vehicle.CargoSpace（略称：CargoSpace） . . . **202**
4.21. Vehicle.CargoSpace.Trunk（略称：Trunk） . . . **202**
4.21.1. 位置付けと機能 . . . **202**
4.21.2. 機能クラス . . . **203**
4.21.3. リスククラス . . . **203**
4.21.4. 構成情報 . . . **203**
4.21.5. 状態 . . . **204**
4.21.6. サービスコール一覧 . . . **204**
4.21.7. イベント . . . **205**
4.21.8. データ型の定義 . . . **205**
4.21.9. サービスコールの詳細規定 . . . **207**
4.22. Vehicle.CargoSpace.TrunkSwitch（略称：TrunkSwitch） . . . **212**
4.22.1. 位置付けと機能 . . . **212**
4.22.2. 機能クラス . . . **213**
4.22.3. 構成情報 . . . **213**
4.22.4. 状態 . . . **213**
4.22.5. サービスコール一覧 . . . **213**
4.22.6. イベント . . . **214**
4.22.7. データ型の定義 . . . **214**
4.22.8. サービスコールの詳細規定 . . . **215**
4.23. Vehicle.Body（略称：Body） . . . **218**
4.24. Vehicle.Body.Hood（略称：Hood） . . . **218**
4.24.1. 位置付けと機能 . . . **218**
4.24.2. 機能クラス . . . **219**
4.24.3. リスククラス . . . **219**
4.24.4. 構成情報 . . . **219**
4.24.5. 状態 . . . **219**
4.24.6. サービスコール一覧 . . . **220**
4.24.7. イベント . . . **221**
4.24.8. データ型の定義 . . . **221**
4.24.9. サービスコールの詳細規定 . . . **223**
4.25. Vehicle.Cabin.HoodSwitch（略称：HoodSwitch） . . . **226**

4.25.1. 位置付けと機能 226
4.25.2. 機能クラス 226
4.25.3. 構成情報 227
4.25.4. 状態 227
4.25.5. サービスコール一覧 228
4.25.6. イベント 228
4.25.7. データ型の定義 228
4.25.8. サービスコールの詳細規定 230
4.26. Vehicle.Body.Horn（略称：Horn） 232
4.26.1. 位置付けと機能 232
4.26.2. 機能クラス 233
4.26.3. リスククラス 233
4.26.4. 構成情報 233
4.26.5. 状態 234
4.26.6. サービスコール一覧 235
4.26.7. イベント 236
4.26.8. データ型の定義 236
4.26.9. サービスコールの詳細規定 238
4.27. Vehicle.Body.Mirror（略称：Mirror） 243
4.27.1. 位置付けと機能 244
4.27.2. 機能クラス 244
4.27.3. リスククラス 244
4.27.4. 構成情報 245
4.27.5. 状態 245
4.27.6. サービスコール一覧 246
4.27.7. イベント 246
4.27.8. データ型の定義 247
4.27.9. サービスコールの詳細規定 249
4.28. Vehicle.Body.MirrorSwitch（略称：MirrorSwitch） 253
4.28.1. 位置付けと機能 254
4.28.2. 機能クラス 254
4.28.3. 構成情報 254
4.28.4. 状態 255
4.28.5. サービスコール一覧 255
4.28.6. イベント 255
4.28.7. データ型の定義 257
4.28.8. サービスコールの詳細規定 259
4.29. Vehicle.Body.Wiper（略称：Wiper） 261
4.29.1. 位置付けと機能 262
4.29.2. 機能クラス 262
4.29.3. リスククラス 262

4.29.4. 構成情報 263
4.29.5. 状態 263
4.29.6. サービスコール一覧 264
4.29.7. イベント 265
4.29.8. データ型の定義 265
4.29.9. サービスコールの詳細規定 267
4.30. Vehicle.Body.WiperSwitch（略称：WiperSwitch） 273
4.30.1. 位置付けと機能 273
4.30.2. 機能クラス 274
4.30.3. 構成情報 274
4.30.4. 状態 275
4.30.5. サービスコール一覧 275
4.30.6. イベント 275
4.30.7. データ型の定義 276
4.30.8. サービスコールの詳細規定 278
4.31. Vehicle.Powertrain.TractionBattery.Charging（略称：Charging） 281
4.31.1. 位置付けと機能 281
4.31.2. 機能クラス 281
4.31.3. リスククラス 281
4.31.4. 構成情報 281
4.31.5. 状態 282
4.31.6. サービスコール一覧 282
4.31.7. イベント 282
4.31.8. データ型の定義 283
4.31.9. サービスコールの詳細規定 285
4.32. Vehicle.Powertrain.TractionBattery（略称：TractionBattery） 290
4.32.1. 位置付けと機能 290
4.32.2. 機能クラス 290
4.32.3. リスククラス 290
4.32.4. 構成情報 291
4.32.5. 状態 291
4.32.6. サービスコール一覧 291
4.32.7. イベント 291
4.32.8. データ型の定義 292
4.32.9. サービスコールの詳細規定 294
4.33. Vehicle.Powertrain.CombustionEngine（略称：CombustionEngine） 296
4.33.1. 位置付けと機能 296
4.33.2. 機能クラス 296
4.33.3. リスククラス 296
4.33.4. 構成情報 297
4.33.5. 状態 297

4.33.6. サービスコール一覧 297
4.33.7. イベント 297
4.33.8. データ型の定義 298
4.33.9. サービスコールの詳細規定 299
4.34. Vehicle.Powertrain.FuelSystem（略称：FuelSystem） 302
4.34.1. 位置付けと機能 302
4.34.2. 機能クラス 302
4.34.3. リスククラス 302
4.34.4. 構成情報 302
4.34.5. 状態 303
4.34.6. サービスコール一覧 303
4.34.7. イベント 303
4.34.8. データ型の定義 304
4.34.9. サービスコールの詳細規定 305
4.35. Vehicle.Infotainment（略称：Infotainment） 307
4.36. Vehicle.HMI（略称：HMI） 307
4.37. Vehicle.HMI.Display（略称：Display） 307
4.37.1. 位置付けと機能 308
4.37.2. 機能クラス 308
4.37.3. リスククラス 308
4.37.4. 構成情報 308
4.37.5. 状態 309
4.37.6. サービスコール一覧 309
4.37.7. イベント 309
4.37.8. データ型の定義 309
4.37.9. サービスコールの詳細規定 311
4.38. Vehicle.HMI.View（略称：View） 313
4.38.1. 位置付けと機能 313
4.38.2. 機能クラス 314
4.38.3. リスククラス 314
4.38.4. 構成情報 314
4.38.5. 状態 314
4.38.6. サービスコール一覧 315
4.38.7. イベント 315
4.38.8. データ型の定義 315
4.38.9. サービスコールの詳細規定 324
4.39. Vehicle.HMI.Input.Keyboard（略称：Keyboard） 330
4.39.1. 位置付けと機能 330
4.39.2. 機能クラス 330
4.39.3. リスククラス 330
4.39.4. 構成情報 330

4.39.5. 状態 330
4.39.6. サービスコール一覧 330
4.39.7. イベント 330
4.39.8. データ型の定義 330
4.39.9. サービスコールの詳細規定 330
4.40. Vehicle.HMI.Input.Mouse（略称：Mouse） 331
4.40.1. 位置付けと機能 332
4.40.2. 機能クラス 332
4.40.3. リスククラス 332
4.40.4. 構成情報 332
4.40.5. 状態 332
4.40.6. サービスコール一覧 332
4.40.7. イベント 332
4.40.8. データ型の定義 332
4.40.9. サービスコールの詳細規定 332
4.41. Vehicle.HMI.Input.Touch（略称：Touch） 334
4.41.1. 位置付けと機能 334
4.41.2. 機能クラス 335
4.41.3. リスククラス 335
4.41.4. 構成情報 335
4.41.5. 状態 335
4.41.6. サービスコール一覧 335
4.41.7. イベント 335
4.41.8. データ型の定義 335
4.41.9. サービスコールの詳細規定 335
4.42. Vehicle.HMI.Input.Joystick（略称：Joystick） 340
4.42.1. 位置付けと機能 340
4.42.2. 機能クラス 341
4.42.3. リスククラス 341
4.42.4. 構成情報 341
4.42.5. 状態 341
4.42.6. サービスコール一覧 341
4.42.7. イベント 341
4.42.8. データ型の定義 341
4.42.9. サービスコールの詳細規定 341
4.43. Vehicle.HMI.Sound.Common.SoundEffect（略称：SoundEffect） 343
4.43.1. 位置付けと機能 343
4.43.2. 機能クラス 344
4.43.3. リスククラス 344
4.43.4. 構成情報 344
4.43.5. 状態 344

4.43.6. サービスコール一覧 344
4.43.7. イベント 344
4.43.8. データ型の定義 344
4.43.9. サービスコールの詳細規定 344
4.44. Vehicle.HMI.Sound.App.SoundEffect（略称：SoundEffect） 346
4.44.1. 位置付けと機能 346
4.44.2. 機能クラス 346
4.44.3. リスククラス 346
4.44.4. 構成情報 346
4.44.5. 状態 346
4.44.6. サービスコール一覧 346
4.44.7. イベント 346
4.44.8. データ型の定義 347
4.44.9. サービスコールの詳細規定 347
4.45. Vehicle.HMI.Sound.App.Music（略称：Music） 351
4.45.1. 位置付けと機能 351
4.45.2. 機能クラス 352
4.45.3. リスククラス 352
4.45.4. 構成情報 352
4.45.5. 状態 352
4.45.6. サービスコール一覧 352
4.45.7. イベント 352
4.45.8. データ型の定義 352
4.45.9. サービスコールの詳細規定 352
4.46. Vehicle.HMI.Sound.App.Voice（略称：Voice） 355
4.46.1. 位置付けと機能 355
4.46.2. 機能クラス 355
4.46.3. リスククラス 355
4.46.4. 構成情報 355
4.46.5. 状態 355
4.46.6. サービスコール一覧 355
4.46.7. イベント 355
4.46.8. データ型の定義 355
4.46.9. サービスコールの詳細規定 356
4.47. Vehicle.HMI.Sound.Alert.SoundEffect（略称：SoundEffect） 361
4.47.1. 位置付けと機能 361
4.47.2. 機能クラス 362
4.47.3. リスククラス 362
4.47.4. 構成情報 362
4.47.5. 状態 362
4.47.6. サービスコール一覧 362

4.47.7. イベント . . . 362
4.47.8. データ型の定義 . . . 362
4.47.9. サービスコールの詳細規定 . . . 362
4.48. Vehicle.HMI.Sound.Alert.Voice（略称：Voice） . . . 363
4.48.1. 位置付けと機能 . . . 363
4.48.2. 機能クラス . . . 363
4.48.3. リスククラス . . . 363
4.48.4. 構成情報 . . . 363
4.48.5. 状態 . . . 363
4.48.6. サービスコール一覧 . . . 363
4.48.7. イベント . . . 363
4.48.8. データ型の定義 . . . 363
4.48.9. サービスコールの詳細規定 . . . 363
4.49. Vehicle.HMI.Switch（略称：Switch） . . . 364
4.49.1. 位置付けと機能 . . . 364
4.49.2. 機能クラス . . . 365
4.49.3. リスククラス . . . 365
4.49.4. 構成情報 . . . 365
4.49.5. 状態 . . . 365
4.49.6. サービスコール一覧 . . . 365
4.49.7. イベント . . . 366
4.49.8. データ型の定義 . . . 366
4.49.9. サービスコールの詳細規定 . . . 368
4.50. Vehicle.Motion（略称：Motion） . . . 373
4.50.1. 位置付けと機能 . . . 373
4.50.2. 機能クラス . . . 375
4.50.3. リスククラス . . . 375
4.50.4. 構成情報 . . . 376
4.50.5. 状態 . . . 376
4.50.6. サービスコール一覧 . . . 377
4.50.7. イベント . . . 378
4.50.8. データ型の定義 . . . 379
4.50.9. サービスコールの詳細規定 . . . 382
4.51. Vehicle.Occupant（略称：Occupant） . . . 389
4.52. Vehicle.Atmosphere（略称：Atmosphere） . . . 389
5. 拡張ビークルAPI . . . 390
5.1. Vehicle.ADADAS（略称：ADADAS） . . . 390
5.1.1. 位置付けと機能 . . . 390
5.1.2. 機能クラス . . . 394
5.1.3. リスククラス . . . 394
5.1.4. 構成情報 . . . 395

5.1.5. 状態 395
5.1.6. サービスコール一覧 396
5.1.7. イベント 396
5.1.8. データ型の定義 397
5.1.9. サービスコールの詳細規定 404
5.2. Vehicle.ADADAS.ACC（略称：ACC） 409
5.2.1. 位置付けと機能 409
5.2.2. 機能クラス 411
5.2.3. リスククラス 412
5.2.4. 構成情報 412
5.2.5. 状態 413
5.2.6. サービスコール一覧 414
5.2.7. イベント 414
5.2.8. データ型の定義 416
5.2.9. サービスコールの詳細規定 421
5.3. Vehicle.ADADAS.LKA（略称：LKA） 428
5.3.1. 位置付けと機能 428
5.3.2. 機能クラス 432
5.3.3. リスククラス 432
5.3.4. 構成情報 432
5.3.5. 状態 433
5.3.6. サービスコール一覧 434
5.3.7. イベント 434
5.3.8. データ型の定義 435
5.3.9. サービスコールの詳細規定 439
5.4. Vehicle.ADADAS.LDW（略称：LDW） 446
5.4.1. 位置付けと機能 446
5.4.2. 機能クラス 449
5.4.3. リスククラス 449
5.4.4. 構成情報 449
5.4.5. 状態 450
5.4.6. サービスコール一覧 451
5.4.7. イベント 451
5.4.8. データ型の定義 452
5.4.9. サービスコールの詳細規定 456
5.5. Vehicle.ADADAS.LCA（略称：LCA） 461
5.5.1. 位置付けと機能 461
5.5.2. 機能クラス 465
5.5.3. リスククラス 465
5.5.4. 構成情報 465
5.5.5. 状態 466

5.5.6. サービスコール一覧 467
5.5.7. イベント 467
5.5.8. データ型の定義 468
5.5.9. サービスコールの詳細規定 474
5.6. Vehicle.ADADAS.AEB（略称：AEB） 479
5.6.1. 位置付けと機能 479
5.6.2. 機能クラス 483
5.6.3. リスククラス 483
5.6.4. 構成情報 483
5.6.5. 状態 484
5.6.6. サービスコール一覧 485
5.6.7. イベント 485
5.6.8. データ型の定義 486
5.6.9. サービスコールの詳細規定 490
5.7. Vehicle.ADADAS.ParkingAssist（略称：ParkingAssist） 496
5.7.1. 位置付けと機能 496
5.7.2. 機能クラス 500
5.7.3. リスククラス 501
5.7.4. 構成情報 501
5.7.5. 状態 502
5.7.6. サービスコール一覧 503
5.7.7. イベント 503
5.7.8. データ型の定義 504
5.7.9. サービスコールの詳細規定 509
5.8. Vehicle.ADADAS.AutomatedParking（略称：AutomatedParking） 514
5.8.1. 位置付けと機能 514
5.8.2. 機能クラス 518
5.8.3. リスククラス 518
5.8.4. 構成情報 518
5.8.5. 状態 519
5.8.6. サービスコール一覧 520
5.8.7. イベント 521
5.8.8. データ型の定義 521
5.8.9. サービスコールの詳細規定 527
5.9. Vehicle.ADADAS.BSM（略称：BSM） 535
5.9.1. 位置付けと機能 535
5.9.2. 機能クラス 538
5.9.3. リスククラス 538
5.9.4. 構成情報 538
5.9.5. 状態 539
5.9.6. サービスコール一覧 540

5.9.7. イベント . . . 541
5.9.8. データ型の定義 . . . 541
5.9.9. サービスコールの詳細規定 . . . 546
5.10. Vehicle.ADADAS.FCW（略称：FCW） . . . 551
5.10.1. 位置付けと機能 . . . 551
5.10.2. 機能クラス . . . 555
5.10.3. リスククラス . . . 555
5.10.4. 構成情報 . . . 555
5.10.5. 状態 . . . 556
5.10.6. サービスコール一覧 . . . 557
5.10.7. イベント . . . 557
5.10.8. データ型の定義 . . . 558
5.10.9. サービスコールの詳細規定 . . . 563
5.11. Vehicle.ADADAS.TSR（略称：TSR） . . . 569
5.11.1. 位置付けと機能 . . . 569
5.11.2. 機能クラス . . . 573
5.11.3. リスククラス . . . 573
5.11.4. 構成情報 . . . 573
5.11.5. 状態 . . . 574
5.11.6. サービスコール一覧 . . . 575
5.11.7. イベント . . . 575
5.11.8. データ型の定義 . . . 576
5.11.9. サービスコールの詳細規定 . . . 580
5.12. Vehicle.ADADAS.RCTA（略称：RCTA） . . . 585
5.12.1. 位置付けと機能 . . . 585
5.12.2. 機能クラス . . . 589
5.12.3. リスククラス . . . 589
5.12.4. 構成情報 . . . 589
5.12.5. 状態 . . . 590
5.12.6. サービスコール一覧 . . . 591
5.12.7. イベント . . . 591
5.12.8. データ型の定義 . . . 592
5.12.9. サービスコールの詳細規定 . . . 596
5.13. Vehicle.CurrentLocation（略称：CurrentLocation） . . . 601
5.13.1. 位置付けと機能 . . . 601
5.13.2. 機能クラス . . . 603
5.13.3. リスククラス . . . 603
5.13.4. 構成情報 . . . 603
5.13.5. 状態 . . . 604
5.13.6. サービスコール一覧 . . . 606
5.13.7. イベント . . . 606

5.13.8. データ型の定義 607
5.13.9. サービスコールの詳細規定 611
5.14. Vehicle.Driver（略称：Driver） 614
5.14.1. 位置付けと機能 614
5.14.2. リスククラス 615
5.14.3. 構成情報 615
5.14.4. 状態 616
5.14.5. サービスコール一覧 616
5.14.6. イベント 617
5.14.7. データ型の定義 617
5.14.8. サービスコールの詳細規定 621
5.15. Vehicle.Driver.Device（略称：DriverDevice） 623
5.15.1. 位置付けと機能 624
5.15.2. リスククラス 624
5.15.3. 構成情報 624
5.15.4. サービスコール一覧 625
5.15.5. イベント 625
5.15.6. データ型の定義 625
5.15.7. サービスコールの詳細規定 627
5.16. Vehicle.Driver.IdentityProfile（略称：IdentityProfile） 629
5.16.1. 位置付けと機能 629
5.16.2. 機能クラス 630
5.16.3. リスククラス 631
5.16.4. 構成情報 631
5.16.5. 状態 631
5.16.6. サービスコール一覧 634
5.16.7. イベント 634
5.16.8. データ型の定義 634
5.16.9. サービスコールの詳細規定 637
5.17. Vehicle.Driver.PreferenceProfile（略称：PreferenceProfile） 640
5.17.1. 位置付けと機能 640
5.17.2. 機能クラス 641
5.17.3. リスククラス 641
5.17.4. 構成情報 641
5.17.5. 状態 641
5.17.6. サービスコール一覧 642
5.17.7. イベント 643
5.17.8. データ型の定義 643
5.17.9. サービスコールの詳細規定 645
5.18. Vehicle.Driver.PhysicalObservation（略称：PhysicalObservation） 648
5.18.1. 位置付けと機能 648

5.18.2. 機能クラス . . . 650
5.18.3. リスククラス . . . 650
5.18.4. 構成情報 . . . 650
5.18.5. 状態 . . . 651
5.18.6. サービスコール一覧 . . . 653
5.18.7. イベント . . . 653
5.18.8. データ型の定義 . . . 654
5.18.9. サービスコールの詳細規定 . . . 660
5.19. Vehicle.Driver.PhysiologicalMeasurement（略称：PhysiologicalMeasurement） . . . 663
5.19.1. 位置付けと機能 . . . 663
5.19.2. 機能クラス . . . 664
5.19.3. リスククラス . . . 664
5.19.4. 構成情報 . . . 665
5.19.5. 状態 . . . 665
5.19.6. サービスコール一覧 . . . 666
5.19.7. イベント . . . 666
5.19.8. データ型の定義 . . . 667
5.19.9. サービスコールの詳細規定 . . . 669
5.20. Vehicle.Driver.CognitiveState（略称：CognitiveState） . . . 672
5.20.1. 位置付けと機能 . . . 672
5.20.2. 機能クラス . . . 673
5.20.3. リスククラス . . . 673
5.20.4. 構成情報 . . . 673
5.20.5. 状態 . . . 674
5.20.6. サービスコール一覧 . . . 676
5.20.7. イベント . . . 676
5.20.8. データ型の定義 . . . 677
5.20.9. サービスコールの詳細規定 . . . 680
5.21. Vehicle.Driver.IntentAndCompliance.Violation（略称：IntentAndComplianceViolation） . . . 683
5.21.1. 位置付けと機能 . . . 683
5.21.2. 機能クラス . . . 684
5.21.3. リスククラス . . . 684
5.21.4. 構成情報 . . . 684
5.21.5. 状態 . . . 685
5.21.6. サービスコール一覧 . . . 686
5.21.7. イベント . . . 686
5.21.8. データ型の定義 . . . 687
5.21.9. サービスコールの詳細規定 . . . 690
5.22. Vehicle.Navigation（略称：Navi） . . . 692
5.22.1. 位置付けと機能 . . . 692

5.22.2. 機能クラス . . . 695
5.22.3. リスククラス . . . 695
5.22.4. 構成情報 . . . 696
5.22.5. 状態 . . . 696
5.22.6. サービスコール一覧 . . . 698
5.22.7. イベント . . . 698
5.22.8. データ型の定義 . . . 701
5.22.9. サービスコールの詳細規定 . . . 709
5.23. Vehicle.SurroundModel（略称：SurroundModel） . . . 717
5.23.1. 位置付けと機能 . . . 717
5.23.2. 機能クラス . . . 719
5.23.3. リスククラス . . . 720
5.23.4. 構成情報 . . . 720
5.23.5. 状態 . . . 721
5.23.6. サービスコール一覧 . . . 722
5.23.7. イベント . . . 723
5.23.8. データ型の定義 . . . 724
5.23.9. サービスコールの詳細規定 . . . 743
5.24. Vehicle.SurroundModel.LogicalStructure（略称：LogicalStructure） . . . 746
5.24.1. 位置付けと機能 . . . 746
5.24.2. 機能クラス . . . 750
5.24.3. リスククラス . . . 750
5.24.4. 構成情報 . . . 750
5.24.5. 状態 . . . 750
5.24.6. サービスコール一覧 . . . 751
5.24.7. イベント . . . 752
5.24.8. データ型の定義 . . . 753
5.24.9. サービスコールの詳細規定 . . . 767
Appendix A: 車両状態の定義 . . . 771
Appendix B: C言語向け物理API（C言語バインディング） . . . 772
B.1. データ型 . . . 772
B.2. 共通データ型 . . . 773
B.3. コアビークルAPIのデータ型 . . . 775
B.3.1. WindowのAPIのデータ型 . . . 775

Chapter 1. Open SDV API仕様の位置付け

1.1. オープンSDVとOpen SDV Initiativeの取り組み

1.1.1. SDVとオープンSDV

自動車産業が100年に1度の大変革期を迎える中，次世代モビリティの鍵としてSDV（Software Defined Vehicle）の概念が大きな注目を浴びている。SDVとは，無線通信を通じたソフトウェア更新（OTA：Over The Air software update）により，機能を継続的にアップデートすることで，新たな価値を創出することができる自動車のことを言う。

従来の自動車は，製造段階で作り込まれたハードウェアが価値の大部分を占めていた。しかし，SDVにおいては価値の源泉がハードウェアからソフトウェアへと大きく移行する。これにより，車両購入後もソフトウェアの更新を通じて機能が進化し続け，ユーザの嗜好に応じたパーソナライゼーションや，新たなモビリティサービスの継続的な提供が可能となる。

このようなSDVのポテンシャルを，個別のOEM（自動車メーカ）に閉じるのではなく，多様なプレイヤーが参画できるエコシステムの構築を目指すのが，オープンSDVの考え方である。具体的には，サードパーティ（OEMやその委託先以外の組織や個人）が開発したアプリケーションをインストールすることができる自動車を，オープンSDVと呼ぶ。

オープンSDVは，モビリティにおけるイノベーションを大きく加速させるポテンシャルを持っている。従来の自動車やオープンでないSDVにおいて，自動車に新しい機能を追加し，新しいサービスを提供するためには，OEMの同意と協力が不可欠であった。しかし，オープンSDVの世界では，サードパーティが自らアプリケーションを開発・提供することで，独自に新しい価値を生み出すことが可能となる。これにより，自動車は単なる移動手段から，スマートフォンのように，サードパーティの多様なサービスを載せる基盤（プラットフォーム）へとその商品性を大きく変化させることになる。

1.1.2. ビークルAPIと標準化

SDVにおいて，アプリケーションから車両の機能を利用し，車両を操作するためには，明確に定義されたインタフェースが必要となる。このインタフェースが，ビークルAPIである。ビークルAPIは，車両の物理的な構造やハードウェアの差異を吸収し，アプリケーション開発者が個々のハードウェアの詳細を意識することなく車両にアクセスする手段を提供する。そして，このビークルAPIを介した要求を受け付け，車両のハードウェアリソースおよびソフトウェアリソースを統合的に管理・制御するソフトウェアプラットフォームが，ビークルOSである。

オープンSDVのエコシステムを形成するためには，OEMの枠を超えたビークルAPIの標準化が不可欠である。サードパーティのアプリケーション開発者の立場からは，開発投資に見合う十分な市場規模を確保するために，開発したアプリケーションが可能な限り多くの車両に共通して適用できることが望まれる。アプリケーションが特定のOEMの車両にしか適用できない状態では，市場規模に限界が生じることから，ビークルAPIが標準化されていることが強く求められる。

また，ビークルAPIの標準化は，SDV化に伴う最大の技術課題であるソフトウェア開発工数の爆発的増加を解決するための手段としても極めて有用である。開発工数を抑制するためには，ハードウェアとソフトウェアのライフサイクルを分離し，ハードウェアの差異に依存せずにソフトウェアを開発できるアーキテクチャへの移行が必要となる。その分離の界面となるビークルAPIを業界全体で標準化することで，OEM間や車種間でのソフトウェア資産の流用・再利用が容易になり，業界全体の開発効率が飛躍的に向上する。

さらに，近年の自動運転技術，特にEnd-to-End（E2E）AI自動運転の急速な進展においても，ビークルAPIの標準化が果たす役割は大きい。高度な自動運転システムを実現するにあたり，OEMが外部の自動運

転システム企業の技術を採用するケースが増加している。この両者間を繋ぐインタフェースを標準化しておくことで，OEMは特定の自動運転システム企業へのロックインを回避し，異なるシステムを柔軟に採用することが可能となる。これは，自動運転分野における健全な競争を促進し，技術革新を持続させるためにも重要である。

現在，国内外においてビークルAPIの標準化に向けた様々な活動が進められている。例えば，国際的なコンソーシアムであるCOVESAは，車両データのモデルであるVSS（Vehicle Signal Specification）と，そのデータにアクセスするためのVISS（Vehicle Information Service Specification）の仕様を策定している。また，車載ソフトウェアや車載ネットワークに関する国内の標準化団体であるJASPARも，API技術WGを設置して車両信号を利用するAPIの開発を進め，2026年4月にはYoriito Vehicle APIをオープンソースソフトウェアとして公開した。さらに，中国のCAAM-SDV（中国自動車工業協会 SDV委員会）も，ビークルAPI仕様を公表している。

一方で，車両制御の専門知識を持たないサードパーティのアプリケーション開発者が，車両の機能を安全かつ容易に利用できるAPIを実現するには，なお検討すべき課題がある。

1.1.3. Open SDV Initiativeの取り組み

Open SDV Initiativeは，近い将来にオープンSDVが実現・普及することを想定して，オープンSDVのためのビークルAPIを策定することを目的に，名古屋大学が産業界に呼びかけて進めている活動である。策定したビークルAPIは，JASPARやCOVESAなどの標準化団体に提案することを考えている。Open SDV Initiativeで策定したビークルAPI仕様を，Open SDV API仕様と呼ぶ。本書は，その仕様を規定するものである。

Open SDV Initiativeは，2024年6月に活動の立ち上げを発表し，2024年10月に本格的に活動を開始した。2026年9月時点で，約60社の企業が活動に参加している。ビークルAPIの策定が急がれる状況であり，他団体との協調のためにも，策定の途上であっても成果を積極的に公表すべきと考えており，活動開始以降，半年ごとにその時点での仕様を公表している。

また，Open SDV Initiativeでは，API仕様の策定に留まらず，それを評価するためのシミュレータや実車を用いたPoC（Proof of Concept）の開発も行っている。さらに，オープンSDVにより実現可能となるサービス検討からビークルAPIへの要求を抽出する活動や，オープンSDVが提供するサービスを体験するためのシミュレーション環境であるMESH（Mobility Experience Simulation Hub）の開発も並行して進めている。

1.1.4. 本仕様の策定状況

Open SDV API仕様の本バージョン（バージョン：202609β）は，仕様策定のために立ち上げたAPIコンセプトWG，ボディWG，AD/ADAS WG，HMI WG，エネルギーWGの5つのWGの現時点の検討成果を取りまとめたものである。本格的な活動開始から2年で，ビークルAPIのコンセプトや個々のAPIの検討が進んできたが，各WGの検討成果間における整合性の確保や，専門的知見を持つメンバの不足により未検討となっている領域の補完など，API群としての網羅性において今後の課題を残している。

今後も，個々のAPIの完成度を上げるとともに，その有効性検証のための実装を通じて，ビークルAPI仕様の継続的な拡充とブラッシュアップを進めていく計画である。

1.2. APIの位置付けと策定方針

1.2.1. 策定するビークルAPIの位置付け

ビークルOSとは，ハードウェアとしての車両を，アプリケーション開発者およびユーザが，より容易に，

より効率的に，かつ安全に利用できるようにすることを目的として，車両のハードウェアリソースおよびソフトウェアリソースを管理し，機能提供を行う一連のソフトウェアをいう。アプリケーションからビークルOSの機能を利用するためのインタフェースをビークルAPIと呼び，本仕様の規定対象とする。

車載コンピュータは車両の一部であるため，車載コンピュータのOSはビークルOSの一部である。ビークルAPIは，車両の持つ様々な機能を提供するインタフェースであり，（コンピュータの）OSが提供するAPIも含む。本仕様では，（コンピュータの）OSのAPIについては，可能な限り既存の仕様を活用することとし，主たる策定対象とはしない。具体的には，AUTOSAR Adaptive Platform（AUTOSAR AP）やAndroid Automotiveが提供する機能のAPIは，原則として策定対象外とする。ただし，既存のOS APIのみではビークルAPIとして不十分である場合には，新たに仕様を策定する。

また，アプリケーションが車両のハードウェアを直接操作したい場合もあるため，車両のハードウェア抽象化層に相当するビークルデバイスインタフェースについても，必要に応じて策定する。

1.2.2. ビークルAPIの利用者と適用範囲

策定するビークルAPIは，OEMおよびその委託先のサプライヤに加え，サードパーティの開発者も利用することを想定する。そのため，自動車分野に必ずしも精通していない開発者にとっても利用しやすい仕様とする。

対象とする車両は，まずは一般の乗用車（シェアリング車両やタクシーを含む）を想定し，可能な範囲で，バスやトラックなどのサービスカーへも適用できるように配慮する。

また，対象とするアプリケーションとしては，ボディ制御，自動運転や運転支援，HMIのパーソナライズ，情報提供，エンターテインメント，プローブデータ活用，エネルギー管理など多岐にわたる用途を想定するが，将来的に新たな用途が出現する可能性を考慮し，高い汎用性および拡張性を備えたAPIとする。

なお，クラウド上のAPI（クラウドサーバ上やスマートフォン上のアプリケーションから，ネットワークを介して車両の機能を操作するためのAPI）も重要な課題であると認識しているが，まずは車両上のAPIにフォーカスする。これは，車両の機能を操作するための論理APIは，車両上とクラウド上で共通化できる部分が多いと考えられるためである。当面は，クラウド上のアプリケーションが車両上のアプリケーションを介して間接的に車両を操作するものと想定する。クラウド上に蓄積されたデータへのアクセスなど，この想定では扱えない機能は，別途検討する。

1.2.3. ビークルAPIに対する要件

ビークルAPIの策定にあたっては，以下の要件を満たすことを目標とする。

(1) アプリケーションの開発を容易にすること

一度の開発で，多くの車両に適用できるアプリケーションが開発できること。また，自動車に詳しくないサードパーティのアプリケーション開発者にとっても使いやすいAPIであること。

(2) OEMによる車両の差別化を阻害しないこと

OEMによる車両の差別化を通じた競争を阻害しないために，ハードウェア構成の自由度を狭めず，多様な車両に適用できるAPIであること。

(3) 安全性に関わるアプリケーションを扱えること

OEMやその委託先が開発するアプリケーションも同じAPIを使用することを想定しているため，安全性に関わるアプリケーションを構築できるAPIとすること。また，安全性に関わるAPIをサードパーティに自由に使用させるわけにはいかない一方で，それを一切許さないものとすると，価値の高いアプリケーション

が開発できなくなる可能性がある。そこで，どのアプリケーションにどのAPIの使用を認めるかを，OEMとユーザが決定できる仕組みを用意して，安全性に関わるアプリケーションを扱えるAPIとすること。

1.2.4. 要件を満たすためのアプローチ

前節の(1)及び(2)の要件を満たすために，本仕様では以下のアプローチを採用 する。

(1) 車両の多様性を吸収するための車両の構成記述

車両のハードウェアの違いは大きいため，APIで完全に隠蔽することは不可能と考えられる。そこで，アプリケーション側で車両の違いを吸収できるよう，車両の構成記述法を定める。

例えば，車両によってウィンドウ（窓）の数が異なることに対しては，ウィンドウのリストをAPIを使って取得し，各ウィンドウに対してAPIを呼び出す方法をとる。このようなケースで，車両ごとの構成の違いを意識せずに使えるAPI（例えば，すべてのウィンドウを操作するAPI）が必要な場合には，ライブラリ等で用意する。

ただし，車両（主にハードウェア）の機能不足により，アプリケーションが動作できない場合があるのはやむを得ない。

(2) サードパーティの開発者が使いやすいサービスベースのAPI

車両の状態を表現するシグナル（車速，ドアの開閉，ウィンドウの開度など）を読み書きするシグナルベースのAPIではなく，「ウィンドウを開閉する」といった操作を要求するサービスベースのAPIとする。シグナルベースのAPIが状態の管理を中心とするのに対し，本仕様では，アプリケーションが要求した操作の管理を重視する。

具体的には，アプリケーションが要求した操作について，達成されたか，他の操作や安全機能の介入によってオーバライドされたか，オーバライドされた場合はその理由を，要求と対応付けて把握できるようにする。これにより，車両の状態変化から自分の操作の結果を推測する必要が少なくなり，車両制御に不慣れなサードパーティのアプリケーション開発者にとっても使いやすいAPIとなる。

また，複数のアプリケーションが同じ対象を操作する場合，シグナルの現在値や変更後の値だけでは，各アプリケーションによる操作を区別して管理できない。そこで，操作の結果を要求元のアプリケーションが把握できるよう，制御要求と要求元のアプリケーションを対応付けて管理する。この仕組みを用いて，要求元や優先度に応じた調停，ロックによる排他制御を行うことで，独立して開発されたアプリケーションの共存を可能にする。

(3) 実装の自由度を確保する抽象度の高いAPI

以下の理由から，APIが提供する機能や動作を規定しつつ，ビークルOSの実現方法や内部の振る舞いを過度に制約しないようにする。APIの機能・動作を実装コードを示すことによって厳密に規定する方法は，ビークルOSの実現方法や内部の振る舞いまで制約するおそれがあるため，本仕様では主たる仕様記述方法として用いない。

(a) OEMによる車両の差別化を妨げないため

APIが提供する機能が共通であっても，その実現方法や性能，操作感などによって，OEMが車両を差別化できることが望ましい。そのため，アプリケーションから利用する上で必要な範囲で機能や動作を規定し，車両やビークルOSの側に実現方法や動作の自由度を残す。

(b) 多様なE/Eアーキテクチャに適用できるようにするため

E/Eアーキテクチャの自由度を狭めないために，アプリケーションを実行するECUは限定しない（ただし，Section 1.4で述べるように，ある程度の想定は行う）。下位層に分散プラットフォーム（AUTOSAR APやROSなど）があれば，異なるECU上で実現されたサービスをネットワーク経由で呼び出

すことができるためである。

また，現状のE/Eアーキテクチャ上でも効率的に実装できるよう，アプリケーションを実行するECUと，車両の標準的な制御を実現するアプリケーションを実行するECUが，CANで接続されている場合にも配慮する。

例えば，状態が頻繁に変化した場合でも，アプリケーションが途中の状態変化をすべて把握する必要はなく，最終的な状態や操作結果がわかればよい場合がある。このような場合には，ユースケースが成立する範囲で，アプリケーションに伝えるイベントの抜けを許容し，通信や状態管理にかかる負担を抑えられる仕様とする。

(c) 多様なプログラミング環境に対応できるようにするため

プログラミング環境（プログラミング言語やコンピュータのOS）に依存しない論理API（セマンティクスレベルのAPI）を先行して策定する。論理APIが共通であれば，物理API（シンタックスレベルのAPI）が変わったとしても，アプリケーションの移植作業は機械的に行える部分が多く，その工数は小さいと考えられる。

論理APIでは，APIが提供する機能や動作，およびAPIを介して受け渡しされるデータが取りうる値とその意味を規定する。数値データについては，単位が異なるとアプリケーションの修正範囲が大きくなるため，単位まで規定する。

論理APIの策定に続いて，代表的なプログラミング環境に対する物理API（バインディング）を規定する。これは，異なるプログラミング環境向けの物理API間でのデータ変換等のオーバヘッドが大きくなることを防ぐためと，シミュレータやPoCの開発にあたって必要なためである。

(4) コアビークルAPIと拡張ビークルAPI

ビークルAPIを定義するためには，アプリケーションとビークルOSの役割分担を決める必要がある。この時，アプリケーション開発者にとっての使いやすさを重視すると，車両の違いを吸収するために必要な機能に加え，複数の操作を組み合わせるなどの利便性を高める機能も求められる。

そこで，ビークルAPIを，車両の違いを吸収するためのコアビークルAPIと，アプリケーション開発者の利便性を高めるための拡張ビークルAPIの2階層で構成する。拡張ビークルAPIは，コアビークルAPIを利用するライブラリ等によって実現することも想定する。

1.2.5. リスクのあるAPIの扱い

Section 1.2.3の(3)の要件に対応して，策定するビークルAPIには，使用により危険を生じる可能性があるAPIを排除しないものとする。

APIの使用により危険を生じる可能性がある場合に，車両およびビークルOSで，危険を防止する機能（安全機能）を持っていることが望ましい。そのため，策定するAPIは，可能な範囲で，車両とビークルOSの持つ安全機能により危険を防止できるようなものとする。

一方で，車両とビークルOSが特定の安全機能を備えることを必須とすると，APIを適用できる車両を限定してしまう可能性がある。そこで，Section 1.2.3の(2)の要件を踏まえて，車両とビークルOSが特定の安全機能を持つことを必須とはしない。

あるAPIを使用することでリスクがあるか，そのようなリスクが許容されるかは，以下の要因によって変化する。

- あるAPIにリスクがあるかどうかは，車両とビークルOSが持つ安全機能によって異なる。
- 現時点ではアプリケーションでの対応が必要なリスクに，技術の発展（車両やビークルOSの安全機能の進化）により，車両やビークルOSで対応できるようになる可能性がある。

- リスクのあるAPIをサードパーティに使わせて良いかどうかが，国や地域の規制によって異なる場合もある。

そこで，どのアプリケーションにどのAPIの使用を認めるかを，OEMとユーザが決定できる仕組みとして，「リスク制御機能」を導入する。リスク制御機能の概要は以下の通り。

(a) リスククラスの定義

オブジェクトの操作によって生じるリスク（人身，財産，物品，環境，プライバシ上のリスク等）をリスククラスとして分類・定義する。

(b) 車両の構成記述でのリスク記述

車両やビークルOSの持つ安全機能で対応できていないリスククラスの集合を，車両の構成記述に記述する。

(c) アプリケーションでのリスクへの対応とOEMによる審査

アプリケーションで対応したリスククラスの集合と使用できる操作優先度の最高値を，アプリケーションのマニフェストに記述し，OEMがその妥当性を審査して署名を付与する。

(d) ユーザの承認

車両の安全機能で対応できておらず，アプリケーションで対応するリスクについては，アプリケーション提供者がその対応を担うことについて，ユーザの承認を求める。

(e) 操作可否の決定

ビークルOSは，以上の情報を元に，APIによる操作の可否を決定する。

1.3. 想定するシステム階層

1.3.1. ソフトウェアのモジュール化/階層化の目的

ソフトウェア（特にOS）をモジュール化（階層化を含む）する際の目的の1つは，ハードウェアや周辺環境（外部との通信仕様や交通規則など）の変更があった場合に，1つのモジュールのみの変更で対応できるようにすることである。

例えば，車両のハードウェアに変更があった場合には，ビークルOSのみを変更すれば，アプリケーションはそのまま使用できることが望ましい。また，外部との通信仕様，交通規則（国・地域によって異なる），道路地図の仕様の違いに対しては，1つのモジュールの差し替えのみで対応できるのが望ましい。

本仕様では，この目的が達成できるように，システムの階層化を行う。

1.3.2. システム全体の階層構造

Figure 1に，想定するシステム全体の階層構造を示す。

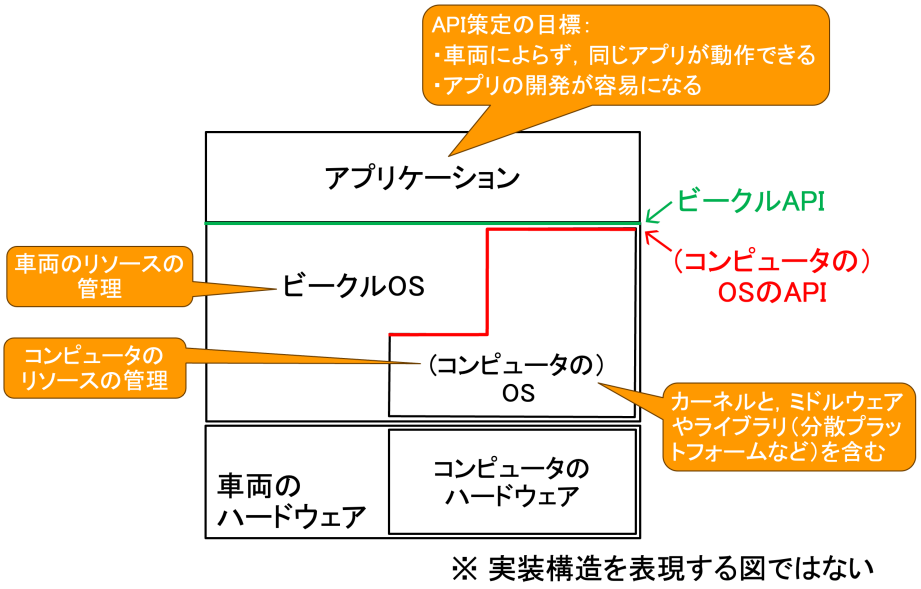


Figure 1. 想定するシステム全体の階層構造

Section 1.2.1で述べた通り，車載コンピュータは車両のハードウェアの一部と位置付けられる。したがって，車両のリソースを管理するビークルOSは，コンピュータのリソースを管理する（コンピュータの）OSを含むものと位置付けられる。

ここで，OSという用語は，広義のOSを意味しており，ソフトウェアプラットフォーム（SPF）と同義で用いている。具体的には，図中の（コンピュータの）OSには，OSカーネルに加えて，ミドルウェアやライブラリ，さらにはAUTOSAR　AP（のARA::COM）やROSのような分散プラットフォームを含んでいる。AUTOSAR APのPlatform Foundation FCsに含まれるサービスは，ここに含まれるものと位置付ける。

1.3.3. ビークルOSの内部構造

Figure 2に，ビークルOSの内部構造を示す。

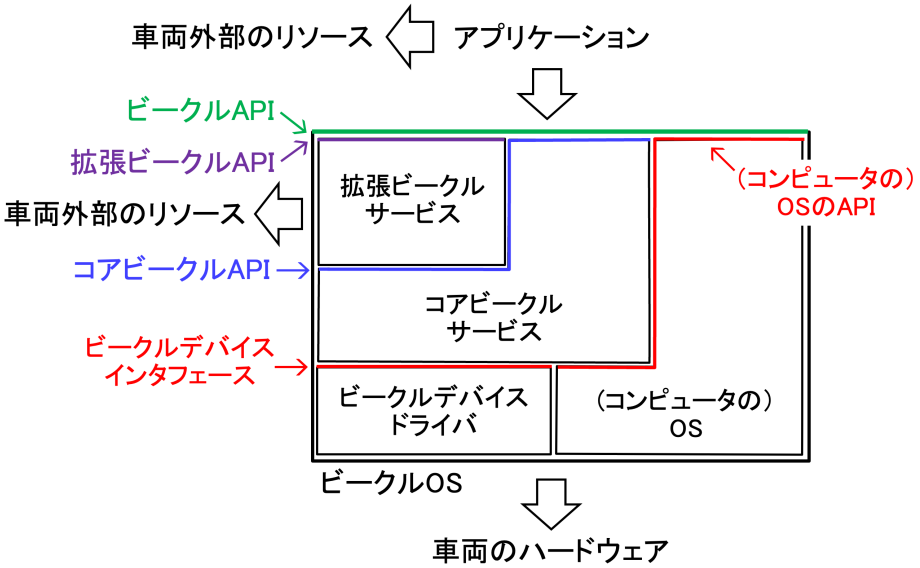


Figure 2. ビークルOSの内部構造

(1) コアビークルサービスとコアビークルAPI

コアビークルサービスは，車両のハードウェアの違いの吸収，車両のハードウェアへのアクセス制御・調停，（可能な範囲での）安全性の確保という3つの役割を担うソフトウェア階層である。コアビークルサービスの機能は，これらの役割を果たせる範囲で最小限に留め，車両のハードウェア以外のリソースを

管理する機能は，原則として持たせない方針とする。

コアビークルサービスは，車両のハードウェアの違いを吸収する役割を持つことから，OEMが提供することを想定している。OEM自身やOEMから委託を受けたサプライヤが開発することもあるが，OSメーカが開発してOEMに提供することや，OEMがオープンソースソフトウェアのコアビークルサービスを利用することも考えられる。

アプリケーションや他のビークルサービスが，コアビークルサービスを使用するためのインタフェースを，コアビークルAPIと呼ぶ。コアビークルAPIは，車両によらず同じアプリケーションを動作させるために鍵となるAPIであり，標準化と継続性が強く求められる。

(2) 拡張ビークルサービスと拡張ビークルAPI

拡張ビークルサービスは，アプリケーションの開発を容易にするためのソフトウェア階層である。例えば道路地図の管理のように，車両のハードウェア以外のリソースを管理する機能は，拡張ビークルサービスとして実現する。また，自動運転ソフトウェアやナビゲーションソフトウェアのように，他のアプリケーションから要求を受けるアプリケーションも，拡張ビークルサービスと位置付けることができる。

拡張ビークルサービスは，OEMが提供する場合もあれば，サードパーティが提供する場合もある。その開発は，OEM，サプライヤ，OSメーカ，アプリケーション開発者のいずれが行う可能性もある。

アプリケーションや他のビークルサービスが，拡張ビークルサービスを使用するためのインタフェースを，拡張ビークルAPIと呼ぶ。拡張ビークルAPIは，アプリケーション開発者の視点で使いやすいAPIであることが重要である。車両外部のリソースを管理する機能に対して，1つのAPIに標準化するのは難しい場合もあることから，類似の機能を持つ拡張ビークルAPIが複数あってもよいものとする。

なお，コアビークルサービスと拡張ビークルサービスを総称して，ビークルサービスと呼ぶ。ビークルAPIは，コアビークルAPI，拡張ビークルAPI，（コンピュータの）OSのAPIで構成される。

(3) ビークルデバイスドライバ

ビークルデバイスドライバは，車両のハードウェアデバイスを，それぞれ単独で抽象化するためのソフトウェア階層である。

ただし，何を1つのハードウェアデバイスとして扱うか（例えば，ハイブリッド車において，パワートレイン全体を1つのハードウェアデバイスと扱うか，エンジンやモータ等を別々のハードウェアデバイスと扱うか）や，ハードウェアデバイスをどの程度抽象化するか（例えば，カメラからの生画像をそのまま扱うか，画像の認識結果を扱うか）は，一通りには定まらない。

1.4. 想定する実装構造

Figure 3に，想定する実装構造を示す。この図は，あくまでも，想定する実装構造の概要を示したものであり，実際の実装構造はより複雑なものとなる。

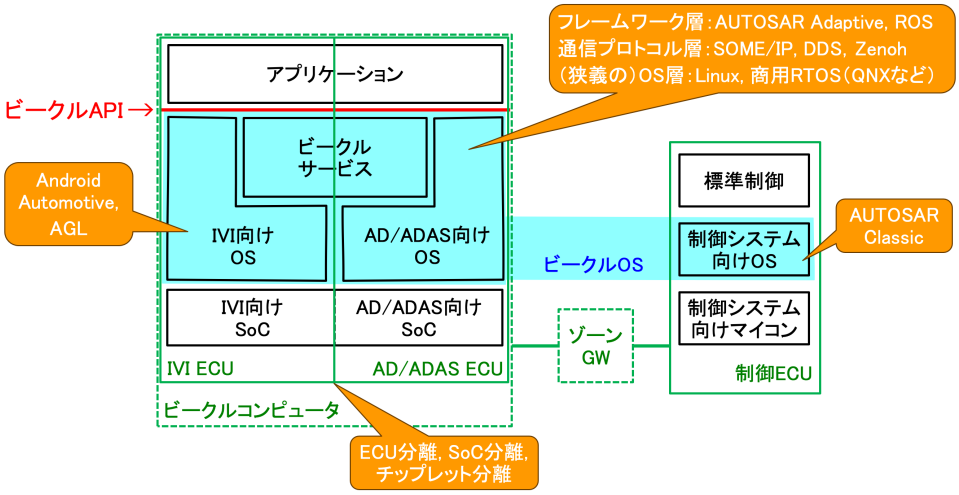


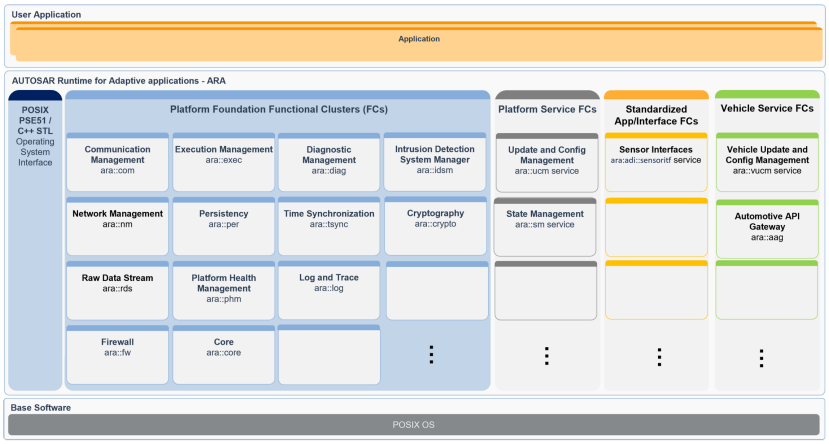
Figure 3. 想定する実装構造

機能安全対応が必須のAD/ADAS向けSoC/OSと，機能安全対応が容易でないIVI向けのSoC/OSは，当面は1つにならないと想定している（IVIの一部の機能には機能安全要求があるため，実際には，IVI向けにも複数のOSが使用される）。IVI向けのSoCとAD/ADAS向けのSoCが，チップレットの形で1つのパッケージに入ることは考えられる。

アプリケーションは，IVI ECUまたはAD/ADAS ECUまたはそれらを統合したビークルコンピュータ上で実行することを想定する。さらに別の方法として，アプリケーション実行専用のECUを用意することも想定する。また，アプリケーションによって異なるECUで実行することや，1つのアプリケーションを複数のECUに分散して実行することも想定する。

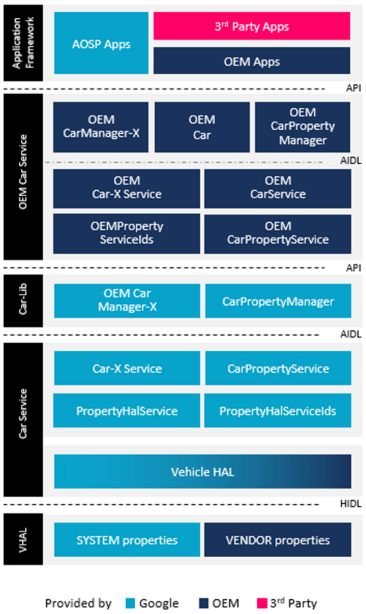
あるECU上で実現されたビークルサービスを，他のECUで実行されているアプリケーションから呼び出すことは可能である。ただし，同じECUで実行されているアプリケーションから呼び出した方が効率は良い。

ビークルサービスの実装構造は，用いる（コンピュータの）OSにより大きく異なる。例えば，AUTOSAR AP上でビークルサービスを実現する場合には，Figure 4のUser ApplicationとARAの間にService層を設け， Service層でビークルサービスを実現することになる。また，Android Automotive上でビークルサービスを実現する場合には，Figure 5の“OEM Car Service”および“Vehicle HAL”で，ビークルサービスを実現することになる。



※ “Explanation of Adaptive Platform Design (R24-11)”より

Figure 4. AUTOSAR APの構成



※ “Android Automotive OS Whitepaper”より

Figure 5. Android Automotiveの構成

Chapter 2. 主要な概念

2.1. APIに関する用語とデータ型

2.1.1. 用語の定義

(1) オブジェクト

ビークルOSが操作対象とするリソース（車両のデバイスや外部リソース）を，ビークルオブジェクト，または単にオブジェクトと呼ぶ。

各オブジェクトに対して，システム内に実体が1つのみ存在する（シングルトン）か，複数存在しうる（マルチインスタンス）かを定める。マルチインスタンスの場合，必要に応じて，オブジェクトの種類をオブジェクトクラス（または単にクラス），個々の実体をオブジェクトインスタンス（または単にインスタンス）と呼んで区別する。

(2) 構成と状態

オブジェクトの属性の中で，静的に決まるものを構成（configuration）または構成情報，動的に変化するものを状態（status）と呼ぶ。

両者の中間的な属性もあるため，構成が変わる場合もあるものとする。

(3) IDとハンドル

ビークルオブジェクトなどの静的なリソースは，その種類毎（オブジェクトの場合はオブジェクトクラス毎）に，ID（オブジェクトの場合はインスタンスID）と呼ばれる番号で識別する。IDは，1から連続する正の整数値とし，IDを表すデータ型を，IdTypeとする。

動的なリソースは，ハンドルと呼ばれる値で識別する。リソースを生成するサービスコールが，ハンドルの値を決定し，アプリケーションに知らせる。ハンドルのデータ型は，リソース毎に定める。

(4) 標準制御

OEMが自動車の出荷時に組み込んでいる制御ロジックを，標準制御と呼ぶ。標準制御は，アプリケーション，ビークルOS，ハードウェアのいずれで実現されている場合もある。

標準制御の動作を止めることができるのが望ましいが，標準制御がハードウェアで実現されている場合など，止められないものがあっても良い。

2.1.2. データ型

本仕様では，以下のデータに対して，データ型を定める。

- 車両の構成情報
- サービスコールのパラメータとリターンパラメータ
- イベント

様々なプログラミング環境（プログラミング言語やOS）に適用できるように，データ型は論理レベルで規定する。実装におけるデータ型との対応については，プログラミング環境向けの物理APIで定める。

データ型の名称は，先頭が大文字のキャメルケース（PascalCase）とし，末尾にTypeを付けるものとする。ただし，基本データ型と文字列型の名称は，この原則の例外とする。

2.1.2.1. 論理レベルのデータ型の定義

論理レベルのデータ型の定義では，データ型が取りうる値とその意味を規定する。数値を表すデータ型については，単位まで規定する。

ただし，アプリケーションIDのデータ型（ApplicationIdType）のように，プログラミング環境に依存するデータ型については，物理APIに規定を委ねる場合もある。

論理レベルのデータ型として，以下のデータ型を用意する。

- 基本データ型（ブール型，符号付き/なし整数型，浮動小数点数型）
- 文字列型
- 固定長配列，可変長配列
- 列挙型
- 列挙型の集合型
- 構造体
- バリアント（タグ付き共用体）
- オプショナル型

一方，文字型，ポインタ型，参照型は用意しない。文字型を用意しないのは，本仕様では必要ないためである。

データ型が取りうる値を記述するために，形式的な記述法を導入する。既存のもので目的に合致するものがあれば使いたいが，合致するものが見つからないため，独自に定義する。

2.1.2.2. データ型の定義と記述法

ここでは，データ型が取りうる値と，それをプログラミング言語のようなスタイルで定義するための記述法とYAMLによる記述法を規定する。

(1) 基本データ型

基本データ型は，ベースとなるデータ型と，オプションの範囲指定で定義/記述する。範囲指定は，数値を表すデータ型（具体的には，ブール型以外のデータ型）に対して，取りうる値の範囲（特殊値を含む）を限定するものである。

ベースとなるデータ型の記述法と，取りうる値は次の通り。

ブール型	Bool	真（true）と偽（false）の2つの値のみを取るデータ型
8/16/32/64ビット 符号付き整数型	Int8/Int16/ Int32/Int64	8/16/32/64ビットの長さの符号付き整数型
8/16/32/64ビット 符号なし整数型	Uint8/Uint16/ Uint32/Uint64	8/16/32/64ビットの長さの符号なし整数型
32/64ビット 浮動小数点数型	Float32/Float64	IEEE 754準拠の32/64ビットの浮動小数点数型

範囲指定は，数値を表すデータ型（ブール型以外の基本データ型）に対して，取りうる値の範囲（特殊値を含む）を限定するものである。取りうる値の範囲は，以下の区間指定の任意個の和集合の形で指定する。ここで，複数の区間指定に含まれる数値があってはならない。

最小値*a*と最大値*b*で指定	[*a*,*b*]
排他的下限*a*と最大値*b*で指定	(*a*,*b*]
最小値*a*と排他的上限*b*で指定	[*a*,*b*)
排他的下限*a*と排他的上限*b*で指定	(*a*,*b*)
単一の数値*a*で指定	*a*
特殊値の定数名*NAME*と数値*value*で指定	*NAME*(*value*)

特殊値の定数名は，1つの数値型の中でユニークであれば良い。特殊値の定数名が数値型を超えてユニークでなければならないプログラミング環境向けの物理APIでは，特殊値の定数名の前に，名前の衝突を避けるための文字列を付加する。

範囲指定は，ベースとなるデータ型の名称の後に，()内に，上記の区間指定の記述（[*a*,*b*]，(*a*,*b*]，[*a*,*b*)，(*a*,*b*)，*a*，*NAME*(*value*)のいずれか）を，|で区切って列挙することで記述する。

範囲指定を持つ基本データ型の例は次の通り。

例）8ビットの長さの符号付き整数により，0以上100以下の値と，UNKNOWN（値は-1），NONZERO（値は-2）のいずれかの値をとるデータ型。

```
Int8 ([0,100] | UNKNOWN(-1) | NONZERO(-2))
```

例）IEEE 754準拠の32ビットの浮動小数点数により，0.0以上で無限大（+INF）より小さい値と，NaNのいずれかの値をとるデータ型。

```
Float32 ([0.0,+INF) | NaN)
```

YAMLでは，範囲指定のない基本データ型は，その名称をYAMLのスカラーで記述する。範囲指定を持つ基本データ型は，ベースとなる基本データ型の名称を*BaseType*とすると，次のように記述する。

```
range:
  baseType: BaseType
  intervals:
    - minimum: a
      maximum: b
    - exclusiveMinimum: a
      maximum: b
    - minimum: a
      exclusiveMaximum: b
    - exclusiveMinimum: a
      exclusiveMaximum: b
    - singleValue: a
    - specialName: NAME
```

```
      specialValue: value
```

上記の範囲指定を持つ基本データ型の例は，YAMLでは次のようになる。

```
range:
  baseType: Int8
  intervals:
    - minimum: 0
      maximum: 100
    - specialName: UNKNOWN
      specialValue: -1
    - specialName: NONZERO
      specialValue: -2
```

```
range:
  baseType: Float32
  intervals:
    - minimum: 0.0
      exclusiveMaximum: +INF
    - singleValue: NaN
```

なお，以上で規定したデータ型の記述法は，取りうる値を記述するものであり，実装時のデータ型を決めるものではない。そのため，例えばInt8型のデータを，物理APIの規定で，32ビットの符号付き整数型で実現することとしても良い。

(2) 文字列型

Unicodeの文字列を取るデータ型。Stringと記述する。

YAMLでは，YAMLのスカラーでStringと記述する。

(3) 固定長配列

指定された要素型のデータの指定された個数の配列。要素型が*ElementType*，要素数が*size*である時，*ElementType*[*size*]と記述する。

YAMLでは，次のように記述する。

```
array:
  elementType: ElementType
  size: size
```

(4) 可変長配列

指定された要素型のデータの任意個数の配列。要素数が0の場合もある。可変長配列型のデータからは，配列の要素数を得ることができるものとする。

要素型が*ElementType*である時，*ElementType*[]と記述する。

YAMLでは，次のように記述する。

```
array:
  elementType: ElementType
```

【補足説明】

物理APIにおいて，配列を置いた領域へのポインタと要素数，ベクタ（例えば，C++のstd::vector），リストなどで実現することを想定している。

(5) 列挙型

あらかじめ定義された列挙子のいずれかを取るデータ型。列挙子は，1つの列挙型の中でユニークであれば良い。

列挙子が$Enumerator_1$，$Enumerator_2$，…，$Enumerator_n$である時，次のように記述する。

```
enum {
    Enumerator₁,
    Enumerator₂,
    …
    Enumeratorₙ
}
```

YAMLでは，次のように記述する。

```
enum:
  - enumerator: Enumerator₁
  - enumerator: Enumerator₂
  …
  - enumerator: Enumeratorₙ
```

列挙子が列挙型を超えてユニークでなければならないプログラミング環境向けの物理APIでは，列挙子の名前の前に，名前の衝突を避けるための文字列を付加する。

(6) 列挙型の集合型

指定された列挙型の列挙子の集合を取るデータ型。列挙型が*EnumType*である時，enumSet(*EnumType*)と記述する。

YAMLでは，次のように記述する。

```
enumSet: EnumType
```

【補足説明】

物理APIにおいて，列挙型を整数型で実現する場合には，列挙型の集合型は，各列挙子が1つのビットに対応する符号なし整数型（ビットマップ）で実現することを想定している。

列挙型以外のデータ型の集合型も考えられるが，可変長配列で扱えば十分と考え，用意しない。

(7) 構造体

フィールド名とデータ型が指定されたフィールドを，複数まとめて扱うデータ型。フィールド毎にデータ型が異なっていても良い。

フィールド名が$fieldName_1$，$fieldName_2$，…，$fieldName_n$，それぞれのデータ型が$FieldType_1$，$FieldType_2$，…，$FieldType_n$である時，次のように記述する。

```
struct {
    fieldName₁ : FieldType₁,
    fieldName₂ : FieldType₂,
    …
    fieldNameₙ : FieldTypeₙ
}
```

YAMLでは，次のように記述する。

```
struct:
  - fieldName: fieldName₁
    fieldType: FieldType₁
  - fieldName: fieldName₂
    fieldType: FieldType₂
  …
  - fieldName: fieldNameₙ
    fieldType: FieldTypeₙ
```

(8) バリアント（タグ付き共用体）

列挙型をパラメータに取り，データの種別を表す列挙子と，その列挙子に対応するデータ（ペイロード）をまとめて扱うデータ型。各列挙子に対するペイロードは，データ型が異なっていても良い。また，ペイロードを持たない列挙子があっても良い。

列挙型が$EnumType$，列挙子が$Enumerator_1$，$Enumerator_2$，…，$Enumerator_n$，それぞれに対応するペイロードのデータ型が$PayloadType_1$，$PayloadType_2$，…，$PayloadType_n$である時，次のように記述する。

```
variant(EnumType) {
    Enumerator₁ : PayloadType₁,
    Enumerator₂ : PayloadType₂,
    …
    Enumeratorₙ : PayloadTypeₙ
}
```

ペイロードを持たない列挙子については，列挙子とそれに対応するペイロードのデータ型を記述しない。

ペイロードのデータ型は，構造体とすることを原則とし，バリアントの記述中に構造体の定義を直接記述する場合には，`: struct`は省略できるものとする。具体的には，$Enumerator_i$に対応するペイロードの

データ型が，フィールド名が$fieldName_{i,1}$，$fieldName_{i,2}$，…，$fieldName_{i,mi}$，それぞれのデータ型が$FieldType_{i,1}$，$FieldType_{i,2}$，…，$FieldType_{i,mi}$である構造体である時，次のように記述できる。

```
variant(EnumType) {
    …
    Enumeratorᵢ {
        fieldNameᵢ,₁ : FieldTypeᵢ,₁,
        fieldNameᵢ,₂ : FieldTypeᵢ,₂,
        …
        fieldNameᵢ,ₘᵢ : FieldTypeᵢ,ₘᵢ
    },
    …
}
```

YAMLでは，上記の2つの記述を，それぞれ次のように記述する。

```
variant:
  parameterType: EnumType
  payloads:
    - enumerator: Enumerator₁
      payloadType: PayloadType₁
    - enumerator: Enumerator₂
      payloadType: PayloadType₂
    …
    - enumerator: Enumeratorₙ
      payloadType: PayloadTypeₙ
```

```
variant:
  parameterType: EnumType
  payloads:
    …
    - enumerator: Enumeratorᵢ
      payloadType:
        - fieldName: fieldNameᵢ,₁
          fieldType: FieldTypeᵢ,₁
        - fieldName: fieldNameᵢ,₂
          fieldType: FieldTypeᵢ,₂
        …
        - fieldName: fieldNameᵢ,ₘᵢ
          fieldType: FieldTypeᵢ,ₘᵢ
    …
```

ペイロードを持たない列挙子については，それに対応するペイロードのデータ型（payloadType）の記述を省略する。または，列挙子（enumerator）とそれに対応するペイロードのデータ型（payloadType）の両方の記述を省略しても良い。

【補足説明】

物理APIにおいて，タグを表す列挙型と（タグのない）共用体，バリアント，関連値付き列挙型などで実現することを想定している。

(9) オプショナル型

基礎となるデータ型の値と，値が存在しないことを示す値（null）のいずれかを取るデータ型。オプショナル型のデータからは，値が存在するか否かを判定でき，値が存在する場合にはその値を取得することができる。

基礎となるデータ型が*UnderlyingType*である時，*UnderlyingType*?と記述する。

YAMLでは，次のように記述する。

```
optional: UnderlyingType
```

(10) データ型への名前付け

*TypeDescription*で記述されたデータ型に，*TypeName*という名前を付ける時は，次のように記述する。

```
type TypeName = TypeDescription;
```

YAMLでは，次のように記述する。

```
typeDefinition:
  name: TypeName
  type: TypeDescription
```

2.1.2.3. YAMLでのデータの記述法

ここでは，車両の構成記述ファイル等で，YAMLによりデータを記述する方法を規定する。データ型をYAMLで記述する方法ではないことに注意すること。

(1) 基本データ型，文字列型

YAMLのスカラーの文法に従って記述する。

(2) 固定長配列，可変長配列

配列の要素を，YAMLのシーケンスで記述する。

(3) 列挙型

列挙子を文字列で記述する。

(4) 列挙型の集合型

集合に含まれる列挙子の文字列を，YAMLのシーケンスで記述する。

(5) 構造体

フィールド名の文字列をキー，フィールド値の記述を値とするYAMLのマッピングで記述する。

(6) バリアント（タグ付き共用体）

列挙子の文字列をキー，ペイロードの記述を値とする1つのエントリを持つYAMLのマッピングで記述する。ペイロードを持たない列挙子の場合は，値をnullとするか，YAMLのマッピングに代えて，列挙子の文字列をYAMLのスカラーとして記述する。

(7) オプショナル型

値が存在しない場合は，nullと記述する。ただし，構造体のフィールドである場合に限り，そのフィールド（フィールド名とフィールド値）の記述を省略することで，値が存在しないことを表現しても良い。

2.2. 車両の構成記述

車両の構成記述は，車両の構造・装備・機能・性能など，車両の静的な属性を表現するものである。

ある車両のウィンドウの構成情報をYAMLで記述した例を，Figure 6に示す。

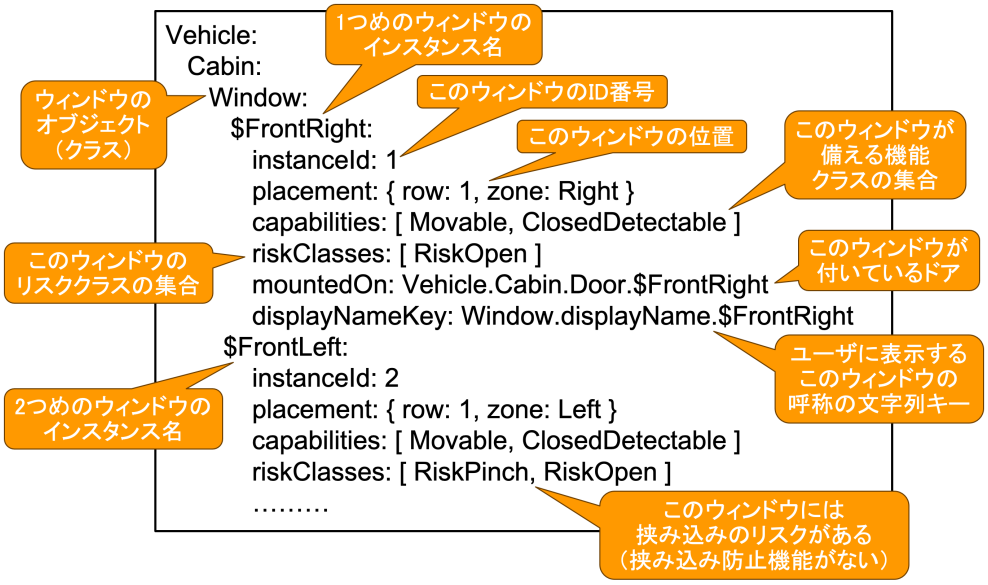


Figure 6. 車両構成記述の例

2.2.1. 構成記述に対する要求

車両の構成記述法を規定するにあたり，以下の要求を設定した。

(1) 車両の構造・機能を，トップダウンに整理する

例えば，洗車モードに遷移させるアプリケーションを作成する場合，水が入る可能性がある開口部をすべて数え挙げられることが必要である。そのためには，キャビンの開口部に何があるか（例えば，ドア，ウィンドウ，ガラス）を整理し，各開口部の性質（例えば，ウィンドウは閉めれば水を通さない）を規定する必要がある。

(2) 車両依存の拡張を考慮した記述方法とする

車両の多様性を考えると，車両の構成記述法を網羅的に規定するのは難しく，車両依存で拡張できる必要がある。

2.2.2. 車両構成の階層構造

構成記述では，車両の構成を階層構造で表現する。操作の対象となるオブジェクトは，階層構造の中間レベルのノードに対応し，その下のノードに，オブジェクトの構成情報を記述する。

オブジェクトや構成情報の正式な名称は，トップレベルのノード名から開始して，各レベルのノード名を.でつないだ名称とする。

2.2.2.1. 階層構造の設計方針

車両構成の階層構造は，COVESA　VSS[1]をベースに，DVE+Uモデル[2]も参考に定める。ただし，以下の設計方針を設定したため，VSSとはかなり違ったものとなっている。

- 多様な車両を記述可能なものとする
- 階層構造のレベルをあまり深くならないようにする
- トップレベルの直下に構成情報を置かない
- マルチインスタンスの扱いを明確にする
- アプリケーションのデータモデルは含めない

[1] https://covesa.github.io/vehicle_signal_specification/

[2] M. Gondo: "(The) Four Principles of SDV", 15th AUTOSAR OPEN CONFERENCE, 2025年6月.

2.2.2.2. トップレベルのノード

車両構成の階層構造のトップレベルのノードは，VSSと同様に，Vehicleのみとする。

【仕様決定の理由】

DVE+Uモデルをベースに考えると，トップレベルのノードをVehicle，Driver，Environmentの3つとすることも考えられるが，Driverは車両の運転者であること，Environmentも車両の周辺環境であることを考えて，COVESA VSSと同様に，トップレベルのノードはVehicleのみとした。

2.2.2.3. 第2レベルのノード

車両構成の階層構造の第2レベルのノードは，現時点では以下のようにする。

VersionOSDVI	Open SDV API仕様のバージョン情報
VehicleIdentification	車両の識別情報
Specification	車両諸元
Cabin	乗員が乗る空間
CargoSpace	荷物を乗せる空間
Body	主に車両の外側にある装備
Infotainment	インフォテイメント
HMI	HMI
Motion	車両の運動制御

CurrentLocation	車両の現在位置
Driver	運転者
Occupant	運転者以外の乗員
SurroundModel	車両の周辺モデル
Atmosphere	車両の環境情報
ADADAS	運転支援・自動運転機能
Energy	駆動用バッテリーなど車両のエネルギー情報
Navigation	ナビゲーション機能

【補足説明】

運転者は，遠隔運転を考慮に入れて，車内にいるとは限らないものとする。

【COVESA VSSとの関係と仕様決定の理由】

VehicleIdentification，Cabin，Body，CurrentLocation，Driver，Occupantは，VSSの第2レベルのノードと同様である。InfotainmentはVSSでは第3レベル（Cabinの下），HMIはVSSでは第4レベル（Cabin.Infotainmentの下）に置かれているが，階層構造のレベルをあまり深くならないようにするという方針から，第2レベルに移動した。Atmosphereは，VSSではExteriorという名称になっている。

VersionOSDVIは，VersionVSSに対応するものとして設けた。

その他は，VSSにはない第2レベルのノードであるが，以下の理由で新設した。

Motionは，VSSでVehicle直下に置かれている車両の移動に関する属性を置くために，Specificationは，Vehicle直下に置かれているその他の車両諸元を置くために設けた。

CargoSpaceは，物流用の車両を扱うために新設した。

SurroundModelは，車両の周辺モデルの表現方法をISO　23150をベースに定めることとしたことから，ISO 23150の用語を導入した。

ADADASは，VSSのADASに相当する第2レベルのノードであるが，VSSのADASの内容が大きく異なるため，名称を変更した。

Energyは，駆動用バッテリーなど車両のエネルギー情報を扱うために新設した。

Navigationは，ナビゲーション機能を扱うために新設した。

【未決定事項】

VSSにあるその他の第2レベルのノード（Powertrain，Chassis，Service，Connectivity，Diagnostics，MotionManagement，ControlUnitなど）については十分に検討できておらず，さらに検討が必要である。

第3レベル以下のノードについては，今後の課題である。

2.2.2.4. インスタンスを表すノードとインスタンスID

マルチインスタンスのオブジェクトに対しては，クラスを表すノードの下に，インスタンスを表すノードを置く。

インスタンスを表すノード名は，インスタンス名の前に$を付けた文字列とする。インスタンス名は，車両毎に自由に決めることができる。

インスタンスの構成情報には，サービスコールなどでインスタンスを数値で識別する場合に用いるインスタンスID（instanceId）を含める。

2.2.2.5. オブジェクトに付属するオブジェクト

オブジェクトに付属するオブジェクトは，前者のオブジェクトのインスタンスの下に置くのではなく，独立したオブジェクトと扱い，付属関係は，構成情報により表現する。

例えば，ウィンドウやガラスに付属しているシェード（日除け）は，各種のシェードをすべてVehicle.Cabin.Shadeのインスタンスとし，構成情報を使ってウィンドウやガラスと紐付ける。これにより，どのシェードも同じサービスコールで操作することができる。

【COVESA VSSとの関係】

VSSでは，リアガラスのシェードはVehicle.Cabin.RearShade，ウィンドウのシェードはVehicle.Cabin.Door.Row1.DriverSide.Shadeなど，サンルーフのシェードはVehicle.Cabin.Sunroof.Shadeとなっている。

2.2.2.6. OEM固有のオブジェクト/属性

OEMは，固有のオブジェクトや属性（構成情報や状態）を追加することができる。

OEM固有のオブジェクト/属性の名称は，先頭に，OEM名と$を付けた文字列とする。

2.2.3. 車両の構成情報の取得方法

車両の構成情報を取得する方法には，構成記述ファイルを参照する方法と，構成情報を取得するためのサービスコールによる方法がある。

2.2.3.1. 構成記述ファイル

車両の構成記述を含むファイルを，車両の構成記述ファイルと呼ぶ。

構成記述ファイルには，車両の構成情報をYAML形式で記述する。具体的には，車両構成の階層構造の各レベルのノードを，ノード名をキー，ノードの構成情報を値とするYAMLのマッピングで記述する（Figure 6）。

【補足説明】

構成記述ファイルのユースケースは以下の通り。

- ビークルOSは，構成記述ファイルを読み込む。ただし，ビークルOSが単一の車両のみに対応するのであれば，ビークルOS自身が車両の構成記述を参照することは必須ではない。
- アプリケーション開発者は，アプリケーションを対応させたい車両群の構成記述ファイルを参考にすることができる。
- シミュレータや各種ツールが，構成記述ファイルを利用することも考えられる。

- アプリケーションは，車両の構成情報の取得するために，構成記述ファイルを読んでも良い。ただし，一般には，getConfig/getConfigAllサービスコールにより取得した方が効率的である。

2.2.3.2. 構成情報を取得するためのサービスコール

シングルトンオブジェクトは，getConfigサービスコールにより，その全構成情報を参照できる。マルチインスタンスオブジェクトは，getConfigAllサービスコールにより，そのすべてのインスタンスの全構成情報を参照できる。

マルチインスタンスオブジェクトに対するgetConfigAllは，インスタンスの全構成情報を含む構造体の可変長配列を返す（配列の要素数は，インスタンスの数となる）。getConfigAllでインスタンス名を取得するために，インスタンスの構成情報の構造体に，インスタンス名を格納するためのString型のフィールド（instanceName）を追加する。ただし，連想配列が扱えるプログラミング環境では，可変長配列を連想配列で実現することで，インスタンス名を連想配列のキーの形で表すことができる。

関連するオブジェクトを表すためのRelatedObjectTypeのデータ型は，プログラミング環境によって異なる場合がある。getConfig/getConfigAllは，構成情報中のRelatedObjectTypeのデータを，そのプログラミング環境向けの物理APIで規定された形に変換して返す。

2.2.4. 車両の構成情報の多言語対応

オブジェクトの構成情報の中で，ユーザに表示することを目的としたオブジェクトの呼称などは，言語によって異なる文字列となる。そこで，以下の方法で多言語対応する。

構成記述ファイル中には，表示用の文字列ではなく，多言語対応する文字列のキーを記述する。言語毎の実際の文字列は，ロケール毎に用意する別のファイル（文字列定義ファイル）に記述する。文字列定義ファイルの例を下に示す。

```
Window:
  displayName:
    $FrontRight: "運転席の窓"
    $FrontLeft: "助手席の窓"
    $RearRight: "後席右側の窓"
    $RearLeft: "後席左側の窓"
```

getConfig/getConfigAllは，構成記述ファイル中の文字列のキーを，呼び出し環境のロケールに対応する文字列に置き換えて返す。

2.2.5. 構成情報の変化

構成情報はオブジェクトの静的な属性であるが，変化することも想定している。ただし，構成情報が頻繁に変化することは想定しておらず，構成情報が変化した場合には，アプリケーションの再起動が必要になっても良いものとする。

オブジェクトの構成情報が変化した場合には，そのオブジェクトに対して，構成情報が変化したことを伝えるイベント（ConfigChanged）を生成する。トップレベルや中間レベルのノードに対応するオブジェクトに対しては，そのノードの下のオブジェクトの構成情報が変化した時にも，ConfigChangedを生成する。

マルチインスタンスオブジェクトにおいて，インスタンスが増えたことを知るためには，イベント通知対象のインスタンスIDを指定せずにイベント通知を要求する必要がある。

2.3. APIの要素と規定内容

オブジェクト毎のAPIは，Chapter 4とChapter 5で規定する。ただし，複数のオブジェクトに共通の事項については，共通規定としてChapter 3で規定する。

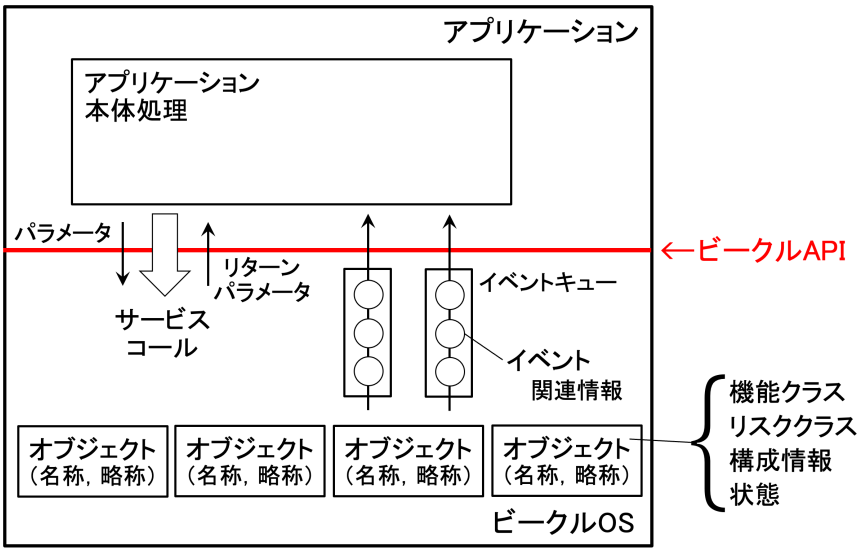


Figure 7. APIの要素

オブジェクト毎のAPI規定では，Figure 7に示すAPIの各要素に関して，以下の内容を規定する。

- 名称と略称
- 位置付けと機能
- 機能クラス
- リスククラス
- 構成情報
- 状態
- サービスコール一覧
- イベント
- データ型の定義
- サービスコールの詳細規定

2.3.1. 名称と略称

オブジェクトクラスの正式な名称は，車両の構成記述における階層構造を示す名称（フルパス名）とする。例えば，ウィンドウのクラスの正式な名称は，Vehicle.Cabin.Windowである。

各オブジェクトクラスに対して，ユニークな略称を与える。例えば，ウィンドウのクラスの略称は，Windowである。略称は，データ型名の一部やObjectKindTypeの列挙子などに使用する。

2.3.2. 位置付けと機能

オブジェクトの位置付けとして，車両内での位置付けや他のオブジェクトとの関係，オブジェクトの性質を説明する。また，オブジェクトが，シングルトンかマルチインスタンスかを規定する。

オブジェクトの機能として，オブジェクトが持つ機能と動作について説明する。オブジェクトがMovableObjectの機能を持つ場合には，そのことを明記する。また，オブジェクトに対する操作の調停方法について説明する。標準的な調停機能が適用される場合には，そのことを明記する。オブジェクトがLockableObjectである場合には，そのことを明記する。

2.3.3. 機能クラス

オブジェクトの備える機能を分類したものを機能クラスと呼ぶ。車両の構成記述で，各オブジェクト（の各インスタンス）に対して，備えている機能クラスの集合を記述する。

オブジェクトが特定の機能安全要求を満たすことは，リスククラスではなく，機能クラスで表現する。例えば，車両のバック時に後方を見るためのリアビューアプリケーションには，画像がフリーズしてはならないという機能安全要求がある。これを満たすためには，フリーズしないビュー（ディスプレイの全体またはその一部分）を用いる必要がある。これを実現するために，ビューがフリーズする可能性がない（または，フリーズを防止する機能を持っている）ことを機能クラスで記述し，リアビューアプリケーションはその機能クラスを持ったビューを選択して使用する設計とする。

機能クラスのデータ型は，オブジェクト毎に，*xxx*CapabilityType（*xxx*はオブジェクト略称）の名称で，列挙型として定義する。

オブジェクト毎のAPIの規定では，オブジェクトが持つ各機能クラスの名称と，どのような機能であるかについて規定する。

【補足説明】

上記の例において，リアビューアプリケーションは，車両が走行中にセンターディスプレイを使用するため，RiskDistractionCid（センターディスプレイに描画することで生じるドライバディストラクションのリスク）に対応できていることが，OEMによる審査で認められる必要がある。リアビューアプリケーションが，フリーズする可能性がないビューを使用する設計になっていることは，このリスク対応の一部として審査対象となる。

2.3.4. リスククラス

オブジェクトを操作することで生じるリスクを分類したものをリスククラスと呼ぶ。リスククラスの詳細については，Section 2.5.2.1を参照のこと。

リスククラスのデータ型は，オブジェクト毎に，*xxx*RiskType（*xxx*はオブジェクト略称）の名称で，列挙型として定義する。

操作することによって何らかの危険を生じる可能性があるオブジェクトのAPIの規定では，オブジェクトが持つ各リスククラスの名称と，どのようなリスクであるかについて規定する。また，危険を生じる可能性がある操作の種類（サービスコールとそのパラメータ）と車両状態について，サービスコールの機能の項で規定する。

2.3.5. 構成情報

オブジェクトの機能・性能など，オブジェクトの属性の中で静的に決まるものを，構成情報と呼ぶ。マルチインスタンスオブジェクトでは，インスタンス毎に構成情報を持つ。

オブジェクト毎のAPIの規定では，オブジェクト毎に，構成情報の名称，データ型，意味などを規定する。

2.3.6. 状態

オブジェクトの属性の中で動的に変化するものを，状態と呼ぶ。マルチインスタンスオブジェクトでは，インスタンス毎に状態を持つ。

オブジェクト毎のAPIの規定では，オブジェクト毎に，状態の名称，データ型，意味などを規定する。

2.3.7. サービスコール

アプリケーションが，ビークルOSのサービスを呼び出すインタフェースを，サービスコールと呼ぶ。

オブジェクト毎のAPIの規定では，オブジェクト毎に，サービスコールの名称と機能，パラメータとリターンパラメータの名称，データ型，意味などを規定する。

2.3.7.1. パラメータとリターンパラメータ

サービスコールは，パラメータとリターンパラメータを持つ。マルチインスタンスオブジェクトに対するサービスコールにおいては，原則として，第1パラメータを，インスタンスを識別するためのインスタンスIDとする。

サービスコールは，正常に実行された場合にはE_OK，エラーになった場合にはエラーの種類を示すエラーコードを返す。これを，サービスコールの戻り値と呼び，リターンパラメータの1つとする。戻り値のデータ型（ReturnType）は，すべてのサービスコールで共通とする。そのため，特定のサービスコールに特化したエラーコードは設けず，自分がロックしていないオブジェクトに対してunlockを呼んだ場合のエラーをE_OBJECT_STATUSとするなど，一般的なエラーコードを用いる。

サービスコールが複数のエラーを検出した場合には，その内のどのエラーコードを返しても良い。物理APIで，エラーコードを追加する場合がある。

別途の規定がない限り，サービスコールがエラーになった場合には，操作対象のオブジェクトの状態は変化しないものとする。

【補足説明】

オブジェクト指向言語など向けの物理APIでは，インスタンスIDを表す第1パラメータは，メソッドのパラメータとはせず，メソッドを受け取るオブジェクトで表しても良い。

2.3.7.2. 並行して呼び出されたサービスコール

並行して呼び出されたサービスコール（あるサービスコールからリターンする前に呼び出されたサービスコール）は，別途の規定がない限り，サービスコールが呼び出されてからリターンするまでの間のある瞬間にアトミックに実行された場合と同等の振る舞いをする。

どの瞬間に実行されるかは規定されないため，先に呼び出されたサービスコールが先に実行されるとは限らない。言い換えると，並行して呼び出されたサービスコール間の実行順序は任意となる。

2.3.7.3. サービスコール実装上の前提

ビークルサービスは以下の情報を取得できるものとする。

- サービスコールを呼び出したアプリケーションID。制御の調停などに用いる。
- サービスコールを呼び出したアプリケーションのマニフェスト。リスク制御と制御の調停に用いる。マニフェスト中のこれらに用いる情報については，Section 2.5.2.3を参照のこと。
- サービスコールを呼び出したアプリケーションのロケール（言語および地域設定）。

2.3.8. イベント

ビークルOSが管理するオブジェクトの状態変化や要求された操作の中止などを，アプリケーションに伝え

るためのデータを，イベントと呼ぶ。イベントは，イベント種別を持つことに加えて，状態変化の原因や状態変化前後の状態などを伝えるために，イベント種別毎の関連情報を持つことができる。

イベントのデータ型は，イベントが発生したインスタンスのID（マルチインスタンスオブジェクトの場合のみ）と，イベント情報をフィールドに持つ構造体とする。イベント情報のデータ型は，イベント種別を表す列挙型をパラメータとし，イベント種別毎の関連情報を表す構造体をペイロードとするバリアントとする。

オブジェクト毎のAPIの規定では，オブジェクト毎に，イベント種別の名称と意味，イベント種別毎の関連情報の名称，データ型，意味，イベントのデータ型などを規定する。

2.3.8.1. イベントの発生条件と発生順序

イベントは，別途の規定がない限り，オブジェクトの状態が変化しない場合には発生するとは限らない。

サービスコールの実行によって生成されるイベントは，サービスコールの実行順序でイベントキューに入れられる。1つのサービスコールで複数のイベントが生成される時には，それらの間に因果関係がある場合は原因となるイベントが先に，因果関係がない場合には任意の順序で，イベントキューに入れられる。

2.3.8.2. イベントキュー

ビークルOSからのイベントはイベントキューに格納され，アプリケーションは，イベントキューからイベントを受け取る。イベントキューは，複数のイベントを入れることができ，FIFO順でイベントを管理する。

アプリケーションは，複数のイベントキューを持つことができる。イベントキューは通知ハンドルで識別する。

イベントの優先度管理をしたい場合には，複数のイベントキューを用いて，アプリケーション側で実現する。

2.3.8.3. 最新値イベント

イベント種別によっては，途中の状態変化をすべて通知する必要はなく，最新の状態変化のみを通知すればよい場合がある。このような場合には，オブジェクト毎のAPIの規定において，同じオブジェクトインスタンスの同じイベント種別について，最新のイベントのみをイベントキューに保持する仕様としても良いものとし，このようなイベント種別を，最新値イベントと呼ぶ。

最新値イベントをイベントキューに格納する際に，同じオブジェクトインスタンスの同じイベント種別のイベントが，そのイベントキューに格納されている場合には，古いイベントを削除し，新しいイベントをイベントキューの末尾に格納する。これにより，当該オブジェクトインスタンスの当該イベント種別について，イベントキューに保持されるイベントは最大1つとなる。

この処理によるイベントの削除は，イベントキューのオーバフローとして扱わず，EventOverflowの生成や，その関連情報である失われたイベント数の増加は行わない。

2.3.8.4. イベント操作のサービスコール

イベントを操作するためのサービスコールとして，オブジェクト毎に，イベント通知を要求するサービスコール（notify），イベント通知を停止するサービスコール（unnotify），イベントを取り出すサービスコール（getEvent）を用意する。

notifyサービスコールが正常に実行されると，イベントキューが生成され，イベントキューを識別するための通知ハンドルが返される。受け取るイベントの種類を限定する場合には，notifyサービスコールのパラメータで指定する。受け取るイベントの種類を指定する方法が複数ある場合には，オブジェクトに対して複数種類のnotifyサービスコールを用意する。

unnotifyサービスコールは，通知ハンドルを指定してイベント通知を停止する。正常に実行されると，通知ハンドルに対応するイベントキューは削除される。

getEventサービスコールは，通知ハンドルで指定したイベントキューからイベントを取り出す。イベントがパラメータ等で渡されてくるプログラミング環境向けの物理APIでは，getEventサービスコールを用意しなくても良い。

2.3.8.5. イベントキューのオーバフロー

イベントキューがオーバフローした場合，イベントキュー中の古いイベントから順に削除する。

イベントを削除した時には，イベントキューがオーバフローしたことを示すイベント（EventOverflow）を生成し，イベントキューの先頭に入れる（イベントキューはFIFO順でイベントを管理するという規定の例外）。

EventOverflowは，関連情報として，失われたイベント数を持つ。イベントキューがオーバフローした時にイベントキューの先頭にEventOverflowがある場合には，失われたイベント数を増加させ，EventOverflowをもう1つ生成することはしない。

2.3.8.6. イベントキューの一本化

プログラミング環境によっては，物理APIの規定により，すべてのイベントを，1つのイベントキューで管理することとしても良い。

この場合，イベントのデータ型（構造体）に，オブジェクトの種別（クラス）と通知ハンドルのフィールドを追加する。

また，notify/unnotifyサービスコールは，イベントキューの生成/削除を行わない。getEventサービスコールは，システム全体で1種類のみとし，パラメータに通知ハンドルを含めない。

最新値イベントについては，同じ通知ハンドルに対するイベントの間でのみ，古いイベントの削除を行う。

2.4. APIの設計方針

2.4.1. サービスコールの設計方針

サービスコールは，機械動作にかかる時間と比べて，短い時間で処理が完了するような設計とする。例えば，ウィンドウを操作するサービスコールは，開閉動作が完了するまで待つのではなく，目標位置を指定して，開閉動作を開始するのみとする。言い換えると，機械動作に対しては，サービスコールは非同期（完了を待たない）とする。

一方，サービスコールの処理に複数のECUが関係する場合に，ネットワーク経由で他のECUと通信する間は，サービスコールの処理が継続しても良い。言い換えると，コンピュータおよびネットワークの動作に対して，サービスコールは同期（動作の完了まで処理が続く）とする。

ただし，サービスコールからリターンするタイミングは，プログラミング環境依存で，サービスコールの処理完了後にリターンする場合（ブロッキング）と，サービスコールの処理完了を待たずにリターンし，リターンパラメータは処理完了後に受け取るようにする場合（ノンブロッキング）がある。

サービスコールの呼び出し頻度/回数に関して，サービスコールを短い周期（具体的には，10ミリ秒未満）で呼び出す必要があるような設計としない。また，サービスコールの粒度を大きくし，呼び出し回数が少なくなるような設計とする。これは，サービスコールが，ネットワークを越えて呼び出されることが想定されるためである。

サービスコールを特定の順序で呼び出さなければならないような設計にはしない。具体的には，POSIXに

おけるファイル/デバイスのopen/closeのようなサービスコールは設けない。

2.4.2. OSに対する想定

ビークルサービスを実現するベースとなる（コンピュータの）OSは，OSが一般的に持つ機能を備えており，アプリケーションはそれらの機能を利用できるものとする。特に，以下の機能を持つものとする。

- ECUを越えたサービス呼び出し機能とPub/Sub通信機能
- 他のECU上で新しいプロセスを立ち上げる機能

2.5. 制御の調停とリスク制御

2.5.1. 制御の調停機能

ビークルOSは，複数のアプリケーションが同一のオブジェクトを操作（または，出力装置に出力）した場合に，それらを調停する機能を持つ。調停機能に必要となるロックと操作優先度の概念を，APIに導入する。

コアビークルサービスには汎用的な調停機能のみを持たせ，アプリケーション特有の調停機能が必要な場合には，拡張ビークルサービスで実現する方針とする。

ここでは，多くのオブジェクトに適用できる標準的な調停機能を規定する。標準的な調停機能では不十分なオブジェクト（例えば，音声出力）に対しては，オブジェクト固有の調停機能を持たせる。

2.5.1.1. 後勝ちの原則

後に実行されたサービスコールによる操作が，先に実行されたサービスコールによる操作をオーバライドすることを原則とする。

例えば，ウィンドウを閉めるサービスコールが実行され，ウィンドウが閉じる動作の途中で，ウィンドウを開くサービスコールが実行されると，最初のサービスコールによる操作はキャンセルされ，ウィンドウは開く動作を始める。なお，後述の操作優先度は，この振る舞いには関係しない。

【補足説明】

後勝ちの原則は，サービスコールが呼び出されたタイミングではなく，実行されたタイミングに適用される。並行して呼び出されたサービスコール間の実行順序は任意であることから，並行して呼び出されたサービスコールの中のどれが勝つかは規定されない。

サービスコールを，短い時間で処理が完了するように設計するとしたことから，この原則が自然と考えられる。

2.5.1.2. ロック機能

あるアプリケーションが制御しているオブジェクトを，他のアプリケーションのインスタンスが操作するのを禁止するために，必要なオブジェクトに対して，ロック機能を持たせる。ロック機能は，ハードウェアを用いて実現される場合（例えば，ドアロック）と，ソフトウェアのみで実現される場合がある。

ロック機能を持つオブジェクトは，ロックを取得するサービスコールとロックを解除するサービスコールを持つ。

【補足説明】

ロック機能のユースケースの例として，洗車モードに遷移させるアプリケーションが，他のアプリケーションやウィンドウ開閉スイッチでウィンドウを開けられないようにするために使用することが挙げられる。

ロックを所有するのは，アプリケーションのインスタンスである。したがって，アプリケーションがマルチスレッド実装されている場合，アプリケーションのインスタンス内のスレッド間では，ロック状態が共有される。あるスレッドがロックを取得すれば，同じインスタンス内の他のスレッドもロック権限を持つ。逆に，同じアプリケーションであっても，別インスタンスであればロックは共有されない。

2.5.1.3. 操作優先度

あるアプリケーションがオブジェクトをロック中に，他のアプリケーションがそれをオーバーライドすることを可能にするために，操作優先度を導入する。操作優先度は，標準的には，ロックに対してのみ意味を持つ。ただし，オブジェクトによっては，操作優先度に追加の意味を持たせる場合もある。

ロックを取得するサービスコールは，パラメータに操作優先度を持つ。アプリケーションAが操作優先度Pを指定してロックを取得した場合，そのオブジェクトは，アプリケーションAによって操作優先度Pでロックされていると言う。

また，ロック機能を持つオブジェクトを操作するサービスコール（オブジェクトの状態を変化させないものを除く）も，パラメータに操作優先度を持つ。

アプリケーションAによって操作優先度Pでロックされているオブジェクトに対して，別のアプリケーションBが操作優先度Qを指定してロック取得/操作するサービスコールを呼び出した場合の振る舞いは次の通り。

- QがPと同じかより低い操作優先度である場合，ロック取得/操作サービスコールは，E_OBJECT_LOCKEDエラーとなる。
- QがPより高い操作優先度である場合，アプリケーションAによるロックは強制解除され，ロック取得/操作サービスコールは成功する（ロック取得サービスコールの場合，アプリケーションBがロックを奪うことになる）。

なお，アプリケーションA自身がPより高い操作優先度Qを指定してロック取得サービスコールを呼び出した場合，ロックの強制解除は行われず，ロックを取得した操作優先度がQに更新される。

アプリケーションが使用できる操作優先度の最高値は，オブジェクトクラス毎に，アプリケーションのマニフェストに記述する（Section 2.5.2.3参照）。ロック取得/操作サービスコールに対して，使用できる最高値よりも高い操作優先度を指定した場合，そのサービスコールはE_DENIED_PRIORITYエラーとなる。

なお，ロック取得/操作サービスコールの操作優先度パラメータはオプションである。操作優先度を指定しない場合，そのアプリケーションが使用できる操作優先度の最高値を指定したものと扱う。

2.5.1.4. 他の調停方法

先勝ちの調停は，ロックを用いて実現できる。例えば，ウィンドウを閉めるサービスコールを呼び出してからウィンドウが完全に閉まるまで，他のアプリケーションによる操作を許したくない場合には，ウィンドウをロックした後にウィンドウを閉めるサービスコールを呼び出し，ウィンドウが完全に閉まったらロックを解除すれば良い。

2.5.2. リスク制御機能

本仕様では，人身，財産，物品，環境，プライバシに害が生じることを危険と呼び，危険を生じる可能性をリスクと呼ぶ。

ビークルOSは，何らかのリスクがあるAPIの使用をアプリケーションに許すかどうかを制御するために，リスク制御の機能を持つ。リスク制御機能は，人身に対するリスクだけでなく，財産・物品・環境に対するリスクやプライバシ上のリスクも対象とする。

ここでは，リスク制御機能の仕様を規定する。

2.5.2.1. リスククラスの定義

操作することによって何らかの危険を生じる可能性があるオブジェクトに対しては，本仕様で，そのオブジェクトに対するリスククラスを定義する。リスククラスは，「リスクを分類したもの」の意味である。

一般には，各オブジェクトに対して複数のリスククラスがある。例えば，ウィンドウに対しては，次のリスククラスがある。

RiskPinch	ウィンドウを閉めることに伴う挟み込みのリスク
RiskOpen	ウィンドウを開けることに伴う諸々のリスク

ある操作が，あるリスククラスに該当する危険を実際に生じる可能性があるかは，操作の種類（サービスコールとそのパラメータ）や車両状態に依存する。例えば，ウィンドウの状態参照やウィンドウを開く操作では，RiskPinchに該当する危険は生じない。また，駐車状態でのビュー（ディスプレイの全体またはその一部分）に対する描画操作では，RiskDistractionCid（センターディスプレイに描画することで生じるドライバディストラクションのリスク）に該当する危険は生じない。リスククラスに該当する危険を生じない条件がある場合には，サービスコールの仕様において，それを明確にする。

プライバシ上のリスクの例として，アプリケーションを通して自車位置が漏洩することが考えられる。これを制御するために，自車位置・方位は，次のリスククラスを持つ。

RiskPrivacy	プライバシ上のリスク

【補足説明】

リスククラスは，ISO　26262と関連づけて説明すると，「ハザードを，その重大度や対策の方法・難易度によって分類したもの」と定義することができる。

リスクをどれだけまとめるか（逆に言うと，どれだけ区別するか）は難しい課題である。アプリケーションを想定し，リスクの重大度と，ビークルOSおよびアプリケーションでの対策方法を検討した上で，決める必要がある。

2.5.2.2. 車両の構成記述でのリスク記述

車両によって備えている安全機能が異なることから，オブジェクトを操作することに伴うリスクも車両によって異なる。例えば，ウィンドウに挟み込み防止機能を持つ車両は，RiskPinchに対応できているため，アプリケーションがRiskPinchに該当する危険を生じる可能性がある操作をしても，実際には危険を生じない。

車両によるリスクの違いを表現するために，車両の構成記述では，各オブジェクト（の各インスタンス）に対して，車両およびビークルOSで対応できていない（危険を防止できていない）リスククラスの集合を記述する。

2.5.2.3. アプリケーションでのリスクへの対応とOEMによる審査

車両およびビークルOSで対応できていないリスクのあるAPIを使用したいアプリケーションは，アプリケーションでリスクに対応する必要がある。例えば，RiskPinchのリスクがあるウィンドウをアプリケーションが閉める場合，音声等で「これからウィンドウを閉めます」と伝え，ユーザがいつでも閉動作を停止できる手段を提供して，閉動作を行うといった方法が考えられる。リスクへの対応方法には，「ユーザに制限事項を説明する」や「アプリケーション提供者が事故に対する責任を取る」といった方法も含まれる。

アプリケーションのマニフェストには，オブジェクトクラス毎に，アプリケーションで対応したリスククラスの集合と，アプリケーションが使用できる操作優先度の最高値を記述する。操作優先度の最高値の記述がないオブジェクトクラスについては，アプリケーションが使用できる操作優先度の最高値は，最も低い操作優先度とする。

アプリケーションのマニフェストには，OEMによる署名が必要である。OEM（またはOEMから委託された者）は，アプリケーションによるリスクへの対応が十分であるか，および，アプリケーションが使用できる操作優先度の最高値が妥当であるかを審査し，十分かつ妥当であると判断した場合には，マニフェストに署名を付与する。

【補足説明】

アプリケーションのマニフェストでは，対応したリスククラスの集合と使用できる操作優先度の最高値を，オブジェクトインスタンス毎ではなく，オブジェクトクラス毎に記述する。これは，オブジェクトインスタンス毎に記述するとした場合，マニフェストが車両に依存したものとなり，アプリケーションの審査の工数が大きくなるためである。

すべてのリスクに対する責任を，アプリケーション提供者に移転できる場合（例えば，プライバシ上のリスクのみがある場合）には，OEMは，リスクへの対応の審査を省略することができる。また，OEM間で審査基準を統一できる場合には，複数のOEMに対して一括して審査を行うことができる。一方，国や地域によって法制度や規制が異なるため，国や地域毎に行うことが必要と思われる。

OEMによる審査において，各リスククラスへの対応が十分であると判断する基準は，本仕様のスコープ外である。

2.5.2.4. ユーザの承認

ビークルOSは，リスクのあるAPIを使用するアプリケーションのインストール時（または，最初の実行時）に，そのリスクに対応する責任が，OEMからアプリケーション提供者に移転されることに関して，ユーザの承認を求める。

例えば，RiskPinchのリスクがあるウィンドウをアプリケーションが閉める場合には，以下のようなメッセージによりユーザの承認を求める。

> 本アプリケーションは，ウィンドウの閉動作を行います。動作中，身体の一部や物品等が挟まれるおそれがあります。周囲の安全を確認し，異常時は直ちに停止してください。
>
> 本アプリケーションの使用により生じた損害について，［OEM］は責任を負いません。責任および補償の範囲は，［アプリケーション提供者］の利用規約に従います。
>
> アプリケーションによるウィンドウの閉動作を許可しますか？

ユーザの承認結果は，ユーザ個人に紐づいたユーザプリファレンスデータ内に格納する。ユーザプリファレンスデータ内に格納した承認結果は，設定などの機能により，ユーザが変更できるようにする。

OEMの純正アプリケーション（標準アプリケーションを含む）については，車両を販売する時点などで，ユーザから包括的な承認を得ておくことで，アプリケーション毎のユーザの承認を省略できるようにする。

2.5.2.5. 操作可否の決定

危険を生じる可能性がある操作に対して，その危険が該当するリスククラスの内，操作対象のオブジェクトインスタンスについて車両およびビークルOSで対応できていないものを，未対応リスククラスと呼ぶ。

未対応リスククラスがある場合，ビークルOSは，以下の2つの条件を満たす場合に限り，その操作をアプリケーションに許可する。

(a) 未対応リスククラスの集合が，アプリケーションで対応したリスククラスの集合に含まれている。

(b) すべての未対応リスククラスについて，そのリスクに対応する責任が，アプリケーション提供者に移転されていることを，ユーザが承認している。

いずれかの条件が満たされない場合，その操作は禁止される。前述の通り，操作が実際に危険を生じる可能性があるかは，操作の種類（サービスコールとそのパラメータ）と車両状態に依存するため，操作が許可されるか禁止されるかは，車両状態の変化などにより変わる場合がある。

アプリケーションがオブジェクトを操作するサービスコールを呼び出した場合，ビークルOSは，その時点の車両状態に基づいて操作の可否を判定する。操作が禁止される場合，サービスコールは，(a)の条件が満たされない場合はE_RISK_APPLICATIONエラーを，(b)の条件が満たされない場合はE_RISK_USERエラーを返す。

操作が許可された後，その操作が継続している間に，車両状態の変化などにより操作が禁止されるべき状態になった場合には，ビークルOSはその操作を中止し，(a)の条件が満たされなくなった場合はRiskApplicationRejectedイベントを，(b)の条件が満たされなくなった場合はRiskUserRejectedイベントを発生させる。操作の中止によって，アプリケーションが取得しているロックは解除されない。操作が継続しているとみなす期間と，操作を中止する方法は，オブジェクト毎のAPIの規定で定める。

必要なサービスコール以外は呼び出せないようにするアクセス制御の機能によりサービスコールの呼び出しが拒否された場合，エラーコードはE_DENIED_ACCESSとする。

Chapter 3. 共通規定

この章では，複数のオブジェクトのAPIに共通する事項について規定する。データ型の名称中の*xxx*は，オブジェクトの略称を示す。

3.1. 共通データ型

この節では，複数のオブジェクトのAPIに共通に使用するデータ型について規定する。

(1) 戻り値のデータ型

サービスコールの戻り値を表すデータ型として，ReturnTypeを用意する。現時点の定義は以下の通り。今後，必要に応じてエラーコードを追加する。

```
type ReturnType = enum {
    E_OK,                   // 正常終了
    E_INVALID_ID,           // IDが不正
    E_INVALID_HANDLE,       // ハンドルが不正
    E_INVALID_PARAMETER,    // その他のパラメータが不正
    E_RISK_APPLICATION,     // アプリケーションが対応していないリスクがある
    E_RISK_USER,            // ユーザがリスク対応責任の移転を承認していない
    E_DENIED_PRIORITY,      // 使用できない操作優先度
    E_DENIED_ACCESS,        // アクセスが認められていない
    E_OBJECT_FAULT,         // 対象オブジェクトが故障中
    E_OBJECT_LOCKED,        // 対象オブジェクトがロックされている
    E_OBJECT_STATUS,        // 対象オブジェクトが指定された操作をできない状
                            // 態にある
    E_INFEASIBLE,           // 指定された操作が現在の状態では実現できない
    E_NO_DATA,              // データがない
    E_NO_MEMORY,            // メモリ不足
    E_NOT_SUPPORTED,        // サポートされていない機能
    E_COMMUNICATION,        // 通信エラー
    E_NOT_OK                // その他のエラー
}
```

E_COMMUNICATIONは，サービスコールが複数のECUで実現される場合で，他のECUとの通信エラーや他のECUのダウンによりサービスコールが正常に実行できなかったことを示す。

E_NO_MEMORY，E_NOT_SUPPORTED，E_COMMUNICATIONは，すべてのサービスコールで発生する可能性があるため，個々のサービスコールが返すエラーコードには列挙しない。

物理APIにおいて，エラーコードを整数値で表現する場合には，ReturnTypeを32ビットの符号付き整数型で実現する。

(2) インスタンスIDのデータ型

オブジェクトのインスタンスIDを表すデータ型として，IdTypeを用意する。定義は以下の通り。

```
type IdType = Uint16 ([1,65535]);
```

オプショナル型をサポートしていないプログラミング環境では，IdType?型のnullを表すために，値0（ID_NULL）を使用する。すなわち，ID_NULLは，IDが指定されていないことを示す。

(3) 操作優先度のデータ型

操作優先度を表すデータ型として，PriorityTypeを用意する。操作優先度は，値が小さい方が優先度が高い。定義は以下の通り。

```
type PriorityType = Uint8 ([1,100]);
```

オプショナル型をサポートしていないプログラミング環境では，PriorityType?型のnullを表すために，値0（PRIORITY_NULL）を使用する。すなわち，PRIORITY_NULLは，操作優先度が指定されていないことを示す。

(4) アプリケーションIDのデータ型

アプリケーションのインスタンスを識別するためのIDをアプリケーションIDと呼び，それを表すデータ型として，ApplicationIdTypeを用意する。具体的な型定義は，物理APIにおいて規定する。

標準制御（OEMが自動車の出荷時に組み込んでいる制御ロジック）に対しても，アプリケーションIDを割り当てる。

(5) オブジェクトを表すデータ型

オブジェクト種別を表すデータ型として，ObjectKindTypeを用意する。定義は以下の通り（列挙子は順次増やしていく）。

```
type ObjectKindType = enum {
    Vehicle,
    VersionOSDVI,
    VehicleId,
    VehicleSpec,
    Cabin,
    Door,
    Window,
    CargoSpace,
    Body,
    Infotainment,
    HMI,
    Motion,
    CurrentLocation,
    Driver,
    Occupant,
    SurroundModel,
    Atmosphere,
    ADADAS,
    Energy,
    Navigation,
    .........
};
```

関連するオブジェクトを表すデータ型として，RelatedObjectTypeを用意する。定義は以下のいずれかから，物理APIで選択する。

```
type RelatedObjectType = String;
```

```
type RelatedObjectType = struct {
    objectKind : ObjectKindType,
    instanceName : String?
};
```

```
type RelatedObjectType = struct {
    objectKind : ObjectKindType,
    instanceId : IdType?
};
```

instanceNameおよびinstanceIdは，マルチインスタンスの場合のみ値を持つ。

YAML形式で記述した構成記述ファイル中では，第1の形式（つまり，文字列）で記述するが，構成情報を参照するサービスコールは，物理APIで定められた形式に変換して返す。

(6) オブジェクトの場所を表すデータ型

キャビン内のオブジェクトの場所を表すデータ型として，PlacementTypeを用意する。定義は以下の通り。

```
type ZoneType = enum {
    Left,       // 左
    Right,      // 右
    Center,     // 中央
    Top,        // 上
    Bottom      // 下
};

type PlacementType = struct {
    row : Uint8,        // 座席の何列目か
    zone : ZoneType     // 左右上下方向の位置
};
```

(7) タイムスタンプ

ナノ秒単位で絶対時刻を表すデータ型として，TimestampNsTypeを用意する。TimestampNsTypeで表される絶対時刻は，UNIXエポックからのナノ秒単位の経過時間（うるう秒を含まない）であり，負の値は用いない（ナノ秒単位の時間精度を要求するものではない）。時刻が不明な場合には，0とする。時刻同期処理によって，時刻の値が不連続に変化する場合がある。

```
type TimestampNsType = Int64;
```



時刻の値が不連続に変化した場合に，アプリケーションにそれを知らせる方法については，今後の課題とする。

3.2. すべてのオブジェクトが持つサービスコール

この節では，原則としてすべてのオブジェクトが持つサービスコールについて規定する。

getConfig

オブジェクトの構成情報の参照（シングルトンの場合）

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
config	*xxx*ConfigType	対象オブジェクトのすべての構成情報

【エラーコード】

なし

【機能】

対象オブジェクトのすべての構成情報を返す。

getConfigAll

オブジェクトのすべてのインスタンスの構成情報の参照（マルチインスタンスの場合）

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
config	*xxx*ConfigType[]	対象オブジェクトのすべてのインスタンスのすべての構成情報

【エラーコード】

なし

【機能】

対象オブジェクトのすべてのインスタンスのすべての構成情報を返す。

getStatus

オブジェクト状態の参照

【パラメータ】

instanceId	IdType	状態参照対象のインスタンスID（マルチインスタンスの場合のみ）

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
status	*xxx*StatusType	対象オブジェクトの状態

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外（マルチインスタンスの場合のみ）

【機能】

状態参照対象のオブジェクト（のインスタンス）の状態を返す。

notify

イベント通知の要求

【パラメータ】

instanceId	IdType?	イベント通知対象のインスタンスID（マルチインスタンスの場合のみ）
eventFilter	enumSet(*xxx* EventKindType)?	通知するイベントの種類の集合

※ 通知するイベントの種類をより細かく指定するために，パラメータを追加しても良い。

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
notifyHandle	NotifyHandleType	生成したイベントキューの通知ハンドル

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外（マルチインスタンスの場合のみ）
E_INVALID_PARAMETER	eventFilterが有効範囲外

【機能】

イベント通知対象のオブジェクト（のインスタンス）で発生したeventFilterで指定した種別のイベントの通知を要求する。イベントを格納するためのイベントキューを生成し，その通知ハンドルをnotifyHandleに返す。

instanceIdを省略した場合，すべてのインスタンスで発生したイベントの通知を要求する。eventFilterを省略した場合，すべての種別のイベントの通知を要求する。

unnotify

イベント通知の停止

【パラメータ】

notifyHandle	NotifyHandleType	通知停止するイベントキューの通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正

【機能】

notifyHandleで指定した通知を停止する。イベントを格納するためのイベント キューを削除する。

getEvent

イベントの取得

【パラメータ】

notifyHandle	NotifyHandleType	イベントを取得するイベントキューの通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
event	*xxx*EventType	取得したイベント

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正
E_NO_DATA	イベントキューが空

【機能】

notifyHandleで指定した通知用のイベントキューから，先頭のイベントを取り出す。取り出したイベントは，イベントキューから削除される。

【補足説明】

イベントがパラメータ等で渡されてくるプログラミング環境向けの物理APIでは，用意しなくても良い。

3.3. 共通のイベント

この節では，複数のオブジェクトに共通するイベントについて規定する。各オブジェクトが持つイベント種別は，オブジェクト毎のAPIの規定で定める。

(1) 構成情報の変更

オブジェクトに対するイベント種別には，そのオブジェクトの構成情報が変化した時に発生する以下のイベント種別を含むことを原則とする。

ConfigChanged	構成情報の変更

マルチインスタンスオブジェクトにおいて，インスタンスが増えたことを知るためには，イベント通知対象のインスタンスIDを指定せずにイベント通知を要求する必要がある。

(2) イベントキューのオーバフロー

オブジェクトに対するイベント種別には，イベントキューがオーバフローし，古いイベントが削除された時に，イベントキューの先頭に入れられる以下のイベント種別を含むことを原則とする。

EventOverflow	イベントキューのオーバフロー

このイベント種別は，以下の関連情報を持つ。

noLostEvents	Uint32	失われたイベントの数

(3) リスク制御機能による操作の中止

リスククラスを持つオブジェクトに対するイベント種別には，リスク制御機能により継続中の操作が中止された時に発生する以下のイベント種別を含む。

RiskApplicationRejected	アプリケーションがリスクに未対応
RiskUserRejected	ユーザがリスク対応責任の移転を未承認

これらのイベント種別は，アプリケーションによる操作が継続している間に，車両状態の変化などにより操作が禁止されるべき状態になり，リスク制御機能が操作を中止した時に発生する。これらのイベント種別は，操作を要求したアプリケーションのみに通知する。

(4) 状態の変化時／定期報告

状態を持つオブジェクトに対するイベント種別には，オブジェクトの状態の全体または一部（これを報告対象の状態と呼ぶ）をアプリケーションに伝えるための以下のイベント種別を含めても良い。

StatusChanged	状態の変化時報告
StatusReported	状態の定期報告

StatusChangedは，報告対象の状態が変化した時に発生する。

StatusReportedは，ビークルOSが報告対象の状態を周期的に更新することを想定したイベント種別であり，その更新の時に，状態の変化の有無にかかわらず発生する。この更新の周期をミリ秒単位に丸めた値を，構成情報のstatusReportIntervalで示す。

これらのイベント種別は，以下の関連情報を持つ。statusのデータ型は，オブジェクト毎のAPIの規定で定める。

timestamp	TimestampNsType	イベントが発生した時刻
status	オブジェクト毎に規定	イベントが発生した時点の報告対象の状態

以下の事項は，オブジェクト毎のAPIの規定で定める。notifyサービスコールのパラメータで指定できるものとしても良い。

- 報告対象の状態（オブジェクトの状態のどの部分を伝えるか）
- 連続的に変化する値を報告対象の状態に含む場合に，前回報告時からどの程度の変化でStatusChangedを発生させるか
- これらのイベント種別を，最新値イベント（Section 2.3.8.3参照）として扱うか

イベント種別にStatusReportedを含むオブジェクトの構成情報には，以下の情報を含める。

statusReportInterval	Uint32	報告対象の状態の更新周期（ミリ秒単位。丸めた値）

3.4. LockableObject

API仕様の記述を簡潔にするために，標準的なロック機能を，LockableObjectとして定義する。オブジェクト毎の仕様に「このオブジェクトはLockableObjectである」という記述があると，ここで説明する標準的なロック機能を持つことが意味する。

LockableObjectは，以下のAPIを持つ。

- ロック/ロック解除のサービスコールを持つ
- 状態参照によりロック状態を参照できる
- ロック操作に関するイベントを持つ

3.4.1. LockableObjectの機能

LockableObjectは，他のアプリケーションがオブジェクトの操作をすることを禁止するためのロック機能を持つ。ロックは，ハードウェアで行われる場合（例えば，ドアロック）と，ソフトウェアでのみ行われる場合がある。ロック機能の詳細については，Section 2.5.1.2を参照のこと。

オブジェクトが異常状態（故障中やオブジェクト依存の異常状態）になった場合，ロックは解除され，その間はロックを取得することができない。

アプリケーションが終了すると，そのアプリケーションが取得していたロックは，ロック解除される。

3.4.2. LockableObjectの状態

LockableObjectは，オブジェクト状態を参照するサービスコールにより，ロック状態を参照することができる。

ロック状態には，以下の情報を含む。

- ロックされているか否か（ロックされている場合にtrue）

- ロックしたアプリケーションID（ロックされている場合のみ）
- ロックを取得した操作優先度（ロックされている場合のみ）

オブジェクトが異常状態の間は，ロックされているか否かはfalseになる。

3.4.3. LockableObjectのデータ型

LockableObjectの状態には，以下のデータ型で表されるロック状態を含む。

```
type LockStatusType = struct {
    locked : Bool,                          // ロックされているか否か
    lockApplication : ApplicationIdType?,   // ロックしたアプリケーションID
    lockPriority : PriorityType?            // ロックを取得した操作優先度
};
```

lockApplicationとlockPriorityは，lockedがtrueの場合にのみ値を持つ。

3.4.4. LockableObjectのイベント

LockableObjectに対するイベント種別には，以下のイベント種別を含む。

OwnLockRevoked	自アプリケーションのロックの強制解除
OtherLockReleased	他のアプリケーション等によるロック解除

OwnLockRevokedは，自アプリケーションがロックしているオブジェクトが，他のアプリケーションが高い操作優先度で操作またはロック取得したことや，異常状態になったことにより，強制的にロック解除された時に発生する。

OtherLockReleasedは，他のアプリケーションがロックしているオブジェクトが，何らかの理由でロック解除され，その結果，ロックを取得できる状態になった時に発生する。ただし，異常状態の間はロックを取得できないため，OtherLockReleasedは，異常状態になった時ではなく，異常状態から回復した時に，異常状態になった時にロックが取得されていたか否かにかかわらず発生する。なお，ロックしているアプリケーションが変わった時には，OtherLockReleasedは発生しない。

ロック状態の遷移と発生するイベントの関係をFigure 8に示す。

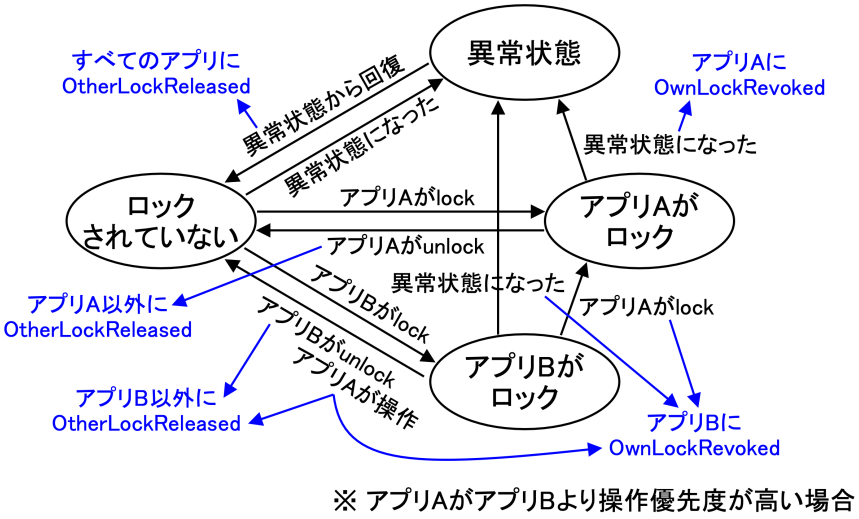


Figure 8. ロック状態の遷移とイベント

【補足説明】

OtherLockReleasedのユースケースとして，自アプリケーションがオブジェクトを操作またはロックできるようになったタイミングを知るために使用することを想定している。

3.4.5. LockableObjectのサービスコール

LockableObjectは，以下のサービスコールを持つ。

lock

ロックの取得

【パラメータ】

instanceId	IdType	ロック対象のインスタンスID（マルチインスタンスの場合のみ）
priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外（マルチインスタンスの場合のみ）
E_INVALID_PARAMETER	priorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_LOCKED	ロック対象のオブジェクトが，priorityと同じかそれより高い優先度でロックされている（自アプリケーションが，priorityと同じかそれより高い優先度でロックしている場合を含む）
E_OBJECT_FAULT	対象オブジェクトが故障中（対象オブジェクトに故障中の状態がある場合のみ）
E_OBJECT_STATUS	対象オブジェクトがロックできない状態にある（対象オブジェクトにそのような状態がある場合のみ）

【機能】

ロック対象のオブジェクト（のインスタンス）を，指定した操作優先度でロックする。

ロックすることで，制御対象の状態（ロック状態を除く）は変化しない。すなわち，現在の動作をそのまま継続する。

unlock

ロックの解除

【パラメータ】

instanceId	IdType	ロック解除対象のインスタンスID（マルチインスタンスの場合のみ）

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外（マルチインスタンスの場合のみ）
E_OBJECT_STATUS	自アプリケーションが，ロック解除対象のオブジェクトをロックしていない

【機能】

ロック解除対象のオブジェクト（のインスタンス）を，ロック解除する。

3.5. MovableObject

API仕様の記述を簡潔にするために，移動する可動部を持つオブジェクトに共通の機能を，MovableObjectとして定義する。オブジェクト毎の仕様に「このオブジェクトはMovableObjectの機能を持つ」という記述があると，ここで説明する機能を持つことを意味する。以下でMovableObjectと言った場合には，MovableObjectの機能を持つオブジェクトを指す。

MovableObjectは，以下のAPIを持つ。

- 状態参照により移動に関する状態を参照できる。
- 移動を開始/停止するサービスコールを持つ。
- 移動に関する状態の変化のイベントを持つ。

3.5.1. MovableObjectの機能

MovableObjectは一次元に移動する可動部を持つ。二次元（またはそれ以上）に移動する可動部を持つオブジェクトも，2つの次元それぞれがMovableObjectと同等の機能を持つという形で規定できる。

MovableObjectの可動部の移動には，時間がかかると想定している。そのため，MovableObjectの状態に，可動部の現在位置に加えて，目標位置を持つ。

現在位置と目標位置は，パーセンテージで表す。片方の移動端を0，反対側の移動端を100で表し，どちら

の移動端を0とするかはオブジェクト毎に定める。現在位置と目標位置は，不明を示す値（UNKNOWN），または，0で表される移動端でないことを示す値（NONZERO）になる場合がある。

現在位置の検知精度は，車両に依存する。現在位置が全くわからない場合，現在位置としてUNKNOWNを返す。現在位置が，0で表される移動端でないことのみがわかっている場合，現在位置としてNONZEROを返す。現在位置が，どちらの移動端でもないことのみがわかっている場合の特殊値は用意していないが，検知精度は車両依存としていることから，現在位置として50を返すことができる。

MovableObjectが正常動作している（故障中やオブジェクト依存の異常状態でない）場合でも，現在位置がUNKNOWNやNONZEROになる場合がある（現在位置の学習中の状況などを想定）。目標位置は，UNKNOWNになる場合がある（故障からの回復の直後などを想定）が，NONZEROになることはない。

MovableObjectが正常動作している場合，現在位置と目標位置がUNKNOWNやNONZEROでなく，現在位置と目標位置が一致していない場合，目標位置に向けて可動部を移動する。現在位置がUNKNOWNまたはNONZEROの場合，目標位置が0または100であれば，目標位置に向けて可動部が移動し，他の目標位置では可動部は移動しない。目標位置がUNKNOWNの場合，可動部は移動しない。

可動部の移動中に，現在位置と目標位置が一致したら，移動を停止する。現在位置の検知精度が低いなどの理由で，現在位置が目標位置に完全には一致させられない場合，目標位置に到達した時点で，目標位置を変更して，現在位置に一致させる。

目標位置の設定精度は，車両に依存する。目標位置に0または100を指定した場合，指定された移動端まで移動する。目標位置にそれ以外の値を指定した場合，最終的な位置は移動端にはならない（移動端まで移動してから，少し戻る可能性はある）。

可動部の移動停止にも，時間がかかると想定している。そのため，移動停止サービスコールは，目標位置を現在位置に設定して移動停止を要求するだけで，移動が停止するまで待つわけではない。移動が停止した時点で，再度，目標位置を変更して，現在位置に一致させる。

MovableObjectは，オプションで，可動部をどのように移動させるか（これを移動プロファイルと呼ぶ）を指定する機能を持つ。移動プロファイルでは，移動速度に基づく3段階（標準/高速/低速）の指定を標準とするが，オブジェクトによっては，さらに多様な動作パターンを指定できる場合もある。また，OEM固有の移動プロファイルを指定できるようにしても良い。移動プロファイルを指定しない場合，標準的な移動プロファイルが適用される。なお，オブジェクトが移動プロファイルをサポートしていない場合，移動プロファイルの指定は無視される。

3.5.2. MovableObjectの状態

MovableObjectは，オブジェクト状態を参照するサービスコールにより，移動に関する状態を参照することができる。

移動に関する状態には，以下の情報を含む。

- 移動主状態
- 現在位置
- 目標位置

移動主状態は，以下のいずれかの状態を表す。

- 停止中
- 位置のパーセンテージが大きくなる方へ移動中
- 位置のパーセンテージが小さくなる方へ移動中

- 故障中

ここでは，恒久故障（修理するまで治らない）と一時故障（自然に治る可能性がある）は区別しない。

移動主状態とその間の状態遷移をFigure 9に示す。

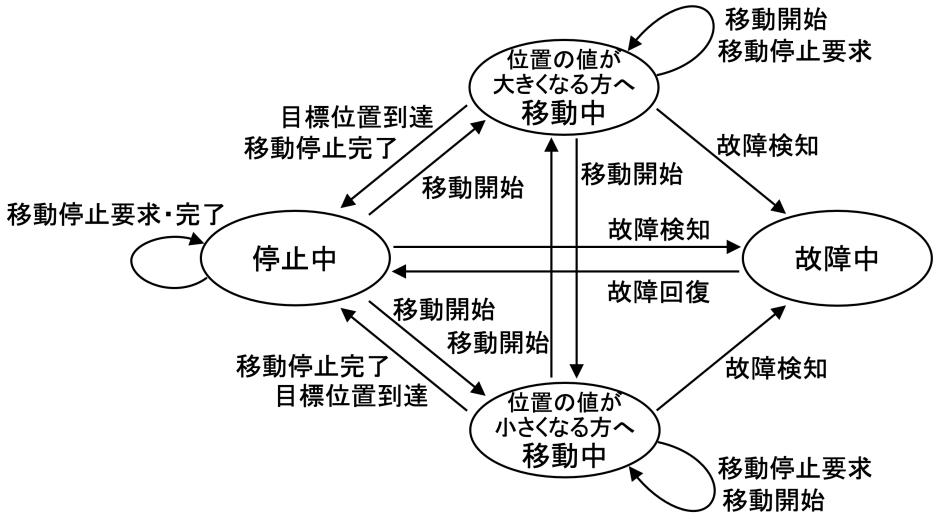


Figure 9. 移動主状態の状態遷移

以上で挙げた以外に，オブジェクト固有の主状態や状態遷移を追加しても良い。

3.5.3. MovableObjectのイベント

MovableObjectに対するイベント種別には，以下のイベント種別を含む。

ControlledBySelf	自アプリケーションが制御
ControlledByOther	他のアプリケーションが制御
TargetReached	指定された目標位置に到達して停止，または，移動停止が完了
FaultDetected	故障検知
FaultRecovered	故障から回復

この内，イベント種別がControlledByOtherかTargetReachedの場合には，以下の関連情報を持つ。

sourceApplication	ApplicationIdType	操作したアプリケーション

ControlledBySelfとControlledByOtherは，移動開始/停止サービスコールが呼び出され，正常終了した場合（戻り値がE_OKの場合）に発生する。

TargetReachedは，移動開始サービスコールによって指定された目標位置に到達して停止した時と，移動停止サービスコールによって要求された移動停止が完了した時に発生する。現在位置を目標位置に指定して呼び出した移動開始サービスコールや，停止中に呼び出した移動停止サービスコールに対しても発生する。

移動開始サービスコールによる操作は，可動部が目標位置に到達して停止するか，移動停止サービスコールまたは他のアプリケーション等の操作によってオーバライドされるまで継続しているものとみなす。リスク制御機能により操作が中止された場合，可動部は停止し，RiskApplicationRejectedまたはRiskUserRejectedが発生する。この場合，TargetReachedは発生しない。

なお，ControlledBySelf，ControlledByOther，TargetReachedは，オブジェクトの状態が変化しない場合にも発生する（イベントの発生条件に関する一般規則の例外）。

他の理由で移動主状態が変化するオブジェクトの場合には，その移動主状態の変化に関するイベント種別を追加する。

MovableObjectに対するイベントが連続して発生した場合に，一部のイベントが抜ける場合がある。ただし，イベントキューがオーバフローした場合を除いて，ControlledBySelfが抜けることはない。また，連続して発生した移動に関するイベント（RiskApplicationRejectedとRiskUserRejectedを含む）の中で，最後のイベントが抜けることはない。

【補足説明】

MovableObjectに対するイベントのユースケースとして，次の2つを想定している。

(a) 自アプリケーションによる移動開始/停止要求が完了したこと（または，他のアプリケーション等によって移動要求がオーバライドされたこと）を知るため。具体的な方法は，startMove/stopMoveのAPIの使用方法の項で説明する。

(b) ウィンドウの状態を常に知りたい場合に，状態を参照するタイミングを知るため。ただし，現在位置のみの変化ではイベントは発生しないため，ウィンドウの開閉動作中の現在位置を常に知りたい時には，開閉動作中はポーリングすることが必要である。

3.5.4. MovableObjectのデータ型

(1) 位置を表すデータ型

MovableObjectの可動部の位置をパーセンテージで表すデータ型として，PositionTypeを用意する。0と100の意味は，オブジェクト毎に定める。定義は以下の通り。

```
type PositionType = Int8 ([0,100] | UNKNOWN(-1) | NONZERO(-2))
```

UNKNOWNは位置が不明であることを，NONZEROは0で表される移動端でないことを示す。

(2) 移動に関する状態のデータ型

MovableObjectは，移動主状態を表すデータ型として，オブジェクト毎に，*xxx*MainStatusTypeを用意する。定義は以下のようになる。

```
type xxxMainStatusType = enum {
    Stopped,        // 停止中
    Increasing,     // 位置のパーセンテージが大きくなる方へ移動中
    Decreasing,     // 位置のパーセンテージが小さくなる方へ移動中
    Fault,          // 故障中
    .........
};
```

IncreasingとDecreasingは，オブジェクト毎に，その動きに合わせて名前を変えても良い。また，オブジェクト毎に，この他の主状態を追加しても良い。

MovableObjectの状態には，移動に関する状態を示す以下のフィールドを含む。

```
type xxxStatusType = struct {
    .........
    mainStatus : xxxMainStatusType, // 移動主状態
    position : PositionType,            // 現在位置
    targetPosition : PositionType,      // 目標位置
    .........
};
```

(3) 移動プロファイルのデータ型

MovableObjectは，移動プロファイルを表すデータ型として，オブジェクト毎に，*xxx*MoveProfileTypeを用意する。定義は以下のようになる。

```
type xxxMoveProfileType = enum {
    Standard,       // 標準的な移動
    Fast,           // 高速で移動
    Slow,           // 低速で移動
    .........
};
```

(4) MovableObjectのイベントのデータ型

MovableObjectのイベント種別には，MovableObjectが持つイベント種別を含む。

```
type xxxEventKindType = enum {
    .........
    ControlledBySelf,       // 自アプリケーションによる制御
    ControlledByOther,      // 他のアプリケーションによる制御
    TargetReached,          // 目標位置到達，移動停止完了
    FaultDetected,          // 故障検知
    FaultRecovered,         // 故障回復
    .........
};
```

MovableObjectのイベントのデータ型は，以下のような定義となる。

```
type xxxEventType = struct {
    instanceId : IdType,    // マルチインスタンスオブジェクトの場合のみ
    eventInfo : variant(xxxEventKindType) {
        .........
        ControlledByOther {
            sourceApplication : ApplicationIdType
        }
        TargetReached {
            sourceApplication : ApplicationIdType
        }
        .........
```

```
    }
};
```

3.5.5. MovableObjectのサービスコール

MovableObjectは，以下のサービスコールを持つ。

startMove

移動開始

【パラメータ】

instanceId	IdType	操作対象のインスタンスID（マルチインスタンスの場合のみ）
targetPosition	PositionType	目標位置
moveProfile	*xxx*MoveProfileType?	移動プロファイル
priority	PriorityType?	操作優先度（LockableObjectの場合のみ）

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

オブジェクト毎に規定する

【機能】

目標位置をtargetPositionに設定して，操作対象のMovableObject（のインスタンス）の可動部の移動を開始する。

可動部の移動には時間がかかると想定しているため，このサービスコールは移動開始を要求するだけで，目標位置に到達するまで待つわけではない。

可動部が停止している状態で，targetPositionに現在位置を指定してこのサービスコールを呼び出した場合でも，ControlledBySelfとTargetReachedは，この順序で発生する。

【APIの使用方法】

自アプリケーションが発行したstartMoveにより要求した可動部の移動が，目標位置に到達して完了したか，または，他のアプリケーション等によって移動要求がオーバライドされたかを知るためには，次の手順をとる。

- 移動に関するイベントの通知を要求しておく。
- この時点までに生成されたイベントを読み出しておく。
- startMoveを発行する。
- ControlledBySelfを受け取るまで，イベントを読み捨てる。
- 関連情報のsourceApplicationが自アプリケーションであるTargetReachedを受け取れば，目標位置に

到達して移動が完了したと判断する。

- RiskApplicationRejectedまたはRiskUserRejectedを受け取れば，リスク制　御機能により移動が中止されたと判断する。
- それ以外のイベントを受け取れば，他のアプリケーション等によって移動要求がオーバライドされたと判断する。

なお，この方法は，1つのアプリケーションが同一のオブジェクトに対して並行してサービスコールを呼び出すことは想定していない。サービスコールを並行して呼び出さないようにするのは，アプリケーションの責任である。

【補足説明】

制御精度が低いなどの理由で，移動が完了した時点の現在位置が，指定した目標位置に完全には一致しないことを許容しているため（この場合，移動が完了した時点で，目標位置を変更して，現在位置に一致させる），現在位置と自アプリケーションが指定した目標位置が一致したことをもって，自アプリケーションが要求した移動が完了したと判断することはできない。

stopMove

移動停止

【パラメータ】

instanceId	IdType	操作対象のインスタンスID（マルチインスタンスの場合のみ）
priority	PriorityType?	操作優先度（LockableObjectの場合のみ）

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

オブジェクト毎に規定する

【機能】

目標位置を現在位置に設定して，操作対象のMovableObject（のインスタンス）の可動部の移動停止を要求する。

可動部の移動停止には時間がかかると想定しているため，このサービスコールは移動停止を要求するだけで，移動が停止するまで待つわけではない。そのため，このサービスコールからのリターン後に主状態を参照しても，停止中（STOPPED）になっているとは限らない。

可動部が停止している状態でこのサービスコールを呼び出した場合でも，ControlledBySelfとTargetReachedは，この順序で発生する。

【APIの使用方法】

自アプリケーションが発行したstopMoveにより要求した可動部の移動停止が完了したか，または，他のアプリケーション等によって停止要求がオーバライドされたかを知るためには，次の手順をとる。

- 移動に関するイベントの通知を要求しておく。

- この時点までに生成されたイベントを読み出しておく。
- stopMoveを発行する。
- ControlledBySelfを受け取るまで，イベントを読み捨てる。
- 関連情報のsourceApplicationが自アプリケーションであるTargetReachedを受け取れば，停止が完了したと判断する。
- それ以外のイベントを受け取れば，他のアプリケーション等によって停止要求がオーバライドされたと判断する。

なお，この方法は，1つのアプリケーションが同一のオブジェクトに対して並行してサービスコールを呼び出すことは想定していない。サービスコールを並行して呼び出さないようにするのは，アプリケーションの責任である。

Chapter 4. コアビークルAPI

4.1. Vehicle（略称：Vehicle）

＜TBD＞

4.2. Vehicle.VersionOSDVI（略称：VersionOSDVI）

4.2.1. 位置付けと機能

Vehicle.VersionOSDVI（略称：VersionOSDVI）は，車両がサポートするOpen SDV API仕様のバージョンを参照するためのシングルトンオブジェクトである。

4.2.2. 構成情報

VersionOSDVIは，以下の構成情報を持つ。

release	String	バージョン番号（リリース年月）
stage	String?	開発ステージ（α，βなど）
suffix	String?	プレリリース識別子（RC1など）

4.2.3. サービスコール一覧

VersionOSDVIに対するサービスコールは以下の通り。

getConfig	VersionOSDVIの構成情報の参照

4.2.4. データ型の定義

4.2.4.1. VersionOSDVIの構成情報（getConfigが返すデータ型）

```
type VersionOSDVIConfigType = struct {
    release : String,
    stage : String?,
    suffix : String?
};
```

4.2.5. サービスコールの詳細規定

getConfig

VersionOSDVIの構成情報の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
config	VersionOSDVIConfigType	VersionOSDVIの構成情報

【エラーコード】

なし

【機能】

VersionOSDVIの構成情報を返す。

4.3. Vehicle.VehicleIdentification（略称：VehicleId）

4.3.1. 位置付けと機能

Vehicle.VehicleIdentification（略称：VehicleId）は，車両の識別情報を参照するためのシングルトンオブジェクトである。



COVESA VSS 6.0のVehicle.VehicleIdentificationに対して，車両の最大乗客数（vehicleMaxCapacity）を追加した。各構成情報の詳細は，COVESA VSSの仕様書に記載されている。

4.3.2. 構成情報

VehicleIdは，以下の構成情報を持つ。

vin	String?	ISO 3779で定義されるの車両識別番号（VIN）
wmi	String?	ISO 3780で定義される世界製造業者識別コード（WMI）
brand	String?	車両のブランドまたは製造メーカ
model	String?	車両のモデル
year	String?	車両のモデルイヤー
acrissCode	String?	ACRISS車種分類コード
bodyType	String?	車両のデザインとボディスタイル
dateVehicleFirstRegistered	String?	車両の初回登録日（ISO 8601形式）
licensePlate	String?	車両のナンバープレート
meetsEmissionStandard	String?	車両が満たしている排出ガス基準

productionDate	String?	製造日（ISO 8601形式）
purchaseDate	String?	購入日（ISO 8601形式）
vehicleModelDate	String?	車両モデルのリリース日（ISO 8601形式）
vehicleConfiguration	String?	車両の構成を示す短い記述
vehicleSeatingCapacity	Uint16?	車両に着座可能な乗客数
vehicleMaxCapacity	Uint16?	車両の最大乗客数
vehicleSpecialUsage	String?	車両の特別な使用目的
vehicleExteriorColor	String?	外装の主たる色（基本色）
vehicleInteriorColor	String?	内装の色または色の組み合わせ
vehicleInteriorType	String?	内装のタイプまたは素材
knownVehicleDamages	String?	既知の損傷（修復済み，未修復の両方）
optionalExtras	String[]	車両の装備オプション

4.3.3. サービスコール一覧

VehicleIdに対するサービスコールは以下の通り。

getConfig	VehicleIdの構成情報の参照

4.3.4. データ型の定義

4.3.4.1. VehicleIdの構成情報（getConfigが返すデータ型）

```
type VehicleIdConfigType = struct {
    vin : String?,
    wmi : String?,
    brand : String?,
    model : String?,
    year : String?,
    acrissCode : String?,
    bodyType : String?,
    dateVehicleFirstRegistered : String?,
    licensePlate : String?,
    meetsEmissionStandard : String?,
```

```
    productionDate : String?,
    purchaseDate : String?,
    vehicleModelDate : String?,
    vehicleConfiguration : String?,
    vehicleSeatingCapacity : Uint16?,
    vehicleMaxCapacity : Uint16?,
    vehicleSpecialUsage : String?,
    vehicleExteriorColor : String?,
    vehicleInteriorColor : String?,
    vehicleInteriorType : String?,
    knownVehicleDamages : String?,
    optionalExtras : String[]
};
```

4.3.5. サービスコールの詳細規定

getConfig

VehicleIdの構成情報の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
config	VehicleIdConfigType	VehicleIdの構成情報

【エラーコード】

なし

【機能】

VehicleIdの構成情報を返す。

4.4. Vehicle.Specification（略称：VehicleSpec）

4.4.1. 位置付けと機能

Vehicle.Specification（略称：VehicleSpec）は，車両の諸元を参照するためのシングルトンオブジェクトである。



COVESA VSS 6.0でVehicleの直下にある車両の諸元に関するデータを，Vehicle.Specificationで参照できるようにした。各構成情報の詳細は，COVESA VSSの仕様書に記載されている。

4.4.2. 構成情報

VehicleSpecは，以下の構成情報を持つ。

curbWeight	Uint16?	整備重量（潤滑油や冷却水に加えて燃料を全量入れた状態で，乗員や貨物を含まない状態での車両重量。単位：kg）
grossWeight	Uint16?	車両総重量（潤滑油や冷却水に加えて燃料を全量入れた状態で，乗員や貨物を満載した状態での車両重量。単位：kg）
maxTowWeight	Uint16?	最大牽引重量（トレーラーの最大重量。単位：kg）
maxTowBallWeight	Uint16?	トレーラーのトウボールにかけられる最大垂直荷重（単位：kg）
length	Uint16?	車両全長（単位：mm）
height	Uint16?	車両全高（単位：mm）
widthExcludingMirrors	Uint16?	SAE J1100-2009 W144で定義されるミラーを含んだ車両全幅（単位：mm）
widthFoldedMirrors	Uint16?	SAE J1100-2009 W145で定義されるミラーを折りたたんだ状態での車両全幅（単位：mm）
turningDiameter	Uint16?	SAE J1100-2009 D102で定義される最小回転直径（単位：mm）。最小回転半径の2倍
roofLoad	Int16?	車両上部の貨物および設置物の許容総重量（単位：kg）
cargoVolume	Float32?	貨物や荷物に利用できる容積（単位：l）
emissionsCO2	Int16?	CO2排出量（単位：g/km）

4.4.3. サービスコール一覧

VehicleSpecに対するサービスコールは以下の通り。

getConfig	VehicleSpecの構成情報の参照

4.4.4. データ型の定義

4.4.4.1. VehicleSpecの構成情報（getConfigが返すデータ型）

```
type VehicleSpecConfigType = struct {
    curbWeight : Uint16?,
    grossWeight : Uint16?,
    maxTowWeight : Uint16?,
    maxTowBallWeight : Uint16?,
    length : Uint16?,
    height : Uint16?,
    widthExcludingMirrors : Uint16?,
```

```
    widthFoldedMirrors : Uint16?,
    turningDiameter : Uint16?,
    roofLoad : Int16?,
    cargoVolume : Float32?,
    emissionsCO2 : Int16?
};
```

4.4.5. サービスコールの詳細規定

getConfig

VehicleSpecの構成情報の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
config	VehicleSpecConfigType	VehicleSpecの構成情報

【エラーコード】

なし

【機能】

VersionSpecの構成情報を返す。

4.5. Vehicle.Cabin（略称：Cabin）

＜TBD＞

4.6. Vehicle.Cabin.Door（略称：Door）

4.6.1. 位置付けと機能

Vehicle.Cabin.Door（略称：Door）は，車両のドア（客室）を操作するためのマルチインスタンスオブジェクトである。

4.6.1.1. ドアの位置付けと性質

開閉できる車室の開口部の中で，車両の走行中は通常閉じておくもの。客室にアクセスでき人が通ることは想定している。

Doorの機能

Doorは，可動部を一次元に移動できるオブジェクトであり，MovableObjectの機能を持つ。

可動部の位置は，いずれの方向の移動に対しても，どれだけ開いているかのパーセンテージ（開閉度，0：全閉，100：全開）で表す。また，0に近づく動作を閉動作，100に近づく動作を開動作と呼ぶ。

挟み込み防止機能を持つドアは，可動部の閉動作中に挟み込みを検知すると，閉動作を停止し，開動作を開始する。挟み込み検知後にどの位置まで開くかは，車両依存である。

可動部が停止中の間も，マニュアル操作にて現在位置が変化することがあり得る。

4.6.1.2. ドアに対する制御の調停

ドアに対する制御の調停には，標準的な調停機能（後勝ちの原則）が適用される。すなわち，ドアを閉めるサービスコールが呼び出され，ドアが閉じる動作中に，開くサービスコールが呼び出されると，最初のサービスコールによる制御はキャンセルされ，ドアは開く動作を始める。

また，ドアは標準的なロック機能を持つLockableObjectである。ドアには，機械的なロック機構は無いのが一般的であり，ソフトウェアでロック機能を実現することを想定している。

ドアが故障または挟み込みを検知した場合，ロックは解除され，故障中と挟み込み回避中は，ロックを取得することができない。

4.6.2. 機能クラス

Doorは，以下の機能クラスを持つ。

Movable	パワードアであり，APIにより移動させることができる。ドアロックアンラッチ機能のみを有する者も含む
ClosedDetectable	ドアが全閉状態であることを検出することができる
MoveProfileSupported	移動プロファイルをサポートしている

4.6.3. リスククラス

Doorは，以下のリスククラスを持つ。

RiskPinch	ドアを閉めることに伴う挟み込みのリスク
RiskOpen	ドアを開けることに伴う諸々のリスク

【今後の検討課題】

RiskOpenをさらに分類するかは，今後の課題である。

4.6.4. 構成情報

Doorの各インスタンスは，以下の構成情報を持つ。

instanceId	IdType	インスタンスが持つId
placement	PlacementType	インスタンスがある場所
capabilities	enumSet (DoorCapabilityType)	そのドアが持つ機能
riskClasses	enumSet(DoorRiskType)	そのドアのリスククラスの集合

mountedOn	RelatedObjectType	設置対象物（ドアまたは車室）
displayName	String	ユーザに表示する呼称を取得するためのキー

上記は，YAML形式で記述した構成記述ファイル中に記述する形式である。構成情報を参照するサービスコールは，この構成情報に対して，次の変換を行った構成情報を返す。

- mountedOnは，物理APIで定められたデータ型に変換して返す。
- displayNameKeyに対しては，それを元に現在のロケールに対応する呼称の文字列を取得し，その文字列を返す。

4.6.5. 状態

Doorの各インスタンスは，以下の状態を持つ。

mainStatus	DoorMainStatusType	主状態
lockStatus	LockStatusType	ロック状態
position	PositionType	現在位置
targetPosition	PositionType	目標位置

ドアの主状態（mainStatus）は，MovableObjectの主状態に加えて，挟み込み回避中の状態をとる。また，位置のパーセンテージが大きくなる方へ移動中を開動作中，位置のパーセンテージが小さくなる方へ移動中を閉動作中と呼ぶ。

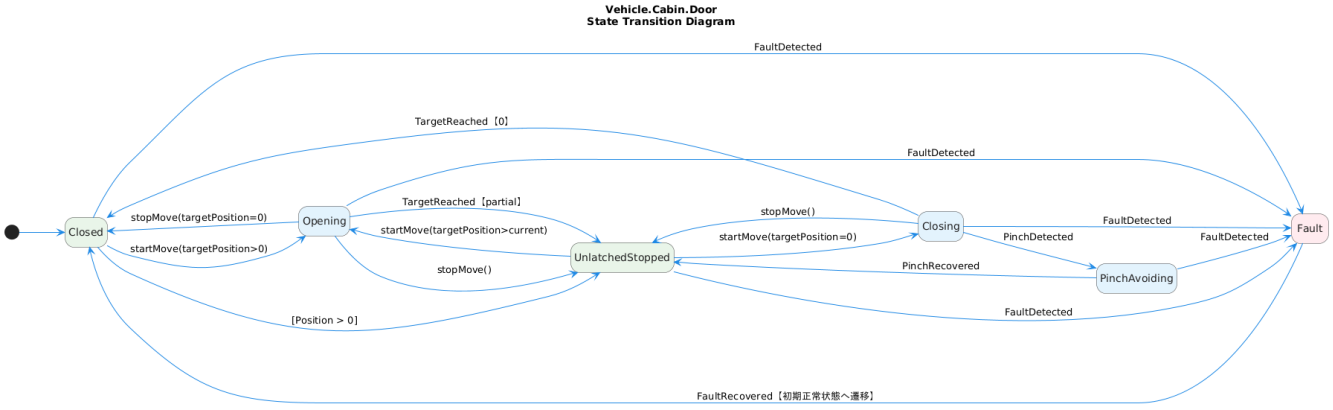


Figure 10. ドアの主状態の状態遷移

ドアの現在位置（position）と目標位置（targetPosition）は，MovableObjectの現在位置と目標位置に関する規定に従う。

ドアロック/アンラッチのみの機能の場合は，開要求のみを受け付け，閉要求は受け付けないものとする。また，アンラッチ完了時をもって目標位置到達と考える。

4.6.6. サービスコール一覧

Doorに対するサービスコールは以下の通り。

getConfigAll	すべてのドアの構成情報の参照
getStatus	ドアの状態の参照
startMove	ドアの開閉の開始

stopMove	ドアの開閉の停止
lock	ドアのロックの取得
unlock	ドアのロックの解除
notify	ドアイベントの通知要求
unnotify	ドアイベントの通知停止
getEvent	ドアイベントの取得

4.6.7. イベント

ドアに対するイベント種別には，MovableObjectの持つイベントとLockableObjectが持つイベントに加えて，挟み込み検知と挟み込みから回復がある。

Doorに対するイベント種別は以下の通り。

ControlledBySelf	自アプリケーションが制御
ControlledByOther	他のアプリケーションが制御
TargetReached	指定された目標位置に到達して停止，または，移動停止が完了
FaultDetected	故障検知
FaultRecovered	故障から回復
PinchDetected	挟み込み検知
PinchRecovered	挟み込みから回復
OwnLockRevoked	自アプリケーションのロックの強制解除
OtherLockReleased	他のアプリケーション等によるロック解除
EventOverflow	イベントキューのオーバフロー
ConfigChanged	構成情報の変更

この内，イベント種別がControlledByOtherの場合には，以下の関連情報を持つ。

sourceApplication	ApplicationIdType	操作したアプリケーション

また，イベント種別がEventOverflowの場合には，以下の関連情報を持つ。

noLostEvents	UInt32	失われたイベントの数

4.6.8. データ型の定義

4.6.8.1. ドアの機能クラス

```
type DoorCapabilityType = enum {
    Movable,
    ClosedDetectable,
```

```
    MoveProfileSupported
};
```

4.6.8.2. ドアに対するリスククラス

```
type DoorRiskType = enum {
    RiskPinch,
    RiskOpen
};
```

4.6.8.3. ドアの構成情報（getConfigAllが返すデータ型）

```
type DoorConfigType = struct {
    instanceId : IdType,
    placement : PlacementType,
    capabilities : enumSet(DoorCapabilityType),
    riskClasses : enumSet(DoorRiskType),
    mountedOn : RelatedObjectType,
    displayName : String
};
```

4.6.8.4. ドア状態

```
type DoorMainStatusType = enum {
    FullyStopped,   // 停止中（全閉）
    UnlatchedStopped,   // 停止中（半ドア　or　ドア開）
    Opening,    // 開動作中
    Closing,    // 閉動作中
    Fault,  // 故障中
    PinchAvoiding   // 挟み込み回避中
};
```

```
type DoorStatusType = struct {
    mainStatus : DoorMainStatusType,
    lockStatus : LockStatusType,
    position : PositionType,
    targetPosition : PositionType
};
```

4.6.8.5. ドアの移動プロファイル

```
type DoorMoveProfileType = enum {
    Standard,   // 標準的な移動
    Fast,   // 高速で移動
```

```
    Slow    // 低速で移動
};
```

4.6.8.6. ドアイベント

```
type DoorEventKindType = enum {
    ControlledBySelf,
    ControlledByOther,
    TargetReached,
    FaultDetected,
    FaultRecovered,
    PinchDetected,
    PinchRecovered,
    OwnLockRevoked,
    OtherLockReleased,
    EventOverflow,
    ConfigChanged
};
```

```
type DoorEventType = struct {
    instanceId : IdType,
    eventInfo : variant(DoorEventKindType) {
        ControlledByOther {
            sourceApplication : ApplicationIdType
        },
        EventOverflow {
            noLostEvents : UInt32
        }
    }
};
```

4.6.9. サービスコールの詳細規定

getConfigAll

すべてのドアの構成情報の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
config	DoorConfigType[]	すべてのドアに関する構成情報

【エラーコード】

なし

【機能】

すべてのドアに関する構成情報を返す。

getStatus

ドアの状態の参照

【パラメータ】

instanceId	IdType	操作対象のドアのID

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
status	DoorStatusType	対象ドアの状態

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外

【機能】

操作対象のドアに関するすべての状態を返す。

startMove

ドアの開閉の開始

【パラメータ】

instanceId	IdType	操作対象のドアのID
targetPosition	PositionType	目標位置（開閉度）
moveProfile	DoorMoveProfileType?	移動プロファイル
priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_INVALID_PARAMETER	targetPosition，priorityが有効範囲外
E_RISK_APPLICATION	アプリケーションが対応していないリスクがある

E_RISK_USER	ユーザがリスクを承認していない
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_FAULT	操作対象のドアが故障中
E_OBJECT_STATUS	操作対象のドアが挟み込み回避中
E_OBJECT_LOCKED	操作対象のドアが，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている

【機能】

目標位置をtargetPositionに設定し，操作対象のドアの開閉動作を開始する。

targetPositionに0を指定すれば閉動作の開始，100を指定すれば開動作の開始の意味になる。中間の値を設定した場合は,現在位置と比較し作動方向を決定する。

アンラッチ機能のみを有するドアの場合は，targetPositionに0より大きい値にて要求された場合,アンラッチ動作を実施。（完全なドア開は保証できない）

リスククラスRiskPinchに対応できていない場合でも，ドアを開く方向に動作を開始しようとした場合には，操作を受け付ける。そうでない場合には，E_RISK_APPLICATIONエラーまたはE_RISK_USERエラーとなる。

リスククラスRiskOpenに対応できていない場合でも，ドアを閉じる方向に動作を開始しようとした場合には，操作を受け付ける。そうでない場合には，E_RISK_APPLICATIONエラーまたはE_RISK_USERエラーとなる。

DoorMoveProfileTypeは，標準，高速，低速を選択可能であるが，制御対象の性能差により絶対速度の保証は困難である。具体的な速度に関しては，各制御対象ごとで決定することを想定している。

stopMove

ドアの開閉の停止

【パラメータ】

instanceId	IdType	操作対象のドアのID
priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_INVALID_PARAMETER	priorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_FAULT	操作対象のドアが故障中
E_OBJECT_STATUS	操作対象のドアが挟み込み回避中

E_OBJECT_LOCKED	操作対象のドアが，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている

【機能】

各次元方向の目標位置を現在位置に設定して，操作対象のドアの各次元方向への開閉動作の停止を要求する。

lock

ドアのロックの取得

【パラメータ】

instanceId	IdType	操作対象のドアのID
priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_INVALID_PARAMETER	priorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_LOCKED	ロック対象のドアが，priorityと同じかそれより高い優先度でロックされている
E_OBJECT_FAULT	ロック対象のドアが故障中
E_OBJECT_STATUS	ロック対象のドアが挟み込み回避中

【機能】

ロック対象のドアを，指定した操作優先度でロックする。

unlock

ドアのロックの解除

【パラメータ】

instanceId	IdType	操作対象のドアのID

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外

E_OBJECT_STATUS	自アプリケーションが，操作対象のドアをロックしていない

【機能】

ロック解除対象のドアを，ロック解除する。

notify

ドアイベントの通知要求

【パラメータ】

instanceId	IdType?	イベント通知対象のドアのID
eventFilter	enumSet (DoorEventKindType)?	通知するイベントの種類の集合

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_INVALID_PARAMETER	eventFilterが有効範囲外

【機能】

イベント通知対象のドアで発生したeventFilterで指定したイベントの通知を要求する。イベントを通知するためのイベントキューを生成し，その通知ハンドルを notifyHandle に返す。

instanceIdを省略した場合，すべてのインスタンスで発生したイベントの通知を要求する。eventFilterを省略した場合，すべての種類のイベントの通知を要求する。

unnotify

ドアイベントの通知停止

【パラメータ】

notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正

【機能】

notifyHandleで指定した通知を停止する。指定した通知用のイベントキューは削除される。

getEvent

ドアイベントの取得

【パラメータ】

notifyHandle	NotifyHandleType	イベントキューを識別するための通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
event	DoorEventType	取得したイベント

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正
E_NO_DATA	イベントキューが空

【機能】

notifyHandleで指定した通知用のイベントキューから，先頭のイベントを取り出す。取り出したイベントは，イベントキューから削除される。



イベントがパラメータ等で渡されてくるプログラミング環境向けの物理APIでは，用意する必要がない。

4.7. Vehicle.Cabin.DoorSwitch（略称：DoorSwitch）

4.7.1. 位置付けと機能

Doorを制御する物理的なスイッチを扱う。スイッチとDoorは直結せず，制御はOS側（ビークルOS）で行うことを仮定する。

4.7.1.1. Doorスイッチの位置付けと性質

Doorスイッチとは，各Doorに配置された乗員が物理操作できるDoor制御デバイスである。

スイッチはレバー形および2段階式である。

Doorスイッチは各Doorに配置され独立した系統を持つ。

スイッチとDoorは直接接続されず，制御要求はビークルOSが受け取る。

スイッチは入力源として機能し，Door制御の主体にはならない。

4.7.1.2. Doorスイッチの機能

Doorスイッチは乗員の操作によるスイッチ入力をOSへ提供し，Doorオブジェクトへ制御要求を伝達する役割を持つ。

各スイッチの系統はconfigData内のswitchCategoryによって識別される。

4.7.1.3. Doorスイッチに対する制御の調停

Doorスイッチは入力イベントの発生源であり，自身が制御競合を調停することはない。

入力はOSが一元管理し，Doorオブジェクトへルーティングされる。

調停はDoor側の後勝ち原則に従う。スイッチ操作が他アプリケーションの制御要求と競合した場合，Door側の調停ロジックにより採用または却下が判断される。

スイッチ自身は制御主体ではなく，アプリケーションや自動制御と同様に「入力の一つ」として扱われる。

4.7.2. 機能クラス

DoorSwitchは，以下の機能クラスを持つ。

SwitchControl	Door操作入力を提供する基本機能

4.7.3. 構成情報

DoorSwitchの各インスタンスは，以下の構成情報を持つ。

instanceId	IdType	インスタンスが持つId
placement	PlacementType	インスタンスがある場所
switchCategory	DoorSwitchCategoryType	スイッチ系統（FrontRight　FrontLeft　RearRight　RearLeft）
capabilities	enumSet (DoorSwitchCapabilityType)	このスイッチがサポートする操作機能の集合
switchType	String	スイッチ種別（OPEN側のみ，CLOSE側のみ，IsDoorChildLockEngagedの機能ON/OFFのスイッチかなど，OEM依存）
displayName	String	ユーザに表示する呼称を取得するためのキー

上記は，YAML形式で記述した構成記述ファイル中に記述する形式である。構成情報を参照するサービスコールは，この構成情報に対して，次の変換を行った構成情報を返す。

- displayNameKeyに対しては，それを元に現在のロケールに対応する呼称の文字列を取得し，その文字列を返す。

4.7.4. 状態

DoorSwitchの各インスタンスは，以下の状態を持つ。

mainStatus	DoorSwitchMainStatusType	スイッチの入力状態（OPEN側のみ，CLOSE側のみ，IsDoorChildLockEngagedの機能ON/OFFのスイッチかなど）

4.7.5. サービスコール一覧

DoorSwitchに対するサービスコールは以下の通り。

getConfigAll	すべてのDoorスイッチの構成情報の参照
getStatus	Doorスイッチの状態参照
notify	Doorスイッチイベントの通知要求
unnotify	Doorスイッチイベントの通知停止
getEvent	Doorスイッチイベントの取得

4.7.6. イベント

Doorスイッチに関するイベントとして，スイッチ状態変化，故障検出/回復，構成変更，イベントキューのオーバーフローなどを定義する。

DoorSwitchに対するイベント種別は以下の通り。

MainStatusChanged	各ドアに対応するスイッチのmainStatus入力状態が変化
FaultDetected	故障検知
FaultRecovered	故障から回復
EventOverflow	イベントキューのオーバフロー
ConfigChanged	構成情報の変更

この内，イベント種別がMainStatusChangedの場合には，以下の関連情報を持つ。

newState	DoorSwitchMainStatusType	スイッチの新しい入力状態

また，イベント種別がEventOverflowの場合には，以下の関連情報を持つ。

noLostEvents	UInt32	失われたイベントの数

4.7.7. データ型の定義

4.7.7.1. Doorスイッチの機能クラス

```
type DoorSwitchCapabilityType = enum {
```

```
    SwitchControl
};
```

4.7.7.2. スイッチ系統

```
type DoorSwitchCategoryType = enum {
    FrontRight, // FrontRightDoorスイッチ系統
    FrontLeft,  // FrontLeftDoorスイッチ系統
    RearRight,  // RearRightDoorスイッチ系統
    RearLeft    // RearLeftDoorスイッチ系統
};
```

4.7.7.3. Doorsスイッチの入力状態

```
type DoorSwitchMainStatusType = enum {
    False,  // OFF - Door停止
    OPEN,   // OPEN - DoorOPEN
    CLOSE,  // Close - DoorClose
    ChildLockON,    // IsDoorChildLockEngagedの機能ON
    ChildLockOFF    // IsDoorChildLockEngagedの機能OFF
};
```

Doorスイッチ構成情報（getConfigAll が返すデータ型）

```
type DoorSwitchConfigType = struct {
    instanceId : IdType,
    placement : PlacementType,
    switchCategory : DoorSwitchCategoryType,
    capabilities : enumSet(DoorSwitchCapabilityType),
    switchType : String,
    displayName : String
};
```

4.7.7.4. Doorスイッチ状態

```
type DoorSwitchStatusType = struct {
    mainStatus : DoorSwitchMainStatusType
};
```

Doorスイッチの状態情報

4.7.7.5. Doorスイッチイベント種別

```
type DoorSwitchEventKindType = enum {
```

```
    MainStatusChanged,
    FaultDetected,
    FaultRecovered,
    EventOverflow,
    ConfigChanged
};
```

```
type DoorSwitchEventType = struct {
    instanceId : IdType,
    switchCategory : DoorSwitchCategoryType,
    eventInfo : variant(DoorSwitchEventKindType) {
        MainStatusChanged {
            newState : DoorSwitchMainStatusType
        },
        EventOverflow {
            noLostEvents : UInt32
        }
    }
};
```

4.7.8. サービスコールの詳細規定

getConfigAll

すべてのDoorスイッチの構成情報の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
config	DoorSwitchConfigType[]	すべてのDoorスイッチに関する構成情報

【機能】

すべてのDoorスイッチに関する構成情報を返す。

getStatus

Doorスイッチの状態参照

【パラメータ】

instanceId	IdType	操作対象のDoorスイッチのID

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
status	DoorSwitchStatusType	対象のDoorスイッチの状態

【エラーコード】

E_INVALID_ID	instanceId が有効範囲外

【機能】

操作対象のDoorスイッチに関する状態を返す。



status には mainStatus（各Door系統のスイッチ入力状態）を含む。

notify

Doorスイッチイベントの通知要求

【パラメータ】

instanceId	IdType?	イベント通知対象のDoorスイッチID（省略時は全インスタンス）
eventFilter	enumSet (DoorSwitchEventKindType)?	通知するイベント種別の集合（省略時は全種）

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
notifyHandle	NotifyHandleType	通知ハンドル

【エラーコード】

E_INVALID_ID	instanceId が有効範囲外
E_INVALID_PARAMETER	eventFilter が有効範囲外

【機能】

イベント通知対象のDoorスイッチで発生したeventFilterで指定したイベントの通知を要求する。イベントを通知するためのイベントキューを生成し，その通知ハンドルをnotifyHandleに返す。

instanceIdを省略した場合，すべてのインスタンスで発生したイベントの通知を要求する。eventFilterを省略した場合，すべての種類のイベントの通知を要求する。

unnotify

Doorスイッチイベントの通知停止

【パラメータ】

notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandle が不正

【機能】

notifyHandleで指定した通知を停止する。指定した通知用のイベントキューは削除される。

getEvent

Doorスイッチイベントの取得

【パラメータ】

notifyHandle	NotifyHandleType	イベントキューを識別する通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
event	DoorSwitchEventType	取得したイベント

【エラーコード】

E_INVALID_HANDLE	notifyHandle が不正
E_NO_DATA	イベントキューが空

【機能】

notifyHandleで指定した通知用のイベントキューから，先頭のイベントを取り出す。取り出したイベントは，イベントキューから削除される。



イベントがパラメータ等で渡されてくるプログラミング環境向けの物理APIでは，用意する必要がない。

4.8. Vehicle.Cabin.InteriorLight（略称：InteriorLight）

4.8.1. 位置付けと機能

Vehicle.Cabin.InteriorLight（略称:　InteriorLight）は，車両の車室内に取り付けられた照明装置を扱うマルチインスタンスオブジェクトである。乗員の視認性向上のため，点灯・消灯，輝度，色，エフェクト（効果）などを制御・参照する。

4.8.1.1. 室内ライトの位置付けと性質

室内ライトとは，車両の車室内に取り付けられた照明装置であり，乗員が車内の状況を確認しやすくすることで視認性の向上を図るものである。

複数の室内ライトを個別に扱うことを前提としたマルチインスタンスオブジェクトである。

対象となるライトには，アンビエントライト，インタラクティブライトバー，ドームライト，グローブボックスライトなどが含まれる。

これらは特に乗降時や夜間走行時に重要な役割を果たし，車内環境の安全性および快適性の向上に寄与する。

また，必要に応じて足元照明やドアカーテシランプといったオプションの室内ライトを追加することも可能である。

4.8.1.2. 室内ライトの機能

室内ライトは，車両内部に設置された照明装置の点灯および消灯を制御する機能を有する。

これに加え，輝度をパーセンテージで設定する機能，RGBカラーコードによる色彩設定機能，さらに文字列によって指定されるライトエフェクトの設定機能を備える。

エフェクト機能の内容はOEM依存であり，主にインタラクティブライトバーなどの特定のライト種別で有効となる。

また，環境光量を参照可能なライトも存在し，周囲の明るさを考慮した制御が可能である。

これらの機能は，輝度制御，色制御，エフェクト制御，環境光参照といった機能クラスとして整理されている。

さらに，現在の輝度，色，エフェクト，環境光量といった状態情報の参照や，構成情報の取得，各種設定変更，ならびに制御状態や故障検知，効果開始・終了などのイベント通知を行う機能を備えている。

4.8.1.3. 機能クラスと有効なサービスコールの対応

各インスタンスで有効となるサービスコールおよびパラメータは，構成情報 capabilities に含まれる機能クラスによって決まる。対応は以下の通りである。

Dimmable ： setIntensity が有効。指定できる値は構成情報 intensityRange（min / max / step）に従う。

ColorControllable : setColor が有効。指定できる値は構成情報 supportedColors に従う。

EffectControllable : setEffect が有効。指定できる値は構成情報 supportedEffects に列挙された識別子に限る。

AmbientAware : 状態 ambientLightLevel が有効。本機能クラスを持たないライトの ambientLightLevel は参照してはならない。

capabilities に含まれない機能クラスに対応するサービスコールは，E_NOT_SUPPORTED を返す。

lightOn，lightOff，getConfigAll，getStatus，getErrorString，notify，unnotify，getEvent は，機能クラスに関わらず有効である。

4.8.1.4. 設定値の保持，点灯時の復元，および実効値の参照

輝度，色，エフェクトの設定値は，点灯状態とは独立に保持される。setIntensity，setColor，setEffect

は，消灯中に呼び出した場合も点灯させず，設定値のみを更新する。更新された設定値は，次回の lightOn で適用される。

lightOn は，直前に保持している輝度・色・エフェクトの設定値で点灯する。構成情報 retainsSettingsWhenOff が false のライトは，消灯時に設定値を車両既定値へ戻すため，lightOn は車両既定の輝度・色で点灯する。設定値を確定させたい場合は，lightOn の前に setIntensity / setColor を呼び出す。

要求値がハードウェアで実現できない場合（段階式の輝度，色数が限られたLEDなど），そのライトが実現可能な最近傍の値へ丸める。値域そのものを外れる要求は丸めずに E_INVALID_PARAMETER を返す。

丸めや近似の結果として実際に適用された値（実効値）は，getStatus が返す状態で参照する。すなわち，要求値と getStatus が返す値は一致しないことがある。ガンマ補正など，指定値から実際の発光までの変換は車両側で行い，その結果を実効値として返す。

4.8.1.5. 制御要求の完了判定

点灯・消灯および輝度・色の変更には時間がかかると想定している（フェード動作を含む）。そのため，サービスコールの戻り値 E_OK は要求の受付を示すものであり，動作の完了を示すものではない。

要求した設定値に到達した時点で TargetReached が発生する。関連情報 sourceApplication が自アプリケーションである TargetReached を受け取った場合に，自アプリケーションの要求が完了したと判断する。

エフェクトを伴う制御では，エフェクトの開始時に EffectStart，終了時に EffectEnd が発生する。

4.8.1.6. 室内ライトに対する制御の調停

室内ライトに対する制御の調停には，標準的な調停機能（後勝ちの原則）が適用され，後から発行された制御要求が優先される性質を持つ。

すなわち，室内ライトを点灯するサービスコールが呼び出され，室内ライト点灯中に，室内ライトを消灯するサービスコールが呼び出されると，最初のサービスコールによる制御はキャンセルされ，室内ライトは消灯する。

他のアプリケーションまたは標準制御によって制御要求が上書きされた場合，ControlledByOther が発生する。物理スイッチ（InteriorLightSwitch）による操作も「入力の一つ」として同じ調停に従う。

4.8.2. 機能クラス

InteriorLightは，以下の機能クラスを持つ。

Dimmable	輝度（Intensity）を制御できる。setIntensity が有効。指定可能範囲は intensityRange に従う
ColorControllable	色（Color）を制御できる（#000000〜#FFFFFF）。setColor が有効。指定可能な色は supportedColors に従う
EffectControllable	エフェクト（effect）識別子による効果設定ができる（OEM依存）。setEffect が有効。指定可能な識別子は supportedEffects に従う
AmbientAware	環境光量（ambientLightLevel）を参照できる

4.8.3. リスククラス

InteriorLightは，以下のリスククラスを持つ。

LightTempOver	ライト温度上限超過 : 温度上昇に伴う発火リスクとする
LightBrightnessOver	高輝度異常 : 運転中の視界不良に繋がるリスク
LightBrightnessUnder	低輝度異常 : 運転中の視界不良に繋がるリスク
LightColorIllegal	色味異常 : 法規に繋がるリスク
LightBlinkIllegal	点滅異常 : 運転中の視界不良に繋がるリスク
LightBatteryLongOn	長時間点灯によるバッテリ上がり : バッテリ上がりはライト以外でも該当するので，共通化するのがよいかも
LightPrivacyVisible	室内が視認されやすい : プライバシー観点ではあり

4.8.4. 構成情報

InteriorLightの各インスタンスは，以下の構成情報を持つ。

instanceId	IdType	インスタンスが持つId
placement	PlacementType	インスタンスがある場所
lightType	InteriorLightType	ライト種別（AmbientLight, InteractiveLightBar, DomeLight, GloveBoxLight など）
capabilities	enumSet (InteriorLightCapabilityType)	そのライトが持つ機能
riskClasses	enumSet (InteriorLightRiskType)	そのライトのリスククラスの集合
intensityRange	IntensityRangeType	そのライトが受け付ける輝度の範囲と分解能（min / max / step）。 Dimmable を持たないライトでは min = max = 100，step = 0 とし，setIntensity は E_NOT_SUPPORTED を返す。
supportedColors	ColorType[]	そのライトが設定できる色の集合。空配列の場合は #000000〜#FFFFFF の全域を連続的に指定できることを示す。 要素が列挙されている場合，setColor に指定した色は，この集合の中で最も近い色へ近似される。 ColorControllable を持たないライトでは空配列とし，setColor は E_NOT_SUPPORTED を返す。

supportedEffects	String[]	そのライトが設定できるライトエフェクト識別子の集合（識別子そのものはOEM依存）。 setEffect には，この集合に含まれる識別子のみ指定できる。含まれない識別子を指定した場合はE_NOT_SUPPORTED を返す。 EffectControllable を持たないライトでは空配列とする。
retainsSettingsWhenOff	Bool	消灯中も輝度・色・エフェクトの設定値を保持するか否か。 true の場合，lightOn は消灯前の設定値で点灯する。false の場合，消灯時に設定値は車両既定値へ戻り，lightOn は車両既定の輝度・色で点灯する。
mountedOn	RelatedObjectType	関連するオブジェクト（設置対象）
displayName	String	ユーザに表示する呼称を取得するためのキー

4.8.5. 状態

InteriorLightの各インスタンスは，以下の状態を持つ。

mainStatus	InteriorLightMainStatusType	主状態（消灯中/点灯中/故障中）
intensity	IntensityType	輝度（%）。1%=最大減衰（最も暗い），100%=減衰なし（最も明るい）。 点灯中は実際に適用されている実効値，消灯中は次回点灯時に適用される設定値を返す。
color	ColorType	色彩（#000000～#FFFFFF）。 点灯中は実際に適用されている実効値（supportedColors による近似後の値），消灯中は次回点灯時に適用される設定値を返す。
effect	String	エフェクト識別子（OEM依存）。構成情報 supportedEffects のいずれか，またはエフェクト未適用を示す空文字列。
ambientLightLevel	PercentageType	環境光量（%）。0%=環境光なし，100%=最大の明るさ。AmbientAware を持つライトでのみ有効
eventEnable	Bool	イベント通知の許可/不許可参照

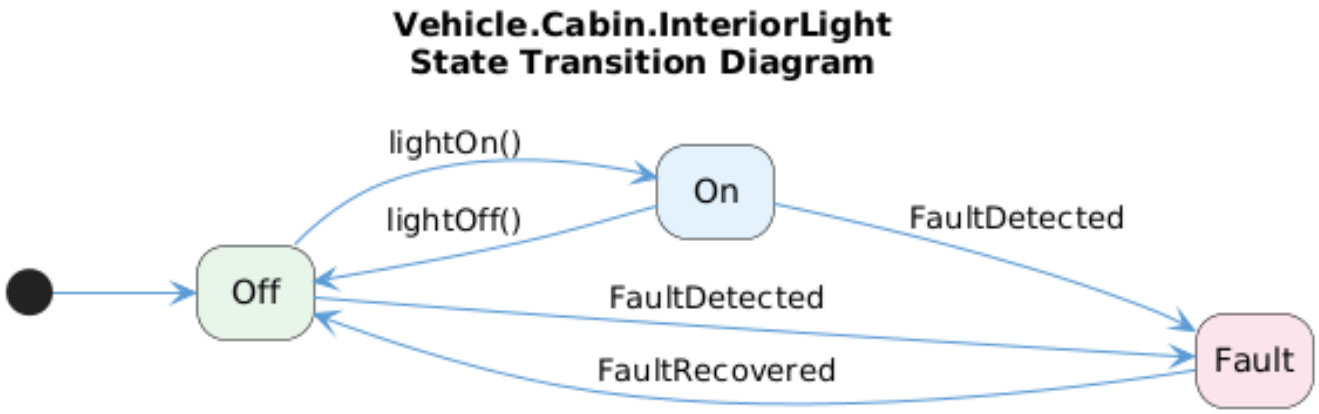


Figure 11. 室内ライトの主状態の状態遷移

4.8.6. サービスコール一覧

InteriorLightに対するサービスコールは以下の通り。

getConfigAll	すべての室内ライトの構成情報の参照
getStatus	室内ライトの状態の参照
getErrorString	エラー情報取得
lightOn	点灯
lightOff	消灯
setIntensity	輝度設定
setColor	色彩設定
setEffect	室内ライトエフェクト設定
notify	室内ライトイベントの通知要求
unnotify	室内ライトイベントの通知停止
getEvent	室内ライトイベントの取得

4.8.7. イベント

室内ライトに関するイベントとして，制御主体やターゲット到達，効果開始/終了，故障検出/回復，構成変更，イベントキューのオーバーフローなどを定義する。InteractiveLightBar　などの一部機能はエフェクト設定時のみ有効。

InteriorLightに対するイベント種別は以下の通り。

ControlledBySelf	自アプリケーションが制御。 lightOn/lightOff/setIntensity/setColor/setEffect が E_OK で完了した時に発生する
ControlledByOther	他のアプリケーションが制御
TargetReached	指定された設定値（点灯/消灯，輝度，色味）に到達。 丸めや近似が行われた場合は，実効値に到達した時点で発生する。
FaultDetected	故障検知。以降，制御要求は E_OBJECT_FAULT を返す

FaultRecovered	故障から回復
EffectStart	ライトエフェクト開始。EffectControllable を持つライトで，エフェクトが適用された時に発生する
EffectEnd	ライトエフェクト終了。エフェクトの解除，消灯，または他のエフェクトへの切り替えにより発生する
EventOverflow	イベントキューのオーバフロー
ConfigChanged	構成情報の変更

この内，イベント種別がControlledByOtherの場合には，以下の関連情報を持つ。

sourceApplication	ApplicationIdType	操作したアプリケーション

また，イベント種別がTargetReachedの場合には，以下の関連情報を持つ。

sourceApplication	ApplicationIdType	到達した要求を発行したアプリケーション

また，イベント種別がEventOverflowの場合には，以下の関連情報を持つ。

noLostEvents	UInt32	失われたイベントの数

4.8.8. データ型の定義

4.8.8.1. 輝度・色・パーセンテージのデータ型

```
type IntensityType = UInt8 ([1,100]);   // 輝度をパーセンテージで表す。1=最大減衰（最も暗い），100=減衰なし（最も明るい）。
代表値: 1=最小輝度，50=中間輝度，100=最大輝度。
0 は消灯を意味しない無効値であり，指定した場合は E_INVALID_PARAMETER を返す。消灯には lightOff を使用する。
```

```
type ColorType = String;    // 色彩をRGBカラーコードで表す文字列。形式は "#RRGGBB"（各チャネル2桁の16進数，#000000～#FFFFFF）。
色空間は sRGB とし，ガンマ補正を含む発光までの変換は車両側で行う。
```

```
type PercentageType = UInt8 ([0,100]);  // 割合をパーセンテージで表す。0=最小，100=最大
```

```
type IntensityRangeType = struct {
    min : IntensityType,
    max : IntensityType,
    step : UInt8
```

```
};
```

4.8.8.2. 室内ライトの機能クラス

```
type InteriorLightCapabilityType = enum {
    Dimmable,
    ColorControllable,
    EffectControllable,
    AmbientAware
};
```

4.8.8.3. 室内ライトに対するリスククラス

```
type InteriorLightRiskType = enum {
    LightTempOver,
    LightBrightnessOver,
    LightBrightnessUnder,
    LightColorIllegal,
    LightBlinkIllegal,
    LightBatteryLongOn,
    LightPrivacyVisible
};
```

室内ライトの構成情報（getConfigAll が返すデータ型）

```
type InteriorLightConfigType = struct {
    instanceId : IdType,
    placement : PlacementType,
    lightType : InteriorLightType,
    capabilities : enumSet(InteriorLightCapabilityType),
    riskClasses : enumSet(InteriorLightRiskType),
    intensityRange : IntensityRangeType,
    supportedColors : ColorType[],
    supportedEffects : String[],
    retainsSettingsWhenOff : Bool,
    mountedOn : RelatedObjectType,
    displayName : String
};
```

4.8.8.4. 室内ライトの主状態

```
type InteriorLightMainStatusType = enum {
    Off,    // 消灯中。輝度・色は次回点灯時に適用される設定値を示す
    On, // 点灯中。輝度・色は実際に適用されている実効値を示す
    Fault   // 故障中。制御要求は受け付けられず E_OBJECT_FAULT を返す
```

```
};
```

4.8.8.5. 室内ライト状態

```
type InteriorLightStatusType = struct {
    mainStatus : InteriorLightMainStatusType,
    intensity : IntensityType,
    color : ColorType,
    effect : String,
    ambientLightLevel : PercentageType,
    eventEnable : Bool
};
```

4.8.8.6. 室内ライト種別

```
type InteriorLightType = enum {
    AmbientLight,   // アンビエントライト
    InteractiveLightBar,    // インタラクティブライトバー
    DomeLight,  // ドームライト
    GloveBoxLight   // グローブボックスライト
};
```

4.8.8.7. 室内ライトイベント種別

```
type InteriorLightEventKindType = enum {
    ControlledBySelf,
    ControlledByOther,
    TargetReached,
    FaultDetected,
    FaultRecovered,
    EffectStart,
    EffectEnd,
    EventOverflow,
    ConfigChanged
};
```

```
type InteriorLightEventType = struct {
    instanceId : IdType,
    eventInfo : variant(InteriorLightEventKindType) {
        ControlledByOther {
            sourceApplication : ApplicationIdType
        },
        TargetReached {
            sourceApplication : ApplicationIdType
        },
        EventOverflow {
```

```
            noLostEvents : UInt32
        }
    }
};
```

4.8.9. サービスコールの詳細規定

getConfigAll

すべての室内ライトの構成情報の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
config	InteriorLightConfigType[]	すべての室内ライトに関する構成情報

【エラーコード】

E_OK	成功
E_NOT_SUPPORTED	未サポート（実装により返却されない場合あり）

【機能】

すべての室内ライトに関する構成情報を返す。

アプリケーションは，制御を行う前に本サービスコールで capabilities，intensityRange，supportedColors，supportedEffects，retainsSettingsWhenOff を取得し，指定可能な値と有効なサービスコールを決定する。これらを固定値として実装してはならない。

getStatus

室内ライトの状態の参照

【パラメータ】

instanceId	IdType	操作対象の室内ライトのID

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
status	InteriorLightStatusType	対象の室内ライトの状態（mainStatus, intensity, color, effect, ambientLightLevel など）

【エラーコード】

E_INVALID_ID	instanceId が有効範囲外

【機能】

操作対象の室内ライトに関するすべての状態を返す。

intensity，color，effect　　は，丸めまたは近似が行われた後に実際に適用されている値（実効値）である。点灯中（mainStatus = On）は実効値，消灯中（mainStatus = Off）は次回点灯時に適用される設定値を返す。

要求値がそのまま保持されているとは限らないため，要求後の値の確認には本サービスコールを使用する。

getErrorString

エラー情報取得

【パラメータ】

instanceId	IdType	操作対象の室内ライトのID

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
errorStatus	String	故障状態を表す文字列（DTC リスト想定）

【機能】

操作対象の室内ライトに関するエラー情報を返す。



本APIはオプション扱い。詳細はOEM依存。

DTC文字列の形式が規定されていない点は，InteriorLightSwitch（4.9章）と共通の課題であり，仕様書全体で機械可読な型へ統一することが望ましい。

lightOn

点灯

【パラメータ】

instanceId	IdType	操作対象の室内ライトのID

【リターンパラメータ】

returnValue	ReturnType	E_OK またはエラー

【エラーコード】

E_INVALID_ID	instanceId が有効範囲外
E_OBJECT_FAULT	対象の室内ライトが故障中（mainStatus = Fault）

【機能】

保持している輝度・色・エフェクトの設定値で点灯させる。

構成情報 retainsSettingsWhenOff が true のライトは消灯前の設定値で点灯する。false のライトは車両既定の輝度・色で点灯する。設定値を確定させたい場合は，lightOn の前に setIntensity / setColor を呼び出す。

点灯中に呼び出した場合，状態は変化せず，ControlledBySelf と TargetReached がこの順序で発生する。

本サービスコールは点灯の開始を要求するものであり，戻り値 E_OK は要求の受付を示す。フェード動作を伴う場合など，設定値に到達するまでには時間がかかる。

lightOff

消灯

【パラメータ】

instanceId	IdType	操作対象の室内ライトのID

【リターンパラメータ】

returnValue	ReturnType	E_OK またはエラー

【エラーコード】

E_INVALID_ID	instanceId が有効範囲外
E_OBJECT_FAULT	対象の室内ライトが故障中（mainStatus = Fault）

【機能】

消灯させる。mainStatus は Off へ遷移する。

retainsSettingsWhenOff が true のライトは，輝度・色・エフェクトの設定値を保持する。false のライトは，設定値を車両既定値へ戻す。

エフェクト適用中に呼び出した場合，エフェクトは終了し EffectEnd が発生する。

消灯中に呼び出した場合，状態は変化せず，ControlledBySelf と TargetReached がこの順序で発生する。

setIntensity

輝度設定

【パラメータ】

instanceId	IdType	操作対象の室内ライトのID
intensity	IntensityType	設定する輝度（%）。1=最大減衰，100=減衰なし。0 は無効値

【リターンパラメータ】

returnValue	ReturnType	E_OK またはエラー

【エラーコード】

E_INVALID_ID	instanceId が有効範囲外
E_INVALID_PARAMETER	intensity が intensityRange の min〜max の範囲外（0 を含む）
E_NOT_SUPPORTED	capabilities に Dimmable を含まない
E_OBJECT_FAULT	対象の室内ライトが故障中（mainStatus = Fault）

【機能】

輝度（%）を設定する。設定値は，次回点灯時にも使用する要求輝度として保持される。

代表値の動作は次の通り。1 は最大減衰（そのライトで最も暗い点灯状態），50 は中間輝度，100 は減衰なし（そのライトで最も明るい点灯状態）を要求する。0 は無効値であり，E_INVALID_PARAMETER を返す。消灯には lightOff を使用する。

消灯中（mainStatus = Off）に呼び出した場合は点灯せず，設定値のみを更新する。

intensityRange の step が 0 でないライトでは，指定値は実現可能な最近傍の値へ丸められる。丸めた後の実効値は getStatus で参照する。

min〜max の範囲外の値は丸めずに E_INVALID_PARAMETER を返す。

setColor

色彩設定

【パラメータ】

instanceId	IdType	操作対象の室内ライトのID
color	ColorType	設定する色彩（「#RRGGBB」形式，sRGB）

【リターンパラメータ】

returnValue	ReturnType	E_OK またはエラー

【エラーコード】

E_INVALID_ID	instanceId が有効範囲外
E_INVALID_PARAMETER	color が "#RRGGBB" 形式でない
E_NOT_SUPPORTED	capabilities に ColorControllable を含まない
E_OBJECT_FAULT	対象の室内ライトが故障中（mainStatus = Fault）

【機能】

色彩（#000000〜#FFFFFF）を設定する。設定値は，次回点灯時にも使用する要求色として保持される。

色指定は sRGB 色空間の各チャネル8ビットとして解釈する。指定値から実際の発光までの変換（ガンマ補正，白色点の補正など）は車両側で行う。

構成情報 supportedColors に色が列挙されているライト（単色LED，限られた色数のLEDなど）では，指定色はその集合の中で最も近い色へ近似される。近似後の実効値は getStatus で参照する。

消灯中（mainStatus = Off）に呼び出した場合は点灯せず，設定値のみを更新する。

color に「#000000」を指定した場合の動作は消灯ではなく，そのライトが表現できる最も暗い色の指定である。消灯には lightOff を使用する。

setEffect

室内ライトエフェクト設定

【パラメータ】

instanceId	IdType	操作対象の室内ライトのID
effect	String	設定するライトエフェクト識別子（構成情報 supportedEffects のいずれか）

【リターンパラメータ】

returnValue	ReturnType	E_OK またはエラー

【エラーコード】

E_INVALID_ID	instanceId が有効範囲外
E_NOT_SUPPORTED	capabilities に EffectControllable を含まない，または effect が supportedEffects に含まれない
E_OBJECT_FAULT	対象の室内ライトが故障中（mainStatus = Fault）

【機能】

ライトエフェクトを識別子で設定する。指定できる識別子は，構成情報 supportedEffects から取得する。識別子そのものはOEM依存であり，本仕様では規定しない。

supportedEffects に含まれない識別子を指定した場合は E_NOT_SUPPORTED を返す。代替エフェクトへのフォールバックは行わない。

エフェクトは，capabilities に EffectControllable を含むインスタンス（主に InteractiveLightBar）でのみ有効である。含まないインスタンスは E_NOT_SUPPORTED を返す。

エフェクトの適用開始時に EffectStart，終了時に EffectEnd が発生する。エフェクト適用中に他のエフェクトを設定した場合は，EffectEnd の後に EffectStart が発生する。

空文字列を指定した場合は，エフェクトを解除し，輝度・色の設定値による点灯状態に戻す。

消灯中（mainStatus = Off）に呼び出した場合は点灯せず，設定値のみを更新する。

notify

室内ライトイベントの通知要求

【パラメータ】

instanceId	IdType?	イベント通知対象の室内ライトID（省略時は全インスタンス）
eventFilter	enumSet (InteriorLightEventKindType)?	通知するイベント種別の集合（省略時は全種）

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
notifyHandle	NotifyHandleType	通知ハンドル

【エラーコード】

E_INVALID_ID	instanceId が有効範囲外
E_INVALID_PARAMETER	eventFilter が有効範囲外

【機能】

イベント通知対象の室内ライトで発生したeventFilterで指定したイベントの通知を要求する。イベントを通知するためのイベントキューを生成し，その通知ハンドルをnotifyHandleに返す。

instanceIdを省略した場合，すべてのインスタンスで発生したイベントの通知を要求する。eventFilterを省略した場合，すべての種類のイベントの通知を要求する。

unnotify

室内ライトイベントの通知停止

【パラメータ】

notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandle が不正

【機能】

notifyHandleで指定した通知を停止する。指定した通知用のイベントキューは削除される。

getEvent

室内ライトイベントの取得

【パラメータ】

notifyHandle	NotifyHandleType	イベントキューを識別する通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
event	InteriorLightEventType	取得したイベント

【エラーコード】

E_INVALID_HANDLE	notifyHandle が不正
E_NO_DATA	イベントキューが空

【機能】

notifyHandleで指定した通知用のイベントキューから，先頭のイベントを取り出す。取り出したイベントは，イベントキューから削除される。



イベントはFIFOで取り出され，取り出し後はキューから削除される。

4.9. Vehicle.Cabin.InteriorLightSwitch（略称：InteriorLightSwitch）

4.9.1. 位置付けと機能

室内ライトを操作する物理的なスイッチを扱う。スイッチとライトは直結せず，制御はOS側（ビークルOS）で行うことを仮定する。点灯・消灯を操作するスイッチであり，操作方式（押している間のみ有効なモーメンタリ，押下ごとに反転するトグル，ドア開閉に連動するもの）の違いを switchType で表す。

4.9.1.1. 室内ライトスイッチの位置付けと性質

室内ライトスイッチとは，車両キャビン内に設置され，乗員が室内ライトの点灯・消灯を物理操作できる入力デバイスである。

スイッチとライトは直接接続されず，制御要求はビークルOSが受け取る。

スイッチは物理的な操作子を持つが，ライトの点灯状態は室内ライトオブジェクトが制御する。スイッチは入力源として機能し，ライト制御主体にはならない。

スイッチが操作対象とする室内ライトは，構成情報 relatedInteriorLight で示される。対応するライト種別は lightType で参照できる。

4.9.1.2. 室内ライトスイッチの機能

室内ライトスイッチは乗員の押下・操作による点灯／消灯入力（switchInput）をOSへ提供し，室内ライトオブジェクトへ制御要求を伝達する役割を持つ。

スイッチ種別（switchType）は操作方式を表し，入力が発生するタイミングと保持期間を決める。OSが実行するライト側のサービスコールは defaultAction で示される。

スイッチの状態はmainStatusとして参照でき，正常・停止・故障などの状態監視が可能である。

アプリケーションは，入力の変化を SwitchStateChanged イベントで受け取ることができる。

4.9.1.3. 物理操作，入力，および実行される室内ライトAPIの対応

物理操作から室内ライトの制御までの経路は，次の通りである。

(1) 乗員がスイッチを操作する。

(2) スイッチの入力状態 switchInput が LightOn または LightOff に変化し，SwitchStateChanged が発生する。

(3) OSが，構成情報 defaultAction に従って relatedInteriorLight に対する室内ライトのサービスコールを実行する。

switchType ごとの動作は以下の通りである。

MomentaryOn：押している間のみ switchInput が LightOn となり，離すと LightOff に戻る。押下で lightOn，解放で lightOff が実行される。

ToggleOnOff：押下ごとに switchInput が LightOn と LightOff の間で反転する。解放しても状態は保持される。反転後の値に応じて lightOn または lightOff が実行される。

DoorLinked：ドアの開閉に連動して switchInput が変化する（開で LightOn，閉で LightOff）。乗員の押下操作ではないため，押下方式に関する規定は適用されない。

接点のばたつきは，構成情報 debounceMs で示される時間で除去した後の状態のみが switchInput に反映される。debounceMs より短い変化に対して SwitchStateChanged は発生しない。

SwitchStateChanged は入力の変化のみを表し，ライトが実際に点灯・消灯したことを表さない。ライト側の動作完了は，InteriorLight（4.12章）の TargetReached および getStatus で判断する。

4.9.1.4. 室内ライトスイッチに対する制御の調停

室内ライトスイッチは入力イベントの発生源であり，自身が制御競合を調停することはない。

入力はOSが一元管理し，室内ライトオブジェクトへルーティングされる。

調停は室内ライト側の後勝ち原則に従う。スイッチ操作が他アプリケーションの制御要求と競合した場合，ライト側の調停ロジックにより採用または却下が判断される。

スイッチ自身は制御主体ではなく，アプリケーションや自動制御と同様に「入力の一つ」として扱われる。

そのため，スイッチ操作によるライトの制御は，室内ライト側では ControlledByOther として観測される。

4.9.2. 機能クラス

InteriorLightSwitchは，以下の機能クラスを持つ。

SwitchOnOffControl	室内ライトをスイッチ入力により点灯または消灯させる
SwitchEffectSelect	スイッチ入力により，関連する室内ライトのエフェクト（構成情報 effect）を選択する

4.9.3. 構成情報

InteriorLightSwitchの各インスタンスは，以下の構成情報を持つ。

instanceId	IdType	インスタンスが持つId
placement	PlacementType	インスタンスがある場所
switchType	InteriorLightSwitchType	スイッチ種別（操作方式）。MomentaryOn / ToggleOnOff / DoorLinked
lightType	InteriorLightType	対応するライト種別（AmbientLight, InteractiveLightBar, DomeLight, GloveBoxLight など）
capabilities	enumSet (InteriorLightSwitchCapabilityType)	このスイッチが扱うライト機能の集合
relatedInteriorLight	RelatedObjectType	このスイッチの操作対象となる室内ライト（ Vehicle.Cabin.InteriorLight のインスタンス）。 スイッチ入力に対してOSが制御要求を行う対象を示す。
defaultAction	InteriorLightSwitchActionType	スイッチ入力に対してOSが relatedInteriorLight へ実行する室内ライトのサービスコール。 アプリケーションは，この値により物理操作がどのライトAPIに変換されるかを判別する。
debounceMs	UInt16	接点のばたつきを除去する時間（ミリ秒）。この時間より短い入力変化は switchInput に反映されず，SwitchStateChanged も発生しない。 0 はデバウンス処理を行わない（またはハードウェアで完結している）ことを示す。
effect	String	このスイッチの操作で選択されるライトエフェクト識別子（OEM依存）。 capabilities に SwitchEffectSelect を含む場合のみ有効であり，relatedInteriorLight の構成情報 supportedEffects に含まれる識別子でなければならない。 エフェクト操作を行わないスイッチでは空文字列とする。
mountedOn	RelatedObjectType	関連するオブジェクト（設置対象）
displayName	String	ユーザに表示する呼称を取得するためのキー

4.9.4. 状態

InteriorLightSwitchの各インスタンスは，以下の状態を持つ。

switchInput	InteriorLightSwitchInputType	スイッチ入力（LightOn=点灯要求 / LightOff=消灯要求）。デバウンス処理後の値
mainStatus	InteriorLightSwitchMainStatusType	スイッチの主状態（正常/停止/故障/電気故障など）

4.9.5. サービスコール一覧

InteriorLightSwitchに対するサービスコールは以下の通り。

getConfigAll	すべての室内ライトスイッチの構成情報の参照
getStatus	スイッチの状態参照
getErrorString	エラー情報取得
notify	室内ライトスイッチイベントの通知要求
unnotify	室内ライトスイッチイベントの通知停止
getEvent	室内ライトスイッチイベントの取得

4.9.6. イベント

室内ライトスイッチに関するイベントとして，スイッチ入力状態の変化，故障検出/回復，構成変更，イベントキューのオーバーフローなどを定義する。

スイッチは制御主体ではないため，制御主体を表すイベント（ControlledBySelf / ControlledByOther）は持たない。制御主体および制御の完了は，室内ライト側のイベントで判断する。

InteriorLightSwitchに対するイベント種別は以下の通り。

SwitchStateChanged	スイッチ入力状態（switchInput）が変化。デバウンス処理後の変化に対して発生する。 本イベントは入力の変化を表すものであり，室内ライトの動作完了を表さない。
FaultDetected	故障検知
FaultRecovered	故障から回復
EventOverflow	イベントキューのオーバフロー
ConfigChanged	構成情報の変更

この内，イベント種別がSwitchStateChangedの場合には，以下の関連情報を持つ。

previousState	InteriorLightSwitchInputType	変化前のスイッチ入力状態
newState	InteriorLightSwitchInputType	変化後のスイッチ入力状態

また，イベント種別がEventOverflowの場合には，以下の関連情報を持つ。

noLostEvents	UInt32	失われたイベントの数

4.9.7. データ型の定義

4.9.7.1. 室内ライトスイッチの機能クラス

```
type InteriorLightSwitchCapabilityType = enum {
    SwitchOnOffControl,
    SwitchEffectSelect
};
```

4.9.7.2. スイッチ状態

```
type InteriorLightSwitchMainStatusType = enum {
    Normal, // 正常状態
    Suspended,  // 停止状態（室内ライトスイッチでは通常想定しない）
    Fault,  // 故障状態（復帰しないことを想定）
    ElectricFault   // 電気故障（短絡/断線などのDTCを返す想定）
};
```

4.9.7.3. スイッチ入力種別

```
type InteriorLightSwitchInputType = enum {
    LightOn,    // 点灯要求。MomentaryOn では押下している間，ToggleOnOff では押下により反転した結果が点灯側である間，DoorLinked ではドアが開いている間，この値となる。
    LightOff    // 消灯要求。MomentaryOn では押下していない間，ToggleOnOff では押下により反転した結果が消灯側である間，DoorLinked ではドアが閉じている間，この値となる。
};
```

4.9.7.4. スイッチ種別（操作方式）

```
type InteriorLightSwitchType = enum {
    MomentaryOn,    // モーメンタリ式。押下している間のみ LightOn となり，解放で LightOff に戻る
    ToggleOnOff,    // オルタネート（トグル）式。押下ごとに LightOn と LightOff が反転し，解放後も保持される
    DoorLinked  // ドア連動式。ドアの開閉に連動して LightOn / LightOff が変化する（乗員の押下操作ではない）
};
```

4.9.7.5. スイッチ入力に対する動作種別

```
type InteriorLightSwitchActionType = enum {
    LightOnOff, // LightOn で lightOn(), LightOff で lightOff() を実行する
    LightOnOnly,    // LightOn で lightOn() を実行する。消灯はこのスイッチでは行わない
 (LightOff では何も実行しない)
    EffectApply // LightOn で lightOn() を実行した後，構成情報 effect で示す識別子で
setEffect() を実行する。LightOff で lightOff() を実行する
};
```

4.9.7.6. 室内ライト種別

```
type InteriorLightType = enum {
    AmbientLight,   // アンビエントライト
    InteractiveLightBar,    // インタラクティブライトバー
    DomeLight,  // ドームライト
    GloveBoxLight   // グローブボックスライト
};
```

4.9.7.7. 故障情報（DTC）

```
type InteriorLightSwitchDtcType = struct {
    dtcCode : UInt32,
    dtcText : String
};
```

室内ライトスイッチ構成情報（getConfigAll が返すデータ型）

```
type InteriorLightSwitchConfigType = struct {
    instanceId : IdType,
    placement : PlacementType,
    switchType : InteriorLightSwitchType,
    lightType : InteriorLightType,
    capabilities : enumSet(InteriorLightSwitchCapabilityType),
    relatedInteriorLight : RelatedObjectType,
    defaultAction : InteriorLightSwitchActionType,
    debounceMs : UInt16,
    effect : String,
    mountedOn : RelatedObjectType,
    displayName : String
};
```

4.9.7.8. 室内ライトスイッチ状態

```
type InteriorLightSwitchStatusType = struct {
    switchInput : InteriorLightSwitchInputType,
    mainStatus : InteriorLightSwitchMainStatusType
```

```
};
```

4.9.7.9. 室内ライトスイッチイベント種別

```
type InteriorLightSwitchEventKindType = enum {
    SwitchStateChanged,
    FaultDetected,
    FaultRecovered,
    EventOverflow,
    ConfigChanged
};
```

```
type InteriorLightSwitchEventType = struct {
    instanceId : IdType,
    eventInfo : variant(InteriorLightSwitchEventKindType) {
        SwitchStateChanged {
            previousState : InteriorLightSwitchInputType ,
            newState : InteriorLightSwitchInputType
        },
        EventOverflow {
            noLostEvents : UInt32
        }
    }
};
```

4.9.8. サービスコールの詳細規定

getConfigAll

すべての室内ライトスイッチの構成情報の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
config	InteriorLightSwitchConfigType[]	静的な構成値の参照。

【機能】

すべての室内ライトスイッチに関する構成情報を返す。

アプリケーションは，本サービスコールで switchType，relatedInteriorLight，defaultAction，debounceMs を取得し，どのスイッチがどの室内ライトをどのように操作するかを判別する。これらを固定値として実装してはならない。

getStatus

スイッチの状態参照

【パラメータ】

instanceId	IdType	操作対象の室内ライトスイッチのID

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
status	InteriorLightSwitchStatusType	対象の室内ライトスイッチの状態

【エラーコード】

E_INVALID_ID	instanceId が有効範囲外

【機能】

操作対象の室内ライトスイッチに関するすべての状態を返す。

switchInput はデバウンス処理後の入力状態であり，室内ライトの点灯状態ではない。ライトの点灯状態は InteriorLight の getStatus で参照する。



status には switchInput（点灯/消灯要求）と mainStatus を含む。

getErrorString

エラー情報取得

【パラメータ】

instanceId	IdType	操作対象の室内ライトスイッチのID

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
errorStatus	InteriorLightSwitchDtcType[]	故障状態を表すDTCのリスト。故障がない場合は空配列

【機能】

操作対象の室内ライトスイッチに関するエラー情報を返す。

mainStatus が ElectricFault の場合は，短絡・断線などのDTCが返る。



本APIはオプション扱い。DTCコードの割り当てはOEM依存。

notify

室内ライトスイッチイベントの通知要求

【パラメータ】

instanceId	IdType?	イベント通知対象の室内ライトスイッチID（省略時は全インスタンス）
eventFilter	enumSet (InteriorLightSwitchEventKindType)?	通知するイベント種別の集合（省略時は全種）

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
notifyHandle	NotifyHandleType	通知ハンドル

【エラーコード】

E_INVALID_ID	instanceId が有効範囲外
E_INVALID_PARAMETER	eventFilter が有効範囲外

【機能】

イベント通知対象の室内ライトスイッチで発生したeventFilterで指定したイベントの通知を要求する。イベントを通知するためのイベントキューを生成し，その通知ハンドルをnotifyHandleに返す。

instanceIdを省略した場合，すべてのインスタンスで発生したイベントの通知を要求する。eventFilterを省略した場合，すべての種類のイベントの通知を要求する。

unnotify

室内ライトスイッチイベントの通知停止

【パラメータ】

notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandle が不正

【機能】

notifyHandleで指定した通知を停止する。指定した通知用のイベントキューは削除される。

getEvent

室内ライトスイッチイベントの取得

【パラメータ】

notifyHandle	NotifyHandleType	イベントキューを識別する通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
event	InteriorLightSwitchEventType	取得したイベント

【エラーコード】

E_INVALID_HANDLE	notifyHandle が不正
E_NO_DATA	イベントキューが空

【機能】

notifyHandleで指定した通知用のイベントキューから，先頭のイベントを取り出す。取り出したイベントは，イベントキューから削除される。



イベントはFIFOで取り出され，取り出し後はキューから削除される。

4.10. Vehicle.Cabin.InteriorMirror（略称：InteriorMirror）

4.10.1. 位置付けと機能

Vehicle.Cabin.InteriorMirror（略称：InteriorMirror）は，車室内の防眩ミラーを操作するためのマルチインスタンスオブジェクトである。

4.10.1.1. 室内ミラーの位置付けと性質

室内ミラーとは，車両の乗員室内に装着されたミラーで，防眩機能（アンチグレア機能）を備えたものである。防眩機能により，夜間走行時にヘッドライトの光による輻射眩花を軽減することができる。

室内ミラーはルームミラーとして運転席の上部に装着されるのが一般的である。

室内ミラーの物理スイッチは，室内ミラーとは別のオブジェクトとして扱う。

4.10.1.2. InteriorMirrorの機能

InteriorMirrorは，防眩機能により鏡面の反射率を制御するオブジェクトであり，MovableObjectの特性を持つ。

防眩モード時は，防眩レベルを段階的に制御できる。防眩レベル（dimmingLevel）は0〜100のパーセンテージで表現される。0%は防眩なし（Undimmed　/　全反射・最も明るい状態），100%は最大防眩（Fully dimmed / 無反射・最も暗い状態）として扱う。

室内ミラーの初期位置，目標位置となる端点は固定であるため，室内ミラーの目標位置はAPIで指定，参照しない設計とする。

室内ミラーが故障を検知した場合，制御を受け付けない。

4.10.1.3. 室内ミラーに対する制御の調停

室内ミラーに対する制御の調停には，標準的な調停機能（後勝ちの原則）が適用される。すなわち，室内ミラーの制御サービスコールが呼び出された室内ミラーが動作中に，別の制御サービスコールが呼び出されると，最初のサービスコールによる制御はキャンセルされ，新しい制御が開始される。

また，室内ミラーは標準的なロック機能を持つLockableObjectである。室内ミラーのロック機能はソフトウェアで実現される。

室内ミラーが故障を検知した場合，ロックは解除され，故障中は，ロックを取得することができない。

4.10.2. 機能クラス

InteriorMirrorは，以下の機能クラスを持つ。

Movable	パワーミラーであり，APIにより防眩レベルを制御できる

4.10.3. リスククラス

InteriorMirrorは，以下のリスククラスを持つ。

RiskDimmingOnTrip	走行中に防眩機能が動作し，走行中に視界を妨げるリスク

4.10.4. 構成情報

InteriorMirrorの各インスタンスは，以下の構成情報を持つ。

instanceId	IdType	インスタンスが持つId
placement	PlacementType	車両内での室内ミラーの位置
capabilities	enumSet (InteriorMirrorCapabilityType)	その室内ミラーが持つ機能
riskClasses	enumSet (InteriorMirrorRiskType)	その室内ミラーのリスククラスの集合
mountedOn	RelatedObjectType	設置対象物（フロントウィンドウなど）
displayName	String	ユーザに表示する呼称を取得するためのキー

上記は，YAML形式で記述した構成記述ファイル中に記述する形式である。構成情報を参照するサービスコールは，この構成情報に対して，次の変換を行った構成情報を返す。

- mountedOnは，物理APIで定められたデータ型に変換して返す。
- displayNameKeyに対しては，それを元に現在のロケールに対応する呼称の文字列を取得し，その文字列を返す。

4.10.5. 状態

InteriorMirrorの各インスタンスは，以下の状態を持つ。

mainStatus	InteriorMirrorMainStatusType	主状態
lockStatus	LockStatusType	ロック状態
dimmingLevel	UInt32	現在の防眩レベル（0%：防眩なし/全反射，100%：最大防眩/無反射）

室内ミラーの主状態（mainStatus）は，各操作状態（停止中，防眩中）と故障状態をとる。

- **停止中（Stopped）**：デフォルト状態，またはstartMoveによって設定された目標防眩度に到達して停止している状態。この状態では室内ミラーは動作していない。
- **防眩中（Dimming）**：startMoveによって設定された目標防眩度に向かって防眩レベルを調整している状態。この状態では，指定された目標防眩レベルへの制御が進行中である。
- **故障中（Fault）**：室内ミラーが故障を検知した状態。この状態では制御を受け付けず，ロック機能も取得できない。

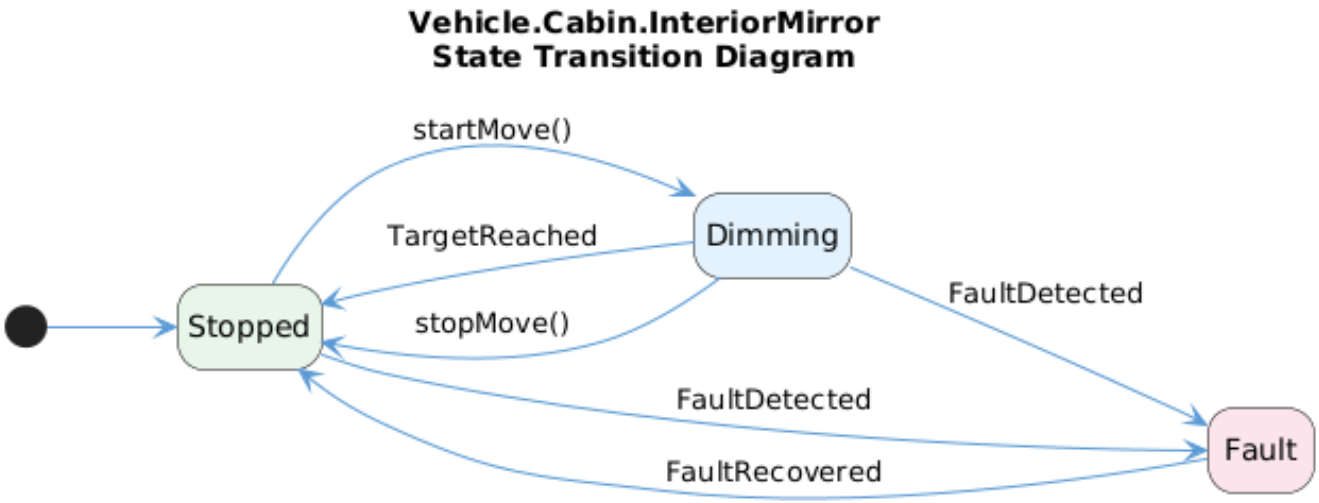


Figure 12. 室内ミラーの主状態の状態遷移

4.10.6. サービスコール一覧

InteriorMirrorに対するサービスコールは以下の通り。

getConfigAll	すべての室内ミラーの構成情報の参照
getStatus	室内ミラーの状態の参照
startMove	室内ミラーの防眩レベルの設定
stopMove	室内ミラーの防眩動作の停止
lock	室内ミラーのロックの取得
unlock	室内ミラーのロックの解除

notify	室内ミラーイベントの通知要求
unnotify	室内ミラーイベントの通知停止
getEvent	室内ミラーイベントの取得

4.10.7. イベント

室内ミラーに対するイベント種別には，MovableObjectの持つイベントとLockableObjectが持つイベントに加えて，故障検知がある。

InteriorMirrorに対するイベント種別は以下の通り。

ControlledBySelf	自アプリケーションが制御
ControlledByOther	他のアプリケーションが制御
TargetReached	指定された防眩レベルに到達して停止，または，動作停止が完了
FaultDetected	故障検知
FaultRecovered	故障から回復
OwnLockRevoked	自アプリケーションのロックの強制解除
OtherLockReleased	他のアプリケーション等によるロック解除
EventOverflow	イベントキューのオーバフロー
ConfigChanged	構成情報の変更

この内，イベント種別がControlledByOtherの場合には，以下の関連情報を持つ。

sourceApplication	ApplicationIdType	操作したアプリケーション

また，イベント種別がEventOverflowの場合には，以下の関連情報を持つ。

noLostEvents	UInt32	失われたイベントの数

4.10.8. データ型の定義

4.10.8.1. 室内ミラーの機能クラス

```
type InteriorMirrorCapabilityType = enum {
    Movable
};
```

4.10.8.2. 室内ミラーに対するリスククラス

```
type InteriorMirrorRiskType = enum {
    RiskDimmingOnTrip
```

```
};
```

4.10.8.3. 室内ミラーの構成情報（getConfigAllが返すデータ型）

```
type InteriorMirrorConfigType = struct {
    instanceId : IdType,
    placement : PlacementType,
    capabilities : enumSet(InteriorMirrorCapabilityType),
    riskClasses : enumSet(InteriorMirrorRiskType),
    mountedOn : RelatedObjectType,
    displayName : String
};
```

4.10.8.4. 室内ミラー状態

```
type InteriorMirrorMainStatusType = enum {
    Stopped,    // 停止中。デフォルト状態またはstartMoveによって設定された目標防眩度に
到達した停止状態。
この状態では室内ミラーは動作していない。
    Dimming,    // 防眩中。startMoveによって設定された目標防眩度へ防眩レベルを調整して
いる状態。
指定された目標防眩レベルへの制御が進行中である。
    Fault   // 故障中。室内ミラーが故障を検知した状態。
この状態では制御を受け付けず，ロック機能も取得できない。
};
```

```
type InteriorMirrorStatusType = struct {
    mainStatus : InteriorMirrorMainStatusType,
    lockStatus : LockStatusType,
    dimmingLevel : UInt32
};
```

室内ミラーの状態情報

4.10.8.5. 室内ミラーイベント

```
type InteriorMirrorEventKindType = enum {
    ControlledBySelf,
    ControlledByOther,
    TargetReached,
    FaultDetected,
    FaultRecovered,
    OwnLockRevoked,
    OtherLockReleased,
    EventOverflow,
    ConfigChanged
```

```
};
```

```
type InteriorMirrorEventType = struct {
    instanceId : IdType,
    eventInfo : variant(InteriorMirrorEventKindType) {
        ControlledByOther {
            sourceApplication : ApplicationIdType
        },
        EventOverflow {
            noLostEvents : UInt32
        }
    }
};
```

4.10.9. サービスコールの詳細規定

getConfigAll

すべての室内ミラーの構成情報の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
config	InteriorMirrorConfigType[]	すべての室内ミラーに関する構成情報

【エラーコード】

なし

【機能】

すべての室内ミラーに関する構成情報を返す。

getStatus

室内ミラーの状態の参照

【パラメータ】

instanceId	IdType	操作対象の室内ミラーのID

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

status	InteriorMirrorStatusType	対象室内ミラーの状態

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外

【機能】

操作対象の室内ミラーに関するすべての状態を返す。

startMove

室内ミラーの防眩レベルの設定

【パラメータ】

instanceId	IdType	操作対象の室内ミラーのID
dimmingLevel	UInt32	要求防眩レベル（0〜100%）。0は防眩なし（全反射），100は最大防眩（無反射）を示す。
priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_INVALID_PARAMETER	dimmingLevel，priorityが有効範囲外
E_RISK_APPLICATION	アプリケーションが対応していないリスクがある
E_RISK_USER	ユーザがリスクを承認していない
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_FAULT	操作対象の室内ミラーが故障中
E_OBJECT_LOCKED	操作対象の室内ミラーが，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている

【機能】

防眩レベルをdimmingLevelに設定し，操作対象の室内ミラーの防眩動作を開始する。

リスククラスRiskDimmingOnTripに対応できていない場合には，E_RISK_APPLICATIONエラーまたはE_RISK_USERエラーとなる。



無段階制御ではなく段階式のハードウェア（ON/OFFのみ，または数段階など）の場合，要求されたdimmingLevelはOSまたはドライバ層にてサポートする最も近い対応値へ丸められる。実際の適用値（実効値）はgetStatusによって取得可能である。

stopMove

室内ミラーの防眩動作の停止

【パラメータ】

instanceId	IdType	操作対象の室内ミラーのID
priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_INVALID_PARAMETER	priorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_FAULT	操作対象の室内ミラーが故障中
E_OBJECT_LOCKED	操作対象の室内ミラーが，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている

【機能】

操作対象の室内ミラーの防眩動作の停止を要求する。

lock

室内ミラーのロックの取得

【パラメータ】

instanceId	IdType	操作対象の室内ミラーのID
priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_INVALID_PARAMETER	priorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_LOCKED	ロック対象の室内ミラーが，priorityと同じかそれより高い優先度でロックされている
E_OBJECT_FAULT	ロック対象の室内ミラーが故障中

【機能】

ロック対象の室内ミラーを，指定した操作優先度でロックする。

unlock

室内ミラーのロックの解除

【パラメータ】

instanceId	IdType	操作対象の室内ミラーのID

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_OBJECT_STATUS	自アプリケーションが，操作対象の室内ミラーをロックしていない

【機能】

ロック解除対象の室内ミラーを，ロック解除する。

notify

室内ミラーイベントの通知要求

【パラメータ】

instanceId	IdType?	イベント通知対象の室内ミラーのID
eventFilter	enumSet (InteriorMirrorEventKindType)?	通知するイベントの種類の集合

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_INVALID_PARAMETER	eventFilterが有効範囲外

【機能】

イベント通知対象の室内ミラーで発生したeventFilterで指定したイベントの通知を要求する。イベントを通知するためのイベントキューを生成し，その通知ハンドルを notifyHandle に返す。

instanceIdを省略した場合，すべてのインスタンスで発生したイベントの通知を要求する。eventFilterを省略した場合，すべての種類のイベントの通知を要求する。

unnotify

室内ミラーイベントの通知停止

【パラメータ】

notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正

【機能】

notifyHandleで指定した通知を停止する。指定した通知用のイベントキューは削除される。

getEvent

室内ミラーイベントの取得

【パラメータ】

notifyHandle	NotifyHandleType	イベントキューを識別するための通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
event	InteriorMirrorEventType	取得したイベント

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正
E_NO_DATA	イベントキューが空

【機能】

notifyHandleで指定した通知用のイベントキューから，先頭のイベントを取り出す。取り出したイベントは，イベントキューから削除される。



イベントがパラメータ等で渡されてくるプログラミング環境向けの物理APIでは，用意する必要がない。

4.11. Vehicle.Cabin.InteriorMirrorSwitch（略称：InteriorMirrorSwitch）

4.11.1. 位置付けと機能

室内ミラーの防眩度を変更する物理的なスイッチを扱う。スイッチとミラーは直結せず，制御はOS側（ビークルOS）で行うことを仮定する。

4.11.1.1. 室内ミラー防眩度変更スイッチの位置付けと性質

室内ミラー防眩度変更スイッチとは，車両内に設置された，乗員が防眩度を物理操作できる入力デバイスである。

スイッチとミラーは直接接続されず，制御要求はビークルOSが受け取る。

スイッチは物理的な操作子を持つが，防眩機能の制御は室内ミラーオブジェクトが管理する。スイッチは入力源として機能し，ミラー制御主体にはならない。

スイッチの操作方式は2Way（ON/OFF），またはMultiWay（ON/OFF/LEVEL_INC/LEVEL_DEC）であり，どの方式に対応するかを構成情報として持つ。

4.11.1.2. 室内ミラー防眩度変更スイッチの機能

室内ミラー防眩度変更スイッチは乗員の操作による防眩度変更入力（ON/OFF/LEVEL_INC/LEVEL_DEC）をOSへ提供し，室内ミラーオブジェクトへ制御要求を伝達する役割を持つ。

スイッチの状態はmainStatusとして参照でき，現在の入力状態の監視が可能である。

capabilitiesで対応する機能範囲を示す。



本スイッチAPIはスイッチの物理的な入力状態を取得するものであり，短押しによる段階的（ステップ）な変化や，長押しによる連続的な変化の解釈，および対象のInteriorMirrorインスタンスへの反映はOSまたはアプリケーション層の責務とする。
LEVEL_INC（Increment）は防眩度を強める（暗くする：dimmingLevelを増加），LEVEL_DEC（Decrement）は防眩度を弱める（明るくする：dimmingLevelを減少）方向への制御要求として扱う。

4.11.1.3. 室内ミラー防眩度変更スイッチに対する制御の調停

室内ミラー防眩度変更スイッチは入力イベントの発生源であり，自身が制御競合を調停することはない。

入力はOSが一元管理し，室内ミラーオブジェクトへルーティングされる。

調停は室内ミラー側の後勝ち原則に従う。スイッチ操作が他アプリケーションの制御要求と競合した場合，ミラー側の調停ロジックにより採用または却下が判断される。

スイッチ自身は制御主体ではなく，アプリケーションや自動制御と同様に「入力の一つ」として扱われる。

4.11.2. 機能クラス

InteriorMirrorSwitchは，以下の機能クラスを持つ。

SwitchControl	スイッチ入力による防眩度変更機能

4.11.3. 構成情報

InteriorMirrorSwitchの各インスタンスは，以下の構成情報を持つ。

instanceId	IdType	インスタンスが持つId
placement	PlacementType	インスタンスがある場所
switchType	InteriorMirrorSwitchType	スイッチ操作方式（2Way/3Way）
capabilities	enumSet (InteriorMirrorSwitchCapabilityType)	このスイッチが扱う防眩機能の集合
displayName	String	ユーザに表示する呼称を取得するためのキー

上記は，YAML形式で記述した構成記述ファイル中に記述する形式である。構成情報を参照するサービスコールは，この構成情報に対して，次の変換を行った構成情報を返す。

- displayNameKeyに対しては，それを元に現在のロケールに対応する呼称の文字列を取得し，その文字列を返す。

4.11.4. 状態

InteriorMirrorSwitchの各インスタンスは，以下の状態を持つ。

mainStatus	InteriorMirrorSwitchMainStatusType	スイッチの入力状態（OFF/ON/LEVEL_INC/LEVEL_DEC）

4.11.5. サービスコール一覧

InteriorMirrorSwitchに対するサービスコールは以下の通り。

getConfigAll	すべての室内ミラー防眩度変更スイッチの構成情報の参照
getStatus	室内ミラー防眩度変更スイッチの状態参照
notify	室内ミラー防眩度変更スイッチイベントの通知要求
unnotify	室内ミラー防眩度変更スイッチイベントの通知停止
getEvent	室内ミラー防眩度変更スイッチイベントの取得

4.11.6. イベント

室内ミラー防眩度変更スイッチに関するイベントとして，スイッチ状態変化，故障検出/回復，構成変更，イベントキューのオーバーフローなどを定義する。

InteriorMirrorSwitchに対するイベント種別は以下の通り。

SwitchStateChanged	スイッチ入力状態が変化

FaultDetected	故障検知
FaultRecovered	故障から回復
EventOverflow	イベントキューのオーバフロー
ConfigChanged	構成情報の変更

この内，イベント種別がSwitchStateChangedの場合には，以下の関連情報を持つ。

newState	InteriorMirrorSwitchMainStatusType	スイッチの新しい入力状態

また，イベント種別がEventOverflowの場合には，以下の関連情報を持つ。

noLostEvents	UInt32	失われたイベントの数

4.11.7. データ型の定義

4.11.7.1. 室内ミラー防眩度変更スイッチの機能クラス

```
type InteriorMirrorSwitchCapabilityType = enum {
    SwitchControl
};
```

4.11.7.2. スイッチ操作方式

```
type InteriorMirrorSwitchType = enum {
    TwoWay, // 2Way操作（OFF/ON）
    MultiWay    // 多Way操作（OFF/ON/LEVEL_INC/LEVEL_DECなど）
};
```

4.11.7.3. スイッチ入力状態

```
type InteriorMirrorSwitchMainStatusType = enum {
    False,  // OFF - 防眩モードの停止，または防眩なし（0%）への変更要求
    True,   // ON - 防眩モードの有効化，または最大防眩（100%）への変更要求
    LevelIncrement, // LEVEL_INC - 防眩度を強める（暗くする：dimmingLevel増加）要求
    LevelDecrement  // LEVEL_DEC - 防眩度を弱める（明るくする：dimmingLevel減少）要求
};
```

室内ミラー防眩度変更スイッチ構成情報（getConfigAll が返すデータ型）

```
type InteriorMirrorSwitchConfigType = struct {
    instanceId : IdType,
    placement : PlacementType,
    switchType : InteriorMirrorSwitchType,
```

```
    capabilities : enumSet(InteriorMirrorSwitchCapabilityType),
    displayName : String
};
```

4.11.7.4. 室内ミラー防眩度変更スイッチ状態

```
type InteriorMirrorSwitchStatusType = struct {
    mainStatus : InteriorMirrorSwitchMainStatusType
};
```

室内ミラー防眩度変更スイッチの状態情報

4.11.7.5. 室内ミラー防眩度変更スイッチイベント種別

```
type InteriorMirrorSwitchEventKindType = enum {
    SwitchStateChanged,
    FaultDetected,
    FaultRecovered,
    EventOverflow,
    ConfigChanged
};
```

```
type InteriorMirrorSwitchEventType = struct {
    instanceId : IdType,
    eventInfo : variant(InteriorMirrorSwitchEventKindType) {
        SwitchStateChanged {
            newState : InteriorMirrorSwitchMainStatusType
        },
        EventOverflow {
            noLostEvents : UInt32
        }
    }
};
```

4.11.8. サービスコールの詳細規定

getConfigAll

すべての室内ミラー防眩度変更スイッチの構成情報の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

config	InteriorMirrorSwitchConfigType[]	すべての室内ミラー防眩度変更スイッチに関する構成情報

【機能】

すべての室内ミラー防眩度変更スイッチに関する構成情報を返す。

getStatus

室内ミラー防眩度変更スイッチの状態参照

【パラメータ】

instanceId	IdType	操作対象の室内ミラー防眩度変更スイッチのID

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
status	InteriorMirrorSwitchStatusType	対象の室内ミラー防眩度変更スイッチの状態

【エラーコード】

E_INVALID_ID	instanceId が有効範囲外

【機能】

操作対象の室内ミラー防眩度変更スイッチに関する状態を返す。



status には mainStatus（スイッチ入力状態）を含む。

notify

室内ミラー防眩度変更スイッチイベントの通知要求

【パラメータ】

instanceId	IdType?	イベント通知対象の室内ミラー防眩度変更スイッチID（省略時は全インスタンス）
eventFilter	enumSet (InteriorMirrorSwitchEventKindType)?	通知するイベント種別の集合（省略時は全種）

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
notifyHandle	NotifyHandleType	通知ハンドル

【エラーコード】

E_INVALID_ID	instanceId が有効範囲外
E_INVALID_PARAMETER	eventFilter が有効範囲外

【機能】

イベント通知対象の室内ミラー防眩度変更スイッチで発生したeventFilterで指定したイベントの通知を要求する。イベントを通知するためのイベントキューを生成し，その通知ハンドルをnotifyHandleに返す。

instanceIdを省略した場合，すべてのインスタンスで発生したイベントの通知を要求する。eventFilterを省略した場合，すべての種類のイベントの通知を要求する。

unnotify

室内ミラー防眩度変更スイッチイベントの通知停止

【パラメータ】

notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandle が不正

【機能】

notifyHandleで指定した通知を停止する。指定した通知用のイベントキューは削除される。

getEvent

室内ミラー防眩度変更スイッチイベントの取得

【パラメータ】

notifyHandle	NotifyHandleType	イベントキューを識別する通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
event	InteriorMirrorSwitchEventType	取得したイベント

【エラーコード】

E_INVALID_HANDLE	notifyHandle が不正
E_NO_DATA	イベントキューが空

【機能】

notifyHandleで指定した通知用のイベントキューから，先頭のイベントを取り出す。取り出したイベントは，イベントキューから削除される。



イベントがパラメータ等で渡されてくるプログラミング環境向けの物理APIでは，用意する必要がない。

4.12. Vehicle.Cabin.Seat（略称：Seat）

4.12.1. 位置付けと機能

Vehicle.Cabin.Seat（略称：Seat）は，車両のシートを操作するためのマルチインスタンスオブジェクトである。

4.12.1.1. シートの位置付けと性質

シートとは，人が座るための場所，複数の部位に分かれていると定義する。シートを開閉するための物理スイッチは，シートとは別のオブジェクトとして扱う。

Seatの機能

Seatは，可動部を一次元に移動できるオブジェクトであり，MovableObjectの機能を持つ。

可動部の位置は，いずれの方向の移動に対しても，どれだけ開いているかのパーセンテージ（開閉度，0：全閉，100：全開）で表す。また，0に近づく動作を閉動作，100に近づく動作を開動作と呼ぶ。

挟み込み防止機能を持つシートは，可動部の閉動作中に挟み込みを検知すると，閉動作を停止し，開動作を開始する。挟み込み検知後にどの位置まで開くかは，車両依存である。

可動部が停止中の間は，現在位置が変化することはない。可動部が強制的に移動されたなどの理由で現在位置が変化した場合には，シートが一時的に故障し，すぐに回復したものと扱う。

4.12.1.2. シートに対する制御の調停

シートに対する制御の調停には，標準的な調停機能（後勝ちの原則）が適用される。すなわち，シートを閉めるサービスコールが呼び出され，シートが閉じる動作中に，開くサービスコールが呼び出されると，最初のサービスコールによる制御はキャンセルされ，シートは開く動作を始める。

4.12.2. 機能クラス

Seatは，以下の機能クラスを持つ。

Movable	パワーシートであり，APIにより移動させることができる

4.12.3. リスククラス

Seatは，以下のリスククラスを持つ。

RiskPinch	シートを作動させることに伴う挟み込みのリスク
RiskDriveMove	運転中にシートを作動させることに伴うリスク

4.12.4. 構成情報

Seatの各インスタンスは，以下の構成情報を持つ。

instanceId	IdType	インスタンスが持つId
placement	PlacementType	インスタンスがある場所
capabilities	enumSet (SeatCapabilityType)	そのシートが持つ機能
riskClasses	enumSet(SeatRiskType)	そのシートのリスククラスの集合
mountedOn	RelatedObjectType	設置対象物（車室）
displayName	String	ユーザに表示する呼称を取得するためのキー

上記は，YAML形式で記述した構成記述ファイル中に記述する形式である。構成情報を参照するサービスコールは，この構成情報に対して，次の変換を行った構成情報を返す。

- mountedOnは，物理APIで定められたデータ型に変換して返す。
- displayNameKeyに対しては，それを元に現在のロケールに対応する呼称の文字列を取得し，その文字列を返す。

4.12.5. 状態

Seatの各インスタンスは，以下の状態を持つ。

mainStatus	SeatMainStatusType	主状態
lockStatus	LockStatusType	ロック状態
isOccupied	boolean	着座状況 True:着座　False:未着座
isBelted	boolean	シートベルト状況　True：装着　Flase：未装着
seatBeltHeight	SeatBeltHeightType	シートベルト高さ
massageLevel	MassageLevelType	マッサージ機能強さ　単位：%
heatingCooling	HeatingCoolingType	シート温度　単位：℃
seatPosition	structOf (Vehicle.Cabin.Seat.SeatPositionType)	シートの現在位置
targetPosition	structOf (Vehicle.Cabin.Seat.SeatPositionType)	目標シート位置

シートの主状態（mainStatus）は，MovableObjectの主状態に加えて，挟み込み回避中の状態をとる（[Fig-シートの主状態の状態遷移]）。

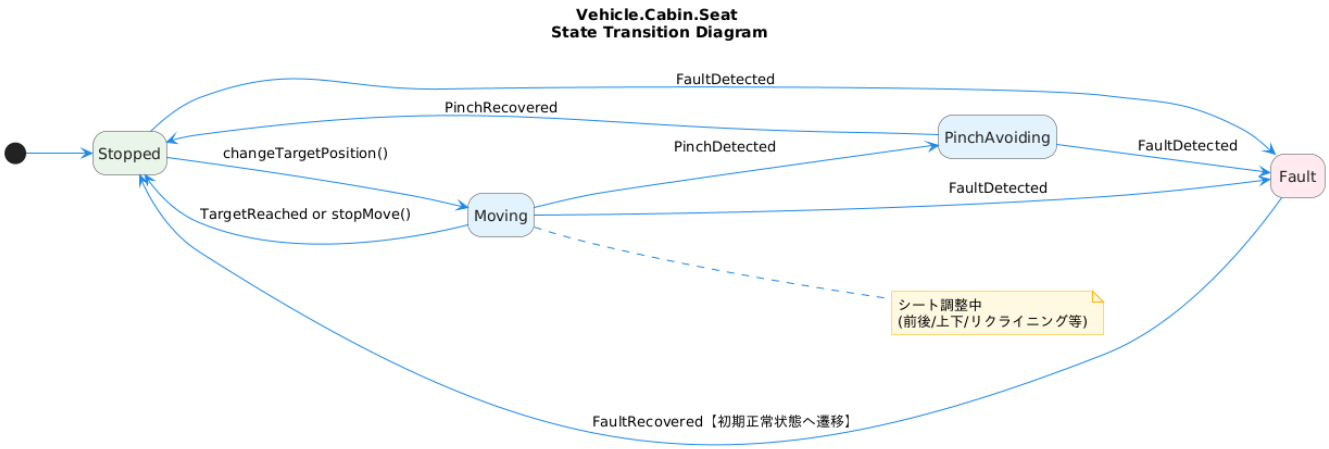


Figure 13. シートの主状態の状態遷移

シートの現在位置（seatPosition）と目標位置（targetPosition），MovableObjectの現在位置と目標位置に関する規定に従う。

4.12.6. サービスコール一覧

Seatに対するサービスコールは以下の通り。

getConfigAll	すべてのシートの構成情報の参照
getStatus	シートの状態の参照
changeSeatBeltHeight	シートベルト高さ変更
changeMassageLevel	マッサージ機能強さ変更
changeHeatingCooling	シート温度変更
changeTargetPosition	シート位置変更
stopMove	シートの開閉の停止
notify	シートイベントの通知要求
unnotify	シートイベントの通知停止
getEvent	シートイベントの取得

4.12.7. イベント

シートに対するイベント種別には，MovableObjectの持つイベントとLockableObjectが持つイベントに加えて，挟み込み検知と挟み込みから回復がある。

Seatに対するイベント種別は以下の通り。

ControlledBySelf	自アプリケーションが制御
ControlledByOther	他のアプリケーションが制御
TargetReached	指定された目標位置に到達して停止，または，移動停止が完了
FaultDetected	故障検知

FaultRecovered	故障から回復
PinchDetected	挟み込み検知
PinchRecovered	挟み込みから回復
OwnLockRevoked	自アプリケーションのロックの強制解除
OtherLockReleased	他のアプリケーション等によるロック解除
EventOverflow	イベントキューのオーバフロー
ConfigChanged	構成情報の変更

この内，イベント種別がControlledByOtherの場合には，以下の関連情報を持つ。

sourceApplication	ApplicationIdType	操作したアプリケーション

また，イベント種別がEventOverflowの場合には，以下の関連情報を持つ。

noLostEvents	UInt32	失われたイベントの数

4.12.8. データ型の定義

4.12.8.1. シートの機能クラス

```
type SeatCapabilityType = enum {
    Movable
};
```

4.12.8.2. シートに対するリスククラス

```
type SeatRiskType = enum {
    RiskPinch,
    RiskDriveMove
};
```

4.12.8.3. シートの構成情報（getConfigAllが返すデータ型）

```
type SeatConfigType = struct {
    instanceId : IdType,
    placement : PlacementType,
    capabilities : enumSet(SeatCapabilityType),
    riskClasses : enumSet(SeatRiskType),
    mountedOn : RelatedObjectType,
    displayName : String
};
```

4.12.8.4. シート状態

```
type SeatMainStatusType = enum {
    Stopped,    // 停止中
    Operating,  // シート動作中
    Fault,  // 故障中
    PinchAvoiding   // 挟み込み回避中
};
```

```
type SeatStatusType = struct {
    mainStatus : SeatMainStatusType,
    lockStatus : LockStatusType,
    isOccupied : boolean,
    isBelted : boolean,
    seatBeltHeight : SeatBeltHeightType,
    massageLevel : MassageLevelType,
    heatingCooling : HeatingCoolingType,
    seatPosition : structOf(Vehicle.Cabin.Seat.SeatPositionType),
    targetPosition : structOf(Vehicle.Cabin.Seat.SeatPositionType)
};
```

4.12.8.5. シートイベント

```
type SeatEventKindType = enum {
    ControlledBySelf,
    ControlledByOther,
    TargetReached,
    FaultDetected,
    FaultRecovered,
    PinchDetected,
    PinchRecovered,
    OwnLockRevoked,
    OtherLockReleased,
    EventOverflow,
    ConfigChanged
};
```

```
type SeatEventType = struct {
    instanceId : IdType,
    eventInfo : variant(SeatEventKindType) {
        ControlledByOther {
            sourceApplication : ApplicationIdType
        },
        EventOverflow {
            noLostEvents : UInt32
        }
    }
```

```
};
```

4.12.8.6. シート位置

```
type SeatPositionType = struct {
    posiotion : UInt32, // 前後位置[mm] 0=もっとも前
    height : UInt32,    // シート高さ[mm] 0=最も低い
    tile : UInt32,  // シート角度[°] 0=シートが最も倒れていない
    recline : UInt32,   // 背もたれ角度[°] 0=背もたれが最も倒れていない
    lumbarSupport : UInt32, //   ランバーサポートの左右（外内）[%] 0:最も内側 100:もっとも外側
    lumbarHeight : UInt32,  // ランバーサポートの高さ[%] 0=最も低い
    sideBolesterSupport : UInt32,   // サイドボルスターのサポート度合い[%]  0:最もサポートが弱い（外側に開く）100:最もサポートが強い（狭い）
    seatingLength : UInt32, // 座面の前後位置[mm] 0=最も後ろ位置
    headrestHeight : UInt32 // ヘッドレストの高さ[mm] 0=最も低い
};
```

4.12.9. サービスコールの詳細規定

getConfigAll

すべてのシートの構成情報の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
config	SeatConfigType[]	すべてのシートに関する構成情報

【エラーコード】

なし

【機能】

すべてのシートに関する構成情報を返す。

getStatus

シートの状態の参照

【パラメータ】

instanceId	IdType	操作対象のシートのID

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
status	SeatStatusType	対象シートの状態

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外

【機能】

操作対象のシートに関するすべての状態を返す。

changeSeatBeltHeight

シートベルト高さ変更

【パラメータ】

instanceId	IdType	操作対象のシートのID
targetSeatBeltHeight	SeatBeltHeight	目標シートベルト高さ
priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_INVALID_PARAMETER	targetSeatBeltHeight，priorityが有効範囲外
E_RISK_APPLICATION	アプリケーションが対応していないリスクがある
E_RISK_USER	ユーザがリスクを承認していない
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_FAULT	操作対象のシートが故障中
E_OBJECT_STATUS	操作対象のシートが挟み込み回避中
E_OBJECT_LOCKED	操作対象のシートが，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている

【機能】

目標シートベルト高さをtype:　SeatBeltHeightに設定し，操作対象のシートのシートベルト高さ調整を開始する。

SeatBeltHeightの単位はmmであり，0は車両として最も低い位置を示す。

changeMassageLevel

マッサージ機能強さ変更

【パラメータ】

instanceId	IdType	操作対象のシートのID
targetMassageLevel	massageLevel	目標マッサージ強度
priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_INVALID_PARAMETER	targetMassageLevel，priorityが有効範囲外
E_RISK_APPLICATION	アプリケーションが対応していないリスクがある
E_RISK_USER	ユーザがリスクを承認していない
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_FAULT	操作対象のシートが故障中
E_OBJECT_STATUS	操作対象のシートが挟み込み回避中
E_OBJECT_LOCKED	操作対象のシートが，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている

【機能】

目標シートベルト高さをtype:massageLevelに設定し，操作対象のシートのマッサージ機能強さ調整を行う。

massageLevelの単位は%であり，0はOFF，100は最も強いマッサージレベルを示す。

changeHeatingCooling

シート温度変更

【パラメータ】

instanceId	IdType	操作対象のシートのID
targetHeatingCoolingLevel	heatingCooling	目標シート温度
priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_INVALID_PARAMETER	targetHeatingCoolingLevel，priorityが有効範囲外
E_RISK_APPLICATION	アプリケーションが対応していないリスクがある
E_RISK_USER	ユーザがリスクを承認していない
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_FAULT	操作対象のシートが故障中
E_OBJECT_STATUS	操作対象のシートが挟み込み回避中
E_OBJECT_LOCKED	操作対象のシートが，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている

【機能】

目標シート温度をtype: heatingCoolingLevelに設定し，操作対象のシートの温度調整を行う。

heatingCoolingLevelの単位は℃であり，0～100の範囲としシートの設定温度を示す。

changeTargetPosition

シート位置変更

【パラメータ】

instanceId	IdType	操作対象のシートのID
targetPosition	structOf (Vehicle.Cabin.Seat.SeatPositionType)	目標シート位置 構造体
priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_INVALID_PARAMETER	targetPosition，priorityが有効範囲外
E_RISK_APPLICATION	アプリケーションが対応していないリスクがある
E_RISK_USER	ユーザがリスクを承認していない
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値

E_OBJECT_FAULT	操作対象のシートが故障中
E_OBJECT_STATUS	操作対象のシートが挟み込み回避中
E_OBJECT_LOCKED	操作対象のシートが，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている

【機能】

目標シートベルト高さを type: targetPositionに設定し，操作対象のシートの位置調整を行う。

位置を変更しない部位については変更前と同じ値を維持することを想定（値を 0 などに上書きしないこと）。

Seatpotisionと同じ要素をメンバーに持つ構造体を返す。

stopMove

シートの開閉の停止

【パラメータ】

instanceId	IdType	操作対象のシートのID
priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_INVALID_PARAMETER	priorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_FAULT	操作対象のシートが故障中
E_OBJECT_STATUS	操作対象のシートが挟み込み回避中
E_OBJECT_LOCKED	操作対象のシートが，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている

【機能】

各次元方向の目標位置を現在位置に設定して，操作対象のシートの各次元方向への開閉動作の停止を要求する。

notify

シートイベントの通知要求

【パラメータ】

instanceId	IdType?	イベント通知対象のシートのID

eventFilter	enumSet (SeatEventKindType)?	通知するイベントの種類の集合

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_INVALID_PARAMETER	eventFilterが有効範囲外

【機能】

イベント通知対象のシートで発生したeventFilterで指定したイベントの通知を要求する。イベントを通知するためのイベントキューを生成し，その通知ハンドルを notifyHandle に返す。

instanceIdを省略した場合，すべてのインスタンスで発生したイベントの通知を要求する。eventFilterを省略した場合，すべての種類のイベントの通知を要求する。

unnotify

シートイベントの通知停止

【パラメータ】

notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正

【機能】

notifyHandleで指定した通知を停止する。指定した通知用のイベントキューは削除される。

getEvent

シートイベントの取得

【パラメータ】

notifyHandle	NotifyHandleType	イベントキューを識別するための通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
event	SeatEventType	取得したイベント

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正
E_NO_DATA	イベントキューが空

【機能】

notifyHandleで指定した通知用のイベントキューから，先頭のイベントを取り出す。取り出したイベントは，イベントキューから削除される。



イベントがパラメータ等で渡されてくるプログラミング環境向けの物理APIでは，用意する必要がない。

4.13. Vehicle.Cabin.SeatSwitch（略称：SeatSwitch）

4.13.1. 位置付けと機能

Vehicle.Cabin.SeatSwitch（略称：SeatSwitch）シートを動作させる物理的なスイッチを扱う。すべてON/OFFの2-Wayスイッチとする。

4.13.1.1. Seatスイッチの位置付けと性質

Seatスイッチとは，各Seatに配置された乗員が物理操作できるSeat制御デバイスである。

スイッチはレバー形および2段階式である。

Seatスイッチは各Seatに配置され独立した系統を持つ。

スイッチとSeatは直接接続されず，制御要求はビークルOSが受け取る。

スイッチは入力源として機能し，Seat制御の主体にはならない。

4.13.1.2. Seatスイッチの機能

Seatスイッチは乗員の操作によるスイッチ入力をOSへ提供し，Seatオブジェクトへ制御要求を伝達する役割を持つ。

各スイッチの系統はconfigData内のswitchCategoryによって識別される。

4.13.1.3. Seatスイッチに対する制御の調停

Seatスイッチは入力イベントの発生源であり，自身が制御競合を調停することはない。

入力はOSが一元管理し，Seatオブジェクトへルーティングされる。

調停はSeat側の後勝ち原則に従う。スイッチ操作が他アプリケーションの制御要求と競合した場合，Seat

側の調停ロジックにより採用または却下が判断される。

スイッチ自身は制御主体ではなく，アプリケーションや自動制御と同様に「入力の一つ」として扱われる。

4.13.2. 機能クラス

SeatSwitchは，以下の機能クラスを持つ。

SwitchControl	Seat操作入力を提供する基本機能

4.13.3. 構成情報

SeatSwitchの各インスタンスは，以下の構成情報を持つ。

instanceId	IdType	インスタンスが持つId
placement	PlacementType	インスタンスがある場所
switchCategory	SeatSwitchCategoryType	スイッチ系統（FrontRight　FrontLeft　RearRight　RearLeft）
capabilities	enumSet (SeatSwitchCapabilityType)	このスイッチがサポートする操作機能の集合
switchType	String	スイッチ種別（UP側，DOWN側，Front側，Reverse側など，OEM依存）
displayName	String	ユーザに表示する呼称を取得するためのキー

上記は，YAML形式で記述した構成記述ファイル中に記述する形式である。構成情報を参照するサービスコールは，この構成情報に対して，次の変換を行った構成情報を返す。

- displayNameKeyに対しては，それを元に現在のロケールに対応する呼称の文字列を取得し，その文字列を返す。

4.13.4. 状態

SeatSwitchの各インスタンスは，以下の状態を持つ。

mainStatus	SeatSwMainStatusType	スイッチの主状態（正常/停止/故障/電気故障など）

4.13.5. サービスコール一覧

SeatSwitchに対するサービスコールは以下の通り。

getConfigAll	シート動作スイッチの構成情報の参照
getStatus	シート動作スイッチの状態の参照
notify	Seatスイッチイベントの通知要求
unnotify	Seatスイッチイベントの通知停止

getEvent	Seatスイッチイベントの取得

4.13.6. イベント

Seatスイッチに関するイベントとして，スイッチ状態変化，故障検出/回復，構成変更，イベントキューのオーバーフローなどを定義する。

SeatSwitchに対するイベント種別は以下の通り。

MainStatusChanged	シート付属スイッチのmainStatus入力状態が変化
FaultDetected	故障検知
FaultRecovered	故障から回復
EventOverflow	イベントキューのオーバフロー
ConfigChanged	構成情報の変更

この内，イベント種別がMainStatusChangedの場合には，以下の関連情報を持つ。

newState	SeatSwitchMainStatusType	スイッチの新しい入力状態

また，イベント種別がEventOverflowの場合には，以下の関連情報を持つ。

noLostEvents	UInt32	失われたイベントの数

4.13.7. データ型の定義

4.13.7.1. Seatスイッチの機能クラス

```
type SeatSwitchCapabilityType = enum {
    SwitchControl
};
```

4.13.7.2. スイッチ系統

```
type SeatSwitchCategoryType = enum {
    FrontRight, // FrontRightSeatスイッチ系統
    FrontLeft,  // FrontLeftSeatスイッチ系統
    RearRight,  // RearRightSeatスイッチ系統
    RearLeft    // RearLeftSeatスイッチ系統
};
```

4.13.7.3. Seatスイッチの入力状態

```
type SeatSwitchMainStatusType = enum {
    NoAction,   // NoAction（操作なし）
    Up, // Up(割付機能により意味が異なる)
```

```
    Down    // Down(割付機能により意味が異なる)
};
```

Seatイッチ構成情報（getConfigAll が返すデータ型）

```
type SeatSwitchConfigType = struct {
    instanceId : IdType,
    placement : PlacementType,
    switchCategory : SeatSwitchCategoryType,
    capabilities : enumSet(SeatSwitchCapabilityType),
    switchType : String,
    displayName : String
};
```

4.13.7.4. Seatスイッチ状態

```
type SeatSwitchStatusType = struct {
    mainStatus : SeatSwMainStatusType
};
```

Seatスイッチの状態情報

4.13.7.5. Seatスイッチイベント種別

```
type SeatSwitchEventKindType = enum {
    MainStatusChanged,
    FaultDetected,
    FaultRecovered,
    EventOverflow,
    ConfigChanged
};
```

```
type SeatSwitchEventType = struct {
    instanceId : IdType,
    switchCategory : SeatSwitchCategoryType,
    eventInfo : variant(SeatSwitchEventKindType) {
        MainStatusChanged {
            newState : SeatSwitchMainStatusType
        },
        EventOverflow {
            noLostEvents : UInt32
        }
    }
};
```

4.13.8. サービスコールの詳細規定

getConfigAll

シート動作スイッチの構成情報の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
config	SeatSwConfigType[]	シート動作スイッチに関する構成情報

【エラーコード】

E_NOT_SUPPORTED	未サポート（実装により返却されない場合あり）

【機能】

シート動作スイッチに関する構成情報を返す。

・識別名

・IdType（API呼び出しに使う）

・位置

・紐づいている機能

・サポートする機能（オプション）

すべてON/OFFのスイッチを仮定するため

スイッチとしてサポートする機能はオプション扱いとする。

getStatus

シート動作スイッチの状態の参照

【パラメータ】

instanceId	IdType	シート動作スイッチのID

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
status	SeatSwStatusType	シート動作スイッチの状態

【エラーコード】

E_INVALID_ID	instanceId が有効範囲外

【機能】

操作対象のSeatスイッチに関する状態を返す。



status には mainStatus（各Seat系統のスイッチ入力状態）を含む。

notify

Seatスイッチイベントの通知要求

【パラメータ】

instanceId	IdType?	イベント通知対象のSeatスイッチID（省略時は全インスタンス）
eventFilter	enumSet (SeatSwitchEventKindType)?	通知するイベント種別の集合（省略時は全種）

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
notifyHandle	NotifyHandleType	通知ハンドル

【エラーコード】

E_INVALID_ID	instanceId が有効範囲外
E_INVALID_PARAMETER	eventFilter が有効範囲外

【機能】

イベント通知対象のSeatスイッチで発生したeventFilterで指定したイベントの通知を要求する。イベントを通知するためのイベントキューを生成し，その通知ハンドルをnotifyHandleに返す。

instanceIdを省略した場合，すべてのインスタンスで発生したイベントの通知を要求する。eventFilterを省略した場合，すべての種類のイベントの通知を要求する。

unnotify

Seatスイッチイベントの通知停止

【パラメータ】

notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandle が不正

【機能】

notifyHandleで指定した通知を停止する。指定した通知用のイベントキューは削除される。

getEvent

Seatスイッチイベントの取得

【パラメータ】

notifyHandle	NotifyHandleType	イベントキューを識別する通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
event	SeatSwitchEventType	取得したイベント

【エラーコード】

E_INVALID_HANDLE	notifyHandle が不正
E_NO_DATA	イベントキューが空

【機能】

notifyHandleで指定した通知用のイベントキューから，先頭のイベントを取り出す。取り出したイベントは，イベントキューから削除される。



イベントがパラメータ等で渡されてくるプログラミング環境向けの物理APIでは，用意する必要がない。

4.14. Vehicle.Cabin.Shade（略称：Shade）

4.14.1. 位置付けと機能

Vehicle.Cabin.Shade（略称：Shade）は，車両のシェイドを操作するためのマルチインスタンスオブジェクトである。

4.14.1.1. シェイドの位置付けと性質

シェイドとは，開閉できる車室の開口部に付属するもので，車両の走行中も開けておいて差し支えないものと定義する。シェイドを開閉するための物理スイッチは，シェイドとは別のオブジェクトとして扱う。

シェイドは，閉じていると光を通さず，開いていると光を通す。

Shadeの機能

Shadeは，可動部を一次元に移動できるオブジェクトであり，MovableObjectの機能を持つ。

可動部の位置は，いずれの方向の移動に対しても，どれだけ開いているかのパーセンテージ（開閉度，0：全閉，100：全開）で表す。また，0に近づく動作を閉動作，100に近づく動作を開動作と呼ぶ。

挟み込み防止機能を持つシェイドは，可動部の閉動作中に挟み込みを検知すると，閉動作を停止し，開動作を開始する。挟み込み検知後にどの位置まで開くかは，車両依存である。

可動部が停止中の間は，現在位置が変化することはない。可動部が強制的に移動されたなどの理由で現在位置が変化した場合には，シェイドが一時的に故障し，すぐに回復したものと扱う。

4.14.1.2. シェイドに対する制御の調停

シェイドに対する制御の調停には，標準的な調停機能（後勝ちの原則）が適用される。すなわち，シェイドを閉めるサービスコールが呼び出され，シェイドが閉じる動作中に，開くサービスコールが呼び出されると，最初のサービスコールによる制御はキャンセルされ，シェイドは開く動作を始める。

4.14.2. 機能クラス

Shadeは，以下の機能クラスを持つ。

Movable	パワーシェイドであり，APIにより移動させることができる
ClosedDetectable	シェイドが全閉状態であることを検出することができる

4.14.3. リスククラス

Shadeは，以下のリスククラスを持つ。

RiskPinch	シェイドを閉めることに伴う挟み込みのリスク

4.14.4. 構成情報

Shadeの各インスタンスは，以下の構成情報を持つ。

instanceId	IdType	インスタンスが持つId
placement	PlacementType	インスタンスがある場所
capabilities	enumSet (ShadeCapabilityType)	そのシェイドが持つ機能
riskClasses	enumSet(ShadeRiskType)	そのシェイドのリスククラスの集合
mountedOn	RelatedObjectType	設置対象物（ドアまたは車室）
displayName	String	ユーザに表示する呼称を取得するためのキー

上記は，YAML形式で記述した構成記述ファイル中に記述する形式である。構成情報を参照するサービスコールは，この構成情報に対して，次の変換を行った構成情報を返す。

- mountedOnは，物理APIで定められたデータ型に変換して返す。
- displayNameKeyに対しては，それを元に現在のロケールに対応する呼称の文字列を取得し，その文字列を返す。

4.14.5. 状態

Shadeの各インスタンスは，以下の状態を持つ。

mainStatus	ShadeMainStatusType	主状態
lockStatus	LockStatusType	ロック状態
position	PositionType	現在位置
targetPosition	PositionType	目標位置

シェイドの主状態（mainStatus）は，MovableObjectの主状態に加えて，挟み込み回避中の状態をとる。また，位置のパーセンテージが大きくなる方へ移動中を開動作中，位置のパーセンテージが小さくなる方へ移動中を閉動作中と呼ぶ。

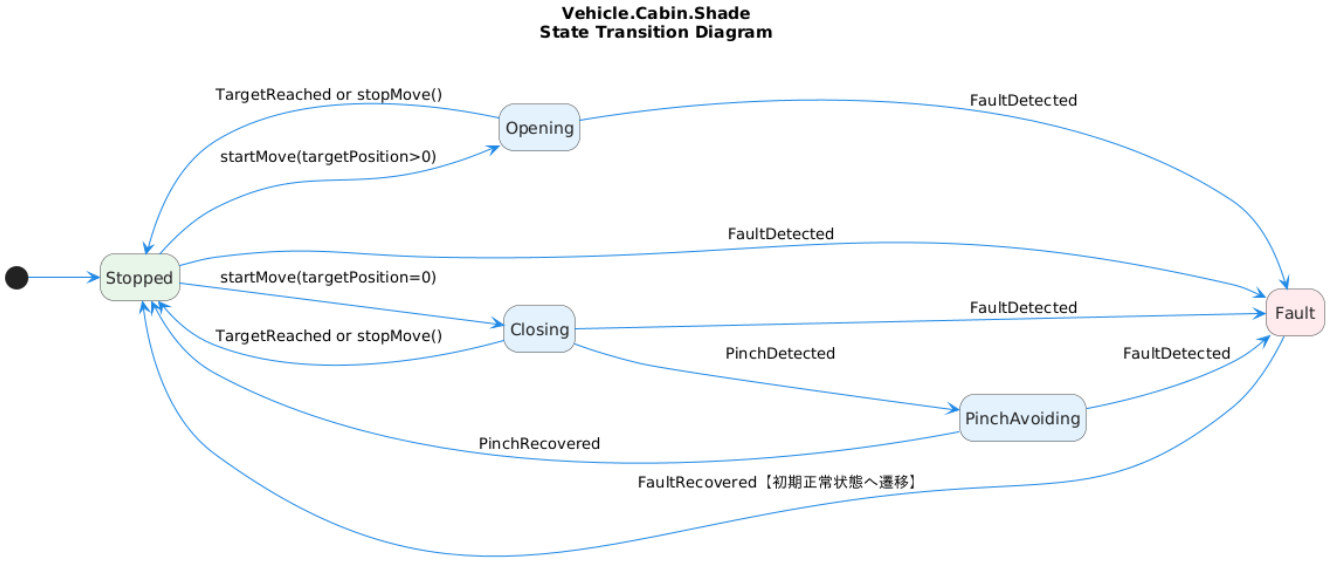


Figure 14. シェイドの主状態の状態遷移

シェイドの現在位置（position）と目標位置（targetPosition），MovableObjectの現在位置と目標位置に関する規定に従う。

4.14.6. サービスコール一覧

Shadeに対するサービスコールは以下の通り。

getConfigAll	すべてのシェイドの構成情報の参照
getStatus	シェイドの状態の参照
startMove	シェイドの開閉の開始
stopMove	シェイドの開閉の停止
notify	シェイドイベントの通知要求
unnotify	シェイドイベントの通知停止
getEvent	シェイドイベントの取得

4.14.7. イベント

シェイドに対するイベント種別には，MovableObjectの持つイベントとLockableObjectが持つイベントに加えて，挟み込み検知と挟み込みから回復がある。

Shadeに対するイベント種別は以下の通り。

ControlledBySelf	自アプリケーションが制御
ControlledByOther	他のアプリケーションが制御
TargetReached	指定された目標位置に到達して停止，または，移動停止が完了
FaultDetected	故障検知
FaultRecovered	故障から回復
PinchDetected	挟み込み検知
PinchRecovered	挟み込みから回復
OwnLockRevoked	自アプリケーションのロックの強制解除
OtherLockReleased	他のアプリケーション等によるロック解除
EventOverflow	イベントキューのオーバフロー
ConfigChanged	構成情報の変更

この内，イベント種別がControlledByOtherの場合には，以下の関連情報を持つ。

sourceApplicatio n	ApplicationIdType	操作したアプリケーション

また，イベント種別がEventOverflowの場合には，以下の関連情報を持つ。

noLostEvents	UInt32	失われたイベントの数

4.14.8. データ型の定義

4.14.8.1. シェイドの機能クラス

```
type ShadeCapabilityType = enum {
    Movable,
    ClosedDetectable
};
```

4.14.8.2. シェイドに対するリスククラス

```
type ShadeRiskType = enum {
    RiskPinch
};
```

4.14.8.3. シェイドの構成情報（getConfigAllが返すデータ型）

```
type ShadeConfigType = struct {
    instanceId : IdType,
    placement : PlacementType,
    capabilities : enumSet(ShadeCapabilityType),
    riskClasses : enumSet(ShadeRiskType),
    mountedOn : RelatedObjectType,
    displayName : String
};
```

4.14.8.4. シェイド状態

```
type ShadeMainStatusType = enum {
    Stopped,    // 停止中
    Opening,    // 開動作中
    Closing,    // 閉動作中
    Fault,  // 故障中
    PinchAvoiding   // 挟み込み回避中
};
```

```
type ShadeStatusType = struct {
    mainStatus : ShadeMainStatusType,
    lockStatus : LockStatusType,
    position : PositionType,
    targetPosition : PositionType
};
```

4.14.8.5. 移動プロファイル

```
type ShadeMoveProfileType = enum {
    Standard,   // 標準的な移動
    Fast,   // 高速で移動
    Slow    // 低速で移動
};
```

4.14.8.6. トランクイベント

```
type ShadeEventKindType = enum {
    ControlledBySelf,
    ControlledByOther,
    TargetReached,
    FaultDetected,
    FaultRecovered,
    PinchDetected,
```

```
    PinchRecovered,
    OwnLockRevoked,
    OtherLockReleased,
    EventOverflow,
    ConfigChanged
};
```

```
type ShadeEventType = struct {
    instanceId : IdType,
    eventInfo : variant(ShadeEventKindType) {
        ControlledByOther {
            sourceApplication : ApplicationIdType
        },
        EventOverflow {
            noLostEvents : UInt32
        }
    }
};
```

4.14.9. サービスコールの詳細規定

getConfigAll

すべてのシェイドの構成情報の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
config	ShadeConfigType[]	すべてのシェイドに関する構成情報

【エラーコード】

なし

【機能】

すべてのシェイドに関する構成情報を返す。

getStatus

シェイドの状態の参照

【パラメータ】

instanceId	IdType	操作対象のシェイドのID

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
status	ShadeStatusType	対象シェイドの状態

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外

【機能】

操作対象のシェイドに関するすべての状態を返す。

startMove

シェイドの開閉の開始

【パラメータ】

instanceId	IdType	操作対象のシェイドのID
targetPosition	PositionType	目標位置（開閉度）
moveProfile	ShadeMoveProfileType?	移動プロファイル
priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_INVALID_PARAMETER	targetPosition，priorityが有効範囲外
E_RISK_APPLICATION	アプリケーションが対応していないリスクがある
E_RISK_USER	ユーザがリスクを承認していない
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_FAULT	操作対象のシェイドが故障中
E_OBJECT_STATUS	操作対象のシェイドが挟み込み回避中

【機能】

目標位置をtargetPositionに設定し，操作対象のシェイドの開閉動作を開始する。

targetPositionに0を指定すれば閉動作の開始，100を指定すれば開動作の開始の意味になる。中間値を設定した場合、現在の位置と比較し作動方向を決定する。targetPositionに0～100の中間を設定した場合、その開度まで作動し止まることを意味する。

リスククラスRiskPinchに対応できていない場合でも，シェイドを開く方向に動作を開始しようとした場合には，操作を受け付ける。そうでない場合には，E_RISK_APPLICATIONエラーまたはE_RISK_USERエ

ラーとなる。

ShadeMoveProfileTypeは、標準、高速、低速を選択可能であるが、制御対象の性能差により絶対速度の保証は困難である。具体的な速度に関しては、各制御対象ごとで決定することを想定している。

stopMove

シェイドの開閉の停止

【パラメータ】

instanceId	IdType	操作対象のシェイドのID
priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_INVALID_PARAMETER	priorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_FAULT	操作対象のシェイドが故障中
E_OBJECT_STATUS	操作対象のシェイドが挟み込み回避中
E_OBJECT_LOCKED	操作対象のシェイドが，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている

【機能】

各次元方向の目標位置を現在位置に設定して，操作対象のシェイドの各次元方向への開閉動作の停止を要求する。

notify

シェイドイベントの通知要求

【パラメータ】

instanceId	IdType?	イベント通知対象のシェイドのID
eventFilter	enumSet (ShadeEventKindType)?	通知するイベントの種類の集合

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_INVALID_PARAMETER	eventFilterが有効範囲外

【機能】

イベント通知対象のシェイドで発生したeventFilterで指定したイベントの通知を要求する。イベントを通知するためのイベントキューを生成し，その通知ハンドルを notifyHandle に返す。

instanceIdを省略した場合，すべてのインスタンスで発生したイベントの通知を要求する。eventFilterを省略した場合，すべての種類のイベントの通知を要求する。

unnotify

シェイドイベントの通知停止

【パラメータ】

notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正

【機能】

notifyHandleで指定した通知を停止する。指定した通知用のイベントキューは削除される。

getEvent

シェイドイベントの取得

【パラメータ】

notifyHandle	NotifyHandleType	イベントキューを識別するための通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
event	ShadeEventType	取得したイベント

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正
E_NO_DATA	イベントキューが空

【機能】

notifyHandleで指定した通知用のイベントキューから，先頭のイベントを取り出す。取り出したイベントは，イベントキューから削除される。



イベントがパラメータ等で渡されてくるプログラミング環境向けの物理APIでは，用意する必要がない。

4.15. Vehicle.Cabin.ShadeSwitch（略称：ShadeSwitch）

4.15.1. 位置付けと機能

Shadeを制御する物理的なスイッチを扱う。スイッチとShadeは直結せず，制御はOS側（ビークルOS）で行うことを仮定する。

4.15.1.1. Shadeスイッチの位置付けと性質

Shadeスイッチとは，各Shadeに配置された乗員が物理操作できるShade制御デバイスである。

スイッチはレバー形および2段階式である。

Shadeスイッチは各Shadeに配置され独立した系統を持つ。

スイッチとShadeは直接接続されず，制御要求はビークルOSが受け取る。

スイッチは入力源として機能し，Shade制御の主体にはならない。

4.15.1.2. Shadeスイッチの機能

Shadeスイッチは乗員の操作によるスイッチ入力をOSへ提供し，Shadeオブジェクトへ制御要求を伝達する役割を持つ。

各スイッチの系統はconfigData内のswitchCategoryによって識別される。

4.15.1.3. Shadeスイッチに対する制御の調停

Shadeスイッチは入力イベントの発生源であり，自身が制御競合を調停することはない。

入力はOSが一元管理し，Shadeオブジェクトへルーティングされる。

調停はShade側の後勝ち原則に従う。スイッチ操作が他アプリケーションの制御要求と競合した場合，Shade側の調停ロジックにより採用または却下が判断される。

スイッチ自身は制御主体ではなく，アプリケーションや自動制御と同様に「入力の一つ」として扱われる。

4.15.2. 機能クラス

ShadeSwitchは，以下の機能クラスを持つ。

SwitchControl	Shade操作入力を提供する基本機能

4.15.3. 構成情報

ShadeSwitchの各インスタンスは，以下の構成情報を持つ。

instanceId	IdType	インスタンスが持つId
placement	PlacementType	インスタンスがある場所
switchCategory	ShadeSwitchCategoryType	スイッチ系統（FrontRight　FrontLeft　RearRight　RearLeft　Roof）
capabilities	enumSet (ShadeSwitchCapabilityType)	このスイッチがサポートする操作機能の集合
switchType	String	スイッチ種別（UP側のみ，DONW側のみ，２段階スイッチかなど，OEM依存）
displayName	String	ユーザに表示する呼称を取得するためのキー

上記は，YAML形式で記述した構成記述ファイル中に記述する形式である。構成情報を参照するサービスコールは，この構成情報に対して，次の変換を行った構成情報を返す。

- displayNameKeyに対しては，それを元に現在のロケールに対応する呼称の文字列を取得し，その文字列を返す。

4.15.4. 状態

ShadeSwitchの各インスタンスは，以下の状態を持つ。

mainStatus	ShadeSwitchMainStatusType	スイッチの入力状態（UP側のみ，DONW側のみ，２段階スイッチ，など）

4.15.5. サービスコール一覧

ShadeSwitchに対するサービスコールは以下の通り。

getConfigAll	すべてのShadeスイッチの構成情報の参照
getStatus	Shadeスイッチの状態参照
notify	Shadeスイッチイベントの通知要求
unnotify	Shadeスイッチイベントの通知停止
getEvent	Shadeスイッチイベントの取得

4.15.6. イベント

Shadeスイッチに関するイベントとして，スイッチ状態変化，故障検出/回復，構成変更，イベントキューのオーバーフローなどを定義する。

ShadeSwitchに対するイベント種別は以下の通り。

MainStatusChanged	Front/RearスイッチのmainStatus入力状態が変化

FaultDetected	故障検知
FaultRecovered	故障から回復
EventOverflow	イベントキューのオーバフロー
ConfigChanged	構成情報の変更

この内，イベント種別がMainStatusChangedの場合には，以下の関連情報を持つ。

newState	ShadeSwitchMainStatusType	スイッチの新しい入力状態

また，イベント種別がEventOverflowの場合には，以下の関連情報を持つ。

noLostEvents	UInt32	失われたイベントの数

4.15.7. データ型の定義

4.15.7.1. Shadeスイッチの機能クラス

```
type ShadeSwitchCapabilityType = enum {
    SwitchControl
};
```

4.15.7.2. スイッチ系統

```
type ShadeSwitchCategoryType = enum {
    FrontRight, // FrontRightShadeスイッチ系統
    FrontLeft,  // FrontLeftShadeスイッチ系統
    RearRight,  // RearRightShadeスイッチ系統
    RearLeft,   // RearLeftShadeスイッチ系統
    Roof    // Roofスイッチ系統
};
```

4.15.7.3. Shadesスイッチの入力状態

```
type ShadeSwitchMainStatusType = enum {
    False,  // OFF - Shade停止
    Up, // UP - ShadeUP:押している間屋根側に移動
    UpAUuto,    // UPAuto - ShadeUP AUTO：自動で屋根側端点まで作動
    Down,   // Down - ShadeDown:押している間床側に移動
    DownAuto    // DownAuto - Shades Down Auto：自動で床側端点まで作動
};
```

Shadeスイッチ構成情報（getConfigAll が返すデータ型）

```
type ShadeSwitchConfigType = struct {
    instanceId : IdType,
    placement : PlacementType,
    switchCategory : ShadeSwitchCategoryType,
    capabilities : enumSet(ShadeSwitchCapabilityType),
    switchType : String,
    displayName : String
};
```

4.15.7.4. Shadeスイッチ状態

```
type ShadeSwitchStatusType = struct {
    mainStatus : ShadeSwitchMainStatusType
};
```

Shadeスイッチの状態情報

4.15.7.5. Shadeスイッチイベント種別

```
type ShadeSwitchEventKindType = enum {
    MainStatusChanged,
    FaultDetected,
    FaultRecovered,
    EventOverflow,
    ConfigChanged
};
```

```
type ShadeSwitchEventType = struct {
    instanceId : IdType,
    switchCategory : ShadeSwitchCategoryType,
    eventInfo : variant(ShadeSwitchEventKindType) {
        MainStatusChanged {
            newState : ShadeSwitchMainStatusType
        },
        EventOverflow {
            noLostEvents : UInt32
        }
    }
};
```

4.15.8. サービスコールの詳細規定

getConfigAll

すべてのShadeスイッチの構成情報の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
config	ShadeSwitchConfigType[]	すべてのShadeスイッチに関する構成情報

【機能】

すべてのShadeスイッチに関する構成情報を返す。

getStatus

Shadeスイッチの状態参照

【パラメータ】

instanceId	IdType	操作対象のShadeスイッチのID

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
status	ShadeSwitchStatusType	対象のShadeスイッチの状態

【エラーコード】

E_INVALID_ID	instanceId が有効範囲外

【機能】

操作対象のShadeスイッチに関する状態を返す。



status には mainStatus（各Shade系統のスイッチ入力状態）を含む。

notify

Shadeスイッチイベントの通知要求

【パラメータ】

instanceId	IdType?	イベント通知対象のShadeスイッチID（省略時は全インスタンス）

eventFilter	enumSet (ShadeSwitchEventKindType)?	通知するイベント種別の集合（省略時は全種）

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
notifyHandle	NotifyHandleType	通知ハンドル

【エラーコード】

E_INVALID_ID	instanceId が有効範囲外
E_INVALID_PARAMETER	eventFilter が有効範囲外

【機能】

イベント通知対象のShadeスイッチで発生したeventFilterで指定したイベントの通知を要求する。イベントを通知するためのイベントキューを生成し，その通知ハンドルをnotifyHandleに返す。

instanceIdを省略した場合，すべてのインスタンスで発生したイベントの通知を要求する。eventFilterを省略した場合，すべての種類のイベントの通知を要求する。

unnotify

Shadeスイッチイベントの通知停止

【パラメータ】

notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandle が不正

【機能】

notifyHandleで指定した通知を停止する。指定した通知用のイベントキューは削除される。

getEvent

Shadeスイッチイベントの取得

【パラメータ】

notifyHandle	NotifyHandleType	イベントキューを識別する通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
event	ShadeSwitchEventType	取得したイベント

【エラーコード】

E_INVALID_HANDLE	notifyHandle が不正
E_NO_DATA	イベントキューが空

【機能】

notifyHandleで指定した通知用のイベントキューから，先頭のイベントを取り出す。取り出したイベントは，イベントキューから削除される。



イベントがパラメータ等で渡されてくるプログラミング環境向けの物理APIでは，用意する必要がない。

4.16. Vehicle.Cabin.Window（略称：Window）

4.16.1. 位置付けと機能

Vehicle.Cabin.Window（略称：Window）は，車両のウィンドウ（窓）を操作するためのマルチインスタンスオブジェクトである。

4.16.1.1. ウィンドウの位置付けと性質

ウィンドウとは，開閉できる車室の開口部で，車両の走行中も開けておいて差し支えないものと定義する。サンルーフやコンバーチブルトップ（オープンカーの開閉する屋根）もウィンドウと扱う。それに対して，フロントウィンドウ，リアウィンドウは，開閉できないため，ウィンドウとは扱わない（ガラスと扱い，フロントガラス，リアガラスと呼ぶことにする）。ウィンドウを開閉するための物理スイッチは，ウィンドウとは別のオブジェクトとして扱う。

ウィンドウは，閉じていると気体・液体を通さず，開いていると気体・液体を通す。光は，開閉に関わらず通すが，光を遮るためのサンシェードを備えている場合がある。人が通ることや大きい荷物を出し入れすることは想定していない。



常に光を遮るウィンドウは，現時点では想定していない。

4.16.1.2. Windowの機能

Windowは，可動部を一次元に移動できるオブジェクトであり，MovableObjectの機能を持つ。

チルト機構を持つサンルーフのように，可動部を二次元に移動できるウィンドウを扱うために，Windowは，オプションで，可動部を第二次元方向に移動させる機能を持つ。この場合，第二次元方向の移動に対してもMovableObjectと同等の機能を持つ。

可動部の位置は，いずれの方向の移動に対しても，どれだけ開いているかのパーセンテージ（開閉度，0：全閉，100：全開）で表す。また，0に近づく動作を閉動作，100に近づく動作を開動作と呼ぶ。

挟み込み防止機能を持つWindowは，可動部の閉動作中に挟み込みを検知すると，閉動作を停止し，開動作を開始する。挟み込み検知後にどの位置まで開くかは，車両依存である。

可動部が停止中の間は，現在位置が変化することはない。可動部が強制的に移動されたなどの理由で現在位置が変化した場合には，Windowが一時的に故障し，すぐに回復したものと扱う。

4.16.1.3. Windowに対する制御の調停

Windowに対する制御の調停には，標準的な調停機能（後勝ちの原則）が適用される。すなわち，Windowを閉めるサービスコールが呼び出され，Windowが閉じる動作中に，開くサービスコールが呼び出されると，最初のサービスコールによる制御はキャンセルされ，Windowは開く動作を始める。

また，Windowは標準的なロック機能を持つLockableObjectである。ウィンドウには，機械的なロック機構は無いのが一般的であり，ソフトウェアでロック機能を実現することを想定している。

Windowが故障または挟み込みを検知した場合，ロックは解除され，故障中と挟み込み回避中は，ロックを取得することができない。

4.16.2. 機能クラス

Windowは，以下の機能クラスを持つ。

Movable	パワーウィンドウであり，APIにより移動させることができる
ClosedDetectable	ウィンドウが全閉状態であることを検出することができる
HasMove2	第二次元方向に移動する機能を持つ
MoveProfileSupported	移動プロファイルをサポートしている

4.16.3. リスククラス

Windowは，以下のリスククラスを持つ。

RiskPinch	ウィンドウを閉めることに伴う挟み込みのリスク
RiskOpen	ウィンドウを開けることに伴う諸々のリスク



RiskOpenをさらに分類するかは，今後の課題である。

4.16.4. 構成情報

Windowの各インスタンスは，以下の構成情報を持つ。

instanceId	IdType	インスタンスID
placement	PlacementType	車室内でのウィンドウの位置
capabilities	enumSet (WindowCapabilityType)	備えている機能クラスの集合
riskClasses	enumSet (WindowRiskType)	対策できていないリスククラスの集合
mountedOn	RelatedObjectType	設置対象物（ドアまたは車室）
displayName	String	ユーザに表示する呼称を取得するためのキー

上記は，YAML形式で記述した構成記述ファイル中に記述する形式である。構成情報を参照するサービスコールは，この構成情報に対して，次の変換を行った構成情報を返す。

- mountedOnは，物理APIで定められたデータ型に変換して返す。
- displayNameに対しては，それを元に現在のロケールに対応する呼称の文字列を取得し，その文字列を以下の構成情報として返す。

displayName	String	ユーザに表示する呼称

4.16.5. 状態

Windowの各インスタンスは，以下の状態を持つ。

mainStatus	WindowMainStatusType	主状態
mainStatus2	WindowMainStatusType?	第二次元方向の主状態
lockStatus	LockStatusType	ロック状態
position	PositionType	現在位置
targetPosition	PositionType	目標位置
position2	PositionType?	第二次元方向の現在位置
targetPosition2	PositionType?	第二次元方向の目標位置

ここで，mainStatus2，position2，targetPosition2は，機能クラスHasMove2を備えている場合にのみ有効である。

Windowの主状態（mainStatusとmainStatus2）は，MovableObjectの主状態に加えて，挟み込み回避中の状態をとる。また，位置のパーセンテージが大きくなる方へ移動中を開動作中，位置のパーセンテージが小さくなる方へ移動中を閉動作中と呼ぶ（[Fig-ウィンドウの主状態の状態遷移]）。

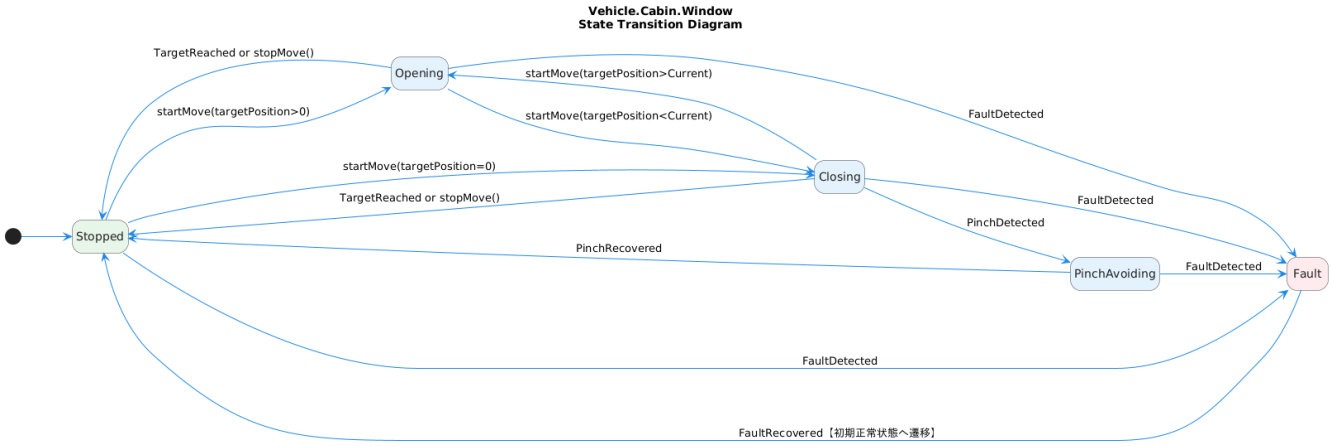


Figure 15. ウインドウの主状態の状態遷移

Windowの現在位置（position）と目標位置（targetPosition），第二次元方向の現在位置（position2）と目標位置（targetPosition2）は，MovableObjectの現在位置と目標位置に関する規定に従う。

4.16.6. サービスコール一覧

Windowに対するサービスコールは以下の通り。

getConfigAll	Windowのすべてのインスタンスの構成情報の参照
getStatus	Windowの状態の参照
startMove	Windowの開閉の開始
startMove2	Windowの第二次元方向の開閉の開始
stopMove	Windowの開閉の停止
lock	Windowのロックの取得
unlock	Windowのロックの解除
notify	Windowイベントの通知要求
unnotify	Windowイベントの通知停止
getEvent	Windowイベントの取得

4.16.7. イベント

Windowに対するイベント種別には，すべてのオブジェクトが持つイベント，MovableObjectが持つイベント，LockableObjectが持つイベントに加えて，挟み込み検知と挟み込みから回復がある。

Windowに対するイベント種別は以下の通り。

ControlledBySelf	自アプリケーションが制御
ControlledByOther	他のアプリケーションが制御
TargetReached	指定された目標位置に到達して停止，または，移動停止が完了
FaultDetected	故障検知
FaultRecovered	故障から回復
PinchDetected	挟み込み検知
PinchRecovered	挟み込みから回復
OwnLockRevoked	自アプリケーションのロックの強制解除
OtherLockReleased	他のアプリケーション等によるロック解除
EventOverflow	イベントキューのオーバフロー
ConfigChanged	構成情報の変更

この内，イベント種別がControlledByOtherの場合には，以下の関連情報を持つ。

sourceApplication	ApplicationIdType	操作したアプリケーション

また，イベント種別がTargetReachedの場合には，以下の関連情報を持つ。

sourceApplication	ApplicationIdType	操作したアプリケーション

また，イベント種別がEventOverflowの場合には，以下の関連情報を持つ。

noLostEvents	UInt32	失われたイベントの数

4.16.8. データ型の定義

(1) Windowの機能クラス

```
type WindowCapabilityType = enum {
    Movable,
    ClosedDetectable,
    HasMove2,
    MoveProfileSupported
};
```

(2) Windowに対するリスククラス

```
type WindowRiskType = enum {
    RiskPinch,
    RiskOpen
};
```

(3) Windowの構成情報（getConfigAllが返すデータ型）

```
type WindowConfigType = struct {
    instanceId : IdType,
    placement : PlacementType,
    capabilities : enumSet(WindowCapabilityType),
    riskClasses : enumSet(WindowRiskType),
    mountedOn : RelatedObjectType,
    displayName : String
};
```

(4) Window状態

```
type WindowMainStatusType = enum {
    Stopped,    // 停止中
    Opening,    // 開動作中
    Closing,    // 閉動作中
    Fault,  // 故障中
    PinchAvoiding   // 挟み込み回避中
};
```

```
type WindowStatusType = struct {
    mainStatus : WindowMainStatusType,
    mainStatus2 : WindowMainStatusType?,
```

```
    lockStatus : LockStatusType,
    position : PositionType,
    targetPosition : PositionType,
    position2 : PositionType?,
    targetPosition2 : PositionType?
};
```

mainStatus2，position2，targetPosition2は，機能クラスHasMove2を備えている場合にのみ値を持つ。

(5) Windowの移動プロファイル

```
type WindowMoveProfileType = enum {
    Standard,   // 標準的な移動
    Fast,   // 高速で移動
    Slow    // 低速で移動
};
```

(6) Windowイベント

```
type WindowEventKindType = enum {
    ControlledBySelf,
    ControlledByOther,
    TargetReached,
    FaultDetected,
    FaultRecovered,
    PinchDetected,
    PinchRecovered,
    OwnLockRevoked,
    OtherLockReleased,
    EventOverflow,
    ConfigChanged
};
```

```
type WindowEventType = struct {
    instanceId : IdType,
    eventInfo : variant(WindowEventKindType) {
        ControlledByOther {
            sourceApplication : ApplicationIdType
        },
        TargetReached {
            sourceApplication : ApplicationIdType
        },
        EventOverflow {
            noLostEvents : UInt32
        }
    }
```

```
};
```

4.16.9. サービスコールの詳細規定

getConfigAll

Windowのすべてのインスタンスの構成情報の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
config	WindowConfigType[]	Windowのすべてのインスタンスに関する構成情報

【エラーコード】

なし

【機能】

Windowのすべてのインスタンスのすべての構成情報を返す。

getStatus

Windowの状態の参照

【パラメータ】

instanceId	IdType	操作対象のWindowのID

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
status	WindowStatusType	対象Windowの状態

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外

【機能】

状態参照対象のWindowに関するすべての状態を返す。

startMove

Windowの開閉の開始

【パラメータ】

instanceId	IdType	操作対象のWindowのID
targetPosition	PositionType	目標位置（開閉度）
moveProfile	WindowMoveProfileType?	移動プロファイル
priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_INVALID_PARAMETER	targetPosition，moveProfile，priorityが有効範囲外
E_RISK_APPLICATION	アプリケーションが対応していないリスクがある
E_RISK_USER	ユーザがリスクを承認していない
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_FAULT	操作対象のWindowが故障中
E_OBJECT_STATUS	操作対象のWindowが挟み込み回避中
E_OBJECT_LOCKED	操作対象のWindowが，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている

【機能】

目標位置をtargetPositionに設定し，操作対象のWindowの開閉動作を開始する。必要な場合には，第二次元方向への開閉動作も開始する（例えば，第二次元方向に閉まっている時しか，第一次元方向に開くことができない場合）。

targetPositionに0を指定すれば閉動作の開始，100を指定すれば開動作の開始の意味になる。
targetPositionに0～100の中間を設定した場合，その開度まで作動し止まることを意味する。

リスククラスRiskPinchに対応できていない場合でも，Windowを開く方向に動作を開始しようとした場合には，操作を受け付ける。そうでない場合には，E_RISK_APPLICATIONエラーまたはE_RISK_USERエラーとなる。

リスククラスRiskOpenに対応できていない場合でも，Windowを閉じる方向に動作を開始しようとした場合には，操作を受け付ける。そうでない場合には，E_RISK_APPLICATIONエラーまたはE_RISK_USERエラーとなる。

WindowMoveProfileTypeは，標準，高速，低速を選択可能であるが，制御対象の性能差により絶対速度の保証は困難である。具体的な速度に関しては，各制御対象ごとで決定することを想定している。

startMove2

Windowの第二次元方向の開閉の開始

【パラメータ】

instanceId	IdType	操作対象のWindowのID
targetPosition2	PositionType	目標位置（開閉度）
moveProfile	WindowMoveProfileType?	移動プロファイル
priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_INVALID_PARAMETER	targetPosition2，moveProfile，priorityが有効範囲外
E_RISK_APPLICATION	アプリケーションが対応していないリスクがある
E_RISK_USER	ユーザがリスクを承認していない
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_FAULT	操作対象のWindowが故障中
E_OBJECT_STATUS	操作対象のWindowが挟み込み回避中
E_OBJECT_LOCKED	操作対象のWindowが，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている
E_NOT_SUPPORTED	第二次元方向に移動する機能を持たない

【機能】

第二次元方向の目標位置をtargetPosition2に設定し，操作対象のWindowの第二次元方向への開閉動作を開始する。必要な場合には，第一次元方向への開閉動作も開始する（例えば，第一次元方向に閉まっている時しか，第二次元方向に開くことができない場合）。

targetPosition2に0を指定すれば閉動作の開始，100を指定すれば開動作の開始の意味になる。
targetPosition2に0～100の中間を設定した場合，その開度まで作動し止まることを意味する。

リスククラスRiskPinchに対応できていない場合でも，Windowを開く方向に動作を開始しようとした場合には，操作を受け付ける。そうでない場合には，E_RISK_APPLICATIONエラーまたはE_RISK_USERエラーとなる。

リスククラスRiskOpenに対応できていない場合でも，Windowを閉じる方向に動作を開始しようとした場合には，操作を受け付ける。そうでない場合には，E_RISK_APPLICATIONエラーまたはE_RISK_USERエラーとなる。

WindowMoveProfileTypeは，標準，高速，低速を選択可能であるが，制御対象の性能差により絶対速度の保証は困難である。具体的な速度に関しては，各制御対象ごとで決定することを想定している。

stopMove

Windowの開閉の停止

【パラメータ】

instanceId	IdType	操作対象のWindowのID
priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_INVALID_PARAMETER	priorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_FAULT	操作対象のWindowが故障中
E_OBJECT_STATUS	操作対象のWindowが挟み込み回避中
E_OBJECT_LOCKED	操作対象のWindowが，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている

【機能】

各次元方向の目標位置を現在位置に設定して，操作対象のWindowの各次元方向への開閉動作の停止を要求する。



1つの次元方向への開閉動作を停止させる機能は用意していない。

lock

Windowのロックの取得

【パラメータ】

instanceId	IdType	操作対象のWindowのID
priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_INVALID_PARAMETER	priorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_LOCKED	ロック対象のWindowが，priorityと同じかそれより高い優先度でロックされている（自アプリケーションが，priorityと同じかそれより高い優先度でロックしている場合を含む）

E_OBJECT_FAULT	ロック対象のWindowが故障中
E_OBJECT_STATUS	ロック対象のWindowが挟み込み回避中

【機能】

ロック対象のWindowを，指定した操作優先度でロックする。

unlock

Windowのロックの解除

【パラメータ】

instanceId	IdType	操作対象のWindowのID

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_OBJECT_STATUS	自アプリケーションが，操作対象のWindowをロックしていない

【機能】

ロック解除対象のWindowを，ロック解除する。

notify

Windowイベントの通知要求

【パラメータ】

instanceId	IdType?	イベント通知対象のWindowのID
eventFilter	enumSet (WindowEventKindType)?	通知するイベントの種類の集合

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_INVALID_PARAMETER	eventFilterが有効範囲外

【機能】

イベント通知対象のWindowで発生したeventFilterで指定した種別のイベントの通知を要求する。イベントを格納するためのイベントキューを生成し，その通知ハンドルをnotifyHandleに返す。

instanceIdを省略した場合，すべてのインスタンスで発生したイベントの通知を要求する。eventFilterを省略した場合，すべての種別のイベントの通知を要求する。

unnotify

Windowイベントの通知停止

【パラメータ】

notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正

【機能】

notifyHandleで指定した通知を停止する。イベントを格納するためのイベントキューを削除する。

getEvent

Windowイベントの取得

【パラメータ】

notifyHandle	NotifyHandleType	イベントを取得するイベントキューの通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
event	WindowEventType	取得したイベント

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正
E_NO_DATA	イベントキューが空

【機能】

notifyHandleで指定した通知用のイベントキューから，先頭のイベントを取り出す。取り出したイベントは，イベントキューから削除される。



イベントがパラメータ等で渡されてくるプログラミング環境向けの物理APIでは，用意する必要がない。

4.17. Vehicle.Cabin.WindowSwitch（略称：WindowSwitch）

4.17.1. 位置付けと機能

Windowを制御する物理的なスイッチを扱う。スイッチとWindowは直結せず，制御はOS側（ビークルOS）で行うことを仮定する。

4.17.1.1. Windowスイッチの位置付けと性質

Windowスイッチとは，各Windowに配置された乗員が物理操作できるWindow制御デバイスである。

スイッチはレバー形および2段階式である。

Windowスイッチは各Windowに配置され独立した系統を持つ。

スイッチとWindowは直接接続されず，制御要求はビークルOSが受け取る。

スイッチは入力源として機能し，Window制御の主体にはならない。

4.17.1.2. Windowスイッチの機能

Windowスイッチは乗員の操作によるスイッチ入力をOSへ提供し，Windowオブジェクトへ制御要求を伝達する役割を持つ。

各スイッチの系統はconfigData内のswitchCategoryによって識別される。

4.17.1.3. Windowスイッチに対する制御の調停

Windowスイッチは入力イベントの発生源であり，自身が制御競合を調停することはない。

入力はOSが一元管理し，Windowオブジェクトへルーティングされる。

調停はWindow側の後勝ち原則に従う。スイッチ操作が他アプリケーションの制御要求と競合した場合，Window側の調停ロジックにより採用または却下が判断される。

スイッチ自身は制御主体ではなく，アプリケーションや自動制御と同様に「入力の一つ」として扱われる。

4.17.2. 機能クラス

WindowSwitchは，以下の機能クラスを持つ。

SwitchControl	Window操作入力を提供する基本機能

4.17.3. 構成情報

WindowSwitchの各インスタンスは，以下の構成情報を持つ。

instanceId	IdType	インスタンスが持つId
placement	PlacementType	インスタンスがある場所
switchCategory	WindowSwitchCategoryType	スイッチ系統（FrontRight　FrontLeft　RearRight　RearLeft　Roof）
capabilities	enumSet (WindowSwitchCapabilityType)	このスイッチがサポートする操作機能の集合
switchType	String	スイッチ種別（開側，閉側，２段階スイッチ，IsWindowChildLockEngagedの機能ON/OFFのスイッチかなど，OEM依存）
displayName	String	ユーザに表示する呼称を取得するためのキー

上記は，YAML形式で記述した構成記述ファイル中に記述する形式である。構成情報を参照するサービスコールは，この構成情報に対して，次の変換を行った構成情報を返す。

- displayNameKeyに対しては，それを元に現在のロケールに対応する呼称の文字列を取得し，その文字列を返す。

4.17.4. 状態

WindowSwitchの各インスタンスは，以下の状態を持つ。

mainStatus	WindowSwitchMainStatusType	スイッチの入力状態（開側，閉側，２段階スイッチ）
windowChildLockStatus	WindowChildLockStatusType	IsWindowChildLockEngagedの機能ON/OFFのスイッチの入力状態。（ON，OFF）

4.17.5. サービスコール一覧

WindowSwitchに対するサービスコールは以下の通り。

getConfigAll	すべてのWindowスイッチの構成情報の参照
getStatus	Windowスイッチの状態参照
notify	Windowスイッチイベントの通知要求
unnotify	Windowスイッチイベントの通知停止
getEvent	Windowスイッチイベントの取得

4.17.6. イベント

Windowスイッチに関するイベントとして，スイッチ状態変化，故障検出/回復，構成変更，イベントキューのオーバーフローなどを定義する。

WindowSwitchに対するイベント種別は以下の通り。

MainStatusChanged	ドア付属スイッチのmainStatus入力状態が変化
ChildLockStatusChanged	IsWindowChildLockEngagedの機能ON/OFFのスイッチの入力状態が変化
FaultDetected	故障検知
FaultRecovered	故障から回復
EventOverflow	イベントキューのオーバフロー
ConfigChanged	構成情報の変更

この内，イベント種別がMainStatusChangedの場合には，以下の関連情報を持つ。

newState	WindowSwitchMainStatusType	スイッチの新しい入力状態

また，イベント種別がChildLockStatusChangedの場合には，以下の関連情報を持つ。

newState	WindowSwitchMainStatusType	スイッチの新しい入力状態

また，イベント種別がEventOverflowの場合には，以下の関連情報を持つ。

noLostEvents	UInt32	失われたイベントの数

4.17.7. データ型の定義

4.17.7.1. Windowスイッチの機能クラス

```
type WindowSwitchCapabilityType = enum {
    SwitchControl
};
```

4.17.7.2. スイッチ系統

```
type WindowSwitchCategoryType = enum {
    FrontRight, // FrontRightWindowスイッチ系統
    FrontLeft,  // FrontLeftWindowスイッチ系統
    RearRight,  // RearRightWindowスイッチ系統
    RearLeft,   // RearLeftWindowスイッチ系統
    Roof    // Roofスイッチ系統
};
```

4.17.7.3. Windowsスイッチの入力状態

```
type WindowSwitchMainStatusType = enum {
    NoAction,   // NoAction（操作なし）
    ManualOpen, // ManualOpen（開1段階目）
    AutoOpen,   // AutoOpen（開2段階目）
    ManualClose,    // ManualClose（閉1段階目）
    AutoClose   // AutoClose（閉2段階目）
};
```

4.17.7.4. IsWindowChildLockEngagedの機能ON/OFFのスイッチの入力状態

```
type WindowChildLockStatusType = enum {
    False,  // IsWindowChildLockEngagedの機能OFF
    True    // IsWindowChildLockEngagedの機能ON
};
```

Windowスイッチ構成情報（getConfigAll が返すデータ型）

```
type WindowSwitchConfigType = struct {
    instanceId : IdType,
    placement : PlacementType,
    switchCategory : WindowSwitchCategoryType,
    capabilities : enumSet(WindowSwitchCapabilityType),
    switchType : String,
    displayName : String
};
```

4.17.7.5. Windowスイッチ状態

```
type WindowSwitchStatusType = struct {
    mainStatus : WindowSwitchMainStatusType,
    windowChildLockStatus : WindowChildLockStatusType
};
```

Windowスイッチの状態情報

4.17.7.6. Windowスイッチイベント種別

```
type WindowSwitchEventKindType = enum {
    MainStatusChanged,
    ChildLockStatusChanged,
    FaultDetected,
    FaultRecovered,
    EventOverflow,
    ConfigChanged
```

```
};
```

```
type WindowSwitchEventType = struct {
    instanceId : IdType,
    switchCategory : WindowSwitchCategoryType,
    eventInfo : variant(WindowSwitchEventKindType) {
        MainStatusChanged {
            newState : WindowSwitchMainStatusType
        },
        ChildLockStatusChanged {
            newState : WindowSwitchMainStatusType
        },
        EventOverflow {
            noLostEvents : UInt32
        }
    }
};
```

4.17.8. サービスコールの詳細規定

getConfigAll

すべてのWindowスイッチの構成情報の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
config	WindowSwitchConfigType[]	すべてのWindowスイッチに関する構成情報

【機能】

すべてのWindowスイッチに関する構成情報を返す。

getStatus

Windowスイッチの状態参照

【パラメータ】

instanceId	IdType	操作対象のWindowスイッチのID

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

status	WindowSwitchStatusType	対象のWindowスイッチの状態

【エラーコード】

E_INVALID_ID	instanceId が有効範囲外

【機能】

操作対象のWindowスイッチに関する状態を返す。



status には mainStatus（各Window系統のスイッチ入力状態）を含む。

notify

Windowスイッチイベントの通知要求

【パラメータ】

instanceId	IdType?	イベント通知対象のWindowスイッチID（省略時は全インスタンス）
eventFilter	enumSet (WindowSwitchEventKindType)?	通知するイベント種別の集合（省略時は全種）

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
notifyHandle	NotifyHandleType	通知ハンドル

【エラーコード】

E_INVALID_ID	instanceId が有効範囲外
E_INVALID_PARAMETER	eventFilter が有効範囲外

【機能】

イベント通知対象のWindowスイッチで発生したeventFilterで指定したイベントの通知を要求する。イベントを通知するためのイベントキューを生成し，その通知ハンドルをnotifyHandleに返す。

instanceIdを省略した場合，すべてのインスタンスで発生したイベントの通知を要求する。eventFilterを省略した場合，すべての種類のイベントの通知を要求する。

unnotify

Windowスイッチイベントの通知停止

【パラメータ】

notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandle が不正

【機能】

notifyHandleで指定した通知を停止する。指定した通知用のイベントキューは削除される。

getEvent

Windowスイッチイベントの取得

【パラメータ】

notifyHandle	NotifyHandleType	イベントキューを識別する通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
event	WindowSwitchEventType	取得したイベント

【エラーコード】

E_INVALID_HANDLE	notifyHandle が不正
E_NO_DATA	イベントキューが空

【機能】

notifyHandleで指定した通知用のイベントキューから，先頭のイベントを取り出す。取り出したイベントは，イベントキューから削除される。



イベントがパラメータ等で渡されてくるプログラミング環境向けの物理APIでは，用意する必要がない。

4.18. Vehicle.Cabin.Hvac（略称：HVAC）

4.18.1. 位置付けと機能

Vehicle.Cabin.Hvac（略称：HVAC）は、車両のHVACを動かすためのマルチインスタンスオブジェクトである。

4.18.1.1. HVACの位置付けと性質

HVACとは、車室内環境を制御する車両搭載システムであり、空調ユニット、ブロワ、エアミックス機構、吹出口切替機構で構成され、乗員快適性およびデフロスト性能を実現するものと定義する。

4.18.1.2. HVACの機能

HVACは、FrontDEF状態とAuto状態を持つオブジェクトである。

FrontDEF変更を受けるとFrontDEF状態ON/OFFを遷移する。

FrontDEF状態がONのときは窓曇りを晴らす機能として、設定温度・内外気・風量・吹出口・A/C状態を窓曇りを晴らすための状態にする。

FrontDEF状態がOFFのときは空調機能として、AUTO変更を受けるとAUTO状態ON/OFFを遷移する。

AUTO状態がONのときは空調機能を自動調整する機能として、ユーザーが指定した設定温度に連動して内外気・風量・吹出口・A/C状態を快適にするための状態にする。

AUTO状態がOFFのときは空調機能を手動調整する機能として、ユーザーが指定した通りに設定温度・内外気・風量・吹出口・A/C状態を個別に状態遷移する。

4.18.1.3. HVACに対する制御の調停

HVACに対する制御の調停には、標準的な調停機能（後勝ちの原則）が適用される。すなわち、HVACの制御サービスコールが呼び出されHVACが動作中に、別の制御サービスコールが呼び出されると、最初のサービスコールによる制御はキャンセルされ、新しい制御が開始される。

4.18.2. 機能クラス

HVACは，以下の機能クラスを持つ。

AirOutletEnable	内外気変更ができる
FrontDefEnable	FrontDEFができる
AirOutletShutEnable	吹出口SHUTができる
SwingEnable	スイング機能がある

4.18.3. リスククラス

HVACは，以下のリスククラスを持つ。

RiskFog	窓曇りにより前方が見えない

4.18.4. 構成情報

HVACの各インスタンスは，以下の構成情報を持つ。

instanceId	IdType	インスタンスが持つId
placement	PlacementType	インスタンスがある場所
capabilities	enumSet (HvacCapabilityType)	そのHVACが持つ機能
riskClasses	enumSet(HvacRiskType)	そのHVACのリスククラスの集合

mountedOn	RelatedObjectType	設置対象物（車室）
displayName	String	ユーザに表示する呼称を取得するためのキー

上記は，YAML形式で記述した構成記述ファイル中に記述する形式である。構成情報を参照するサービスコールは，この構成情報に対して，次の変換を行った構成情報を返す。

- mountedOnは，物理APIで定められたデータ型に変換して返す。
- displayNameKeyに対しては，それを元に現在のロケールに対応する呼称の文字列を取得し，その文字列を返す。

4.18.5. 状態

HVACの各インスタンスは，以下の状態を持つ。

targetTemperature	Float32	設定温度
isOutsideAir	Bool	内外気
blowerLevel	Uint8	風量
airOutlet	HvacAirOutletPattern?	吹出口
isAc	Bool	A/C状態
isAuto	Bool	AUTO状態
isFrontDef	Bool	FrontDEF状態
isRearDef	Bool	RearDEF状態
isSwing	Bool	スイング状態
ambientTemperature	Float32	外気温
indoorTemperature	Float32	室内温度
indoorHumidity	Uint8	室内湿度
airOutletTemperature	Float32	吹出口温度

4.18.6. サービスコール一覧

HVACに対するサービスコールは以下の通り。

getConfigAll	すべてのHVACの構成情報の参照
getStatus	HVACの状態の参照
setTargetTemperature	設定温度変更
setOutsideAir	内外気変更
setBlowerLevel	風量変更

setAirOutlet	吹出口変更
setAc	A/C変更
setAuto	AUTO変更
setFrontDef	FrontDEF変更
setRearDef	RearDEF変更
setOff	OFF変更
setSwing	スイング変更
notify	HVACイベントの通知要求
unnotify	HVACイベントの通知停止
getEvent	HVACイベントの取得

4.18.7. イベント

HVACに対するイベント種別は以下の通り。

EventOverflow	イベントキューのオーバフロー
ConfigChanged	構成情報の変更

この内，イベント種別がEventOverflowの場合には，以下の関連情報を持つ。

noLostEvents	UInt32	失われたイベントの数

4.18.8. データ型の定義

4.18.8.1. HVACの機能クラス

```
type HvacCapabilityType = enum {
    AirOutletEnable,
    FrontDefEnable,
    AirOutletShutEnable,
    SwingEnable
};
```

4.18.8.2. HVACに対するリスククラス

```
type HvacRiskType = enum {
    RiskFog
};
```

4.18.8.3. HVACの構成情報（getConfigAllが返すデータ型）

```
type HvacConfigType = struct {
```

```
    instanceId : IdType,
    placement : PlacementType,
    capabilities : enumSet(HvacCapabilityType),
    riskClasses : enumSet(HvacRiskType),
    mountedOn : RelatedObjectType,
    displayName : String
};
```

4.18.8.4. 吹出口パターン

```
type HvacAirOutletPattern = enum {
    Shut,   // 閉
    Face,   // 頭部
    BiLevel,    // 頭部と足元
    Foot,   // 足元
    FootDef,    // 足元とフロントガラス
    Def // フロントガラス
};
```

4.18.8.5. HVAC状態

```
type HvacStatusType = struct {
    targetTemperature : Float32,
    isOutsideAir : Bool,
    blowerLevel : Uint8,
    airOutlet : HvacAirOutletPattern?,
    isAc : Bool,
    isAuto : Bool,
    isFrontDef : Bool,
    isRearDef : Bool,
    isSwing : Bool,
    ambientTemperature : Float32,
    indoorTemperature : Float32,
    indoorHumidity : Uint8,
    airOutletTemperature : Float32
};
```

4.18.8.6. HVACイベント

```
type HvacEventKindType = enum {
    EventOverflow,
    ConfigChanged
};
```

```
type HvacEventType = struct {
    instanceId : IdType,
```

```
    eventInfo : variant(HvacEventKindType) {
        EventOverflow {
            noLostEvents : UInt32
        }
    }
};
```

4.18.9. サービスコールの詳細規定

getConfigAll

すべてのHVACの構成情報の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
config	HvacConfigType[]	すべてのHVACに関する構成情報

【エラーコード】

なし

【機能】

すべてのHVACに関する構成情報を返す。

getStatus

HVACの状態の参照

【パラメータ】

instanceId	IdType	操作対象のHVACのID

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
status	HvacStatusType	対象HVACの状態

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外

【機能】

操作対象のHVACに関するすべての状態を返す。

setTargetTemperature

設定温度変更

【パラメータ】

instanceId	IdType	操作対象のHVACのID
targetTemperature	Float32	設定温度
priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_INVALID_PARAMETER	targetTemperature，priorityが有効範囲外
E_RISK_APPLICATION	アプリケーションが対応していないリスクがある
E_RISK_USER	ユーザがリスクを承認していない
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_FAULT	操作対象のHVACが故障中

【機能】

設定温度を変更する。

setOutsideAir

内外気変更

【パラメータ】

instanceId	IdType	操作対象のHVACのID
outsideAir	Bool	内外気（True:外気、False:内気）
priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_INVALID_PARAMETER	priorityが有効範囲外
E_RISK_APPLICATION	アプリケーションが対応していないリスクがある

E_RISK_USER	ユーザがリスクを承認していない
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_FAULT	操作対象のHVACが故障中

【機能】

内外気を変更する。

setBlowerLevel

風量変更

【パラメータ】

instanceId	IdType	操作対象のHVACのID
blowerLevel	Uint8	風量
priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_INVALID_PARAMETER	blowerLevel，priorityが有効範囲外
E_RISK_APPLICATION	アプリケーションが対応していないリスクがある
E_RISK_USER	ユーザがリスクを承認していない
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_FAULT	操作対象のHVACが故障中

【機能】

風量を変更する。

setAirOutlet

吹出口変更

【パラメータ】

instanceId	IdType	操作対象のHVACのID
airOutlet	HvacAirOutletPattern?	吹出口
priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_INVALID_PARAMETER	airOutlet，priorityが有効範囲外
E_RISK_APPLICATION	アプリケーションが対応していないリスクがある
E_RISK_USER	ユーザがリスクを承認していない
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_FAULT	操作対象のHVACが故障中

【機能】

吹出口を変更する。

setAc

A/C変更

【パラメータ】

instanceId	IdType	操作対象のHVACのID
ac	Bool	A/C（True:A/C ON、False:A/C OFF）
priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_INVALID_PARAMETER	priorityが有効範囲外
E_RISK_APPLICATION	アプリケーションが対応していないリスクがある
E_RISK_USER	ユーザがリスクを承認していない
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_FAULT	操作対象のHVACが故障中

【機能】

A/Cを変更する。

setAuto

AUTO変更

【パラメータ】

instanceId	IdType	操作対象のHVACのID
auto	Bool	AUTO（True:AUTO）
priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_INVALID_PARAMETER	priorityが有効範囲外
E_RISK_APPLICATION	アプリケーションが対応していないリスクがある
E_RISK_USER	ユーザがリスクを承認していない
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_FAULT	操作対象のHVACが故障中

【機能】

AUTOにする。

setFrontDef

FrontDEF変更

【パラメータ】

instanceId	IdType	操作対象のHVACのID
frontDef	Bool	FrontDEF（True:DEF、False:DEF解除）
priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_INVALID_PARAMETER	priorityが有効範囲外
E_RISK_APPLICATION	アプリケーションが対応していないリスクがある

E_RISK_USER	ユーザがリスクを承認していない
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_FAULT	操作対象のHVACが故障中

【機能】

FrontDEFを変更する。

setRearDef

RearDEF変更

【パラメータ】

instanceId	IdType	操作対象のHVACのID
rearDef	Bool	RearDEF（True:DEF、False:DEF解除）
priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_INVALID_PARAMETER	rearDef，priorityが有効範囲外
E_RISK_APPLICATION	アプリケーションが対応していないリスクがある
E_RISK_USER	ユーザがリスクを承認していない
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_FAULT	操作対象のHVACが故障中

【機能】

RearDEFを変更する。

setOff

OFF変更

【パラメータ】

instanceId	IdType	操作対象のHVACのID
blowerOff	Bool	OFF（True:OFF）
priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_INVALID_PARAMETER	priorityが有効範囲外
E_RISK_APPLICATION	アプリケーションが対応していないリスクがある
E_RISK_USER	ユーザがリスクを承認していない
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_FAULT	操作対象のHVACが故障中

【機能】

OFFにする。

setSwing

スイング変更

【パラメータ】

instanceId	IdType	操作対象のHVACのID
swing	Bool	スイング（True:ON、False:OFF）
priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_INVALID_PARAMETER	priorityが有効範囲外
E_RISK_APPLICATION	アプリケーションが対応していないリスクがある
E_RISK_USER	ユーザがリスクを承認していない
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_FAULT	操作対象のHVACが故障中

【機能】

スイングを変更する。

notify

HVACイベントの通知要求

【パラメータ】

instanceId	IdType?	イベント通知対象のHVACのID
eventFilter	enumSet (HvacEventKindType)?	通知するイベントの種類の集合

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_INVALID_PARAMETER	eventFilterが有効範囲外

【機能】

イベント通知対象のウィンドウで発生したeventFilterで指定したイベントの通知を要求する。イベントを通知するためのイベントキューを生成し，その通知ハンドルを notifyHandle に返す。

instanceIdを省略した場合，すべてのインスタンスで発生したイベントの通知を要求する。eventFilterを省略した場合，すべての種類のイベントの通知を要求する。

unnotify

HVACイベントの通知停止

【パラメータ】

notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正

【機能】

notifyHandleで指定した通知を停止する。指定した通知用のイベントキューは削除される。

getEvent

HVACイベントの取得

【パラメータ】

notifyHandle	NotifyHandleType	イベントキューを識別するための通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
event	HvacEventType	取得したイベント

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正
E_NO_DATA	イベントキューが空

【機能】

notifyHandleで指定した通知用のイベントキューから，先頭のイベントを取り出す。取り出したイベントは，イベントキューから削除される。



イベントがパラメータ等で渡されてくるプログラミング環境向けの物理APIでは，用意する必要がない。

4.19. Vehicle.Cabin.HvacSwitch（略称：HVACSwitch）

4.19.1. 位置付けと機能

HVACを制御する物理的なスイッチを扱う。スイッチとHVACは直結せず、制御はOS側（ビークルOS）で行うことを仮定する。

4.19.1.1. HVACスイッチの位置付けと性質

HVACスイッチとは、乗員が物理操作できるAC制御デバイスである。

スイッチはモーメンタリ式またはこれらの組み合わせで構成される。

HVACスイッチはヒーコンパネルに配置され、各スイッチは独立した入力系統を持つ。

スイッチとHVACは直接接続されず、制御要求はビークルOSが受け取る。

HVACスイッチは温度設定要求、風量設定要求、吹出口設定要求、内外気切替要求および空調起動停止要求を生成する入力源として機能し、HVAC制御の主体にはならない。

スイッチの操作結果は制御要求としてシステムに通知され、最終的な空調動作はHVAC制御ロジックによって決定される。

4.19.1.2. HVACスイッチの機能

HVACスイッチは乗員の操作によるスイッチ入力をOSへ提供し、HVACオブジェクトへ制御要求を伝達する役割を持つ。

各スイッチの系統はconfigData内のswitchCategoryによって識別される。

4.19.1.3. HVACスイッチに対する制御の調停

HVACスイッチは温度設定、風量設定、吹出口設定等の入力イベントの発生源であり、自身が制御競合を調停することはない。

入力はビークルOSが受け取り、HVAC制御機能へ通知する。

調停はHVAC制御機能が実施する。乗員によるスイッチ操作要求が、自動空調制御、デフロスト制御または車両保護制御と競合した場合、HVAC制御機能の優先度規則に従って採否を決定する。

HVACスイッチは制御主体ではなく、乗員要求を通知する入力源としてのみ機能する。空調状態の最終決定および維持はHVAC制御機能が担う。これは自動制御および他システムからの要求と同等の扱いである。

4.19.2. 機能クラス

HVACSwitchは，以下の機能クラスを持つ。

SwitchControl	スイッチ入力できる基本機能
ChildLockEnable	チャイルドロックできる

4.19.3. 構成情報

HVACSwitchの各インスタンスは，以下の構成情報を持つ。

instanceId	IdType	インスタンスが持つId
placement	PlacementType	インスタンスがある場所
switchCategory	enumSet (HvacSwitchCategoryType)	スイッチ系統
capabilities	enumSet (HvacSwitchCapabilityType)	そのHVACSwitchが持つ機能
displayName	String	ユーザに表示する呼称を取得するためのキー

上記は，YAML形式で記述した構成記述ファイル中に記述する形式である。構成情報を参照するサービスコールは，この構成情報に対して，次の変換を行った構成情報を返す。

- displayNameKeyに対しては，それを元に現在のロケールに対応する呼称の文字列を取得し，その文字列を返す。

4.19.4. 状態

HVACSwitchの各インスタンスは，以下の状態を持つ。

targetTemperaturePressed	Float32	設定温度絶対値指示
targetTemperatureUpPressed	Bool	設定温度Up
targetTemperatureDownPressed	Bool	設定温度Down
outsideAirPressed	Bool	内外気変更
blowerLevelPressed	Uint8	風量絶対値指示
blowerLevelUpPressed	Bool	風量Up
blowerLevelDownPressed	Bool	風量Down
airOutletPressed	HvacSwitchAirOutletPattern?	吹出口絶対値指示
airOutletUpPressed	Bool	吹出口Up
airOutletDownPressed	Bool	吹出口Down
acPressed	Bool	A/C変更
autoPressed	Bool	AUTO変更
frontDefPressed	Bool	FrontDEF変更
rearDefPressed	Bool	RearDEF変更
offPressed	Bool	OFF変更
swingPressed	Bool	スイング変更
airOutletShutPressed	Bool	吹出口SHUT変更

4.19.5. サービスコール一覧

HVACSwitchに対するサービスコールは以下の通り。

getConfigAll	すべてのHVACスイッチの構成情報の参照
getStatus	HVACスイッチの状態の参照
notify	HVACスイッチイベントの通知要求

unnotify	HVACスイッチイベントの通知停止
getEvent	HVACスイッチイベントの取得

4.19.6. イベント

HVACスイッチに関するイベントとして、スイッチ状態変化、故障検出/回復、構成変更、イベントキューのオーバーフローなどを定義する。

HVACSwitchに対するイベント種別は以下の通り。

TargetTemperatureChanged	設定温度変更の入力状態が変化
TargetTemperatureUpChanged	設定温度Upの入力状態が変化
TargetTemperatureDownChanged	設定温度Downの入力状態が変化
OutsideAirChanged	内外気変更の入力状態が変化
BlowerLevelChanged	風量変更の入力状態が変化
BlowerLevelUpChanged	風量Upの入力状態が変化
BlowerLevelDownChanged	風量Downの入力状態が変化
AirOutletChanged	吹出口変更の入力状態が変化
AirOutletUpChanged	吹出口Upの入力状態が変化
AirOutletDownChanged	吹出口Downの入力状態が変化
AcChanged	A/C変更の入力状態が変化
AutoChanged	AUTO変更の入力状態が変化
FrontDefChanged	FrontDEF変更の入力状態が変化
RearDefChanged	RearDEF変更の入力状態が変化
OffChanged	OFF変更の入力状態が変化
SwingChanged	スイング変更の入力状態が変化
AirOutletShutChanged	吹出口SHUT変更の入力状態が変化
EventOverflow	イベントキューのオーバフロー
ConfigChanged	構成情報の変更

この内，イベント種別がEventOverflowの場合には，以下の関連情報を持つ。

noLostEvents	UInt32	失われたイベントの数

4.19.7. データ型の定義

4.19.7.1. HVACスイッチの機能クラス

```
type HvacSwitchCapabilityType = enum {
    SwitchControl,
    ChildLockEnable
};
```

4.19.7.2. スイッチ系統

```
type HvacSwitchCategoryType = enum {
    FrontHVAC,  // FrontHVACスイッチ系統
    RearHVAC    // RearHVACスイッチ系統
};
```

4.19.7.3. HVACスイッチの構成情報（getConfigAllが返すデータ型）

```
type HvacSwitchConfigType = struct {
    instanceId : IdType,
    placement : PlacementType,
    switchCategory : enumSet(HvacSwitchCategoryType),
    capabilities : enumSet(HvacSwitchCapabilityType),
    displayName : String
};
```

4.19.7.4. 吹出口パターン

```
type HvacSwitchAirOutletPattern = enum {
    Face,   // 頭部
    BiLevel,    // 頭部と足元
    Foot,   // 足元
    FootDef,    // 足元とフロントガラス
    Def // フロントガラス
};
```

4.19.7.5. HVACスイッチ状態

```
type HvacSwitchStatusType = struct {
    targetTemperaturePressed : Float32,
    targetTemperatureUpPressed : Bool,
    targetTemperatureDownPressed : Bool,
    outsideAirPressed : Bool,
    blowerLevelPressed : Uint8,
    blowerLevelUpPressed : Bool,
    blowerLevelDownPressed : Bool,
    airOutletPressed : HvacSwitchAirOutletPattern?,
```

```
    airOutletUpPressed : Bool,
    airOutletDownPressed : Bool,
    acPressed : Bool,
    autoPressed : Bool,
    frontDefPressed : Bool,
    rearDefPressed : Bool,
    offPressed : Bool,
    swingPressed : Bool,
    airOutletShutPressed : Bool
};
```

4.19.7.6. HVACスイッチイベント

```
type HvacSwitchEventKindType = enum {
    TargetTemperatureChanged,
    TargetTemperatureUpChanged,
    TargetTemperatureDownChanged,
    OutsideAirChanged,
    BlowerLevelChanged,
    BlowerLevelUpChanged,
    BlowerLevelDownChanged,
    AirOutletChanged,
    AirOutletUpChanged,
    AirOutletDownChanged,
    AcChanged,
    AutoChanged,
    FrontDefChanged,
    RearDefChanged,
    OffChanged,
    SwingChanged,
    AirOutletShutChanged,
    EventOverflow,
    ConfigChanged
};
```

```
type HvacSwitchEventType = struct {
    instanceId : IdType,
    eventInfo : variant(HvacSwitchEventKindType) {
        EventOverflow {
            noLostEvents : UInt32
        }
    }
};
```

4.19.8. サービスコールの詳細規定

getConfigAll

すべてのHVACスイッチの構成情報の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
config	HvacSwitchConfigType[]	すべてのHVACスイッチに関する構成情報

【エラーコード】

なし

【機能】

すべてのHVACスイッチに関する構成情報を返す。

getStatus

HVACスイッチの状態の参照

【パラメータ】

instanceId	IdType	操作対象のHVACスイッチのID

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
status	HvacSwitchStatusType	対象HVACスイッチの状態

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外

【機能】

操作対象のHVACスイッチに関するすべての状態を返す。

notify

HVACスイッチイベントの通知要求

【パラメータ】

instanceId	IdType?	イベント通知対象のHVACスイッチのID

eventFilter	enumSet (HvacSwitchEventKindType)?	通知するイベントの種類の集合

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_INVALID_PARAMETER	eventFilterが有効範囲外

【機能】

イベント通知対象のHVACスイッチで発生したeventFilterで指定したイベントの通知を要求する。イベントを通知するためのイベントキューを生成し，その通知ハンドルを notifyHandle に返す。

instanceIdを省略した場合，すべてのインスタンスで発生したイベントの通知を要求する。eventFilterを省略した場合，すべての種類のイベントの通知を要求する。

unnotify

HVACスイッチイベントの通知停止

【パラメータ】

notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正

【機能】

notifyHandleで指定した通知を停止する。指定した通知用のイベントキューは削除される。

getEvent

HVACスイッチイベントの取得

【パラメータ】

notifyHandle	NotifyHandleType	イベントキューを識別するための通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
event	HvacSwitchEventType	取得したイベント

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正
E_NO_DATA	イベントキューが空

【機能】

notifyHandleで指定した通知用のイベントキューから，先頭のイベントを取り出す。取り出したイベントは，イベントキューから削除される。



イベントがパラメータ等で渡されてくるプログラミング環境向けの物理APIでは，用意する必要がない。

4.20. Vehicle.CargoSpace（略称：CargoSpace）

＜TBD＞

4.21. Vehicle.CargoSpace.Trunk（略称：Trunk）

4.21.1. 位置付けと機能

Vehicle.CargoSpace.Trunk（略称：Trunk）は，車両のトランク（荷室）を操作するためのマルチインスタンスオブジェクトである。

4.21.1.1. トランクの位置付けと性質

開閉できる車室の開口部の中で，車両の走行中は通常閉じておくもの。または，大きい荷物を出し入れすることを想定しているもの。人が通ることは想定していない。

Trunkの機能

Trunkは，可動部を一次元に移動できるオブジェクトであり，MovableObjectの機能を持つ。

可動部の位置は，いずれの方向の移動に対しても，どれだけ開いているかのパーセンテージ（開閉度，0：全閉，100：全開）で表す。また，0に近づく動作を閉動作，100に近づく動作を開動作と呼ぶ。

挟み込み防止機能を持つトランクは，可動部の閉動作中に挟み込みを検知すると，閉動作を停止し，開動作を開始する。挟み込み検知後にどの位置まで開くかは，車両依存である。

可動部が停止中の間も，マニュアル操作にて現在位置が変化することがあり得る。

4.21.1.2. トランクに対する制御の調停

トランクに対する制御の調停には，標準的な調停機能（後勝ちの原則）が適用される。すなわち，トランクを閉めるサービスコールが呼び出され，トランクが閉じる動作中に，開くサービスコールが呼び出されると，最初のサービスコールによる制御はキャンセルされ，トランクは開く動作を始める。

また，トランクは標準的なロック機能を持つLockableObjectである。トランクには，機械的なロック機構は無いのが一般的であり，ソフトウェアでロック機能を実現することを想定している。

トランクが故障または挟み込みを検知した場合，ロックは解除され，故障中と挟み込み回避中は，ロックを取得することができない。

4.21.2. 機能クラス

Trunkは，以下の機能クラスを持つ。

Movable	パワートランクであり，APIにより移動させることができる。トランクロックアンラッチ機能のみを有する者も含む
ClosedDetectable	トランクが全閉状態であることを検出することができる
MoveProfileSupported	移動プロファイルをサポートしている

4.21.3. リスククラス

Trunkは，以下のリスククラスを持つ。

RiskPinch	トランクを閉めることに伴う挟み込みのリスク
RiskOpen	トランクを開けることに伴う諸々のリスク

【今後の検討課題】

RiskOpenをさらに分類するかは，今後の課題である。

4.21.4. 構成情報

Trunkの各インスタンスは，以下の構成情報を持つ。

instanceId	IdType	インスタンスが持つId
placement	PlacementType	インスタンスがある場所
capabilities	enumSet (TrunkCapabilityType)	そのトランクが持つ機能
riskClasses	enumSet(TrunkRiskType)	そのトランクのリスククラスの集合
mountedOn	RelatedObjectType	設置対象物（ドアまたは車室）
displayName	String	ユーザに表示する呼称を取得するためのキー

上記は，YAML形式で記述した構成記述ファイル中に記述する形式である。構成情報を参照するサービスコールは，この構成情報に対して，次の変換を行った構成情報を返す。

- mountedOnは，物理APIで定められたデータ型に変換して返す。
- displayNameKeyに対しては，それを元に現在のロケールに対応する呼称の文字列を取得し，その文字列を返す。

4.21.5. 状態

Trunkの各インスタンスは，以下の状態を持つ。

mainStatus	TrunkMainStatusType	主状態
lockStatus	LockStatusType	ロック状態
position	PositionType	現在位置
targetPosition	PositionType	目標位置

トランクの主状態（mainStatus）は，MovableObjectの主状態に加えて，挟み込み回避中の状態をとる。また，位置のパーセンテージが大きくなる方へ移動中を開動作中，位置のパーセンテージが小さくなる方へ移動中を閉動作中と呼ぶ。

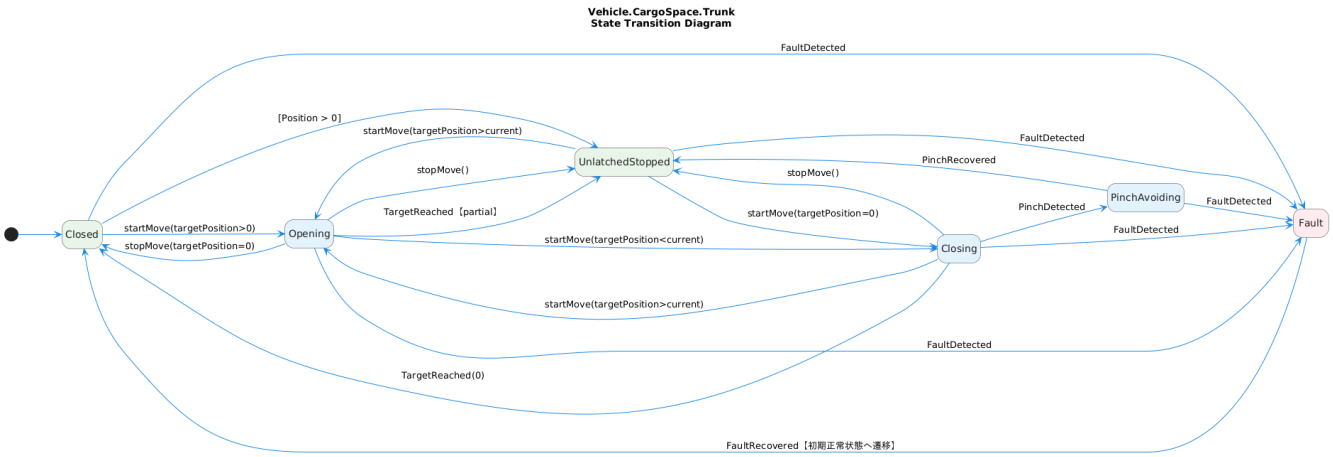


Figure 16. トランクの主状態の状態遷移

トランクの現在位置（position）と目標位置（targetPosition）は，MovableObjectの現在位置と目標位置に関する規定に従う。

トランクロック/アンラッチのみの機能の場合は，開要求のみを受け付け，閉要求は受け付けないものとする。また，アンラッチ完了時をもって目標位置到達と考える。

4.21.6. サービスコール一覧

Trunkに対するサービスコールは以下の通り。

getConfigAll	すべてのトランクの構成情報の参照
getStatus	トランクの状態の参照
startMove	トランクの開閉の開始
stopMove	トランクの開閉の停止
lock	トランクのロックの取得
unlock	トランクのロックの解除
notify	トランクイベントの通知要求
unnotify	トランクイベントの通知停止
getEvent	トランクイベントの取得

4.21.7. イベント

トランクに対するイベント種別には，MovableObjectの持つイベントとLockableObjectが持つイベントに加えて，挟み込み検知と挟み込みから回復がある。

Trunkに対するイベント種別は以下の通り。

ControlledBySelf	自アプリケーションが制御
ControlledByOther	他のアプリケーションが制御
TargetReached	指定された目標位置に到達して停止，または，移動停止が完了
FaultDetected	故障検知
FaultRecovered	故障から回復
PinchDetected	挟み込み検知
PinchRecovered	挟み込みから回復
OwnLockRevoked	自アプリケーションのロックの強制解除
OtherLockReleased	他のアプリケーション等によるロック解除
EventOverflow	イベントキューのオーバフロー
ConfigChanged	構成情報の変更

この内，イベント種別がControlledByOtherの場合には，以下の関連情報を持つ。

sourceApplicatio n	ApplicationIdType	操作したアプリケーション

また，イベント種別がEventOverflowの場合には，以下の関連情報を持つ。

noLostEvents	UInt32	失われたイベントの数

4.21.8. データ型の定義

4.21.8.1. トランクの機能クラス

```
type TrunkCapabilityType = enum {
    Movable,
    ClosedDetectable,
    MoveProfileSupported
};
```

4.21.8.2. トランクに対するリスククラス

```
type TrunkRiskType = enum {
    RiskPinch,
    RiskOpen
```

```
};
```

4.21.8.3. トランクの構成情報（getConfigAllが返すデータ型）

```
type TrunkConfigType = struct {
    instanceId : IdType,
    placement : PlacementType,
    capabilities : enumSet(TrunkCapabilityType),
    riskClasses : enumSet(TrunkRiskType),
    mountedOn : RelatedObjectType,
    displayName : String
};
```

4.21.8.4. トランク状態

```
type TrunkMainStatusType = enum {
    FullyStopped,   // 停止中（全閉）
    UnlatchedStopped,   // 停止中（半ドア  or  ドア開）
    Opening,    // 開動作中
    Closing,    // 閉動作中
    Fault,  // 故障中
    PinchAvoiding   // 挟み込み回避中
};
```

```
type TrunkStatusType = struct {
    mainStatus : TrunkMainStatusType,
    lockStatus : LockStatusType,
    position : PositionType,
    targetPosition : PositionType
};
```

4.21.8.5. 移動プロファイル

```
type TrunkMoveProfileType = enum {
    Standard,   // 標準的な移動
    Fast,   // 高速で移動
    Slow    // 低速で移動
};
```

4.21.8.6. トランクイベント

```
type TrunkEventKindType = enum {
    ControlledBySelf,
    ControlledByOther,
```

```
    TargetReached,
    FaultDetected,
    FaultRecovered,
    PinchDetected,
    PinchRecovered,
    OwnLockRevoked,
    OtherLockReleased,
    EventOverflow,
    ConfigChanged
};
```

```
type TrunkEventType = struct {
    instanceId : IdType,
    eventInfo : variant(TrunkEventKindType) {
        ControlledByOther {
            sourceApplication : ApplicationIdType
        },
        EventOverflow {
            noLostEvents : UInt32
        }
    }
};
```

4.21.9. サービスコールの詳細規定

getConfigAll

すべてのトランクの構成情報の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
config	TrunkConfigType[]	すべてのトランクに関する構成情報

【エラーコード】

なし

【機能】

すべてのトランクに関する構成情報を返す。

getStatus

トランクの状態の参照

【パラメータ】

instanceId	IdType	操作対象のトランクのID

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
status	TrunkStatusType	対象トランクの状態

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外

【機能】

操作対象のトランクに関するすべての状態を返す。

startMove

トランクの開閉の開始

【パラメータ】

instanceId	IdType	操作対象のトランクのID
targetPosition	PositionType	目標位置（開閉度）
moveProfile	TrunkMoveProfileType?	移動プロファイル
priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_INVALID_PARAMETER	targetPosition，priorityが有効範囲外
E_RISK_APPLICATION	アプリケーションが対応していないリスクがある
E_RISK_USER	ユーザがリスクを承認していない
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_FAULT	操作対象のトランクが故障中
E_OBJECT_STATUS	操作対象のトランクが挟み込み回避中
E_OBJECT_LOCKED	操作対象のトランクが，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている

【機能】

目標位置をtargetPositionに設定し，操作対象のトランクの開閉動作を開始する。

targetPositionに0を指定すれば閉動作の開始，100を指定すれば開動作の開始の意味になる。

アンラッチ機能のみを有するトランクの場合は，targetPositionに0より大きい値にて要求された場合,アンラッチ動作を実施。（完全なトランク開は保証できない）

リスククラスRiskPinchに対応できていない場合でも，トランクを開く方向に動作を開始しようとした場合には，操作を受け付ける。そうでない場合には，E_RISK_APPLICATIONエラーまたはE_RISK_USERエラーとなる。

リスククラスRiskOpenに対応できていない場合でも，トランクを閉じる方向に動作を開始しようとした場合には，操作を受け付ける。そうでない場合には，E_RISK_APPLICATIONエラーまたはE_RISK_USERエラーとなる。

TrunkMoveProfileTypeは，標準，高速，低速を選択可能であるが，制御対象の性能差により絶対速度の保証は困難である。具体的な速度に関しては，各制御対象ごとで決定することを想定している。

stopMove

トランクの開閉の停止

【パラメータ】

instanceId	IdType	操作対象のトランクのID
priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_INVALID_PARAMETER	priorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_FAULT	操作対象のトランクが故障中
E_OBJECT_STATUS	操作対象のトランクが挟み込み回避中
E_OBJECT_LOCKED	操作対象のトランクが，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている

【機能】

各次元方向の目標位置を現在位置に設定して，操作対象のトランクの各次元方向への開閉動作の停止を要求する。

lock

トランクのロックの取得

【パラメータ】

instanceId	IdType	操作対象のトランクのID

priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_INVALID_PARAMETER	priorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_LOCKED	ロック対象のトランクが，priorityと同じかそれより高い優先度でロックされている
E_OBJECT_FAULT	ロック対象のトランクが故障中
E_OBJECT_STATUS	ロック対象のトランクが挟み込み回避中

【機能】

ロック対象のトランクを，指定した操作優先度でロックする。

unlock

トランクのロックの解除

【パラメータ】

instanceId	IdType	操作対象のトランクのID

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_OBJECT_STATUS	自アプリケーションが，操作対象のトランクをロックしていない

【機能】

ロック解除対象のトランクを，ロック解除する。

notify

トランクイベントの通知要求

【パラメータ】

instanceId	IdType?	イベント通知対象のトランクのID

eventFilter	enumSet (TrunkEventKindType)?	通知するイベントの種類の集合

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_INVALID_PARAMETER	eventFilterが有効範囲外

【機能】

イベント通知対象のトランクで発生したeventFilterで指定したイベントの通知を要求する。イベントを通知するためのイベントキューを生成し，その通知ハンドルを notifyHandle に返す。

instanceIdを省略した場合，すべてのインスタンスで発生したイベントの通知を要求する。eventFilterを省略した場合，すべての種類のイベントの通知を要求する。

unnotify

トランクイベントの通知停止

【パラメータ】

notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正

【機能】

notifyHandleで指定した通知を停止する。指定した通知用のイベントキューは削除される。

getEvent

トランクイベントの取得

【パラメータ】

notifyHandle	NotifyHandleType	イベントキューを識別するための通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
event	TrunkEventType	取得したイベント

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正
E_NO_DATA	イベントキューが空

【機能】

notifyHandleで指定した通知用のイベントキューから，先頭のイベントを取り出す。取り出したイベントは，イベントキューから削除される。



イベントがパラメータ等で渡されてくるプログラミング環境向けの物理APIでは，用意する必要がない。

4.22. Vehicle.CargoSpace.TrunkSwitch（略称：TrunkSwitch）

4.22.1. 位置付けと機能

Trunkを制御する物理的なスイッチを扱う。スイッチとTrunkは直結せず，制御はOS側（ビークルOS）で行うことを仮定する。

4.22.1.1. Trunkスイッチの位置付けと性質

Trunkスイッチとは，各Trunkに配置された乗員が物理操作できるTrunk制御デバイスである。

また，フットセンサによって足で操作し，自動で開閉するケースもある。

スイッチとTrunkは直接接続されず，制御要求はビークルOSが受け取る。

スイッチは入力源として機能し，Trunk制御の主体にはならない。

4.22.1.2. Trunkスイッチの機能

Trunkスイッチは乗員の操作によるスイッチ入力をOSへ提供し，Trunkオブジェクトへ制御要求を伝達する役割を持つ。

各スイッチの系統はconfigData内のswitchCategoryによって識別される。

4.22.1.3. Trunkスイッチに対する制御の調停

Trunkスイッチは入力イベントの発生源であり，自身が制御競合を調停することはない。

入力はOSが一元管理し，Trunkオブジェクトへルーティングされる。

調停はTrunk側の後勝ち原則に従う。スイッチ操作が他アプリケーションの制御要求と競合した場合，Trunk側の調停ロジックにより採用または却下が判断される。

スイッチ自身は制御主体ではなく，アプリケーションや自動制御と同様に「入力の一つ」として扱われる。

4.22.2. 機能クラス

TrunkSwitchは，以下の機能クラスを持つ。

SwitchControl	Trunk操作入力を提供する基本機能

4.22.3. 構成情報

TrunkSwitchの各インスタンスは，以下の構成情報を持つ。

instanceId	IdType	インスタンスが持つId
placement	PlacementType	インスタンスがある場所
switchCategory	TrunkSwitchCategoryType	スイッチ系統（Front Rear）
capabilities	enumSet (TrunkSwitchCapabilityType)	このスイッチがサポートする操作機能の集合
switchType	String	スイッチ種別（Open，Close，など，OEM依存）
displayName	String	ユーザに表示する呼称を取得するためのキー

上記は，YAML形式で記述した構成記述ファイル中に記述する形式である。構成情報を参照するサービスコールは，この構成情報に対して，次の変換を行った構成情報を返す。

- displayNameKeyに対しては，それを元に現在のロケールに対応する呼称の文字列を取得し，その文字列を返す。

4.22.4. 状態

TrunkSwitchの各インスタンスは，以下の状態を持つ。

mainStatus	TrunkSwitchMainStatusType	スイッチの入力状態（Open，Closeなど）

4.22.5. サービスコール一覧

TrunkSwitchに対するサービスコールは以下の通り。

getConfigAll	すべてのTrunkスイッチの構成情報の参照
getStatus	Trunkスイッチの状態参照
notify	Trunkスイッチイベントの通知要求
unnotify	Trunkスイッチイベントの通知停止
getEvent	Trunkスイッチイベントの取得

4.22.6. イベント

Trunkスイッチに関するイベントとして，スイッチ状態変化，故障検出/回復，構成変更，イベントキューのオーバーフローなどを定義する。

TrunkSwitchに対するイベント種別は以下の通り。

MainStatusChanged	Front/RearスイッチのmainStatus入力状態が変化
FaultDetected	故障検知
FaultRecovered	故障から回復
EventOverflow	イベントキューのオーバフロー
ConfigChanged	構成情報の変更

この内，イベント種別がMainStatusChangedの場合には，以下の関連情報を持つ。

newState	TrunkSwitchMainStatusType	スイッチの新しい入力状態

また，イベント種別がEventOverflowの場合には，以下の関連情報を持つ。

noLostEvents	UInt32	失われたイベントの数

4.22.7. データ型の定義

4.22.7.1. Trunkスイッチの機能クラス

```
type TrunkSwitchCapabilityType = enum {
    SwitchControl
};
```

4.22.7.2. スイッチ系統

```
type TrunkSwitchCategoryType = enum {
    Front,  // FrontTrunkスイッチ系統
    Rear    // RearTrunkスイッチ系統
};
```

4.22.7.3. Trunksスイッチの入力状態

```
type TrunkSwitchMainStatusType = enum {
    False,  // OFF - Trunk停止
    Open,   // Open - TrunkOpen
    Close   // Close - TrunkClose
};
```

Trunkスイッチ構成情報（getConfigAll が返すデータ型）

```
type TrunkSwitchConfigType = struct {
    instanceId : IdType,
    placement : PlacementType,
    switchCategory : TrunkSwitchCategoryType,
    capabilities : enumSet(TrunkSwitchCapabilityType),
    switchType : String,
    displayName : String
};
```

4.22.7.4. Trunkスイッチ状態

```
type TrunkSwitchStatusType = struct {
    mainStatus : TrunkSwitchMainStatusType
};
```

Trunkスイッチの状態情報

4.22.7.5. Trunkスイッチイベント種別

```
type TrunkSwitchEventKindType = enum {
    MainStatusChanged,
    FaultDetected,
    FaultRecovered,
    EventOverflow,
    ConfigChanged
};
```

```
type TrunkSwitchEventType = struct {
    instanceId : IdType,
    switchCategory : TrunkSwitchCategoryType,
    eventInfo : variant(TrunkSwitchEventKindType) {
        MainStatusChanged {
            newState : TrunkSwitchMainStatusType
        },
        EventOverflow {
            noLostEvents : UInt32
        }
    }
};
```

4.22.8. サービスコールの詳細規定

getConfigAll

すべてのTrunkスイッチの構成情報の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
config	TrunkSwitchConfigType[]	すべてのTrunkスイッチに関する構成情報

【機能】

すべてのTrunkスイッチに関する構成情報を返す。

getStatus

Trunkスイッチの状態参照

【パラメータ】

instanceId	IdType	操作対象のTrunkスイッチのID

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
status	TrunkSwitchStatusType	対象のTrunkスイッチの状態

【エラーコード】

E_INVALID_ID	instanceId が有効範囲外

【機能】

操作対象のTrunkスイッチに関する状態を返す。



status には mainStatus（Front/Rear系統のスイッチ入力状態）を含む。

notify

Trunkスイッチイベントの通知要求

【パラメータ】

instanceId	IdType?	イベント通知対象のTrunkスイッチID（省略時は全インスタンス）

eventFilter	enumSet (TrunkSwitchEventKindType)?	通知するイベント種別の集合（省略時は全種）

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
notifyHandle	NotifyHandleType	通知ハンドル

【エラーコード】

E_INVALID_ID	instanceId が有効範囲外
E_INVALID_PARAMETER	eventFilter が有効範囲外

【機能】

イベント通知対象のTrunkスイッチで発生したeventFilterで指定したイベントの通知を要求する。イベントを通知するためのイベントキューを生成し，その通知ハンドルをnotifyHandleに返す。

instanceIdを省略した場合，すべてのインスタンスで発生したイベントの通知を要求する。eventFilterを省略した場合，すべての種類のイベントの通知を要求する。

unnotify

Trunkスイッチイベントの通知停止

【パラメータ】

notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandle が不正

【機能】

notifyHandleで指定した通知を停止する。指定した通知用のイベントキューは削除される。

getEvent

Trunkスイッチイベントの取得

【パラメータ】

notifyHandle	NotifyHandleType	イベントキューを識別する通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
event	TrunkSwitchEventType	取得したイベント

【エラーコード】

E_INVALID_HANDLE	notifyHandle が不正
E_NO_DATA	イベントキューが空

【機能】

notifyHandleで指定した通知用のイベントキューから，先頭のイベントを取り出す。取り出したイベントは，イベントキューから削除される。



イベントがパラメータ等で渡されてくるプログラミング環境向けの物理APIでは，用意する必要がない。

4.23. Vehicle.Body（略称：Body）

＜TBD＞

4.24. Vehicle.Body.Hood（略称：Hood）

4.24.1. 位置付けと機能

Vehicle.Body.Hood（略称：Hood）は，車両のボンネットを操作するためのシングルトンオブジェクトである。

4.24.1.1. ボンネットの位置付けと性質

ボンネットとは，エンジンルーム(フロントコンパートメント)を覆う外板であり，車両の走行中は閉じていることを前提として定義する。ボンネットを開けるためのオープナーは，ボンネットとは別のオブジェクトとして扱う。

ボンネットは，閉じている状態ではエンジン等の内部機器を外部環境から保護するとともに，衝突時の衝撃緩和，空力性能，熱対策，騒音低減および車体構造の補助といった車両性能・安全性能の一部を担う役割を持つ。

開いている状態では，内部の点検・整備，部品交換および診断作業を行うためのアクセスを可能にし，車両の保守性および維持管理性を確保する役割を持つ。

4.24.1.2. Hoodの機能

HoodのAPI制御機能は，全閉・ラッチ状態からラッチ解除とポップアップを一連の動作として要求するstartPopup()に限定する。

startPopup()はラッチ解除だけ，または任意位置への移動を要求するAPIではなく，ラッチ解除と所定のポップアップ位置への到達の両方を要求する。

APIによる閉動作，任意位置への移動，および全開動作は提供しない。自動閉鎖も行わない。

ポップアップ完了後にボンネットをさらに開く動作，およびポップアップ状態から閉じて再ラッチする動作は手動で行うことを想定する。

手動でさらに開かれた場合も，API上はポップアップ完了後の状態として扱う。手動で閉じてラッチが再係合し，全閉を検出した時点でHoodはClosed状態に戻る。

Hoodは固定されたポップアップ動作のみを提供するため，任意の現在位置・目標位置を扱うMovableObjectとしては定義しない。

4.24.2. 機能クラス

Hoodは，以下の機能クラスを持つ。

PopupControl	APIによりラッチ解除と所定位置までのポップアップを一連の動作として要求できる
ClosedDetectable	ボンネットが全閉かつ再ラッチされた状態であることを検出できる

4.24.3. リスククラス

Hoodは，以下のリスククラスを持つ。

RiskOpen	ボンネットを開ける／開いていることに伴う諸々のリスク

【今後の検討課題】

RiskOpenをさらに分類するかは，今後の課題である。

4.24.4. 構成情報

Hoodの各インスタンスは，以下の構成情報を持つ。

capabilities	enumSet (HoodCapabilityType)	ボンネットが持つ機能
riskClasses	enumSet(HoodRiskType)	ボンネットのリスククラスの集合
mountedOn	RelatedObjectType	設置対象物（ドアまたは車室）
displayName	String	ユーザに表示する呼称を取得するためのキー

上記は，YAML形式で記述した構成記述ファイル中に記述する形式である。構成情報を参照するサービスコールは，この構成情報に対して，次の変換を行った構成情報を返す。

- mountedOnは，物理APIで定められたデータ型に変換して返す。
- displayNameKeyに対しては，それを元に現在のロケールに対応する呼称の文字列を取得し，その文字列を返す。

4.24.5. 状態

Hoodの各インスタンスは，以下の状態を持つ。

mainStatus	HoodMainStatusType	主状態（全閉・ラッチ，ポップアップ処理中，ポップアップ完了，故障）
priority	PriorityType?	実行中のポップアップ要求に適用されている操作優先度。要求がない場合は未設定

HoodはMovableObjectとして定義せず，現在位置および目標位置を公開しない。

mainStatus=Closedは，ボンネットが全閉でありラッチが係合していることを表す。

startPopup()の要求を正常に受け付けた後はmainStatus=PoppingUpとなり，ラッチ解除とポップアップを実行する。

ラッチ解除が完了し，かつ所定のポップアップ位置への到達が完了した時点でmainStatus=Popupとなり，PopupReachedを通知する。

ポップアップ完了後に手動でさらに開いた場合，および手動で閉じている途中は，全閉・再ラッチが検出されるまでmainStatus=Popupのままとする。

手動で全閉しラッチが再係合したことを検出した時点でmainStatus=Closedに戻る。APIによる自動閉鎖は行わない。

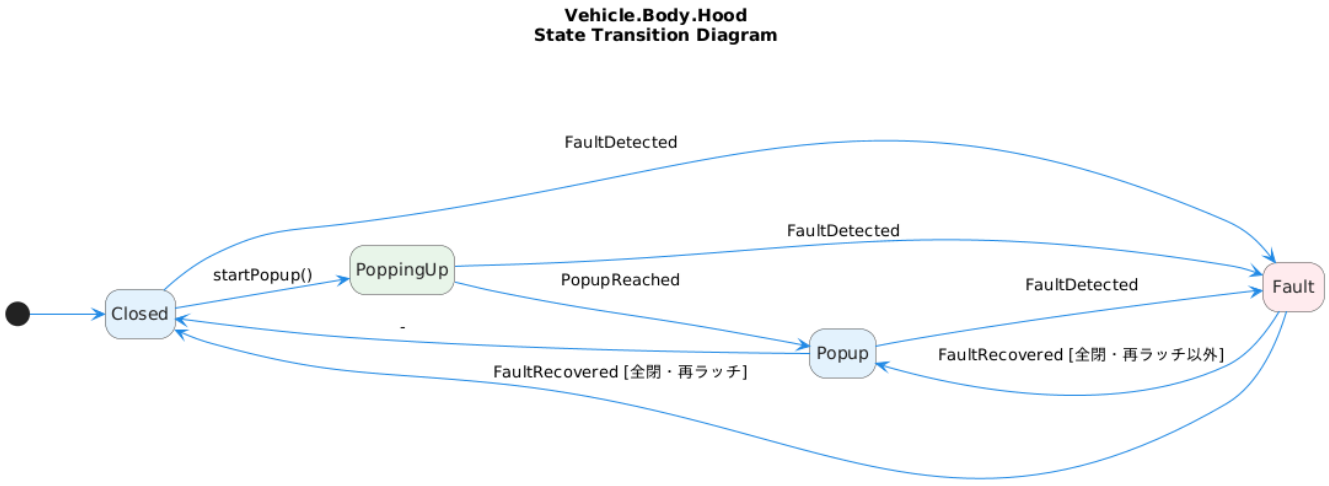


Figure 17. ボンネットの主状態の状態遷移

4.24.6. サービスコール一覧

Hoodに対するサービスコールは以下の通り。

getConfig	ボンネットの構成情報の参照
getStatus	ボンネットの状態の参照
startPopup	ボンネットのポップアップ開始
notify	ボンネットイベントの通知要求
unnotify	ボンネットイベントの通知停止
getEvent	ボンネットイベントの取得

4.24.7. イベント

ボンネットに対するイベント種別として，制御主体の変更，ポップアップ完了，故障検知/回復，構成変更，イベントキューのオーバーフローなどを定義する。

PopupReachedは，startPopup()要求に対するラッチ解除とポップアップの両方が完了したことを示す完了イベントである。

Hoodに対するイベント種別は以下の通り。

ControlledBySelf	自アプリケーションが制御
ControlledByOther	他のアプリケーションが制御
PopupReached	startPopup()で要求したラッチ解除と所定のポップアップ位置への到達が両方とも正常完了し，mainStatus=Popupとなったときに通知する。要求受付時，ラッチ解除のみの完了時，手動開放時，手動閉操作時には通知しない
FaultDetected	故障検知
FaultRecovered	故障から回復
EventOverflow	イベントキューのオーバフロー
ConfigChanged	構成情報の変更

この内，イベント種別がControlledByOtherの場合には，以下の関連情報を持つ。

sourceApplication	ApplicationIdType	操作したアプリケーション

また，イベント種別がEventOverflowの場合には，以下の関連情報を持つ。

noLostEvents	UInt32	失われたイベントの数

4.24.8. データ型の定義

4.24.8.1. ボンネットの機能クラス

```
type HoodCapabilityType = enum {
    PopupControl,
    ClosedDetectable
};
```

4.24.8.2. ボンネットに対するリスククラス

```
type HoodRiskType = enum {
    RiskOpen
};
```

4.24.8.3. ボンネットの構成情報（getConfigが返すデータ型）

```
type HoodConfigType = struct {
    capabilities : enumSet(HoodCapabilityType),
    riskClasses : enumSet(HoodRiskType),
    mountedOn : RelatedObjectType,
    displayName : String
};
```

4.24.8.4. ボンネットの状態

```
type HoodMainStatusType = enum {
    Closed, // 全閉かつラッチ係合状態
    PoppingUp,  // startPopup()要求受付後，ラッチ解除またはポップアップが完了していない状態
    Popup,  // ラッチ解除と所定のポップアップ位置への到達が両方とも完了した状態。手動でさらに開かれた場合も再ラッチされるまでは本状態として扱う
    Fault   // 故障中
};
```

4.24.8.5. ボンネットの状態情報（getStatusが返すデータ型）

```
type HoodStatusType = struct {
    mainStatus : HoodMainStatusType,
    priority : PriorityType?
};
```

4.24.8.6. ボンネットイベント種別

```
type HoodEventKindType = enum {
    ControlledBySelf,
    ControlledByOther,
    PopupReached,
    FaultDetected,
    FaultRecovered,
    EventOverflow,
    ConfigChanged
};
```

```
type HoodEventType = struct {
    eventInfo : variant(HoodEventKindType) {
        ControlledByOther {
            sourceApplication : ApplicationIdType
        },
        EventOverflow {
```

```
            noLostEvents : UInt32
        }
    }
};
```

4.24.9. サービスコールの詳細規定

getConfig

ボンネットの構成情報の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
config	HoodConfigType	ボンネットに関する構成情報

【エラーコード】

E_OK	成功
E_NOT_SUPPORTED	未サポート（実装により返却されない場合あり）

【機能】

シングルトンのボンネットに関する構成情報をHoodConfigTypeで返す。

getStatus

ボンネットの状態の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
status	HoodStatusType	ボンネットの状態

【エラーコード】

E_OK	成功

【機能】

シングルトンのボンネットに関するすべての状態を返す。

startPopup

ボンネットのポップアップ開始

【パラメータ】

priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_PARAMETER	priorityが有効範囲外
E_RISK_APPLICATION	アプリケーションが対応していないリスクがある
E_RISK_USER	ユーザがリスクを承認していない
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_FAULT	操作対象のボンネットが故障中
E_OBJECT_STATUS	ボンネットが全閉・ラッチ状態ではないためポップアップを開始できない
E_OBJECT_LOCKED	操作対象のボンネットが，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている

【機能】

シングルトンのボンネットに対し，ラッチ解除と所定位置までのポップアップを一連の動作として要求する。

呼出し可能な開始状態はmainStatus=Closedの全閉・ラッチ状態に限定する。

E_OKは要求を受け付けたことを示し，ラッチ解除またはポップアップの完了を示すものではない。要求受付後，mainStatus=PoppingUpとしてラッチ解除とポップアップを実行する。

ラッチ解除と所定のポップアップ位置への到達が両方とも正常完了した時点でmainStatus=Popupへ遷移し，その要求に対してPopupReachedイベントを通知する。ラッチ解除のみが完了した時点ではPopupReachedを通知しない。

APIは任意の目標位置を受け付けず，閉動作，全開動作，および自動閉鎖を行わない。ポップアップ完了後の全開および再閉鎖・再ラッチは手動操作による。

完了前に故障等でポップアップ動作を完了できなかった場合はPopupReachedを通知しない。

notify

ボンネットイベントの通知要求

【パラメータ】

eventType	HoodEventKindType?	通知するイベントの種類の集合

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【エラーコード】

E_INVALID_PARAMETER	eventTypeが有効範囲外

【機能】

シングルトンのボンネットで発生したeventTypeで指定したイベントの通知を要求する。イベントを通知するためのイベントキューを生成し，その通知ハンドルをnotifyHandleに返す。

eventTypeを省略した場合，すべての種類のイベントの通知を要求する。

unnotify

ボンネットイベントの通知停止

【パラメータ】

notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正

【機能】

notifyHandleで指定した通知を停止する。指定した通知用のイベントキューは削除される。

getEvent

ボンネットイベントの取得

【パラメータ】

notifyHandle	NotifyHandleType	イベントキューを識別するための通知ハンドル
eventDetail	EventDetailType	イベント詳細情報

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正
E_NO_DATA	イベントキューが空

【機能】

notifyHandleで指定した通知用のイベントキューから，先頭のイベントを取り出す。取り出したイベントは，イベントキューから削除される。



イベントがパラメータ等で渡されてくるプログラミング環境向けの物理APIでは，用意する必要がない。

4.25. Vehicle.Cabin.HoodSwitch（略称：HoodSwitch）

4.25.1. 位置付けと機能

Vehicle.Cabin.HoodSwitch（略称：HoodSwitch）は，ボンネットのポップアップ操作に用いる物理的なオープナー／スイッチ入力を扱うシングルトンオブジェクトである。

HoodSwitchは入力デバイスであり，Hoodを直接駆動しない。入力検知後のHood制御要求はOS側（ビークルOS）が行う。

4.25.1.1. HoodSwitchの位置付けと性質

HoodSwitchは，運転者によるオープナー操作の押下／解除状態を検知してOSへ通知する。

HoodSwitchとHoodは直接接続されず，HoodSwitchのPressedイベントはHoodのポップアップ開始または完了を意味しない。

OSはPressedイベントを受信した後，構成情報のvehicleStateConditionを評価し，条件を満たす場合にrelatedHoodで指定されたHoodのstartPopup()を呼び出す。

vehicleStateConditionを満たさない場合，OSはHoodへのポップアップ要求を行わない。

4.25.1.2. HoodSwitchの機能

HoodSwitchは，物理入力をデバウンス処理した上でPressed/Released状態として保持し，状態遷移をイベントとして通知する。

ReleasedからPressedへの有効な遷移時にPressedイベントを通知し，PressedからReleasedへの有効な遷移時にReleasedイベントを通知する。

Pressedイベントはオープナー操作を検知したことのみを表し，Hoodのラッチ解除，ポップアップ開始，またはポップアップ完了を表さない。

Hoodの動作完了はHoodオブジェクトのPopupReachedイベントで確認する。

4.25.2. 機能クラス

HoodSwitchは，以下の機能クラスを持つ。

SwitchControl	Hood操作用の物理入力を検知し，Pressed/Released状態およびイベントを提供する

4.25.3. 構成情報

HoodSwitchの各インスタンスは，以下の構成情報を持つ。

displayName	String	ユーザに表示する呼称を取得するためのキー
relatedHood	RelatedObjectType	このHoodSwitchの操作に対応するHood。OSは受付条件を満たすPressedイベントに対して，このHoodのstartPopup()を呼び出す
interactionType	String	HoodSwitchの物理的な操作方式を表す構成値。許容値は共通規定または車両構成で定義する
vehicleStateCondition	String	PressedイベントをHoodのポップアップ要求として受け付けるためにOSが評価する車両状態条件。具体的な条件式・条件値は車両構成で定義する
debounceMs	UInt32	物理入力のチャタリングを除去するため，入力状態が連続して安定している必要がある時間[ms]

上記は，YAML形式で記述した構成記述ファイル中に記述する形式である。

- relatedHoodは，物理APIで定められた関連オブジェクト型に変換して返す。
- displayNameKeyに対しては，それを元に現在のロケールに対応する呼称の文字列を取得し，その文字列を返す。
- vehicleStateConditionはHoodSwitch自身がHoodを駆動する条件ではなく，OSがPressedイベント受信後にHood.startPopup()を要求するかを判断するための受付条件である。

4.25.4. 状態

HoodSwitchの各インスタンスは，以下の状態を持つ。

mainStatus	HoodSwitchMainStatusType	HoodSwitchデバイスの主状態（正常/故障）
interactionStatus	HoodSwitchInteractionStatusType	デバウンス後の有効な操作状態（Pressed/Released）

interactionStatusは物理入力にdebounceMsのデバウンス処理を適用した後の状態を示す。

Released状態で押下入力がdebounceMs以上安定した場合にPressedへ遷移し，Pressedイベントを1回通知する。

Pressed状態で解除入力がdebounceMs以上安定した場合にReleasedへ遷移し，Releasedイベントを1回通知する。

デバウンス時間未満の一時的な入力変化では状態を変更せず，Pressed/Releasedイベントも通知しない。

4.25.5. サービスコール一覧

HoodSwitchに対するサービスコールは以下の通り。

getConfig	HoodSwitchの構成情報の参照
getStatus	HoodSwitchの状態の参照
notify	HoodSwitchイベントの通知要求
unnotify	HoodSwitchイベントの通知停止
getEvent	HoodSwitchイベントの取得

4.25.6. イベント

HoodSwitchに対するイベント種別として，Pressed/Released，故障検知/回復，構成変更，イベントキューのオーバーフローなどを定義する。

Pressed/ReleasedはHoodSwitchの入力状態変化を通知するイベントであり，Hoodの動作状態や動作完了を通知するイベントではない。

HoodSwitchに対するイベント種別は以下の通り。

Pressed	デバウンス後にHoodSwitchがReleasedからPressedへ遷移し，オープナー操作を検知したことを通知する。Hoodのポップアップ開始または完了を意味しない
Released	デバウンス後にHoodSwitchがPressedからReleasedへ遷移したことを通知する
FaultDetected	故障検知
FaultRecovered	故障から回復
EventOverflow	イベントキューのオーバフロー
ConfigChanged	構成情報の変更

この内，イベント種別がEventOverflowの場合には，以下の関連情報を持つ。

noLostEvents	UInt32	失われたイベントの数

4.25.7. データ型の定義

4.25.7.1. HoodSwitchの機能クラス

```
type HoodSwitchCapabilityType = enum {
    SwitchControl
};
```

4.25.7.2. HoodSwitchの構成情報（getConfigが返すデータ型）

```
type HoodSwitchConfigType = struct {
```

```
    displayName : String,
    relatedHood : RelatedObjectType,
    interactionType : String,
    vehicleStateCondition : String,
    debounceMs : UInt32
};
```

4.25.7.3. HoodSwitchの主状態

```
type HoodSwitchMainStatusType = enum {
    Normal, // 正常
    Fault   // 故障中
};
```

4.25.7.4. HoodSwitchの操作状態

```
type HoodSwitchInteractionStatusType = enum {
    Released,   // オープナー操作が解除されている
    Pressed // オープナー操作が押下されている
};
```

4.25.7.5. HoodSwitchの状態情報（getStatusが返すデータ型）

```
type HoodSwitchStatusType = struct {
    mainStatus : HoodSwitchMainStatusType,
    interactionStatus : HoodSwitchInteractionStatusType
};
```

4.25.7.6. HoodSwitchイベント種別

```
type HoodSwitchEventKindType = enum {
    Pressed,
    Released,
    FaultDetected,
    FaultRecovered,
    EventOverflow,
    ConfigChanged
};
```

```
type HoodSwitchEventType = struct {
    eventInfo : variant(HoodSwitchEventKindType) {
        EventOverflow {
            noLostEvents : UInt32
        }
```

```
    }
};
```

4.25.8. サービスコールの詳細規定

getConfig

HoodSwitchの構成情報の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
config	HoodSwitchConfigType	HoodSwitchに関する構成情報

【エラーコード】

E_OK	成功
E_NOT_SUPPORTED	未サポート（実装により返却されない場合あり）

【機能】

シングルトンのHoodSwitchに関する構成情報をHoodSwitchConfigTypeで返す。

getStatus

HoodSwitchの状態の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
status	HoodSwitchStatusType	HoodSwitchのデバイス状態および Pressed/Released状態

【エラーコード】

E_OK	成功

【機能】

シングルトンのHoodSwitchに関するすべての状態を返す。

notify

HoodSwitchイベントの通知要求

【パラメータ】

eventType	HoodSwitchEventKindType?	通知するイベントの種類の集合

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【エラーコード】

E_INVALID_PARAMETER	eventTypeが有効範囲外

【機能】

シングルトンのHoodSwitchで発生したeventTypeで指定したイベントの通知を要求する。イベントを通知するためのイベントキューを生成し，その通知ハンドルをnotifyHandleに返す。

eventTypeを省略した場合，すべての種類のイベントの通知を要求する。

Pressedイベントを受信したOSはvehicleStateConditionを評価し，条件を満たす場合にrelatedHoodのstartPopup()を要求する。Pressedイベント自体はHoodの動作開始または完了を保証しない。

unnotify

HoodSwitchイベントの通知停止

【パラメータ】

notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正

【機能】

notifyHandleで指定した通知を停止する。指定した通知用のイベントキューは削除される。

getEvent

HoodSwitchイベントの取得

【パラメータ】

notifyHandle	NotifyHandleType	イベントキューを識別するための通知ハンドル
eventDetail	EventDetailType	イベント詳細情報

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正
E_NO_DATA	イベントキューが空

【機能】

notifyHandleで指定したHoodSwitch通知用イベントキューから，先頭のイベントを取り出す。取り出したイベントはイベントキューから削除される。



イベントがパラメータ等で渡されてくるプログラミング環境向けの物理APIでは，用意する必要がない。

4.26. Vehicle.Body.Horn（略称：Horn）

4.26.1. 位置付けと機能

Vehicle.Body.Horn（略称：Horn）は，車両のクラクション（警音器）を吹鳴させるためのシングルインスタンスオブジェクトである。

4.26.1.1. クラクションの位置付けと性質

クラクションとは，車両外部に対して可聴音を発する警報装置であり，車両の走行状態に関わらず吹鳴可能なものと定義する。

4.26.1.2. Hornの機能

Hornは，非吹鳴，通常吹鳴，アラート吹鳴，故障の状態を持つ。

startHorn()は，timeMsで指定した有限時間だけ通常吹鳴を要求する。指定時間経過後は自動的に非吹鳴状態へ遷移し，TargetTimeReachedを通知する。

stopHorn()は，通常吹鳴またはアラート吹鳴を停止し，非吹鳴状態への遷移を要求する。

alertHorn()は，PatternTypeで指定した吹鳴時間，停止時間，繰り返し回数に従って断続的な吹鳴を要求する。

alertHorn()でrepeatCountが1以上の場合は，指定回数の吹鳴完了後に自動的に非吹鳴状態へ遷移し，TargetCountReachedを通知する。repeatCount=0の場合は回数無制限とし，stopHorn()が呼び出されるまでパターンを繰り返す。

4.26.1.3. クラクションに対する制御の調停

クラクションに対する制御の調停には，標準的な調停機能（後勝ちの原則）が適用される。

すなわち，クラクションを吹鳴するサービスコールが呼び出され吹鳴中に，停止するサービスコールが呼び出されると，最初のサービスコールによる制御はキャンセルされ，クラクションは停止する。。

警告状態の設定要求または解除要求も，同様に後勝ちで調停される。

4.26.1.4. 音量制御

音量調整の可否はHornConfig.volumeAdjustableで示す。

volumeAdjustable=falseの車両では，クラクション音量は車両側で規定された固定値とし，setVolume()はE_NOT_SUPPORTEDを返す。

volumeAdjustable=trueの車両では，setVolume()によりHornConfig.minVolumePercentからmaxVolumePercentまでの範囲で音量を設定できる。

4.26.2. 機能クラス

Hornは，以下の機能クラスを持つ。

VolumeControllable	音量（Volume）をAPIで制御できる
PatternBlow	指定パターンによる断続吹鳴を実行できる



VolumeControllable
法規対応のため要検討箇所。現状はどの%でも一定の音量とする。

4.26.3. リスククラス

Hornは，以下のリスククラスを持つ。

RiskBlow	ユーザの意図しない吹鳴に伴う諸々のリスク
RiskUnblow	ユーザの意図した吹鳴をしないことに伴う諸々のリスク
RiskStop	吹鳴が止まってしまうことに伴う諸々のリスク
RiskUnstop	吹鳴が止まらないことに伴う諸々のリスク

【今後の検討課題】

RiskBlowとRiskUnstopに関して，現状は別のリスクとして検討しているが，今後の作業で分ける必要がないと判断した場合，統合する。

4.26.4. 構成情報

Hornの各インスタンスは，以下の構成情報を持つ。

capabilities	enumSet (HornCapabilityType)	クラクションが持つ機能

riskClasses	enumSet(HornRiskType)	クラクションのリスククラスの集合
mountedOn	RelatedObjectType	設置対象物（車両または車室）
displayName	String	ユーザに表示する呼称を取得するためのキー
minDurationMs	TimeType	startHorn()およびPatternType.blowDurationMsで指定可能な最小吹鳴時間[ms]。0より大きい値とする
maxDurationMs	TimeType	startHorn()およびPatternType.blowDurationMsで指定可能な最大吹鳴時間[ms]
volumeAdjustable	Bool	trueの場合はsetVolume()による音量変更をサポートし，falseの場合は固定音量でE_NOT_SUPPORTEDを返す
minVolumePercent	VolumeType	volumeAdjustable=trueの場合に指定可能な最小音量[%]
maxVolumePercent	VolumeType	volumeAdjustable=trueの場合に指定可能な最大音量[%]
supportedPatterns	array(PatternType)	車両がサポートする代表的なアラート吹鳴パターン。任意パターン指定を許可しない実装では，この一覧に含まれるパターンのみ受け付ける

上記は，YAML形式で記述した構成記述ファイル中に記述する形式である。

- mountedOnは，物理APIで定められたデータ型に変換して返す。
- displayNameKeyに対しては，それを元に現在のロケールに対応する呼称の文字列を取得し，その文字列を返す。

minDurationMsおよびmaxDurationMsは，車両ごとの有効な吹鳴時間範囲を示す。

volumeAdjustable=falseの場合，minVolumePercentおよびmaxVolumePercentは固定音量値を表す同一値としてもよい。

4.26.5. 状態

Hornの各インスタンスは，以下の状態を持つ。

mainStatus	HornMainStatusType	主状態（非吹鳴，通常吹鳴，アラート吹鳴，故障）
requestedDurationMs	TimeType?	startHorn()で受理された吹鳴時間[ms]。通常吹鳴中のみ有効
priority	PriorityType?	現在実行中の要求に適用されている操作優先度
volume	VolumeType	現在クラクションに適用される有効音量[%]。音量変更非対応の場合も車両で定められた固定音量に対応する値を返す。
pattern	PatternType?	alertHorn()で受理された吹鳴パターン。アラート吹鳴中のみ有効

クラクションの主状態は，非吹鳴状態，通常吹鳴状態，警告吹鳴状態，および故障状態とする。

非吹鳴状態は，クラクションが吹鳴していない状態を示す。通常吹鳴状態は，startHorn()による吹鳴要求を受け付け，指定された吹鳴時間が経過するまで吹鳴している状態を示す。

警告吹鳴状態は，alertHorn()による要求を受け付け，指定された吹鳴パターンに従って吹鳴している状態を示す。指定された繰り返し回数の完了，またはstopHorn()による停止要求によって警告吹鳴状態を終了する。

故障状態は，クラクションの正常な吹鳴制御を継続できない故障を検出している状態を示す。

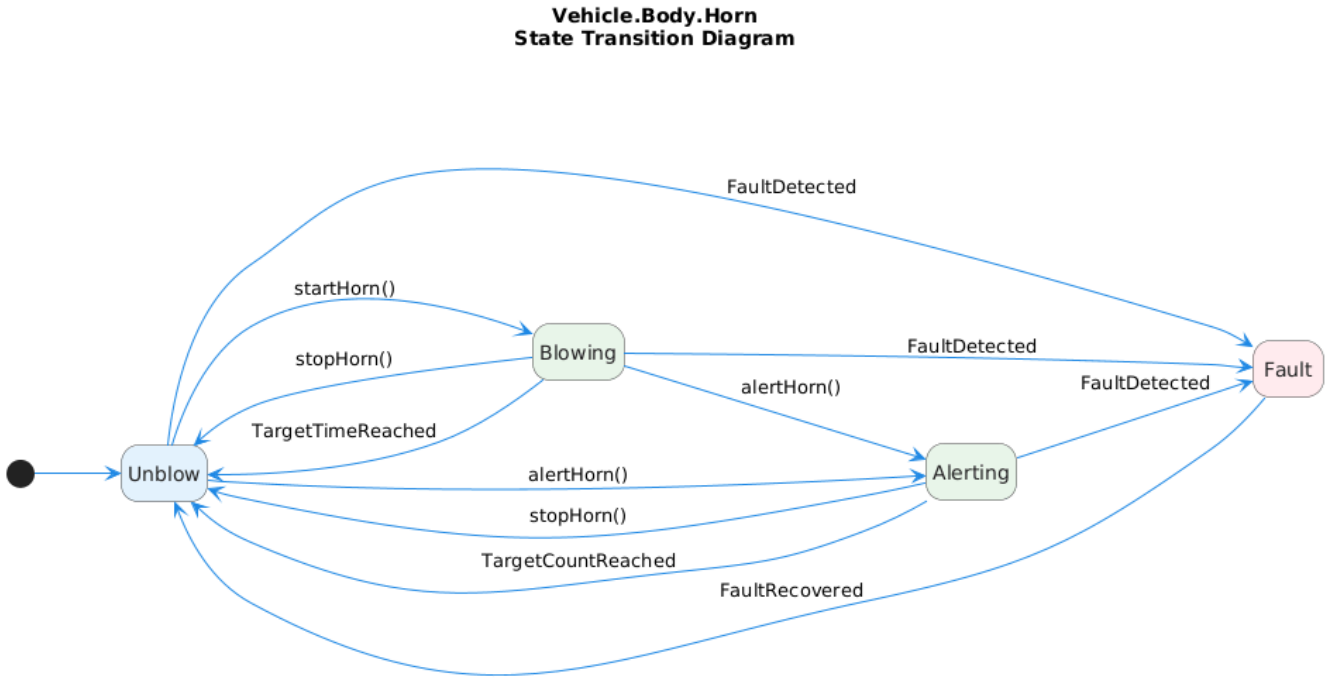


Figure 18. クラクションの主状態の状態遷移

4.26.6. サービスコール一覧

Hornに対するサービスコールは以下の通り。

getConfig	クラクションの構成情報の参照
getStatus	クラクションの状態の参照
startHorn	時間指定による吹鳴開始
stopHorn	吹鳴停止
setVolume	音量設定
alertHorn	パターン指定によるアラート吹鳴
notify	クラクションイベントの通知要求
unnotify	クラクションイベントの通知停止
getEvent	クラクションイベントの取得

4.26.7. イベント

クラクションに対するイベント種別として，制御主体の変更，通常吹鳴完了，パターン吹鳴完了，故障検知/回復，構成変更，イベントキューのオーバーフローなどを定義する。

Hornに対するイベント種別は以下の通り。

ControlledBySelf	自アプリケーションが制御
ControlledByOther	他のアプリケーションが制御
TargetCountReached	alertHorn()でrepeatCountに1以上を指定した場合に，指定回数の吹鳴が完了して非吹鳴状態へ遷移したことを通知する
TargetTimeReached	startHorn()で指定したtimeMsが経過し，通常吹鳴を終了して非吹鳴状態へ遷移したことを通知する
FaultDetected	故障検知
FaultRecovered	故障から回復
EventOverflow	イベントキューのオーバフロー
ConfigChanged	構成情報の変更

この内，イベント種別がControlledByOtherの場合には，以下の関連情報を持つ。

sourceApplication	ApplicationIdType	操作したアプリケーション

また，イベント種別がEventOverflowの場合には，以下の関連情報を持つ。

noLostEvents	UInt32	失われたイベントの数

4.26.8. データ型の定義

4.26.8.1. クラクションの機能クラス

```
type HornCapabilityType = enum {
    VolumeControllable,
    PatternBlow
};
```

4.26.8.2. 吹鳴時間型

```
type TimeType = baseType {
};
```

4.26.8.3. 音量型

```
type VolumeType = baseType {
```

```
};
```

4.26.8.4. アラート吹鳴パターン

```
type PatternType = struct {
    blowDurationMs : TimeType,  // 1回あたりの吹鳴時間[ms]。0は無効
    pauseDurationMs : TimeType, // 各吹鳴間の停止時間[ms]。0は停止時間なし
    repeatCount : UInt32    // 吹鳴回数。1以上は指定回数で自動終了し，0はstopHorn()が
呼ばれるまで無期限に繰り返す
};
```

4.26.8.5. クラクションに対するリスククラス

```
type HornRiskType = enum {
    RiskBlow,
    RiskUnblow,
    RiskStop,
    RiskUnstop
};
```

4.26.8.6. クラクションの構成情報（getConfigが返すデータ型）

```
type HornConfigType = struct {
    capabilities : enumSet(HornCapabilityType),
    riskClasses : enumSet(HornRiskType),
    mountedOn : RelatedObjectType,
    displayName : String,
    minDurationMs : TimeType,
    maxDurationMs : TimeType,
    volumeAdjustable : Bool,
    minVolumePercent : VolumeType,
    maxVolumePercent : VolumeType,
    supportedPatterns : array(PatternType)
};
```

4.26.8.7. クラクションの状態

```
type HornMainStatusType = enum {
    Unblow, // 非吹鳴中
    Blowing,    // startHorn()による通常吹鳴中
    Alerting,   // alertHorn()によるアラート吹鳴中
    Fault   // 故障中
};
```

4.26.8.8. クラクションの状態情報（getStatusが返すデータ型）

```
type HornStatusType = struct {
    mainStatus : HornMainStatusType,
    requestedDurationMs : TimeType?,
    priority : PriorityType?,
    volume : VolumeType,
    pattern : PatternType?
};
```

4.26.8.9. クラクションイベント種別

```
type HornEventKindType = enum {
    ControlledBySelf,
    ControlledByOther,
    TargetCountReached,
    TargetTimeReached,
    FaultDetected,
    FaultRecovered,
    EventOverflow,
    ConfigChanged
};
```

```
type HornEventType = struct {
    eventInfo : variant(HornEventKindType) {
        ControlledByOther {
            sourceApplication : ApplicationIdType
        },
        EventOverflow {
            noLostEvents : UInt32
        }
    }
};
```

4.26.9. サービスコールの詳細規定

getConfig

クラクションの構成情報の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

config	HornConfigType	クラクションに関する構成情報

【エラーコード】

E_OK	成功
E_NOT_SUPPORTED	未サポート（実装により返却されない場合あり）

【機能】

クラクションの構成情報を返す。

getStatus

クラクションの状態の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
status	HornStatusType	クラクションの状態

【エラーコード】

E_OK	成功

【機能】

クラクションに関するすべての状態を返す。

startHorn

時間指定による吹鳴開始

【パラメータ】

timeMs	TimeType	吹鳴時間[ms]
priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_PARAMETER	timeMsが0，HornConfigの有効範囲外，またはpriorityが有効範囲外
E_RISK_APPLICATION	アプリケーションが対応していないリスクがある

E_RISK_USER	ユーザがリスクを承認していない
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_FAULT	クラクションが故障中
E_OBJECT_LOCKED	クラクションが，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている

【機能】

timeMsで指定した時間だけクラクションを吹鳴する。

timeMs=0は無効としE_INVALID_PARAMETERを返す。また，timeMsはHornConfig.minDurationMs以上かつHornConfig.maxDurationMs以下でなければならない。

要求を正常に受け付けるとmainStatus=Blowingへ遷移する。指定時間経過後は自動的にmainStatus=Unblowへ遷移し，TargetTimeReachedを通知する。

吹鳴中にstopHorn()または別のstartHorn()/alertHorn()が受理された場合，本要求はキャンセルされTargetTimeReachedは通知しない。

stopHorn

吹鳴停止

【パラメータ】

priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_PARAMETER	priorityが有効範囲外
E_RISK_APPLICATION	アプリケーションが対応していないリスクがある
E_RISK_USER	ユーザがリスクを承認していない
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_FAULT	クラクションが故障中
E_OBJECT_LOCKED	クラクションが，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている

【機能】

startHorn()による通常吹鳴またはalertHorn()によるアラート吹鳴の停止を要求する。

要求を正常に受け付けると実行中の吹鳴要求をキャンセルし，mainStatus=Unblowへ遷移する。

stopHorn()によって終了した場合，TargetTimeReachedおよびTargetCountReachedは通知しない。

setVolume

音量設定

【パラメータ】

volume	VolumeType	音量[%] 法規対応のため要検討箇所。現状変更できる枠組みのみ作成し，どの%でも一定の音量とする。
priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_NOT_SUPPORTED	HornConfig.volumeAdjustable=falseで音量変更をサポートしていない
E_INVALID_PARAMETER	volumeがHornConfigの有効範囲外，またはpriorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_FAULT	クラクションが故障中

【機能】

HornConfig.volumeAdjustable=trueの場合に限り，クラクションの音量変更を要求する。

volumeはHornConfig.minVolumePercent以上かつHornConfig.maxVolumePercent以下でなければならない。

volumeAdjustable=falseの車両では音量は車両側の固定値を使用し，本APIはE_NOT_SUPPORTEDを返す。

alertHorn

パターン指定によるアラート吹鳴

【パラメータ】

pattern	PatternType	吹鳴時間，停止時間，繰り返し回数で構成されるアラート吹鳴パターン
priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_PARAMETER	pattern.blowDurationMsが0または有効範囲外，patternの内容が未サポート，またはpriorityが有効範囲外

E_RISK_APPLICATION	アプリケーションが対応していないリスクがある
E_RISK_USER	ユーザがリスクを承認していない
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_FAULT	クラクションが故障中
E_OBJECT_LOCKED	クラクションが，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている

【機能】

pattern.blowDurationMs，pattern.pauseDurationMs，pattern.repeatCountで指定したパターンに従ってアラート吹鳴を開始する。

blowDurationMs=0は無効としE_INVALID_PARAMETERを返す。blowDurationMsはHornConfig.minDurationMs以上かつHornConfig.maxDurationMs以下でなければならない。

pauseDurationMs=0は吹鳴間に停止時間を設けないことを意味する。

repeatCountが1以上の場合は指定回数の吹鳴完了後に自動的にmainStatus=Unblowへ遷移し，TargetCountReachedを通知する。

repeatCount=0の場合は無期限繰り返しを意味し，stopHorn()が受理されるまでmainStatus=Alertingを維持する。この場合TargetCountReachedは通知しない。

実装がsupportedPatternsによる限定方式を採用する場合，HornConfig.supportedPatternsに含まれないpatternはE_INVALID_PARAMETERとする。

notify

クラクションイベントの通知要求

【パラメータ】

eventType	HornEventKindType?	通知するイベントの種類の集合

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
notifyHandle	NotifyHandleType	通知ハンドル

【エラーコード】

E_INVALID_PARAMETER	eventTypeが有効範囲外

【機能】

シングルトンのクラクションで発生したeventTypeで指定したイベントの通知を要求する。

eventTypeを省略した場合，すべての種類のイベントの通知を要求する。

unnotify

クラクションイベントの通知停止

【パラメータ】

notifyHandle	NotifyHandleType	通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正

【機能】

notifyHandleで指定した通知を停止する。指定した通知用のイベントキューは削除される。

getEvent

クラクションイベントの取得

【パラメータ】

notifyHandle	NotifyHandleType	イベントキューを識別するための通知ハンドル
eventDetail	EventDetailType	イベント詳細情報

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正
E_NO_DATA	イベントキューが空

【機能】

notifyHandleで指定した通知用のイベントキューから，先頭のイベントを取り出す。取り出したイベントは，イベントキューから削除される。



イベントがパラメータ等で渡されてくるプログラミング環境向けの物理APIでは，用意する必要がない。

4.27. Vehicle.Body.Mirror（略称：Mirror）

4.27.1. 位置付けと機能

Vehicle.Body.Mirror（略称：Mirror）は，車両のミラーを操作するためのマルチインスタンスオブジェクトである。

4.27.1.1. ミラーの位置付けと性質

車両の外部に設置され，運転者の視界を補助するための装置。走行中の安全性を確保するために重要な役割を果たす。

ミラーの機能

ミラーは，可動部を一次元に移動できるオブジェクトであり，MovableObjectの機能を持つ。

可動部の位置は，いずれの方向の移動に対しても，どれだけ開いているかのパーセンテージ（開閉度，0：全閉（格納），100：全開（展開））で表す。また，0に近づく動作を格納動作，100に近づく動作を展開動作と呼ぶ。

挟み込み防止機能を持つミラーは，可動部の格納動作中に挟み込みを検知すると，格納動作を停止し，展開動作を開始する。挟み込み検知後にどの位置まで展開するかは，車両依存である

可動部が停止中の間も，マニュアル操作にて現在位置が変化することがあり得る。

4.27.1.2. ミラーに対する制御の調停

ミラーに対する制御の調停には，標準的な調停機能（後勝ちの原則）が適用される。すなわち，ミラーを格納するサービスコールが呼び出され，ミラーが格納動作中に，展開するサービスコールが呼び出されると，最初のサービスコールによる制御はキャンセルされ，ミラーは展開動作を始める。

また，ミラーは標準的なロック機能を持つLockableObjectである。ミラーには，機械的なロック機構は無いのが一般的であり，ソフトウェアでロック機能を実現することを想定している。

ミラーが故障または挟み込みを検知した場合，ロックは解除され，故障中と挟み込み回避中は，ロックを取得することができない。

4.27.2. 機能クラス

Mirrorは，以下の機能クラスを持つ。

Movable	パワーミラーであり，APIにより移動させることができる。
ClosedDetectable	ミラーが全閉（格納）状態であることを検出することができる

4.27.3. リスククラス

Mirrorは，以下のリスククラスを持つ。

RiskPinch	ミラーを閉めることに伴う挟み込みのリスク
RiskCollision	ミラーを開けることに伴う，歩行者・隣接車両・構造物等への接触のリスク
RiskVisibility	ミラーが正規位置まで展開しきれないことに伴う，後方視界不良のリスク

4.27.4. 構成情報

Mirrorの各インスタンスは，以下の構成情報を持つ。

instanceId	IdType	インスタンスが持つId
placement	PlacementType	インスタンスがある場所
capabilities	enumSet (MirrorCapabilityType)	そのミラーが持つ機能
riskClasses	enumSet(MirrorRiskType)	そのミラーのリスククラスの集合
mountedOn	RelatedObjectType	設置対象物（ドアまたは車体）
displayName	String	ユーザに表示する呼称を取得するためのキー

上記は，YAML形式で記述した構成記述ファイル中に記述する形式である。構成情報を参照するサービスコールは，この構成情報に対して，次の変換を行った構成情報を返す。

- mountedOnは，物理APIで定められたデータ型に変換して返す。
- displayNameKeyに対しては，それを元に現在のロケールに対応する呼称の文字列を取得し，その文字列を返す。

4.27.5. 状態

Mirrorの各インスタンスは，以下の状態を持つ。

mainStatus	MirrorMainStatusType	主状態
lockStatus	LockStatusType	ロック状態
position	PositionType	現在位置
targetPosition	PositionType	目標位置

ミラーの主状態（mainStatus）は，MovableObjectの主状態に加えて，挟み込み回避中の状態をとる。また，位置のパーセンテージが大きくなる方へ移動中を展開動作中，位置のパーセンテージが小さくなる方へ移動中を格納動作中と呼ぶ。

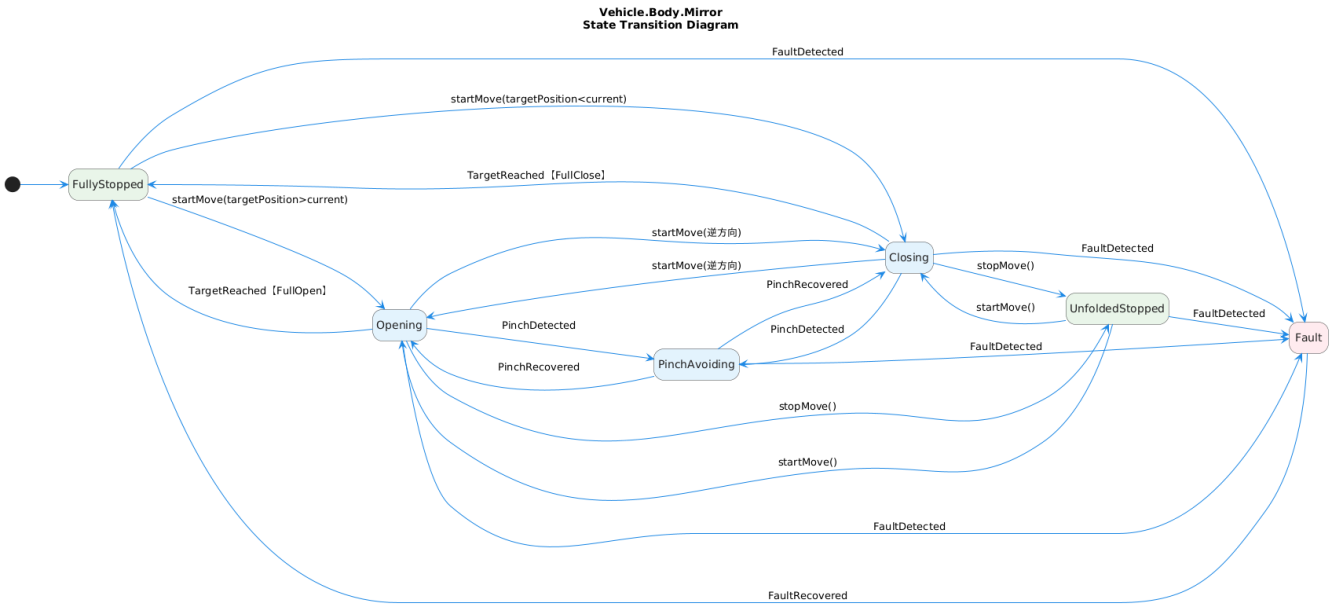


Figure 19. ミラーの主状態の状態遷移

ミラーの現在位置（position）と目標位置（targetPosition）は，MovableObjectの現在位置と目標位置に関する規定に従う。

ミラーロック機能のみの場合は，展開要求のみを受け付け，格納要求は受け付けないものとする。また，ロック解除完了時をもって目標位置到達と考える。

4.27.6. サービスコール一覧

Mirrorに対するサービスコールは以下の通り。

getConfigAll	すべてのミラーの構成情報の参照
getStatus	ミラーの状態の参照
startMove	ミラーの格納・展開の開始
stopMove	ミラーの格納・展開の停止
lock	ミラーのロックの取得
unlock	ミラーのロックの解除
notify	ミラーイベントの通知要求
unnotify	ミラーイベントの通知停止
getEvent	ミラーイベントの取得

4.27.7. イベント

ミラーに対するイベント種別には，MovableObjectの持つイベントとLockableObjectが持つイベントに加えて，挟み込み検知と挟み込みから回復がある。

Mirrorに対するイベント種別は以下の通り。

ControlledBySelf	自アプリケーションが制御
ControlledByOther	他のアプリケーションが制御

TargetReached	指定された目標位置に到達して停止，または，移動停止が完了
FaultDetected	故障検知
FaultRecovered	故障から回復
PinchDetected	挟み込み検知
PinchRecovered	挟み込みから回復
OwnLockRevoked	自アプリケーションのロックの強制解除
OtherLockReleased	他のアプリケーション等によるロック解除
EventOverflow	イベントキューのオーバフロー
ConfigChanged	構成情報の変更

この内，イベント種別がControlledByOtherの場合には，以下の関連情報を持つ。

sourceApplication	ApplicationIdType	操作したアプリケーション

また，イベント種別がEventOverflowの場合には，以下の関連情報を持つ。

noLostEvents	UInt32	失われたイベントの数

4.27.8. データ型の定義

4.27.8.1. ミラーの機能クラス

```
type MirrorCapabilityType = enum {
    Movable,
    ClosedDetectable
};
```

4.27.8.2. ミラーに対するリスククラス

```
type MirrorRiskType = enum {
    RiskPinch,
    RiskCollision,
    RiskVisibility
};
```

4.27.8.3. ミラーの構成情報（getConfigAllが返すデータ型）

```
type MirrorConfigType = struct {
    instanceId : IdType,
    placement : PlacementType,
    capabilities : enumSet(MirrorCapabilityType),
    riskClasses : enumSet(MirrorRiskType),
```

```
    mountedOn : RelatedObjectType,
    displayName : String
};
```

4.27.8.4. ミラー状態

```
type MirrorMainStatusType = enum {
    FullyStopped,   // 停止中（全閉（格納））
    UnfoldedStopped,    // 停止中（展開）
    Opening,    // 開動作中
    Closing,    // 格納動作中
    Fault,  // 故障中
    PinchAvoiding   // 挟み込み回避中
};
```

```
type MirrorStatusType = struct {
    mainStatus : MirrorMainStatusType,
    lockStatus : LockStatusType,
    position : PositionType,
    targetPosition : PositionType
};
```

4.27.8.5. ミラーイベント

```
type MirrorEventKindType = enum {
    ControlledBySelf,
    ControlledByOther,
    TargetReached,
    FaultDetected,
    FaultRecovered,
    PinchDetected,
    PinchRecovered,
    OwnLockRevoked,
    OtherLockReleased,
    EventOverflow,
    ConfigChanged
};
```

```
type MirrorEventType = struct {
    instanceId : IdType,
    eventInfo : variant(MirrorEventKindType) {
        ControlledByOther {
            sourceApplication : ApplicationIdType
        },
        EventOverflow {
            noLostEvents : UInt32
```

```
        }
    }
};
```

4.27.9. サービスコールの詳細規定

getConfigAll

すべてのミラーの構成情報の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
config	MirrorConfigType[]	すべてのミラーに関する構成情報

【エラーコード】

なし

【機能】

すべてのミラーに関する構成情報を返す。

getStatus

ミラーの状態の参照

【パラメータ】

instanceId	IdType	操作対象のミラーのID

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
status	MirrorStatusType	対象ミラーの状態

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外

【機能】

操作対象のミラーに関するすべての状態を返す。

startMove

ミラーの格納・展開の開始

【パラメータ】

instanceId	IdType	操作対象のミラーのID
targetPosition	PositionType	目標位置（開閉度）
priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_INVALID_PARAMETER	targetPosition，priorityが有効範囲外
E_RISK_APPLICATION	アプリケーションが対応していないリスクがある
E_RISK_USER	ユーザがリスクを承認していない
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_FAULT	操作対象のドアが故障中
E_OBJECT_STATUS	操作対象のドアが挟み込み回避中
E_OBJECT_LOCKED	操作対象のドアが，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている

【機能】

目標位置をtargetPositionに設定し，操作対象のミラーの格納・展開動作を開始する。

targetPositionに0を指定すれば格納動作の開始，100を指定すれば展開動作の開始の意味になる。

ミラーのため一般的に途中開度で止まることを想定していないためtargetPositionに0～100の中間値を指定した場合，E_INVALID_PARAMETERエラーとなる。

リスククラスRiskPinchに対応できていない場合でも，ミラーを展開する方向に動作を開始しようとした場合には，操作を受け付ける。そうでない場合には，E_RISK_APPLICATIONエラーまたはE_RISK_USERエラーとなる。

リスククラスRiskCollisionに対応できていない場合でも，ミラーを格納する方向に動作を開始しようとした場合には，操作を受け付ける。そうでない場合には，E_RISK_APPLICATIONエラーまたはE_RISK_USERエラーとなる。

stopMove

ミラーの格納・展開の停止

【パラメータ】

instanceId	IdType	操作対象のミラーのID
priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_INVALID_PARAMETER	priorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_FAULT	操作対象のミラーが故障中
E_OBJECT_STATUS	操作対象のミラーが挟み込み回避中
E_OBJECT_LOCKED	操作対象のミラーが，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている

【機能】

各次元方向の目標位置を現在位置に設定して，操作対象のミラーの各次元方向への格納・展開動作の停止を要求する。

lock

ミラーのロックの取得

【パラメータ】

instanceId	IdType	操作対象のミラーのID
priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_INVALID_PARAMETER	priorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_LOCKED	ロック対象のミラーが，priorityと同じかそれより高い優先度でロックされている
E_OBJECT_FAULT	ロック対象のミラーが故障中
E_OBJECT_STATUS	ロック対象のミラーが挟み込み回避中

【機能】

ロック対象のミラーを，指定した操作優先度でロックする。

unlock

ミラーのロックの解除

【パラメータ】

instanceId	IdType	操作対象のミラーのID

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_OBJECT_STATUS	自アプリケーションが，操作対象のミラーをロックしていない

【機能】

ロック解除対象のミラーを，ロック解除する。

notify

ミラーイベントの通知要求

【パラメータ】

instanceId	IdType?	イベント通知対象のミラーのID
eventFilter	enumSet (MirrorEventKindType)?	通知するイベントの種類の集合

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_INVALID_PARAMETER	eventFilterが有効範囲外

【機能】

イベント通知対象のミラーで発生したeventFilterで指定したイベントの通知を要求する。イベントを通知するためのイベントキューを生成し，その通知ハンドルを notifyHandle に返す。

instanceIdを省略した場合，すべてのインスタンスで発生したイベントの通知を要求する。eventFilterを省略した場合，すべての種類のイベントの通知を要求する。

unnotify

ミラーイベントの通知停止

【パラメータ】

notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正

【機能】

notifyHandleで指定した通知を停止する。指定した通知用のイベントキューは削除される。

getEvent

ミラーイベントの取得

【パラメータ】

notifyHandle	NotifyHandleType	イベントキューを識別するための通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
event	MirrorEventType	取得したイベント

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正
E_NO_DATA	イベントキューが空

【機能】

notifyHandleで指定した通知用のイベントキューから，先頭のイベントを取り出す。取り出したイベントは，イベントキューから削除される。



イベントがパラメータ等で渡されてくるプログラミング環境向けの物理APIでは，用意する必要がない。

4.28. Vehicle.Body.MirrorSwitch（略称：MirrorSwitch）

4.28.1. 位置付けと機能

Mirrorを制御する物理的なスイッチを扱う。スイッチとMirrorは直結せず，制御はOS側（ビークルOS）で行うことを仮定する。

4.28.1.1. Mirrorスイッチの位置付けと性質

Mirrorスイッチとは，運転席に配置され乗員が物理操作できるMirror制御デバイスである。

Mirrorスイッチは各Mirrorに配置され独立した系統を持つ。

スイッチとMirrorは直接接続されず，制御要求はビークルOSが受け取る。

スイッチは入力源として機能し，Mirror制御の主体にはならない。

4.28.1.2. Mirrorスイッチの機能

Mirrorスイッチは乗員の操作によるスイッチ入力をOSへ提供し，Mirrorオブジェクトへ制御要求を伝達する役割を持つ。

各スイッチの系統はconfigData内のswitchCategoryによって識別される。

4.28.1.3. Mirrorスイッチに対する制御の調停

Mirrorスイッチは入力イベントの発生源であり，自身が制御競合を調停することはない。

入力はOSが一元管理し，Mirrorオブジェクトへルーティングされる。

調停はMirror側の後勝ち原則に従う。スイッチ操作が他アプリケーションの制御要求と競合した場合，Mirror側の調停ロジックにより採用または却下が判断される。

スイッチ自身は制御主体ではなく，アプリケーションや自動制御と同様に「入力の一つ」として扱われる。

4.28.2. 機能クラス

MirrorSwitchは，以下の機能クラスを持つ。

SwitchControl	Mirror操作入力を提供する基本機能

4.28.3. 構成情報

MirrorSwitchの各インスタンスは，以下の構成情報を持つ。

instanceId	IdType	インスタンスが持つId
placement	PlacementType	インスタンスがある場所
switchCategory	MirrorSwitchCategoryType	スイッチ系統（ミラー調整スイッチ，ミラー折り畳みスイッチ，ミラーヒーティングスイッチ，ミラー防眩スイッチ）

capabilities	enumSet (MirrorSwitchCapabilityType)	このスイッチがサポートする操作機能の集合
switchType	String	スイッチ種別（ミラー調整スイッチ，ミラー折り畳みスイッチ，ミラーヒーティングスイッチ，ミラー防眩スイッチかなど，OEM依存）
displayName	String	ユーザに表示する呼称を取得するためのキー

上記は，YAML形式で記述した構成記述ファイル中に記述する形式である。構成情報を参照するサービスコールは，この構成情報に対して，次の変換を行った構成情報を返す。

- displayNameKeyに対しては，それを元に現在のロケールに対応する呼称の文字列を取得し，その文字列を返す。

4.28.4. 状態

MirrorSwitchの各インスタンスは，以下の状態を持つ。

mainStatus	MirrorSwitchMainStatusType	スイッチの入力状態（MirrorAdjust系統の場合：OPEN/CLOSE/LEFT/RIGHT）
mirrorFoldingStatus	MirrorSwitchMirrorFoldingStatusType	ミラー格納スイッチの入力状態。（ON，OFF）
mirrorHeatingStatus	MirrorSwitchMirrorHeatingStatusType	ミラーヒーティングスイッチの入力状態。（ON，OFF）
heaterLevel	UInt32	ヒーティングON場合のレベル。0〜100。ダイアルで調整可能。
mirrorAntiglareStatus	MirrorSwitchMirrorAntiglareStatusType	ミラー防眩スイッチの入力状態。（ON，OFF）
antiglareLevel	UInt32	防眩がONの場合ののレベル。0〜100。ダイアルで調整可能。

4.28.5. サービスコール一覧

MirrorSwitchに対するサービスコールは以下の通り。

getConfigAll	すべてのMirrorスイッチの構成情報の参照
getStatus	Mirrorスイッチの状態参照
notify	Mirrorスイッチイベントの通知要求
unnotify	Mirrorスイッチイベントの通知停止
getEvent	Mirrorスイッチイベントの取得

4.28.6. イベント

Mirrorスイッチに関するイベントとして，スイッチ状態変化，故障検出/回復，構成変更，イベントキューのオーバーフローなどを定義する。

MirrorSwitchに対するイベント種別は以下の通り。

MainStatusChanged	ミラー調整に対応するスイッチのmainStatus入力状態が変化
FoldingStatusChanged	ミラー格納に対応するスイッチの入力状態が変化
HeatingStatusChanged	ミラーヒーティングに対応するスイッチの入力状態が変化
HeaterLevelChanged	ヒーターモードON時のヒーターレベルが変化
AntiglareStatusChanged	ミラー防眩に対応するスイッチの入力状態が変化
AntiglareLevelChanged	防眩モードON時の防眩レベルが変化
FaultDetected	故障検知
FaultRecovered	故障から回復
EventOverflow	イベントキューのオーバフロー
ConfigChanged	構成情報の変更

この内，イベント種別がMainStatusChangedの場合には，以下の関連情報を持つ。

newState	MirrorSwitchMainStatusType	スイッチの新しい入力状態

また，イベント種別がFoldingStatusChangedの場合には，以下の関連情報を持つ。

newState	MirrorSwitchMirrorFoldingStatusType	Foldingスイッチの新しい入力状態

また，イベント種別がHeatingStatusChangedの場合には，以下の関連情報を持つ。

newState	MirrorSwitchMirrorHeatingStatusType	Heatingスイッチの新しい入力状態

また，イベント種別がHeaterLevelChangedの場合には，以下の関連情報を持つ。

newHeaterLevel	UInt32	新しいレベル

また，イベント種別がAntiglareStatusChangedの場合には，以下の関連情報を持つ。

newState	MirrorSwitchMirrorAntiglareStatusType	Heatingスイッチの新しい入力状態

また，イベント種別がAntiglareLevelChangedの場合には，以下の関連情報を持つ。

newAntiglareLevel	UInt32	新しいレベル

また，イベント種別がEventOverflowの場合には，以下の関連情報を持つ。

noLostEvents	UInt32	失われたイベントの数

4.28.7. データ型の定義

4.28.7.1. Mirrorスイッチの機能クラス

```
type MirrorSwitchCapabilityType = enum {
    SwitchControl
};
```

4.28.7.2. スイッチ系統

```
type MirrorSwitchCategoryType = enum {
    MirrorAdjust,   // MirrorAdjustスイッチ系統
    MirrorFolding,  // MirrorFoldingスイッチ系統
    MirrorHeating,  // MirrorHeatingスイッチ系統
    MirrorAntiglare // MirrorAntiglareスイッチ系統
};
```

4.28.7.3. Mirrorsスイッチの入力状態

```
type MirrorSwitchMainStatusType = enum {
    False,  // OFF - Mirror停止
    Up, // UP - MirrorUP
    Down,   // Down - MirrorDown
    Right,  // Right - MirrorRight
    Left    // Left - MirrorLeft
};
```

4.28.7.4. Mirror格納スイッチの入力状態

```
type MirrorSwitchMirrorFoldingStatusType = enum {
    False,  // OFF - ミラー展開
    True    // ON - ミラー格納
};
```

4.28.7.5. Mirrorヒーティングの入力状態

```
type MirrorSwitchMirrorHeatingStatusType = enum {
    False,  // OFF - 機能停止
    True    // ON - 機能有効
};
```

4.28.7.6. Mirror防眩スイッチの入力状態

```
type MirrorSwitchMirrorAntiglareStatusType = enum {
```

```
    False,  // OFF - 機能停止
    True    // ON - 機能有効
};
```

Mirrorスイッチ構成情報（getConfigAll が返すデータ型）

```
type MirrorSwitchConfigType = struct {
    instanceId : IdType,
    placement : PlacementType,
    switchCategory : MirrorSwitchCategoryType,
    capabilities : enumSet(MirrorSwitchCapabilityType),
    switchType : String,
    displayName : String
};
```

4.28.7.7. Mirrorスイッチ状態

```
type MirrorSwitchStatusType = struct {
    mainStatus : MirrorSwitchMainStatusType,
    mirrorFoldingStatus : MirrorSwitchMirrorFoldingStatusType,
    mirrorHeatingStatus : MirrorSwitchMirrorHeatingStatusType,
    heaterLevel : UInt32,
    mirrorAntiglareStatus : MirrorSwitchMirrorAntiglareStatusType,
    antiglareLevel : UInt32
};
```

Mirrorスイッチの状態情報

4.28.7.8. Mirrorスイッチイベント種別

```
type MirrorSwitchEventKindType = enum {
    MainStatusChanged,
    FoldingStatusChanged,
    HeatingStatusChanged,
    HeaterLevelChanged,
    AntiglareStatusChanged,
    AntiglareLevelChanged,
    FaultDetected,
    FaultRecovered,
    EventOverflow,
    ConfigChanged
};
```

```
type MirrorSwitchEventType = struct {
    instanceId : IdType,
    switchCategory : MirrorSwitchCategoryType,
```

```
    eventInfo : variant(MirrorSwitchEventKindType) {
        MainStatusChanged {
            newState : MirrorSwitchMainStatusType
        },
        FoldingStatusChanged {
            newState : MirrorSwitchMirrorFoldingStatusType
        },
        HeatingStatusChanged {
            newState : MirrorSwitchMirrorHeatingStatusType
        },
        HeaterLevelChanged {
            newHeaterLevel : UInt32
        },
        AntiglareStatusChanged {
            newState : MirrorSwitchMirrorAntiglareStatusType
        },
        AntiglareLevelChanged {
            newAntiglareLevel : UInt32
        },
        EventOverflow {
            noLostEvents : UInt32
        }
    }
};
```

4.28.8. サービスコールの詳細規定

getConfigAll

すべてのMirrorスイッチの構成情報の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
config	MirrorSwitchConfigType[]	すべてのMirrorスイッチに関する構成情報

【機能】

すべてのMirrorスイッチに関する構成情報を返す。

getStatus

Mirrorスイッチの状態参照

【パラメータ】

instanceId	IdType	操作対象のMirrorスイッチのID

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
status	MirrorSwitchStatusType	対象のMirrorスイッチの状態

【エラーコード】

E_INVALID_ID	instanceId が有効範囲外

【機能】

操作対象のMirrorスイッチに関する状態を返す。



status には mainStatus（各Mirror系統のスイッチ入力状態）を含む。

notify

Mirrorスイッチイベントの通知要求

【パラメータ】

instanceId	IdType?	イベント通知対象のMirrorスイッチID（省略時は全インスタンス）
eventFilter	enumSet (MirrorSwitchEventKindType)?	通知するイベント種別の集合（省略時は全種）

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
notifyHandle	NotifyHandleType	通知ハンドル

【エラーコード】

E_INVALID_ID	instanceId が有効範囲外
E_INVALID_PARAMETER	eventFilter が有効範囲外

【機能】

イベント通知対象のMirrorスイッチで発生したeventFilterで指定したイベントの通知を要求する。イベントを通知するためのイベントキューを生成し，その通知ハンドルをnotifyHandleに返す。

instanceIdを省略した場合，すべてのインスタンスで発生したイベントの通知を要求する。eventFilterを省略した場合，すべての種類のイベントの通知を要求する。

unnotify

Mirrorスイッチイベントの通知停止

【パラメータ】

notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandle が不正

【機能】

notifyHandleで指定した通知を停止する。指定した通知用のイベントキューは削除される。

getEvent

Mirrorスイッチイベントの取得

【パラメータ】

notifyHandle	NotifyHandleType	イベントキューを識別する通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
event	MirrorSwitchEventType	取得したイベント

【エラーコード】

E_INVALID_HANDLE	notifyHandle が不正
E_NO_DATA	イベントキューが空

【機能】

notifyHandleで指定した通知用のイベントキューから，先頭のイベントを取り出す。取り出したイベントは，イベントキューから削除される。



イベントがパラメータ等で渡されてくるプログラミング環境向けの物理APIでは，用意する必要がない。

4.29. Vehicle.Body.Wiper（略称：Wiper）

4.29.1. 位置付けと機能

Vehicle.Body.Wiper（略称：Wiper）は，自動車に付着した雨水，雪，汚れなどを払拭する装置を操作するためのマルチインスタンスオブジェクトである。

4.29.1.1. ワイパーの位置付けと性質

ワイパーとは，自動車に付着した雨水，雪，汚れなどを払拭する装置で，払拭ブレード，アーム，それを駆動させるモーター，リンク機構を備えている。ワイパーはフロントウィンドウ，リアウィンドウ，ヘッドライト，サイドミラー，カメラなど複数の場所に装着される可能性がある。

また，ワイパーに併設されるウォッシャー機能（洗浄液による払拭）も本APIで扱う。

ワイパーの物理スイッチは，ワイパーとは別のオブジェクトとして扱う。

4.29.1.2. Wiperの機能

Wiperは，払拭動作を一次元的に移動するオブジェクトであり，MovableObjectと同等の機能を持つ。

ワイパーは，ブレードを往復動作させてフロントガラスを払拭する「払拭動作」を基本動作とする。

ワイパーの制御モードには，払拭動作を連続して実行する連続モード，および払拭動作を一定時間間隔で実行する間欠モードがあるが，これらの動作はAPIで指定する動作速度レベルと一時停止時間により制御される。

ワイパーの初期位置，目標位置となる端点は固定であるため，ワイパーの目標位置はAPIで指定，参照しない設計とする。

ワイパーが故障または凍結を検知した場合，制御を受け付けない。

4.29.1.3. ワイパーに対する制御の調停

ワイパーに対する制御の調停には，標準的な調停機能（後勝ちの原則）が適用される。すなわち，ワイパーの動作サービスコールが呼び出されたワイパーが動作中に，別の制御サービスコールが呼び出されると，最初のサービスコールによる制御はキャンセルされ，新しい制御が開始される。

また，ワイパーは標準的なロック機能を持つLockableObjectである。ワイパーのロック機能はソフトウェアで実現される。

ワイパーが故障または凍結を検知した場合，ロックは解除され，故障中と凍結中は，ロックを取得することができない。

4.29.2. 機能クラス

Wiperは，以下の機能クラスを持つ。

Movable	パワーワイパーであり，APIにより動作させることができる
HasWasher	ウォッシャー機能を備えている

4.29.3. リスククラス

Wiperは，以下のリスククラスを持つ。

RiskWipingOnTrip	走行中にワイパーが作動し，走行中に視界を妨げるリスク

RiskWiperWearing	ワイパーが摩耗した状態での操作により，窓に傷がつき視界不良となるリスク
RiskWiperBatteryDrain	ワイパーの連続操作によりバッテリーが消費し，走行不能となるリスク
RiskWiperHighFrequency	ワイパーの高周期操作により，ワイパー及びモータの早期摩耗・故障リスク

4.29.4. 構成情報

Wiperの各インスタンスは，以下の構成情報を持つ。

instanceId	IdType	インスタンスが持つId
placement	PlacementType	車両側面でのワイパーの位置（例：フロント，リア，サイドなど）
capabilities	enumSet (WiperCapabilityType)	そのワイパーが持つ機能
riskClasses	enumSet(WiperRiskType)	そのワイパーのリスククラスの集合
mountedOn	RelatedObjectType	設置対象物（ウィンドウガラスなど）
displayName	String	ユーザに表示する呼称を取得するためのキー

上記は，YAML形式で記述した構成記述ファイル中に記述する形式である。構成情報を参照するサービスコールは，この構成情報に対して，次の変換を行った構成情報を返す。

- mountedOnは，物理APIで定められたデータ型に変換して返す。
- displayNameKeyに対しては，それを元に現在のロケールに対応する呼称の文字列を取得し，その文字列を返す。

4.29.5. 状態

Wiperの各インスタンスは，以下の状態を持つ。

mainStatus	WiperMainStatusType	主状態
lockStatus	LockStatusType	ロック状態
frequencyLevel	UInt32	現在の動作速度レベル（0〜100%）。100%は車両が動作可能な最速レベル
intervalLevel	UInt32	現在の一時停止時間（秒単位，0〜30秒）
wiperWearLevel	UInt32	ワイパーの摩耗レベル（0〜100%）0%：摩耗無し，100%：摩耗，要交換
washerStatus	WiperWasherStatusType	ウォッシャーの状態（OFF：停止，ON：作動中）
washerWaterLevel	UInt32	ウォッシャー液の残量レベル（0〜100%）。0%：残量無し，100%：満タン

ワイパーの主状態（mainStatus）は，高頻度の状態遷移およびイベントの発生を防ぐため，「停止中」，「払拭動作中」，「故障中」，「凍結中」の状態をとる。

- **停止中（Stopped）**：デフォルト状態で，払拭動作が行われていない状態。この状態ではAPIからの制御が受け付けられ，ロック取得が可能である。
- **払拭動作中（Moving）**：startMoveによって設定された速度レベルで払拭ブレードが往復動作している状態。指定された速度レベルでの払拭制御が進行中である。APIからの新しい制御が呼び出された場合，後勝ちの原則により速度レベルが変更される。
- **故障中（Fault）**：ワイパーが故障を検知した状態。この状態ではAPIからの制御を受け付けず，ロック機能も取得できない。
- **凍結中（Frozen）**：ワイパーの払拭ブレードが凍結により動作できない状態を検知した状態。この状態ではAPIからの制御を受け付けず，ロック機能も取得できない。

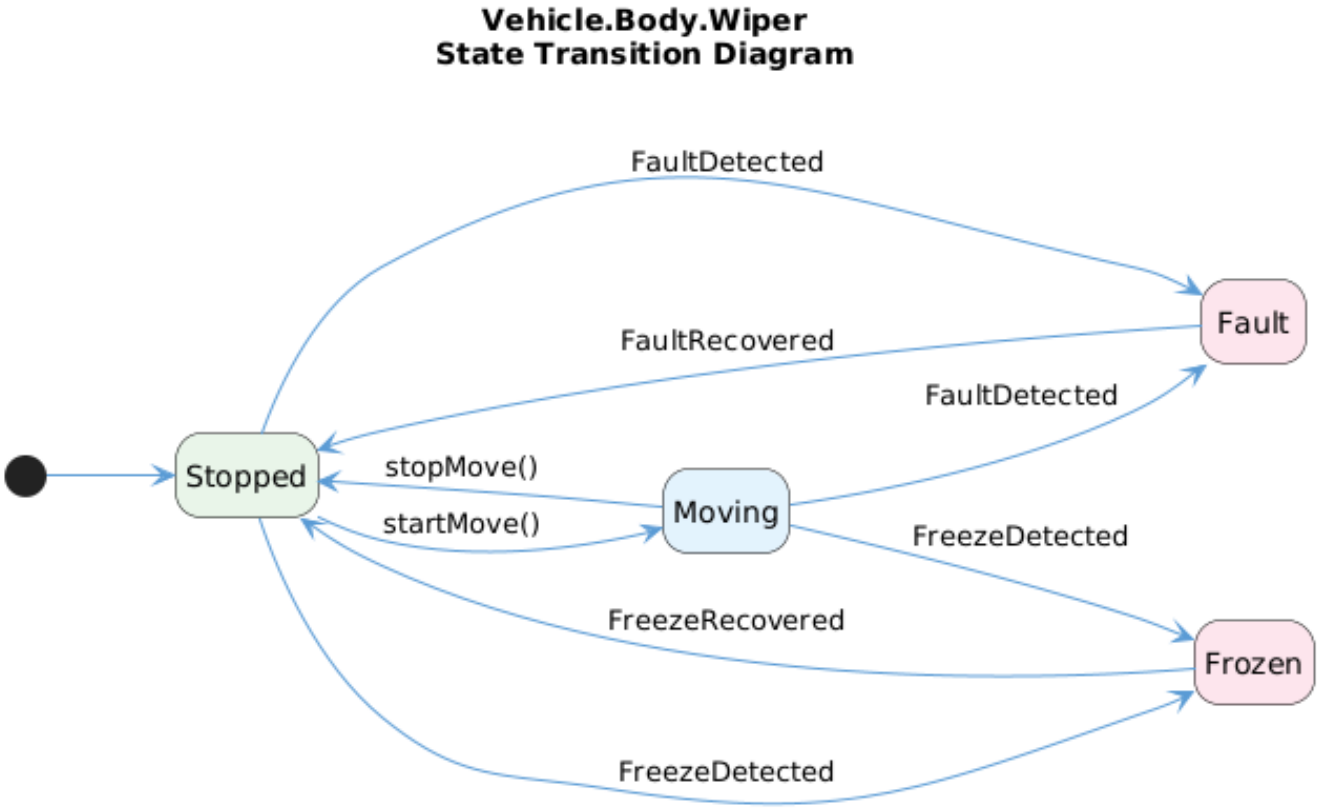


Figure 20. ワイパーの主状態の状態遷移

4.29.6. サービスコール一覧

Wiperに対するサービスコールは以下の通り。

getConfigAll	すべてのワイパーの構成情報の参照
getStatus	ワイパーの状態の参照
startMove	ワイパーの動作の開始
stopMove	ワイパーの動作の停止
startWasher	ウォッシャーの動作の開始
stopWasher	ウォッシャーの動作の停止
lock	ワイパーのロックの取得

unlock	ワイパーのロックの解除
notify	ワイパーイベントの通知要求
unnotify	ワイパーイベントの通知停止
getEvent	ワイパーイベントの取得

4.29.7. イベント

ワイパーに対するイベント種別には，MovableObjectの持つイベントとLockableObjectが持つイベントに加えて，故障検知と凍結検知がある。

Wiperに対するイベント種別は以下の通り。

ControlledBySelf	自アプリケーションが制御
ControlledByOther	他のアプリケーションが制御
TargetReached	指定された目標位置に到達して停止，または，移動停止が完了
FaultDetected	故障検知
FaultRecovered	故障から回復
FreezeDetected	凍結検知
FreezeRecovered	凍結から回復
OwnLockRevoked	自アプリケーションのロックの強制解除
OtherLockReleased	他のアプリケーション等によるロック解除
EventOverflow	イベントキューのオーバフロー
ConfigChanged	構成情報の変更

この内，イベント種別がControlledByOtherの場合には，以下の関連情報を持つ。

sourceApplication	ApplicationIdType	操作したアプリケーション

また，イベント種別がEventOverflowの場合には，以下の関連情報を持つ。

noLostEvents	UInt32	失われたイベントの数

4.29.8. データ型の定義

4.29.8.1. ワイパーの機能クラス

```
type WiperCapabilityType = enum {
    Movable,
    HasWasher
};
```

4.29.8.2. ワイパーに対するリスククラス

```
type WiperRiskType = enum {
    RiskWipingOnTrip,
    RiskWiperWearing,
    RiskWiperBatteryDrain,
    RiskWiperHighFrequency
};
```

4.29.8.3. ワイパーの構成情報（getConfigAllが返すデータ型）

```
type WiperConfigType = struct {
    instanceId : IdType,
    placement : PlacementType,
    capabilities : enumSet(WiperCapabilityType),
    riskClasses : enumSet(WiperRiskType),
    mountedOn : RelatedObjectType,
    displayName : String
};
```

4.29.8.4. ワイパー状態

```
type WiperMainStatusType = enum {
    Stopped,    // 停止中。払拭動作が行われていない状態。
この状態ではAPIからの制御が受け付けられ，ロック取得が可能である。
    Moving, // 払拭動作中。startMoveによって設定された速度レベルで払拭ブレードが往復動
作している状態。
指定された速度レベルでの払拭制御が進行中である。
    Fault,  // 故障中。ワイパーが故障を検知した状態。
この状態ではAPIからの制御を受け付けず，ロック機能も取得できない。
    Frozen  // 凍結中。ワイパーの払拭ブレードが凍結により動作できない状態を検知した状
態。
この状態ではAPIからの制御を受け付けず，ロック機能も取得できない。
};
```

```
type WiperWasherStatusType = enum {
    False,  // 停止
    True    // 作動中
};
```

```
type WiperStatusType = struct {
    mainStatus : WiperMainStatusType,
    lockStatus : LockStatusType,
    frequencyLevel : UInt32,
    intervalLevel : UInt32,
```

```
    wiperWearLevel : UInt32,
    washerStatus : WiperWasherStatusType,
    washerWaterLevel : UInt32
};
```

ワイパーの状態情報

4.29.8.5. ワイパーイベント

```
type WiperEventKindType = enum {
    ControlledBySelf,
    ControlledByOther,
    TargetReached,
    FaultDetected,
    FaultRecovered,
    FreezeDetected,
    FreezeRecovered,
    OwnLockRevoked,
    OtherLockReleased,
    EventOverflow,
    ConfigChanged
};
```

```
type WiperEventType = struct {
    instanceId : IdType,
    eventInfo : variant(WiperEventKindType) {
        ControlledByOther {
            sourceApplication : ApplicationIdType
        },
        EventOverflow {
            noLostEvents : UInt32
        }
    }
};
```

4.29.9. サービスコールの詳細規定

getConfigAll

すべてのワイパーの構成情報の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

config	WiperConfigType[]	すべてのワイパーに関する構成情報

【エラーコード】

なし

【機能】

すべてのワイパーに関する構成情報を返す。

getStatus

ワイパーの状態の参照

【パラメータ】

instanceId	IdType	操作対象のワイパーのID

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
status	WiperStatusType	対象ワイパーの状態

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外

【機能】

操作対象のワイパーに関するすべての状態を返す。

startMove

ワイパーの動作の開始

【パラメータ】

instanceId	IdType	操作対象のワイパーのID
frequencyLevel	UInt32	動作速度レベル（0〜100%）。
intervalLevel	UInt32?	間欠動作による各サイクル後の停止時間（0〜30秒）。
priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外

E_INVALID_PARAMETER	frequencyLevel，intervalLevel，priorityが有効範囲外
E_RISK_APPLICATION	アプリケーションが対応していないリスクがある
E_RISK_USER	ユーザがリスクを承認していない
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_FAULT	操作対象のワイパーが故障中
E_OBJECT_STATUS	操作対象のワイパーが凍結中
E_OBJECT_LOCKED	操作対象のワイパーが，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている

【機能】

動作速度レベルをfrequencyLevelに設定し，操作対象のワイパーの動作を開始する。

frequencyLevelは車両ごとのワイパー性能に応じた速度で動作し，100%はその車両のワイパーが動作可能な最速のレベルを表す。

0を指定した場合は，動作停止要求ではなく無効値（エラー）として扱う。動作停止にはstopMoveを使用すること。

intervalLevel省略時は連続払拭とし，指定時は各払拭サイクル完了後の停止秒数とする。

リスククラスRiskWipingOnTrip， RiskWiperWearing， RiskWiperBatteryDrainに対応できていない場合には，E_RISK_APPLICATIONエラーまたはE_RISK_USERエラーとなる。



MIST/LOW/HIGHなどの動作モードによって動作させたい場合，それぞれのモードに応じたレベル（%）を別途仕様化すること。例）MIST : 10%， LOW : 30%， HIGH : 50%

stopMove

ワイパーの動作の停止

【パラメータ】

instanceId	IdType	操作対象のワイパーのID
priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_INVALID_PARAMETER	priorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_FAULT	操作対象のワイパーが故障中

E_OBJECT_STATUS	操作対象のワイパーが凍結中
E_OBJECT_LOCKED	操作対象のワイパーが，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている

【機能】

操作対象のワイパーの動作停止を要求する。

ブレードは停止位置から初期位置（パーク位置）への復帰を試みる。

startWasher

ウォッシャーの動作の開始

【パラメータ】

instanceId	IdType	操作対象のワイパーのID
priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_INVALID_PARAMETER	priorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_FAULT	操作対象のワイパーが故障中
E_OBJECT_STATUS	操作対象のワイパーに十分なウォッシャー液がない
E_OBJECT_LOCKED	操作対象のワイパーが，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている
E_NOT_SUPPORTED	ウォッシャー機能を持たない

【機能】

操作対象のワイパーのウォッシャー機能を動作させる。



ウォッシャー液噴射に伴うワイパーブレードの連動払拭（MIST動作など）の有無や払拭回数については，車両の物理仕様に依存する。

stopWasher

ウォッシャーの動作の停止

【パラメータ】

instanceId	IdType	操作対象のワイパーのID
priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_INVALID_PARAMETER	priorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_LOCKED	操作対象のワイパーが，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている
E_NOT_SUPPORTED	ウォッシャー機能を持たない

【機能】

操作対象のワイパーのウォッシャー機能の動作停止を要求する。

lock

ワイパーのロックの取得

【パラメータ】

instanceId	IdType	操作対象のワイパーのID
priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_INVALID_PARAMETER	priorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_LOCKED	ロック対象のワイパーが，priorityと同じかそれより高い優先度でロックされている
E_OBJECT_FAULT	ロック対象のワイパーが故障中
E_OBJECT_STATUS	ロック対象のワイパーが凍結中

【機能】

ロック対象のワイパーを，指定した操作優先度でロックする。

unlock

ワイパーのロックの解除

【パラメータ】

instanceId	IdType	操作対象のワイパーのID

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_OBJECT_STATUS	自アプリケーションが，操作対象のワイパーをロックしていない

【機能】

ロック解除対象のワイパーを，ロック解除する。

notify

ワイパーイベントの通知要求

【パラメータ】

instanceId	IdType?	イベント通知対象のワイパーのID
eventFilter	enumSet (WiperEventKindType)?	通知するイベントの種類の集合

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_INVALID_PARAMETER	eventFilterが有効範囲外

【機能】

イベント通知対象のワイパーで発生したeventFilterで指定したイベントの通知を要求する。イベントを通知するためのイベントキューを生成し，その通知ハンドルを notifyHandle に返す。

instanceIdを省略した場合，すべてのインスタンスで発生したイベントの通知を要求する。eventFilterを省略した場合，すべての種類のイベントの通知を要求する。

unnotify

ワイパーイベントの通知停止

【パラメータ】

notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正

【機能】

notifyHandleで指定した通知を停止する。指定した通知用のイベントキューは削除される。

getEvent

ワイパーイベントの取得

【パラメータ】

notifyHandle	NotifyHandleType	イベントキューを識別するための通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
event	WiperEventType	取得したイベント

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正
E_NO_DATA	イベントキューが空

【機能】

notifyHandleで指定した通知用のイベントキューから，先頭のイベントを取り出す。取り出したイベントは，イベントキューから削除される。



イベントがパラメータ等で渡されてくるプログラミング環境向けの物理APIでは，用意する必要がない。

4.30. Vehicle.Body.WiperSwitch（略称：WiperSwitch）

4.30.1. 位置付けと機能

ワイパーを制御する物理的なスイッチを扱う。スイッチとワイパーは直結せず，制御はOS側（ビークルOS）で行うことを仮定する。

4.30.1.1. ワイパースイッチの位置付けと性質

ワイパースイッチとは，車両のハンドル周辺に設置された，乗員が物理操作できるワイパー制御デバイスである。

スイッチはレバー形およびダイアル式であり，運転席のハンドルに装着されている。

ワイパースイッチはFront（フロント用），Rear（リア用），Washer（ウォッシャー用）の3つの独立した系統を持つ。

スイッチとワイパーは直接接続されず，制御要求はビークルOSが受け取る。

スイッチは入力源として機能し，ワイパー制御の主体にはならない。

4.30.1.2. ワイパースイッチの機能

ワイパースイッチは乗員の操作によるスイッチ入力をOSへ提供し，ワイパーオブジェクトへ制御要求を伝達する役割を持つ。

Front/Rearスイッチは，OFF，MIST，INT，LOW，HIGH，AUTOの6つの状態を持つ。INTモードでは，ダイアルにより0〜30秒の間欠時間を調整することができる。

Washerスイッチは，ON/OFFの2つの状態を持つ。

各スイッチの系統はconfigData内のswitchCategoryによって識別される。



本スイッチAPIは入力状態を取得するものであり，操作対象のWiperインスタンスとの紐付け（関連付け）は，車両の構成情報に基づきOSまたはアプリケーション層で管理・ルーティングされる。スイッチの各モード（LOW, HIGH, INT等）が，対象となるWiper APIの具体的なfrequencyLevel（%）やintervalLevel（秒）のどの値にマッピングされるかは，車両ごとの仕様（ビークルOS等の実装）に依存する。

4.30.1.3. ワイパースイッチに対する制御の調停

ワイパースイッチは入力イベントの発生源であり，自身が制御競合を調停することはない。

入力はOSが一元管理し，ワイパーオブジェクトへルーティングされる。

調停はワイパー側の後勝ち原則に従う。スイッチ操作が他アプリケーションの制御要求と競合した場合，ワイパー側の調停ロジックにより採用または却下が判断される。

スイッチ自身は制御主体ではなく，アプリケーションや自動制御と同様に「入力の一つ」として扱われる。

4.30.2. 機能クラス

WiperSwitchは，以下の機能クラスを持つ。

SwitchControl	ワイパー操作入力を提供する基本機能

4.30.3. 構成情報

WiperSwitchの各インスタンスは，以下の構成情報を持つ。

instanceId	IdType	インスタンスが持つId
placement	PlacementType	インスタンスがある場所
switchCategory	WiperSwitchCategoryType	スイッチ系統（Front，Rear，Washer）
capabilities	enumSet (WiperSwitchCapabilityType)	このスイッチがサポートする操作機能の集合
switchType	String	スイッチ種別（Lever形またはDial形など，OEM依存）
displayName	String	ユーザに表示する呼称を取得するためのキー

上記は，YAML形式で記述した構成記述ファイル中に記述する形式である。構成情報を参照するサービスコールは，この構成情報に対して，次の変換を行った構成情報を返す。

- displayNameKeyに対しては，それを元に現在のロケールに対応する呼称の文字列を取得し，その文字列を返す。

4.30.4. 状態

WiperSwitchの各インスタンスは，以下の状態を持つ。

mainStatus	WiperSwitchMainStatusType	スイッチの入力状態（Front/Rear系統の場合：OFF，MIST，INT，LOW，HIGH，AUTO）
washerStatus	WiperSwitchWasherStatusType	Washerスイッチの入力状態。switchCategoryがWasherの場合に有効（ON，OFF）
intervalTime	UInt32	スイッチがINT状態の場合の間欠時間（秒単位，0〜30）。ダイアル等で調整可能。

4.30.5. サービスコール一覧

WiperSwitchに対するサービスコールは以下の通り。

getConfigAll	すべてのワイパースイッチの構成情報の参照
getStatus	ワイパースイッチの状態参照
notify	ワイパースイッチイベントの通知要求
unnotify	ワイパースイッチイベントの通知停止
getEvent	ワイパースイッチイベントの取得

4.30.6. イベント

ワイパースイッチに関するイベントとして，スイッチ状態変化，故障検出/回復，構成変更，イベントキューのオーバーフローなどを定義する。

WiperSwitchに対するイベント種別は以下の通り。

MainStatusChanged	Front/RearスイッチのmainStatus入力状態が変化

WasherStatusChanged	Washerスイッチの入力状態が変化
IntervalTimeChanged	Intermittent（INT）モード時の間欠時間が変化
FaultDetected	故障検知
FaultRecovered	故障から回復
EventOverflow	イベントキューのオーバフロー
ConfigChanged	構成情報の変更

この内，イベント種別がMainStatusChangedの場合には，以下の関連情報を持つ。

newState	WiperSwitchMainStatusType	スイッチの新しい入力状態

また，イベント種別がWasherStatusChangedの場合には，以下の関連情報を持つ。

newState	WiperSwitchWasherStatusType	Washerスイッチの新しい入力状態

また，イベント種別がIntervalTimeChangedの場合には，以下の関連情報を持つ。

newIntervalTime	UInt32	新しい間欠時間（秒）

また，イベント種別がEventOverflowの場合には，以下の関連情報を持つ。

noLostEvents	UInt32	失われたイベントの数

4.30.7. データ型の定義

4.30.7.1. ワイパースイッチの機能クラス

```
type WiperSwitchCapabilityType = enum {
    SwitchControl
};
```

4.30.7.2. スイッチ系統

```
type WiperSwitchCategoryType = enum {
    Front,  // フロントワイパースイッチ系統
    Rear,   // リアワイパースイッチ系統
    Washer  // ウォッシャースイッチ系統
};
```

4.30.7.3. Front/Rearスイッチの入力状態

```
type WiperSwitchMainStatusType = enum {
```

```
    False,  // OFF - ワイパー停止要求。
    Mist,   // MIST - ミスト動作。スイッチがこの状態にある間，あるいは1回の操作として，一時的なワイパー動作（通常は1回の払拭，または保持中の連続動作）を要求する。

    Intermittent,   // INT - 間欠動作。ダイアル等でintervalTimeを調整し，指定された間隔での払拭を要求する。

    Low,    // LOW - 低速連続動作。ワイパーのfrequencyLevelにおける低速相当のレベルを要求する。

    High,   // HIGH - 高速連続動作。ワイパーのfrequencyLevelにおける高速（最大100%等）相当のレベルを要求する。

    Auto    // AUTO - 自動動作。雨量センサー等の情報に基づき，OSやアプリケーション層が自動的にワイパーのfrequencyLevelやintervalLevelを制御することを要求する。

};
```

4.30.7.4. Washerスイッチの入力状態

```
type WiperSwitchWasherStatusType = enum {
    False,  // OFF - ウォッシャー停止
    True    // ON - ウォッシャー液噴射動作
};
```

ワイパースイッチ構成情報（getConfigAll が返すデータ型）

```
type WiperSwitchConfigType = struct {
    instanceId : IdType,
    placement : PlacementType,
    switchCategory : WiperSwitchCategoryType,
    capabilities : enumSet(WiperSwitchCapabilityType),
    switchType : String,
    displayName : String
};
```

4.30.7.5. ワイパースイッチ状態

```
type WiperSwitchStatusType = struct {
    mainStatus : WiperSwitchMainStatusType,
    washerStatus : WiperSwitchWasherStatusType,
    intervalTime : UInt32
};
```

ワイパースイッチの状態情報

4.30.7.6. ワイパースイッチイベント種別

```
type WiperSwitchEventKindType = enum {
    MainStatusChanged,
    WasherStatusChanged,
    IntervalTimeChanged,
    FaultDetected,
    FaultRecovered,
    EventOverflow,
    ConfigChanged
};
```

```
type WiperSwitchEventType = struct {
    instanceId : IdType,
    switchCategory : WiperSwitchCategoryType,
    eventInfo : variant(WiperSwitchEventKindType) {
        MainStatusChanged {
            newState : WiperSwitchMainStatusType
        },
        WasherStatusChanged {
            newState : WiperSwitchWasherStatusType
        },
        IntervalTimeChanged {
            newIntervalTime : UInt32
        },
        EventOverflow {
            noLostEvents : UInt32
        }
    }
};
```

4.30.8. サービスコールの詳細規定

getConfigAll

すべてのワイパースイッチの構成情報の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
config	WiperSwitchConfigType[]	すべてのワイパースイッチに関する構成情報

【機能】

すべてのワイパースイッチに関する構成情報を返す。

getStatus

ワイパースイッチの状態参照

【パラメータ】

instanceId	IdType	操作対象のワイパースイッチのID

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
status	WiperSwitchStatusType	対象のワイパースイッチの状態

【エラーコード】

E_INVALID_ID	instanceId が有効範囲外

【機能】

操作対象のワイパースイッチに関する状態を返す。



status　には　mainStatus（Front/Rear系統のスイッチ入力状態），washerStatus（Washer系統のスイッチ入力状態），intervalTime（INT時の間欠時間）を含む。

notify

ワイパースイッチイベントの通知要求

【パラメータ】

instanceId	IdType?	イベント通知対象のワイパースイッチID（省略時は全インスタンス）
eventFilter	enumSet (WiperSwitchEventKindType)?	通知するイベント種別の集合（省略時は全種）

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
notifyHandle	NotifyHandleType	通知ハンドル

【エラーコード】

E_INVALID_ID	instanceId が有効範囲外
E_INVALID_PARAMETER	eventFilter が有効範囲外

【機能】

イベント通知対象のワイパースイッチで発生したeventFilterで指定したイベントの通知を要求する。イベントを通知するためのイベントキューを生成し，その通知ハンドルをnotifyHandleに返す。

instanceIdを省略した場合，すべてのインスタンスで発生したイベントの通知を要求する。eventFilterを省略した場合，すべての種類のイベントの通知を要求する。

unnotify

ワイパースイッチイベントの通知停止

【パラメータ】

notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandle が不正

【機能】

notifyHandleで指定した通知を停止する。指定した通知用のイベントキューは削除される。

getEvent

ワイパースイッチイベントの取得

【パラメータ】

notifyHandle	NotifyHandleType	イベントキューを識別する通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
event	WiperSwitchEventType	取得したイベント

【エラーコード】

E_INVALID_HANDLE	notifyHandle が不正
E_NO_DATA	イベントキューが空

【機能】

notifyHandleで指定した通知用のイベントキューから，先頭のイベントを取り出す。取り出したイベントは，イベントキューから削除される。



イベントがパラメータ等で渡されてくるプログラミング環境向けの物理APIでは，用意

する必要がない。

4.31. Vehicle.Powertrain.TractionBattery.Charging（略称：Charging）

4.31.1. 位置付けと機能

Vehicle.Powertrain.TractionBattery.Charging（略称：Charging）は，駆動用バッテリーの充電を制御・監視するためのオブジェクトである。充電ポート（給電口）ごとにインスタンスを持つマルチインスタンスオブジェクトである。



本オブジェクトは VSS v6 の Vehicle.Powertrain.TractionBattery.Charging 配下（ChargingPort.*.IsFlapOpen/IsChargingCableConnected, StartStopCharging, ChargeLimit, ChargeCurrent, ChargeVoltage, ChargeRate, TimeToComplete 等）に対応する。給電口の開閉・充電の開始/停止・充電上限の設定を操作系サービスコールとして提供する。

4.31.1.1. 充電制御の調停とロック

充電の開始・停止・上限設定には，標準的な調停機能（後勝ちの原則）が適用される。安全上の理由（故障検知・ケーブル未接続など）がある場合，操作はエラーとなる。

4.31.2. 機能クラス

Chargingは，以下の機能クラスを持つ。

FlapControllable	給電口フラップをAPIにより開閉できる（VSS: IsFlapOpen actuator）
ChargeControllable	充電の開始・停止をAPIにより制御できる（VSS: StartStopCharging）
ChargeLimitSettable	充電上限（目標SoC）を設定できる（VSS: ChargeLimit）

4.31.3. リスククラス

Chargingは，以下のリスククラスを持つ。

T.B.D	T.B.D

4.31.4. 構成情報

Chargingの各インスタンスは，以下の構成情報を持つ。

instanceId	IdType	インスタンスが持つId（充電ポートを識別）
capabilities	enumSet (ChargingCapabilityType)	その充電ポートが持つ機能
supportedInletTypes	String[]	対応するインレット種別（VSS: SupportedInletTypes）

displayName	String	ユーザに表示する呼称を取得するためのキー

4.31.5. 状態

Chargingの各インスタンスは，以下の状態を持つ。

mainStatus	ChargingMainStatusType	主状態
isCableConnected	Boolean	充電ケーブルの接続状態（VSS: IsChargingCableConnected）
isFlapOpen	Boolean	給電口フラップの開閉状態（VSS: IsFlapOpen）
chargeLimit	PercentType	充電上限（目標SoC・単位：%，VSS: ChargeLimit）
chargeCurrent	CurrentType?	充電電流（単位：A，VSS: ChargeCurrent）
chargeVoltage	VoltageType?	充電電圧（単位：V，VSS: ChargeVoltage）
chargeRate	ChargeRateType?	充電レート（単位：km/h，VSS: ChargeRate）
timeToComplete	UInt32?	充電完了までの推定残り時間（単位：秒，VSS: TimeToComplete）

chargeCurrent・chargeVoltage・chargeRate・timeToCompleteは，充電中（mainStatusがCharging）のときにのみ有効な値を持つ。

4.31.6. サービスコール一覧

Chargingに対するサービスコールは以下の通り。

getConfigAll	すべての充電ポートの構成情報の参照
getStatus	充電ポートの状態の参照
openChargePort	給電口（フラップ）の開放
closeChargePort	給電口（フラップ）の閉鎖
startCharging	充電の開始
stopCharging	充電の停止
setChargeLimit	充電上限（目標SoC）の設定
notify	充電イベントの通知要求
unnotify	充電イベントの通知停止
getEvent	充電イベントの取得

4.31.7. イベント

充電に対するイベント種別には，充電の開始/停止，ケーブルの接続/切断，目標到達，故障の検知/回復がある。

Chargingに対するイベント種別は以下の通り。

ChargingStarted	充電開始
ChargingStopped	充電停止
CableConnected	充電ケーブル接続検知
CableDisconnected	充電ケーブル切断検知
TargetReached	目標充電量（chargeLimit）に到達し充電完了
FaultDetected	故障検知
FaultRecovered	故障から回復
EventOverflow	イベントキューのオーバフロー
ConfigChanged	構成情報の変更

この内，イベント種別がEventOverflowの場合には，以下の関連情報を持つ。

noLostEvents	UInt32	失われたイベントの数

4.31.8. データ型の定義

4.31.8.1. 充電の機能クラス

```
type ChargingCapabilityType = enum {
    FlapControllable,
    ChargeControllable,
    ChargeLimitSettable
};
```

4.31.8.2. 充電に対するリスククラス

```
type ChargingRiskType = enum {
    T.B.D
};
```

4.31.8.3. 充電の構成情報（getConfigAllが返すデータ型）

```
type ChargingConfigType = struct {
    instanceId : IdType,
    capabilities : enumSet(ChargingCapabilityType),
    supportedInletTypes : String[],
    displayName : String
};
```

4.31.8.4. 割合・電流・電圧・充電レートの型

```
type PercentType = Float32; // 割合（単位：%）
```

PercentTypeの有効範囲は0.0以上100.0以下とする。値が測定できない場合は，特殊値Unknown（-1.0）とする。本型はENERGYの各オブジェクトで共通に用いる。

```
type CurrentType = Float32; // 電流（単位：A）
```

```
type VoltageType = Float32; // 電圧（単位：V）
```

```
type ChargeRateType = Float32;  // 充電レート（単位：km/h）
```

4.31.8.5. 充電状態

```
type ChargingMainStatusType = enum {
    Idle,   // 非充電（待機）
    Charging,   // 充電中
    Completed,  // 充電完了
    Fault   // 故障中
};
```

```
type ChargingStatusType = struct {
    mainStatus : ChargingMainStatusType,
    isCableConnected : Boolean,
    isFlapOpen : Boolean,
    chargeLimit : PercentType,
    chargeCurrent : CurrentType?,
    chargeVoltage : VoltageType?,
    chargeRate : ChargeRateType?,
    timeToComplete : UInt32?
};
```

chargeCurrent・chargeVoltage・chargeRate・timeToCompleteは充電中にのみ値を持つ。

4.31.8.6. 充電イベント

```
type ChargingEventKindType = enum {
    ChargingStarted,
    ChargingStopped,
    CableConnected,
    CableDisconnected,
    TargetReached,
    FaultDetected,
```

```
    FaultRecovered,
    EventOverflow,
    ConfigChanged
};
```

```
type ChargingEventType = struct {
    instanceId : IdType,
    eventInfo : variant(ChargingEventKindType) {
        EventOverflow {
            noLostEvents : UInt32
        }
    }
};
```

4.31.9. サービスコールの詳細規定

getConfigAll

すべての充電ポートの構成情報の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
config	ChargingConfigType[]	すべての充電ポートに関する構成情報

【エラーコード】

なし

【機能】

すべての充電ポートに関する構成情報を返す。

getStatus

充電ポートの状態の参照

【パラメータ】

instanceId	IdType	参照対象の充電ポートのID

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
status	ChargingStatusType	対象充電ポートの状態

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外

【機能】

参照対象の充電ポートに関するすべての状態（接続状態・充電状態・電流電圧等）を返す。

openChargePort

給電口（フラップ）の開放

【パラメータ】

instanceId	IdType	操作対象の充電ポートのID
priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_INVALID_PARAMETER	priorityが有効範囲外
E_NOT_SUPPORTED	給電口フラップの開閉機能を持たない
E_OBJECT_FAULT	操作対象の充電ポートが故障中

【機能】

操作対象の充電ポートの給電口フラップを開放する（VSS: IsFlapOpen = true）。

closeChargePort

給電口（フラップ）の閉鎖

【パラメータ】

instanceId	IdType	操作対象の充電ポートのID
priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_INVALID_PARAMETER	priorityが有効範囲外

E_NOT_SUPPORTED	給電口フラップの開閉機能を持たない
E_OBJECT_STATUS	充電ケーブルが接続中のため閉鎖できない
E_OBJECT_FAULT	操作対象の充電ポートが故障中

【機能】

操作対象の充電ポートの給電口フラップを閉鎖する（VSS: IsFlapOpen = false）。

startCharging

充電の開始

【パラメータ】

instanceId	IdType	操作対象の充電ポートのID
priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_INVALID_PARAMETER	priorityが有効範囲外
E_NOT_SUPPORTED	充電制御機能を持たない
E_OBJECT_STATUS	充電ケーブルが未接続，または既に充電中
E_OBJECT_FAULT	操作対象の充電ポートが故障中

【機能】

操作対象の充電ポートで充電を開始する（VSS: StartStopCharging = startCharging）。ケーブル接続・認証が完了している必要がある。

stopCharging

充電の停止

【パラメータ】

instanceId	IdType	操作対象の充電ポートのID
priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_INVALID_PARAMETER	priorityが有効範囲外
E_NOT_SUPPORTED	充電制御機能を持たない
E_OBJECT_STATUS	充電中でない
E_OBJECT_FAULT	操作対象の充電ポートが故障中

【機能】

操作対象の充電ポートで充電を停止する（VSS: StartStopCharging = stopCharging）。

setChargeLimit

充電上限（目標SoC）の設定

【パラメータ】

instanceId	IdType	操作対象の充電ポートのID
chargeLimit	PercentType	充電上限（目標SoC・単位：%）
priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_INVALID_PARAMETER	chargeLimitまたはpriorityが有効範囲外
E_NOT_SUPPORTED	充電上限の設定機能を持たない
E_OBJECT_FAULT	操作対象の充電ポートが故障中

【機能】

充電上限（目標SoC）をchargeLimitに設定する（VSS: ChargeLimit）。chargeLimitに特殊値Unknown（-1.0）を指定した場合は，E_INVALID_PARAMETERエラーとなる。

notify

充電イベントの通知要求

【パラメータ】

instanceId	IdType?	イベント通知対象の充電ポートのID
eventFilter	enumSet (ChargingEventKindType)?	通知するイベントの種類の集合

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_INVALID_PARAMETER	eventFilterが有効範囲外

【機能】

イベント通知対象の充電ポートで発生したeventFilterで指定したイベントの通知を要求する。イベントを通知するためのイベントキューを生成し，その通知ハンドルを notifyHandle に返す。

unnotify

充電イベントの通知停止

【パラメータ】

notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正

【機能】

notifyHandleで指定した通知を停止する。指定した通知用のイベントキューは削除される。

getEvent

充電イベントの取得

【パラメータ】

notifyHandle	NotifyHandleType	イベントキューを識別するための通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
event	ChargingEventType	取得したイベント

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正
E_NO_DATA	イベントキューが空

【機能】

notifyHandleで指定した通知用のイベントキューから，先頭のイベントを取り出す。取り出したイベントは，イベントキューから削除される。



イベントがパラメータ等で渡されてくるプログラミング環境向けの物理APIでは，用意する必要がない。

4.32. Vehicle.Powertrain.TractionBattery（略称：TractionBattery）

4.32.1. 位置付けと機能

Vehicle.Powertrain.TractionBattery（略称：TractionBattery）は，車両の駆動用バッテリーの状態（充電状態SoC・健全度SoH・容量・温度など）を参照するためのマルチインスタンスオブジェクトである。充電の制御は別オブジェクト Vehicle.Powertrain.TractionBattery.Charging で扱う。



本オブジェクトの状態は VSS v6 の Vehicle.Powertrain.TractionBattery 配下（StateOfCharge, StateOfHealth, GrossCapacity, Temperature 等）に対応する。参照専用であり，状態を変更するサービスコールは持たない。

4.32.1.1. SoCの定義

SoCは，駆動用バッテリーの満充電容量に対する残容量の割合をパーセンテージ（0：空，100：満充電）で表したものと定義する。SoCには，バッテリー管理システムが推定した実SoC（socActual）と，ユーザに表示するための表示SoC（socDisplay）の2種類がある（VSS: StateOfCharge.Current / StateOfCharge.Displayed）。

4.32.1.2. TractionBatteryの機能

TractionBatteryは参照専用のオブジェクトである。SoC・SoH・温度が推定できない場合（起動直後や故障中など）は，対応する値として特殊値Unknownを返す。

4.32.2. 機能クラス

TractionBatteryは，以下の機能クラスを持つ。

ActualSocAvailable	実SoC（socActual）を参照することができる
SohAvailable	健全度（SoH）を参照することができる
TemperatureAvailable	バッテリー温度を参照することができる

4.32.3. リスククラス

TractionBatteryは，以下のリスククラスを持つ。

T.B.D	T.B.D

4.32.4. 構成情報

TractionBatteryの各インスタンスは，以下の構成情報を持つ。

instanceId	IdType	インスタンスが持つId
capabilities	enumSet (TractionBatteryCapability Type)	その駆動用バッテリーが持つ機能
grossCapacity	UInt32	総容量（公称容量・単位：Wh）。VSS: GrossCapacity
displayName	String	ユーザに表示する呼称を取得するためのキー

4.32.5. 状態

TractionBatteryの各インスタンスは，以下の状態を持つ。

mainStatus	TractionBatteryMainStatus Type	主状態
socDisplay	SocType	表示SoC（VSS: StateOfCharge.Displayed）
socActual	SocType?	実SoC（VSS: StateOfCharge.Current）
stateOfHealth	PercentType?	健全度SoH（VSS: StateOfHealth・単位：%）
temperature	TemperatureType?	バッテリー温度（VSS: Temperature.Average・単位：℃）

ここで，socActualは機能クラスActualSocAvailable，stateOfHealthはSohAvailable，temperatureはTemperatureAvailableを備えている場合にのみ有効である。

4.32.6. サービスコール一覧

TractionBatteryに対するサービスコールは以下の通り。

getConfigAll	すべての駆動用バッテリーの構成情報の参照
getStatus	駆動用バッテリーの状態の参照
notify	駆動用バッテリーイベントの通知要求
unnotify	駆動用バッテリーイベントの通知停止
getEvent	駆動用バッテリーイベントの取得

4.32.7. イベント

駆動用バッテリーに対するイベント種別には，SoCの変化と故障の検知・回復がある。

TractionBatteryに対するイベント種別は以下の通り。

SocChanged	表示SoCが1%以上変化
FaultDetected	故障検知
FaultRecovered	故障から回復
EventOverflow	イベントキューのオーバフロー
ConfigChanged	構成情報の変更

この内，イベント種別がSocChangedの場合には，以下の関連情報を持つ。

socDisplay	SocType	変化後の表示SoC

また，イベント種別がEventOverflowの場合には，以下の関連情報を持つ。

noLostEvents	UInt32	失われたイベントの数

4.32.8. データ型の定義

4.32.8.1. 駆動用バッテリーの機能クラス

```
type TractionBatteryCapabilityType = enum {
    ActualSocAvailable,
    SohAvailable,
    TemperatureAvailable
};
```

4.32.8.2. 駆動用バッテリーに対するリスククラス

```
type TractionBatteryRiskType = enum {
    T.B.D
};
```

4.32.8.3. 駆動用バッテリーの構成情報（getConfigAllが返すデータ型）

```
type TractionBatteryConfigType = struct {
    instanceId : IdType,
    capabilities : enumSet(TractionBatteryCapabilityType),
    grossCapacity : UInt32,
    displayName : String
};
```

4.32.8.4. SoC・温度の型

```
type SocType = Float32; // SoC（単位：%）
```

SocTypeの有効範囲は0.0以上100.0以下とする。SoCが推定できない場合は，特殊値Unknown（-1.0）とする。

```
type TemperatureType = Float32; // 温度（単位：℃）。取得不能時はUnknown（-1000.0）。
```

4.32.8.5. 駆動用バッテリー状態

```
type TractionBatteryMainStatusType = enum {
    Normal, // 正常
    Fault   // 故障中
};
```

```
type TractionBatteryStatusType = struct {
    mainStatus : TractionBatteryMainStatusType,
    socDisplay : SocType,
    socActual : SocType?,
    stateOfHealth : PercentType?,
    temperature : TemperatureType?
};
```

socActual・stateOfHealth・temperatureは，対応する機能クラスを備えている場合にのみ値を持つ。

4.32.8.6. 駆動用バッテリーイベント

```
type TractionBatteryEventKindType = enum {
    SocChanged,
    FaultDetected,
    FaultRecovered,
    EventOverflow,
    ConfigChanged
};
```

```
type TractionBatteryEventType = struct {
    instanceId : IdType,
    eventInfo : variant(TractionBatteryEventKindType) {
        SocChanged {
            socDisplay : SocType
        },
        EventOverflow {
            noLostEvents : UInt32
        }
    }
};
```

4.32.9. サービスコールの詳細規定

getConfigAll

すべての駆動用バッテリーの構成情報の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
config	TractionBatteryConfigType[]	すべての駆動用バッテリーに関する構成情報

【エラーコード】

なし

【機能】

すべての駆動用バッテリーに関する構成情報を返す。

getStatus

駆動用バッテリーの状態の参照

【パラメータ】

instanceId	IdType	参照対象の駆動用バッテリーのID

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
status	TractionBatteryStatusType	対象駆動用バッテリーの状態

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外

【機能】

参照対象の駆動用バッテリーに関するすべての状態（SoC・SoH・温度を含む）を返す。

notify

駆動用バッテリーイベントの通知要求

【パラメータ】

instanceId	IdType?	イベント通知対象の駆動用バッテリーのID

eventFilter	enumSet (TractionBatteryEventKindType)?	通知するイベントの種類の集合

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_INVALID_PARAMETER	eventFilterが有効範囲外

【機能】

イベント通知対象の駆動用バッテリーで発生したeventFilterで指定したイベントの通知を要求する。イベントを通知するためのイベントキューを生成し，その通知ハンドルを notifyHandle に返す。

instanceIdを省略した場合，すべてのインスタンスで発生したイベントの通知を要求する。eventFilterを省略した場合，すべての種類のイベントの通知を要求する。

unnotify

駆動用バッテリーイベントの通知停止

【パラメータ】

notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正

【機能】

notifyHandleで指定した通知を停止する。指定した通知用のイベントキューは削除される。

getEvent

駆動用バッテリーイベントの取得

【パラメータ】

notifyHandle	NotifyHandleType	イベントキューを識別するための通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
event	TractionBatteryEventType	取得したイベント

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正
E_NO_DATA	イベントキューが空

【機能】

notifyHandleで指定した通知用のイベントキューから，先頭のイベントを取り出す。取り出したイベントは，イベントキューから削除される。



イベントがパラメータ等で渡されてくるプログラミング環境向けの物理APIでは，用意する必要がない。

4.33. Vehicle.Powertrain.CombustionEngine（略称：CombustionEngine）

4.33.1. 位置付けと機能

Vehicle.Powertrain.CombustionEngine（略称：CombustionEngine）は，内燃機関（エンジン）の運転状態を参照するためのマルチインスタンスオブジェクトである。



本オブジェクトの状態は VSS v6 の Vehicle.Powertrain.CombustionEngine 配下（IsRunning, Speed 等）に対応する。参照専用であり，状態を変更するサービスコールは持たない。

4.33.1.1. CombustionEngineの機能

エンジンの始動／停止の検知（isRunning）と，回転数（speed，単位：rpm）の参照を提供する。回転数が取得できない場合は特殊値Unknownを返す。

4.33.2. 機能クラス

CombustionEngineは，以下の機能クラスを持つ。

RunningDetectable	エンジンの運転状態（始動/停止）を検出できる（VSS: IsRunning）
SpeedAvailable	エンジン回転数を参照できる（VSS: Speed）

4.33.3. リスククラス

CombustionEngineは，以下のリスククラスを持つ。

T.B.D	T.B.D

4.33.4. 構成情報

CombustionEngineの各インスタンスは，以下の構成情報を持つ。

instanceId	IdType	インスタンスが持つId
capabilities	enumSet (CombustionEngineCapabilityType)	そのエンジンが持つ機能
displacement	UInt16?	排気量（単位：cm^3，VSS: Displacement）
maxPower	UInt16?	最高出力（単位：kW，VSS: MaxPower）
displayName	String	ユーザに表示する呼称を取得するためのキー

4.33.5. 状態

CombustionEngineの各インスタンスは，以下の状態を持つ。

mainStatus	CombustionEngineMainStatusType	主状態
isRunning	Boolean	エンジン運転状態（VSS: IsRunning）
speed	RpmType?	エンジン回転数（単位：rpm，VSS: Speed）

ここで，speedは機能クラスSpeedAvailableを備えている場合にのみ有効である。

4.33.6. サービスコール一覧

CombustionEngineに対するサービスコールは以下の通り。

getConfigAll	すべてのエンジンの構成情報の参照
getStatus	エンジンの状態の参照
notify	エンジンイベントの通知要求
unnotify	エンジンイベントの通知停止
getEvent	エンジンイベントの取得

4.33.7. イベント

エンジンに対するイベント種別には，始動／停止と故障の検知・回復がある。

CombustionEngineに対するイベント種別は以下の通り。

EngineStarted	エンジン始動を検知
EngineStopped	エンジン停止を検知
FaultDetected	故障検知
FaultRecovered	故障から回復

EventOverflow	イベントキューのオーバフロー
ConfigChanged	構成情報の変更

この内，イベント種別がEventOverflowの場合には，以下の関連情報を持つ。

noLostEvents	UInt32	失われたイベントの数

4.33.8. データ型の定義

4.33.8.1. エンジンの機能クラス

```
type CombustionEngineCapabilityType = enum {
    RunningDetectable,
    SpeedAvailable
};
```

4.33.8.2. エンジンに対するリスククラス

```
type CombustionEngineRiskType = enum {
    T.B.D
};
```

4.33.8.3. エンジンの構成情報（getConfigAllが返すデータ型）

```
type CombustionEngineConfigType = struct {
    instanceId : IdType,
    capabilities : enumSet(CombustionEngineCapabilityType),
    displacement : UInt16?,
    maxPower : UInt16?,
    displayName : String
};
```

4.33.8.4. 回転数の型

```
type RpmType = Float32; // エンジン回転数（単位：rpm）。取得不能時はUnknown（-1.0）。
```

4.33.8.5. エンジン状態

```
type CombustionEngineMainStatusType = enum {
    Stopped,    // 停止中
    Running,    // 運転中
    Fault   // 故障中
};
```

```
type CombustionEngineStatusType = struct {
    mainStatus : CombustionEngineMainStatusType,
    isRunning : Boolean,
    speed : RpmType?
};
```

speedは，機能クラスSpeedAvailableを備えている場合にのみ値を持つ。

4.33.8.6. エンジンイベント

```
type CombustionEngineEventKindType = enum {
    EngineStarted,
    EngineStopped,
    FaultDetected,
    FaultRecovered,
    EventOverflow,
    ConfigChanged
};
```

```
type CombustionEngineEventType = struct {
    instanceId : IdType,
    eventInfo : variant(CombustionEngineEventKindType) {
        EventOverflow {
            noLostEvents : UInt32
        }
    }
};
```

4.33.9. サービスコールの詳細規定

getConfigAll

すべてのエンジンの構成情報の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
config	CombustionEngineConfigType[]	すべてのエンジンに関する構成情報

【エラーコード】

なし

【機能】

すべてのエンジンに関する構成情報を返す。

getStatus

エンジンの状態の参照

【パラメータ】

instanceId	IdType	参照対象のエンジンのID

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
status	CombustionEngineStatusType	対象エンジンの状態

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外

【機能】

参照対象のエンジンの状態（運転状態・回転数を含む）を返す。

notify

エンジンイベントの通知要求

【パラメータ】

instanceId	IdType?	イベント通知対象のエンジンのID
eventFilter	enumSet (CombustionEngineEventKindType)?	通知するイベントの種類の集合

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_INVALID_PARAMETER	eventFilterが有効範囲外

【機能】

イベント通知対象のエンジンで発生したeventFilterで指定したイベントの通知を要求する。イベントを通知するためのイベントキューを生成し，その通知ハンドルを notifyHandle に返す。

unnotify

エンジンイベントの通知停止

【パラメータ】

notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正

【機能】

notifyHandleで指定した通知を停止する。指定した通知用のイベントキューは削除される。

getEvent

エンジンイベントの取得

【パラメータ】

notifyHandle	NotifyHandleType	イベントキューを識別するための通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
event	CombustionEngineEventType	取得したイベント

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正
E_NO_DATA	イベントキューが空

【機能】

notifyHandleで指定した通知用のイベントキューから，先頭のイベントを取り出す。取り出したイベントは，イベントキューから削除される。



イベントがパラメータ等で渡されてくるプログラミング環境向けの物理APIでは，用意する必要がない。

4.34. Vehicle.Powertrain.FuelSystem（略称：FuelSystem）

4.34.1. 位置付けと機能

Vehicle.Powertrain.FuelSystem（略称：FuelSystem）は，燃料系（燃料残量・燃費・燃料種別など）を参照するためのマルチインスタンスオブジェクトである。



本オブジェクトの状態・構成は VSS v6 の Vehicle.Powertrain.FuelSystem 配下（SupportedFuelTypes, RelativeLevel, InstantConsumption, Range 等）に対応する。参照専用である。

4.34.1.1. FuelSystemの機能

燃料種別（supportedFuelTypes）・燃料残量割合（relativeLevel）・瞬間燃料消費（instantConsumption）・航続可能距離（range）を参照する。燃料消費要因分離のための基礎データを提供する。

4.34.2. 機能クラス

FuelSystemは，以下の機能クラスを持つ。

ConsumptionAvailable	瞬間燃料消費を参照できる（VSS: InstantConsumption）
RangeAvailable	航続可能距離を参照できる（VSS: Range）

4.34.3. リスククラス

FuelSystemは，以下のリスククラスを持つ。

T.B.D	T.B.D

4.34.4. 構成情報

FuelSystemの各インスタンスは，以下の構成情報を持つ。

instanceId	IdType	インスタンスが持つId
capabilities	enumSet (FuelSystemCapabilityType)	その燃料系が持つ機能
supportedFuelTypes	String[]	対応する燃料種別（VSS: SupportedFuelTypes）
tankCapacity	Float32?	タンク容量（単位：L，VSS: TankCapacity）
displayName	String	ユーザに表示する呼称を取得するためのキー

4.34.5. 状態

FuelSystemの各インスタンスは，以下の状態を持つ。

mainStatus	FuelSystemMainStatusType	主状態
relativeLevel	PercentType	燃料残量割合（単位：%，VSS: RelativeLevel）
instantConsumption	ConsumptionType?	瞬間燃料消費（単位：L/100km，VSS: InstantConsumption）
range	UInt32?	航続可能距離（単位：m，VSS: Range）
isFuelLevelLow	Boolean	燃料残量低下フラグ（VSS: IsFuelLevelLow）

ここで，instantConsumptionは機能クラスConsumptionAvailable，rangeはRangeAvailableを備えている場合にのみ有効である。

4.34.6. サービスコール一覧

FuelSystemに対するサービスコールは以下の通り。

getConfigAll	すべての燃料系の構成情報の参照
getStatus	燃料系の状態の参照
notify	燃料系イベントの通知要求
unnotify	燃料系イベントの通知停止
getEvent	燃料系イベントの取得

4.34.7. イベント

燃料系に対するイベント種別には，燃料残量低下・空と故障の検知・回復がある。

FuelSystemに対するイベント種別は以下の通り。

FuelLevelLow	燃料残量が低下（VSS: IsFuelLevelLow）
FuelLevelEmpty	燃料残量が空（VSS: IsFuelLevelEmpty）
FaultDetected	故障検知
FaultRecovered	故障から回復
EventOverflow	イベントキューのオーバフロー
ConfigChanged	構成情報の変更

この内，イベント種別がEventOverflowの場合には，以下の関連情報を持つ。

noLostEvents	UInt32	失われたイベントの数

4.34.8. データ型の定義

4.34.8.1. 燃料系の機能クラス

```
type FuelSystemCapabilityType = enum {
    ConsumptionAvailable,
    RangeAvailable
};
```

4.34.8.2. 燃料系に対するリスククラス

```
type FuelSystemRiskType = enum {
    T.B.D
};
```

4.34.8.3. 燃料系の構成情報（getConfigAllが返すデータ型）

```
type FuelSystemConfigType = struct {
    instanceId : IdType,
    capabilities : enumSet(FuelSystemCapabilityType),
    supportedFuelTypes : String[],
    tankCapacity : Float32?,
    displayName : String
};
```

4.34.8.4. 消費の型

```
type ConsumptionType = Float32; // 燃料消費（単位：L/100km）
```

4.34.8.5. 燃料系状態

```
type FuelSystemMainStatusType = enum {
    Normal, // 正常
    Fault   // 故障中
};
```

```
type FuelSystemStatusType = struct {
    mainStatus : FuelSystemMainStatusType,
    relativeLevel : PercentType,
    instantConsumption : ConsumptionType?,
    range : UInt32?,
    isFuelLevelLow : Boolean
};
```

instantConsumption・rangeは，対応する機能クラスを備えている場合にのみ値を持つ。

4.34.8.6. 燃料系イベント

```
type FuelSystemEventKindType = enum {
    FuelLevelLow,
    FuelLevelEmpty,
    FaultDetected,
    FaultRecovered,
    EventOverflow,
    ConfigChanged
};
```

```
type FuelSystemEventType = struct {
    instanceId : IdType,
    eventInfo : variant(FuelSystemEventKindType) {
        EventOverflow {
            noLostEvents : UInt32
        }
    }
};
```

4.34.9. サービスコールの詳細規定

getConfigAll

すべての燃料系の構成情報の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
config	FuelSystemConfigType[]	すべての燃料系に関する構成情報

【エラーコード】

なし

【機能】

すべての燃料系に関する構成情報（対応燃料種別・タンク容量を含む）を返す。

getStatus

燃料系の状態の参照

【パラメータ】

instanceId	IdType	参照対象の燃料系のID

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
status	FuelSystemStatusType	対象燃料系の状態

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外

【機能】

参照対象の燃料系の状態（残量・瞬間燃費・航続距離を含む）を返す。

notify

燃料系イベントの通知要求

【パラメータ】

instanceId	IdType?	イベント通知対象の燃料系のID
eventFilter	enumSet (FuelSystemEventKindType)?	通知するイベントの種類の集合

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_INVALID_PARAMETER	eventFilterが有効範囲外

【機能】

イベント通知対象の燃料系で発生したeventFilterで指定したイベントの通知を要求する。イベントを通知するためのイベントキューを生成し，その通知ハンドルを notifyHandle に返す。

unnotify

燃料系イベントの通知停止

【パラメータ】

notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正

【機能】

notifyHandleで指定した通知を停止する。指定した通知用のイベントキューは削除される。

getEvent

燃料系イベントの取得

【パラメータ】

notifyHandle	NotifyHandleType	イベントキューを識別するための通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
event	FuelSystemEventType	取得したイベント

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正
E_NO_DATA	イベントキューが空

【機能】

notifyHandleで指定した通知用のイベントキューから，先頭のイベントを取り出す。取り出したイベントは，イベントキューから削除される。



イベントがパラメータ等で渡されてくるプログラミング環境向けの物理APIでは，用意する必要がない。

4.35. Vehicle.Infotainment（略称：Infotainment）

＜TBD＞

4.36. Vehicle.HMI（略称：HMI）

＜TBD＞

4.37. Vehicle.HMI.Display（略称：Display）

4.37.1. 位置付けと機能

Vehicle.HMI.Display (略称: Display) は，車両内に搭載された物理的なディスプレイを表すマルチインスタンスオブジェクトである。

Displayは、液晶ディスプレイ(LCD)，ヘッドアップディスプレイ(HUD)，電子サイドミラー等，様々な種類の表示装置を抽象化する。 DisplayはLockableObjectの機能を持つ。

4.37.1.1. Displayの位置付けと性質

Displayは，車内に向けて情報を表示する物理的な画面である。各Displayは固有のIDを持ち，その種別，視認可能な人物，物理サイズ，解像度などの構成情報を持つ。

Displayはほぼ平面であると仮定する。ディスプレイの形状はBoundingBox(外接矩形)のサイズで表現する。

Displayは一つ以上のViewを含むことができる。アプリケーションはDisplayに直接描画するのではなく，Display上に定義されたViewを通じて描画を行う。



フリップダウンディスプレイのように，視認性が状態によって変化するDisplayも存在する。

4.37.2. 機能クラス

Displayは，以下の機能クラスを持つ。

Movable	Displayの位置を変えることができる(例:可動式モニタ)
Openable	Displayを開閉できる(例:フリップダウンモニタ)

4.37.3. リスククラス

Displayは，以下のリスククラスを持つ。

None

4.37.4. 構成情報

Displayの各インスタンスは，以下の構成情報を持つ。

instanceId	IdType	ディスプレイのID
kind	DisplayKindType	ディスプレイの種別
displayName	String	ディスプレイの名称
visibleTo	enumSet(VisibleToType)	このディスプレイを視認できる人物の集合
boundingBoxWidth	Float	ディスプレイのBoundingBoxの幅(単位:メートル)
boundingBoxHeight	Float	ディスプレイのBoundingBoxの高さ(単位:メートル)

shape	DisplayShapeType	ディスプレイの形状
resolutionX	Uint32	水平方向の解像度(単位:px)
resolutionY	Uint32	垂直方向の解像度(単位:px)
refreshRate	Uint32	リフレッシュレート(単位:Hz)
displayTechnology	DisplayTechnologyType	ディスプレイの表示技術の種別
capabilities	enumSet (DisplayCapabilityType)	このディスプレイが持つ機能クラスの集合

4.37.5. 状態

Displayの各インスタンスは，以下の状態を持つ。

availability	DisplayAvailabilityType	ディスプレイの使用可否状態

4.37.6. サービスコール一覧

Displayに対するサービスコールは以下の通り。

getConfigAll	すべてのディスプレイの構成情報の取得
getStatus	ディスプレイの状態の取得
notify	ディスプレイイベントの通知要求
unnotify	ディスプレイイベントの通知停止
getEvent	ディスプレイイベントの取得

4.37.7. イベント

Displayに対するイベント種別

Displayに対するイベント種別は以下の通り。

AvailabilityChanged	ディスプレイの使用可否状態の変化

4.37.8. データ型の定義

4.37.8.1. ディスプレイの種別

```
type DisplayKindType = enum {
    CenterDisplay,  // センターディスプレイ
    InstrumentPanel,    // インパネ(ドライバーの目の前)
    HUD,    // ヘッドアップディスプレイ
    SideMirror, // サイドミラー(電子ミラー)
    RearMirror, // リアミラー
    RearMonitor,    // 後席モニタ
```

```
    PassengerDisplay,   // 助手席ディスプレイ
    Other   // その他
};
```

4.37.8.2. ディスプレイの形状

```
type DisplayShapeType = enum {
    Rectangle,  // 矩形
    Circle, // 円形
    Other   // その他
};
```

4.37.8.3. ディスプレイの表示技術

```
type DisplayTechnologyType = enum {
    LCD,    // 液晶ディスプレイ
    HUD // ヘッドアップディスプレイ(投影型)
};
```

4.37.8.4. ディスプレイの機能クラス

```
type DisplayCapabilityType = enum {
    Movable,
    Openable
};
```

4.37.8.5. ディスプレイを視認できる人物の種別

```
type VisibleToType = enum {
    Driver, // 運転者
    FrontPassenger, // 助手席乗員
    RearLeft,   // 後部座席左側乗員
    RearRight,  // 後部座席右側乗員
    Outside // 車外の人(今後の課題)
};
```

4.37.8.6. ディスプレイの使用可否状態

```
type DisplayAvailabilityType = enum {
    Available,  // 使用可能
    Unavailable // 使用不可(故障等)
};
```

4.37.8.7. ディスプレイの構成情報

```
type DisplayConfigType = struct {
    instanceId : IdType,
    kind : DisplayKindType,
    displayName : String,
    visibleTo : enumSet(VisibleToType),
    boundingBoxWidth : Float,
    boundingBoxHeight : Float,
    shape : DisplayShapeType,
    resolutionX : Uint32,
    resolutionY : Uint32,
    refreshRate : Uint32,
    displayTechnology : DisplayTechnologyType,
    capabilities : enumSet(DisplayCapabilityType)
};
```

4.37.8.8. ディスプレイの状態

```
type DisplayStatusType = struct {
    availability : DisplayAvailabilityType
};
```

4.37.9. サービスコールの詳細規定

getConfigAll

すべてのディスプレイの構成情報の取得

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
configs	DisplayConfigType[]	車内のすべてのディスプレイの構成情報

【エラーコード】

なし

【機能】

車内のすべてのディスプレイの構成情報を返す。

getStatus

ディスプレイの状態の取得

【パラメータ】

instanceId	IdType	対象ディスプレイのID

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
status	DisplayStatusType	対象ディスプレイの状態

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外

【機能】

指定したディスプレイの使用可否状態を返す。

notify

ディスプレイイベントの通知要求

【パラメータ】

instanceId	IdType?	イベント通知対象のディスプレイのID

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
notifyHandle	NotifyHandleType	通知ハンドル

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外

【機能】

指定したディスプレイで発生したイベントの通知を要求する。instanceIdを省略した場合，すべてのインスタンスのイベントを通知する。

unnotify

ディスプレイイベントの通知停止

【パラメータ】

notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正

【機能】

notifyHandleで指定した通知を停止する。

getEvent

ディスプレイイベントの取得

【パラメータ】

notifyHandle	NotifyHandleType	イベントキューを識別するための通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
event	DisplayEventType	取得したイベント

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正
E_NO_DATA	イベントキューが空

【機能】

notifyHandleで指定した通知用のイベントキューから，先頭のイベントを取り出す。

4.38. Vehicle.HMI.View（略称：View）

4.38.1. 位置付けと機能

Vehicle.HMI.View(略称:View)は，物理ディスプレイ(Vehicle.HMI.Display)上に定義された仮想的な描画領域を表すマルチインスタンスオブジェクトである。

アプリケーションはViewを通じてGraphicsObjectを描画することで，ディスプレイに映像・UI等を表示する。 ViewはLockableObjectの機能を持つ。

4.38.1.1. Viewの位置付けと性質

ViewはDisplayのBoundingBox左上を原点とするピクセル座標系で位置と大きさを定義する。Viewは矩形であると仮定する。

各ViewはひとつのDisplayに属し，複数のアプリケーションが同じViewを利用しようとした場合，優先度に基づく使用権(ロック)の仕組みにより排他制御が行われる。

アプリケーションがViewに描画するには，まずlock()を呼び出して使用権を取得し，その後attach()でGraphicsObjectを指定する。

NOTE 矩形以外の形状のViewへの拡張は今後の課題である。

4.38.1.2. Viewの機能クラスと状態監視

Viewは機能クラスとして FreezeDetection を持つことができる。FreezeDetection 機能クラスを持つ View は、当該 View 上での描画処理が所定の時間内に進行していない状態（フリーズ）を検知する仕組みを備える。

4.38.2. 機能クラス

Viewは，以下の機能クラスを持つ。

FreezeDetection	フリーズ検出が可能

4.38.3. リスククラス

Viewは，以下のリスククラスを持つ。

RiskDistractionHud	ヘッドアップディスプレイと同等のドライバディストラクションのリスク（運転者が見るために，視線の移動がほとんど必要ない。運転中は，運転に直結する情報のみ表示して良い）
RiskDistractionIP	インパネ内のディスプレイと同等のドライバディストラクションのリスク（運転者が見るために，少しの視線移動が必要である。運転中は，運転に関連する情報のみ表示して良い）
RiskDistractionCid	センターディスプレイと同等のドライバディストラクションのリスク（運転者が見るために，大きい視線移動が必要である）

4.38.4. 構成情報

Viewの各インスタンスは，以下の構成情報を持つ。

instanceId	IdType	ViewのID
displayId	IdType	このViewを含むDisplayのID
positionX	Uint32	DisplayのBoundingBox左上からのX座標(単位:px)
positionY	Uint32	DisplayのBoundingBox左上からのY座標(単位:px)
width	Uint32	Viewの幅(単位:px)
height	Uint32	Viewの高さ(単位:px)
visibleTo	enumSet(VisibleToType)	このViewを視認できる人物の集合
riskClasses	enumSet(ViewRiskType)	このViewのリスククラスの集合

4.38.5. 状態

Viewの各インスタンスは，以下の状態を持つ。

status	ViewStatusType	Viewの現在の状態

4.38.6. サービスコール一覧

Viewに対するサービスコールは以下の通り。

getConfigAll	すべてのViewの構成情報の取得
getStatus	Viewの状態の取得
lock	Viewの使用権の確保
unlock	Viewの使用権の破棄
attach	GraphicsObjectツリーのViewへのアタッチ
setBackground	背景テクスチャの指定
notify	Viewイベントの通知要求
unnotify	Viewイベントの通知停止
getEvent	Viewイベントの取得
detach	GraphicsObjectツリーのViewからのデタッチ
setVariableParameter	I/O変数への値の設定
getVariableParameter	I/O変数の現在値の取得

4.38.7. イベント

Viewに対するイベント種別には，状態変化がある。

Viewに対するイベント種別は以下の通り。

StatusChanged	Viewの状態変化

4.38.8. データ型の定義

4.38.8.1. Viewのリスククラス

```
type ViewRiskType = enum {
    RiskDistractionHud,
    RiskDistractionIP,
    RiskDistractionCid
};
```

4.38.8.2. Viewの状態種別

```
type ViewStatusType = enum {
    UNAVAILABLE,    // 警告またはDisplayがオフになった等の理由によりアプリが使用不可
    AVAILABLE,  // 使用可能
```

```
    IN_USE, // アプリが使用中
    FAULT   // 何等かの理由でViewが正常に表示できない
};
```

4.38.8.3. テクスチャのフィットタイプ

```
type FitType = enum {
    FILL,   // Viewのサイズに合うようテクスチャを引き延ばす
    CONTAIN,    // アスペクト比を維持したままテクスチャがViewに収まるよう拡大縮小する
    COVER   // アスペクト比を維持したままテクスチャがViewを埋めるよう拡大縮小する
};
```

4.38.8.4. Viewの構成情報

```
type ViewConfigType = struct {
    instanceId : IdType,
    displayId : IdType,
    positionX : Uint32,
    positionY : Uint32,
    width : Uint32,
    height : Uint32,
    visibleTo : enumSet(VisibleToType),
    riskClasses : enumSet(ViewRiskType)
};
```

4.38.8.5. Viewの状態

```
type ViewStatusDataType = struct {
    status : ViewStatusType
};
```

4.38.8.6. GraphicsObject共通型

```
type Vec2 = struct {
    x : Float32,
    y : Float32
};
```

```
type Vec3 = struct {
    x : Float32,
    y : Float32,
    z : Float32
};
```

```
type Color = struct {
    r : Float32,
    g : Float32,
    b : Float32,
    a : Float32
};
```

```
type UvArea = struct {
    origin : Vec2,
    size : Vec2
};
```

4.38.8.7. I/O変数の値型

```
type IoVariantKind = enum {
    Sint32, // 32ビット符号付き整数
    Float32,    // 32ビット浮動小数点数
    String, // 文字列
    Bool,   // 真偽値
    Color,  // RGBAカラー
    Vec2,   // 2次元ベクトル
    Vec3,   // 3次元ベクトル
    UvArea  // UV座標系矩形領域
};
```

```
type IoVariant = variant(IoVariantKind) {
    Sint32 {
        value : Int32
    },
    Float32 {
        value : Float32
    },
    String {
        value : String
    },
    Bool {
        value : Bool
    },
    Color {
        value : Color
    },
    Vec2 {
        value : Vec2
    },
    Vec3 {
        value : Vec3
```

```
    },
    UvArea {
        value : UvArea
    }
};
```

4.38.8.8. Bindable型（直接値またはI/O変数バインド）

```
type IoVariableId = Uint32; // I/O変数のID
```

```
type BindableKind = enum {
    Value,  // 直接値
    Bind    // I/O変数へのバインド参照
};
```

```
type BindableBool = variant(BindableKind) {
    Value {
        value : Bool
    },
    Bind {
        bindId : IoVariableId
    }
};
```

```
type BindableFloat32 = variant(BindableKind) {
    Value {
        value : Float32
    },
    Bind {
        bindId : IoVariableId
    }
};
```

```
type BindableString = variant(BindableKind) {
    Value {
        value : String
    },
    Bind {
        bindId : IoVariableId
    }
};
```

```
type BindableColor = variant(BindableKind) {
```

```
    Value {
        value : Color
    },
    Bind {
        bindId : IoVariableId
    }
};
```

```
type BindableVec2 = variant(BindableKind) {
    Value {
        value : Vec2
    },
    Bind {
        bindId : IoVariableId
    }
};
```

```
type BindableVec3 = variant(BindableKind) {
    Value {
        value : Vec3
    },
    Bind {
        bindId : IoVariableId
    }
};
```

```
type BindableUvArea = variant(BindableKind) {
    Value {
        value : UvArea
    },
    Bind {
        bindId : IoVariableId
    }
};
```

4.38.8.9. GraphicsObjectパラメータ型

```
type GraphicsObjectParams = struct {
    visible : BindableBool?,    // デフォルト: true
    anchor_point : BindableVec2?,   // 回転・拡縮の中心。デフォルト: 描画サイズの中心
    position : BindableVec3?,   // anchor_pointの親座標系での配置位置。デフォルト: {x:
0, y:0, z:0}
    area : Vec2?,   // BoundingBoxサイズのオーバーライド。デフォルト: 描画サイズから自
動算出
    rotation : BindableFloat32?,    // 度数法。デフォルト: 0
    scale : BindableVec2?   // デフォルト: {x:1.0 y:1.0}
```

```
};
```

```
type TextureParams = struct {
    base : GraphicsObjectParams?,
    texture_id : BindableSint32?,   // 未設定時は何も描画しない
    color : BindableColor?, // 乗算ブレンド。デフォルト: {r:1,g:1,b:1,a:1}
    uv_area : BindableUvArea?   // デフォルト: テクスチャ全体
};
```

```
type CircleParams = struct {
    base : GraphicsObjectParams?,
    color : BindableColor?, // デフォルト: {r:1,g:1,b:1,a:1}
    radius : BindableFloat32?   // デフォルト: 50
};
```

```
type RectangleParams = struct {
    base : GraphicsObjectParams?,
    color : BindableColor?, // デフォルト: {r:1,g:1,b:1,a:1}
    width : BindableFloat32?,   // デフォルト: 100
    height : BindableFloat32?   // デフォルト: 100
};
```

```
type TriangleParams = struct {
    base : GraphicsObjectParams?,
    color : BindableColor?, // デフォルト: {r:1,g:1,b:1,a:1}
    side_length : BindableFloat32?  // 正三角形の1辺。デフォルト: 100
};
```

```
type LineParams = struct {
    base : GraphicsObjectParams?,
    color : BindableColor?, // デフォルト: {r:1,g:1,b:1,a:1}
    length : BindableFloat32?,  // 線の方向はrotationで指定。デフォルト: 100
    width : BindableFloat32?    // デフォルト: 1
};
```

```
type TextParams = struct {
    base : GraphicsObjectParams?,
    text_string : BindableString?,  // デフォルト: ""
    color : BindableColor?, // デフォルト: {r:1,g:1,b:1,a:1}
    font_size : BindableFloat32?,   // ピクセル単位。デフォルト: 16
    font_id : BindableSint32?   // 未設定時はシステムデフォルト
};
```

```
type ArcParams = struct {
    base : GraphicsObjectParams?,
    radius : BindableFloat32?,  // デフォルト: 50
    start_angle : BindableFloat32?, // 0度 = 12時方向、時計回り。デフォルト: 0
    end_angle : BindableFloat32?,   // start_angle < end_angle。デフォルト: 360
    line_width : BindableFloat32?,  // デフォルト: 1
    color : BindableColor?  // デフォルト: {r:1g:1b:1a:1}
};
```

```
type VideoPlayState = enum {
    Playing,    // 再生中
    Stopped // 先頭に戻る
};
```

```
type VideoParams = struct {
    base : GraphicsObjectParams?,
    video_id : BindableSint32?, // 未設定時は何も描画しない
    play_state : VideoPlayState?    // デフォルト: Stopped
};
```

```
type Alignment = enum {
    UpperLeft,  // 左上
    UpperCenter,    // 上中央
    UpperRight, // 右上
    MiddleLeft, // 左中央
    MiddleCenter,   // 中央
    MiddleRight,    // 右中央
    LowerLeft,  // 左下
    LowerCenter,    // 下中央
    LowerRight  // 右下
};
```

```
type LayoutParams = struct {
    base : GraphicsObjectParams?,
    alignment : Alignment?, // デフォルト: UpperLeft
    padding : BindableVec2? // x=左右、y=上下（片側分）。デフォルト: {x:0 y:0}
};
```

```
type DirectionalLayoutParams = struct {
    base : GraphicsObjectParams?,
    alignment : Alignment?, // デフォルト: UpperLeft
    padding : BindableVec2?,    // x=左右、y=上下（片側分）。デフォルト: {x:0, y:0}
    spacing : BindableFloat32?  // 子要素間の間隔。デフォルト: 0
```

```
};
```

4.38.8.10. I/O変数定義

```
type IoVariableDef = struct {
    id : IoVariableId,  // I/O変数のID
    type : IoVariantKind,   // I/O変数の値型
    default_value : IoVariant?  // 初期値。省略時はゼロ値
};
```

4.38.8.11. GraphicsObjectツリー構造型

```
type GraphicsObjectKind = enum {
    Texture,    // テクスチャ画像
    Arc,    // 円弧
    Circle, // 円
    Rectangle,  // 矩形
    Triangle,   // 正三角形
    Line,   // 直線
    Text,   // テキスト
    Video,  // ビデオ
    Layout, // レイアウト
    VerticalLayout, // 縦方向レイアウト
    HorizontalLayout    // 横方向レイアウト
};
```

```
type GraphicsObjectNode = variant(GraphicsObjectKind) {
    Texture {
        params : TextureParams?,
        children : GraphicsObjectNode[]?
    },
    Arc {
        params : ArcParams?,
        children : GraphicsObjectNode[]?
    },
    Circle {
        params : CircleParams?,
        children : GraphicsObjectNode[]?
    },
    Rectangle {
        params : RectangleParams?,
        children : GraphicsObjectNode[]?
    },
    Triangle {
        params : TriangleParams?,
        children : GraphicsObjectNode[]?
    },
```

```
    Line {
        params : LineParams?,
        children : GraphicsObjectNode[]?
    },
    Text {
        params : TextParams?,
        children : GraphicsObjectNode[]?
    },
    Video {
        params : VideoParams?,
        children : GraphicsObjectNode[]?
    },
    Layout {
        params : LayoutParams?,
        children : GraphicsObjectNode[]?
    },
    VerticalLayout {
        params : DirectionalLayoutParams?,
        children : GraphicsObjectNode[]?
    },
    HorizontalLayout {
        params : DirectionalLayoutParams?,
        children : GraphicsObjectNode[]?
    }
};
```

```
type GraphicsObjectTree = struct {
    io_variables : IoVariableDef[],
    root : GraphicsObjectNode
};
```

4.38.8.12. GraphicsObjectエラーコードと結果型

```
type GraphicsObjectErrorCode = enum {
    E_INVALID_ID,   // IDが不正
    E_INVALID_HANDLE,   // ハンドルが不正
    E_INVALID_PARAMETER,    // パラメータが不正
    E_OBJECT_STATUS // attach()時に同一view_idに既存ツリーが存在する
};
```

```
type TreeHandle = IdType;   // attach()が返すツリーハンドル
```

```
type GraphicsObjectResultKind = enum {
    Ok, // 正常終了
    Err // エラー終了
```

```
};
```

```
type AttachResult = variant(GraphicsObjectResultKind) {
    Ok {
        tree_handle : TreeHandle
    },
    Err {
        GraphicsObjectErrorCode : GraphicsObjectErrorCode
    }
};
```

```
type VoidResult = variant(GraphicsObjectResultKind) {
    Ok,
    Err {
        GraphicsObjectErrorCode : GraphicsObjectErrorCode
    }
};
```

```
type GetParameterResult = variant(GraphicsObjectResultKind) {
    Ok {
        value : IoVariant
    },
    Err {
        GraphicsObjectErrorCode : GraphicsObjectErrorCode
    }
};
```

4.38.9. サービスコールの詳細規定

getConfigAll

すべてのViewの構成情報の取得

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
configs	ViewConfigType[]	車内のすべてのViewの構成情報

【エラーコード】

なし

【機能】

車内のすべてのViewの構成情報を返す。

getStatus

Viewの状態の取得

【パラメータ】

instanceId	IdType	対象ViewのID

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
status	ViewStatusType	Viewの現在の状態

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外

【機能】

指定したViewの現在の状態を返す。

lock

Viewの使用権の確保

【パラメータ】

instanceId	IdType	対象ViewのID
priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_OBJECT_LOCKED	自分より優先度の高いアプリがすでに使用権を確保している
E_OBJECT_STATUS	ViewがUNAVAILABLEまたはFAULT状態

【機能】

Viewの使用権の確保を試みる。

自分より優先度の高いアプリがすでに確保していたらlockに失敗する。

逆に，あるアプリが一度lockしたViewであっても，別アプリがより高い優先度でlockしたら使用権を奪われる。

例:ゲーム(優先度:3)再生中に，車両接近通報装置(優先度:4)がlockして，ゲーム画面が車両接近通報装置に切り替わる。

unlock

Viewの使用権の破棄

【パラメータ】

instanceId	IdType	対象ViewのID

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_OBJECT_STATUS	自アプリケーションが対象Viewをロックしていないため Unlockできない

【機能】

Viewの使用権を破棄する。

attach

GraphicsObjectツリーのViewへのアタッチ

【パラメータ】

view_id	IdType	対象ViewのID
tree	GraphicsObjectTree	アタッチするGraphicsObjectツリー

【リターンパラメータ】

result	AttachResult	成功時はtree_handle、失敗時はエラーコード

【エラーコード】

E_OBJECT_STATUS	同一view_idに既存ツリーが存在する

【機能】

GraphicsObjectツリーをViewにアタッチする。返却されるtreeHandleを使ってI/O変数の操作を行う。同一view_idに既にツリーがアタッチされている場合はE_OBJECT_STATUSを返す。

setBackground

背景テクスチャの指定

【パラメータ】

instanceId	IdType	対象ViewのID
textureId	IdType	テクスチャID
fitType	FitType	フィットタイプ

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	instanceIdまたはtextureIdが有効範囲外
E_OBJECT_STATUS	対象ViewがLock済みでない

【機能】

Viewにテクスチャを fitTypeで指定した方法で背景として設定する。

notify

Viewイベントの通知要求

【パラメータ】

instanceId	IdType?	イベント通知対象のViewのID

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
notifyHandle	NotifyHandleType	通知ハンドル

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外

【機能】

指定したViewで発生したイベントの通知を要求する。instanceIdを省略した場合，すべてのインスタンスのイベントを通知する。

unnotify

Viewイベントの通知停止

【パラメータ】

notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正

【機能】

notifyHandleで指定した通知を停止する。

getEvent

Viewイベントの取得

【パラメータ】

notifyHandle	NotifyHandleType	イベントキューを識別するための通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
event	ViewEventType	取得したイベント

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正
E_NO_DATA	イベントキューが空

【機能】

notifyHandleで指定した通知用のイベントキューから，先頭のイベントを取り出す。

detach

GraphicsObjectツリーのViewからのデタッチ

【パラメータ】

tree_handle	TreeHandle	対象ツリーのハンドル

【リターンパラメータ】

result	VoidResult	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	tree_handleが不正

【機能】

Viewから描画ツリーを取り除く。

setVariableParameter

I/O変数への値の設定

【パラメータ】

tree_handle	TreeHandle	対象ツリーのハンドル
io_variable_id	IoVariableId	設定対象のI/O変数ID
value	IoVariant	設定する値。IoVariableDef.typeと一致させること

【リターンパラメータ】

result	VoidResult	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	tree_handleが不正
E_INVALID_ID	io_variable_idが存在しない
E_INVALID_PARAMETER	valueのkindがI/O変数の型と不一致

【機能】

I/O変数に値を設定する。バインドされているすべてのparamに即時反映される。

getVariableParameter

I/O変数の現在値の取得

【パラメータ】

tree_handle	TreeHandle	対象ツリーのハンドル
io_variable_id	IoVariableId	取得対象のI/O変数ID

【リターンパラメータ】

result	GetParameterResult	成功時はIoVariant値、失敗時はエラーコード

【エラーコード】

E_INVALID_HANDLE	tree_handleが不正

E_INVALID_ID	io_variable_idが存在しない

【機能】

I/O変数の現在値を取得する。

4.39. Vehicle.HMI.Input.Keyboard（略称：Keyboard）

4.39.1. 位置付けと機能

Vehicle.HMI.Input.Keyboard（略称: Keyboard）は，車載キーボード(物理ボタン含む)からの入力を受け取る機能を提供するシングルインスタンスオブジェクトである。

4.39.1.1. Keyboardの位置付けと性質

Keyboardは，キー押下・キーリリースイベントの通知，入力文字の取得などの機能を提供する。物理キーボードおよびステアリングスイッチなどのハードウェアボタンからの入力を統一的に扱う。

4.39.2. 機能クラス

4.39.3. リスククラス

4.39.4. 構成情報

4.39.5. 状態

4.39.6. サービスコール一覧

Keyboardに対するサービスコールは以下の通り。

startKeyboardInput	入力の開始(キーボードを表示)
completeKeyboardInput	入力の完了(キーボードを非表示)
onTextInput	文字入力を検知

4.39.7. イベント

4.39.8. データ型の定義

4.39.9. サービスコールの詳細規定

startKeyboardInput

入力の開始(キーボードを表示)

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

なし

【機能】



ソフトウェアキーボードを想定する。

completeKeyboardInput

入力の完了(キーボードを非表示)

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

なし

【機能】

onTextInput

文字入力を検知

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
inputChar	String	入力された文字

【エラーコード】

なし

【機能】

4.40. Vehicle.HMI.Input.Mouse（略称：Mouse）

4.40.1. 位置付けと機能

Vehicle.HMI.Input.Mouse（略称: Mouse）は，ポインティングデバイス(マウス，トラックパッド等)からの入力を受け取る機能を提供するシングルインスタンスオブジェクトである。

4.40.1.1. Mouseの位置付けと性質

Mouseは，ポインタ座標の取得，ボタンクリックイベントの通知，スクロール操作の検出などの機能を提供する。



本オブジェクトは大部分が仕様未定義である。今後、詳細な定義を行う。

4.40.2. 機能クラス

4.40.3. リスククラス

4.40.4. 構成情報

4.40.5. 状態

4.40.6. サービスコール一覧

Mouseに対するサービスコールは以下の通り。

getPosition	ポインタの位置を取得
isDown	ボタンが押されているかどうかを判定
isPressed	ボタンがそのフレームで押されたかどうかを判定
isReleased	ボタンがそのフレームで放されたかどうかを判定
getScroll	そのフレームのスクロール量を取得

4.40.7. イベント

4.40.8. データ型の定義

4.40.8.1. 型定義

```
type MouseButtonKind = enum {
    Left,   // 左ボタン
    Right,  // 右ボタン
    Middle  // 中ボタン(ホイールクリック)
};
```

4.40.9. サービスコールの詳細規定

getPosition

ポインタの位置を取得

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
position	Vector2	ポインタの位置(2次元ベクトル)

【エラーコード】

なし

【機能】

isDown

ボタンが押されているかどうかを判定

【パラメータ】

buttonKind	MouseButtonKind	ボタンの種類

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
result	Bool	ボタンが押されているかどうか

【エラーコード】

なし

【機能】

isPressed

ボタンがそのフレームで押されたかどうかを判定

【パラメータ】

buttonKind	MouseButtonKind	ボタンの種類

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
result	Bool	ボタンがそのフレームで押されたかどうか

【エラーコード】

なし

【機能】

isReleased

ボタンがそのフレームで放されたかどうかを判定

【パラメータ】

buttonKind	MouseButtonKind	ボタンの種類

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
result	Bool	ボタンがそのフレームで放されたかどうか

【エラーコード】

なし

【機能】

getScroll

そのフレームのスクロール量を取得

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
scrollAmount	Vector2	スクロール量(2次元ベクトル。x: 水平方向, y: 垂直方向)

【エラーコード】

なし

【機能】

4.41. Vehicle.HMI.Input.Touch（略称：Touch）

4.41.1. 位置付けと機能

Vehicle.HMI.Input.Touch (略称: Touch) は，タッチパネルからのタッチ入力を受け取る機能を提供するシングルインスタンスオブジェクトである。

4.41.1.1. Touchの位置付けと性質

Touchは，タッチ開始・移動・終了イベントの通知，マルチタッチのサポート，タッチ座標の取得などの機能を提供する。タッチスクリーンを持つDisplayに対応した操作インタフェースを提供する。

4.41.2. 機能クラス

4.41.3. リスククラス

4.41.4. 構成情報

4.41.5. 状態

4.41.6. サービスコール一覧

Touchに対するサービスコールは以下の通り。

getTouchNum	タッチ点数を取得
getTouchId	タッチ点のIDを取得
getPosition	タッチ点の位置を取得
isDown	タッチ点が押されているかどうかを判定
isPressed	タッチ点がそのフレームで押されたかどうかを判定
isReleased	タッチ点がそのフレームで放されたかどうかを判定
isTapped	タッチ点がそのフレームでタップされたかどうかを判定
isDoubleTapped	タッチ点がそのフレームでダブルタップされたかどうかを判定
isLongTapped	タッチ点がそのフレームでロングタップされたかどうかを判定
isFlicked	タッチ点がそのフレームでフリックされたかどうかを判定
getFlickPosition	フリックの始点位置を取得
getFlickDirection	フリックの方向を取得
invalidate	入力の無効化

4.41.7. イベント

4.41.8. データ型の定義

4.41.9. サービスコールの詳細規定

getTouchNum

タッチ点数を取得

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
touchNum	Uint32	タッチ点数

【エラーコード】

なし

【機能】

getTouchId

タッチ点のIDを取得

【パラメータ】

index	Uint32	タッチ点の認識順番

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
touchId	Uint32	タッチ点のID

【エラーコード】

なし

【機能】

getPosition

タッチ点の位置を取得

【パラメータ】

touchId	Uint32	タッチ点のID

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
position	Vector2	タッチ点の位置(2次元ベクトル)

【エラーコード】

なし

【機能】

isDown

タッチ点が押されているかどうかを判定

【パラメータ】

touchId	Uint32	タッチ点のID

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
result	Bool	タッチ点が押されているかどうか

【エラーコード】

なし

【機能】

isPressed

タッチ点がそのフレームで押されたかどうかを判定

【パラメータ】

touchId	Uint32	タッチ点のID

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
result	Bool	タッチ点がそのフレームで押されたかどうか

【エラーコード】

なし

【機能】

isReleased

タッチ点がそのフレームで放されたかどうかを判定

【パラメータ】

touchId	Uint32	タッチ点のID

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
result	Bool	タッチ点がそのフレームで放されたかどうか

【エラーコード】

なし

【機能】

isTapped

タッチ点がそのフレームでタップされたかどうかを判定

【パラメータ】

touchId	Uint32	タッチ点のID

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
result	Bool	タッチ点がそのフレームでタップされたかどうか

【エラーコード】

なし

【機能】

isDoubleTapped

タッチ点がそのフレームでダブルタップされたかどうかを判定

【パラメータ】

touchId	Uint32	タッチ点のID

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
result	Bool	タッチ点がそのフレームでダブルタップされたかどうか

【エラーコード】

なし

【機能】

isLongTapped

タッチ点がそのフレームでロングタップされたかどうかを判定

【パラメータ】

touchId	Uint32	タッチ点のID

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
result	Bool	タッチ点がそのフレームでロングタップされたかどうか

【エラーコード】

なし

【機能】

isFlicked

タッチ点がそのフレームでフリックされたかどうかを判定

【パラメータ】

touchId	Uint32	タッチ点のID

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
result	Bool	タッチ点がそのフレームでフリックされたかどうか

【エラーコード】

なし

【機能】

getFlickPosition

フリックの始点位置を取得

【パラメータ】

touchId	Uint32	タッチ点のID

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
position	Vector2	フリックの始点位置(2次元ベクトル)

【エラーコード】

なし

【機能】

getFlickDirection

フリックの方向を取得

【パラメータ】

touchId	Uint32	タッチ点のID

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
direction	Vector2	フリックの方向(2次元ベクトル)

【エラーコード】

なし

【機能】

invalidate

入力の無効化

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

なし

【機能】



この関数実行後から次のフレーム開始までの間、

タッチの入力値の取得を無効化する

4.42. Vehicle.HMI.Input.Joystick（略称：Joystick）

4.42.1. 位置付けと機能

Vehicle.HMI.Input.Joystick（略称: Joystick）は，ジョイスティック(方向スイッチ，ボタン)からの入力を受け取る機能を提供するシングルインスタンスオブジェクトである。

4.42.1.1. Joystickの位置付けと性質

Joystickは，方向入力(上下左右)，決定・戻るボタン，アナログ軸の値取得などの機能を提供する。センターコンソールのコントローラやステアリングスイッチなどの操作デバイスを統一的に扱う。

4.42.2. 機能クラス

4.42.3. リスククラス

4.42.4. 構成情報

4.42.5. 状態

4.42.6. サービスコール一覧

Joystickに対するサービスコールは以下の通り。

isDown	ボタンが押されているかどうかを判定
isPressed	ボタンがそのフレームで押されたかどうかを判定
isReleased	ボタンがそのフレームで放されたかどうかを判定
getAnalog	アナログ軸の入力値を取得
invalidate	入力の無効化

4.42.7. イベント

4.42.8. データ型の定義

4.42.8.1. 型定義

```
type JoystickKeyKind = enum {
    Up, // 上方向
    Down,   // 下方向
    Left,   // 左方向
    Right,  // 右方向
    Decide, // 決定
    Back    // 戻る
};
```

```
type JoystickAxisKind = enum {
    X,  // 左右方向の傾き量（左: -1.0 〜 右: +1.0）
    Y   // 上下方向の傾き量（下: -1.0 〜 上: +1.0）
};
```

4.42.9. サービスコールの詳細規定

isDown

ボタンが押されているかどうかを判定

【パラメータ】

keyKind	JoystickKeyKind	キーの種類

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
result	Bool	ボタンが押されているかどうか

【エラーコード】

なし

【機能】

isPressed

ボタンがそのフレームで押されたかどうかを判定

【パラメータ】

keyKind	JoystickKeyKind	キーの種類

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
result	Bool	ボタンがそのフレームで押されたかどうか

【エラーコード】

なし

【機能】

isReleased

ボタンがそのフレームで放されたかどうかを判定

【パラメータ】

keyKind	JoystickKeyKind	キーの種類

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
result	Bool	ボタンがそのフレームで放されたかどうか

【エラーコード】

なし

【機能】

getAnalog

アナログ軸の入力値を取得

【パラメータ】

axisKind	JoystickAxisKind	軸の種類

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
analogValue	Float32	軸の入力値（-1.0 〜 +1.0）

【エラーコード】

なし

【機能】

invalidate

入力の無効化

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

なし

【機能】



この関数実行後から次のフレーム開始までの間、

ジョイスティックの入力値の取得を無効化する

4.43. Vehicle.HMI.Sound.Common.SoundEffect（略称：SoundEffect）

4.43.1. 位置付けと機能

Vehicle.HMI.Sound.Common.SoundEffect (略称: SoundEffect) は，システム共通の効果音の再生管理機能を提供するシングルインスタンスオブジェクトである。

4.43.1.1. SoundEffect(Common)の位置付けと性質

Vehicle.HMI.Sound.Common.SoundEffectは，操作フィードバック音など，複数のアプリケーションで共有される共通効果音の再生・停止機能を提供する。共通効果音はシステム定義のサウンドセットから選択して使用する。

4.43.2. 機能クラス

4.43.3. リスククラス

4.43.4. 構成情報

4.43.5. 状態

4.43.6. サービスコール一覧

SoundEffectに対するサービスコールは以下の通り。

play	共通サウンドの再生
overwrite	共通サウンドのデータを上書き
reset	共通サウンドのデータをリセット

4.43.7. イベント

4.43.8. データ型の定義

4.43.8.1. 型定義

```
type CommonSoundKind = enum {
    OK, // 操作確定音
    Cancel, // キャンセル音
    NG, // 操作不可音
    CursorMove, // カーソル移動音
    Alert,  // 警告音
    Open,   // 画面オープン音
    Close   // 画面クローズ音
};
```

4.43.9. サービスコールの詳細規定

play

共通サウンドの再生

【パラメータ】

soundKind	CommonSoundKind	共通サウンドの種類

volume	Uint8	音量(0～255)
priority	Uint8 ([0,100])	優先順位(0～100)

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

なし

【機能】



Sound.App.SoundEffect.playの「ビークルOSカーネルに実装する機能等」の欄を参照

overwrite

共通サウンドのデータを上書き

【パラメータ】

soundKind	CommonSoundKind	共通サウンドの種類
soundId	Uint32	置き換え先の音源ID

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

なし

【機能】



デフォルトで入っていない音源にはidを割り振り、idを指定して再生や置き換えなどを行う。

reset

共通サウンドのデータをリセット

【パラメータ】

soundKind	CommonSoundKind	共通サウンドの種類

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

なし

【機能】

4.44. Vehicle.HMI.Sound.App.SoundEffect（略称：SoundEffect）

4.44.1. 位置付けと機能

Vehicle.HMI.Sound.App.SoundEffect（略称: SoundEffect）は，アプリケーション固有の効果音の再生管理機能を提供するシングルインスタンスオブジェクトである。

4.44.1.1. SoundEffect(App)の位置付けと性質

Vehicle.HMI.Sound.App.SoundEffectは，アプリケーションが独自に定義した効果音の登録・再生・停止機能を提供する。アプリケーションはサウンドリソースを登録し，任意のタイミングで再生できる。

4.44.2. 機能クラス

4.44.3. リスククラス

4.44.4. 構成情報

4.44.5. 状態

4.44.6. サービスコール一覧

SoundEffectに対するサービスコールは以下の通り。

getStatus	SoundEffectハンドルのステータスを取得
getPlaying	再生中のSoundEffectを取得
getScheduled	スケジュールされたSoundEffectを取得
play	SoundEffectを再生
playLoop	SoundEffectをループ再生
schedulePlay	SoundEffectの再生をスケジュールする
pause	SoundEffectの中断
unPause	SoundEffectの再開
stop	SoundEffectの停止
cancelSchedule	スケジュールをキャンセル

4.44.7. イベント

4.44.8. データ型の定義

4.44.9. サービスコールの詳細規定

getStatus

SoundEffectハンドルのステータスを取得

【パラメータ】

soundEffectHandle	Uint32	操作対象のSoundEffectハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
status	enum([{'enumerator': 'PLAYING', 'description': '再生中'}, {'enumerator': 'WAITING', 'description': '再生待機中'}, {'enumerator': 'STOPPED', 'description': '再生終了'}, {'enumerator': 'PAUSED', 'description': 'ポーズ中'}])	SoundEffectハンドルのステータス

【エラーコード】

なし

【機能】



SoundEffectの再生中はPLAYING、

SchedulePlayによる再生待機中はWAITING、

再生が完了したもの、あるいはStopやCancelScheduleによって二度と再開されなくなったものはSTOPPED、

ポーズ中のものはPAUSEDとなる。

getPlaying

再生中のSoundEffectを取得

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
handles	Uint32[3]	SoundEffectハンドルの配列(長さは最大3)

【エラーコード】

なし

【機能】

getScheduled

スケジュールされたSoundEffectを取得

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
handles	Uint32[]	SoundEffectハンドルの配列

【エラーコード】

なし

【機能】

play

SoundEffectを再生

【パラメータ】

soundId	Uint32	音源を表すid
volume	Uint8	音量
priority	Uint8 ([0,100])	優先順位(0〜100)

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
soundEffectHandle	Uint32	SoundEffectハンドル

【エラーコード】

なし

【機能】



Sound.CommonとSoundEffectと合わせた同時再生数の最大値は3つ。

4つ目を再生すると残り3つのうち最も優先順位が低いものが停止する。

playLoop

SoundEffectをループ再生

【パラメータ】

soundId	Uint32	音源を表すid
volume	Uint8	音量
priority	Uint8 ([0,100])	優先順位(0〜100)
loopCount	Uint16 ([1,256])	ループの回数(最大256回)
loopInterval	Uint32	ループの間隔(ミリ秒)

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
soundEffectHandle	Uint32	SoundEffectハンドル

【エラーコード】

なし

【機能】

schedulePlay

SoundEffectの再生をスケジュールする

【パラメータ】

soundId	Uint32	音源を表すid
volume	Uint8	音量
priority	Uint8 ([0,100])	優先順位(0〜100)
delay	Uint32	遅延させる時間(ミリ秒)

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
soundEffectHandle	Uint32	SoundEffectハンドル

【エラーコード】

なし

【機能】



引数で指定した時間が経過してから再生を行う。

pause

SoundEffectの中断

【パラメータ】

soundEffectHandle	Uint32	SoundEffectハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

なし

【機能】



ハンドルで指定したSoundEffectをポーズする。

下記のUnPauseメソッドをポーズしたSoundEffectに対して呼ぶことで再生が再開される。

unPause

SoundEffectの再開

【パラメータ】

soundEffectHandle	Uint32	SoundEffectハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

なし

【機能】

stop

SoundEffectの停止

【パラメータ】

soundEffectHandle	Uint32	SoundEffectハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

なし

【機能】



ハンドルで指定したSoundEffectを停止する。

Pauseとは異なり、再開することはない。

cancelSchedule

スケジュールをキャンセル

【パラメータ】

soundEffectHandle	Uint32	SoundEffectハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

なし

【機能】



再生が予定されているSoundEffectに対してキャンセル処理を行う。

4.45. Vehicle.HMI.Sound.App.Music（略称：Music）

4.45.1. 位置付けと機能

Vehicle.HMI.Sound.App.Music (略称: Music) は，アプリケーションによる音楽・BGMの再生管理機能を提供するシングルインスタンスオブジェクトである。

4.45.1.1. Musicの位置付けと性質

Musicは，音楽ファイルの再生，一時停止，停止，音量調整などの再生制御機能を提供する。複数のアプリケーションが同時に音楽を再生しようとした場合，優先度に基づいて制御される。

4.45.2. 機能クラス

4.45.3. リスククラス

Musicは，以下のリスククラスを持つ。

T.B.D	T.B.D

4.45.4. 構成情報

4.45.5. 状態

4.45.6. サービスコール一覧

Musicに対するサービスコールは以下の通り。

play	Musicの再生
pause	Musicの中断
unPause	Musicの再開
stop	Musicの停止
fadeIn	Musicのフェードイン
fadeOut	Musicのフェードアウト

4.45.7. イベント

4.45.8. データ型の定義

4.45.9. サービスコールの詳細規定

play

Musicの再生

【パラメータ】

soundId	Uint32	音源を表すid
volume	Uint8	音量

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

なし

【機能】



同時に再生できるMusicは1つのみであり、

2つ目を再生すると現在再生中のものが停止する。

常時ループ再生され、アプリの終了または中断で自動的に停止する。

pause

Musicの中断

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

なし

【機能】

unPause

Musicの再開

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

なし

【機能】

stop

Musicの停止

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

なし

【機能】

fadeIn

Musicのフェードイン

【パラメータ】

fadeInDuration	Uint32	フェードインにかける時間(ミリ秒)

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

なし

【機能】

fadeOut

Musicのフェードアウト

【パラメータ】

fadeOutDuration	Uint32	フェードアウトにかける時間(ミリ秒)

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

なし

【機能】

4.46. Vehicle.HMI.Sound.App.Voice（略称：Voice）

4.46.1. 位置付けと機能

Vehicle.HMI.Sound.App.Voice （略称: Voice） は，アプリケーションによる音声(テキスト読み上げ，ナレーション等)の再生管理機能を提供するシングルインスタンスオブジェクトである。

4.46.1.1. Voice(App)の位置付けと性質

Vehicle.HMI.Sound.App.Voiceは，音声ファイルの再生，テキスト音声合成(TTS)による読み上げ，再生状態の監視などの機能を提供する。ナビゲーション案内やアシスタント音声など，アプリが必要とする音声出力を管理する。

4.46.2. 機能クラス

4.46.3. リスククラス

4.46.4. 構成情報

4.46.5. 状態

4.46.6. サービスコール一覧

Voiceに対するサービスコールは以下の通り。

getStatus	voiceハンドルの状態を取得
getPlaying	再生中のボイスを取得
getWaiting	待機中のボイスを取得
play	ボイスの再生
playText	テキスト読み上げによるボイスの再生
playBlank	無音の再生
interrupt	ボイスの割り込み再生
interruptText	テキスト読み上げによるボイスの割り込み再生
stop	ボイスの停止
pause	ボイスの中断
unPause	ボイスの再開
skip	再生中のボイスをスキップ

4.46.7. イベント

4.46.8. データ型の定義

4.46.8.1. 型定義

```
type VoiceInterruptBehavior = enum {
    PAUSE,  // 途中から再開
    STOP,   // 再吹鳴なし
    REPLAY  // 頭から再度再生
};
```

4.46.9. サービスコールの詳細規定

getStatus

voiceハンドルの状態を取得

【パラメータ】

voiceHandle	Uint32	voiceハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
status	enum([{'enumerator': 'PLAYING', 'description': '再生中'}, {'enumerator': 'WAITING', 'description': '再生待機中'}, {'enumerator': 'STOPPED', 'description': '再生終了'}, {'enumerator': 'PAUSED', 'description': 'ポーズ中'}])	状態

【エラーコード】

なし

【機能】



Voiceの再生中はPLAYING、

バッファに積まれている間や、中断時の挙動をREPLAYにしたことによる再生待ちはWAITING、

Stopメソッドや中断時の挙動をSTOPに設定したことによる再生終了はSTOPPED、

Pauseメソッドや中断時の挙動をPAUSEに設定したことによるポーズはPAUSEDとなる。

getPlaying

再生中のボイスを取得

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
voiceHandle	Uint32	voiceハンドル

【エラーコード】

なし

【機能】

getWaiting

待機中のボイスを取得

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
voiceHandles	Uint32[]	voiceハンドルの配列

【エラーコード】

なし

【機能】

play

ボイスの再生

【パラメータ】

soundId	Uint32	音源を表すid
volume	Uint8	音量
interruptBehavior	VoiceInterruptBehavior	中断された時の挙動

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
voiceHandle	Uint32	voiceハンドル

【エラーコード】

なし

【機能】



同時に再生できるボイスは1つのみである。

内部の振る舞いはFIFOバッファであり、

2つ目以降はバッファの末尾に積まれる。

下記のInterruptメソッドや、警告サウンドによってボイスが中断された時の挙動を引数で設定し、

PAUSEの場合は割り込み中はボイスをポーズして完了時に再開、

STOPの場合は割り込みが起きた瞬間にボイスを終了(再吹鳴なし)、

REPLAYの場合は割り込みが終わると再び最初から再生を行う。

playText

テキスト読み上げによるボイスの再生

【パラメータ】

text	String	読み上げるテキスト
volume	Uint8	音量
interruptBehavior	VoiceInterruptBehavior	中断された時の挙動

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
voiceHandle	Uint32	voiceハンドル

【エラーコード】

なし

【機能】



音源を再生するのとは異なり、引数にテキストをとって読み上げるAPIである。

playBlank

無音の再生

【パラメータ】

duration	Uint32	再生時間(ミリ秒)

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
voiceHandle	Uint32	voiceハンドル

【エラーコード】

なし

【機能】



複数のボイスを間をあけて再生したい場合に用いる。

interrupt

ボイスの割り込み再生

【パラメータ】

soundId	Uint32	音源を表すid
volume	Uint8	音量
interruptBehavior	VoiceInterruptBehavior	中断された時の挙動

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
voiceHandle	Uint32	voiceハンドル

【エラーコード】

なし

【機能】



現在再生中のボイスを停止し、割り込んで再生を行う。

interruptText

テキスト読み上げによるボイスの割り込み再生

【パラメータ】

text	String	読み上げるテキスト
volume	Uint8	音量
interruptBehavior	VoiceInterruptBehavior	中断された時の挙動

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
voiceHandle	Uint32	voiceハンドル

【エラーコード】

なし

【機能】



interruptのテキスト読み上げ版。

stop

ボイスの停止

【パラメータ】

voiceHandle	Uint32	voiceハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

なし

【機能】



引数にとったボイスが再生中ならば下記のSkipと同様、バッファに積まれている場合はバッファからそのボイスを取り除く。

pause

ボイスの中断

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

なし

【機能】



現在再生中のボイスを中断する。

下記のUnPauseメソッドを呼ぶことで再開する。

unPause

ボイスの再開

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

なし

【機能】

skip

再生中のボイスをスキップ

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

なし

【機能】



現在再生中のボイスをスキップし、バッファに積まれた次のボイスを再生する。

4.47. Vehicle.HMI.Sound.Alert.SoundEffect（略称：SoundEffect）

4.47.1. 位置付けと機能

Vehicle.HMI.Sound.Alert.SoundEffect (略称: SoundEffect) は，警告・通知用効果音の再生管理機能を提供するシングルインスタンスオブジェクトである。

4.47.1.1. SoundEffect(Alert)の位置付けと性質

Vehicle.HMI.Sound.Alert.SoundEffectは，衝突警告，車線逸脱警告などの安全に関わる警告音や，システム通知音の再生機能を提供する。アラート音は高い優先度を持ち，他の音声出力に割り込んで再生される。

4.47.2. 機能クラス

4.47.3. リスククラス

4.47.4. 構成情報

4.47.5. 状態

4.47.6. サービスコール一覧

SoundEffectに対するサービスコールは以下の通り。

play	警告SoundEffectの再生

4.47.7. イベント

4.47.8. データ型の定義

4.47.9. サービスコールの詳細規定

play

警告SoundEffectの再生

【パラメータ】

soundId	Uint32	音源を表すid

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

なし

【機能】



警告サウンドが吹鳴するとき、他の全てのサウンドは音量が絞られ、ユーザーサウンドのボイスは停止する。

警告サウンドは1音のみで、後に再生されたものを優先し、音量は既定の最大音量で固定となる。

4.48. Vehicle.HMI.Sound.Alert.Voice（略称：Voice）

4.48.1. 位置付けと機能

Vehicle.HMI.Sound.Alert.Voice（略称: Voice）は，警告・通知用音声メッセージの再生管理機能を提供するシングルインスタンスオブジェクトである。

4.48.1.1. Voice(Alert)の位置付けと性質

Vehicle.HMI.Sound.Alert.Voiceは，安全警告や緊急通知のための音声メッセージの再生機能を提供する。アラート音声は最高優先度で出力され，他の音声再生に割り込む。

4.48.2. 機能クラス

4.48.3. リスククラス

4.48.4. 構成情報

4.48.5. 状態

4.48.6. サービスコール一覧

Voiceに対するサービスコールは以下の通り。

play	警告ボイスの再生
playText	テキスト読み上げによる警告ボイスの再生

4.48.7. イベント

4.48.8. データ型の定義

4.48.8.1. 型定義

```
type VoiceInterruptBehavior = enum {
    PAUSE,  // 途中から再開
    STOP,   // 再吹鳴なし
    REPLAY  // 頭から再度再生
};
```

4.48.9. サービスコールの詳細規定

play

警告ボイスの再生

【パラメータ】

soundId	Uint32	音源を表すid
interruptBehavior	VoiceInterruptBehavior	中断された時の挙動

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

なし

【機能】

NOTE 上記のSound.Alert.SoundEffect.Playを参照

playText

テキスト読み上げによる警告ボイスの再生

【パラメータ】

text	String	読み上げるテキスト
interruptBehavior	VoiceInterruptBehavior	中断された時の挙動

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

なし

【機能】

NOTE 上記のSound.Alert.SoundEffect.Playを参照

4.49. Vehicle.HMI.Switch（略称：Switch）

4.49.1. 位置付けと機能

Vehicle.HMI.Switch は、車両の物理スイッチをアプリケーション上の論理スイッチとしてマッピングし、制御するためのマルチインスタンスオブジェクトである。

4.49.1.1. 本APIで扱うスイッチの範囲

ユーザーの操作に対し、物理的なストロークやフィードバックを伴うスイッチ（プッシュ、トグル、ロータリー）を対象とし、静電容量方式のタッチパネルや、ジェスチャー入力等の物理的ストロークを伴わないインターフェースは対象外とする。

4.49.2. 機能クラス

Switchは，以下の機能クラスを持つ。

StatusMonitorable	スイッチの押下状態を取得することができる
LedControllable	スイッチに付随するLEDの輝度をAPIから制御することができる
Customiziable	物理スイッチに対する論理機能のマッピング（用途）を変更することができる
ModeChangeable	スイッチの動作モード（Default/Custom/GameMode）を切り替えることができる

4.49.3. リスククラス

Switchは，以下のリスククラスを持つ。

T.B.D	T.B.D

4.49.4. 構成情報

Switchの各インスタンスは，以下の構成情報を持つ。

instanceId	IdType	インスタンスが持つId
switchType	SwitchInteractionType	スイッチ種別
ledInstallation	ComponentInstallationType	LEDの実装有無
position	String	スイッチの搭載位置を返す。 搭載位置は段数を特定できるものとする。 段数内のスイッチの割り振りは運転手から違い順に割り振る

4.49.5. 状態

Switchの各インスタンスは，以下の状態を持つ。

switchStatus	SwitchStatusType	現在の物理スイッチの動作状態。 ※詳細はSwitchStatusType の制約事項を参照。

4.49.6. サービスコール一覧

Switchに対するサービスコールは以下の通り。

getConfigAll	コンフィグ情報取得
getStatus	スイッチの押下状態
getSwitchMode	スイッチの現在モード
getSwitchCustomize	スイッチのカスタマイズ情報
getSwitchLEDBrightness	スイッチのLED輝度を取得する
setSwitchLEDBrightness	スイッチのLED輝度を設定する
setSwitchMode	スイッチの現在モードを変更する
setSwitchCustomize	スイッチのカスタマイズ
notify	スイッチイベントの通知要求
unnotify	スイッチイベントの通知停止
getEvent	スイッチイベントの取得

4.49.7. イベント

スイッチに対するイベント種別にはスイッチの押下、スイッチに付随するLEDの輝度変更、論理機能の変更、動作モードの変更がある。

Switchに対するイベント種別は以下の通り。

SwitchStateChanged	スイッチの押下状態が変更
BrightnessChanged	スイッチのLEDの輝度が変更
CustomizationChanged	スイッチのカスタマイズ変更
OperationModeChanged	スイッチの現在モード変更

4.49.8. データ型の定義

4.49.8.1. スイッチ種別

```
type SwitchInteractionType = enum {
    Momentary,  // 押している間だけONになるスイッチ
    Toggle, // 押すたびに状態が保持・反転するスイッチ
    Rotary  // 回転操作を伴うスイッチ
};
```

4.49.8.2. 部品実装の有無

```
type ComponentInstallationType = enum {
    NotInstalled,   // 未実装
    Installed   // 実装済み
};
```

4.49.8.3. モーメンタリスイッチの押下状態

```
type MomentarySwitchStatusType = enum {
    Released,   // リリース状態
    Pressed // 押下状態
};
```

4.49.8.4. トグルスイッチの押下状態

```
type ToggleSwitchStatusType = enum {
    Inactive,   // OFF状態
    Active  // ON状態
};
```

4.49.8.5. ロータリースイッチの押下状態

```
type RotarySwitchStatusType = enum {
    NoAction,   // 変化なし
    RotateRight,    // 右回転
    RotateLeft  // 左回転
};
```

4.49.8.6. スイッチの押下状態

```
type SwitchStatusType = struct {
    momentarySwichiStatus : MomentarySwitchStatusType,
    toggleSwichiStatus : ToggleSwitchStatusType,
    rotarySwichiStatus : RotarySwitchStatusType
};
```

4.49.8.7. スイッチのカスタマイズ状態

```
type CustomizeModeType = enum {
    Default,    // カスタマイズされていな状態を示す
    Customiz,   // スイッチをユーザー任意に設定しているモードを示す
    GameMode    // OS側の機能をバイパスし、アプリへ押下状態のみを通知するモード
};
```

4.49.8.8. スイッチの構成情報(getConfigAllが返すデータ型)

```
type SwitchConfigType = struct {
    instanceId : IdType,
    switchType : SwitchInteractionType,
    ledInstallation : ComponentInstallationType,
```

```
    position : String
};
```

4.49.8.9. スイッチイベント

```
type SwitchEventKindType = enum {
    SwitchStateChanged,
    BrightnessChanged,
    CustomizationChanged,
    OperationModeChanged
};
```

```
type SwitchEventType = struct {
    instanceId : IdType,
    eventInfo : SwitchEventKindType
};
```

4.49.9. サービスコールの詳細規定

getConfigAll

コンフィグ情報取得

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
config	SwitchConfigType[]	すべてのスイッチに関する構成情報

【エラーコード】

なし

【機能】

すべてのスイッチに関する構成情報を返す。

getStatus

スイッチの押下状態

【パラメータ】

instanceId	IdType	操作対象のスイッチのID

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
switchStatus	SwitchStatusType	モーメンタリ、トグル、ロータリスイッチの状態

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外

【機能】

操作対象のスイッチの押下状態を返す。

返却される switchStatus の中身は、物理スイッチの種別（switchType）に依存する。

- Momentary の場合: momentaryStatus フィールドのみを参照。
- Toggle の場合: toggleStatus フィールドのみを参照。
- Rotary の場合: rotaryStatus フィールドのみを参照。



OSにてスイッチのチャタリング除去を行い通知する。チャタリングと連打は各OEMにて判断する。

getSwitchMode

スイッチの現在モード

【パラメータ】

instanceId	IdType	操作対象のスイッチのID

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
customizMode	CustomizeModeType	スイッチのカスタマイズ状態

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外

【機能】

操作対象のスイッチの現在モードを返す。

割り当たっているカスタマイズの情報はgetSwitchCustomizeで取得する。

getSwitchCustomize

スイッチのカスタマイズ情報

【パラメータ】

instanceId	IdType	操作対象のスイッチのID

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
function	String	カスタマイズ後の割り当て機能。 戻り値の例としてドライバーのエアコン温度を上げる機能が割り当たっている場合は以下を返す。 "HVAC.TempUpDriverSide"

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外

【機能】

スイッチに割り当たっている機能を取得する機能

getSwitchLEDBrightness

スイッチのLED輝度を取得する

【パラメータ】

instanceId	IdType	操作対象のスイッチのID

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
brightness	UInt32	**正常状態** [0%〜100%], **故障状態** [101以上]

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外

【機能】

スイッチのLED輝度を取得する。

setSwitchLEDBrightness

スイッチのLED輝度を設定する

【パラメータ】

instanceId	IdType	操作対象のスイッチのID
brightness	UInt32	0〜100を指定する。

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外

【機能】

スイッチのLED輝度を設定する。

setSwitchMode

スイッチの現在モードを変更する

【パラメータ】

instanceId	IdType	操作対象のスイッチのID
customizMode	CustomizeModeType	スイッチのカスタマイズ状態

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外

【機能】

スイッチモードを変更する

機能割り当てはsetSwitchCustomizeにて行う。

setSwitchCustomize

スイッチのカスタマイズ

【パラメータ】

instanceId	IdType	操作対象のスイッチのID
function	String	カスタマイズ後の割り当て機能。 例としてドライバーのエアコン温度を上げる機能が割り当てる場合は"HVAC.TempUpDriverSide"を設定する。

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外

【機能】

スイッチモードがCustomizの場合に任意の機能を割り当てる。

スイッチモードはgetSwitchModeで取得する。

notify

スイッチイベントの通知要求

【パラメータ】

instanceId	IdType?	操作対象のスイッチのID
eventFilter	SwitchEventKindType?	通知するイベントの種類の集合

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【エラーコード】

E_INVALID_ID	instanceIdが有効範囲外
E_INVALID_PARAMETER	eventFilterが有効範囲外

【機能】

イベント通知対象のスイッチで発生したeventFilterで指定したイベントの通知を要求する。イベントを通知するためのイベントキューを生成し，その通知ハンドルを notifyHandle に返す。

instanceIdを省略した場合，すべてのインスタンスで発生したイベントの通知を要求する。eventFilterを省略した場合，すべての種類のイベントの通知を要求する。

unnotify

スイッチイベントの通知停止

【パラメータ】

notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正

【機能】

notifyHandleで指定した通知を停止する。指定した通知用のイベントキューは削除される。

getEvent

スイッチイベントの取得

【パラメータ】

notifyHandle	NotifyHandleType	イベントキューを識別するための通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
event	SwitchEventType	取得したイベント

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正
E_NO_DATA	イベントキューが空

【機能】

notifyHandleで指定した通知用のイベントキューから，先頭のイベントを取り出す。取り出したイベントは，イベントキューから削除される。



イベントがパラメータ等で渡されてくるプログラミング環境向けの物理APIでは，用意する必要がない。

4.50. Vehicle.Motion（略称：Motion）

4.50.1. 位置付けと機能

Vehicle.Motion（略称：Motion）は，車両の運動を操作するためのシングルトンオブジェクトである。Motionの状態を参照することにより，車両の現在の速度や加速度を取得することができる。また，Motionを制御することにより，車両のパワートレイン，ブレーキ，ステアリングなどを制御することができる。

4.50.1.1. Motionの機能

Motionは，車両を縦方向の運動と横方向の運動を独立して制御する機能を持つ。縦方向の運動は，目標速度を指定して制御する方法（速度制御）と，停止するまでの距離を指定して制御する方法（停止制御）がある。横方向の運動は，目標軌跡を指定して制御する（軌跡追従制御）。

縦方向の速度制御では，指定された目標速度に向けて加減速または速度維持を行う。加減速する場合には，指定された応答プロファイルによって定まる加減速度制限をかける（Figure 21）。ただし，応答プロ

ファイルが最大性能での加減速（Maximum）の場合には，加減速度制限はかけない。

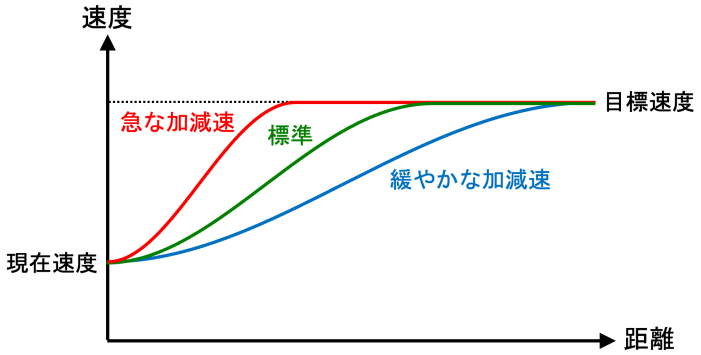


Figure 21. 応答プロファイルの速度制御への適用イメージ

縦方向の停止制御では，指定された目標停止位置に向けて速度維持または減速を行い，目標位置で車両を停止させる。減速する場合には，指定された応答プロファイルによって定まる減速度制限をかける（Figure 22）。ただし，応答プロファイルが最大性能での加減速（Maximum）の場合には，減速度制限をかけず，車両の最大性能で停止する。車両が目標停止位置で停止した後は，停止状態を維持する。停止状態を維持する間も，停止制御が継続しているものと扱う。

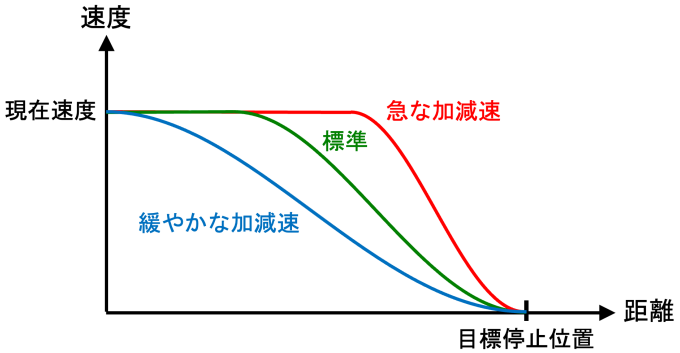


Figure 22. 応答プロファイルの停止制御への適用イメージ

なお，ビークルOSは，必要に応じて，縦方向の運動制御に対してジャーク制限をかける。

横方向の軌跡追従制御では，指定された目標軌跡に沿って走行するように制御を行う。軌跡追従制御において，指定された許容横方向誤差を満たせない場合，車両を減速させる場合がある。

アプリケーションは，目標軌跡を設定するサービスコールを継続的に呼び出して，十分に先までの目標軌跡を設定することを想定している。軌跡追従制御中に目標軌跡の残っている点が少なくなり，安定した横方向制御が難しくなった場合，ビークルOSはイベント（LateralPathInsufficient）を用いてアプリケーションに通知する。また，車両が目標軌跡の最後の点に到達することはないのが原則であるが，アプリケーションが想定したようにサービルコールを呼び出さず，目標軌跡の最後の点に到達した場合，その後は，その時点の曲率を維持するように制御を継続する。

4.50.1.2. 運転者の操作による制御と安全機能

運転者は，アクセルペダル，ブレーキペダル，ステアリングホイールなどを操作することで，車両の縦方向および横方向の運動を制御することができる（自動運転時には，運転者の操作による制御を無効化することもある）。運転者の操作による制御は，アプリケーションで実現された標準制御と扱う。すなわち，「運転者の操作による制御」というアプリケーションが，アクセルペダル，ブレーキペダル，ステアリングホイールなどの状態を元に，ある操作優先度で，車両の運動を制御するものと扱う。ただし，このアプリケーションは，MotionのAPIを利用するわけではない（Figure 23）。

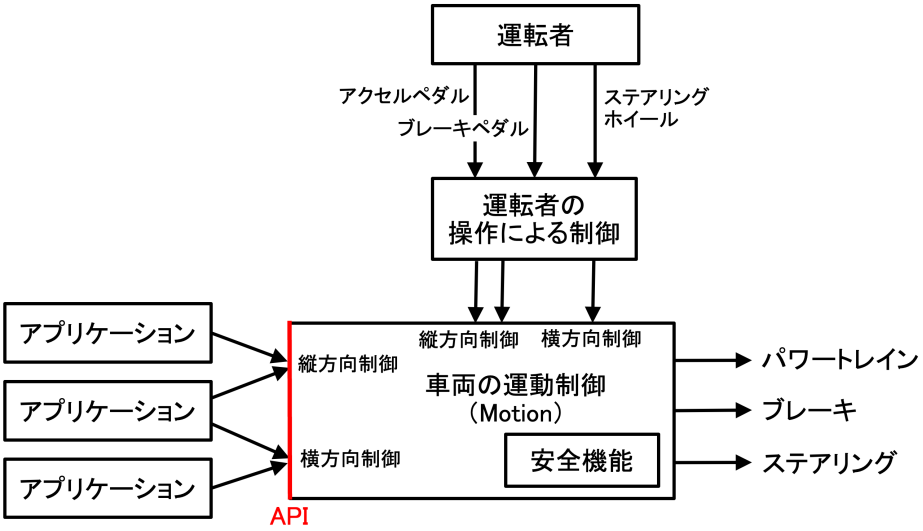


Figure 23. Motionの制御の構造

ビークルOSが車両運動に関してどのような安全機能を持つかは，車両依存である。ビークルOSが何らかの危険を検知した場合，ビークルOSの安全機能は，縦方向または横方向またはその両方の制御をオーバライドする。

目標軌跡に対する追従制御中に，許容横方向誤差を満たすために車両を減速させている間は，縦方向制御は安全機能によりオーバライドされたものと扱う。減速しても許容横方向誤差を満たせない場合には，横方向および縦方向の制御は安全機能によりオーバライドされ，車両の振る舞いは安全機能に委ねられる。

4.50.1.3. Motionに対する制御の調停

Motionに対する制御の調停には，縦方向制御と横方向制御のそれぞれに対して，標準的な調停機能（後勝ちの原則）が適用される。また，縦方向制御と横方向制御のそれぞれに対して，LockableObjectと同等の標準的なロック機能を持つ。

車両の故障により制御できなくなった場合や，安全機能が制御をオーバライドした場合，ロックは解除され，それらの状態から回復するまで，ロックを取得することはできない。

4.50.2. 機能クラス

Motionは，以下の機能クラスを持つ。

ControlLongitudinal	APIにより縦方向制御ができる
ControlLateral	APIにより横方向制御ができる

4.50.3. リスククラス

Motionは，以下のリスククラスを持つ。

RiskVehicleInstability	車両の運動安定性を失うリスク（スピン，横滑り，横転など）
RiskCollision	自車両の責任で何らかの物体（他の車両，歩行者，障害物，落下物など）に衝突するリスク
RiskViolation	交通規則に違反するリスク
RiskRoadHazard	道路の損傷，欠損，路肩崩落などにより安全に走行可能な路面が確保できないリスク

NOTE

以上のリスククラスの定義は，現時点でのイメージを示したものであり，より詳細な検討が必要である。

4.50.4. 構成情報

Motionは，以下の構成情報を持つ。

capabilities	enumSet (MotionCapabilityType)	備えている機能クラスの集合
riskClasses	enumSet(MotionRiskType)	対策できていないリスククラスの集合

この他に，車両の運動性能に関する情報を追加する（TBD）。

4.50.5. 状態

Motionは，以下の状態を持つ。

speed	SpeedType	現在速度
acceleration	AccelerationType	現在加速度
angularVelocity	AngularVelocityType	現在回転速度
longitudinalLockStatus	LockStatusType	縦方向のロック状態
lateralLockStatus	LockStatusType	横方向のロック状態
longitudinalControlStatus	LongitudinalControlStatusType	縦方向制御の状態
targetSpeed	SpeedType?	目標速度
distanceToStop	UInt32?	目標停止位置までの残り道のり距離（単位：mm）
responseProfile	LongitudinalResponseProfileType?	応答プロファイル
lateralControlStatus	LateralControlStatusType	横方向制御の状態
targetPath	RelativePathType?	残り目標軌跡
lateralTolerance	UInt32?	許容横方向誤差（単位：mm）

targetSpeedは，縦方向が速度制御されている間のみ，distanceToStopは，縦方向が停止制御されている間のみ，responseProfileは，縦方向が速度制御または停止制御されている間のみ有効である。また，targetPathとlateralToleranceは，横方向が軌跡追従制御されている間のみ有効である。

縦方向制御の状態（longitudinalControlStatus）の状態遷移を，Figure 24に示す。故障回復時やオーバライド解除時には，速度制御中と停止制御中のいずれに遷移する場合もある。

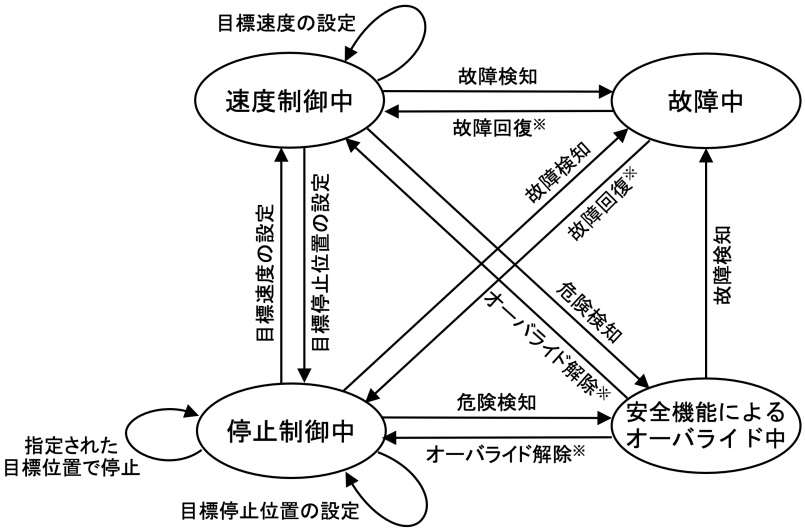


Figure 24. Motionの縦方向制御の状態遷移

また，横方向制御の状態（lateralControlStatus）の状態遷移を，Figure 25に示す。故障回復時やオーバライド解除時には，軌跡追従制御中と曲率維持中のいずれに遷移する場合もある。

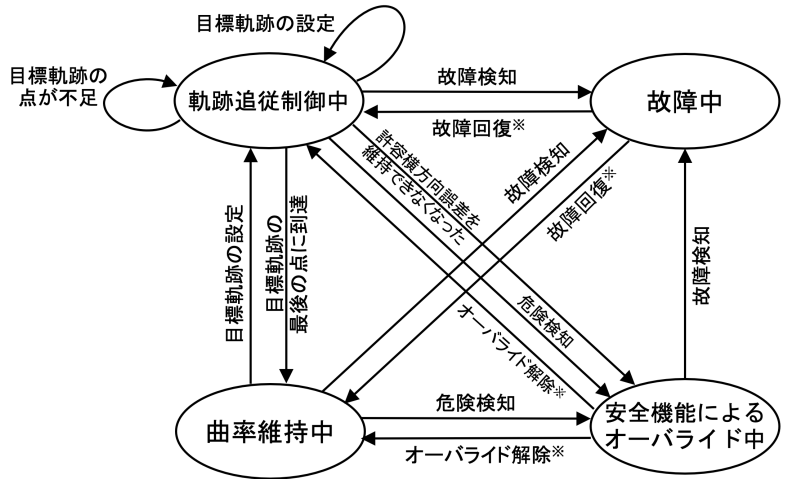


Figure 25. Motionの横方向制御の状態遷移

distanceToStopは，目標停止位置までの残り道のり距離であり，サービスコールにより目標停止位置までの道のり距離を設定した直後はその値となるが，その後は，走行した距離分減少した値となる。

targetPathは，目標軌跡の残り部分であり，サービスコールにより目標軌跡を設定した直後はその値となるが，その後は，走行に伴って到達した点が除かれたものとなる。

NOTE

speed，acceleration，angularVelocityの定義は，COVESA VSS 6.0に合わせた。

4.50.6. サービスコール一覧

Motionに対するサービスコールは以下の通り。

getConfig	Motionの構成情報の参照
getStatus	Motionの状態の参照
setSpeed	目標速度の設定（縦方向制御）
stopDistance	目標停止位置の設定（縦方向制御）

setPath	目標軌跡の設定（横方向制御）
lockLongitudinal	Motionの縦方向制御のロックの取得
unlockLongitudinal	Motionの縦方向制御のロックの解除
lockLateral	Motionの横方向制御のロックの取得
unlockLateral	Motionの横方向制御のロックの解除
notify	Motionイベントの通知要求
unnotify	Motionイベントの通知停止
getEvent	Motionイベントの取得

4.50.7. イベント

Motionに対するイベント種別は以下の通り。

LongitudinalControlledBySelf	自アプリケーションが縦方向制御
LongitudinalControlledByOther	他のアプリケーションが縦方向制御
LongitudinalStopCompleted	指定された目標位置で停止
LongitudinalFaultDetected	縦方向制御の故障検知
LongitudinalFaultRecovered	縦方向制御が故障から回復
LongitudinalOverrideActivated	安全機能が縦方向制御をオーバライド
LongitudinalOverrideReleased	縦方向制御のオーバライドが解除
LateralControlledBySelf	自アプリケーションが横方向制御
LateralControlledByOther	他のアプリケーションが横方向制御
LateralPathEndReached	指定された目標軌跡の最後の点に到達
LateralPathInsufficient	指定された目標軌跡の点が不足
LateralFaultDetected	横方向制御の故障検知
LateralFaultRecovered	横方向制御が故障から回復
LateralToleranceExceeded	許容横方向誤差を満たせなくなった
LateralOverrideActivated	安全機能が横方向制御をオーバライド
LateralOverrideReleased	横方向制御のオーバライドが解除
OwnLongitudinalLockRevoked	自アプリケーションの縦方向制御のロックの強制解除

OtherLongitudinalLockReleased	他のアプリケーション等による縦方向制御のロック解除
OwnLateralLockRevoked	自アプリケーションの横方向制御のロックの強制解除
OtherLateralLockReleased	他のアプリケーション等による横方向制御のロック解除
EventOverflow	イベントキューのオーバフロー
ConfigChanged	構成情報の変更

横方向制御において許容横方向誤差を満たせなくなった場合，横方向制御は安全機能によりオーバライドされるが，この時は，LateralToleranceExceededのみが発生し，LateralOverrideActivatedは発生しない。

この内，イベント種別がLongitudinalControlledByOtherの場合には，以下の関連情報を持つ。

sourceApplication	ApplicationIdType	操作したアプリケーション

また，イベント種別がLongitudinalStopCompletedの場合には，以下の関連情報を持つ。

sourceApplication	ApplicationIdType	操作したアプリケーション

また，イベント種別がLateralControlledByOtherの場合には，以下の関連情報を持つ。

sourceApplication	ApplicationIdType	操作したアプリケーション

また，イベント種別がLateralPathEndReachedの場合には，以下の関連情報を持つ。

sourceApplication	ApplicationIdType	操作したアプリケーション

また，イベント種別がLateralPathInsufficientの場合には，以下の関連情報を持つ。

sourceApplication	ApplicationIdType	操作したアプリケーション

また，イベント種別がEventOverflowの場合には，以下の関連情報を持つ。

noLostEvents	UInt32	失われたイベントの数

4.50.8. データ型の定義

(1) Motionの機能クラス

```
type MotionCapabilityType = enum {
    ControlLongitudinal,
    ControlLateral
```

```
};
```

(2) Motionに対するリスククラス

```
type MotionRiskType = enum {
    RiskVehicleInstability,
    RiskCollision,
    RiskViolation,
    RiskRoadHazard
};
```

(3) Motionの構成情報（getConfigが返すデータ型）

```
type MotionConfigType = struct {
    capabilities : enumSet(MotionCapabilityType),
    riskClasses : enumSet(MotionRiskType)
};
```

(4) Motionの状態

```
type SpeedType = Float32;   // 速度（単位：km/h）。正の値は前進，負の値は後退を示す
```

```
type AccelerationType = struct {
    longitudinal : Float32, // 縦方向加速度（単位：m/s^2）
    lateral : Float32,  // 横方向加速度（単位：m/s^2）
    vertical : Float32  // 鉛直方向加速度（単位：m/s^2）
};
```

```
type AngularVelocityType = struct {
    roll : Float32, // ロール（前後軸まわりの回転速度，単位：度/s）
    pitch : Float32,    // ピッチ（左右軸まわりの回転速度，単位：度/s）
    yaw : Float32   // ヨー（鉛直軸まわりの回転速度，単位：度/s）
};
```

```
type LongitudinalControlStatusType = enum {
    SpeedControl,   // 速度制御中
    StopControl,    // 停止制御中
    Fault,  // 故障中
    SafetyOverride  // 安全機能によるオーバライド中
};
```

```
type LateralControlStatusType = enum {
```

```
    PathControl,    // 軌跡追従制御中
    CurvatureHold,  // 曲率維持中
    Fault,  // 故障中
    SafetyOverride  // 安全機能によるオーバライド中
};
```

```
type MotionStatusType = struct {
    speed : SpeedType,
    acceleration : AccelerationType,
    angularVelocity : AngularVelocityType,
    longitudinalLockStatus : LockStatusType,
    lateralLockStatus : LockStatusType,
    longitudinalControlStatus : LongitudinalControlStatusType,
    targetSpeed : SpeedType?,
    distanceToStop : UInt32?,
    responseProfile : LongitudinalResponseProfileType?,
    lateralControlStatus : LateralControlStatusType,
    targetPath : RelativePathType?,
    lateralTolerance : UInt32?
};
```

(5) Motionの応答プロファイル

```
type LongitudinalResponseProfileType = enum {
    Standard,   // 標準的な加減速
    Maximum,    // 車両の最大性能での加減速
    Rapid,  // 急な加減速
    Gentle  // 緩やかな加減速
};
```

(6) 走行軌跡

```
type Relative2DPositionType = struct {
    x : Int32,  // 車両の前後方向の相対位置（単位：mm，前方が正）
    y : Int32   // 車両の左右方向の相対位置（単位：mm，左方向が正）
};
```

Relative2DPositionTypeは，車両に固定された座標系（車両座標系）における相対位置を表すデータ型である。相対位置の原点は車両の後軸中心とする。

```
type RelativePathType = struct {
    points : Relative2DPositionType[],  // 軌跡を構成する点の列
    referenceTime : TimestampMsType // pointsが参照する車両座標系の基準時刻
};
```

アプリケーションが自車位置や周辺環境を認識した時点から，このデータ型を用いるサービスコールを呼

び出すまでの間に，車両は移動している。アプリケーションが，軌跡計算の元となった車両状態の取得時刻をreferenceTimeに指定し，pointsの各点はreferenceTime時点の車両座標系を基準とした相対位置を表すとすることで，車両の移動による位置ズレを補正した正確な軌跡の指定が可能となる。



Relative2DPositionTypeは，将来的には，共通データ型に移動すべきと思われる。

4.50.9. サービスコールの詳細規定

getConfig

Motionの構成情報の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
config	MotionConfigType	Motionの構成情報

【エラーコード】

なし

【機能】

Motionの構成情報を返す。

getStatus

Motionの状態の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
status	MotionStatusType	Motionの状態

【エラーコード】

なし

【機能】

Motionに関するすべての状態を返す。

setSpeed

目標速度の設定（縦方向制御）

【パラメータ】

targetSpeed	SpeedType	目標速度
responseProfile	LongitudinalResponseProfileType?	応答プロファイル
priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_PARAMETER	targetSpeed，responseProfile，priorityが有効範囲外
E_RISK_APPLICATION	アプリケーションが対応していないリスクがある
E_RISK_USER	ユーザがリスクを承認していない
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_FAULT	Motionの縦方向制御が故障中
E_OBJECT_STATUS	Motionの縦方向制御が安全機能にオーバライドされている
E_OBJECT_LOCKED	Motionの縦方向制御が，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている
E_NOT_SUPPORTED	縦方向制御する機能を持たない

【機能】

目標速度をtargetSpeed，応答プロファイルをresponseProfileに設定して，車両の速度制御を開始する。

停止制御中に目標速度を設定した場合，目標停止位置の設定は解除され，停止制御は中断される。

responseProfileを省略した場合，標準的な加減速（Standard）を指定したものと扱う。

リスククラスRiskViolationに対応できていない場合でも，車両が私有地を走行していると判断できた場合には，操作を受け付ける。そうでない場合には，E_RISK_APPLICATIONエラーまたはE_RISK_USERエラーとなる。

stopDistance

目標停止位置の設定（縦方向制御）

【パラメータ】

distanceToStop	UInt32	目標停止位置までの道のり距離（単位：mm）
responseProfile	LongitudinalResponseProfileType?	応答プロファイル

priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_PARAMETER	distanceToStop，responseProfile，priorityが有効範囲外
E_RISK_APPLICATION	アプリケーションが対応していないリスクがある
E_RISK_USER	ユーザがリスクを承認していない
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_FAULT	Motionの縦方向制御が故障中
E_OBJECT_STATUS	Motionの縦方向制御が安全機能にオーバライドされている
E_OBJECT_LOCKED	Motionの縦方向制御が，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている
E_INFEASIBLE	車両が停止中。指定された距離で停止できない
E_NOT_SUPPORTED	縦方向制御する機能を持たない

【機能】

目標停止位置までの道のり距離（目標軌跡に沿った距離）をdistanceToStop，応答プロファイルをresponseProfileに設定して，車両の停止制御を開始する。

車両が停止中にこのサービスコールを呼び出した場合は，distanceToStopが0でない限り，E_INFEASIBLEエラーとなる。

distanceToStopで指定された距離で停止できない場合は，次のように振る舞う。応答プロファイルが最大性能での加減速（Maximum）の場合には，車両に可能な最も短い距離で停止するように制御する。それ以外の場合は，E_INFEASIBLEエラーとなる。

速度制御中に目標停止位置を設定した場合，目標速度の設定は解除され，速度制御は中断される。

responseProfileを省略した場合，標準的な加減速（Standard）を指定したものと扱う。

リスククラスRiskViolationに対応できていない場合でも，車両が私有地を走行していると判断できた場合には，操作を受け付ける。そうでない場合には，E_RISK_APPLICATIONエラーまたはE_RISK_USERエラーとなる。

setPath

目標軌跡の設定（横方向制御）

【パラメータ】

path	RelativePathType	目標軌跡
lateralTolerance	UInt32?	許容横方向誤差（単位：mm）

priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_PARAMETER	path，lateralTolerance，priorityが有効範囲外
E_RISK_APPLICATION	アプリケーションが対応していないリスクがある
E_RISK_USER	ユーザがリスクを承認していない
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_FAULT	Motionの横方向制御が故障中
E_OBJECT_STATUS	Motionの横方向制御が安全機能にオーバライドされている
E_OBJECT_LOCKED	Motionの横方向制御が，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている
E_INFEASIBLE	指定された許容横方向誤差を満たせない
E_NOT_SUPPORTED	横方向制御する機能を持たない

【機能】

目標軌跡をpath，許容横方向誤差をlateralToleranceに設定し，軌跡追従制御を開始する。

車両を減速させてもlateralToleranceで指定された許容横方向誤差を満たせない場合には，E_INFEASIBLEエラーとなる。

lateralToleranceを省略した場合，横方向誤差に関する制約は設けないものと扱う。

リスククラスRiskViolationに対応できていない場合でも，車両が私有地を走行していると判断できた場合には，操作を受け付ける。そうでない場合には，E_RISK_APPLICATIONエラーまたはE_RISK_USERエラーとなる。

lockLongitudinal

Motionの縦方向制御のロックの取得

【パラメータ】

priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_PARAMETER	priorityが有効範囲外

E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_LOCKED	Motionの縦方向制御が，priorityと同じかそれより高い優先度でロックされている（自アプリケーションが，priorityと同じかそれより高い優先度でロックしている場合を含む）
E_OBJECT_FAULT	Motionの縦方向制御が故障中
E_OBJECT_STATUS	Motionの縦方向制御が安全機能にオーバライドされている

【機能】

Motionの縦方向制御を，指定した操作優先度でロックする。

unlockLongitudinal

Motionの縦方向制御のロックの解除

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_OBJECT_STATUS	自アプリケーションが，Motionの縦方向制御をロックしていない

【機能】

Motionの縦方向制御を，ロック解除する。

lockLateral

Motionの横方向制御のロックの取得

【パラメータ】

priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_PARAMETER	priorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値

E_OBJECT_LOCKED	Motionの横方向制御が，priorityと同じかそれより高い優先度でロックされている（自アプリケーションが，priorityと同じかそれより高い優先度でロックしている場合を含む）
E_OBJECT_FAULT	Motionの横方向制御が故障中
E_OBJECT_STATUS	Motionの横方向制御が安全機能にオーバライドされている

【機能】

Motionの横方向制御を，指定した操作優先度でロックする。

unlockLateral

Motionの横方向制御のロックの解除

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_OBJECT_STATUS	自アプリケーションが，Motionの横方向制御をロックしていない

【機能】

Motionの横方向制御を，ロック解除する。

notify

Motionイベントの通知要求

【パラメータ】

eventFilter	enumSet (MotionEventKindType)?	通知するイベントの種類の集合

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【エラーコード】

E_INVALID_PARAMETER	eventFilterが有効範囲外

【機能】

Motionで発生したeventFilterで指定した種別のイベントの通知を要求する。イベントを格納するためのイベントキューを生成し，その通知ハンドルをnotifyHandleに返す。

unnotify

Motionイベントの通知停止

【パラメータ】

notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正

【機能】

notifyHandleで指定した通知を停止する。イベントを格納するためのイベントキューを削除する。

getEvent

Motionイベントの取得

【パラメータ】

notifyHandle	NotifyHandleType	イベントを取得するイベントキューの通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
event	MotionEventType	取得したイベント

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正
E_NO_DATA	イベントキューが空

【機能】

notifyHandleで指定した通知用のイベントキューから，先頭のイベントを取り出す。取り出したイベントは，イベントキューから削除される。



イベントがパラメータ等で渡されてくるプログラミング環境向けの物理APIでは，用意する必要がない。

4.51. Vehicle.Occupant（略称：Occupant）

＜TBD＞

4.52. Vehicle.Atmosphere（略称：Atmosphere）

＜TBD＞

Chapter 5. 拡張ビークルAPI

5.1. Vehicle.ADADAS（略称：ADADAS）

5.1.1. 位置付けと機能

Vehicle.ADADAS（略称：ADADAS）は，車両構成の階層において運転支援・自動運転（AD/ADAS）のノードに対応するシングルインスタンスオブジェクトである。このノードの下には，個々のAD/ADAS機能を提供するオブジェクト（Vehicle.ADADAS.ACC，Vehicle.ADADAS.LDW等）が置かれる。本オブジェクトは，その車両に搭載されているAD/ADAS機能の目録（どの機能が搭載され，どのオブジェクトがそれを提供するか）と，AD/ADASシステム全体について一つに定まる状態（DDTの責任主体，移管の段階，運行設計領域の充足状態，縮退運転と最小リスク状態への移行の状態）を提供する。

個々の機能の設定値・作動状態・出力・操作は本オブジェクトでは扱わない。それらは，目録が指し示す機能オブジェクトが提供する。

5.1.1.1. ADADASの位置付けと性質

本オブジェクトは，AD/ADAS機能の全体像をアプリケーションに提示する集約のオブジェクトである。機能ごとに定まる値（設定値・作動状態・出力・故障の区分）は保持せず，それらを提供する下位のオブジェクトへの案内（目録）と，AD/ADASシステム全体について一つに定まる状態を提供する。アプリケーションは，まず本オブジェクトを参照して車両に何が搭載されているかを知り，次に目録が指すオブジェクトを参照して個々の機能を扱う。

本オブジェクトは車両構成の階層の中間のノードに対応するため，その下に置かれた機能オブジェクトの構成情報が変化した場合にも，本オブジェクトのConfigChangedイベントが発生する。

5.1.1.2. 目録としての役割

構成情報（getConfig）は，その車両に搭載されているAD/ADAS機能の目録を返す。目録の各要素は，機能の種別（functionType），その機能を提供するオブジェクトの完全修飾名（providingObject），および設計上対応する自動化レベル（automationLevel）を持つ。アプリケーションは，providingObjectにより次に参照すべきオブジェクトを知り，そのオブジェクトの構成情報から，当該機能に対して行える操作と設定の値域を得る。

目録は，その機能が車両に搭載されているか（静的または準静的な事実）のみを表す。機能がいま有効か，作動中かは，目録ではなく各機能オブジェクトの状態（getStatus）が表す。したがって，機能の有効化・無効化や作動の開始・終了によって構成情報は変化せず，これらによるConfigChangedイベントも発生しない。

本オブジェクトのConfigChangedイベントが発生するのは，ソフトウェアの更新により搭載機能が追加・削除された場合，機能オブジェクトのインスタンスが増減した場合，および下位に置かれた機能オブジェクトの構成情報が変化した場合である。最後の場合が含まれるのは，本オブジェクトが車両構成の階層の中間のノードに対応するためである。

下位に置かれた機能オブジェクトの構成情報が変化した場合，本オブジェクトの構成情報は変化しないことがある。この場合にアプリケーションが参照すべきなのは，目録のprovidingObjectが指すオブジェクトのgetConfigである。

目録に現れる機能のうち，実体が複数ある機能は，当該機能オブジェクトをマルチインスタンスオブジェクトとして定義する。その場合，アプリケーションは当該オブジェクトのgetConfigAllによりインスタンスを列挙する。目録はインスタンスの識別子を持たない。



本オブジェクトおよび目録が指し示す各機能オブジェクトは，関連オブジェクト（RelatedObjectType）として参照されうる。このため，共通データ型ObjectKindTypeには，本オブジェクトおよび各機能オブジェクトに対応する列挙子を設ける。

5.1.1.3. AD/ADASシステム全体の状態

状態（getStatus）は，AD/ADASシステム全体について一つに定まる状態を返す。すなわち，いまこの車両の動的運転タスク（DDT）を実行している主体（ddtOwner），移管の段階（handoverPhase），運行設計領域の充足状態（oddStatus），縮退運転の状態（fallbackState），および最小リスク状態への移行の状態（mrcState）である。

oddStatusは，運行設計領域の充足状態のみを表す。運行設計領域の内容そのもの（対象とする道路・天候・速度域等の条件）は，本オブジェクトでは扱わない。

これらを本オブジェクトに集約するのは，「いまこの車両を運転しているのは誰か」に対する答えが機能ごとに別々にあってはならないためである。機能ごとに定まる状態（作動状態・設定値・出力・故障の区分）は，本オブジェクトではなく各機能オブジェクトが持つ。

AD/ADASシステム全体を提供できない故障を表す状態は持たない。AD/ADASを統括する制御装置の故障のように複数の機能に及ぶ故障も，影響を受ける各機能オブジェクトの作動状態operatingStateがFaultとなることで表れる。本オブジェクト自体が状態を提供できない場合は，getStatusがE_OBJECT_FAULTエラーを返す。

状態の取得は，要求応答（getStatus）と，各状態項目の変化のイベントによる。

5.1.1.4. DDTの責任の表し方

DDTの責任は，主体を表すddtOwnerと，移管の段階を表すhandoverPhaseの2項目で表す。ddtOwnerは，運転者（Driver），AD/ADASシステム（AutomationSystem），遠隔の操作者（RemoteOperator）のいずれがDDTを実行しているかを表し，AD/ADASシステムが実行している場合には，どの機能オブジェクトが実行しているかをfunctionで示す。functionが値を持つのは，kindがAutomationSystemである場合のみである。

kindが表すのはDDTを実行している主体の種別であり，オブジェクトの名称ではない。列挙子AutomationSystemは，車両に搭載されたAD/ADASの機能が運転の操作を実行していることを表す。

functionは機能オブジェクトの配列である。縦方向の運動と横方向の運動を別の機能が担う構成のように，複数の機能が同時にDDTを担うことがあるためである。kindがAutomationSystemである場合，functionは1つ以上の要素を持ち，それ以外の場合，functionは空の配列となる。

handoverPhaseは，責任の所在が安定している状態（Stable），運転者とシステムが分担している状態（Shared），移管が進行中である状態（Transitioning）を表す。handoverPhaseがTransitioningである間，ddtOwnerは移管元（移管が完了するまで実際にDDTを実行している側）を指す。移管先は状態としては持たず，TransitionRequestedイベントの関連情報（targetOwnerKind）で渡す。移管先は「これから誰に渡すか」であって現在の状態ではないためである。

運転支援の機能のみを搭載する車両では，支援の作動中もDDTの責任は運転者にあり，ddtOwnerは常にDriverである。車両の運動に介入する運転支援の機能が，現に介入を行っている間は，handoverPhaseをSharedとする。継続的に介入する機能（ACC，車線中央を維持するLKA等）は作動中（Active）である間，断続的に介入する機能（AEB，LCA，車線逸脱を防止するLKA等）は現に介入を行っている間（各機能オブジェクトが対応する状態を提供する場合は，その状態による）が，これに当たる。いずれの機能も介入を行っていない間，handoverPhaseはStableとなる。警報・提示のみを行う機能（車両の運動に介入しない機能）の作動は，handoverPhaseに影響しない。

5.1.1.5. AD/ADAS機能共通の作動状態モデル

各機能オブジェクトは，作動状態operatingStateと，作動の開始・継続ができるかを表すavailabilityを共通の形で持つ。以下をAD/ADAS機能共通の作動状態モデルとして規定する。各機能オブジェクトは，これに従う。

operatingStateは，その機能が現にどの作動状態にあるかを表す（値の定義はデータ型AutomationOperatingStateTypeの定義に示す）。availabilityは，その機能がいま作動を開始または継続できるかを表す。両者は別の軸であり，「作動していないが，いつでも作動を開始できる」と「作動していないが，条件を満たさないため作動を開始できない」を区別するために用いる。機能が有効であることと，現に作動していることは，別の項目とはせず，いずれもoperatingStateの値として表す。

作動開始の要求（activate）の受理により機能が有効となり，作動終了の要求（deactivate）の受理により機能が無効となる。いずれの要求も，正常終了は要求が受理されたことのみを意味し，目的の作動状態への遷移が完了したことを意味しない。遷移の完了は，状態の参照（getStatus）またはイベントにより確認する。要求の受理から遷移の完了までの間は，過渡状態（Activating，Deactivating）が現れる。

operatingStateとavailabilityの組合せには，次の規則を適用する。

- availabilityがAvailable以外の値である場合，unavailableReasonが値を持つ。Availableである場合，unavailableReasonは値を持たない。
- operatingStateがFaultである場合，availabilityはUnavailableとなり，unavailableReasonはFunctionFaultとなる。このとき，状態faultCategoryが故障の区分を表す。operatingStateがFault以外である場合，faultCategoryは値を持たない。
- 機能が無効（Off）である間，availabilityは，その機能を有効にできるかどうか（故障の有無等）のみを表し，有効化後にのみ意味を持つ作動条件（車速範囲，シフト位置，対象の認識状態等）は反映しない。作動を開始する要求（activate）が受理されると，機能は有効となる。このときの作動状態は，作動の条件が整っていればActive，整っていなければStandbyとなる（有効化の直後にどの作動状態をとるかは，各機能オブジェクトの規定による）。有効化後の作動条件は，Standbyにある間のavailabilityおよびunavailableReasonで表す。
- availabilityがTemporarilyUnavailableまたはUnavailableであり，かつoperatingStateがActiveである組合せは，過渡的にのみ現れる。作動を継続できないため，車両は直ちにSuspended，Standby，Fallback，MRCまたはFaultのいずれかへ遷移する。
- 運転者の操作または他の経路による車両の運動の制御により作動が抑止されている間は，operatingStateをSuspended，availabilityをTemporarilyUnavailable，unavailableReasonをControlOverriddenとする。
- 他のAD/ADAS機能がその運転操作を担っている間は，operatingStateをSuspended，availabilityをTemporarilyUnavailable，unavailableReasonをSupersededByOtherFunctionとし，blockingFunctionがその機能を指す。運転操作の全体を担う機能が作動している間，その運転操作に含まれる個別の機能は，この組合せで表す。
- unavailableReasonがDependentFunctionFaultまたはSupersededByOtherFunctionである場合，blockingFunctionが，作動を妨げている機能オブジェクトを指す。それ以外の場合，blockingFunctionは空の配列となる。

作動開始の要求および設定値を変更する要求に対する応答には，次の規則を適用する。

- availabilityがTemporarilyUnavailableである場合，作動開始の要求はE_INFEASIBLEエラーとなる。条件が回復すれば作動できるためである。

- availabilityがUnavailableであり，unavailableReasonがFunctionFaultである場合，作動開始の要求はE_OBJECT_FAULTエラーとなる。
- availabilityがUnavailableであり，unavailableReasonがFunctionFault以外である場合，作動開始の要求はE_OBJECT_STATUSエラーとなる。
- 過渡状態（Activating，Deactivating）にある間，および縮退運転・最小リスク状態への移行を実行している間（Fallback，MRC）は，設定値を変更する要求はE_OBJECT_STATUSエラーとなる。オブジェクト自身の状態が要求を受け付けられないためである。

受け付けられなかった要求は留保されない。アプリケーションは，作動状態および利用可否の変化のイベント（OperatingStateChanged，AvailabilityChanged）により，要求を行える状態になったことを知る。

機能どうしの依存の構造（ある機能がどの機能を用いて実現されているか）は，実装そのものであるため宣言しない。アプリケーションが必要とするのは「いま何が作動を妨げているか」であり，それをunavailableReasonとblockingFunctionで表す。

5.1.1.6. データ型の命名

本オブジェクトに固有のデータ型は，略称を接頭辞とする名称（ADADASXxxType）で定義する。複数のAD/ADAS機能オブジェクトが共通に用いるデータ型は，特定のオブジェクトに属さないため，Automationを接頭辞とする名称（AutomationXxxType）で定義する。

5.1.1.7. ADADASに対する制御の調停

本オブジェクトおよび各機能オブジェクトに対する制御の調停には，標準的な調停機能（後勝ちの原則）が適用される。すなわち，後に実行されたサービスコールによる操作が，先の操作をオーバライドする。

本オブジェクトは，標準的なロック機能を持つLockableObjectである。状態を変化させるサービスコール（acknowledgeTransition）およびロックを取得するサービスコール（lock）は，操作優先度priorityを持つ。ロック中の本オブジェクトに対して，ロックと同じかそれより低い優先度で操作した場合はE_OBJECT_LOCKEDエラーとなり，より高い優先度で操作した場合は，既存のロックが強制解除されて操作が実行される。アプリケーションが使用できる最高値より高い優先度を指定した場合はE_DENIED_PRIORITYエラー，操作優先度が有効範囲外である場合はE_INVALID_PARAMETERエラーとなる。

自アプリケーションのロックが強制解除された場合はOwnLockRevokedイベントが，他のアプリケーション等によりロックが解除された場合はOtherLockReleasedイベントが発生する。本オブジェクトが故障している間は，ロックを取得することができない。

運転者による直接の操作（ステアリングスイッチによる操作，制動操作による解除等）は，本APIを用いない一つのアプリケーションによる操作と見なし，後勝ちの原則に従う。この操作は，アプリケーションが取得したロックおよび操作優先度によって妨げられることはなく，常に有効となる。運転の責任を負うのは運転者であり，アプリケーションによる排他が運転者の操作に優先してはならないためである。



Vehicle.Motionは，アプリケーションが車両の運動そのものを制御するためのオブジェクトであり，本APIの機能オブジェクトは，車両に搭載された運転支援機能に対して目標値を設定するものである。両者は別の経路であり，同時に用いられた場合も後勝ちの原則に従う。AD/ADAS機能の作動中にVehicle.Motionにより同じ方向の運動が制御された場合，当該機能はavailabilityをTemporarilyUnavailable，unavailableReasonをControlOverriddenとして，作動を中断する。ただし，安全のために介入する機能（AEB等）は，この後勝ちの規定の対象とせず，その機能の介入がVehicle.Motionによる制御に優先する。この例外の対象となる機能は，個々の機能オブジェクトの規定で宣言する。

5.1.1.8. 操作・参照に伴うリスク

DDTの移管への応答（acknowledgeTransition）は，運転操作の責任主体を変える操作である。運転者が引き継ぎの準備を終えていない状態で移管が成立すると，走行の安全に影響する。これをリスククラスRiskHandoverとして定義し，acknowledgeTransitionをリスク制御の対象とする。

DDTの責任主体・移管の段階・運行設計領域の充足状態は，運転者がいつ運転していたかを第三者が推定できる情報である。これをリスククラスRiskPrivacyとして定義し，状態を参照するサービスコール（getStatus）をリスク制御の対象とする。イベントの発生の観測からも同じことを推定できるため，イベントの通知要求（notify）およびイベントの取得（getEvent）も同様にRiskPrivacyの対象とする。

いずれのリスククラスについても，当該リスクに対応していないアプリケーションからの呼び出しはE_RISK_APPLICATIONエラー，ユーザが当該リスクを承認していない場合はE_RISK_USERエラーとなる。

構成情報の参照（getConfig）は，車両に搭載されている機能の目録を返すものであり，リスク制御の対象としない。

5.1.2. 機能クラス

機能クラスは，そのAD/ADASシステムが内部に備える性質を表す。

ADADASは，以下の機能クラスを持つ。

OEDRCapable	物体および事象の検知と応答（OEDR）を自ら行う
ODDManaged	運行設計領域の充足状態を自ら判定し，管理する
DriverTransitionCapable	運転者とAD/ADASシステムの間でDDTの移管を自ら管理する
FallbackCapable	縮退運転を自ら実行する
MRCCapable	最小リスク状態への移行を自ら実行する
RemoteSupportCapable	遠隔支援または遠隔介入を利用する
RouteFollowingCapable	指定された経路に沿った走行を行う

機能クラスODDManagedを備える場合は状態oddStatusが，FallbackCapableを備える場合は状態fallbackStateが，MRCCapableを備える場合は状態mrcStateが値を持つ。これらを備えない場合，対応する状態は値なしとなる。機能クラスDriverTransitionCapableを備えない場合，サービスコールacknowledgeTransitionを用いることはできない。

5.1.3. リスククラス

ADADASは，以下のリスククラスを持つ。

RiskHandover	運転操作の責任主体が，運転者の準備が整わないまま切り替わるリスク
RiskPrivacy	走行状態および運転者の運転への関与状態の開示により，個人の行動を推定できることに伴うプライバシのリスク

5.1.4. 構成情報

ADADASは，以下の構成情報を持つ。

capabilities	enumSet (ADADASCapabilityType)	その車両のAD/ADASシステムが持つ機能
riskClasses	enumSet (ADADASRiskType)	車両およびビークルOSで対応できていないリスククラスの集合
functions	ADADASFunctionItemType[]	その車両に搭載されているAD/ADAS機能の目録
highestAutomationLevel	AutomationLevelType	搭載されている機能が設計上対応する自動化レベルのうち最も高いもの
timestampAccuracy	UInt32	この車両のタイムスタンプ誤差上限の宣言。単位はμs

上記は，YAML形式で記述した構成記述ファイル中に記述する形式である。本オブジェクトはシングルインスタンスオブジェクトであるため，構成記述では，車両を表すノードの下に，本オブジェクトに対応するノードADADASを置き，その下に上記の構成情報を記述する。目録が指し示す各機能オブジェクトは，同じADADASノードの下に，それぞれのオブジェクトを表すノード（ACC，LDW等）として置く。実体が複数ある機能のオブジェクトに対しては，オブジェクトを表すノードの下に，インスタンス名の前に$を付けた文字列をノード名とするインスタンスのノードを置き，その構成情報にインスタンスIDを含める。

riskClassesには，車両およびビークルOSで対応できていない（危険を防止できていない）リスククラスの集合を記述する。functionsのprovidingObjectには，その機能を提供するオブジェクトの完全修飾名（例：Vehicle.ADADAS.ACC）を記述する。

構成情報を参照するサービスコールは，この構成情報に対して，次の変換を行った構成情報を返す。

- functionsの各要素のdisplayNameに対しては，それを元に現在のロケールに対応する呼称の文字列を取得し，その文字列を返す。

5.1.5. 状態

ADADASは，以下の状態を持つ。

timestamp	TimestampUsType	状態の測定時刻。単位はμs。車内共通時刻基盤に基づく
dataQualifier	DataQualifierType	このサンプルの品質。InvalidまたはUnavailableの場合，以下の状態の値を読んではならない。値の変化はDataQualifierChangedイベントにより通知される
ddtOwner	ADADASDDTOwnerType	いま動的運転タスク（DDT）を実行している主体。handoverPhaseがTransitioningである間は移管元を指す
handoverPhase	ADADASHandoverPhaseType	DDTの責任の移管の段階

oddStatus	ADADASODDStatusType?	運行設計領域の充足状態。機能クラスODDManagedを備えている場合にのみ値を持つ
fallbackState	ADADASFallbackStateType?	縮退運転の状態。機能クラスFallbackCapableを備えている場合にのみ値を持つ
mrcState	ADADASMRCStateType?	最小リスク状態への移行の状態。機能クラスMRCCapableを備えている場合にのみ値を持つ
lockStatus	LockStatusType	ロック状態

状態は，呼び出し時点のAD/ADASシステム全体の状態を返す冪等なスナップショットである。個々の機能の作動状態・設定値・出力・故障の区分は含まない。それらは，目録のprovidingObjectが指す機能オブジェクトのgetStatusにより取得する。

dataQualifierは，取得できたサンプルの品質を表す。本オブジェクト自体が状態を提供できない場合，getStatusはE_OBJECT_FAULTエラーを返す。これに対しdataQualifierは，出力されたサンプルがどれだけ信用できるかを表す。値が信用できないことはdataQualifierによって表すため，本仕様の各列挙型には，値が信用できないことを表す列挙子（Unknownに相当するもの）を設けない。

5.1.6. サービスコール一覧

ADADASに対するサービスコールは以下の通り。

getConfig	AD/ADAS機能の目録の参照
getStatus	AD/ADASシステム全体の状態の参照
acknowledgeTransition	DDTの責任の移管の要求への応答
lock	ADADASのロックの取得
unlock	ADADASのロックの解除
notify	ADADASイベントの通知要求
unnotify	ADADASイベントの通知停止
getEvent	ADADASイベントの取得

5.1.7. イベント

イベントは，事象が発生したことのみを伝え，状態の値そのものを運ばない。ただし，状態の参照では得られない情報（移管の要求の識別子・理由・猶予時間等）は，イベントの関連情報として渡す。アプリケーションは，イベントを受けた後，必要に応じてgetStatusにより値を取得する。

ADADASに対するイベント種別は以下の通り。

DDTOwnerChanged	DDTを実行している主体の変化
HandoverPhaseChanged	DDTの責任の移管の段階の変化
ODDStatusChanged	運行設計領域の充足状態の変化
TransitionRequested	DDTの責任の移管の要求
FallbackStarted	縮退運転の開始

FallbackCompleted	縮退運転の完了
MRCStarted	最小リスク状態への移行の開始
MRCCompleted	最小リスク状態への移行の完了
DataQualifierChanged	dataQualifierの変化
OwnLockRevoked	自アプリケーションのロックの強制解除
OtherLockReleased	他のアプリケーション等によるロック解除
EventOverflow	イベントキューのオーバフロー
ConfigChanged	構成情報の変更（搭載機能の追加・削除，機能オブジェクトのインスタンスの増減，下位の機能オブジェクトの構成情報の変化）

この内，イベント種別がTransitionRequestedの場合には，以下の関連情報を持つ。

transitionId	IdType	この移管の要求を識別するためのID。acknowledgeTransitionで指定する
targetOwnerKind	ADADASDDTOwnerKindType	移管先の主体の種別
reason	ADADASTransitionReasonType	移管を要求する理由
gracePeriod	UInt32	移管が完了すべき猶予時間。単位はms。猶予時間内に応答が無い場合，車両は縮退運転または最小リスク状態への移行を開始する

また，イベント種別がFallbackStartedの場合には，以下の関連情報を持つ。

reason	ADADASTransitionReasonType	縮退運転を開始した理由

また，イベント種別がEventOverflowの場合には，以下の関連情報を持つ。

noLostEvents	UInt32	失われたイベントの数

5.1.8. データ型の定義

5.1.8.1. ADADASの機能クラス

```
type ADADASCapabilityType = enum {
    OEDRCapable,
    ODDManaged,
    DriverTransitionCapable,
    FallbackCapable,
    MRCCapable,
    RemoteSupportCapable,
    RouteFollowingCapable
};
```

5.1.8.2. ADADASのリスククラス

```
type ADADASRiskType = enum {
    RiskHandover,
    RiskPrivacy
};
```

5.1.8.3. タイムスタンプ

```
type TimestampUsType = UInt64;  // 車内共通時刻基盤の起点を基準とする測定時刻。単位はμs。複数のビークルAPIで共通に用いる共通データ型であり，本仕様内に定義する。
```

5.1.8.4. サンプル品質の種別

```
type DataQualifierType = enum {
    Normal, // 正常
    Unavailable,    // 利用不可
    Degraded,   // 劣化
    TestMode,   // テストモード
    Invalid // 無効（値を読んではならない）
};
```

複数のビークルAPIで共通に用いる共通データ型であり，本仕様内に定義する。

5.1.8.5. AD/ADAS機能の種別

```
type AutomationFunctionType = enum {
    ACC,    // 車間距離制御付定速走行装置
    LKA,    // 車線維持支援
    LDW,    // 車線逸脱警報
    LCA,    // 車線変更支援
    AEB,    // 衝突被害軽減ブレーキ
    ParkingAssist,  // 駐車支援
    AutomatedParking,   // 自動駐車
    TrafficJamAssist,   // 渋滞時運転支援
    HighwayAssist,  // 高速道路運転支援
    NavigationOnAutopilot,  // 経路に沿った自動運転支援
    ADS,    // 自動運転システム
    Other,  // その他の機能
    BSM,    // 死角検知
    FCW,    // 前方衝突警報
    TSR,    // 標識認識
    RCTA    // 後退時交差車両警報
};
```

目録の各要素が，どの種別の機能であるかを表す型である。機能の種別は表示および記録のために用い

る。アプリケーションは，機能に対して行える操作の判定に本型を用いず，目録のprovidingObjectが指す機能オブジェクトの構成情報により判定する。

```
type AutomationLevelType = enum {
    SAE0,   // 運転自動化なし
    SAE1,   // 運転支援
    SAE2,   // 部分運転自動化
    SAE3,   // 条件付運転自動化
    SAE4,   // 高度運転自動化
    SAE5,   // 完全運転自動化
    NotApplicable   // 自動化レベルの区分を適用しない機能
};
```

目録の各要素の機能が設計上対応する自動化レベル，および搭載されている機能のうち最も高い自動化レベル（構成情報highestAutomationLevel）を表す型である。

5.1.8.6. AD/ADAS機能共通の作動状態

以下の5つのデータ型は，AD/ADAS機能オブジェクトが共通に用いるデータ型であり，本仕様内に定義する。

```
type AutomationOperatingStateType = enum {
    Off,    // 機能が無効化されており，作動していない状態
    Standby,    // 機能が有効であり，作動の条件が整えば作動できるが，現在は作動していない状態
    Activating, // 作動開始の要求を受理し，作動開始の処理を行っている過渡状態
    Active, // 作動中。車両の運動への介入または警報の出力を現に行っている状態
    Suspended,  // 作動が一時的に解除され，中断している状態。直前の作動時の設定値を保持しており，抑止する要因が解消すれば作動を再開できる
    Deactivating,   // 作動終了の要求を受理し，作動終了の処理を行っている過渡状態
    Fallback,   // 縮退運転を実行している状態
    MRC,    // 最小リスク状態へ移行し，これを維持している状態
    Fault   // 故障により機能を提供できない状態
};
```

各機能オブジェクトが現にどの作動状態にあるかを表す型である。機能によっては現れない列挙子がある。

```
type AutomationAvailabilityType = enum {
    Available,  // 作動を開始または継続できる
    TemporarilyUnavailable, // 一時的に条件を満たさないため，作動を開始または継続できない（条件が回復すれば作動できる）
    Unavailable // 故障等により，作動を開始または継続できない
};
```

その機能がいま作動を開始または継続できるかを表す型である。作動しているか否かは作動状態operatingStateが表すものであり，本型はこれを表さない。

```
type AutomationUnavailableReasonType = enum {
    ODDOutside, // 走行環境が運行設計領域の外にある
    SensorDegraded, // センサの性能低下・遮蔽により，必要な認識の性能が得られない
    DependentFunctionFault, // この機能が用いている他のAD/ADAS機能が故障している（
blockingFunctionがその機能を指す）
    DriverMonitoringUnavailable,    // 運転者の監視が行えない
    VehicleStateNotReady,   // 車速・変速機の位置・制動等の車両の状態が作動の条件を満
たさない
    FunctionFault,  // この機能自体が故障している
    ControlOverridden,  // 運転者の操作または他の経路による車両の運動の制御により，作
動が抑止されている
    UserDisabled,   // 利用者の設定により機能が無効にされている
    SupersededByOtherFunction,  // 他のAD/ADAS機能がその運転操作を担っているため作動し
ない（blockingFunctionがその機能を指す）
    CalibrationRequired,    // 較正が必要で作動できない
    Other   // その他の理由
};
```

availabilityがAvailable以外の値である場合に，作動を開始または継続できない理由を表す型である。機能によっては現れない列挙子がある。

```
type AutomationOutputLevelType = enum {
    Detected,   // 事象を検知したが，運転者への提示はまだ行っていない
    Advisory,   // 注意喚起（表示等の弱い提示）
    Warning,    // 警報（音・振動を伴う提示）
    Critical    // 緊急（車両運動への介入を伴う）
};
```

機能の出力の段階を表す型である。Detected，Advisory，Warning，Criticalの順に強い段階を表す。機能によっては現れない段階がある。

```
type AutomationFunctionOutputType = struct {
    active : Bool,  // その機能が事象の検知または運転者への提示を現に行っているか
    level : AutomationOutputLevelType?  // 出力の段階。activeがtrueであり，かつ段階を
提供できる機能である場合にのみ値を持つ
};
```

機能が外部（運転者）に対して現に行っている出力を，有無と段階のみで表す型である。出力の判断の根拠（検知した車線の形状・物体の種別等）は本型に含まない。

5.1.8.7. AD/ADAS機能共通の故障の区分

以下のデータ型は，AD/ADAS機能オブジェクトが共通に用いるデータ型であり，本仕様内に定義する。

```
type AutomationFaultCategoryType = enum {
    SensorFault,    // センサの故障
    ActuatorFault,  // アクチュエータの故障
    ComputationFault,   // 演算装置の故障
```

```
    CommunicationFault, // 車内通信の故障
    PowerFault, // 電源の異常
    WarningDeviceFault, // 警報を提示する装置の故障
    Other   // その他の故障
};
```

機能を提供できない故障の区分を表す型である。機能によっては現れない列挙子がある。故障の区分は，各機能オブジェクトの状態faultCategoryおよびFaultDetectedイベントの関連情報により渡される。

5.1.8.8. AD/ADAS機能の目録の要素

```
type ADADASFunctionItemType = struct {
    functionType : AutomationFunctionType,  // 機能の種別
    providingObject : String,   // 機能を提供するオブジェクトの完全修飾名
    automationLevel : AutomationLevelType,  // その機能が設計上対応する自動化レベル
    displayName : String?   // ユーザに表示する呼称。車両が呼称を公開する場合にのみ値
を持つ
};
```

搭載されている1つのAD/ADAS機能を表す型である。機能の設定の値域・作動状態・操作は本型には含まれず，providingObjectが指すオブジェクトから取得する。実体が複数ある機能では，providingObjectが指すオブジェクトがマルチインスタンスオブジェクトとなり，本型の要素は機能につき1つである。

5.1.8.9. DDTの責任

```
type ADADASDDTOwnerKindType = enum {
    Driver, // 運転者
    AutomationSystem,   // AD/ADASシステム（車両に搭載されたAD/ADASの機能が動的運転タ
スクを実行していることを表す）
    RemoteOperator  // 遠隔の操作者
};
```

```
type ADADASDDTOwnerType = struct {
    kind : ADADASDDTOwnerKindType,  // DDTを実行している主体の種別
    function : RelatedObjectType[]  // DDTを実行している機能オブジェクト。kindが
AutomationSystemである場合にのみ要素を持つ
};
```

kindがAutomationSystemである場合にのみfunctionが1つ以上の要素を持ち，それ以外の場合には空の配列となる。functionは，アプリケーションが，いまDDTを実行している機能の詳細を参照するために次に引くオブジェクトを示す。縦方向の運動と横方向の運動を別の機能が担う構成のように，複数の機能が同時にDDTを担う場合には，そのすべてを列挙する。

```
type ADADASHandoverPhaseType = enum {
    Stable, // 責任の所在が安定しており，移管は進行していない
    Shared, // 運転者とAD/ADASシステムが運転の操作を分担している（監視の義務は運転者に
```

```
ある）
    Transitioning   // 移管が進行中である（ddtOwnerは移管元を指す）
};
```

```
type ADADASTransitionReasonType = enum {
    ODDBoundary,    // 運行設計領域の境界に接近している
    ODDExit,    // 運行設計領域を外れた
    FunctionFault,  // 機能が故障した
    SensorDegraded, // センサの検知性能が低下した
    DriverRequest,  // 運転者による要求
    DriverUnresponsive, // 運転者が応答しない
    RemoteOperatorRequest,  // 遠隔の操作者による要求
    RouteCompleted, // 指定された経路の走行が完了した
    Other   // その他の理由
};
```

```
type ADADASTransitionResponseType = enum {
    Accepted,   // 移管の要求を受諾する
    Rejected    // 移管の要求を拒否する
};
```

5.1.8.10. 運行設計領域・縮退運転・最小リスク状態

```
type ADADASODDStatusType = enum {
    NotApplicable,  // 運行設計領域を適用する機能が作動していない
    Satisfied,  // 運行設計領域の内にある
    BoundaryApproaching,    // 運行設計領域の境界に接近している
    NotSatisfied    // 運行設計領域の外にある
};
```

```
type ADADASFallbackStateType = enum {
    NotRequired,    // 縮退運転を要しない
    Requested,  // 縮退運転が要求された
    Preparing,  // 縮退運転の準備中
    Executing,  // 縮退運転の実行中
    Completed,  // 縮退運転が完了した
    Failed  // 縮退運転が失敗した
};
```

```
type ADADASMRCStateType = enum {
    NotRequired,    // 最小リスク状態への移行を要しない
    Preparing,  // 最小リスク状態への移行の準備中
    Executing,  // 最小リスク状態への移行の実行中
    Achieved,   // 最小リスク状態に到達し，これを維持している
```

```
    Failed  // 最小リスク状態への移行が失敗した
};
```

5.1.8.11. ADADASの構成情報（getConfigが返すデータ型）

```
type ADADASConfigType = struct {
    capabilities : enumSet(ADADASCapabilityType),
    riskClasses : enumSet(ADADASRiskType),
    functions : ADADASFunctionItemType[],
    highestAutomationLevel : AutomationLevelType,
    timestampAccuracy : UInt32
};
```

5.1.8.12. ADADAS状態

```
type ADADASStatusType = struct {
    timestamp : TimestampUsType,
    dataQualifier : DataQualifierType,
    ddtOwner : ADADASDDTOwnerType,
    handoverPhase : ADADASHandoverPhaseType,
    oddStatus : ADADASODDStatusType?,
    fallbackState : ADADASFallbackStateType?,
    mrcState : ADADASMRCStateType?,
    lockStatus : LockStatusType
};
```

AD/ADASシステム全体について一つに定まる状態の冪等なスナップショットである。個々の機能の作動状態・設定値・出力は含まない。

5.1.8.13. ADADASイベント

```
type ADADASEventKindType = enum {
    DDTOwnerChanged,
    HandoverPhaseChanged,
    ODDStatusChanged,
    TransitionRequested,
    FallbackStarted,
    FallbackCompleted,
    MRCStarted,
    MRCCompleted,
    DataQualifierChanged,
    OwnLockRevoked,
    OtherLockReleased,
    EventOverflow,
    ConfigChanged
};
```

```
type ADADASEventType = struct {
    eventInfo : variant(ADADASEventKindType) {
        TransitionRequested {
            transitionId : IdType ,
            targetOwnerKind : ADADASDDTOwnerKindType ,
            reason : ADADASTransitionReasonType ,
            gracePeriod : UInt32
        },
        FallbackStarted {
            reason : ADADASTransitionReasonType
        },
        EventOverflow {
            noLostEvents : UInt32
        }
    }
};
```

5.1.9. サービスコールの詳細規定

getConfig

AD/ADAS機能の目録の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
config	ADADASConfigType	AD/ADAS機能の目録とAD/ADASシステム全体の能力の宣言

【エラーコード】

なし

【機能】

その車両に搭載されているAD/ADAS機能の目録（機能の種別，機能を提供するオブジェクトの完全修飾名，設計上対応する自動化レベル）と，AD/ADASシステム全体の能力・リスククラスの宣言を返す。

アプリケーションは，本サービスコールにより，車両に何が搭載されているかと，次にどのオブジェクトを参照すべきかを知る。機能がいま有効か，作動中かは，本サービスコールでは得られない。それらは，providingObjectが指す機能オブジェクトのgetStatusにより取得する。

getStatus

AD/ADASシステム全体の状態の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
status	ADADASStatusType	AD/ADASシステム全体の状態

【エラーコード】

E_OBJECT_FAULT	本オブジェクトが故障しており，状態を提供できない
E_RISK_APPLICATION	アプリケーションがRiskPrivacyに対応していない
E_RISK_USER	ユーザがRiskPrivacyを承認していない

【機能】

AD/ADASシステム全体について一つに定まる状態（DDTを実行している主体，移管の段階，運行設計領域の充足状態，縮退運転の状態，最小リスク状態への移行の状態，ロック状態）を返す。

呼び出しのたびに，その時点の状態の全体を返す冪等な応答である。値を読む前にdataQualifierを確認する。dataQualifierがInvalidまたはUnavailableである場合，各状態の値を読んではならない。

acknowledgeTransition

DDTの責任の移管の要求への応答

【パラメータ】

transitionId	IdType	応答の対象となる移管の要求のID（TransitionRequestedイベントの関連情報で通知されたもの）
response	ADADASTransitionResponseType	移管の要求に対する応答
priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	transitionIdが有効範囲外，または該当する移管の要求が存在しない
E_INVALID_PARAMETER	response，priorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_LOCKED	本オブジェクトが，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている

E_OBJECT_STATUS	該当する移管の要求が既に完了または失効しており，応答を受け付けられない
E_OBJECT_FAULT	本オブジェクトが故障しており，応答を受け付けられない
E_RISK_APPLICATION	アプリケーションがRiskHandoverに対応していない
E_RISK_USER	ユーザがRiskHandoverを承認していない

【機能】

TransitionRequestedイベントで通知されたDDTの責任の移管の要求に対して，応答を通知する。transitionIdには，応答の対象となる移管の要求のIDを指定する。

正常終了は，応答が受理されたことのみを意味し，移管が完了したことを意味しない。移管の完了は，handoverPhaseがStableに戻り，ddtOwnerが移管先を指すことにより確認する。responseにRejectedを指定した場合，車両は縮退運転または最小リスク状態への移行を開始することがある。

本サービスコールは，運転操作の責任主体を変えるものであり，リスククラスRiskHandoverの対象である。また，操作優先度priorityを持ち，ロックによる保護の対象である。

本サービスコールは，機能クラスDriverTransitionCapableを備えている場合にのみ用いることができる。



猶予時間（TransitionRequestedイベントの関連情報gracePeriod）を超えて応答しなかった場合，車両は応答が無かったものとして扱う。その後に同じtransitionIdで応答すると，E_OBJECT_STATUSエラーとなる。

lock

ADADASのロックの取得

【パラメータ】

priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_PARAMETER	priorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_LOCKED	本オブジェクトが，priorityと同じかそれより高い優先度でロックされている
E_OBJECT_FAULT	本オブジェクトが故障しており，ロックを取得できない

【機能】

本オブジェクトを，指定した操作優先度でロックする。ロックを取得したアプリケーションは，状態を変化させるサービスコール（acknowledgeTransition）を，同じかそれより低い優先度の他のアプリケーションによる操作から保護する。

既存のロックより高い優先度でロックを取得した場合，既存のロックは強制解除され，そのロックを取得していたアプリケーションにはOwnLockRevokedイベントが通知される。

運転者による直接の操作は，ロックおよび操作優先度によって妨げられない。

unlock

ADADASのロックの解除

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_OBJECT_STATUS	自アプリケーションが，本オブジェクトをロックしていない

【機能】

自アプリケーションが取得した本オブジェクトのロックを解除する。

notify

ADADASイベントの通知要求

【パラメータ】

eventFilter	enumSet (ADADASEventKindType)?	通知するイベントの種類の集合

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【エラーコード】

E_INVALID_PARAMETER	eventFilterが有効範囲外
E_RISK_APPLICATION	アプリケーションがRiskPrivacyに対応していない
E_RISK_USER	ユーザがRiskPrivacyを承認していない

【機能】

ADADASで発生したeventFilterで指定したイベントの通知を要求する。イベントを通知するためのイベントキューを生成し，その通知ハンドルをnotifyHandleに返す。

eventFilterを省略した場合，すべての種類のイベントの通知を要求する。

イベントの発生の観測からもDDTの責任主体・運行設計領域の充足状態の変化を推定できるため，本サービスコールもRiskPrivacyのリスク制御の対象である。

unnotify

ADADASイベントの通知停止

【パラメータ】

notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正

【機能】

notifyHandleで指定した通知を停止する。指定した通知用のイベントキューは削除される。

getEvent

ADADASイベントの取得

【パラメータ】

notifyHandle	NotifyHandleType	イベントキューを識別するための通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
event	ADADASEventType	取得したイベント

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正
E_NO_DATA	イベントキューが空
E_RISK_APPLICATION	アプリケーションがRiskPrivacyに対応していない
E_RISK_USER	ユーザがRiskPrivacyを承認していない

【機能】

notifyHandleで指定した通知用のイベントキューから，先頭のイベントを取り出す。取り出したイベントは，イベントキューから削除される。

イベントの内容からもDDTの責任主体・運行設計領域の充足状態の変化を推定できるため，本サービスコールもRiskPrivacyのリスク制御の対象である。



イベントがパラメータ等で渡されてくるプログラミング環境向けの物理APIでは，用意する必要がない。

5.2. Vehicle.ADADAS.ACC（略称：ACC）

5.2.1. 位置付けと機能

Vehicle.ADADAS.ACC（略称：ACC）は，車両に搭載された車間距離制御付定速走行装置（ACC）を参照・操作するためのシングルインスタンスオブジェクトである。目標速度と追従時間間隔の設定，作動の開始・終了の要求，および現在の作動状態・制御モード・設定値・車間距離の参照を提供する。

本オブジェクトは，Vehicle.ADADASの目録に，機能の種別ACC，提供するオブジェクトの完全修飾名Vehicle.ADADAS.ACCとして現れる。DDTの責任主体・運行設計領域の充足状態・縮退運転・最小リスク状態など，AD/ADASシステム全体について一つに定まる状態は，本オブジェクトではなくVehicle.ADADASが提供する。

5.2.1.1. ACCの位置付けと性質

ACCは，運転者が設定した目標速度で走行し，先行車を検知した場合には設定された追従時間間隔を保って追従する運転支援の機能である。ACCの作動中も，動的運転タスク（DDT）の責任は運転者にあり，運転者には周囲の監視の義務が残る。

本オブジェクトは，ACCが用いるセンサの構成（カメラ・レーダ等），認識・判断・制御のアルゴリズム，制御装置の構成を規定しない。先行車の検知結果そのもの（相対速度・物体の種別等）は本オブジェクトでは扱わない。

追従の制御の設定値は，車間の時間間隔（単位：s）で表す。同じ時間間隔でも車速により車間の距離は変わるが，運転者が設定するのは追従の余裕であり，時間間隔がこれを直接に表すためである。これに対し，現に保っている車間の距離（単位：m）は，追従の結果として生じる値であり，読み取り専用の状態followingDistanceとして提供する。アプリケーションは車間の距離を設定することができない。

5.2.1.2. 作動状態と操作の意味

本オブジェクトの作動状態operatingStateおよび利用可否availabilityは，Vehicle.ADADASが規定するAD/ADAS機能共通の作動状態モデルに従う。本節では，ACCに固有の制約のみを規定する。

ACCにおける各作動状態の意味は，次のとおりである。

- Off … ACCの主スイッチが入っておらず，作動の要求を受け付けない状態。
- Standby … ACCの主スイッチが入っており，目標速度が設定されれば直ちに車速の制御を開始できるが，現在は車速を制御していない状態。
- Activating … 作動開始の要求（activate）を受理し，Standbyへ遷移する処理を行っている過渡状態。
- Active … 車速の制御を現に行っている状態。制御モードcontrolModeが値を持つ。
- Suspended … 車速の制御が一時的に解除された状態。目標速度と追従時間間隔は保持されており，抑止する要因が解消すれば，復帰の操作（setTargetSpeed，または運転者による復帰の操作）により車速の制御を再開できる。運転者の制動操作等により制御が解除された場合のほか，他のAD/ADAS機能が縦方向の運動を担っている間も，この状態となる。
- Deactivating … 作動終了の要求（deactivate）を受理し，Offへ遷移する処理を行っている過渡状態。

- Fault ... 故障により機能を提供できない状態。故障の区分は状態faultCategoryが表す。

ACCでは，FallbackおよびMRCの列挙子は現れない。縮退運転および最小リスク状態への移行は，DDTの責任を持つ機能が行うものであり，運転支援の機能であるACCはこれを行わないためである。

運転操作の全体を担う他のAD/ADAS機能が縦方向の運動を担っている間，本オブジェクトのoperatingStateはSuspended，availabilityはTemporarilyUnavailable，unavailableReasonはSupersededByOtherFunctionとなり，blockingFunctionがその機能を指す。

作動を開始する要求（activate）は，ACCの主スイッチを入れる操作に相当し，OffからStandbyへの遷移を要求する。目標速度を設定する要求（setTargetSpeed）は，運転者による設定（SET）の操作に相当し，Standbyにおいては目標速度の設定と車速の制御の開始を，Activeにおいては目標速度の変更を，Suspendedにおいては車速の制御の再開を，それぞれ要求する。作動を終了する要求（deactivate）は，主スイッチを切る操作に相当し，Offへの遷移を要求する。

いずれの要求も，正常終了は要求が受理されたことのみを意味し，目的の作動状態への遷移が完了したことを意味しない。遷移の完了は，getStatusまたはOperatingStateChangedイベントにより確認する。

要求を受け付けられない場合の応答は，AD/ADAS機能共通の作動状態モデルの規定による。すなわち，availabilityがTemporarilyUnavailableである場合の作動開始の要求はE_INFEASIBLEエラー，unavailableReasonがFunctionFaultである場合はE_OBJECT_FAULTエラー，それ以外のUnavailableである場合はE_OBJECT_STATUSエラーとなる。また，過渡状態（Activating，Deactivating）にある間の設定値の変更の要求はE_OBJECT_STATUSエラーとなる。

5.2.1.3. 操作の可否の宣言

本オブジェクトに対してアプリケーションが行える操作は，構成情報supportedOperationsが宣言する。supportedOperationsの列挙子は，操作を行うサービスコールと1対1に対応する。アプリケーションは，画面を構成する前にsupportedOperationsを参照し，宣言されていない操作の要素を提示してはならない。

supportedOperationsに含まれない操作のサービスコールを呼び出した場合，E_NOT_SUPPORTEDエラーとなる。supportedOperationsは空集合となることがあり，その場合，本オブジェクトは参照のみが可能な読み取り専用の機能として振る舞う。

supportedOperationsが操作の可否（恒久的な可否）を表すのに対し，availabilityは，いまその操作を受け付けられるか（一時的な可否）を表す。この2つは別の判定であり，アプリケーションは，前者により要素を提示するか否かを，後者により要素を操作可能とするか否かを決める。

5.2.1.4. 設定の値域

目標速度と追従時間間隔の設定できる範囲および刻みは，構成情報targetSpeedRange，followingGapRangeが宣言する。アプリケーションは，これらの範囲外の値を設定してはならない。範囲外の値を指定した場合，E_INVALID_PARAMETERエラーとなる。

5.2.1.5. ACCに対する制御の調停

本オブジェクトに対する制御の調停には，標準的な調停機能（後勝ちの原則）が適用される。すなわち，あるアプリケーションが目標速度を設定した後に，別のアプリケーションが異なる目標速度を設定した場合，後の設定が有効となる。

本オブジェクトは，標準的なロック機能を持つLockableObjectである。状態を変化させるサービスコール

（activate，deactivate，setTargetSpeed，setFollowingGap）およびロックを取得するサービスコール（lock）は，操作優先度priorityを持つ。ロック中の本オブジェクトに対して，ロックと同じかそれより低い優先度で操作した場合はE_OBJECT_LOCKEDエラーとなり，より高い優先度で操作した場合は，既存のロックが強制解除されて操作が実行される。アプリケーションが使用できる最高値より高い優先度を指定した場合はE_DENIED_PRIORITYエラー，操作優先度が有効範囲外である場合はE_INVALID_PARAMETERエラーとなる。

自アプリケーションのロックが強制解除された場合はOwnLockRevokedイベントが，他のアプリケーション等によりロックが解除された場合はOtherLockReleasedイベントが発生する。作動状態がFaultとなった場合，ロックは解除され，その間は新たにロックを取得することができない。

運転者による直接の操作（ステアリングスイッチによる設定，制動操作による解除等）は，本APIを用いない一つのアプリケーションによる操作と見なし，後勝ちの原則に従う。この操作は，アプリケーションが取得したロックおよび操作優先度によって妨げられることはなく，常に有効となる。運転の責任を負うのは運転者であり，アプリケーションによる排他が運転者の操作に優先してはならないためである。



Vehicle.Motionは，アプリケーションが車両の運動そのものを制御するためのオブジェクトであり，本オブジェクトは，車両に搭載されたACCに対して目標値を設定するものである。両者は別の経路であり，同時に用いられた場合も後勝ちの原則に従う。ACCの作動中にVehicle.Motionにより縦方向の運動が制御された場合，本オブジェクトはavailabilityをTemporarilyUnavailable，unavailableReasonをControlOverriddenとし，operatingStateをSuspendedとする。

5.2.1.6. 操作・参照に伴うリスク

ACCの作動の開始・終了，目標速度および追従時間間隔の変更は，車両の縦方向の運動を変えるため，先行車・障害物との衝突のリスクを伴う。これをリスククラスRiskCollisionとして定義し，activate，deactivate，setTargetSpeed，setFollowingGapをリスク制御の対象とする。とくにdeactivateは，車速の制御が失われることにより，運転者が引き継げない場合に衝突のリスクを生じる。

目標速度の設定は，走行中の道路の制限速度を超える速度で走行することにつながりうる。これをリスククラスRiskViolationとして定義し，setTargetSpeedをリスク制御の対象とする。

ACCの作動状態・設定値・車間距離は，運転者がいつどのように運転していたかを第三者が推定できる情報である。これをリスククラスRiskPrivacyとして定義し，状態を参照するサービスコール（getStatus），イベントの通知要求（notify）およびイベントの取得（getEvent）をリスク制御の対象とする。

いずれのリスククラスについても，当該リスクに対応していないアプリケーションからの呼び出しはE_RISK_APPLICATIONエラー，ユーザが当該リスクを承認していない場合はE_RISK_USERエラーとなる。

構成情報の参照（getConfig）は，車両に搭載されたACCの静的な性質を返すものであり，リスク制御の対象としない。

5.2.2. 機能クラス

ACCは，以下の機能クラスを持つ。

SpeedControl	目標速度による定速走行の制御を行うことができる（制御モードConstantSpeed）
VehicleFollowing	先行車に追従し，設定された追従時間間隔を保つ制御を行うことができる（制御モードFollowing）

StopAndGo	先行車の停止に追従して停止し，再発進することができる（制御モードStopped）
ApproachWarning	先行車への接近の警報を出力することができる（状態functionOutput，FunctionOutputChangedイベント）

5.2.3. リスククラス

ACCは，以下のリスククラスを持つ。

RiskCollision	ACCの作動の開始・終了および目標速度・追従時間間隔の変更により，車両の縦方向の運動が変わり，先行車・障害物との衝突を生じるリスク
RiskViolation	目標速度の設定により，走行中の道路の制限速度を超えるなど，交通規則に違反するリスク
RiskPrivacy	走行状態および運転者の運転への関与状態の開示により，個人の行動を推定できることに伴うプライバシのリスク

5.2.4. 構成情報

ACCは，以下の構成情報を持つ。

capabilities	enumSet (ACCCapabilityType)	その車両のACCが持つ機能
riskClasses	enumSet(ACCRiskType)	車両およびビークルOSで対応できていないリスククラスの集合
supportedOperations	enumSet (ACCOperationType)	アプリケーションから要求できる操作の集合。空集合となることがある
displayName	String?	ユーザに表示する呼称を取得するためのキー。車両が呼称を公開する場合にのみ値を持つ
targetSpeedRange	ACCSpeedRangeType?	目標速度として設定できる範囲と刻み。supportedOperationsにSetTargetSpeedを含む場合にのみ値を持つ
followingGapRange	ACCGapRangeType?	追従時間間隔として設定できる範囲と刻み。supportedOperationsにSetFollowingGapを含む場合にのみ値を持つ
stopAndGo	Bool	先行車の停止に追従して停止し，再発進する制御に対応するか。機能クラスStopAndGoを備えている場合にtrueとなる
statusReportInterval	UInt32 ([1,4294967295])?	状態報告イベント（StatusReported）が発生する周期。単位はms。値なしの場合，StatusReportedイベントは発生しない
timestampAccuracy	UInt32	この車両のタイムスタンプ誤差上限の宣言。単位はμs

上記は，YAML形式で記述した構成記述ファイル中に記述する形式である。本オブジェクトはシングルインスタンスオブジェクトであるため，構成記述では，車両を表すノードの下に置かれたADADASノードの下に，オブジェクトを表すノードACCを置き，その下に上記の構成情報を記述する。このノードは，

Vehicle.ADADASの目録において，providingObjectがVehicle.ADADAS.ACCである要素に対応する。

riskClassesには，車両およびビークルOSで対応できていない（危険を防止できていない）リスククラスの集合を記述する。

構成情報を参照するサービスコールは，この構成情報に対して，次の変換を行った構成情報を返す。

- displayNameに対しては，それを元に現在のロケールに対応する呼称の文字列を取得し，その文字列を返す。

5.2.5. 状態

ACCは，以下の状態を持つ。

timestamp	TimestampUsType	状態の測定時刻。単位はμs。車内共通時刻基盤に基づく
dataQualifier	DataQualifierType	このサンプルの品質。InvalidまたはUnavailableの場合，以下の状態の値を読んではならない。値の変化はDataQualifierChangedイベントにより通知される
operatingState	AutomationOperatingStateType	現在の作動状態。ACCではFallbackおよびMRCは現れない
availability	AutomationAvailabilityType	いま作動を開始または継続できるか
unavailableReason	AutomationUnavailableReasonType?	作動を開始または継続できない理由。availabilityがAvailable以外の値である場合にのみ値を持つ
blockingFunction	RelatedObjectType[]	作動を妨げている機能オブジェクト。unavailableReasonがDependentFunctionFaultまたはSupersededByOtherFunctionである場合にのみ要素を持つ
faultCategory	AutomationFaultCategoryType?	故障の区分。operatingStateがFaultである場合にのみ値を持つ
controlMode	ACCControlModeType?	現在の制御モード。operatingStateがActiveである場合にのみ値を持つ
targetSpeed	SpeedType?	現在設定されている目標速度。単位はkm/h。operatingStateがOffまたはFaultである場合には値を持たない
followingGap	ACCFollowingGapType?	現在設定されている追従時間間隔。単位はs。機能クラスVehicleFollowingを備え，かつoperatingStateがOffまたはFaultでない場合にのみ値を持つ
followingDistance	ACCFollowingDistanceType?	現に保っている先行車との車間の距離。単位はm。車両が車間距離を提供でき，かつ先行車に追従している場合（制御モードがFollowingまたはStoppedである場合）にのみ値を持つ

functionOutput	AutomationFunctionOutputType?	先行車への接近の警報の出力の状況。機能クラスApproachWarningを備えている場合にのみ値を持つ
lockStatus	LockStatusType	ロック状態

状態は，呼び出し時点のACCの状態を返す冪等なスナップショットである。値を読む前にdataQualifierを確認する。

ACCが作動しているか否かは，operatingStateにより判定する。availabilityは，いま作動を開始または継続できるかを表すものであり，作動しているか否かを表すものではない。operatingStateとavailabilityの組合せの規則は，Vehicle.ADADASが規定するAD/ADAS機能共通の作動状態モデルによる。

targetSpeedおよびfollowingGapは，設定されている値であり，現在の車速および現在の車間の距離ではない。現在の車速はVehicle.Motionが提供する。followingDistanceは，追従の制御の結果として現に保っている車間の距離であり，設定値ではない。自車の前端から先行車の後端までの距離を表す。アプリケーションからこの値を設定する手段は無い。先行車を検知していない間（制御モードがConstantSpeedである間）は値を持たず，車両が車間の距離を提供できない場合も値を持たない。

故障により機能を提供できない場合，operatingStateはFaultとなり，faultCategoryがその区分を表す。これにより，FaultDetectedイベントを受け取っていないアプリケーションも，状態の参照のみで故障の区分を知ることができる。本オブジェクト自体が状態を提供できない場合は，getStatusがE_OBJECT_FAULTエラーを返す。この場合も，StatusReportedイベントは発生し続け，その関連情報statusのdataQualifierはUnavailableとなる。dataQualifierがUnavailableである場合，dataQualifier以外の値を読んではならない。

5.2.6. サービスコール一覧

ACCに対するサービスコールは以下の通り。

getConfig	ACCの構成情報の参照
getStatus	ACCの状態の参照
activate	ACCの作動の開始の要求
deactivate	ACCの作動の終了の要求
setTargetSpeed	ACCの目標速度の設定
setFollowingGap	ACCの追従時間間隔の設定
lock	ACCのロックの取得
unlock	ACCのロックの解除
notify	ACCイベントの通知要求
unnotify	ACCイベントの通知停止
getEvent	ACCイベントの取得

5.2.7. イベント

イベントは，事象が発生したことのみを伝え，状態の値そのものを運ばない。ただし，事象の発生時点の情報として有用なもの（故障の区分等）は，イベントの関連情報としても渡す。アプリケーションは，イベントを受けた後，必要に応じてgetStatusにより値を取得する。状態の取得は，要求応答（getStatus）

と，各状態項目の変化イベントによる。あわせて，構成情報statusReportIntervalが宣言する周期で，状態の全量を報告するStatusReportedイベントが発生する。

ACCに対するイベント種別は以下の通り。

OperatingStateChanged	作動状態の変化
AvailabilityChanged	作動を開始または継続できるかの変化（作動を妨げている理由の変化を含む）
ControlModeChanged	制御モードの変化（定速走行と追従の切り替わり等）
SettingChanged	目標速度または追従時間間隔の設定値の変化
FunctionOutputChanged	先行車への接近の警報の出力の状況（状態functionOutput）の変化。出力の開始・段階の変化・終了のいずれでも発生する
FaultDetected	故障の検知（ACCの機能を提供できない状態）
FaultRecovered	故障からの回復
DataQualifierChanged	dataQualifierの変化
StatusReported	状態の報告（構成情報statusReportIntervalの周期ごとに発生する）
OwnLockRevoked	自アプリケーションのロックの強制解除
OtherLockReleased	他のアプリケーション等によるロック解除
EventOverflow	イベントキューのオーバフロー
ConfigChanged	構成情報の変更（ソフトウェアの更新による対応する操作・設定の値域の変更）

StatusReportedイベントの関連情報statusは，報告時点のACCの状態の全量であり，getStatusが返す状態と同じ型（ACCStatusType）である。関連情報timestamp（TimestampMsType）と，状態が持つtimestamp（TimestampUsType）とは単位が異なり，起点が同一であることも保証されないため，互いに比較することはできない。状態の更新周期と構成情報statusReportIntervalとは一致するとは限らない。先行車との車間の距離followingDistanceのように，走行に伴って連続的に変化する値は，変化のたびにイベントを発生させると発生頻度が過大になるため，専用の変化イベントを持たない。この種の値の最新の状況は，StatusReportedまたはgetStatusにより取得する。

この内，イベント種別がFaultDetectedの場合には，以下の関連情報を持つ。

faultCategory	AutomationFaultCategoryType	故障の区分

また，イベント種別がStatusReportedの場合には，以下の関連情報を持つ。

timestamp	TimestampMsType	報告が発生した時刻
status	ACCStatusType	報告時点のACCの状態（全量）

また，イベント種別がEventOverflowの場合には，以下の関連情報を持つ。

noLostEvents	UInt32	失われたイベントの数

5.2.8. データ型の定義

5.2.8.1. ACCの機能クラス

```
type ACCCapabilityType = enum {
    SpeedControl,
    VehicleFollowing,
    StopAndGo,
    ApproachWarning
};
```

5.2.8.2. ACCのリスククラス

```
type ACCRiskType = enum {
    RiskCollision,
    RiskViolation,
    RiskPrivacy
};
```

5.2.8.3. タイムスタンプ

```
type TimestampUsType = UInt64;  // 車内共通時刻基盤の起点を基準とする測定時刻。単位はμs。複数のビークルAPIで共通に用いる共通データ型であり，本仕様内に定義する。
```

5.2.8.4. サンプル品質の種別

```
type DataQualifierType = enum {
    Normal, // 正常
    Unavailable,    // 利用不可
    Degraded,   // 劣化
    TestMode,   // テストモード
    Invalid // 無効（値を読んではならない）
};
```

複数のビークルAPIで共通に用いる共通データ型であり，本仕様内に定義する。

5.2.8.5. 速度

```
type SpeedType = Float32;   // 速度。単位はkm/h。正の値は前進，負の値は後退を示す。複数のビークルAPIで共通に用いる共通データ型であり，本仕様内に定義する。
```

5.2.8.6. AD/ADAS機能共通の作動状態

以下の5つのデータ型は，AD/ADAS機能オブジェクトが共通に用いるデータ型であり，本仕様内に定義す

る。ACCにおける各列挙子の具体的な意味は，位置付けと機能の節に示す。

```
type AutomationOperatingStateType = enum {
    Off,    // 機能が無効化されており，作動していない状態
    Standby,    // 機能が有効であり，作動の条件が整えば作動できるが，現在は作動していない状態
    Activating, // 作動開始の要求を受理し，作動開始の処理を行っている過渡状態
    Active, // 作動中。車両の運動への介入または警報の出力を現に行っている状態
    Suspended,  // 作動が一時的に解除され，中断している状態。直前の作動時の設定値を保持しており，抑止する要因が解消すれば作動を再開できる
    Deactivating,   // 作動終了の要求を受理し，作動終了の処理を行っている過渡状態
    Fallback,   // 縮退運転を実行している状態
    MRC,    // 最小リスク状態へ移行し，これを維持している状態
    Fault   // 故障により機能を提供できない状態
};
```

各機能オブジェクトが現にどの作動状態にあるかを表す型である。機能によっては現れない列挙子がある。

```
type AutomationAvailabilityType = enum {
    Available,  // 作動を開始または継続できる
    TemporarilyUnavailable, // 一時的に条件を満たさないため，作動を開始または継続できない（条件が回復すれば作動できる）
    Unavailable // 故障等により，作動を開始または継続できない
};
```

その機能がいま作動を開始または継続できるかを表す型である。作動しているか否かは作動状態operatingStateが表すものであり，本型はこれを表さない。

```
type AutomationUnavailableReasonType = enum {
    ODDOutside, // 走行環境が運行設計領域の外にある
    SensorDegraded, // センサの性能低下・遮蔽により，必要な認識の性能が得られない
    DependentFunctionFault, // この機能が用いている他のAD/ADAS機能が故障している（blockingFunctionがその機能を指す）
    DriverMonitoringUnavailable,    // 運転者の監視が行えない
    VehicleStateNotReady,   // 車速・変速機の位置・制動等の車両の状態が作動の条件を満たさない
    FunctionFault,  // この機能自体が故障している
    ControlOverridden,  // 運転者の操作または他の経路による車両の運動の制御により，作動が抑止されている
    UserDisabled,   // 利用者の設定により機能が無効にされている
    SupersededByOtherFunction,  // 他のAD/ADAS機能がその運転操作を担っているため作動しない（blockingFunctionがその機能を指す）
    CalibrationRequired,    // 較正が必要で作動できない
    Other   // その他の理由
};
```

availabilityがAvailable以外の値である場合に，作動を開始または継続できない理由を表す型である。機

能によっては現れない列挙子がある。

```
type AutomationOutputLevelType = enum {
    Detected,   // 事象を検知したが，運転者への提示はまだ行っていない
    Advisory,   // 注意喚起（表示等の弱い提示）
    Warning,    // 警報（音・振動を伴う提示）
    Critical    // 緊急（車両運動への介入を伴う）
};
```

機能の出力の段階を表す型である。Detected，Advisory，Warning，Criticalの順に強い段階を表す。機能によっては現れない段階がある。

```
type AutomationFunctionOutputType = struct {
    active : Bool,  // その機能が事象の検知または運転者への提示を現に行っているか
    level : AutomationOutputLevelType?  // 出力の段階。activeがtrueであり，かつ段階を
提供できる機能である場合にのみ値を持つ
};
```

機能が外部（運転者）に対して現に行っている出力を，有無と段階のみで表す型である。出力の判断の根拠（検知した車線の形状・物体の種別等）は本型に含まない。

5.2.8.7. AD/ADAS機能共通の故障の区分

以下のデータ型は，AD/ADAS機能オブジェクトが共通に用いるデータ型であり，本仕様内に定義する。ACCでは，警報を提示する装置の故障（WarningDeviceFault）は，機能クラスApproachWarningを備えている場合にのみ現れる。

```
type AutomationFaultCategoryType = enum {
    SensorFault,    // センサの故障
    ActuatorFault,  // アクチュエータの故障
    ComputationFault,   // 演算装置の故障
    CommunicationFault, // 車内通信の故障
    PowerFault, // 電源の異常
    WarningDeviceFault, // 警報を提示する装置の故障
    Other   // その他の故障
};
```

機能を提供できない故障の区分を表す型である。機能によっては現れない列挙子がある。故障の区分は，各機能オブジェクトの状態faultCategoryおよびFaultDetectedイベントの関連情報により渡される。

5.2.8.8. ACCの操作

```
type ACCOperationType = enum {
    Activate,   // 作動の開始の要求（サービスコールactivate）
    Deactivate, // 作動の終了の要求（サービスコールdeactivate）
    SetTargetSpeed, // 目標速度の設定（サービスコールsetTargetSpeed）
    SetFollowingGap // 追従時間間隔の設定（サービスコールsetFollowingGap）
```

```
};
```

アプリケーションから要求できる操作を表す型である。列挙子は，操作を行うサービスコールと1対1に対応する。すべてのアプリケーションが呼び出すことができるサービスコール（getConfig，getStatus，notify，unnotify，getEvent，lock，unlock）は，本型の対象としない。

5.2.8.9. ACCの制御モードと設定の値域

```
type ACCControlModeType = enum {
    ConstantSpeed,  // 先行車を検知しておらず，目標速度による定速走行の制御を行っている
    Following,  // 先行車を検知し，追従時間間隔を保つ制御を行っている
    Stopped // 先行車の停止に追従して停止し，停止を保持している（機能クラスStopAndGoを備えている場合にのみ現れる）
};
```

```
type ACCFollowingGapType = Float32; // 先行車に追従する際の車間の時間間隔。単位はs。
```

```
type ACCFollowingDistanceType = Float32 ([0.0,+INF));
```

先行車との車間の距離を表す型である。単位はm。自車の前端から先行車の後端までの距離とし，車体の長さを含まない。追従の制御の結果として現に保っている距離であり，設定値ではない。

```
type ACCSpeedRangeType = struct {
    minimum : SpeedType,    // 設定できる目標速度の下限。単位はkm/h
    maximum : SpeedType,    // 設定できる目標速度の上限。単位はkm/h
    step : SpeedType    // 設定できる目標速度の刻み。単位はkm/h
};
```

```
type ACCGapRangeType = struct {
    minimum : ACCFollowingGapType,  // 設定できる追従時間間隔の下限。単位はs
    maximum : ACCFollowingGapType,  // 設定できる追従時間間隔の上限。単位はs
    step : ACCFollowingGapType  // 設定できる追従時間間隔の刻み。単位はs
};
```

5.2.8.10. ACCの構成情報（getConfigが返すデータ型）

```
type ACCConfigType = struct {
    capabilities : enumSet(ACCCapabilityType),
    riskClasses : enumSet(ACCRiskType),
    supportedOperations : enumSet(ACCOperationType),
    displayName : String?,
    targetSpeedRange : ACCSpeedRangeType?,
```

```
    followingGapRange : ACCGapRangeType?,
    stopAndGo : Bool,
    statusReportInterval : UInt32 ([1,4294967295])?,
    timestampAccuracy : UInt32
};
```

5.2.8.11. ACC状態

```
type ACCStatusType = struct {
    timestamp : TimestampUsType,
    dataQualifier : DataQualifierType,
    operatingState : AutomationOperatingStateType,
    availability : AutomationAvailabilityType,
    unavailableReason : AutomationUnavailableReasonType?,
    blockingFunction : RelatedObjectType[],
    faultCategory : AutomationFaultCategoryType?,
    controlMode : ACCControlModeType?,
    targetSpeed : SpeedType?,
    followingGap : ACCFollowingGapType?,
    followingDistance : ACCFollowingDistanceType?,
    functionOutput : AutomationFunctionOutputType?,
    lockStatus : LockStatusType
};
```

呼び出し時点のACCの状態の冪等なスナップショットである。

5.2.8.12. ACCイベント

```
type ACCEventKindType = enum {
    OperatingStateChanged,
    AvailabilityChanged,
    ControlModeChanged,
    SettingChanged,
    FunctionOutputChanged,
    FaultDetected,
    FaultRecovered,
    DataQualifierChanged,
    StatusReported,
    OwnLockRevoked,
    OtherLockReleased,
    EventOverflow,
    ConfigChanged
};
```

```
type ACCEventType = struct {
    eventInfo : variant(ACCEventKindType) {
        FaultDetected {
```

```
            faultCategory : AutomationFaultCategoryType
        },
        StatusReported {
            timestamp : TimestampMsType ,
            status : ACCStatusType
        },
        EventOverflow {
            noLostEvents : UInt32
        }
    }
};
```

5.2.9. サービスコールの詳細規定

getConfig

ACCの構成情報の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
config	ACCConfigType	ACCの構成情報

【エラーコード】

なし

【機能】

その車両のACCが持つ機能，アプリケーションから要求できる操作の集合，および設定の値域を返す。

アプリケーションは，本サービスコールにより，画面を構成する前に，どの操作の要素を提示すべきかと，設定の値の範囲・刻みを決める。

getStatus

ACCの状態の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
status	ACCStatusType	ACCの状態

【エラーコード】

E_OBJECT_FAULT	本オブジェクトが故障しており，状態を提供できない
E_RISK_APPLICATION	アプリケーションがRiskPrivacyに対応していない
E_RISK_USER	ユーザがRiskPrivacyを承認していない

【機能】

ACCの現在の作動状態・利用可否・故障の区分・制御モード・設定値・車間の距離・警報の出力の状況・ロック状態を返す。

呼び出しのたびに，その時点の状態の全体を返す冪等な応答である。値を読む前にdataQualifierを確認する。

activate

ACCの作動の開始の要求

【パラメータ】

priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_PARAMETER	priorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_LOCKED	本オブジェクトが，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている
E_OBJECT_STATUS	作動を開始できる作動状態でない（既に作動している，またはavailabilityがUnavailableであり，その理由が故障以外である）
E_OBJECT_FAULT	本オブジェクトが故障しており，要求を受け付けられない（unavailableReasonがFunctionFaultである場合を含む）
E_INFEASIBLE	availabilityがTemporarilyUnavailableであり，作動の条件以外の一時的な理由（故障等）により，現在はACCを有効にできない
E_RISK_APPLICATION	アプリケーションがRiskCollisionに対応していない
E_RISK_USER	ユーザがRiskCollisionを承認していない

【機能】

ACCに作動の開始を要求する。本サービスコールは，ACCの主スイッチを入れる操作に相当し，OffからStandbyへの遷移を要求する。車速の制御は開始されない。車速の制御を開始するには，続けて目標速度を設定する（setTargetSpeed）。

正常終了は要求が受理されたことのみを意味し，作動が開始したことを意味しない。車両は，車速・変速

機の位置・制動の状態・センサの状態・走行の環境等を確認して遷移する。遷移の完了は，getStatusまたはOperatingStateChangedイベントにより確認する。

作動を開始できない場合の応答は，利用可否availabilityとその理由unavailableReasonにより定まる。availabilityがTemporarilyUnavailableである場合はE_INFEASIBLEエラー，Unavailableでありunavailable ReasonがFunctionFaultである場合はE_OBJECT_FAULTエラー，Unavailableでありそれ以外の理由である場合はE_OBJECT_STATUSエラーとなる。

本サービスコールは，構成情報supportedOperationsにActivateを含む場合にのみ用いることができる。

deactivate

ACCの作動の終了の要求

【パラメータ】

priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_PARAMETER	priorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_LOCKED	本オブジェクトが，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている
E_OBJECT_STATUS	作動を終了できる作動状態でない（既にOffである等）
E_OBJECT_FAULT	本オブジェクトが故障しており，要求を受け付けられない
E_RISK_APPLICATION	アプリケーションがRiskCollisionに対応していない
E_RISK_USER	ユーザがRiskCollisionを承認していない

【機能】

ACCに作動の終了を要求する。本サービスコールは，ACCの主スイッチを切る操作に相当し，Offへの遷移を要求する。車速の制御が行われていた場合，制御は終了し，車速の制御は運転者に戻る。

正常終了は要求が受理されたことのみを意味し，作動が終了したことを意味しない。遷移の完了は，getStatusまたはOperatingStateChangedイベントにより確認する。

本サービスコールは，構成情報supportedOperationsにDeactivateを含む場合にのみ用いることができる。



本サービスコールにより車速の制御が終了すると，車速の維持は運転者に委ねられる。運転者が引き継ぐ準備を終えていない状態で作動を終了させると，先行車との衝突のリスクを生じる。

setTargetSpeed

ACCの目標速度の設定

【パラメータ】

targetSpeed	SpeedType	設定する目標速度。単位はkm/h。構成情報targetSpeedRangeの範囲内で，かつその刻みに一致する値でなければならない
priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_PARAMETER	targetSpeedが構成情報targetSpeedRangeの範囲外もしくは刻みに一致しない，またはpriorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_LOCKED	本オブジェクトが，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている
E_OBJECT_STATUS	目標速度を設定できる作動状態でない（OffまたはFaultである，もしくは過渡状態（Activating，Deactivating）にある）
E_OBJECT_FAULT	本オブジェクトが故障しており，要求を受け付けられない
E_INFEASIBLE	現在の車両の状態または走行の環境では，指定された目標速度による制御を実現できない
E_RISK_APPLICATION	アプリケーションがRiskCollisionまたはRiskViolationに対応していない
E_RISK_USER	ユーザがRiskCollisionまたはRiskViolationを承認していない

【機能】

ACCの目標速度を設定する。本サービスコールは，運転者による設定（SET）の操作に相当する。作動状態がStandbyである場合には目標速度の設定と車速の制御の開始を，Activeである場合には目標速度の変更を，Suspendedである場合には車速の制御の再開を，それぞれ要求する。

正常終了は要求が受理されたことのみを意味し，車両が目標速度に到達したことを意味しない。設定値の変化はSettingChangedイベント，作動状態の変化はOperatingStateChangedイベントにより通知される。

過渡状態（Activating，Deactivating）にある間は，E_OBJECT_STATUSエラーとなる。要求は留保されないため，アプリケーションは，OperatingStateChangedイベントにより遷移の完了を確認した上で，あらためて要求する。

本サービスコールは，構成情報supportedOperationsにSetTargetSpeedを含む場合にのみ用いることができる。

setFollowingGap

ACCの追従時間間隔の設定

【パラメータ】

followingGap	ACCFollowingGapType	設定する追従時間間隔。単位はs。構成情報followingGapRangeの範囲内で，かつその刻みに一致する値でなければならない
priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_PARAMETER	followingGapが構成情報followingGapRangeの範囲外もしくは刻みに一致しない，またはpriorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_LOCKED	本オブジェクトが，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている
E_OBJECT_STATUS	追従時間間隔を設定できる作動状態でない（OffまたはFaultである，もしくは過渡状態（Activating，Deactivating）にある）
E_OBJECT_FAULT	本オブジェクトが故障しており，要求を受け付けられない
E_RISK_APPLICATION	アプリケーションがRiskCollisionに対応していない
E_RISK_USER	ユーザがRiskCollisionを承認していない

【機能】

ACCが先行車に追従する際の車間の時間間隔を設定する。設定は，作動状態がOffおよびFaultでなく，かつ過渡状態にない間，いつでも行うことができる。設定した値は，先行車を検知していない間も保持される。

正常終了は要求が受理されたことのみを意味し，車間が設定値に達したことを意味しない。設定値の変化はSettingChangedイベントにより通知される。設定した時間間隔に対して現に保たれている車間の距離は，状態followingDistanceにより参照する。

過渡状態（Activating，Deactivating）にある間は，E_OBJECT_STATUSエラーとなる。要求は留保されないため，アプリケーションは，OperatingStateChangedイベントにより遷移の完了を確認した上で，あらためて要求する。

本サービスコールは，構成情報supportedOperationsにSetFollowingGapを含む場合にのみ用いることができる。

lock

ACCのロックの取得

【パラメータ】

priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_PARAMETER	priorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_LOCKED	本オブジェクトが，priorityと同じかそれより高い優先度でロックされている
E_OBJECT_FAULT	本オブジェクトが故障しており，ロックを取得できない

【機能】

本オブジェクトを，指定した操作優先度でロックする。ロックを取得したアプリケーションは，状態を変化させるサービスコール（activate，deactivate，setTargetSpeed，setFollowingGap）を，同じかそれより低い優先度の他のアプリケーションによる操作から保護する。

既存のロックより高い優先度でロックを取得した場合，既存のロックは強制解除され，そのロックを取得していたアプリケーションにはOwnLockRevokedイベントが通知される。

運転者による直接の操作は，ロックおよび操作優先度によって妨げられない。

unlock

ACCのロックの解除

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_OBJECT_STATUS	自アプリケーションが，本オブジェクトをロックしていない

【機能】

自アプリケーションが取得した本オブジェクトのロックを解除する。

notify

ACCイベントの通知要求

【パラメータ】

eventFilter	enumSet (ACCEventKindType)?	通知するイベントの種類の集合

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【エラーコード】

E_INVALID_PARAMETER	eventFilterが有効範囲外
E_RISK_APPLICATION	アプリケーションがRiskPrivacyに対応していない
E_RISK_USER	ユーザがRiskPrivacyを承認していない

【機能】

ACCで発生したeventFilterで指定したイベントの通知を要求する。イベントを通知するためのイベントキューを生成し，その通知ハンドルをnotifyHandleに返す。

eventFilterを省略した場合，すべての種類のイベントの通知を要求する。

イベントの発生の観測からも運転の状況を推定できるため，本サービスコールもRiskPrivacyのリスク制御の対象である。

unnotify

ACCイベントの通知停止

【パラメータ】

notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正

【機能】

notifyHandleで指定した通知を停止する。指定した通知用のイベントキューは削除される。

getEvent

ACCイベントの取得

【パラメータ】

notifyHandle	NotifyHandleType	イベントキューを識別するための通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
event	ACCEventType	取得したイベント

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正
E_NO_DATA	イベントキューが空
E_RISK_APPLICATION	アプリケーションがRiskPrivacyに対応していない
E_RISK_USER	ユーザがRiskPrivacyを承認していない

【機能】

notifyHandleで指定した通知用のイベントキューから，先頭のイベントを取り出す。取り出したイベントは，イベントキューから削除される。

イベントの内容からも運転の状況を推定できるため，本サービスコールもRiskPrivacyのリスク制御の対象である。



イベントがパラメータ等で渡されてくるプログラミング環境向けの物理APIでは，用意する必要がない。

5.3. Vehicle.ADADAS.LKA（略称：LKA）

5.3.1. 位置付けと機能

Vehicle.ADADAS.LKA（略称：LKA）は，車両に搭載された車線維持支援（LKA）を参照・操作するためのシングルインスタンスオブジェクトである。作動の開始・終了の要求，支援の強さの設定，および現在の作動状態・支援の強さ・操舵の介入の状況の参照を提供する。

本オブジェクトは，Vehicle.ADADASの目録に，機能の種別LKA，提供するオブジェクトの完全修飾名Vehicle.ADADAS.LKAとして現れる。DDTの責任主体・運行設計領域の充足状態・縮退運転・最小リスク状態など，AD/ADASシステム全体について一つに定まる状態は，本オブジェクトではなくVehicle.ADADASが提供する。

5.3.1.1. LKAの位置付けと性質

LKAは，外界のセンサ（カメラ等）により走行車線を認識し，操舵を支援して車線内の走行を維持する，横方向の運動の運転支援の機能である。LKAの作動中も，動的運転タスク（DDT）の責任は運転者にあり，運転者には周囲の監視と操舵の義務が残る。

支援の方式には，車線の中央付近を走行するよう継続的に操舵を支援するものと，車線を逸脱するおそれがある場合にのみ逸脱を防ぐ方向へ操舵に介入するものとがある。その車両のLKAがどの方式を備えるかは，機能クラスLaneCenteringおよびDepartureInterventionが宣言する。LKAは，これらの少なくとも一方を備える。

本オブジェクトは，LKAが用いるセンサの構成，車線の認識と操舵の制御のアルゴリズムを規定しない。検知した車線の形状・位置，自車の車線内の位置，操舵の介入の量（操舵角・操舵トルク）は本オブジェクトでは扱わない。



検知した車線の形状・位置はVehicle.SurroundModelが，車線の逸脱の警報はVehicle.ADADAS.LDWが，現在の操舵の状態はVehicle.Motionが提供する。

5.3.1.2. 作動状態と操作の意味

本オブジェクトの作動状態operatingStateおよび利用可否availabilityは，Vehicle.ADADASが規定するAD/ADAS機能共通の作動状態モデルに従う。本節では，LKAに固有の制約のみを規定する。

LKAにおける各作動状態の意味は，次のとおりである。

- Off … LKAが無効化されており，操舵の支援を行わない状態。アプリケーションによる作動の終了の要求，および運転者の設定のいずれによっても，この状態となる。
- Standby … LKAが有効であるが，車速が支援の対象となる範囲の外にある，車線を認識できない等の理由で，操舵の支援を行っていない状態。
- Activating … 作動開始の要求（activate）を受理し，遷移の処理を行っている過渡状態。
- Active … LKAが有効であり，車線を認識して，車線内の走行を維持するための操舵の支援を行っている状態。機能クラスLaneCenteringを備える場合は継続的な操舵の支援を，DepartureInterventionを備える場合は逸脱のおそれに応じた操舵の介入を，この状態において行う。
- Suspended … 操舵の支援が一時的に解除された状態。運転者による操舵の操作等により支援が抑止されている場合，および他のAD/ADAS機能が横方向の運動を担っている場合に，この状態となる。
- Deactivating … 作動終了の要求（deactivate）を受理し，Offへ遷移する処理を行っている過渡状態。
- Fault … 故障により機能を提供できない状態。故障の区分は状態faultCategoryが表す。

LKAでは，FallbackおよびMRCの列挙子は現れない。縮退運転および最小リスク状態への移行は，DDTの責任を持つ機能が行うものであり，運転支援の機能であるLKAはこれを行わないためである。

LKAが有効である間に車速が構成情報assistSpeedRangeの範囲の外にある場合は，Suspendedではなく，StandbyかつavailabilityがTemporarilyUnavailable，unavailableReasonがVehicleStateNotReadyで表す。

運転者による操舵の操作等により支援が抑止されている間は，operatingStateはSuspended，availabilityはTemporarilyUnavailable，unavailableReasonはControlOverriddenとなる。運転操作の全体を担う他のAD/ADAS機能が横方向の運動を担っている間は，operatingStateはSuspended，availabilityはTemporarilyUnavailable，unavailableReasonはSupersededByOtherFunctionとなり，blockingFunctionがその機能を指す。Suspendedから支援を再開するか否か，およびその時期は，抑止する要因の解消を受けて車両が定める。本オブジェクトは，支援の再開を要求するサービスコールを持たない。

operatingStateがOffである間，availabilityは，LKAを有効にできるか（故障の有無等）のみを表し，有効化した後にのみ意味を持つ作動の条件（車速が構成情報assistSpeedRangeの範囲にあるか，車線を認識できているか等）を反映しない。すなわち，これらの条件を満たさないことのみを理由として，Offである間のavailabilityがTemporarilyUnavailableとなることはない。故障等により有効にできない場合を除き，Offである間のavailabilityはAvailableであり，アプリケーションは作動の開始（activate）を要求できる。作動の条件を満たしているか否かは，有効化した後に，availabilityおよびunavailableReasonにより表される。

運転者の設定によりLKAが無効化されている場合も，operatingStateはOffとなり，availabilityは前述の規定に従う。このときアプリケーションが行う作動の開始の要求は，後勝ちの原則に従い，運転者の設定より後に行われた操作として受理される。

作動を開始する要求（activate）は，LKAを有効にする操作に相当し，Offからの遷移を要求する。要求が受理された後，作動の条件が整っている場合はActiveへ，整っていない場合はStandbyへ遷移する。作動の条件が整っていないことは，作動を開始する要求を受け付けられない理由とはならない。作動を終了する要求（deactivate）は，LKAを無効にする操作に相当し，Offへの遷移を要求する。いずれの要求も，正常終了は要求が受理されたことのみを意味し，目的の作動状態への遷移が完了したことを意味しない。遷移の完了は，getStatusまたはOperatingStateChangedイベントにより確認する。

要求を受け付けられない場合の応答は，AD/ADAS機能共通の作動状態モデルの規定による。すなわち，availabilityがTemporarilyUnavailableである場合の作動開始の要求はE_INFEASIBLEエラー，unavailableReasonがFunctionFaultである場合はE_OBJECT_FAULTエラー，それ以外のUnavailableである場合はE_OBJECT_STATUSエラーとなる。また，過渡状態（Activating，Deactivating）にある間の設定値の変更の要求はE_OBJECT_STATUSエラーとなる。

5.3.1.3. 操作の可否の宣言

本オブジェクトに対してアプリケーションが行える操作は，構成情報supportedOperationsが宣言する。supportedOperationsの列挙子は，操作を行うサービスコールと1対1に対応する。アプリケーションは，画面を構成する前にsupportedOperationsを参照し，宣言されていない操作の要素を提示してはならない。

supportedOperationsに含まれない操作のサービスコールを呼び出した場合，E_NOT_SUPPORTEDエラーとなる。supportedOperationsは空集合となることがあり，その場合，本オブジェクトは参照のみが可能な読み取り専用の機能として振る舞う。車両が，アプリケーションからの作動の開始・終了そのものを認めない場合は，supportedOperationsにActivateおよびDeactivateを含めないことで表す。

supportedOperationsが操作の可否（恒久的な可否）を表すのに対し，availabilityは，いまその操作を受け付けられるか（一時的な可否）を表す。この2つは別の判定であり，アプリケーションは，前者により要素を提示するか否かを，後者により要素を操作可能とするか否かを決める。

5.3.1.4. 支援の強さ

機能クラスAssistLevelAdjustableを備えるLKAは，操舵の支援の強さを段階で切り替えることができる。支援の強さは，1から構成情報assistLevelCountまでの整数の段階で表し，値が大きいほど強い支援を表す。各段階における操舵の支援の具体的な強さ（介入を開始する時期・操舵の力の大きさ等）は，車両が定める。

現在の支援の強さは状態assistLevelが表す。支援の強さは，LKAの作動状態によらず保持される設定であり，作動状態がOffである間も設定することができる。

AssistLevelAdjustableを備え，かつ構成情報supportedOperationsにSetAssistLevelを含む場合，アプリケーションはsetAssistLevelにより支援の強さを設定できる。assistLevelCountを超える値を指定した場合，E_INVALID_PARAMETERエラーとなる。

5.3.1.5. 操舵の介入の出力

機能クラスInterventionReportedを備えるLKAは，車線の逸脱を防ぐための操舵の介入を現に行っているかを，状態functionOutputにより提供する。activeは，逸脱のおそれに応じた操舵の介入を現に行っているかを表す。継続的な操舵の支援（機能クラスLaneCentering）そのものは介入として扱わず，operatingStateがActiveであることにより表す。

LKAは車両の運動への介入を出力とするため，levelは，activeがtrueである間Criticalとなる。LKAでは，Detected，AdvisoryおよびWarningは現れない。車線の逸脱の警報は，本オブジェクトではなくVehicle.ADADAS.LDWが提供する。

InterventionReportedは，機能クラスDepartureInterventionを備える場合にのみ備えることができる。functionOutputの値が変化したときは，FunctionOutputChangedイベントが発生する。

5.3.1.6. LKAに対する制御の調停

本オブジェクトに対する制御の調停には，標準的な調停機能（後勝ちの原則）が適用される。すなわち，あるアプリケーションが作動を開始した後に，別のアプリケーションが作動の終了を要求した場合，後の要求が有効となる。

本オブジェクトは，標準的なロック機能を持つLockableObjectである。状態を変化させるサービスコール（activate，deactivate，setAssistLevel）およびロックを取得するサービスコール（lock）は，操作優先度priorityを持つ。ロック中の本オブジェクトに対して，ロックと同じかそれより低い優先度で操作した場合はE_OBJECT_LOCKEDエラーとなり，より高い優先度で操作した場合は，既存のロックが強制解除されて操作が実行される。アプリケーションが使用できる最高値より高い優先度を指定した場合はE_DENIED_PRIORITYエラー，操作優先度が有効範囲外である場合はE_INVALID_PARAMETERエラーとなる。

自アプリケーションのロックが強制解除された場合はOwnLockRevokedイベントが，他のアプリケーション等によりロックが解除された場合はOtherLockReleasedイベントが発生する。作動状態がFaultとなった場合，ロックは解除され，その間は新たにロックを取得することができない。

運転者による直接の操作（ステアリングスイッチ・車載の設定画面等による有効化・無効化，操舵による支援の解除等）は，本APIを用いない一つのアプリケーションによる操作と見なし，後勝ちの原則に従う。この操作は，アプリケーションが取得したロックおよび操作優先度によって妨げられることはなく，常に有効となる。運転の責任を負うのは運転者であり，アプリケーションによる排他が運転者の操作に優先してはならないためである。



Vehicle.Motionは，アプリケーションが車両の運動そのものを制御するためのオブジェクトであり，本オブジェクトは，車両に搭載されたLKAの作動と支援の強さを設定するものである。両者は別の経路であり，同時に用いられた場合も後勝ちの原則に従う。LKAの作動中にVehicle.Motionにより横方向の運動が制御された場合，本オブジェクトはavailabilityをTemporarilyUnavailable，unavailableReasonをControlOverriddenとし，operatingStateをSuspendedとする。

5.3.1.7. 操作・参照に伴うリスク

LKAの作動の開始・終了および支援の強さの変更は，車両の横方向の運動を変えるため，車線の逸脱による他車・障害物との衝突のリスクを伴う。これをリスククラスRiskCollisionとして定義し，activate，deactivate，setAssistLevelをリスク制御の対象とする。とくにdeactivateおよび支援を弱める設定は，操舵の支援が失われることにより，運転者が引き継げない場合に衝突のリスクを生じる。

LKAの作動状態・支援の強さ・操舵の介入の履歴は，運転者がいつどのように運転していたかを第三者が推定できる情報である。これをリスククラスRiskPrivacyとして定義し，状態を参照するサービスコール（getStatus），イベントの通知要求（notify）およびイベントの取得（getEvent）をリスク制御の対象とする。

いずれのリスククラスについても，当該リスクに対応していないアプリケーションからの呼び出しはE_RISK_APPLICATIONエラー，ユーザが当該リスクを承認していない場合はE_RISK_USERエラーとなる。

構成情報の参照（getConfig）は，車両に搭載されたLKAの静的な性質を返すものであり，リスク制御の対象としない。

5.3.2. 機能クラス

LKAは，以下の機能クラスを持つ。

LaneCentering	作動状態がActiveである間，車線の中央付近を走行するよう，継続的に操舵を支援する
DepartureIntervention	作動状態がActiveである間，車線を逸脱するおそれがある場合に，逸脱を防ぐ方向へ操舵に介入する
AssistLevelAdjustable	支援の強さを段階で切り替えることができる（構成情報assistLevelCount，状態assistLevel）
InterventionReported	逸脱を防ぐための操舵の介入を現に行っているかを提供する（状態functionOutput，FunctionOutputChangedイベント）。DepartureInterventionを備える場合にのみ備えることができる

5.3.3. リスククラス

LKAは，以下のリスククラスを持つ。

RiskCollision	LKAの作動の開始・終了および支援の強さの変更により，車両の横方向の運動が変わり，車線の逸脱による他車・障害物との衝突を生じるリスク
RiskPrivacy	走行状態および運転者の運転への関与状態の開示により，個人の行動を推定できることに伴うプライバシのリスク

5.3.4. 構成情報

LKAは，以下の構成情報を持つ。

capabilities	enumSet (LKACapabilityType)	その車両のLKAが持つ機能
riskClasses	enumSet(LKARiskType)	車両およびビークルOSで対応できていないリスククラスの集合
supportedOperations	enumSet (LKAOperationType)	アプリケーションから要求できる操作の集合。空集合となることがある
displayName	String?	ユーザに表示する呼称を取得するためのキー。車両が呼称を公開する場合にのみ値を持つ
assistSpeedRange	LKASpeedRangeType	操舵の支援の対象となる車速の範囲。この範囲の外では，作動状態はStandbyとなる
assistLevelCount	LKAAssistLevelType?	設定できる支援の強さの段階の数（最も強い段階の値）。機能クラスAssistLevelAdjustableを備えている場合にのみ値を持つ
timestampAccuracy	UInt32	この車両のタイムスタンプ誤差上限の宣言。単位はμs

上記は，YAML形式で記述した構成記述ファイル中に記述する形式である。本オブジェクトはシングルイ

ンスタンスオブジェクトであるため，構成記述では，車両を表すノードの下に置かれたADADASノードの下に，オブジェクトを表すノードLKAを置き，その下に上記の構成情報を記述する。このノードは，Vehicle.ADADASの目録において，providingObjectがVehicle.ADADAS.LKAである要素に対応する。

riskClassesには，車両およびビークルOSで対応できていない（危険を防止できていない）リスククラスの集合を記述する。assistLevelCountは2以上とする。

構成情報を参照するサービスコールは，この構成情報に対して，次の変換を行った構成情報を返す。

- displayNameに対しては，それを元に現在のロケールに対応する呼称の文字列を取得し，その文字列を返す。

5.3.5. 状態

LKAは，以下の状態を持つ。

timestamp	TimestampUsType	状態の測定時刻。単位はμs。車内共通時刻基盤に基づく
dataQualifier	DataQualifierType	このサンプルの品質。InvalidまたはUnavailableの場合，以下の状態の値を読んではならない。値の変化はDataQualifierChangedイベントにより通知される
operatingState	AutomationOperatingStateType	現在の作動状態。LKAではFallbackおよびMRCは現れない
availability	AutomationAvailabilityType	いま操舵の支援を開始または継続できるか
unavailableReason	AutomationUnavailableReasonType?	操舵の支援を開始または継続できない理由。availabilityがAvailable以外の値である場合にのみ値を持つ
blockingFunction	RelatedObjectType[]	操舵の支援を妨げている機能オブジェクト。unavailableReasonがDependentFunctionFaultまたはSupersededByOtherFunctionである場合にのみ要素を持つ
faultCategory	AutomationFaultCategoryType?	故障の区分。operatingStateがFaultである場合にのみ値を持つ
assistLevel	LKAAssistLevelType?	現在設定されている支援の強さ。機能クラスAssistLevelAdjustableを備え，かつoperatingStateがFaultでない場合にのみ値を持つ
functionOutput	AutomationFunctionOutputType?	逸脱を防ぐための操舵の介入の状況。機能クラスInterventionReportedを備えている場合にのみ値を持つ
lockStatus	LockStatusType	ロック状態

状態は，呼び出し時点のLKAの状態を返す冪等なスナップショットである。値を読む前にdataQualifierを確認する。

LKAが有効であるか否かは，operatingStateがOffであるか否かにより判定する。LKAが操舵の支援を現に行っているか否かは，operatingStateがActiveであるか否かにより判定する。availabilityは，いま操舵の

支援を開始または継続できるかを表すものであり，支援を行っているか否かを表すものではない。operatingStateとavailabilityの組合せの規則は，Vehicle.ADADASが規定するAD/ADAS機能共通の作動状態モデルによる。

operatingStateがActive以外である間，functionOutputのactiveは常にfalseである。

故障により機能を提供できない場合，operatingStateはFaultとなり，faultCategoryがその区分を表す。これにより，FaultDetectedイベントを受け取っていないアプリケーションも，状態の参照のみで故障の区分を知ることができる。本オブジェクト自体が状態を提供できない場合は，getStatusがE_OBJECT_FAULTエラーを返す。

5.3.6. サービスコール一覧

LKAに対するサービスコールは以下の通り。

getConfig	LKAの構成情報の参照
getStatus	LKAの状態の参照
activate	LKAの作動の開始の要求
deactivate	LKAの作動の終了の要求
setAssistLevel	LKAの支援の強さの設定
lock	LKAのロックの取得
unlock	LKAのロックの解除
notify	LKAイベントの通知要求
unnotify	LKAイベントの通知停止
getEvent	LKAイベントの取得

5.3.7. イベント

イベントは，事象が発生したことのみを伝え，状態の値そのものを運ばない。ただし，事象の発生時点の情報として有用なもの（故障の区分等）は，イベントの関連情報としても渡す。アプリケーションは，イベントを受けた後，必要に応じてgetStatusにより値を取得する。

状態の取得は，要求応答（getStatus）と，各状態項目の変化イベントによる。

LKAに対するイベント種別は以下の通り。

OperatingStateChanged	作動状態の変化（アプリケーションの要求および運転者の設定・操作による変化を含む）
AvailabilityChanged	操舵の支援を開始または継続できるかの変化（支援を妨げている理由の変化を含む）
SettingChanged	支援の強さの設定値の変化
FunctionOutputChanged	逸脱を防ぐための操舵の介入の状況（状態functionOutput）の変化。介入の開始・終了のいずれでも発生する
FaultDetected	故障の検知（LKAの機能を提供できない状態）
FaultRecovered	故障からの回復

DataQualifierChanged	状態の品質（状態dataQualifier）の変化
OwnLockRevoked	自アプリケーションのロックの強制解除
OtherLockReleased	他のアプリケーション等によるロック解除
EventOverflow	イベントキューのオーバフロー
ConfigChanged	構成情報の変更（ソフトウェアの更新による対応する操作・支援の対象となる車速の範囲・支援の強さの段階の数の変更）

この内，イベント種別がFaultDetectedの場合には，以下の関連情報を持つ。

faultCategory	AutomationFaultCategoryType	故障の区分

また，イベント種別がEventOverflowの場合には，以下の関連情報を持つ。

noLostEvents	UInt32	失われたイベントの数

5.3.8. データ型の定義

5.3.8.1. LKAの機能クラス

```
type LKACapabilityType = enum {
    LaneCentering,
    DepartureIntervention,
    AssistLevelAdjustable,
    InterventionReported
};
```

5.3.8.2. LKAのリスククラス

```
type LKARiskType = enum {
    RiskCollision,
    RiskPrivacy
};
```

5.3.8.3. タイムスタンプ

```
type TimestampUsType = UInt64;  // 車内共通時刻基盤の起点を基準とする測定時刻。単位はμs。複数のビークルAPIで共通に用いる共通データ型であり，本仕様内に定義する。
```

5.3.8.4. サンプル品質の種別

```
type DataQualifierType = enum {
    Normal, // 正常
    Unavailable,    // 利用不可
```

```
    Degraded,   // 劣化
    TestMode,   // テストモード
    Invalid // 無効（値を読んではならない）
};
```

複数のビークルAPIで共通に用いる共通データ型であり，本仕様内に定義する。

5.3.8.5. 速度

```
type SpeedType = Float32;   // 速度。単位はkm/h。正の値は前進，負の値は後退を示す。複数のビークルAPIで共通に用いる共通データ型であり，本仕様内に定義する。
```

5.3.8.6. AD/ADAS機能共通の作動状態

以下の5つのデータ型は，AD/ADAS機能オブジェクトが共通に用いるデータ型であり，本仕様内に定義する。LKAにおける各列挙子の具体的な意味は，位置付けと機能の節に示す。

```
type AutomationOperatingStateType = enum {
    Off,    // 機能が無効化されており，作動していない状態
    Standby,    // 機能が有効であり，作動の条件が整えば作動できるが，現在は作動していない状態
    Activating, // 作動開始の要求を受理し，作動開始の処理を行っている過渡状態
    Active, // 作動中。車両の運動への介入または警報の出力を現に行っている状態
    Suspended,  // 作動が一時的に解除され，中断している状態。直前の作動時の設定値を保持しており，抑止する要因が解消すれば作動を再開できる
    Deactivating,   // 作動終了の要求を受理し，作動終了の処理を行っている過渡状態
    Fallback,   // 縮退運転を実行している状態
    MRC,    // 最小リスク状態へ移行し，これを維持している状態
    Fault   // 故障により機能を提供できない状態
};
```

各機能オブジェクトが現にどの作動状態にあるかを表す型である。機能によっては現れない列挙子がある。

```
type AutomationAvailabilityType = enum {
    Available,  // 作動を開始または継続できる
    TemporarilyUnavailable, // 一時的に条件を満たさないため，作動を開始または継続できない（条件が回復すれば作動できる）
    Unavailable // 故障等により，作動を開始または継続できない
};
```

その機能がいま作動を開始または継続できるかを表す型である。作動しているか否かは作動状態operatingStateが表すものであり，本型はこれを表さない。

```
type AutomationUnavailableReasonType = enum {
    ODDOutside, // 走行環境が運行設計領域の外にある
    SensorDegraded, // センサの性能低下・遮蔽により，必要な認識の性能が得られない
```

```
    DependentFunctionFault, // この機能が用いている他のAD/ADAS機能が故障している（
blockingFunctionがその機能を指す）
    DriverMonitoringUnavailable,    // 運転者の監視が行えない
    VehicleStateNotReady,   // 車速・変速機の位置・制動等の車両の状態が作動の条件を満
たさない
    FunctionFault,  // この機能自体が故障している
    ControlOverridden,  // 運転者の操作または他の経路による車両の運動の制御により，作
動が抑止されている
    UserDisabled,   // 利用者の設定により機能が無効にされている
    SupersededByOtherFunction,  // 他のAD/ADAS機能がその運転操作を担っているため作動し
ない（blockingFunctionがその機能を指す）
    CalibrationRequired,    // 較正が必要で作動できない
    Other   // その他の理由
};
```

availabilityがAvailable以外の値である場合に，作動を開始または継続できない理由を表す型である。機能によっては現れない列挙子がある。

```
type AutomationOutputLevelType = enum {
    Detected,   // 事象を検知したが，運転者への提示はまだ行っていない
    Advisory,   // 注意喚起（表示等の弱い提示）
    Warning,    // 警報（音・振動を伴う提示）
    Critical    // 緊急（車両運動への介入を伴う）
};
```

機能の出力の段階を表す型である。Detected，Advisory，Warning，Criticalの順に強い段階を表す。機能によっては現れない段階がある。

```
type AutomationFunctionOutputType = struct {
    active : Bool,  // その機能が事象の検知または運転者への提示を現に行っているか
    level : AutomationOutputLevelType?  // 出力の段階。activeがtrueであり，かつ段階を
提供できる機能である場合にのみ値を持つ
};
```

機能が外部（運転者）に対して現に行っている出力を，有無と段階のみで表す型である。出力の判断の根拠（検知した車線の形状・物体の種別等）は本型に含まない。

5.3.8.7. AD/ADAS機能共通の故障の区分

以下のデータ型は，AD/ADAS機能オブジェクトが共通に用いるデータ型であり，本仕様内に定義する。LKAは警報の出力を行わないため，警報を提示する装置の故障（WarningDeviceFault）は現れない。

```
type AutomationFaultCategoryType = enum {
    SensorFault,    // センサの故障
    ActuatorFault,  // アクチュエータの故障
    ComputationFault,   // 演算装置の故障
    CommunicationFault, // 車内通信の故障
    PowerFault, // 電源の異常
```

```
    WarningDeviceFault, // 警報を提示する装置の故障
    Other   // その他の故障
};
```

機能を提供できない故障の区分を表す型である。機能によっては現れない列挙子がある。故障の区分は，各機能オブジェクトの状態faultCategoryおよびFaultDetectedイベントの関連情報により渡される。

5.3.8.8. LKAの操作

```
type LKAOperationType = enum {
    Activate,   // 作動の開始の要求（サービスコールactivate）
    Deactivate, // 作動の終了の要求（サービスコールdeactivate）
    SetAssistLevel  // 支援の強さの設定（サービスコールsetAssistLevel）
};
```

アプリケーションから要求できる操作を表す型である。列挙子は，操作を行うサービスコールと1対1に対応する。すべてのアプリケーションが呼び出すことができるサービスコール（getConfig，getStatus，notify，unnotify，getEvent，lock，unlock）は，本型の対象としない。

5.3.8.9. 支援の強さと支援の対象となる車速の範囲

```
type LKAAssistLevelType = UInt8 ([1,255]);
```

操舵の支援の強さの段階を表す型である。1が最も弱い段階であり，値が大きいほど強い支援を表す。各段階における具体的な支援の強さは車両が定める。

```
type LKASpeedRangeType = struct {
    minimum : SpeedType,    // 操舵の支援の対象となる車速の下限。単位はkm/h
    maximum : SpeedType?    // 操舵の支援の対象となる車速の上限。単位はkm/h。上限を設
けない場合は値を持たない
};
```

5.3.8.10. LKAの構成情報（getConfigが返すデータ型）

```
type LKAConfigType = struct {
    capabilities : enumSet(LKACapabilityType),
    riskClasses : enumSet(LKARiskType),
    supportedOperations : enumSet(LKAOperationType),
    displayName : String?,
    assistSpeedRange : LKASpeedRangeType,
    assistLevelCount : LKAAssistLevelType?,
    timestampAccuracy : UInt32
};
```

5.3.8.11. LKA状態

```
type LKAStatusType = struct {
    timestamp : TimestampUsType,
    dataQualifier : DataQualifierType,
    operatingState : AutomationOperatingStateType,
    availability : AutomationAvailabilityType,
    unavailableReason : AutomationUnavailableReasonType?,
    blockingFunction : RelatedObjectType[],
    faultCategory : AutomationFaultCategoryType?,
    assistLevel : LKAAssistLevelType?,
    functionOutput : AutomationFunctionOutputType?,
    lockStatus : LockStatusType
};
```

呼び出し時点のLKAの状態の冪等なスナップショットである。

5.3.8.12. LKAイベント

```
type LKAEventKindType = enum {
    OperatingStateChanged,
    AvailabilityChanged,
    SettingChanged,
    FunctionOutputChanged,
    FaultDetected,
    FaultRecovered,
    DataQualifierChanged,
    OwnLockRevoked,
    OtherLockReleased,
    EventOverflow,
    ConfigChanged
};
```

```
type LKAEventType = struct {
    eventInfo : variant(LKAEventKindType) {
        FaultDetected {
            faultCategory : AutomationFaultCategoryType
        },
        EventOverflow {
            noLostEvents : UInt32
        }
    }
};
```

5.3.9. サービスコールの詳細規定

getConfig

LKAの構成情報の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
config	LKAConfigType	LKAの構成情報

【エラーコード】

なし

【機能】

その車両のLKAが持つ機能，アプリケーションから要求できる操作の集合，支援の対象となる車速の範囲，および設定できる支援の強さの段階の数を返す。

アプリケーションは，本サービスコールにより，画面を構成する前に，どの操作の要素を提示すべきかと，支援の強さの設定の範囲を決める。

getStatus

LKAの状態の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
status	LKAStatusType	LKAの状態

【エラーコード】

E_OBJECT_FAULT	本オブジェクトが故障しており，状態を提供できない
E_RISK_APPLICATION	アプリケーションがRiskPrivacyに対応していない
E_RISK_USER	ユーザがRiskPrivacyを承認していない

【機能】

LKAの現在の作動状態・利用可否・故障の区分・支援の強さ・操舵の介入の状況・ロック状態を返す。

呼び出しのたびに，その時点の状態の全体を返す冪等な応答である。値を読む前にdataQualifierを確認する。

activate

LKAの作動の開始の要求

【パラメータ】

priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_PARAMETER	priorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_LOCKED	本オブジェクトが，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている
E_OBJECT_STATUS	作動を開始できる作動状態でない（既に有効である，またはavailabilityがUnavailableであり，その理由が故障以外である）
E_OBJECT_FAULT	本オブジェクトが故障しており，要求を受け付けられない（unavailableReasonがFunctionFaultである場合を含む）
E_INFEASIBLE	availabilityがTemporarilyUnavailableであり，作動の条件以外の一時的な理由（故障等）により，現在はLKAを有効にできない
E_RISK_APPLICATION	アプリケーションがRiskCollisionに対応していない
E_RISK_USER	ユーザがRiskCollisionを承認していない

【機能】

LKAに作動の開始を要求する。本サービスコールは，LKAを有効にする操作に相当し，Offからの遷移を要求する。作動の条件（車速が支援の対象となる範囲にあり，車線を認識できている等）が整っている場合はActiveへ，整っていない場合はStandbyへ遷移する。作動の条件が整っていないことは，本要求を受け付けられない理由とはならない。Offである間のavailabilityは，これらの作動の条件を反映しない。

正常終了は要求が受理されたことのみを意味し，操舵の支援が開始したことを意味しない。遷移の完了は，getStatusまたはOperatingStateChangedイベントにより確認する。

作動を開始できない場合の応答は，利用可否availabilityとその理由unavailableReasonにより定まる。availabilityがTemporarilyUnavailableである場合はE_INFEASIBLEエラー，Unavailableでありunavailable ReasonがFunctionFaultである場合はE_OBJECT_FAULTエラー，Unavailableでありそれ以外の理由である場合はE_OBJECT_STATUSエラーとなる。

本サービスコールは，構成情報supportedOperationsにActivateを含む場合にのみ用いることができる。

deactivate

LKAの作動の終了の要求

【パラメータ】

priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_PARAMETER	priorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_LOCKED	本オブジェクトが，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている
E_OBJECT_STATUS	作動を終了できる作動状態でない（既にOffである等）
E_OBJECT_FAULT	本オブジェクトが故障しており，要求を受け付けられない
E_RISK_APPLICATION	アプリケーションがRiskCollisionに対応していない
E_RISK_USER	ユーザがRiskCollisionを承認していない

【機能】

LKAに作動の終了を要求する。本サービスコールは，LKAを無効にする操作に相当し，Offへの遷移を要求する。操舵の支援が行われていた場合，支援は終了し，車線内の走行の維持は運転者に委ねられる。

正常終了は要求が受理されたことのみを意味し，作動が終了したことを意味しない。遷移の完了は，getStatusまたはOperatingStateChangedイベントにより確認する。

本サービスコールは，構成情報supportedOperationsにDeactivateを含む場合にのみ用いることができる。



本サービスコールにより操舵の支援が終了すると，車線内の走行の維持は運転者に委ねられる。運転者が引き継ぐ準備を終えていない状態で作動を終了させると，車線の逸脱による衝突のリスクを生じる。アプリケーションは，無効にしたことを運転者が認識できるようにしなければならない。

setAssistLevel

LKAの支援の強さの設定

【パラメータ】

assistLevel	LKAAssistLevelType	設定する支援の強さ。1以上，構成情報assistLevelCount以下の値でなければならない
priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_PARAMETER	assistLevelが構成情報assistLevelCountを超える，またはpriorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_LOCKED	本オブジェクトが，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている
E_OBJECT_STATUS	支援の強さを設定できる作動状態でない（過渡状態（Activating，Deactivating）にある）
E_OBJECT_FAULT	本オブジェクトが故障しており，要求を受け付けられない
E_RISK_APPLICATION	アプリケーションがRiskCollisionに対応していない
E_RISK_USER	ユーザがRiskCollisionを承認していない

【機能】

LKAの操舵の支援の強さを設定する。設定は，作動状態がFaultでなく，かつ過渡状態にない間，いつでも行うことができる。作動状態がActiveである間に設定した場合，設定した強さによる支援が直ちに適用される。

正常終了は要求が受理されたことのみを意味する。設定値の変化はSettingChangedイベントにより通知される。

過渡状態（Activating，Deactivating）にある間は，E_OBJECT_STATUSエラーとなる。要求は留保されないため，アプリケーションは，OperatingStateChangedイベントにより遷移の完了を確認した上で，あらためて要求する。

本サービスコールは，機能クラスAssistLevelAdjustableを備え，かつ構成情報supportedOperationsにSetAssistLevelを含む場合にのみ用いることができる。

lock

LKAのロックの取得

【パラメータ】

priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_PARAMETER	priorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_LOCKED	本オブジェクトが，priorityと同じかそれより高い優先度でロックされている
E_OBJECT_FAULT	本オブジェクトが故障しており，ロックを取得できない

【機能】

本オブジェクトを，指定した操作優先度でロックする。ロックを取得したアプリケーションは，状態を変化させるサービスコール（activate，deactivate，setAssistLevel）を，同じかそれより低い優先度の他のアプリケーションによる操作から保護する。

既存のロックより高い優先度でロックを取得した場合，既存のロックは強制解除され，そのロックを取得していたアプリケーションにはOwnLockRevokedイベントが通知される。

運転者による直接の操作は，ロックおよび操作優先度によって妨げられない。

unlock

LKAのロックの解除

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_OBJECT_STATUS	自アプリケーションが，本オブジェクトをロックしていない

【機能】

自アプリケーションが取得した本オブジェクトのロックを解除する。

notify

LKAイベントの通知要求

【パラメータ】

eventFilter	enumSet (LKAEventKindType)?	通知するイベントの種類の集合

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【エラーコード】

E_INVALID_PARAMETER	eventFilterが有効範囲外
E_RISK_APPLICATION	アプリケーションがRiskPrivacyに対応していない
E_RISK_USER	ユーザがRiskPrivacyを承認していない

【機能】

LKAで発生したeventFilterで指定したイベントの通知を要求する。イベントを通知するためのイベントキューを生成し，その通知ハンドルをnotifyHandleに返す。

eventFilterを省略した場合，すべての種類のイベントの通知を要求する。

イベントの発生の観測からも運転の状況を推定できるため，本サービスコールもRiskPrivacyのリスク制御の対象である。

unnotify

LKAイベントの通知停止

【パラメータ】

notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正

【機能】

notifyHandleで指定した通知を停止する。指定した通知用のイベントキューは削除される。

getEvent

LKAイベントの取得

【パラメータ】

notifyHandle	NotifyHandleType	イベントキューを識別するための通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
event	LKAEventType	取得したイベント

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正
E_NO_DATA	イベントキューが空
E_RISK_APPLICATION	アプリケーションがRiskPrivacyに対応していない
E_RISK_USER	ユーザがRiskPrivacyを承認していない

【機能】

notifyHandleで指定した通知用のイベントキューから，先頭のイベントを取り出す。取り出したイベントは，イベントキューから削除される。

イベントの内容からも運転の状況を推定できるため，本サービスコールもRiskPrivacyのリスク制御の対象である。



イベントがパラメータ等で渡されてくるプログラミング環境向けの物理APIでは，用意する必要がない。

5.4. Vehicle.ADADAS.LDW（略称：LDW）

5.4.1. 位置付けと機能

Vehicle.ADADAS.LDW（略称：LDW）は，車両に搭載された車線逸脱警報（LDW）を参照・操作するためのシングルインスタンスオブジェクトである。作動の開始・終了の要求と，現在の作動状態および警報の出力の状況の参照を提供する。

本オブジェクトは，Vehicle.ADADASの目録に，機能の種別LDW，提供するオブジェクトの完全修飾名Vehicle.ADADAS.LDWとして現れる。DDTの責任主体・運行設計領域の充足状態・縮退運転・最小リスク状態など，AD/ADASシステム全体について一つに定まる状態は，本オブジェクトではなくVehicle.ADADASが提供する。

5.4.1.1. LDWの位置付けと性質

LDWは，自車が走行車線から逸脱しようとしていることを検知し，運転者に警報を出力する運転支援の機能である。LDWは車両の運動に介入しない。警報のみを行い，操舵の介入は行わないため，LDWの作動中も車両の運動の制御は運転者が行う。

本オブジェクトは，LDWが用いるセンサの構成，車線の認識と逸脱の判定のアルゴリズムを規定しない。検知した車線の形状・位置，逸脱の方向・量といった判定の根拠は本オブジェクトでは扱わない。本オブジェクトが提供するのは，機能が作動しているか（operatingState），いま作動を継続できるか（availability），現に警報を出力しているか（functionOutput），および故障の区分（faultCategory）である。

5.4.1.2. 作動の開始・終了の操作

本オブジェクトは，作動の開始（activate）と終了（deactivate）をアプリケーションから要求できる。車両が備える設定の手段（車載の設定画面等）により運転者がLDWを有効化・無効化した場合も，同じ作動状態operatingStateに反映される。

アプリケーションが行える操作は，構成情報supportedOperationsが宣言する。supportedOperationsの列挙子は，操作を行うサービスコールと1対1に対応する。アプリケーションは，画面を構成する前にsupportedOperationsを参照し，宣言されていない操作の要素を提示してはならない。宣言されていない操作のサービスコールを呼び出した場合，E_NOT_SUPPORTEDエラーとなる。

supportedOperationsが操作の可否（恒久的な可否）を表すのに対し，availabilityは，いまその操作を受け付けられるか（一時的な可否）を表す。この2つは別の判定であり，アプリケーションは，前者により要素を提示するか否かを，後者により要素を操作可能とするか否かを決める。

警報を出力しないようにすることは，車線の逸脱に運転者が気付かない状態を生じうる。作動の終了の要求に伴うリスクは，操作・参照に伴うリスクの節に規定する。

5.4.1.3. 作動状態

本オブジェクトの作動状態operatingStateおよび利用可否availabilityは，Vehicle.ADADASが規定するAD/ADAS機能共通の作動状態モデルに従う。本節では，LDWに固有の制約のみを規定する。

LDWにおける各作動状態の意味は，次のとおりである。

- Off … LDWが無効化されており，車線の逸脱を監視していない状態。アプリケーションによる作動の終了の要求，および運転者の設定のいずれによっても，この状態となる。
- Standby … LDWが有効であるが，車速が警報の対象となる範囲の外にある等の理由で，車線の逸脱を監視していない状態。
- Activating … 作動開始の要求（activate）を受理し，遷移の処理を行っている過渡状態。
- Active … LDWが有効であり，車線の逸脱を監視している状態。この状態において，逸脱を検知すると出力が生じ，functionOutputのactiveがtrueとなる。
- Deactivating … 作動終了の要求（deactivate）を受理し，Offへ遷移する処理を行っている過渡状態。
- Suspended … 他のAD/ADAS機能が横方向の運動を担っているため，車線の逸脱の監視と警報を行わない状態。この場合，availabilityはTemporarilyUnavailable，unavailableReasonはSupersededByOtherFunctionとなり，blockingFunctionがその機能を指す。
- Fault … 故障により機能を提供できない状態。故障の区分は状態faultCategoryが表す。

LDWでは，FallbackおよびMRCの列挙子は現れない。縮退運転および最小リスク状態への移行は，DDTの責任を持つ機能が行うものであり，警報のみを行うLDWはこれを行わないためである。

車速が警報の対象となる範囲を外れた場合は，Suspendedではなく，StandbyかつavailabilityがTemporarilyUnavailableで表す。operatingStateがOffである間，availabilityは，LDWを有効にできるか（故障の有無等）のみを表し，有効化した後にのみ意味を持つ作動の条件（車速が警報の対象となる範囲にあるか，車線を認識できているか等）を反映しない。故障等により有効にできない場合を除き，Offである間のavailabilityはAvailableであり，アプリケーションは作動の開始（activate）を要求できる。作動の条件を満たしているか否かは，有効化した後に，availabilityおよびunavailableReasonにより表される。

運転者の設定によりLDWが無効化されている場合も，operatingStateはOffとなり，availabilityは前述の規定に従う。このときアプリケーションが行う作動の開始の要求は，後勝ちの原則に従い，運転者の設定より後に行われた操作として受理される。アプリケーションは，作動状態の変化をOperatingStateChangedイベント，利用可否の変化をAvailabilityChangedイベントにより知ることができる。

車両が，アプリケーションからの作動の開始・終了そのものを認めない場合は，構成情報supportedOperationsにActivateおよびDeactivateを含めないことで表す。

警報を出力しているか否かは，作動状態ではなくfunctionOutputで表す。作動状態Activeは「監視しており，いつでも警報を出力できる状態」を表し，警報の出力そのものは示さない。

5.4.1.4. 警報の出力

警報の出力の状況は，状態functionOutputが表す。activeは，車線の逸脱を検知しているか，または運転者への提示を現に行っているかを表す。levelは出力の段階を表し，機能クラスWarningLevelReportedを備えている場合にのみ値を持つ。

段階により，「車線の逸脱を検知したが，まだ運転者への提示を行っていない状態」（Detected）と，

「警報を出力している状態」（Advisory，Warning）とを区別できる。Advisoryは表示等による弱い提示を，Warningは音・振動を伴う提示を表す。LDWは車両の運動に介入しないため，Criticalは現れない。その機能が出しうる最も強い段階は，構成情報maximumWarningLevelが宣言する。

警報の提示の方法（表示・音・触覚）は，機能クラスVisualWarning，AudibleWarning，HapticWarningが宣言する。どの方法で提示するかは車両実装に依存し，警報1回ごとの提示の方法は本オブジェクトでは表さない。

functionOutputの値が変化したときは，FunctionOutputChangedイベントが発生する。逸脱の検知（Detected）の開始，提示の段階の変化（DetectedからAdvisory，AdvisoryからWarning等），および出力の終了のいずれでも発生する。イベントは事象の発生のみを伝え，値そのものは運ばない。値は，イベントを受けたアプリケーションがgetStatusにより取得する。



逸脱の方向（左・右），検知した車線の形状・位置は本オブジェクトでは提供しない。これらはVehicle.SurroundModelの担当である。

5.4.1.5. LDWに対する制御の調停

本オブジェクトに対する制御の調停には，標準的な調停機能（後勝ちの原則）が適用される。すなわち，あるアプリケーションが作動を開始した後に，別のアプリケーションが作動の終了を要求した場合，後の要求が有効となる。

本オブジェクトは，標準的なロック機能を持つLockableObjectである。状態を変化させるサービスコール（activate，deactivate）およびロックを取得するサービスコール（lock）は，操作優先度priorityを持つ。ロック中の本オブジェクトに対して，ロックと同じかそれより低い優先度で操作した場合はE_OBJECT_LOCKEDエラーとなり，より高い優先度で操作した場合は，既存のロックが強制解除されて操作が実行される。アプリケーションが使用できる最高値より高い優先度を指定した場合はE_DENIED_PRIORITYエラー，操作優先度が有効範囲外である場合はE_INVALID_PARAMETERエラーとなる。

自アプリケーションのロックが強制解除された場合はOwnLockRevokedイベントが，他のアプリケーション等によりロックが解除された場合はOtherLockReleasedイベントが発生する。作動状態がFaultとなった場合，ロックは解除され，その間は新たにロックを取得することができない。

運転者による直接の操作（車載の設定画面等による有効化・無効化）は，本APIを用いない一つのアプリケーションによる操作と見なし，後勝ちの原則に従う。この操作は，アプリケーションが取得したロックおよび操作優先度によって妨げられることはなく，常に有効となる。運転の責任を負うのは運転者であり，アプリケーションによる排他が運転者の操作に優先してはならないためである。

5.4.1.6. 操作・参照に伴うリスク

LDWの作動を終了させると，車線の逸脱の警報が出力されなくなり，運転者が逸脱に気付かないまま走行することにつながりうる。これをリスククラスRiskCollisionとして定義し，activateおよびdeactivateをリスク制御の対象とする。作動の開始も，警報の出力により運転者の注意を引くものであり，同じリスククラスの対象とする。

LDWの作動状態および警報の出力の履歴は，運転者がいつどのように運転していたかを第三者が推定できる情報である。これをリスククラスRiskPrivacyとして定義し，状態を参照するサービスコール（getStatus），イベントの通知要求（notify）およびイベントの取得（getEvent）をリスク制御の対象とする。

いずれのリスククラスについても，当該リスクに対応していないアプリケーションからの呼び出しはE_RISK_APPLICATIONエラー，ユーザが当該リスクを承認していない場合はE_RISK_USERエラーとなる。

構成情報の参照（getConfig）は，車両に搭載されたLDWの静的な性質を返すものであり，リスク制御の対象としない。

5.4.2. 機能クラス

LDWは，以下の機能クラスを持つ。

WarningLevelReported	出力の段階（検知，注意喚起，警報）を提供することができる（状態functionOutputのlevel）
VisualWarning	表示による警報を提示することができる
AudibleWarning	音による警報を提示することができる
HapticWarning	触覚（ステアリングまたは座席の振動等）による警報を提示することができる

5.4.3. リスククラス

LDWは，以下のリスククラスを持つ。

RiskCollision	LDWの作動の開始・終了により警報の出力の有無が変わり，運転者が車線の逸脱に気付かないまま走行して衝突を生じるリスク
RiskPrivacy	走行状態および運転者の運転への関与状態の開示により，個人の行動を推定できることに伴うプライバシのリスク

5.4.4. 構成情報

LDWは，以下の構成情報を持つ。

capabilities	enumSet (LDWCapabilityType)	その車両のLDWが持つ機能
riskClasses	enumSet(LDWRiskType)	車両およびビークルOSで対応できていないリスククラスの集合
supportedOperations	enumSet (LDWOperationType)	アプリケーションから要求できる操作の集合。空集合となることがある
displayName	String?	ユーザに表示する呼称を取得するためのキー。車両が呼称を公開する場合にのみ値を持つ
warningSpeedRange	LDWSpeedRangeType	警報の対象となる車速の範囲。この範囲の外では，作動状態はStandbyとなる
maximumWarningLevel	AutomationOutputLevelType?	この機能が出しうる最も強い段階。機能クラスWarningLevelReportedを備えている場合にのみ値を持つ
timestampAccuracy	UInt32	この車両のタイムスタンプ誤差上限の宣言。単位はμs

上記は，YAML形式で記述した構成記述ファイル中に記述する形式である。本オブジェクトはシングルインスタンスオブジェクトであるため，構成記述では，車両を表すノードの下に置かれたADADASノードの下に，オブジェクトを表すノードLDWを置き，その下に上記の構成情報を記述する。このノードは，Vehicle.ADADASの目録において，providingObjectがVehicle.ADADAS.LDWである要素に対応する。

riskClassesには，車両およびビークルOSで対応できていない（危険を防止できていない）リスククラスの集合を記述する。

構成情報を参照するサービスコールは，この構成情報に対して，次の変換を行った構成情報を返す。

- displayNameに対しては，それを元に現在のロケールに対応する呼称の文字列を取得し，その文字列を返す。

5.4.5. 状態

LDWは，以下の状態を持つ。

timestamp	TimestampUsType	状態の測定時刻。単位はμs。車内共通時刻基盤に基づく
dataQualifier	DataQualifierType	このサンプルの品質。InvalidまたはUnavailableの場合，以下の状態の値を読んではならない。値の変化はDataQualifierChangedイベントにより通知される
operatingState	AutomationOperatingStateType	現在の作動状態。LDWではFallbackおよびMRCは現れない
availability	AutomationAvailabilityType	いま車線の逸脱の監視を開始または継続できるか
unavailableReason	AutomationUnavailableReasonType?	監視を開始または継続できない理由。availabilityがAvailable以外の値である場合にのみ値を持つ
blockingFunction	RelatedObjectType[]	監視を妨げている機能オブジェクト。unavailableReasonがDependentFunctionFaultまたはSupersededByOtherFunctionである場合にのみ要素を持つ
faultCategory	AutomationFaultCategoryType?	故障の区分。operatingStateがFaultである場合にのみ値を持つ
functionOutput	AutomationFunctionOutputType	警報の出力の状況。levelは機能クラスWarningLevelReportedを備えている場合にのみ値を持つ
lockStatus	LockStatusType	ロック状態

状態は，呼び出し時点のLDWの状態を返す冪等なスナップショットである。値を読む前にdataQualifierを確認する。

LDWが警報を出せる状態にあるかは，operatingStateにより判定する。いま車線の逸脱を検知しているか，警報を出しているかは，functionOutputにより判定する。operatingStateがActive以外である間，functionOutputのactiveは常にfalseである。

operatingStateとavailabilityの組合せの規則は，Vehicle.ADADASが規定するAD/ADAS機能共通の作動状態モデルによる。車速が構成情報warningSpeedRangeの範囲の外にある場合，availabilityはTemporarilyUnavailable，unavailableReasonはVehicleStateNotReadyとなり，operatingStateはStandbyとなる。

故障により機能を提供できない場合，operatingStateはFaultとなり，faultCategoryがその区分を表す。

これにより，FaultDetectedイベントを受け取っていないアプリケーションも，状態の参照のみで故障の区分を知ることができる。本オブジェクト自体が状態を提供できない場合は，getStatusがE_OBJECT_FAULTエラーを返す。

5.4.6. サービスコール一覧

LDWに対するサービスコールは以下の通り。

getConfig	LDWの構成情報の参照
getStatus	LDWの状態の参照
activate	LDWの作動の開始の要求
deactivate	LDWの作動の終了の要求
lock	LDWのロックの取得
unlock	LDWのロックの解除
notify	LDWイベントの通知要求
unnotify	LDWイベントの通知停止
getEvent	LDWイベントの取得

5.4.7. イベント

イベントは，事象が発生したことのみを伝え，状態の値そのものを運ばない。ただし，事象の発生時点の情報として有用なもの（故障の区分等）は，イベントの関連情報としても渡す。アプリケーションは，イベントを受けた後，必要に応じてgetStatusにより値を取得する。

LDWに対するイベント種別は以下の通り。

OperatingStateChanged	作動状態の変化（アプリケーションの要求および運転者の設定による有効化・無効化を含む）
AvailabilityChanged	車線の逸脱の監視を開始または継続できるかの変化（監視を妨げている理由の変化を含む）
FunctionOutputChanged	車線の逸脱の警報の出力の状況（状態functionOutput）の変化。出力の開始・段階の変化・終了のいずれでも発生する
FaultDetected	故障の検知（LDWの機能を提供できない状態）
FaultRecovered	故障からの回復
DataQualifierChanged	dataQualifierの変化
OwnLockRevoked	自アプリケーションのロックの強制解除
OtherLockReleased	他のアプリケーション等によるロック解除
EventOverflow	イベントキューのオーバフロー
ConfigChanged	構成情報の変更（ソフトウェアの更新による対応する操作・警報の対象となる車速の範囲・出力の段階の変更）

この内，イベント種別がFaultDetectedの場合には，以下の関連情報を持つ。

faultCategory	AutomationFaultCategoryType	故障の区分

また，イベント種別がEventOverflowの場合には，以下の関連情報を持つ。

noLostEvents	UInt32	失われたイベントの数

5.4.8. データ型の定義

5.4.8.1. LDWの機能クラス

```
type LDWCapabilityType = enum {
    WarningLevelReported,
    VisualWarning,
    AudibleWarning,
    HapticWarning
};
```

5.4.8.2. LDWのリスククラス

```
type LDWRiskType = enum {
    RiskCollision,
    RiskPrivacy
};
```

5.4.8.3. タイムスタンプ

```
type TimestampUsType = UInt64;  // 車内共通時刻基盤の起点を基準とする測定時刻。単位はμs。複数のビークルAPIで共通に用いる共通データ型であり，本仕様内に定義する。
```

5.4.8.4. サンプル品質の種別

```
type DataQualifierType = enum {
    Normal, // 正常
    Unavailable,    // 利用不可
    Degraded,   // 劣化
    TestMode,   // テストモード
    Invalid // 無効（値を読んではならない）
};
```

複数のビークルAPIで共通に用いる共通データ型であり，本仕様内に定義する。

5.4.8.5. 速度

```
type SpeedType = Float32;   // 速度。単位はkm/h。正の値は前進，負の値は後退を示す。複数のビークルAPIで共通に用いる共通データ型であり，本仕様内に定義する。
```

5.4.8.6. AD/ADAS機能共通の作動状態

以下の5つのデータ型は，AD/ADAS機能オブジェクトが共通に用いるデータ型であり，本仕様内に定義する。LDWにおける各列挙子の具体的な意味は，位置付けと機能の節に示す。

```
type AutomationOperatingStateType = enum {
    Off,    // 機能が無効化されており，作動していない状態
    Standby,    // 機能が有効であり，作動の条件が整えば作動できるが，現在は作動していない状態
    Activating, // 作動開始の要求を受理し，作動開始の処理を行っている過渡状態
    Active, // 作動中。車両の運動への介入または警報の出力を現に行っている状態
    Suspended,  // 作動が一時的に解除され，中断している状態。直前の作動時の設定値を保持しており，抑止する要因が解消すれば作動を再開できる
    Deactivating,   // 作動終了の要求を受理し，作動終了の処理を行っている過渡状態
    Fallback,   // 縮退運転を実行している状態
    MRC,    // 最小リスク状態へ移行し，これを維持している状態
    Fault   // 故障により機能を提供できない状態
};
```

各機能オブジェクトが現にどの作動状態にあるかを表す型である。機能によっては現れない列挙子がある。

```
type AutomationAvailabilityType = enum {
    Available,  // 作動を開始または継続できる
    TemporarilyUnavailable, // 一時的に条件を満たさないため，作動を開始または継続できない（条件が回復すれば作動できる）
    Unavailable // 故障等により，作動を開始または継続できない
};
```

その機能がいま作動を開始または継続できるかを表す型である。作動しているか否かは作動状態operatingStateが表すものであり，本型はこれを表さない。

```
type AutomationUnavailableReasonType = enum {
    ODDOutside, // 走行環境が運行設計領域の外にある
    SensorDegraded, // センサの性能低下・遮蔽により，必要な認識の性能が得られない
    DependentFunctionFault, // この機能が用いている他のAD/ADAS機能が故障している（blockingFunctionがその機能を指す）
    DriverMonitoringUnavailable,    // 運転者の監視が行えない
    VehicleStateNotReady,   // 車速・変速機の位置・制動等の車両の状態が作動の条件を満たさない
    FunctionFault,  // この機能自体が故障している
    ControlOverridden,  // 運転者の操作または他の経路による車両の運動の制御により，作動が抑止されている
    UserDisabled,   // 利用者の設定により機能が無効にされている
```

```
    SupersededByOtherFunction,  // 他のAD/ADAS機能がその運転操作を担っているため作動しない（blockingFunctionがその機能を指す）
    CalibrationRequired,    // 較正が必要で作動できない
    Other   // その他の理由
};
```

availabilityがAvailable以外の値である場合に，作動を開始または継続できない理由を表す型である。機能によっては現れない列挙子がある。

```
type AutomationOutputLevelType = enum {
    Detected,   // 事象を検知したが，運転者への提示はまだ行っていない
    Advisory,   // 注意喚起（表示等の弱い提示）
    Warning,    // 警報（音・振動を伴う提示）
    Critical    // 緊急（車両運動への介入を伴う）
};
```

機能の出力の段階を表す型である。Detected，Advisory，Warning，Criticalの順に強い段階を表す。機能によっては現れない段階がある。

```
type AutomationFunctionOutputType = struct {
    active : Bool,  // その機能が事象の検知または運転者への提示を現に行っているか
    level : AutomationOutputLevelType?  // 出力の段階。activeがtrueであり，かつ段階を提供できる機能である場合にのみ値を持つ
};
```

機能が外部（運転者）に対して現に行っている出力を，有無と段階のみで表す型である。出力の判断の根拠（検知した車線の形状・物体の種別等）は本型に含まない。

5.4.8.7. AD/ADAS機能共通の故障の区分

以下のデータ型は，AD/ADAS機能オブジェクトが共通に用いるデータ型であり，本仕様内に定義する。LDWは車両の運動に介入しないため，アクチュエータの故障（ActuatorFault）は現れない。

```
type AutomationFaultCategoryType = enum {
    SensorFault,    // センサの故障
    ActuatorFault,  // アクチュエータの故障
    ComputationFault,   // 演算装置の故障
    CommunicationFault, // 車内通信の故障
    PowerFault, // 電源の異常
    WarningDeviceFault, // 警報を提示する装置の故障
    Other   // その他の故障
};
```

機能を提供できない故障の区分を表す型である。機能によっては現れない列挙子がある。故障の区分は，各機能オブジェクトの状態faultCategoryおよびFaultDetectedイベントの関連情報により渡される。

5.4.8.8. LDWの操作

```
type LDWOperationType = enum {
    Activate,   // 作動の開始の要求（サービスコールactivate）
    Deactivate  // 作動の終了の要求（サービスコールdeactivate）
};
```

アプリケーションから要求できる操作を表す型である。列挙子は，操作を行うサービスコールと1対1に対応する。すべてのアプリケーションが呼び出すことができるサービスコール（getConfig，getStatus，notify，unnotify，getEvent，lock，unlock）は，本型の対象としない。

5.4.8.9. 警報の対象となる車速の範囲

```
type LDWSpeedRangeType = struct {
    minimum : SpeedType,    // 警報の対象となる車速の下限。単位はkm/h
    maximum : SpeedType?    // 警報の対象となる車速の上限。単位はkm/h。上限を設けない
場合は値を持たない
};
```

5.4.8.10. LDWの構成情報（getConfigが返すデータ型）

```
type LDWConfigType = struct {
    capabilities : enumSet(LDWCapabilityType),
    riskClasses : enumSet(LDWRiskType),
    supportedOperations : enumSet(LDWOperationType),
    displayName : String?,
    warningSpeedRange : LDWSpeedRangeType,
    maximumWarningLevel : AutomationOutputLevelType?,
    timestampAccuracy : UInt32
};
```

5.4.8.11. LDW状態

```
type LDWStatusType = struct {
    timestamp : TimestampUsType,
    dataQualifier : DataQualifierType,
    operatingState : AutomationOperatingStateType,
    availability : AutomationAvailabilityType,
    unavailableReason : AutomationUnavailableReasonType?,
    blockingFunction : RelatedObjectType[],
    faultCategory : AutomationFaultCategoryType?,
    functionOutput : AutomationFunctionOutputType,
    lockStatus : LockStatusType
};
```

呼び出し時点のLDWの状態の冪等なスナップショットである。

5.4.8.12. LDWイベント

```
type LDWEventKindType = enum {
    OperatingStateChanged,
    AvailabilityChanged,
    FunctionOutputChanged,
    FaultDetected,
    FaultRecovered,
    DataQualifierChanged,
    OwnLockRevoked,
    OtherLockReleased,
    EventOverflow,
    ConfigChanged
};
```

```
type LDWEventType = struct {
    eventInfo : variant(LDWEventKindType) {
        FaultDetected {
            faultCategory : AutomationFaultCategoryType
        },
        EventOverflow {
            noLostEvents : UInt32
        }
    }
};
```

5.4.9. サービスコールの詳細規定

getConfig

LDWの構成情報の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
config	LDWConfigType	LDWの構成情報

【エラーコード】

なし

【機能】

その車両のLDWが持つ機能，アプリケーションから要求できる操作の集合，警報の対象となる車速の範囲，およびこの機能が出しうる最も強い段階を返す。

アプリケーションは，本サービスコールにより，画面を構成する前に，どの操作の要素を提示すべきかを決める。

getStatus

LDWの状態の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
status	LDWStatusType	LDWの状態

【エラーコード】

E_OBJECT_FAULT	本オブジェクトが故障しており，状態を提供できない
E_RISK_APPLICATION	アプリケーションがRiskPrivacyに対応していない
E_RISK_USER	ユーザがRiskPrivacyを承認していない

【機能】

LDWの現在の作動状態・利用可否・故障の区分・警報の出力の状況・ロック状態を返す。

呼び出しのたびに，その時点の状態の全体を返す冪等な応答である。値を読む前にdataQualifierを確認する。

activate

LDWの作動の開始の要求

【パラメータ】

priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_PARAMETER	priorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_LOCKED	本オブジェクトが，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている
E_OBJECT_STATUS	作動を開始できる作動状態でない（既に有効である，またはavailabilityがUnavailableであり，その理由が故障以外である）

E_OBJECT_FAULT	本オブジェクトが故障しており，要求を受け付けられない（unavailableReasonがFunctionFaultである場合を含む）
E_INFEASIBLE	availabilityがTemporarilyUnavailableであり，作動の条件以外の一時的な理由（故障等）により，現在はLDWを有効にできない
E_RISK_APPLICATION	アプリケーションがRiskCollisionに対応していない
E_RISK_USER	ユーザがRiskCollisionを承認していない

【機能】

LDWに作動の開始を要求する。本サービスコールは，LDWを有効にする操作に相当し，Offからの遷移を要求する。作動の条件（車速が警報の対象となる範囲にある等）が整っている場合はActiveへ，整っていない場合はStandbyへ遷移する。作動の条件が整っていないことは，本要求を受け付けられない理由とはならない。Offである間のavailabilityは，これらの作動の条件を反映しない。

正常終了は要求が受理されたことのみを意味し，作動が開始したことを意味しない。遷移の完了は，getStatusまたはOperatingStateChangedイベントにより確認する。

作動を開始できない場合の応答は，利用可否availabilityとその理由unavailableReasonにより定まる。availabilityがTemporarilyUnavailableである場合はE_INFEASIBLEエラー，Unavailableでありunavailable ReasonがFunctionFaultである場合はE_OBJECT_FAULTエラー，Unavailableでありそれ以外の理由である場合はE_OBJECT_STATUSエラーとなる。

本サービスコールは，構成情報supportedOperationsにActivateを含む場合にのみ用いることができる。

deactivate

LDWの作動の終了の要求

【パラメータ】

priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_PARAMETER	priorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_LOCKED	本オブジェクトが，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている
E_OBJECT_STATUS	作動を終了できる作動状態でない（既にOffである等）
E_OBJECT_FAULT	本オブジェクトが故障しており，要求を受け付けられない
E_RISK_APPLICATION	アプリケーションがRiskCollisionに対応していない
E_RISK_USER	ユーザがRiskCollisionを承認していない

【機能】

LDWに作動の終了を要求する。本サービスコールは，LDWを無効にする操作に相当し，Offへの遷移を要求する。

正常終了は要求が受理されたことのみを意味し，作動が終了したことを意味しない。遷移の完了は，getStatusまたはOperatingStateChangedイベントにより確認する。

本サービスコールは，構成情報supportedOperationsにDeactivateを含む場合にのみ用いることができる。



本サービスコールによりLDWが無効になると，車線の逸脱の警報は出力されなくなる。運転者が逸脱に気付かないまま走行することにつながるため，アプリケーションは，無効にしたことを運転者が認識できるようにしなければならない。

lock

LDWのロックの取得

【パラメータ】

priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_PARAMETER	priorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_LOCKED	本オブジェクトが，priorityと同じかそれより高い優先度でロックされている
E_OBJECT_FAULT	本オブジェクトが故障しており，ロックを取得できない

【機能】

本オブジェクトを，指定した操作優先度でロックする。ロックを取得したアプリケーションは，状態を変化させるサービスコール（activate，deactivate）を，同じかそれより低い優先度の他のアプリケーションによる操作から保護する。

既存のロックより高い優先度でロックを取得した場合，既存のロックは強制解除され，そのロックを取得していたアプリケーションにはOwnLockRevokedイベントが通知される。

運転者による直接の操作は，ロックおよび操作優先度によって妨げられない。

unlock

LDWのロックの解除

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_OBJECT_STATUS	自アプリケーションが，本オブジェクトをロックしていない

【機能】

自アプリケーションが取得した本オブジェクトのロックを解除する。

notify

LDWイベントの通知要求

【パラメータ】

eventFilter	enumSet (LDWEventKindType)?	通知するイベントの種類の集合

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【エラーコード】

E_INVALID_PARAMETER	eventFilterが有効範囲外
E_RISK_APPLICATION	アプリケーションがRiskPrivacyに対応していない
E_RISK_USER	ユーザがRiskPrivacyを承認していない

【機能】

LDWで発生したeventFilterで指定したイベントの通知を要求する。イベントを通知するためのイベントキューを生成し，その通知ハンドルをnotifyHandleに返す。

eventFilterを省略した場合，すべての種類のイベントの通知を要求する。

警報の発生の観測からも運転の傾向を推定できるため，本サービスコールもRiskPrivacyのリスク制御の対象である。

unnotify

LDWイベントの通知停止

【パラメータ】

notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正

【機能】

notifyHandleで指定した通知を停止する。指定した通知用のイベントキューは削除される。

getEvent

LDWイベントの取得

【パラメータ】

notifyHandle	NotifyHandleType	イベントキューを識別するための通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
event	LDWEventType	取得したイベント

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正
E_NO_DATA	イベントキューが空
E_RISK_APPLICATION	アプリケーションがRiskPrivacyに対応していない
E_RISK_USER	ユーザがRiskPrivacyを承認していない

【機能】

notifyHandleで指定した通知用のイベントキューから，先頭のイベントを取り出す。取り出したイベントは，イベントキューから削除される。

イベントの内容からも運転の傾向を推定できるため，本サービスコールもRiskPrivacyのリスク制御の対象である。



イベントがパラメータ等で渡されてくるプログラミング環境向けの物理APIでは，用意する必要がない。

5.5. Vehicle.ADADAS.LCA（略称：LCA）

5.5.1. 位置付けと機能

Vehicle.ADADAS.LCA（略称：LCA）は，車両に搭載された車線変更支援（LCA）を参照・操作するためのシングルインスタンスオブジェクトである。作動の開始・終了の要求と，現在の作動状態・車線変更の進行の段階・変更の方向・警報の出力の状況の参照を提供する。

本オブジェクトは，Vehicle.ADADASの目録に，機能の種別LCA，提供するオブジェクトの完全修飾名Vehicle.ADADAS.LCAとして現れる。DDTの責任主体・運行設計領域の充足状態・縮退運転・最小リスク状態など，AD/ADASシステム全体について一つに定まる状態は，本オブジェクトではなくVehicle.ADADASが提供する。

5.5.1.1. LCAの位置付けと性質

LCAは，運転者による車線変更の意図の表明を起点として，外界のセンサ（レーダ・カメラ等）により変更先の車線および後方から接近する車両を確認し，車線変更を支援する運転支援の機能である。支援の形態は，機能クラスが宣言する。機能クラスLaneChangeExecutionを備える場合，LCAは車線変更のための操舵を行う。機能クラスApproachWarningを備える場合，LCAは変更先の車線に接近する車両についての警報を出力する。

LCAの作動中も，動的運転タスク（DDT）の責任は運転者にあり，運転者には周囲の監視の義務が残る。

本オブジェクトは，LCAが用いるセンサの構成，車線と周囲の車両の認識のアルゴリズム，車線変更の可否の判断と操舵の制御のアルゴリズムを規定しない。検知した車線の形状・位置，接近する車両の位置・相対速度といった判断の根拠は本オブジェクトでは扱わない。



検知した車線および周囲の物体は，Vehicle.SurroundModelが提供する。

5.5.1.2. 運転者の意図の受け付け

車線変更は，運転者による意図の表明を車両が検知することにより開始される。意図の表明の手段（方向指示器の操作，ステアリングスイッチの操作等）およびその判定は，車両実装による。変更の方向は，運転者が表明した方向とする。

本オブジェクトは，アプリケーションから車線変更の開始または中止を要求する手段を提供しない。アプリケーションが行えるのは，LCAの作動の開始・終了の要求と，車線変更の進行の参照である。

5.5.1.3. 作動状態と車線変更の進行

本オブジェクトの作動状態operatingStateおよび利用可否availabilityは，Vehicle.ADADASが規定するAD/ADAS機能共通の作動状態モデルに従う。本節では，LCAに固有の制約のみを規定する。

LCAにおける各作動状態の意味は，次のとおりである。

- Off … LCAが無効化されており，運転者の意図を受け付けない状態。アプリケーションによる作動の終了の要求，および運転者の設定のいずれによっても，この状態となる。
- Standby … LCAが有効であるが，車速が作動の対象となる範囲の外にある等の理由で，運転者の意図を受け付けていない状態。
- Activating … 作動開始の要求（activate）を受理し，遷移の処理を行っている過渡状態。
- Active … LCAが有効であり，運転者の意図を受け付けて車線変更の支援を行える状態。車線変更の支援を現に行っている間を含む。
- Suspended … 他のAD/ADAS機能が横方向の運動を担っている間，または他の経路により横方向の運動が制御されている間，車線変更の支援を行わない状態。
- Deactivating … 作動終了の要求（deactivate）を受理し，Offへ遷移する処理を行っている過渡状態。
- Fault … 故障により機能を提供できない状態。故障の区分は状態faultCategoryが表す。

LCAでは，FallbackおよびMRCの列挙子は現れない。縮退運転および最小リスク状態への移行は，DDTの

責任を持つ機能が行うものであり，運転支援の機能であるLCAはこれを行わないためである。

LCAが有効である間に車速が構成情報operatingSpeedRangeの範囲の外にある場合は，Standbyかつ availabilityがTemporarilyUnavailable，unavailableReasonがVehicleStateNotReadyで表す。運転操作の全体を担う他のAD/ADAS機能が横方向の運動を担っている間は，operatingStateはSuspended，availabilityはTemporarilyUnavailable，unavailableReasonはSupersededByOtherFunctionとなり，blockingFunctionがその機能を指す。

operatingStateがOffである間，availabilityは，LCAを有効にできるか（故障の有無等）のみを表し，有効化した後にのみ意味を持つ作動の条件（車速が構成情報operatingSpeedRangeの範囲にあるか等）を反映しない。すなわち，これらの条件を満たさないことのみを理由として，Offである間のavailabilityがTemporarilyUnavailableとなることはない。故障等により有効にできない場合を除き，Offである間のavailabilityはAvailableであり，アプリケーションは作動の開始（activate）を要求できる。作動の条件を満たしているか否かは，有効化した後に，availabilityおよびunavailableReasonにより表される。運転者の設定によりLCAが無効化されている場合も，同様である。

車線変更の進行は，作動状態ではなく状態laneChangePhaseで表す。作動状態Activeは「運転者の意図を受け付けられる状態」を表し，車線変更を現に行っているか否かは示さない。laneChangePhaseは，機能クラスLaneChangeExecutionを備え，operatingStateがActiveであり，かつ車線変更の支援を現に行っている間にのみ値を持つ。各段階の意味は，次のとおりである。

- Checking … 運転者の意図を受け付け，変更先の車線の安全を確認している段階。接近する車両がある等の理由で直ちに車線変更を開始できない間も，この段階にとどまる。
- Executing … 車線変更のための操舵を行っている段階。
- Aborting … 車線変更を中止し，元の車線へ戻るための操舵を行っている段階。

車線変更が完了した場合はLaneChangeCompletedイベントが，中止された場合はLaneChangeAbortedイベントが発生し，laneChangePhaseは値を持たなくなる。中止の際に元の車線へ戻る操舵を行うか，操舵を直ちに運転者に戻すかは，車両実装による。元の車線へ戻る操舵を行う間，laneChangePhaseはAbortingとなり，LaneChangeAbortedイベントはその操舵を終えた時点で発生する。

運転者の操舵・制動等の操作により車線変更が中止された場合，LaneChangeAbortedイベントの関連情報reasonはDriverOverrideとなる。この場合，operatingStateはActiveのままであり，運転者は再び意図を表明することにより次の車線変更の支援を受けられる。

5.5.1.4. 警報の出力

機能クラスApproachWarningを備える場合，変更先の車線に接近する車両についての警報の出力の状況は，状態functionOutputが表す。activeは，接近する車両を検知しているか，または運転者への提示を現に行っているかを表し，levelは出力の段階を表す。LCAが接近する車両のために車線変更を中止する場合も，その警報の出力はfunctionOutputで表す。

functionOutputの値が変化したときは，FunctionOutputChangedイベントが発生する。

5.5.1.5. 作動の開始・終了の操作

本オブジェクトは，作動の開始（activate）と終了（deactivate）をアプリケーションから要求できる。車両が備える設定の手段（車載の設定画面等）により運転者がLCAを有効化・無効化した場合も，同じ作動状態operatingStateに反映される。

アプリケーションが行える操作は，構成情報supportedOperationsが宣言する。supportedOperationsの列挙子は，操作を行うサービスコールと1対1に対応する。アプリケーションは，画面を構成する前にsupportedOperationsを参照し，宣言されていない操作の要素を提示してはならない。宣言されていない操作のサービスコールを呼び出した場合，E_NOT_SUPPORTEDエラーとなる。車両が，アプリケーションからの作動の開始・終了そのものを認めない場合は，構成情報supportedOperationsにActivateおよびDeactivateを含めないことで表す。

supportedOperationsが操作の可否（恒久的な可否）を表すのに対し，availabilityは，いまその操作を受け付けられるか（一時的な可否）を表す。この2つは別の判定であり，アプリケーションは，前者により要素を提示するか否かを，後者により要素を操作可能とするか否かを決める。

要求を受け付けられない場合の応答は，AD/ADAS機能共通の作動状態モデルの規定による。すなわち，availabilityがTemporarilyUnavailableである場合の作動開始の要求はE_INFEASIBLEエラー，unavailableReasonがFunctionFaultである場合はE_OBJECT_FAULTエラー，それ以外のUnavailableである場合はE_OBJECT_STATUSエラーとなる。

5.5.1.6. LCAに対する制御の調停

本オブジェクトに対する制御の調停には，標準的な調停機能（後勝ちの原則）が適用される。すなわち，あるアプリケーションが作動を開始した後に，別のアプリケーションが作動の終了を要求した場合，後の要求が有効となる。

本オブジェクトは，標準的なロック機能を持つLockableObjectである。状態を変化させるサービスコール（activate，deactivate）およびロックを取得するサービスコール（lock）は，操作優先度priorityを持つ。ロック中の本オブジェクトに対して，ロックと同じかそれより低い優先度で操作した場合はE_OBJECT_LOCKEDエラーとなり，より高い優先度で操作した場合は，既存のロックが強制解除されて操作が実行される。アプリケーションが使用できる最高値より高い優先度を指定した場合はE_DENIED_PRIORITYエラー，操作優先度が有効範囲外である場合はE_INVALID_PARAMETERエラーとなる。

自アプリケーションのロックが強制解除された場合はOwnLockRevokedイベントが，他のアプリケーション等によりロックが解除された場合はOtherLockReleasedイベントが発生する。作動状態がFaultとなった場合，ロックは解除され，その間は新たにロックを取得することができない。

運転者による直接の操作（意図の表明，操舵・制動による中止，車載の設定画面等による有効化・無効化）は，本APIを用いない一つのアプリケーションによる操作と見なし，後勝ちの原則に従う。この操作は，アプリケーションが取得したロックおよび操作優先度によって妨げられることはなく，常に有効となる。運転の責任を負うのは運転者であり，アプリケーションによる排他が運転者の操作に優先してはならないためである。



Vehicle.Motionは，アプリケーションが車両の運動そのものを制御するためのオブジェクトであり，本オブジェクトは，車両に搭載されたLCAの作動を開始・終了させるものである。両者は別の経路であり，同時に用いられた場合も後勝ちの原則に従う。LCAの作動中にVehicle.Motionにより横方向の運動が制御された場合，本オブジェクトはavailabilityをTemporarilyUnavailable，unavailableReasonをControlOverriddenとし，operatingStateをSuspendedとする。このとき車線変更の支援を行っていた場合，車線変更は中止され，LaneChangeAbortedイベントの関連情報reasonはControlOverriddenとなる。

5.5.1.7. 操作・参照に伴うリスク

LCAの作動の開始・終了は，車線変更のための操舵および接近する車両についての警報の有無を変えるた

め，隣接する車線の車両・障害物との衝突のリスクを伴う。これをリスククラスRiskCollisionとして定義し，activateおよびdeactivateをリスク制御の対象とする。とくにdeactivateは，車線変更の操舵の途中で制御が失われることにより，運転者が引き継げない場合に衝突のリスクを生じる。

LCAの作動状態・車線変更の進行と方向・警報の出力の履歴は，運転者がいつどのように運転していたかを第三者が推定できる情報である。これをリスククラスRiskPrivacyとして定義し，状態を参照するサービスコール（getStatus），イベントの通知要求（notify）およびイベントの取得（getEvent）をリスク制御の対象とする。

いずれのリスククラスについても，当該リスクに対応していないアプリケーションからの呼び出しはE_RISK_APPLICATIONエラー，ユーザが当該リスクを承認していない場合はE_RISK_USERエラーとなる。

構成情報の参照（getConfig）は，車両に搭載されたLCAの静的な性質を返すものであり，リスク制御の対象としない。

5.5.2. 機能クラス

LCAは，以下の機能クラスを持つ。

LaneChangeExecution	運転者の意図を受けて，車線変更のための操舵を行うことができる（状態laneChangePhase，LaneChangePhaseChanged・LaneChangeCompleted・LaneChangeAbortedイベント）
ApproachWarning	変更先の車線に接近する車両についての警報を出力することができる（状態functionOutput，FunctionOutputChangedイベント）

5.5.3. リスククラス

LCAは，以下のリスククラスを持つ。

RiskCollision	LCAの作動の開始・終了により車線変更の操舵および警報の有無が変わり，隣接する車線の車両・障害物との衝突を生じるリスク
RiskPrivacy	走行状態および運転者の運転への関与状態の開示により，個人の行動を推定できることに伴うプライバシのリスク

5.5.4. 構成情報

LCAは，以下の構成情報を持つ。

capabilities	enumSet (LCACapabilityType)	その車両のLCAが持つ機能
riskClasses	enumSet(LCARiskType)	車両およびビークルOSで対応できていないリスククラスの集合
supportedOperations	enumSet (LCAOperationType)	アプリケーションから要求できる操作の集合。空集合となることがある
displayName	String?	ユーザに表示する呼称を取得するためのキー。車両が呼称を公開する場合にのみ値を持つ
operatingSpeedRange	LCASpeedRangeType	作動の対象となる車速の範囲。この範囲の外では，作動状態はStandbyとなる

timestampAccuracy	UInt32	この車両のタイムスタンプ誤差上限の宣言。単位はμs

上記は，YAML形式で記述した構成記述ファイル中に記述する形式である。本オブジェクトはシングルインスタンスオブジェクトであるため，構成記述では，車両を表すノードの下に置かれたADADASノードの下に，オブジェクトを表すノードLCAを置き，その下に上記の構成情報を記述する。このノードは，Vehicle.ADADASの目録において，providingObjectがVehicle.ADADAS.LCAである要素に対応する。

riskClassesには，車両およびビークルOSで対応できていない（危険を防止できていない）リスククラスの集合を記述する。

構成情報を参照するサービスコールは，この構成情報に対して，次の変換を行った構成情報を返す。

- displayNameに対しては，それを元に現在のロケールに対応する呼称の文字列を取得し，その文字列を返す。

5.5.5. 状態

LCAは，以下の状態を持つ。

timestamp	TimestampUsType	状態の測定時刻。単位はμs。車内共通時刻基盤に基づく
dataQualifier	DataQualifierType	このサンプルの品質。InvalidまたはUnavailableの場合，以下の状態の値を読んではならない。値の変化はDataQualifierChangedイベントにより通知される
operatingState	AutomationOperatingStateType	現在の作動状態。LCAではFallbackおよびMRCは現れない
availability	AutomationAvailabilityType	いま運転者の意図の受け付けを開始または継続できるか
unavailableReason	AutomationUnavailableReasonType?	意図の受け付けを開始または継続できない理由。availabilityがAvailable以外の値である場合にのみ値を持つ
blockingFunction	RelatedObjectType[]	作動を妨げている機能オブジェクト。unavailableReasonがDependentFunctionFaultまたはSupersededByOtherFunctionである場合にのみ要素を持つ
faultCategory	AutomationFaultCategoryType?	故障の区分。operatingStateがFaultである場合にのみ値を持つ
laneChangePhase	LCALaneChangePhaseType?	車線変更の進行の段階。機能クラスLaneChangeExecutionを備え，operatingStateがActiveであり，かつ車線変更の支援を現に行っている場合にのみ値を持つ
laneChangeDirection	LCADirectionType?	支援している車線変更の方向。laneChangePhaseが値を持つ場合にのみ値を持つ。Abortingである間も，中止した車線変更の方向を表す

functionOutput	AutomationFunctionOutputType?	変更先の車線に接近する車両についての警報の出力の状況。機能クラスApproachWarningを備えている場合にのみ値を持つ
lockStatus	LockStatusType	ロック状態

状態は，呼び出し時点のLCAの状態を返す冪等なスナップショットである。値を読む前にdataQualifierを確認する。

LCAが運転者の意図を受け付けられる状態にあるかは，operatingStateにより判定する。いま車線変更の支援を行っているかは，laneChangePhaseが値を持つか否かにより判定する。operatingStateがActive以外である間，laneChangePhaseは値を持たず，functionOutputのactiveは常にfalseである。

operatingStateとavailabilityの組合せの規則は，Vehicle.ADADASが規定するAD/ADAS機能共通の作動状態モデルによる。

故障により機能を提供できない場合，operatingStateはFaultとなり，faultCategoryがその区分を表す。これにより，FaultDetectedイベントを受け取っていないアプリケーションも，状態の参照のみで故障の区分を知ることができる。本オブジェクト自体が状態を提供できない場合は，getStatusがE_OBJECT_FAULTエラーを返す。

5.5.6. サービスコール一覧

LCAに対するサービスコールは以下の通り。

getConfig	LCAの構成情報の参照
getStatus	LCAの状態の参照
activate	LCAの作動の開始の要求
deactivate	LCAの作動の終了の要求
lock	LCAのロックの取得
unlock	LCAのロックの解除
notify	LCAイベントの通知要求
unnotify	LCAイベントの通知停止
getEvent	LCAイベントの取得

5.5.7. イベント

イベントは，事象が発生したことのみを伝え，状態の値そのものを運ばない。ただし，事象の発生時点の情報として有用なもの（故障の区分，車線変更の方向と中止の理由等）は，イベントの関連情報としても渡す。アプリケーションは，イベントを受けた後，必要に応じてgetStatusにより値を取得する。

車線変更が完了または中止された場合，LaneChangePhaseChangedイベントに加えて，LaneChangeCompletedイベントまたはLaneChangeAbortedイベントが発生する。

状態の取得は，要求応答（getStatus）と，各状態項目の変化イベントによる。

LCAに対するイベント種別は以下の通り。

OperatingStateChanged	作動状態の変化（アプリケーションの要求および運転者の設定による有効化・無効化を含む）
AvailabilityChanged	運転者の意図の受け付けを開始または継続できるかの変化（作動を妨げている理由の変化を含む）
LaneChangePhaseChanged	車線変更の進行の段階（状態laneChangePhase）の変化。車線変更の支援の開始・段階の変化・終了のいずれでも発生する
LaneChangeCompleted	車線変更の完了
LaneChangeAborted	車線変更の中止
FunctionOutputChanged	変更先の車線に接近する車両についての警報の出力の状況（状態functionOutput）の変化。出力の開始・段階の変化・終了のいずれでも発生する
FaultDetected	故障の検知（LCAの機能を提供できない状態）
FaultRecovered	故障からの回復
DataQualifierChanged	状態の品質（状態dataQualifier）の変化
OwnLockRevoked	自アプリケーションのロックの強制解除
OtherLockReleased	他のアプリケーション等によるロック解除
EventOverflow	イベントキューのオーバフロー
ConfigChanged	構成情報の変更（ソフトウェアの更新による対応する操作・作動の対象となる車速の範囲の変更）

この内，イベント種別がLaneChangeCompletedの場合には，以下の関連情報を持つ。

direction	LCADirectionType	完了した車線変更の方向

また，イベント種別がLaneChangeAbortedの場合には，以下の関連情報を持つ。

direction	LCADirectionType	中止した車線変更の方向
reason	LCAAbortReasonType	車線変更を中止した理由

また，イベント種別がFaultDetectedの場合には，以下の関連情報を持つ。

faultCategory	AutomationFaultCategoryType	故障の区分

また，イベント種別がEventOverflowの場合には，以下の関連情報を持つ。

noLostEvents	UInt32	失われたイベントの数

5.5.8. データ型の定義

5.5.8.1. LCAの機能クラス

```
type LCACapabilityType = enum {
    LaneChangeExecution,
    ApproachWarning
```

```
};
```

5.5.8.2. LCAのリスククラス

```
type LCARiskType = enum {
    RiskCollision,
    RiskPrivacy
};
```

5.5.8.3. タイムスタンプ

```
type TimestampUsType = UInt64;  // 車内共通時刻基盤の起点を基準とする測定時刻。単位はμs。複数のビークルAPIで共通に用いる共通データ型であり，本仕様内に定義する。
```

5.5.8.4. サンプル品質の種別

```
type DataQualifierType = enum {
    Normal, // 正常
    Unavailable,    // 利用不可
    Degraded,   // 劣化
    TestMode,   // テストモード
    Invalid // 無効（値を読んではならない）
};
```

複数のビークルAPIで共通に用いる共通データ型であり，本仕様内に定義する。

5.5.8.5. 速度

```
type SpeedType = Float32;   // 速度。単位はkm/h。正の値は前進，負の値は後退を示す。複数のビークルAPIで共通に用いる共通データ型であり，本仕様内に定義する。
```

5.5.8.6. AD/ADAS機能共通の作動状態

以下の5つのデータ型は，AD/ADAS機能オブジェクトが共通に用いるデータ型であり，本仕様内に定義する。LCAにおける各列挙子の具体的な意味は，位置付けと機能の節に示す。

```
type AutomationOperatingStateType = enum {
    Off,    // 機能が無効化されており，作動していない状態
    Standby,    // 機能が有効であり，作動の条件が整えば作動できるが，現在は作動していない状態
    Activating, // 作動開始の要求を受理し，作動開始の処理を行っている過渡状態
    Active, // 作動中。車両の運動への介入または警報の出力を現に行っている状態
    Suspended,  // 作動が一時的に解除され，中断している状態。直前の作動時の設定値を保持しており，抑止する要因が解消すれば作動を再開できる
```

```
    Deactivating,   // 作動終了の要求を受理し，作動終了の処理を行っている過渡状態
    Fallback,   // 縮退運転を実行している状態
    MRC,    // 最小リスク状態へ移行し，これを維持している状態
    Fault   // 故障により機能を提供できない状態
};
```

各機能オブジェクトが現にどの作動状態にあるかを表す型である。機能によっては現れない列挙子がある。

```
type AutomationAvailabilityType = enum {
    Available,  // 作動を開始または継続できる
    TemporarilyUnavailable, // 一時的に条件を満たさないため，作動を開始または継続できない（条件が回復すれば作動できる）
    Unavailable // 故障等により，作動を開始または継続できない
};
```

その機能がいま作動を開始または継続できるかを表す型である。作動しているか否かは作動状態operatingStateが表すものであり，本型はこれを表さない。

```
type AutomationUnavailableReasonType = enum {
    ODDOutside, // 走行環境が運行設計領域の外にある
    SensorDegraded, // センサの性能低下・遮蔽により，必要な認識の性能が得られない
    DependentFunctionFault, // この機能が用いている他のAD/ADAS機能が故障している（blockingFunctionがその機能を指す）
    DriverMonitoringUnavailable,    // 運転者の監視が行えない
    VehicleStateNotReady,   // 車速・変速機の位置・制動等の車両の状態が作動の条件を満たさない
    FunctionFault,  // この機能自体が故障している
    ControlOverridden,  // 運転者の操作または他の経路による車両の運動の制御により，作動が抑止されている
    UserDisabled,   // 利用者の設定により機能が無効にされている
    SupersededByOtherFunction,  // 他のAD/ADAS機能がその運転操作を担っているため作動しない（blockingFunctionがその機能を指す）
    CalibrationRequired,    // 較正が必要で作動できない
    Other   // その他の理由
};
```

availabilityがAvailable以外の値である場合に，作動を開始または継続できない理由を表す型である。機能によっては現れない列挙子がある。

```
type AutomationOutputLevelType = enum {
    Detected,   // 事象を検知したが，運転者への提示はまだ行っていない
    Advisory,   // 注意喚起（表示等の弱い提示）
    Warning,    // 警報（音・振動を伴う提示）
    Critical    // 緊急（車両運動への介入を伴う）
};
```

機能の出力の段階を表す型である。Detected，Advisory，Warning，Criticalの順に強い段階を表す。機

能によっては現れない段階がある。

```
type AutomationFunctionOutputType = struct {
    active : Bool,  // その機能が事象の検知または運転者への提示を現に行っているか
    level : AutomationOutputLevelType?  // 出力の段階。activeがtrueであり，かつ段階を
提供できる機能である場合にのみ値を持つ
};
```

機能が外部（運転者）に対して現に行っている出力を，有無と段階のみで表す型である。出力の判断の根拠（検知した車線の形状・物体の種別等）は本型に含まない。

5.5.8.7. AD/ADAS機能共通の故障の区分

以下のデータ型は，AD/ADAS機能オブジェクトが共通に用いるデータ型であり，本仕様内に定義する。LCAでは，アクチュエータの故障（ActuatorFault）は機能クラスLaneChangeExecutionを備えている場合にのみ，警報を提示する装置の故障（WarningDeviceFault）は機能クラスApproachWarningを備えている場合にのみ現れる。

```
type AutomationFaultCategoryType = enum {
    SensorFault,    // センサの故障
    ActuatorFault,  // アクチュエータの故障
    ComputationFault,   // 演算装置の故障
    CommunicationFault, // 車内通信の故障
    PowerFault, // 電源の異常
    WarningDeviceFault, // 警報を提示する装置の故障
    Other   // その他の故障
};
```

機能を提供できない故障の区分を表す型である。機能によっては現れない列挙子がある。故障の区分は，各機能オブジェクトの状態faultCategoryおよびFaultDetectedイベントの関連情報により渡される。

5.5.8.8. LCAの操作

```
type LCAOperationType = enum {
    Activate,   // 作動の開始の要求（サービスコールactivate）
    Deactivate  // 作動の終了の要求（サービスコールdeactivate）
};
```

アプリケーションから要求できる操作を表す型である。列挙子は，操作を行うサービスコールと1対1に対応する。すべてのアプリケーションが呼び出すことができるサービスコール（getConfig，getStatus，notify，unnotify，getEvent，lock，unlock）は，本型の対象としない。

5.5.8.9. 車線変更の進行

```
type LCALaneChangePhaseType = enum {
    Checking,   // 運転者の意図を受け付け，変更先の車線の安全を確認している
    Executing,  // 車線変更のための操舵を行っている
    Aborting    // 車線変更を中止し，元の車線へ戻るための操舵を行っている
```

```
};
```

車線変更の支援を現に行っている間の進行の段階を表す型である。車線変更の支援を行っていない間は，この型の状態は値を持たない。

```
type LCADirectionType = enum {
    Left,   // 自車の進行方向に対して左側の車線への変更
    Right   // 自車の進行方向に対して右側の車線への変更
};
```

車線変更の方向を表す型である。方向は，運転者が意図を表明した方向とする。

```
type LCAAbortReasonType = enum {
    ApproachingVehicle, // 変更先の車線に接近する車両を検知した
    DriverOverride, // 運転者の操舵・制動等の操作により中止した
    IntentWithdrawn,    // 運転者が意図を取り下げた（方向指示器の解除等）
    Timeout,    // 車線変更を開始できないまま，車両実装が定める時間を経過した
    ConditionLost,  // 車線を認識できない，車速が作動の対象となる範囲を外れた等，作動の条件を満たさなくなった
    ControlOverridden,  // 他の経路による横方向の運動の制御により中止した
    Deactivated,    // 作動の終了の要求または運転者の設定によりLCAが無効化された
    Other   // その他の理由
};
```

車線変更を中止した理由を表す型である。LaneChangeAbortedイベントの関連情報として渡される。

5.5.8.10. 作動の対象となる車速の範囲

```
type LCASpeedRangeType = struct {
    minimum : SpeedType,    // 作動の対象となる車速の下限。単位はkm/h
    maximum : SpeedType?    // 作動の対象となる車速の上限。単位はkm/h。上限を設けない場合は値を持たない
};
```

5.5.8.11. LCAの構成情報（getConfigが返すデータ型）

```
type LCAConfigType = struct {
    capabilities : enumSet(LCACapabilityType),
    riskClasses : enumSet(LCARiskType),
    supportedOperations : enumSet(LCAOperationType),
    displayName : String?,
    operatingSpeedRange : LCASpeedRangeType,
    timestampAccuracy : UInt32
};
```

5.5.8.12. LCA状態

```
type LCAStatusType = struct {
    timestamp : TimestampUsType,
    dataQualifier : DataQualifierType,
    operatingState : AutomationOperatingStateType,
    availability : AutomationAvailabilityType,
    unavailableReason : AutomationUnavailableReasonType?,
    blockingFunction : RelatedObjectType[],
    faultCategory : AutomationFaultCategoryType?,
    laneChangePhase : LCALaneChangePhaseType?,
    laneChangeDirection : LCADirectionType?,
    functionOutput : AutomationFunctionOutputType?,
    lockStatus : LockStatusType
};
```

呼び出し時点のLCAの状態の冪等なスナップショットである。

5.5.8.13. LCAイベント

```
type LCAEventKindType = enum {
    OperatingStateChanged,
    AvailabilityChanged,
    LaneChangePhaseChanged,
    LaneChangeCompleted,
    LaneChangeAborted,
    FunctionOutputChanged,
    FaultDetected,
    FaultRecovered,
    DataQualifierChanged,
    OwnLockRevoked,
    OtherLockReleased,
    EventOverflow,
    ConfigChanged
};
```

```
type LCAEventType = struct {
    eventInfo : variant(LCAEventKindType) {
        LaneChangeCompleted {
            direction : LCADirectionType
        },
        LaneChangeAborted {
            direction : LCADirectionType ,
            reason : LCAAbortReasonType
        },
        FaultDetected {
            faultCategory : AutomationFaultCategoryType
        },
```

```
        EventOverflow {
            noLostEvents : UInt32
        }
    }
};
```

5.5.9. サービスコールの詳細規定

getConfig

LCAの構成情報の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
config	LCAConfigType	LCAの構成情報

【エラーコード】

なし

【機能】

その車両のLCAが持つ機能，アプリケーションから要求できる操作の集合，および作動の対象となる車速の範囲を返す。

アプリケーションは，本サービスコールにより，画面を構成する前に，どの操作の要素を提示すべきかを決める。

getStatus

LCAの状態の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
status	LCAStatusType	LCAの状態

【エラーコード】

E_OBJECT_FAULT	本オブジェクトが故障しており，状態を提供できない
E_RISK_APPLICATION	アプリケーションがRiskPrivacyに対応していない

E_RISK_USER	ユーザがRiskPrivacyを承認していない

【機能】

LCAの現在の作動状態・利用可否・故障の区分・車線変更の進行の段階と方向・警報の出力の状況・ロック状態を返す。

呼び出しのたびに，その時点の状態の全体を返す冪等な応答である。値を読む前にdataQualifierを確認する。

activate

LCAの作動の開始の要求

【パラメータ】

priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_PARAMETER	priorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_LOCKED	本オブジェクトが，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている
E_OBJECT_STATUS	作動を開始できる作動状態でない（既に有効である，またはavailabilityがUnavailableであり，その理由が故障以外である）
E_OBJECT_FAULT	本オブジェクトが故障しており，要求を受け付けられない（unavailableReasonがFunctionFaultである場合を含む）
E_INFEASIBLE	availabilityがTemporarilyUnavailableであり，作動の条件以外の一時的な理由（故障等）により，現在はLCAを有効にできない
E_RISK_APPLICATION	アプリケーションがRiskCollisionに対応していない
E_RISK_USER	ユーザがRiskCollisionを承認していない

【機能】

LCAに作動の開始を要求する。本サービスコールは，LCAを有効にする操作に相当し，Offからの遷移を要求する。作動の条件（車速が作動の対象となる範囲にある等）が整っている場合はActiveへ，整っていない場合はStandbyへ遷移する。作動の条件が整っていないことは，本要求を受け付けられない理由とはならない。Offである間のavailabilityは，これらの作動の条件を反映しない。車線変更は開始されない。車線変更の支援は，Activeにおいて運転者が意図を表明したときに開始される。

正常終了は要求が受理されたことのみを意味し，作動が開始したことを意味しない。遷移の完了は，getStatusまたはOperatingStateChangedイベントにより確認する。

作動を開始できない場合の応答は，利用可否availabilityとその理由unavailableReasonにより定まる。

availabilityがTemporarilyUnavailableである場合はE_INFEASIBLEエラー，Unavailableであり unavailableReasonがFunctionFaultである場合はE_OBJECT_FAULTエラー，Unavailableでありそれ以外の理由である場合はE_OBJECT_STATUSエラーとなる。

本サービスコールは，構成情報supportedOperationsにActivateを含む場合にのみ用いることができる。

deactivate

LCAの作動の終了の要求

【パラメータ】

priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_PARAMETER	priorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_LOCKED	本オブジェクトが，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている
E_OBJECT_STATUS	作動を終了できる作動状態でない（既にOffである等）
E_OBJECT_FAULT	本オブジェクトが故障しており，要求を受け付けられない
E_RISK_APPLICATION	アプリケーションがRiskCollisionに対応していない
E_RISK_USER	ユーザがRiskCollisionを承認していない

【機能】

LCAに作動の終了を要求する。本サービスコールは，LCAを無効にする操作に相当し，Offへの遷移を要求する。車線変更の支援が行われていた場合，車線変更は中止され，LaneChangeAbortedイベント（関連情報reasonはDeactivated）が発生する。車線変更のための操舵が行われていた場合，制御は終了し，操舵は運転者に戻る。

正常終了は要求が受理されたことのみを意味し，作動が終了したことを意味しない。遷移の完了は，getStatusまたはOperatingStateChangedイベントにより確認する。

本サービスコールは，構成情報supportedOperationsにDeactivateを含む場合にのみ用いることができる。



車線変更のための操舵の途中で本サービスコールによりLCAが無効になると，操舵は運転者に委ねられる。運転者が引き継ぐ準備を終えていない状態で作動を終了させると，隣接する車線の車両との衝突のリスクを生じる。また，LCAが無効になると，変更先の車線に接近する車両についての警報も出力されなくなるため，アプリケーションは，無効にしたことを運転者が認識できるようにしなければならない。

lock

LCAのロックの取得

【パラメータ】

priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_PARAMETER	priorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_LOCKED	本オブジェクトが，priorityと同じかそれより高い優先度でロックされている
E_OBJECT_FAULT	本オブジェクトが故障しており，ロックを取得できない

【機能】

本オブジェクトを，指定した操作優先度でロックする。ロックを取得したアプリケーションは，状態を変化させるサービスコール（activate，deactivate）を，同じかそれより低い優先度の他のアプリケーションによる操作から保護する。

既存のロックより高い優先度でロックを取得した場合，既存のロックは強制解除され，そのロックを取得していたアプリケーションにはOwnLockRevokedイベントが通知される。

運転者による直接の操作は，ロックおよび操作優先度によって妨げられない。

unlock

LCAのロックの解除

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_OBJECT_STATUS	自アプリケーションが，本オブジェクトをロックしていない

【機能】

自アプリケーションが取得した本オブジェクトのロックを解除する。

notify

LCAイベントの通知要求

【パラメータ】

eventFilter	enumSet (LCAEventKindType)?	通知するイベントの種類の集合

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【エラーコード】

E_INVALID_PARAMETER	eventFilterが有効範囲外
E_RISK_APPLICATION	アプリケーションがRiskPrivacyに対応していない
E_RISK_USER	ユーザがRiskPrivacyを承認していない

【機能】

LCAで発生したeventFilterで指定したイベントの通知を要求する。イベントを通知するためのイベントキューを生成し，その通知ハンドルをnotifyHandleに返す。

eventFilterを省略した場合，すべての種類のイベントの通知を要求する。

車線変更の発生の観測からも運転の傾向を推定できるため，本サービスコールもRiskPrivacyのリスク制御の対象である。

unnotify

LCAイベントの通知停止

【パラメータ】

notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正

【機能】

notifyHandleで指定した通知を停止する。指定した通知用のイベントキューは削除される。

getEvent

LCAイベントの取得

【パラメータ】

notifyHandle	NotifyHandleType	イベントキューを識別するための通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
event	LCAEventType	取得したイベント

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正
E_NO_DATA	イベントキューが空
E_RISK_APPLICATION	アプリケーションがRiskPrivacyに対応していない
E_RISK_USER	ユーザがRiskPrivacyを承認していない

【機能】

notifyHandleで指定した通知用のイベントキューから，先頭のイベントを取り出す。取り出したイベントは，イベントキューから削除される。

イベントの内容からも運転の傾向を推定できるため，本サービスコールもRiskPrivacyのリスク制御の対象である。



イベントがパラメータ等で渡されてくるプログラミング環境向けの物理APIでは，用意する必要がない。

5.6. Vehicle.ADADAS.AEB（略称：AEB）

5.6.1. 位置付けと機能

Vehicle.ADADAS.AEB（略称：AEB）は，車両に搭載された衝突被害軽減ブレーキ（AEB）を参照・操作するためのシングルインスタンスオブジェクトである。作動の開始・終了の要求，作動の時機（感度）の設定，および現在の作動状態・警報と制動の介入の状況・設定値の参照を提供する。

本オブジェクトは，Vehicle.ADADASの目録に，機能の種別AEB，提供するオブジェクトの完全修飾名Vehicle.ADADAS.AEBとして現れる。DDTの責任主体・運行設計領域の充足状態・縮退運転・最小リスク状態など，AD/ADASシステム全体について一つに定まる状態は，本オブジェクトではなくVehicle.ADADASが提供する。

5.6.1.1. AEBの位置付けと性質

AEBは，外界のセンサ（レーダ・カメラ等）により前方の物体との衝突の危険を検知し，運転者への警報と，自動的な制動の介入を行う運転支援の機能である。AEBは常時待機して衝突の危険を監視し，衝突の危険を検知した場合にのみ介入する。AEBの作動中も，動的運転タスク（DDT）の責任は運転者にある。

本オブジェクトは，AEBが用いるセンサの構成，衝突の危険の判定のアルゴリズム，制動の制御の方法を規定しない。検知した物体の種別・位置・相対速度，衝突までの予測時間といった判定の根拠は本オブジェクトでは扱わない。本オブジェクトが提供するのは，機能が有効であるか・現に介入しているか（operatingState），いま監視と介入を継続できるか（availability），現に行っている警報と制動の介入の段階（functionOutput），作動の時機の設定（sensitivity），および故障の区分（faultCategory）である。

検知の対象（車両・歩行者・自転車等）ごとに機能を有効・無効にする手段は持たない。

AEBは安全のために介入する機能であり，Vehicle.Motionによる車両の運動の制御と競合した場合は，AEBの介入が優先する。その規定は「AEBに対する制御の調停」の節に示す。

5.6.1.2. 作動状態

本オブジェクトの作動状態operatingStateおよび利用可否availabilityは，Vehicle.ADADASが規定するAD/ADAS機能共通の作動状態モデルに従う。本節では，AEBに固有の制約のみを規定する。

AEBにおける各作動状態の意味は，次のとおりである。

- Off … AEBが無効化されており，衝突の危険を監視していない状態。アプリケーションによる作動の終了の要求，および運転者の設定のいずれによっても，この状態となる。
- Standby … AEBが有効であり，衝突の危険を監視しているが，警報および制動の介入を行っていない状態。AEBが有効である間の通常の状態である。
- Activating … 作動開始の要求（activate）を受理し，Standbyへ遷移する処理を行っている過渡状態。
- Active … 衝突の危険を検知し，運転者への警報または制動の介入を現に行っている状態。
- Suspended … 介入が，運転者の操作により抑止されている状態。
- Deactivating … 作動終了の要求（deactivate）を受理し，Offへ遷移する処理を行っている過渡状態。
- Fault … 故障により機能を提供できない状態。故障の区分は状態faultCategoryが表す。

AEBでは，FallbackおよびMRCの列挙子は現れない。縮退運転および最小リスク状態への移行は，DDTの責任を持つ機能が行うものであり，運転支援の機能であるAEBはこれを行わないためである。

StandbyとActiveの間の遷移は，衝突の危険の検知により車両が行う。アプリケーションがActiveへの遷移を要求する手段は無い。衝突の危険が解消した場合，AEBはActiveからStandbyへ遷移し，監視を続ける。

AEBが介入している間（Active）に，運転者の操作（アクセルペダルの踏み込み，操舵による回避等）により介入が抑止された場合，operatingStateはSuspended，availabilityはTemporarilyUnavailable，unavailableReasonはControlOverriddenとなる。抑止する要因が解消すると，AEBはStandbyへ遷移し，監視を続ける。運転者の操作が介入を抑止するか否か，およびその条件は車両による。

AEBが介入できる車速の範囲等の作動の条件は，AEBが有効である間（Standby）のavailabilityおよびunavailableReasonが表す。Standbyである間に車速等の車両の状態が作動の条件を満たさない場合，operatingStateはStandbyのまま，availabilityはTemporarilyUnavailable，unavailableReasonはVehicleStateNotReadyとなる。

operatingStateがOffである間，availabilityは，車速等の作動の条件によらず，AEBを有効にできるか（故障の有無等）のみを表す。すなわち，車速等の車両の状態が作動の条件を満たさないことのみを理由として，availabilityがTemporarilyUnavailableとなることはない。アプリケーションは，停車中を含

め，作動の開始（activate）を要求でき，受理された場合，operatingStateはStandbyへ遷移する。

運転者の設定によりAEBが無効化されている場合も，operatingStateはOffとなる。このとき，AEBを有効にできる状態であればavailabilityはAvailableであり，アプリケーションは作動の開始（activate）を要求できる。この要求は，後勝ちの原則に従い，運転者の設定より後に行われた操作として受理される。

車両によっては，車両の始動のたびにAEBを有効な状態に戻す。アプリケーションは，作動の終了の要求によるOffが，次の始動以降も保たれることを前提にしてはならない。作動状態の変化は，OperatingStateChangedイベントにより知ることができる。

車両が，アプリケーションからの作動の開始・終了そのものを認めない場合は，構成情報supportedOperationsにActivateおよびDeactivateを含めないことで表す。

5.6.1.3. 警報と制動の介入の表し方

警報と制動の介入の状況は，状態functionOutputが表す。activeは，衝突の危険を検知しているか，または警報もしくは制動の介入を現に行っているかを表す。AEBでは，activeがtrueである間，levelは常に値を持つ。

levelの各段階は，AEBにおいて次の意味を持つ。

- Detected ... 衝突の危険を検知したが，運転者への警報および制動の介入をまだ行っていない。
- Advisory ... 表示による注意喚起を行っている。制動の介入は行っていない。
- Warning ... 音・振動を伴う警報を行っている。制動の介入は行っていない。
- Critical ... 制動の介入を行っている。警報を伴うか否かによらない。予備的な制動（部分制動）と，衝突を回避または被害を軽減するための制動（緊急制動）のいずれもこの段階で表し，両者を区別しない。

これにより，警報のみを行っているか（AdvisoryまたはWarning），制動の介入を行っているか（Critical）を区別できる。

levelがAdvisory，WarningまたはCriticalである間，operatingStateはActiveである。levelがDetectedである間，operatingStateはStandbyのままである。operatingStateがStandbyおよびActive以外である間，functionOutputのactiveは常にfalseである。

functionOutputの値が変化したときは，FunctionOutputChangedイベントが発生する。イベントは事象の発生のみを伝え，値そのものは運ばない。値は，イベントを受けたアプリケーションがgetStatusにより取得する。



検知した物体の種別・位置・相対速度は本オブジェクトでは提供しない。これらはVehicle.SurroundModelの担当である。

5.6.1.4. 作動の時機（感度）の設定

AEBは，衝突の危険を運転者に提示する時機を，早い（Early），標準（Standard），遅い（Late）の段階で切り替えられることがある。切り替えられるか否かは機能クラスSensitivityAdjustableが，切り替えられる段階は構成情報sensitivityLevelsが宣言する。現在の設定は状態sensitivityが表す。

アプリケーションから感度を設定できるか否かは，機能クラスとは別に，構成情報supportedOperationsがSetSensitivityを含むか否かで表す。機能クラスSensitivityAdjustableを備えるが

supportedOperationsにSetSensitivityを含まない車両では，アプリケーションは設定を参照できるが変更できない。

5.6.1.5. 操作の可否の宣言

本オブジェクトに対してアプリケーションが行える操作は，構成情報supportedOperationsが宣言する。supportedOperationsの列挙子は，操作を行うサービスコールと1対1に対応する。アプリケーションは，画面を構成する前にsupportedOperationsを参照し，宣言されていない操作の要素を提示してはならない。宣言されていない操作のサービスコールを呼び出した場合，E_NOT_SUPPORTEDエラーとなる。

supportedOperationsが操作の可否（恒久的な可否）を表すのに対し，availabilityは，いまその操作を受け付けられるか（一時的な可否）を表す。この2つは別の判定であり，アプリケーションは，前者により要素を提示するか否かを，後者により要素を操作可能とするか否かを決める。

5.6.1.6. AEBに対する制御の調停

本オブジェクトに対する制御の調停には，標準的な調停機能（後勝ちの原則）が適用される。すなわち，あるアプリケーションが作動を開始した後に，別のアプリケーションが作動の終了を要求した場合，後の要求が有効となる。

本オブジェクトは，標準的なロック機能を持つLockableObjectである。状態を変化させるサービスコール（activate，deactivate，setSensitivity）およびロックを取得するサービスコール（lock）は，操作優先度priorityを持つ。ロック中の本オブジェクトに対して，ロックと同じかそれより低い優先度で操作した場合はE_OBJECT_LOCKEDエラーとなり，より高い優先度で操作した場合は，既存のロックが強制解除されて操作が実行される。アプリケーションが使用できる最高値より高い優先度を指定した場合はE_DENIED_PRIORITYエラー，操作優先度が有効範囲外である場合はE_INVALID_PARAMETERエラーとなる。

自アプリケーションのロックが強制解除された場合はOwnLockRevokedイベントが，他のアプリケーション等によりロックが解除された場合はOtherLockReleasedイベントが発生する。作動状態がFaultとなった場合，ロックは解除され，その間は新たにロックを取得することができない。

運転者による直接の操作（車載の設定画面等による有効化・無効化・感度の変更，アクセルペダル・操舵による介入の抑止）は，本APIを用いない一つのアプリケーションによる操作と見なし，後勝ちの原則に従う。この操作は，アプリケーションが取得したロックおよび操作優先度によって妨げられることはなく，常に有効となる。運転の責任を負うのは運転者であり，アプリケーションによる排他が運転者の操作に優先してはならないためである。

Vehicle.Motionは，アプリケーションが車両の運動そのものを制御するためのオブジェクトであり，本オブジェクトは，車両に搭載されたAEBの有効・無効と設定を扱うものである。両者は別の経路である。AEBは安全のために介入する機能であるため，AEBの制動の介入は，Vehicle.Motionによる縦方向の運動の制御に優先する。AEBが介入している間（Active）にVehicle.Motionにより縦方向の制御が指示された場合，AEBの介入が継続し，Vehicle.Motionの指示は実行されない。AEBが介入していない間（Standby）は，Vehicle.Motionの指示がそのまま実行される。

5.6.1.7. 操作・参照に伴うリスク

AEBの作動を終了させると，衝突の危険に対する警報と制動の介入が行われなくなり，衝突を回避または被害を軽減できなくなる。作動の開始は，衝突の危険の誤った検知による予期しない制動を生じうるものであり，後続車との衝突につながりうる。感度の変更は，警報の時機を変え，運転者の回避の操作の遅れにつながりうる。これらをリスククラスRiskCollisionとして定義し，activate，deactivateおよびsetSensitivityをリスク制御の対象とする。

AEBの作動状態および警報と制動の介入の履歴は，運転者がいつどのように運転していたか（衝突の危険

に至った状況を含む）を第三者が推定できる情報である。これをリスククラスRiskPrivacyとして定義し，状態を参照するサービスコール（getStatus），イベントの通知要求（notify）およびイベントの取得（getEvent）をリスク制御の対象とする。

いずれのリスククラスについても，当該リスクに対応していないアプリケーションからの呼び出しはE_RISK_APPLICATIONエラー，ユーザが当該リスクを承認していない場合はE_RISK_USERエラーとなる。

構成情報の参照（getConfig）は，車両に搭載されたAEBの静的な性質を返すものであり，リスク制御の対象としない。

5.6.2. 機能クラス

AEBは，以下の機能クラスを持つ。

SensitivityAdjustable	衝突の危険を運転者に提示する時機（感度）を段階的に切り替えることができる（状態sensitivity，構成情報sensitivityLevels）

5.6.3. リスククラス

AEBは，以下のリスククラスを持つ。

RiskCollision	AEBの作動の開始・終了および感度の変更により警報と制動の介入の有無・時機が変わり，衝突を回避または被害を軽減できなくなる，または予期しない制動により衝突を生じるリスク
RiskPrivacy	走行状態および運転者の運転への関与状態の開示により，個人の行動を推定できることに伴うプライバシのリスク

5.6.4. 構成情報

AEBは，以下の構成情報を持つ。

capabilities	enumSet (AEBCapabilityType)	その車両のAEBが持つ機能
riskClasses	enumSet(AEBRiskType)	車両およびビークルOSで対応できていないリスククラスの集合
supportedOperations	enumSet (AEBOperationType)	アプリケーションから要求できる操作の集合。空集合となることがある
displayName	String?	ユーザに表示する呼称を取得するためのキー。車両が呼称を公開する場合にのみ値を持つ
sensitivityLevels	enumSet (AEBSensitivityType)	切り替えられる感度の段階の集合。機能クラスSensitivityAdjustableを備える場合は2つ以上の要素を持ち，備えない場合は空集合となる
timestampAccuracy	UInt32	この車両のタイムスタンプ誤差上限の宣言。単位はμs

上記は，YAML形式で記述した構成記述ファイル中に記述する形式である。本オブジェクトはシングルインスタンスオブジェクトであるため，構成記述では，車両を表すノードの下に置かれたADADASノードの下に，オブジェクトを表すノードAEBを置き，その下に上記の構成情報を記述する。このノードは，

Vehicle.ADADASの目録において，providingObjectがVehicle.ADADAS.AEBである要素に対応する。

riskClassesには，車両およびビークルOSで対応できていない（危険を防止できていない）リスククラスの集合を記述する。

構成情報を参照するサービスコールは，この構成情報に対して，次の変換を行った構成情報を返す。

- displayNameに対しては，それを元に現在のロケールに対応する呼称の文字列を取得し，その文字列を返す。

5.6.5. 状態

AEBは，以下の状態を持つ。

timestamp	TimestampUsType	状態の測定時刻。単位はμs。車内共通時刻基盤に基づく
dataQualifier	DataQualifierType	このサンプルの品質。InvalidまたはUnavailableの場合，以下の状態の値を読んではならない。値の変化はDataQualifierChangedイベントにより通知される
operatingState	AutomationOperatingStateType	現在の作動状態。AEBではFallbackおよびMRCは現れない
availability	AutomationAvailabilityType	いま衝突の危険の監視と介入を開始または継続できるか
unavailableReason	AutomationUnavailableReasonType?	監視と介入を開始または継続できない理由。availabilityがAvailable以外の値である場合にのみ値を持つ
blockingFunction	RelatedObjectType[]	監視と介入を妨げている機能オブジェクト。unavailableReasonがDependentFunctionFaultまたはSupersededByOtherFunctionである場合にのみ要素を持つ
faultCategory	AutomationFaultCategoryType?	故障の区分。operatingStateがFaultである場合にのみ値を持つ
functionOutput	AutomationFunctionOutputType	警報と制動の介入の状況。activeがtrueである間，levelは常に値を持つ
sensitivity	AEBSensitivityType?	現在設定されている感度。機能クラスSensitivityAdjustableを備えている場合にのみ値を持つ
lockStatus	LockStatusType	ロック状態

状態は，呼び出し時点のAEBの状態を返す冪等なスナップショットである。値を読む前にdataQualifierを確認する。

AEBが有効であるか，現に介入しているかは，operatingStateにより判定する。警報のみを行っているか，制動の介入を行っているかは，functionOutputのlevelにより判定する。operatingStateとavailabilityの組合せの規則は，Vehicle.ADADASが規定するAD/ADAS機能共通の作動状態モデルによる。

故障により機能を提供できない場合，operatingStateはFaultとなり，faultCategoryがその区分を表す。

これにより，FaultDetectedイベントを受け取っていないアプリケーションも，状態の参照のみで故障の区分を知ることができる。本オブジェクト自体が状態を提供できない場合は，getStatusがE_OBJECT_FAULTエラーを返す。

5.6.6. サービスコール一覧

AEBに対するサービスコールは以下の通り。

getConfig	AEBの構成情報の参照
getStatus	AEBの状態の参照
activate	AEBの作動の開始の要求
deactivate	AEBの作動の終了の要求
setSensitivity	AEBの感度の設定
lock	AEBのロックの取得
unlock	AEBのロックの解除
notify	AEBイベントの通知要求
unnotify	AEBイベントの通知停止
getEvent	AEBイベントの取得

5.6.7. イベント

イベントは，事象が発生したことのみを伝え，状態の値そのものを運ばない。ただし，事象の発生時点の情報として有用なもの（故障の区分等）は，イベントの関連情報としても渡す。アプリケーションは，イベントを受けた後，必要に応じてgetStatusにより値を取得する。

状態の取得は，要求応答（getStatus）と，各状態項目の変化イベントによる。

AEBに対するイベント種別は以下の通り。

OperatingStateChanged	作動状態の変化（アプリケーションの要求および運転者の設定による有効化・無効化，介入の開始・終了・抑止を含む）
AvailabilityChanged	衝突の危険の監視と介入を開始または継続できるかの変化（監視と介入を妨げている理由の変化を含む）
FunctionOutputChanged	警報と制動の介入の状況（状態functionOutput）の変化。検知の開始・段階の変化・終了のいずれでも発生する
SettingChanged	感度の設定値の変化（アプリケーションの要求および運転者の設定による変更を含む）
FaultDetected	故障の検知（AEBの機能を提供できない状態）
FaultRecovered	故障からの回復
DataQualifierChanged	状態の品質（状態dataQualifier）の変化
OwnLockRevoked	自アプリケーションのロックの強制解除
OtherLockReleased	他のアプリケーション等によるロック解除
EventOverflow	イベントキューのオーバフロー

ConfigChanged		構成情報の変更（ソフトウェアの更新による対応する操作・切り替えられる感度の段階の変更）

この内，イベント種別がFaultDetectedの場合には，以下の関連情報を持つ。

faultCategory	AutomationFaultCategoryType	故障の区分

また，イベント種別がEventOverflowの場合には，以下の関連情報を持つ。

noLostEvents	UInt32	失われたイベントの数

5.6.8. データ型の定義

5.6.8.1. AEBの機能クラス

```
type AEBCapabilityType = enum {
    SensitivityAdjustable
};
```

5.6.8.2. AEBのリスククラス

```
type AEBRiskType = enum {
    RiskCollision,
    RiskPrivacy
};
```

5.6.8.3. タイムスタンプ

```
type TimestampUsType = UInt64;  // 車内共通時刻基盤の起点を基準とする測定時刻。単位はμs。複数のビークルAPIで共通に用いる共通データ型であり，本仕様内に定義する。
```

5.6.8.4. サンプル品質の種別

```
type DataQualifierType = enum {
    Normal, // 正常
    Unavailable,    // 利用不可
    Degraded,   // 劣化
    TestMode,   // テストモード
    Invalid // 無効（値を読んではならない）
};
```

複数のビークルAPIで共通に用いる共通データ型であり，本仕様内に定義する。

5.6.8.5. AD/ADAS機能共通の作動状態

以下の5つのデータ型は，AD/ADAS機能オブジェクトが共通に用いるデータ型であり，本仕様内に定義する。AEBにおける各列挙子の具体的な意味は，位置付けと機能の節に示す。

```
type AutomationOperatingStateType = enum {
    Off,    // 機能が無効化されており，作動していない状態
    Standby,    // 機能が有効であり，作動の条件が整えば作動できるが，現在は作動していない状態
    Activating, // 作動開始の要求を受理し，作動開始の処理を行っている過渡状態
    Active, // 作動中。車両の運動への介入または警報の出力を現に行っている状態
    Suspended,  // 作動が一時的に解除され，中断している状態。直前の作動時の設定値を保持しており，抑止する要因が解消すれば作動を再開できる
    Deactivating,   // 作動終了の要求を受理し，作動終了の処理を行っている過渡状態
    Fallback,   // 縮退運転を実行している状態
    MRC,    // 最小リスク状態へ移行し，これを維持している状態
    Fault   // 故障により機能を提供できない状態
};
```

各機能オブジェクトが現にどの作動状態にあるかを表す型である。機能によっては現れない列挙子がある。

```
type AutomationAvailabilityType = enum {
    Available,  // 作動を開始または継続できる
    TemporarilyUnavailable, // 一時的に条件を満たさないため，作動を開始または継続できない（条件が回復すれば作動できる）
    Unavailable // 故障等により，作動を開始または継続できない
};
```

その機能がいま作動を開始または継続できるかを表す型である。作動しているか否かは作動状態operatingStateが表すものであり，本型はこれを表さない。

```
type AutomationUnavailableReasonType = enum {
    ODDOutside, // 走行環境が運行設計領域の外にある
    SensorDegraded, // センサの性能低下・遮蔽により，必要な認識の性能が得られない
    DependentFunctionFault, // この機能が用いている他のAD/ADAS機能が故障している（blockingFunctionがその機能を指す）
    DriverMonitoringUnavailable,    // 運転者の監視が行えない
    VehicleStateNotReady,   // 車速・変速機の位置・制動等の車両の状態が作動の条件を満たさない
    FunctionFault,  // この機能自体が故障している
    ControlOverridden,  // 運転者の操作または他の経路による車両の運動の制御により，作動が抑止されている
    UserDisabled,   // 利用者の設定により機能が無効にされている
    SupersededByOtherFunction,  // 他のAD/ADAS機能がその運転操作を担っているため作動しない（blockingFunctionがその機能を指す）
    CalibrationRequired,    // 較正が必要で作動できない
    Other   // その他の理由
```

```
};
```

availabilityがAvailable以外の値である場合に，作動を開始または継続できない理由を表す型である。機能によっては現れない列挙子がある。

```
type AutomationOutputLevelType = enum {
    Detected,   // 事象を検知したが，運転者への提示はまだ行っていない
    Advisory,   // 注意喚起（表示等の弱い提示）
    Warning,    // 警報（音・振動を伴う提示）
    Critical    // 緊急（車両運動への介入を伴う）
};
```

機能の出力の段階を表す型である。Detected，Advisory，Warning，Criticalの順に強い段階を表す。機能によっては現れない段階がある。

```
type AutomationFunctionOutputType = struct {
    active : Bool,  // その機能が事象の検知または運転者への提示を現に行っているか
    level : AutomationOutputLevelType?  // 出力の段階。activeがtrueであり，かつ段階を
提供できる機能である場合にのみ値を持つ
};
```

機能が外部（運転者）に対して現に行っている出力を，有無と段階のみで表す型である。出力の判断の根拠（検知した車線の形状・物体の種別等）は本型に含まない。

5.6.8.6. AD/ADAS機能共通の故障の区分

以下のデータ型は，AD/ADAS機能オブジェクトが共通に用いるデータ型であり，本仕様内に定義する。

```
type AutomationFaultCategoryType = enum {
    SensorFault,    // センサの故障
    ActuatorFault,  // アクチュエータの故障
    ComputationFault,   // 演算装置の故障
    CommunicationFault, // 車内通信の故障
    PowerFault, // 電源の異常
    WarningDeviceFault, // 警報を提示する装置の故障
    Other   // その他の故障
};
```

機能を提供できない故障の区分を表す型である。機能によっては現れない列挙子がある。故障の区分は，各機能オブジェクトの状態faultCategoryおよびFaultDetectedイベントの関連情報により渡される。

5.6.8.7. AEBの操作

```
type AEBOperationType = enum {
    Activate,   // 作動の開始の要求（サービスコールactivate）
    Deactivate, // 作動の終了の要求（サービスコールdeactivate）
    SetSensitivity  // 感度の設定（サービスコールsetSensitivity）
```

```
};
```

アプリケーションから要求できる操作を表す型である。列挙子は，操作を行うサービスコールと1対1に対応する。すべてのアプリケーションが呼び出すことができるサービスコール（getConfig，getStatus，notify，unnotify，getEvent，lock，unlock）は，本型の対象としない。

5.6.8.8. AEBの感度

```
type AEBSensitivityType = enum {
    Early,  // 標準より早い時機に衝突の危険を運転者に提示する
    Standard,   // 標準の時機に衝突の危険を運転者に提示する
    Late    // 標準より遅い時機に衝突の危険を運転者に提示する
};
```

衝突の危険を運転者に提示する時機の段階を表す型である。各段階に対応する具体的な時機は車両による。車両によっては現れない段階がある。

5.6.8.9. AEBの構成情報（getConfigが返すデータ型）

```
type AEBConfigType = struct {
    capabilities : enumSet(AEBCapabilityType),
    riskClasses : enumSet(AEBRiskType),
    supportedOperations : enumSet(AEBOperationType),
    displayName : String?,
    sensitivityLevels : enumSet(AEBSensitivityType),
    timestampAccuracy : UInt32
};
```

5.6.8.10. AEB状態

```
type AEBStatusType = struct {
    timestamp : TimestampUsType,
    dataQualifier : DataQualifierType,
    operatingState : AutomationOperatingStateType,
    availability : AutomationAvailabilityType,
    unavailableReason : AutomationUnavailableReasonType?,
    blockingFunction : RelatedObjectType[],
    faultCategory : AutomationFaultCategoryType?,
    functionOutput : AutomationFunctionOutputType,
    sensitivity : AEBSensitivityType?,
    lockStatus : LockStatusType
};
```

呼び出し時点のAEBの状態の冪等なスナップショットである。

5.6.8.11. AEBイベント

```
type AEBEventKindType = enum {
    OperatingStateChanged,
    AvailabilityChanged,
    FunctionOutputChanged,
    SettingChanged,
    FaultDetected,
    FaultRecovered,
    DataQualifierChanged,
    OwnLockRevoked,
    OtherLockReleased,
    EventOverflow,
    ConfigChanged
};
```

```
type AEBEventType = struct {
    eventInfo : variant(AEBEventKindType) {
        FaultDetected {
            faultCategory : AutomationFaultCategoryType
        },
        EventOverflow {
            noLostEvents : UInt32
        }
    }
};
```

5.6.9. サービスコールの詳細規定

getConfig

AEBの構成情報の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
config	AEBConfigType	AEBの構成情報

【エラーコード】

なし

【機能】

その車両のAEBが持つ機能，アプリケーションから要求できる操作の集合，および切り替えられる感度の段階を返す。

アプリケーションは，本サービスコールにより，画面を構成する前に，どの操作の要素を提示すべきかと，感度の選択肢を決める。

getStatus

AEBの状態の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
status	AEBStatusType	AEBの状態

【エラーコード】

E_OBJECT_FAULT	本オブジェクトが故障しており，状態を提供できない
E_RISK_APPLICATION	アプリケーションがRiskPrivacyに対応していない
E_RISK_USER	ユーザがRiskPrivacyを承認していない

【機能】

AEBの現在の作動状態・利用可否・故障の区分・警報と制動の介入の状況・感度・ロック状態を返す。

呼び出しのたびに，その時点の状態の全体を返す冪等な応答である。値を読む前にdataQualifierを確認する。

activate

AEBの作動の開始の要求

【パラメータ】

priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_PARAMETER	priorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値

E_OBJECT_LOCKED	本オブジェクトが，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている
E_OBJECT_STATUS	作動を開始できる作動状態でない（既に有効である，またはavailabilityがUnavailableであり，その理由が故障以外である）
E_OBJECT_FAULT	本オブジェクトが故障しており，要求を受け付けられない（unavailableReasonがFunctionFaultである場合を含む）
E_INFEASIBLE	availabilityがTemporarilyUnavailableであり，作動の条件以外の一時的な理由（故障等）により，現在はAEBを有効にできない
E_RISK_APPLICATION	アプリケーションがRiskCollisionに対応していない
E_RISK_USER	ユーザがRiskCollisionを承認していない

【機能】

AEBに作動の開始を要求する。本サービスコールは，AEBを有効にする操作に相当し，OffからStandbyへの遷移を要求する。

正常終了は要求が受理されたことのみを意味し，作動が開始したことを意味しない。遷移の完了は，getStatusまたはOperatingStateChangedイベントにより確認する。

作動を開始できない場合の応答は，利用可否availabilityとその理由unavailableReasonにより定まる。availabilityがTemporarilyUnavailableである場合はE_INFEASIBLEエラー，Unavailableでありunavailable ReasonがFunctionFaultである場合はE_OBJECT_FAULTエラー，Unavailableでありそれ以外の理由である場合はE_OBJECT_STATUSエラーとなる。

本サービスコールは，構成情報supportedOperationsにActivateを含む場合にのみ用いることができる。

deactivate

AEBの作動の終了の要求

【パラメータ】

priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_PARAMETER	priorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_LOCKED	本オブジェクトが，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている
E_OBJECT_STATUS	作動を終了できる作動状態でない（既にOffである等）
E_OBJECT_FAULT	本オブジェクトが故障しており，要求を受け付けられない
E_RISK_APPLICATION	アプリケーションがRiskCollisionに対応していない

E_RISK_USER	ユーザがRiskCollisionを承認していない

【機能】

AEBに作動の終了を要求する。本サービスコールは，AEBを無効にする操作に相当し，Offへの遷移を要求する。

正常終了は要求が受理されたことのみを意味し，作動が終了したことを意味しない。遷移の完了は，getStatusまたはOperatingStateChangedイベントにより確認する。

本サービスコールは，構成情報supportedOperationsにDeactivateを含む場合にのみ用いることができる。



本サービスコールによりAEBが無効になると，衝突の危険に対する警報と制動の介入は行われなくなる。介入中（Active）に無効にした場合は，行っている制動の介入も終了する。アプリケーションは，無効にしたことを運転者が認識できるようにしなければならない。

setSensitivity

AEBの感度の設定

【パラメータ】

sensitivity	AEBSensitivityType	設定する感度。構成情報sensitivityLevelsに含まれる値でなければならない
priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_PARAMETER	sensitivityが構成情報sensitivityLevelsに含まれない，またはpriorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_LOCKED	本オブジェクトが，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている
E_OBJECT_STATUS	感度を設定できる作動状態でない（過渡状態（Activating，Deactivating）にある）
E_OBJECT_FAULT	本オブジェクトが故障しており，要求を受け付けられない
E_RISK_APPLICATION	アプリケーションがRiskCollisionに対応していない
E_RISK_USER	ユーザがRiskCollisionを承認していない

【機能】

AEBが衝突の危険を運転者に提示する時機（感度）を設定する。設定は，作動状態がFaultでなく，かつ過渡状態にない間，いつでも行うことができる。作動状態がOffである間に設定した値も保持される。

正常終了は要求が受理されたことのみを意味する。設定値の変化はSettingChangedイベントにより通知される。

過渡状態（Activating，Deactivating）にある間は，E_OBJECT_STATUSエラーとなる。要求は留保されないため，アプリケーションは，OperatingStateChangedイベントにより遷移の完了を確認した上で，あらためて要求する。

本サービスコールは，機能クラスSensitivityAdjustableを備え，かつ構成情報supportedOperationsにSetSensitivityを含む場合にのみ用いることができる。

lock

AEBのロックの取得

【パラメータ】

priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_PARAMETER	priorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_LOCKED	本オブジェクトが，priorityと同じかそれより高い優先度でロックされている
E_OBJECT_FAULT	本オブジェクトが故障しており，ロックを取得できない

【機能】

本オブジェクトを，指定した操作優先度でロックする。ロックを取得したアプリケーションは，状態を変化させるサービスコール（activate，deactivate，setSensitivity）を，同じかそれより低い優先度の他のアプリケーションによる操作から保護する。

既存のロックより高い優先度でロックを取得した場合，既存のロックは強制解除され，そのロックを取得していたアプリケーションにはOwnLockRevokedイベントが通知される。

運転者による直接の操作は，ロックおよび操作優先度によって妨げられない。

unlock

AEBのロックの解除

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_OBJECT_STATUS	自アプリケーションが，本オブジェクトをロックしていない

【機能】

自アプリケーションが取得した本オブジェクトのロックを解除する。

notify

AEBイベントの通知要求

【パラメータ】

eventFilter	enumSet (AEBEventKindType)?	通知するイベントの種類の集合

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【エラーコード】

E_INVALID_PARAMETER	eventFilterが有効範囲外
E_RISK_APPLICATION	アプリケーションがRiskPrivacyに対応していない
E_RISK_USER	ユーザがRiskPrivacyを承認していない

【機能】

AEBで発生したeventFilterで指定したイベントの通知を要求する。イベントを通知するためのイベントキューを生成し，その通知ハンドルをnotifyHandleに返す。

eventFilterを省略した場合，すべての種類のイベントの通知を要求する。

介入の発生の観測からも運転の状況を推定できるため，本サービスコールもRiskPrivacyのリスク制御の対象である。

unnotify

AEBイベントの通知停止

【パラメータ】

notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正

【機能】

notifyHandleで指定した通知を停止する。指定した通知用のイベントキューは削除される。

getEvent

AEBイベントの取得

【パラメータ】

notifyHandle	NotifyHandleType	イベントキューを識別するための通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
event	AEBEventType	取得したイベント

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正
E_NO_DATA	イベントキューが空
E_RISK_APPLICATION	アプリケーションがRiskPrivacyに対応していない
E_RISK_USER	ユーザがRiskPrivacyを承認していない

【機能】

notifyHandleで指定した通知用のイベントキューから，先頭のイベントを取り出す。取り出したイベントは，イベントキューから削除される。

イベントの内容からも運転の状況を推定できるため，本サービスコールもRiskPrivacyのリスク制御の対象である。



イベントがパラメータ等で渡されてくるプログラミング環境向けの物理APIでは，用意する必要がない。

5.7. Vehicle.ADADAS.ParkingAssist（略称：ParkingAssist）

5.7.1. 位置付けと機能

Vehicle.ADADAS.ParkingAssist（略称：ParkingAssist）は，車両に搭載された駐車支援を参照・操作するためのシングルインスタンスオブジェクトである。作動の開始・終了の要求と，現在の作動状態，駐車の支援の段階，および障害物への近接の警報の出力の状況の参照を提供する。

本オブジェクトは，Vehicle.ADADASの目録に，機能の種別ParkingAssist，提供するオブジェクトの完全

修飾名Vehicle.ADADAS.ParkingAssistとして現れる。DDTの責任主体・運行設計領域の充足状態・縮退運転・最小リスク状態など，AD/ADASシステム全体について一つに定まる状態は，本オブジェクトではなくVehicle.ADADASが提供する。

5.7.1.1. ParkingAssistの位置付けと性質

ParkingAssistは，外界センサ（超音波ソナー・カメラ等）により駐車可能な空間および周囲の障害物を検知し，運転者に案内・警報を提示する，または駐車の操舵を部分的に支援する運転支援の機能である。駐車の運転操作は運転者が主体的に行い，ParkingAssistの作動中も，動的運転タスク（DDT）の責任は運転者にある。

本オブジェクトが扱う支援は，次のとおりである。

- 駐車可能な空間の検出（機能クラスSpaceDetection）
- 駐車の操作の案内（機能クラスParkingGuidance）
- 駐車の案内の間の操舵の支援（機能クラスSteeringAssist）
- 周囲の障害物への近接の警報（機能クラスObstacleWarning）

操舵の支援を行う間も，加速・制動および変速の操作は運転者が行う。運転者の運転操作なしに駐車を完了し，駐車の実行中のDDTの責任をAD/ADASシステムが負う機能は，本オブジェクトではなく自動駐車（Vehicle.ADADAS.AutomatedParking）が扱う。操舵・加速・制動・変速のすべてを車両が行うが，運転者が運転の監視の責任を負う形態（SAE　J3016のレベル2に相当する形態）は，本オブジェクトおよびVehicle.ADADAS.AutomatedParkingのいずれの対象でもない。また，障害物との衝突を避けるための自動の制動は，本オブジェクトでは扱わない。

本オブジェクトは，ParkingAssistが用いるセンサの構成，空間の検出・経路の算出・障害物の検知のアルゴリズムを規定しない。検出した空間および障害物の位置・形状・距離は本オブジェクトでは扱わない。本オブジェクトが提供するのは，機能が作動しているか（operatingState），いま作動を継続できるか（availability），駐車の支援がどの段階にあるか（parkingPhase），現に警報を出力しているか（functionOutput），および故障の区分（faultCategory）である。



検出した空間および障害物の位置・形状・距離は，Vehicle.SurroundModelの担当である。

5.7.1.2. 作動の開始・終了の操作

本オブジェクトは，作動の開始（activate）と終了（deactivate）をアプリケーションから要求できる。車両が備える操作の手段（駐車支援のスイッチ，車載の設定画面等）により運転者がParkingAssistを有効化・無効化した場合も，同じ作動状態operatingStateに反映される。

アプリケーションが行える操作は，構成情報supportedOperationsが宣言する。supportedOperationsの列挙子は，操作を行うサービスコールと1対1に対応する。アプリケーションは，画面を構成する前にsupportedOperationsを参照し，宣言されていない操作の要素を提示してはならない。宣言されていない操作のサービスコールを呼び出した場合，E_NOT_SUPPORTEDエラーとなる。

supportedOperationsが操作の可否（恒久的な可否）を表すのに対し，availabilityは，いまその操作を受け付けられるか（一時的な可否）を表す。この2つは別の判定であり，アプリケーションは，前者により要素を提示するか否かを，後者により要素を操作可能とするか否かを決める。

駐車する空間の選択と，駐車の案内の開始は，運転者が車両の備える操作の手段により行う。本オブジェクトは，これらをアプリケーションから要求する手段を持たない。

5.7.1.3. 作動状態

本オブジェクトの作動状態operatingStateおよび利用可否availabilityは，Vehicle.ADADASが規定するAD/ADAS機能共通の作動状態モデルに従う。本節では，ParkingAssistに固有の制約のみを規定する。

ParkingAssistにおける各作動状態の意味は，次のとおりである。

- Off … ParkingAssistが無効化されており，空間の検出・案内・警報のいずれも行っていない状態。アプリケーションによる作動の終了の要求，および運転者の操作のいずれによっても，この状態となる。
- Standby … ParkingAssistが有効であるが，車速が構成情報maximumOperatingSpeedを超えている等の理由で，支援を行っていない状態。
- Activating … 作動開始の要求（activate）を受理し，遷移の処理を行っている過渡状態。
- Active … ParkingAssistが有効であり，備える機能クラスに応じて，空間の検出・案内・操舵の支援・障害物の監視を行っている状態。
- Suspended … 駐車の案内の間の操舵の支援が一時的に抑止されている状態。運転者の操舵または他の経路による操舵の制御により抑止されている場合と，他のAD/ADAS機能が駐車の運転操作を担っている場合とがある。
- Deactivating … 作動終了の要求（deactivate）を受理し，Offへ遷移する処理を行っている過渡状態。
- Fault … 故障により機能を提供できない状態。故障の区分は状態faultCategoryが表す。

ParkingAssistでは，FallbackおよびMRCの列挙子は現れない。縮退運転および最小リスク状態への移行は，DDTの責任を持つ機能が行うものであり，運転支援の機能であるParkingAssistはこれを行わないためである。

ParkingAssistが有効である間に車速が構成情報maximumOperatingSpeedを超えた場合は，Suspendedではなく，StandbyかつavailabilityがTemporarilyUnavailable，unavailableReasonがVehicleStateNotReadyで表す。

operatingStateがOffである間，availabilityは，ParkingAssistを有効にできるか（故障の有無等）のみを表す。車速等，有効化した後にのみ意味を持つ作動の条件は，Offの間のavailabilityに反映しない。すなわち，車速が構成情報maximumOperatingSpeedを超えていることのみを理由として，Offの間にavailabilityがTemporarilyUnavailableとなることはない。作動の条件を満たしているか否かは，有効化した後に，availabilityおよびunavailableReasonにより表す。

運転者の操舵または他の経路による操舵の制御により操舵の支援が抑止されている間は，operatingStateはSuspended，availabilityはTemporarilyUnavailable，unavailableReasonはControlOverriddenとなる。自動駐車等，駐車の運転操作の全体を担う他のAD/ADAS機能が作動している間は，operatingStateはSuspended，availabilityはTemporarilyUnavailable，unavailableReasonはSupersededByOtherFunctionとなり，blockingFunctionがその機能を指す。

車両が，アプリケーションからの作動の開始・終了そのものを認めない場合は，構成情報supportedOperationsにActivateおよびDeactivateを含めないことで表す。

5.7.1.4. 駐車の支援の段階

駐車の支援がどの段階にあるかは，状態parkingPhaseが表す。parkingPhaseは，機能クラスSpaceDetectionまたはParkingGuidanceを備え，かつoperatingStateがActiveまたはSuspendedである場合にのみ値を持つ。各段階の意味は，次のとおりである。

- Searching … 駐車可能な空間を探索している。
- SpaceFound … 駐車可能な空間を検出し，運転者による案内の開始の操作を待っている。
- Guiding … 駐車の操作の案内を行っている。機能クラスSteeringAssistを備える場合，この段階では車両が操舵を行う。
- Completed … 駐車の案内が完了した。
- Aborted … 駐車の案内を完了せずに終了した。

CompletedおよびAbortedは，直前の駐車の支援の結果を表し，次の探索または案内が始まるまで保持される。

Suspendedの間，parkingPhaseは中断前の値を保持する。抑止する要因が解消した後に案内を再開するか中止するかは車両による。中止した場合，parkingPhaseはAbortedとなる。

状態parkingMethodは，検出した空間，または案内の対象となっている空間の駐車方式（縦列・並列・斜め）を表す。車両が駐車方式を判別できない場合は値を持たない。車両が支援できる駐車方式は，構成情報supportedParkingMethodsが宣言する。駐車方式はアプリケーションから指定することはできない。

parkingPhaseの値が変化したときは，ParkingPhaseChangedイベントが発生する。

5.7.1.5. 障害物への近接の警報

障害物への近接の警報の出力の状況は，状態functionOutputが表す。functionOutputは，機能クラスObstacleWarningを備えている場合にのみ値を持つ。activeは，障害物への近接を検知しているか，または運転者への提示を現に行っているかを表す。levelは出力の段階を表し，機能クラスWarningLevelReportedを備えている場合にのみ値を持つ。

段階により，「障害物への近接を検知したが，まだ運転者への提示を行っていない状態」（Detected），「表示等による弱い提示を行っている状態」（Advisory），「音・振動を伴う警報を出力している状態」（Warning）を区別できる。ParkingAssistは障害物に対する自動の制動を行わないため，Criticalは現れない。levelは提示の強さを表すものであり，障害物までの距離の区分を表すものではない。

operatingStateがOff，Standby，Activating，DeactivatingまたはFaultである間，functionOutputのactiveは常にfalseである。Suspendedの間に警報を出力しているか否かは，functionOutputにより判定する。

functionOutputの値が変化したときは，FunctionOutputChangedイベントが発生する。近接の検知（Detected）の開始，提示の段階の変化，および出力の終了のいずれでも発生する。



障害物の方向（前方・後方・側方）および障害物までの距離は，本オブジェクトでは提供しない。

5.7.1.6. ParkingAssistに対する制御の調停

本オブジェクトに対する制御の調停には，標準的な調停機能（後勝ちの原則）が適用される。すなわち，あるアプリケーションが作動を開始した後に，別のアプリケーションが作動の終了を要求した場合，後の要求が有効となる。

本オブジェクトは，標準的なロック機能を持つLockableObjectである。状態を変化させるサービスコール（activate，deactivate）およびロックを取得するサービスコール（lock）は，操作優先度priorityを持つ。ロック中の本オブジェクトに対して，ロックと同じかそれより低い優先度で操作した場合はE_OBJECT_LOCKEDエラーとなり，より高い優先度で操作した場合は，既存のロックが強制解除されて操

作が実行される。アプリケーションが使用できる最高値より高い優先度を指定した場合はE_DENIED_PRIORITYエラー，操作優先度が有効範囲外である場合はE_INVALID_PARAMETERエラーとなる。

自アプリケーションのロックが強制解除された場合はOwnLockRevokedイベントが，他のアプリケーション等によりロックが解除された場合はOtherLockReleasedイベントが発生する。作動状態がFaultとなった場合，ロックは解除され，その間は新たにロックを取得することができない。

運転者による直接の操作（駐車支援のスイッチによる有効化・無効化，操舵による介入等）は，本APIを用いない一つのアプリケーションによる操作と見なし，後勝ちの原則に従う。この操作は，アプリケーションが取得したロックおよび操作優先度によって妨げられることはなく，常に有効となる。運転の責任を負うのは運転者であり，アプリケーションによる排他が運転者の操作に優先してはならないためである。



Vehicle.Motionは，アプリケーションが車両の運動そのものを制御するためのオブジェクトであり，本オブジェクトは，車両に搭載された駐車支援の作動を要求するものである。両者は別の経路であり，同時に用いられた場合も後勝ちの原則に従う。ParkingAssistが操舵の支援を行っている間にVehicle.Motionにより横方向の運動が制御された場合，本オブジェクトはavailabilityをTemporarilyUnavailable，unavailableReasonをControlOverriddenとし，operatingStateをSuspendedとする。

5.7.1.7. 操作・参照に伴うリスク

ParkingAssistの作動の開始・終了は，操舵の支援の有無と，障害物への近接の警報の有無を変える。とくに作動の終了は，案内の途中で操舵の支援が失われること，および運転者が障害物への近接に気付かないまま駐車の操作を続けることにつながりうる。これをリスククラスRiskCollisionとして定義し，activateおよびdeactivateをリスク制御の対象とする。

ParkingAssistの作動状態および駐車の支援の段階は，運転者がいつ駐車したかを第三者が推定できる情報である。これをリスククラスRiskPrivacyとして定義し，状態を参照するサービスコール（getStatus），イベントの通知要求（notify）およびイベントの取得（getEvent）をリスク制御の対象とする。

いずれのリスククラスについても，当該リスクに対応していないアプリケーションからの呼び出しはE_RISK_APPLICATIONエラー，ユーザが当該リスクを承認していない場合はE_RISK_USERエラーとなる。

構成情報の参照（getConfig）は，車両に搭載されたParkingAssistの静的な性質を返すものであり，リスク制御の対象としない。

5.7.2. 機能クラス

ParkingAssistは，以下の機能クラスを持つ。

SpaceDetection	駐車可能な空間を検出することができる（状態parkingPhaseのSearching，SpaceFound）
ParkingGuidance	駐車の操作を運転者に案内することができる（状態parkingPhaseのGuiding，Completed，Aborted）
SteeringAssist	駐車の案内の間，車両が操舵を行うことができる
ObstacleWarning	周囲の障害物への近接を検知し，運転者に警報を提示することができる（状態functionOutput，FunctionOutputChangedイベント）

WarningLevelReported	障害物への近接の警報の出力の段階（検知，注意喚起，警報）を提供することができる（状態functionOutputのlevel）

5.7.3. リスククラス

ParkingAssistは，以下のリスククラスを持つ。

RiskCollision	ParkingAssistの作動の開始・終了により操舵の支援および障害物への近接の警報の有無が変わり，周囲の障害物・歩行者との衝突を生じるリスク
RiskPrivacy	走行状態および運転者の運転への関与状態の開示により，個人の行動を推定できることに伴うプライバシのリスク

5.7.4. 構成情報

ParkingAssistは，以下の構成情報を持つ。

capabilities	enumSet (ParkingAssistCapabilityType)	その車両のParkingAssistが持つ機能
riskClasses	enumSet (ParkingAssistRiskType)	車両およびビークルOSで対応できていないリスククラスの集合
supportedOperations	enumSet (ParkingAssistOperationType)	アプリケーションから要求できる操作の集合。空集合となることがある
displayName	String?	ユーザに表示する呼称を取得するためのキー。車両が呼称を公開する場合にのみ値を持つ
maximumOperatingSpeed	SpeedType	支援を行う車速の上限。単位はkm/h。車速がこれを超える間，作動状態はStandbyとなる
supportedParkingMethods	enumSet (ParkingAssistMethodType)	支援できる駐車方式の集合。機能クラスSpaceDetectionおよびParkingGuidanceをいずれも備えない場合は空集合となる
timestampAccuracy	UInt32	この車両のタイムスタンプ誤差上限の宣言。単位はμs

上記は，YAML形式で記述した構成記述ファイル中に記述する形式である。本オブジェクトはシングルインスタンスオブジェクトであるため，構成記述では，車両を表すノードの下に置かれたADADASノードの下に，オブジェクトを表すノードParkingAssistを置き，その下に上記の構成情報を記述する。このノードは，Vehicle.ADADASの目録において，providingObjectがVehicle.ADADAS.ParkingAssistである要素に対応する。

riskClassesには，車両およびビークルOSで対応できていない（危険を防止できていない）リスククラスの集合を記述する。

構成情報を参照するサービスコールは，この構成情報に対して，次の変換を行った構成情報を返す。

- displayNameに対しては，それを元に現在のロケールに対応する呼称の文字列を取得し，その文字列を返す。

5.7.5. 状態

ParkingAssistは，以下の状態を持つ。

timestamp	TimestampUsType	状態の測定時刻。単位はμs。車内共通時刻基盤に基づく
dataQualifier	DataQualifierType	このサンプルの品質。InvalidまたはUnavailableの場合，以下の状態の値を読んではならない。値の変化はDataQualifierChangedイベントにより通知される
operatingState	AutomationOperatingStateType	現在の作動状態。ParkingAssistではFallbackおよびMRCは現れない
availability	AutomationAvailabilityType	いま支援を開始または継続できるか
unavailableReason	AutomationUnavailableReasonType?	支援を開始または継続できない理由。availabilityがAvailable以外の値である場合にのみ値を持つ
blockingFunction	RelatedObjectType[]	支援を妨げている機能オブジェクト。unavailableReasonがDependentFunctionFaultまたはSupersededByOtherFunctionである場合にのみ要素を持つ
faultCategory	AutomationFaultCategoryType?	故障の区分。operatingStateがFaultである場合にのみ値を持つ
parkingPhase	ParkingAssistPhaseType?	駐車の支援の段階。機能クラスSpaceDetectionまたはParkingGuidanceを備え，かつoperatingStateがActiveまたはSuspendedである場合にのみ値を持つ
parkingMethod	ParkingAssistMethodType?	検出した空間，または案内の対象となっている空間の駐車方式。parkingPhaseがSpaceFound，Guiding，CompletedまたはAbortedであり，かつ車両が駐車方式を判別できる場合にのみ値を持つ
functionOutput	AutomationFunctionOutputType?	障害物への近接の警報の出力の状況。機能クラスObstacleWarningを備えている場合にのみ値を持つ。levelは機能クラスWarningLevelReportedを備えている場合にのみ値を持つ
lockStatus	LockStatusType	ロック状態

状態は，呼び出し時点のParkingAssistの状態を返す冪等なスナップショットである。値を読む前にdataQualifierを確認する。

ParkingAssistが支援を行える状態にあるかは，operatingStateにより判定する。駐車の支援がどの段階にあるかはparkingPhaseにより，いま障害物への近接を検知しているか，警報を出しているかはfunctionOutputにより判定する。

operatingStateとavailabilityの組合せの規則は，Vehicle.ADADASが規定するAD/ADAS機能共通の作動状態モデルによる。

故障により機能を提供できない場合，operatingStateはFaultとなり，faultCategoryがその区分を表す。これにより，FaultDetectedイベントを受け取っていないアプリケーションも，状態の参照のみで故障の区分を知ることができる。本オブジェクト自体が状態を提供できない場合は，getStatusがE_OBJECT_FAULTエラーを返す。

5.7.6. サービスコール一覧

ParkingAssistに対するサービスコールは以下の通り。

getConfig	ParkingAssistの構成情報の参照
getStatus	ParkingAssistの状態の参照
activate	ParkingAssistの作動の開始の要求
deactivate	ParkingAssistの作動の終了の要求
lock	ParkingAssistのロックの取得
unlock	ParkingAssistのロックの解除
notify	ParkingAssistイベントの通知要求
unnotify	ParkingAssistイベントの通知停止
getEvent	ParkingAssistイベントの取得

5.7.7. イベント

イベントは，事象が発生したことのみを伝え，状態の値そのものを運ばない。ただし，事象の発生時点の情報として有用なもの（故障の区分等）は，イベントの関連情報としても渡す。アプリケーションは，イベントを受けた後，必要に応じてgetStatusにより値を取得する。

状態の取得は，要求応答（getStatus）と，各状態項目の変化イベントによる。

ParkingAssistに対するイベント種別は以下の通り。

OperatingStateChanged	作動状態の変化（アプリケーションの要求および運転者の操作による有効化・無効化を含む）
AvailabilityChanged	支援を開始または継続できるかの変化（支援を妨げている理由の変化を含む）
ParkingPhaseChanged	駐車の支援の段階（状態parkingPhase）の変化
FunctionOutputChanged	障害物への近接の警報の出力の状況（状態functionOutput）の変化。出力の開始・段階の変化・終了のいずれでも発生する
FaultDetected	故障の検知（ParkingAssistの機能を提供できない状態）
FaultRecovered	故障からの回復
DataQualifierChanged	状態の品質（状態dataQualifier）の変化
OwnLockRevoked	自アプリケーションのロックの強制解除
OtherLockReleased	他のアプリケーション等によるロック解除
EventOverflow	イベントキューのオーバフロー

ConfigChanged		構成情報の変更（ソフトウェアの更新による対応する操作・支援を行う車速の上限・支援できる駐車方式の変更）

この内，イベント種別がFaultDetectedの場合には，以下の関連情報を持つ。

faultCategory	AutomationFaultCategoryType	故障の区分

また，イベント種別がEventOverflowの場合には，以下の関連情報を持つ。

noLostEvents	UInt32	失われたイベントの数

5.7.8. データ型の定義

5.7.8.1. ParkingAssistの機能クラス

```
type ParkingAssistCapabilityType = enum {
    SpaceDetection,
    ParkingGuidance,
    SteeringAssist,
    ObstacleWarning,
    WarningLevelReported
};
```

5.7.8.2. ParkingAssistのリスククラス

```
type ParkingAssistRiskType = enum {
    RiskCollision,
    RiskPrivacy
};
```

5.7.8.3. タイムスタンプ

```
type TimestampUsType = UInt64;  // 車内共通時刻基盤の起点を基準とする測定時刻。単位はμs。複数のビークルAPIで共通に用いる共通データ型であり，本仕様内に定義する。
```

5.7.8.4. サンプル品質の種別

```
type DataQualifierType = enum {
    Normal, // 正常
    Unavailable,    // 利用不可
    Degraded,   // 劣化
    TestMode,   // テストモード
    Invalid // 無効（値を読んではならない）
};
```

複数のビークルAPIで共通に用いる共通データ型であり，本仕様内に定義する。

5.7.8.5. 速度

```
type SpeedType = Float32;   // 速度。単位はkm/h。正の値は前進，負の値は後退を示す。複数のビークルAPIで共通に用いる共通データ型であり，本仕様内に定義する。
```

5.7.8.6. AD/ADAS機能共通の作動状態

以下の5つのデータ型は，AD/ADAS機能オブジェクトが共通に用いるデータ型であり，本仕様内に定義する。ParkingAssistにおける各列挙子の具体的な意味は，位置付けと機能の節に示す。

```
type AutomationOperatingStateType = enum {
    Off,    // 機能が無効化されており，作動していない状態
    Standby,    // 機能が有効であり，作動の条件が整えば作動できるが，現在は作動していない状態
    Activating, // 作動開始の要求を受理し，作動開始の処理を行っている過渡状態
    Active, // 作動中。車両の運動への介入または警報の出力を現に行っている状態
    Suspended,  // 作動が一時的に解除され，中断している状態。直前の作動時の設定値を保持しており，抑止する要因が解消すれば作動を再開できる
    Deactivating,   // 作動終了の要求を受理し，作動終了の処理を行っている過渡状態
    Fallback,   // 縮退運転を実行している状態
    MRC,    // 最小リスク状態へ移行し，これを維持している状態
    Fault   // 故障により機能を提供できない状態
};
```

各機能オブジェクトが現にどの作動状態にあるかを表す型である。機能によっては現れない列挙子がある。

```
type AutomationAvailabilityType = enum {
    Available,  // 作動を開始または継続できる
    TemporarilyUnavailable, // 一時的に条件を満たさないため，作動を開始または継続できない（条件が回復すれば作動できる）
    Unavailable // 故障等により，作動を開始または継続できない
};
```

その機能がいま作動を開始または継続できるかを表す型である。作動しているか否かは作動状態operatingStateが表すものであり，本型はこれを表さない。

```
type AutomationUnavailableReasonType = enum {
    ODDOutside, // 走行環境が運行設計領域の外にある
    SensorDegraded, // センサの性能低下・遮蔽により，必要な認識の性能が得られない
    DependentFunctionFault, // この機能が用いている他のAD/ADAS機能が故障している（blockingFunctionがその機能を指す）
    DriverMonitoringUnavailable,    // 運転者の監視が行えない
    VehicleStateNotReady,   // 車速・変速機の位置・制動等の車両の状態が作動の条件を満たさない
```

```
    FunctionFault,  // この機能自体が故障している
    ControlOverridden,  // 運転者の操作または他の経路による車両の運動の制御により，作動が抑止されている
    UserDisabled,   // 利用者の設定により機能が無効にされている
    SupersededByOtherFunction,  // 他のAD/ADAS機能がその運転操作を担っているため作動しない（blockingFunctionがその機能を指す）
    CalibrationRequired,    // 較正が必要で作動できない
    Other   // その他の理由
};
```

availabilityがAvailable以外の値である場合に，作動を開始または継続できない理由を表す型である。機能によっては現れない列挙子がある。

```
type AutomationOutputLevelType = enum {
    Detected,   // 事象を検知したが，運転者への提示はまだ行っていない
    Advisory,   // 注意喚起（表示等の弱い提示）
    Warning,    // 警報（音・振動を伴う提示）
    Critical    // 緊急（車両運動への介入を伴う）
};
```

機能の出力の段階を表す型である。Detected，Advisory，Warning，Criticalの順に強い段階を表す。機能によっては現れない段階がある。

```
type AutomationFunctionOutputType = struct {
    active : Bool,  // その機能が事象の検知または運転者への提示を現に行っているか
    level : AutomationOutputLevelType?  // 出力の段階。activeがtrueであり，かつ段階を提供できる機能である場合にのみ値を持つ
};
```

機能が外部（運転者）に対して現に行っている出力を，有無と段階のみで表す型である。出力の判断の根拠（検知した車線の形状・物体の種別等）は本型に含まない。

5.7.8.7. AD/ADAS機能共通の故障の区分

以下のデータ型は，AD/ADAS機能オブジェクトが共通に用いるデータ型であり，本仕様内に定義する。ParkingAssistでは，アクチュエータの故障（ActuatorFault）は機能クラスSteeringAssistを備えている場合にのみ，警報を提示する装置の故障（WarningDeviceFault）は機能クラスObstacleWarningを備えている場合にのみ現れる。

```
type AutomationFaultCategoryType = enum {
    SensorFault,    // センサの故障
    ActuatorFault,  // アクチュエータの故障
    ComputationFault,   // 演算装置の故障
    CommunicationFault, // 車内通信の故障
    PowerFault, // 電源の異常
    WarningDeviceFault, // 警報を提示する装置の故障
    Other   // その他の故障
};
```

機能を提供できない故障の区分を表す型である。機能によっては現れない列挙子がある。故障の区分は，各機能オブジェクトの状態faultCategoryおよびFaultDetectedイベントの関連情報により渡される。

5.7.8.8. ParkingAssistの操作

```
type ParkingAssistOperationType = enum {
    Activate,   // 作動の開始の要求（サービスコールactivate）
    Deactivate  // 作動の終了の要求（サービスコールdeactivate）
};
```

アプリケーションから要求できる操作を表す型である。列挙子は，操作を行うサービスコールと1対1に対応する。すべてのアプリケーションが呼び出すことができるサービスコール（getConfig，getStatus，notify，unnotify，getEvent，lock，unlock）は，本型の対象としない。

5.7.8.9. 駐車の支援の段階と駐車方式

```
type ParkingAssistPhaseType = enum {
    Searching,  // 駐車可能な空間を探索している
    SpaceFound, // 駐車可能な空間を検出し，運転者による案内の開始の操作を待っている
    Guiding,    // 駐車の操作の案内を行っている（機能クラスSteeringAssistを備える場合
は操舵の支援を含む）
    Completed,  // 駐車の案内が完了した
    Aborted // 駐車の案内を完了せずに終了した
};
```

駐車の支援がどの段階にあるかを表す型である。

```
type ParkingAssistMethodType = enum {
    Parallel,   // 縦列駐車（車路に沿った向きに駐車する）
    Perpendicular,  // 並列駐車（車路に直交する向きに駐車する）
    Angled  // 斜め駐車（車路に対して斜めの向きに駐車する）
};
```

駐車の方式を表す型である。

5.7.8.10. ParkingAssistの構成情報（getConfigが返すデータ型）

```
type ParkingAssistConfigType = struct {
    capabilities : enumSet(ParkingAssistCapabilityType),
    riskClasses : enumSet(ParkingAssistRiskType),
    supportedOperations : enumSet(ParkingAssistOperationType),
    displayName : String?,
    maximumOperatingSpeed : SpeedType,
    supportedParkingMethods : enumSet(ParkingAssistMethodType),
    timestampAccuracy : UInt32
};
```

5.7.8.11. ParkingAssist状態

```
type ParkingAssistStatusType = struct {
    timestamp : TimestampUsType,
    dataQualifier : DataQualifierType,
    operatingState : AutomationOperatingStateType,
    availability : AutomationAvailabilityType,
    unavailableReason : AutomationUnavailableReasonType?,
    blockingFunction : RelatedObjectType[],
    faultCategory : AutomationFaultCategoryType?,
    parkingPhase : ParkingAssistPhaseType?,
    parkingMethod : ParkingAssistMethodType?,
    functionOutput : AutomationFunctionOutputType?,
    lockStatus : LockStatusType
};
```

呼び出し時点のParkingAssistの状態の冪等なスナップショットである。

5.7.8.12. ParkingAssistイベント

```
type ParkingAssistEventKindType = enum {
    OperatingStateChanged,
    AvailabilityChanged,
    ParkingPhaseChanged,
    FunctionOutputChanged,
    FaultDetected,
    FaultRecovered,
    DataQualifierChanged,
    OwnLockRevoked,
    OtherLockReleased,
    EventOverflow,
    ConfigChanged
};
```

```
type ParkingAssistEventType = struct {
    eventInfo : variant(ParkingAssistEventKindType) {
        FaultDetected {
            faultCategory : AutomationFaultCategoryType
        },
        EventOverflow {
            noLostEvents : UInt32
        }
    }
};
```

5.7.9. サービスコールの詳細規定

getConfig

ParkingAssistの構成情報の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
config	ParkingAssistConfigType	ParkingAssistの構成情報

【エラーコード】

なし

【機能】

その車両のParkingAssistが持つ機能，アプリケーションから要求できる操作の集合，支援を行う車速の上限，および支援できる駐車方式を返す。

アプリケーションは，本サービスコールにより，画面を構成する前に，どの操作の要素を提示すべきかを決める。

getStatus

ParkingAssistの状態の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
status	ParkingAssistStatusType	ParkingAssistの状態

【エラーコード】

E_OBJECT_FAULT	本オブジェクトが故障しており，状態を提供できない
E_RISK_APPLICATION	アプリケーションがRiskPrivacyに対応していない
E_RISK_USER	ユーザがRiskPrivacyを承認していない

【機能】

ParkingAssistの現在の作動状態・利用可否・故障の区分・駐車の支援の段階・駐車方式・警報の出力の状況・ロック状態を返す。

呼び出しのたびに，その時点の状態の全体を返す冪等な応答である。値を読む前にdataQualifierを確認する。

activate

ParkingAssistの作動の開始の要求

【パラメータ】

priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_PARAMETER	priorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_LOCKED	本オブジェクトが，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている
E_OBJECT_STATUS	作動を開始できる作動状態でない（既に有効である，またはavailabilityがUnavailableであり，その理由が故障以外である）
E_OBJECT_FAULT	本オブジェクトが故障しており，要求を受け付けられない（unavailableReasonがFunctionFaultである場合を含む）
E_INFEASIBLE	availabilityがTemporarilyUnavailableであり，作動の条件以外の一時的な理由（故障等）により，現在はParkingAssistを有効にできない
E_RISK_APPLICATION	アプリケーションがRiskCollisionに対応していない
E_RISK_USER	ユーザがRiskCollisionを承認していない

【機能】

ParkingAssistに作動の開始を要求する。本サービスコールは，駐車支援を有効にする操作に相当し，Offからの遷移を要求する。故障等によりParkingAssistを有効にできない場合を除き，要求は作動の条件（車速が構成情報maximumOperatingSpeed以下である等）によらず受理され，operatingStateはStandbyへ遷移する。その後，作動の条件が整っている場合はActiveへ遷移し，整っていない間はStandbyのままとなり，availabilityおよびunavailableReasonがその理由を表す。駐車の案内は開始されない。案内の開始は運転者が行う。

正常終了は要求が受理されたことのみを意味し，作動が開始したことを意味しない。遷移の完了は，getStatusまたはOperatingStateChangedイベントにより確認する。

作動を開始できない場合の応答は，利用可否availabilityとその理由unavailableReasonにより定まる。availabilityがTemporarilyUnavailableである場合はE_INFEASIBLEエラー，Unavailableでありunavailable ReasonがFunctionFaultである場合はE_OBJECT_FAULTエラー，Unavailableでありそれ以外の理由である場合はE_OBJECT_STATUSエラーとなる。

本サービスコールは，構成情報supportedOperationsにActivateを含む場合にのみ用いることができる。

deactivate

ParkingAssistの作動の終了の要求

【パラメータ】

priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_PARAMETER	priorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_LOCKED	本オブジェクトが，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている
E_OBJECT_STATUS	作動を終了できる作動状態でない（既にOffである等）
E_OBJECT_FAULT	本オブジェクトが故障しており，要求を受け付けられない
E_RISK_APPLICATION	アプリケーションがRiskCollisionに対応していない
E_RISK_USER	ユーザがRiskCollisionを承認していない

【機能】

ParkingAssistに作動の終了を要求する。本サービスコールは，駐車支援を無効にする操作に相当し，Offへの遷移を要求する。駐車の案内を行っていた場合，案内は終了し，操舵の支援を行っていた場合，操舵は運転者に戻る。

正常終了は要求が受理されたことのみを意味し，作動が終了したことを意味しない。遷移の完了は，getStatusまたはOperatingStateChangedイベントにより確認する。

本サービスコールは，構成情報supportedOperationsにDeactivateを含む場合にのみ用いることができる。



本サービスコールによりParkingAssistが無効になると，障害物への近接の警報は出力されなくなり，案内の途中であれば操舵の支援も失われる。運転者が障害物への近接に気付かないまま駐車の操作を続けること，および操舵を引き継ぐ準備を終えていないことにつながるため，アプリケーションは，無効にしたことを運転者が認識できるようにしなければならない。

lock

ParkingAssistのロックの取得

【パラメータ】

priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_PARAMETER	priorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_LOCKED	本オブジェクトが，priorityと同じかそれより高い優先度でロックされている
E_OBJECT_FAULT	本オブジェクトが故障しており，ロックを取得できない

【機能】

本オブジェクトを，指定した操作優先度でロックする。ロックを取得したアプリケーションは，状態を変化させるサービスコール（activate，deactivate）を，同じかそれより低い優先度の他のアプリケーションによる操作から保護する。

既存のロックより高い優先度でロックを取得した場合，既存のロックは強制解除され，そのロックを取得していたアプリケーションにはOwnLockRevokedイベントが通知される。

運転者による直接の操作は，ロックおよび操作優先度によって妨げられない。

unlock

ParkingAssistのロックの解除

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_OBJECT_STATUS	自アプリケーションが，本オブジェクトをロックしていない

【機能】

自アプリケーションが取得した本オブジェクトのロックを解除する。

notify

ParkingAssistイベントの通知要求

【パラメータ】

eventFilter	enumSet (ParkingAssistEventKindType)?	通知するイベントの種類の集合

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【エラーコード】

E_INVALID_PARAMETER	eventFilterが有効範囲外
E_RISK_APPLICATION	アプリケーションがRiskPrivacyに対応していない
E_RISK_USER	ユーザがRiskPrivacyを承認していない

【機能】

ParkingAssistで発生したeventFilterで指定したイベントの通知を要求する。イベントを通知するためのイベントキューを生成し，その通知ハンドルをnotifyHandleに返す。

eventFilterを省略した場合，すべての種類のイベントの通知を要求する。

イベントの発生の観測からも駐車の時刻を推定できるため，本サービスコールもRiskPrivacyのリスク制御の対象である。

unnotify

ParkingAssistイベントの通知停止

【パラメータ】

notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正

【機能】

notifyHandleで指定した通知を停止する。指定した通知用のイベントキューは削除される。

getEvent

ParkingAssistイベントの取得

【パラメータ】

notifyHandle	NotifyHandleType	イベントキューを識別するための通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
event	ParkingAssistEventType	取得したイベント

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正
E_NO_DATA	イベントキューが空
E_RISK_APPLICATION	アプリケーションがRiskPrivacyに対応していない
E_RISK_USER	ユーザがRiskPrivacyを承認していない

【機能】

notifyHandleで指定した通知用のイベントキューから，先頭のイベントを取り出す。取り出したイベントは，イベントキューから削除される。

イベントの内容からも駐車の時刻を推定できるため，本サービスコールもRiskPrivacyのリスク制御の対象である。



イベントがパラメータ等で渡されてくるプログラミング環境向けの物理APIでは，用意する必要がない。

5.8. Vehicle.ADADAS.AutomatedParking（略称：AutomatedParking）

5.8.1. 位置付けと機能

Vehicle.ADADAS.AutomatedParking（略称：AutomatedParking）は，車両に搭載された自動駐車の機能を参照・操作するためのシングルインスタンスオブジェクトである。作動の開始・終了の要求，駐車枠の候補の参照，入庫と出庫の実行の要求，および現在の作動状態と駐車の進捗の参照を提供する。

本オブジェクトは，Vehicle.ADADASの目録に，機能の種別AutomatedParking，提供するオブジェクトの完全修飾名Vehicle.ADADAS.AutomatedParkingとして現れる。DDTの責任主体・運行設計領域の充足状態・縮退運転・最小リスク状態など，AD/ADASシステム全体について一つに定まる状態は，本オブジェクトではなくVehicle.ADADASが提供する。

5.8.1.1. AutomatedParkingの位置付けと性質

自動駐車は，外界のセンサ（超音波ソナー・カメラ等）により駐車枠とその周囲を認識し，運転者の運転操作なしに，駐車枠への入庫と駐車枠からの出庫を行う機能である。駐車の実行中は，操舵・加速・制動・変速のすべてを車両が行い，動的運転タスク（DDT）を実行する主体はAD/ADASシステムとなる。

これに対し，駐車支援（Vehicle.ADADAS.ParkingAssist）は，運転者が運転操作の一部または全部を行う駐車を支援する機能であり，支援の作動中もDDTの責任は運転者にある。操舵のみを車両が行い，加速・制動を運転者が行う機能は，駐車支援に当たる。本オブジェクトが扱うのは，駐車の実行中に運転者が運転操作を行わず，駐車の実行中のDDTの責任をAD/ADASシステムが負う機能に限る。操舵・加速・制動・変速のすべてを車両が行うが，運転者が運転の監視の責任を負う形態（SAE J3016のレベル2に相当する形態）は，本オブジェクトおよびVehicle.ADADAS.ParkingAssistのいずれの対象でもない。

本オブジェクトは，自動駐車が用いるセンサの構成，駐車枠の認識の方法，駐車の経路の計画と制御のアルゴリズムを規定しない。認識した駐車枠の形状・位置，周囲の物体，計画した経路は本オブジェクトでは扱わない。

駐車する駐車枠の指定は，本オブジェクトのサービスコールにより行う。経路案内（Vehicle.Navigation）の目的地は経路の終点であり，駐車枠を指定するものではない。

5.8.1.2. 駐車の手順

自動駐車は，次の手順で行う。

- アプリケーションまたは運転者が作動を開始すると（activate），機能が有効となり，車両は駐車枠の探索を始める。探索中の車両の運転は運転者が行う。
- 車両が駐車枠を検出すると，それを駐車枠の候補とする。アプリケーションは，候補の一覧をgetParkingSlotsにより取得する。
- アプリケーションが候補の中から1つを指定して入庫を要求すると（startParkIn），車両はその駐車枠への入庫を実行する。駐車枠に停車している状態から出庫を要求すると（startParkOut），車両は出庫を実行する。
- 車両が入庫または出庫を終えると，車両は停止を保持した状態で機能を終了する。

駐車の進捗は，状態parkingPhaseが表す。parkingPhaseは，駐車枠の探索中（Searching），駐車枠の候補がある状態（SlotDetected），入庫または出庫の実行中（Maneuvering），完了（Completed），中断（Interrupted）のいずれかである。中断した場合は，状態interruptionReasonがその理由を表す。

入庫と出庫の別は，状態maneuverが表す。入庫の実行中は，状態targetSlotが目標の駐車枠を指す。

parkingPhaseは，作動状態operatingStateがOffまたはFaultである間，直前の駐車の結果（CompletedまたはInterrupted）を保持する。作動の開始が受理されると，parkingPhaseはSearchingとなる。起動後に一度も作動していない場合，parkingPhaseは値を持たない。

5.8.1.3. 駐車枠の候補

駐車枠の候補は，車両が認識し，入庫できると判断した駐車枠である。候補は，作動状態がStandbyである間に探索され，getParkingSlotsにより参照できる。候補の追加・消失は，ParkingSlotsChangedイベントにより通知される。

各候補は，候補の識別子（slotId），駐車枠の種別（縦列・並列・斜め），および自車から見た駐車枠の方向（前後左右）を持つ。駐車枠の位置・形状の座標は持たない。アプリケーションは，候補の種別と方向により運転者に候補を提示し，選ばれた候補のslotIdを入庫の要求（startParkIn）で指定する。

slotIdは，その駐車枠が候補である間，同じ値を保つ。作動状態がStandbyである間に，候補でなくなった駐車枠のslotIdを指定した入庫の要求は，E_INVALID_IDエラーとなる。候補でなくなった駐車枠のslotIdは，別の駐車枠に割り当てられることがある。ただし，入庫の目標となった駐車枠の識別子（状態targetSlot）は，その入庫が完了または中断により終了するまで，他の駐車枠に割り当てられない。アプリケーションは，ParkingSlotsChangedイベントを受けた場合，保持している候補の一覧を取得し直す。

作動状態がStandby以外である間，候補の一覧は空となる。作動状態がSuspendedである間の入庫の再開の要求では，slotIdの妥当性を候補の一覧ではなく，中断している入庫の目標の駐車枠（状態targetSlot）との一致により判定する。

5.8.1.4. 作動状態と操作の意味

本オブジェクトの作動状態operatingStateおよび利用可否availabilityは，Vehicle.ADADASが規定するAD/ADAS機能共通の作動状態モデルに従う。本節では，自動駐車に固有の制約のみを規定する。

自動駐車における各作動状態の意味は，次のとおりである。

- Off … 自動駐車が無効化されており，駐車枠の探索も駐車の実行も行っていない状態。
- Standby … 自動駐車が有効であり，駐車枠を探索している状態。車両の運転は運転者が行っている。この状態から入庫または出庫を開始できる。
- Activating … 入庫または出庫の要求（startParkIn，startParkOut）を受理し，駐車の実行を開始する処理を行っている過渡状態。
- Active … 入庫または出庫を現に実行している状態。車両が操舵・加速・制動・変速のすべてを行う。
- Suspended … 入庫または出庫の実行が一時的に中断され，車両が停止を保持している状態。目標の駐車枠を保持しており，抑止する要因が解消すれば，駐車の再開の要求により実行を再開できる。
- Deactivating … 作動終了の要求（deactivate）を受理し，Offへ遷移する処理を行っている過渡状態。
- Fault … 故障により機能を提供できない状態。故障の区分は状態faultCategoryが表す。

自動駐車では，FallbackおよびMRCの列挙子は現れない。駐車の実行を継続できなくなった場合，車両はその場で停止し，停止を保持した上で駐車を中断する。この中断は，parkingPhaseのInterruptedとして表す。

作動を開始する要求（activate）は，自動駐車を有効にする操作に相当し，OffからStandbyへの遷移を要求する。車両の運動の制御は開始されない。入庫の要求（startParkIn）および出庫の要求（startParkOut）は，Standbyにおいては駐車の実行の開始を，Suspendedにおいては中断している駐車の実行の再開を，それぞれ要求する。作動を終了する要求（deactivate）は，Offへの遷移を要求する。駐車の実行中（ActiveまたはSuspended）に作動の終了を要求した場合，車両は停止を保持した上で駐車を中断し，Offへ遷移する。

入庫または出庫を終えた場合，parkingPhaseはCompletedとなり，operatingStateはOffとなる。駐車を中断した場合，parkingPhaseはInterruptedとなり，operatingStateはOff（故障による中断の場合はFault）となる。続けて駐車を行うには，あらためて作動の開始を要求する。

いずれの要求も，正常終了は要求が受理されたことのみを意味し，目的の作動状態への遷移が完了したことを意味しない。遷移の完了は，getStatusまたはOperatingStateChangedイベントにより確認する。

要求を受け付けられない場合の応答は，AD/ADAS機能共通の作動状態モデルの規定による。すなわち，availabilityがTemporarilyUnavailableである場合の作動開始の要求はE_INFEASIBLEエラー，unavailableReasonがFunctionFaultである場合はE_OBJECT_FAULTエラー，それ以外のUnavailableである場合はE_OBJECT_STATUSエラーとなる。

5.8.1.5. DDTの責任との関係

入庫または出庫の実行中（operatingStateがActiveまたはSuspendedである間），Vehicle.ADADASの状態ddtOwnerは，kindがAutomationSystemであり，functionに本オブジェクトを含む。handoverPhaseはStableである。

駐車の実行の開始は，運転者またはアプリケーションの要求（startParkIn，startParkOut）によるものであり，Vehicle.ADADASのTransitionRequestedイベントを伴わない。ddtOwnerは，operatingStateが

Activeとなった時点で本オブジェクトを指すものに変わる。

駐車の完了または中断により，DDTの責任は運転者に戻る。このとき車両は停止を保持しており，DDTの責任は，Vehicle.ADADASのTransitionRequestedイベントおよびacknowledgeTransitionによる応答を経ずに運転者に戻る。停止を保持した車両では，運転者が運転操作を始めるまで車両の運動が生じないためである。

運転者が降車した状態での駐車の実行（車外からの遠隔操作による駐車）は，本オブジェクトでは扱わない。入庫または出庫の要求は，運転者が運転席にいることを条件とする。駐車の実行中に運転者が運転席を離れた場合，またはドアが開かれた場合，車両は駐車を中断する。

5.8.1.6. 操作の可否の宣言

本オブジェクトに対してアプリケーションが行える操作は，構成情報supportedOperationsが宣言する。supportedOperationsの列挙子は，操作を行うサービスコールと1対1に対応する。アプリケーションは，画面を構成する前にsupportedOperationsを参照し，宣言されていない操作の要素を提示してはならない。

supportedOperationsに含まれない操作のサービスコールを呼び出した場合，E_NOT_SUPPORTEDエラーとなる。supportedOperationsは空集合となることがあり，その場合，本オブジェクトは参照のみが可能な読み取り専用の機能として振る舞う。

supportedOperationsが操作の可否（恒久的な可否）を表すのに対し，availabilityは，いまその操作を受け付けられるか（一時的な可否）を表す。この2つは別の判定であり，アプリケーションは，前者により要素を提示するか否かを，後者により要素を操作可能とするか否かを決める。

5.8.1.7. AutomatedParkingに対する制御の調停

本オブジェクトに対する制御の調停には，標準的な調停機能（後勝ちの原則）が適用される。すなわち，あるアプリケーションが入庫を要求した後に，別のアプリケーションが作動の終了を要求した場合，後の要求が有効となる。

本オブジェクトは，標準的なロック機能を持つLockableObjectである。状態を変化させるサービスコール（activate，deactivate，startParkIn，startParkOut）およびロックを取得するサービスコール（lock）は，操作優先度priorityを持つ。ロック中の本オブジェクトに対して，ロックと同じかそれより低い優先度で操作した場合はE_OBJECT_LOCKEDエラーとなり，より高い優先度で操作した場合は，既存のロックが強制解除されて操作が実行される。アプリケーションが使用できる最高値より高い優先度を指定した場合はE_DENIED_PRIORITYエラー，操作優先度が有効範囲外である場合はE_INVALID_PARAMETERエラーとなる。

自アプリケーションのロックが強制解除された場合はOwnLockRevokedイベントが，他のアプリケーション等によりロックが解除された場合はOtherLockReleasedイベントが発生する。作動状態がFaultとなった場合，ロックは解除され，その間は新たにロックを取得することができない。

運転者による直接の操作（車載のスイッチによる開始・終了，制動操作・操舵操作等）は，本APIを用いない一つのアプリケーションによる操作と見なし，後勝ちの原則に従う。この操作は，アプリケーションが取得したロックおよび操作優先度によって妨げられることはなく，常に有効となる。運転の責任を負うのは運転者であり，アプリケーションによる排他が運転者の操作に優先してはならないためである。駐車の実行中に運転者が運転操作を行った場合，車両は駐車の実行を一時中断（Suspended）するか中断（Interrupted）する。



Vehicle.Motionは，アプリケーションが車両の運動そのものを制御するためのオブジェクトであり，本オブジェクトは，車両に搭載された自動駐車に対して駐車の実行を要求するものである。両者は別の経路であり，同時に用いられた場合も後勝ちの原則に従う。駐車の実行中にVehicle.Motionにより車両の運動が制御された場合，本オブジェク

トはavailabilityをTemporarilyUnavailable，unavailableReasonをControlOverriddenとし，operatingStateをSuspendedとする。

5.8.1.8. 操作・参照に伴うリスク

入庫および出庫の実行は，運転者の運転操作なしに車両を動かすため，周囲の人・車両・障害物との衝突のリスクを伴う。これをリスククラスRiskCollisionとして定義し，startParkInおよびstartParkOutをリスク制御の対象とする。作動の終了（deactivate）は，駐車の実行中には車両を停止させるものであり，作動の開始（activate）は駐車の実行につながるものであるため，いずれも同じリスククラスの対象とする。

自動駐車の作動状態・駐車の進捗・駐車枠の候補は，運転者がいつどこに駐車したかを第三者が推定できる情報である。これをリスククラスRiskPrivacyとして定義し，状態を参照するサービスコール（getStatus），駐車枠の候補の参照（getParkingSlots），イベントの通知要求（notify）およびイベントの取得（getEvent）をリスク制御の対象とする。

いずれのリスククラスについても，当該リスクに対応していないアプリケーションからの呼び出しはE_RISK_APPLICATIONエラー，ユーザが当該リスクを承認していない場合はE_RISK_USERエラーとなる。

構成情報の参照（getConfig）は，車両に搭載された自動駐車の静的な性質を返すものであり，リスク制御の対象としない。

5.8.2. 機能クラス

AutomatedParkingは，以下の機能クラスを持つ。

ParallelParking	縦列駐車の駐車枠へ入庫することができる
PerpendicularParking	並列駐車の駐車枠へ入庫することができる
AngledParking	斜め駐車の駐車枠へ入庫することができる
ParkOut	駐車枠から出庫することができる

駐車枠の候補には，機能クラスが宣言する種別の駐車枠のみが現れる。機能クラスParkOutを備えない場合，出庫の要求（startParkOut）を用いることはできない。

5.8.3. リスククラス

AutomatedParkingは，以下のリスククラスを持つ。

RiskCollision	運転者の運転操作なしに車両が動くことにより，周囲の人・車両・障害物との衝突を生じるリスク
RiskPrivacy	走行状態および運転者の運転への関与状態の開示により，個人の行動を推定できることに伴うプライバシのリスク

5.8.4. 構成情報

AutomatedParkingは，以下の構成情報を持つ。

capabilities	enumSet (AutomatedParkingCapabilityType)	その車両の自動駐車が持つ機能
riskClasses	enumSet (AutomatedParkingRiskType)	車両およびビークルOSで対応できていないリスククラスの集合
supportedOperations	enumSet (AutomatedParkingOperationType)	アプリケーションから要求できる操作の集合。空集合となることがある
displayName	String?	ユーザに表示する呼称を取得するためのキー。車両が呼称を公開する場合にのみ値を持つ
timestampAccuracy	UInt32	この車両のタイムスタンプ誤差上限の宣言。単位はμs

上記は，YAML形式で記述した構成記述ファイル中に記述する形式である。本オブジェクトはシングルインスタンスオブジェクトであるため，構成記述では，車両を表すノードの下に置かれたADADASノードの下に，オブジェクトを表すノードAutomatedParkingを置き，その下に上記の構成情報を記述する。このノードは，Vehicle.ADADASの目録において，providingObjectがVehicle.ADADAS.AutomatedParkingである要素に対応する。

riskClassesには，車両およびビークルOSで対応できていない（危険を防止できていない）リスククラスの集合を記述する。

構成情報を参照するサービスコールは，この構成情報に対して，次の変換を行った構成情報を返す。

- displayNameに対しては，それを元に現在のロケールに対応する呼称の文字列を取得し，その文字列を返す。

5.8.5. 状態

AutomatedParkingは，以下の状態を持つ。

timestamp	TimestampUsType	状態の測定時刻。単位はμs。車内共通時刻基盤に基づく
dataQualifier	DataQualifierType	このサンプルの品質。InvalidまたはUnavailableの場合，以下の状態の値を読んではならない。値の変化はDataQualifierChangedイベントにより通知される
operatingState	AutomationOperatingStateType	現在の作動状態。自動駐車ではFallbackおよびMRCは現れない
availability	AutomationAvailabilityType	いま駐車の実行を開始または継続できるか
unavailableReason	AutomationUnavailableReasonType?	駐車の実行を開始または継続できない理由。availabilityがAvailable以外の値である場合にのみ値を持つ

blockingFunction	RelatedObjectType[]	駐車の実行を妨げている機能オブジェクト。unavailableReasonがDependentFunctionFaultまたはSupersededByOtherFunctionである場合にのみ要素を持つ
faultCategory	AutomationFaultCategoryType?	故障の区分。operatingStateがFaultである場合にのみ値を持つ
parkingPhase	AutomatedParkingPhaseType?	駐車の進捗。起動後に一度も作動していない場合は値を持たない
maneuver	AutomatedParkingManeuverType?	実行中または直前に実行した駐車が入庫か出庫か。parkingPhaseがManeuvering，CompletedまたはInterruptedである場合にのみ値を持つ
targetSlot	IdType?	入庫の目標の駐車枠の識別子。maneuverがParkInであり，かつparkingPhaseがManeuveringである場合にのみ値を持つ
interruptionReason	AutomatedParkingInterruptionReasonType?	駐車を中断した理由。parkingPhaseがInterruptedである場合にのみ値を持つ
lockStatus	LockStatusType	ロック状態

状態は，呼び出し時点の自動駐車の状態を返す冪等なスナップショットである。値を読む前にdataQualifierを確認する。

自動駐車が駐車を実行しているか否かは，operatingStateにより判定する。駐車の手順のどの段階にあるかは，parkingPhaseにより判定する。operatingStateとavailabilityの組合せの規則は，Vehicle.ADADASが規定するAD/ADAS機能共通の作動状態モデルによる。

故障により機能を提供できない場合，operatingStateはFaultとなり，faultCategoryがその区分を表す。駐車の実行中に故障した場合は，あわせてparkingPhaseがInterrupted，interruptionReasonがFunctionFaultとなる。本オブジェクト自体が状態を提供できない場合は，getStatusがE_OBJECT_FAULTエラーを返す。

5.8.6. サービスコール一覧

AutomatedParkingに対するサービスコールは以下の通り。

getConfig	AutomatedParkingの構成情報の参照
getStatus	AutomatedParkingの状態の参照
getParkingSlots	駐車枠の候補の参照
activate	AutomatedParkingの作動の開始の要求
deactivate	AutomatedParkingの作動の終了の要求
startParkIn	入庫の実行の要求
startParkOut	出庫の実行の要求
lock	AutomatedParkingのロックの取得
unlock	AutomatedParkingのロックの解除

notify	AutomatedParkingイベントの通知要求
unnotify	AutomatedParkingイベントの通知停止
getEvent	AutomatedParkingイベントの取得

5.8.7. イベント

イベントは，事象が発生したことのみを伝え，状態の値そのものを運ばない。ただし，事象の発生時点の情報として有用なもの（故障の区分等）は，イベントの関連情報としても渡す。アプリケーションは，イベントを受けた後，必要に応じてgetStatusまたはgetParkingSlotsにより値を取得する。

状態の取得は，要求応答（getStatus）と，各状態項目の変化イベントによる。

AutomatedParkingに対するイベント種別は以下の通り。

OperatingStateChanged	作動状態の変化
AvailabilityChanged	駐車の実行を開始または継続できるかの変化（駐車の実行を妨げている理由の変化を含む）
ParkingPhaseChanged	駐車の進捗（状態parkingPhase）の変化。完了および中断を含む
ParkingSlotsChanged	駐車枠の候補の追加または消失
FaultDetected	故障の検知（自動駐車の機能を提供できない状態）
FaultRecovered	故障からの回復
DataQualifierChanged	状態の品質（状態dataQualifier）の変化
OwnLockRevoked	自アプリケーションのロックの強制解除
OtherLockReleased	他のアプリケーション等によるロック解除
EventOverflow	イベントキューのオーバフロー
ConfigChanged	構成情報の変更（ソフトウェアの更新による対応する機能・操作の変更）

この内，イベント種別がFaultDetectedの場合には，以下の関連情報を持つ。

faultCategory	AutomationFaultCategoryType	故障の区分

また，イベント種別がEventOverflowの場合には，以下の関連情報を持つ。

noLostEvents	UInt32	失われたイベントの数

5.8.8. データ型の定義

5.8.8.1. AutomatedParkingの機能クラス

```
type AutomatedParkingCapabilityType = enum {
    ParallelParking,
    PerpendicularParking,
```

```
    AngledParking,
    ParkOut
};
```

5.8.8.2. AutomatedParkingのリスククラス

```
type AutomatedParkingRiskType = enum {
    RiskCollision,
    RiskPrivacy
};
```

5.8.8.3. タイムスタンプ

```
type TimestampUsType = UInt64;  // 車内共通時刻基盤の起点を基準とする測定時刻。単位はμs。複数のビークルAPIで共通に用いる共通データ型であり，本仕様内に定義する。
```

5.8.8.4. サンプル品質の種別

```
type DataQualifierType = enum {
    Normal, // 正常
    Unavailable,    // 利用不可
    Degraded,   // 劣化
    TestMode,   // テストモード
    Invalid // 無効（値を読んではならない）
};
```

複数のビークルAPIで共通に用いる共通データ型であり，本仕様内に定義する。

5.8.8.5. AD/ADAS機能共通の作動状態

以下の5つのデータ型は，AD/ADAS機能オブジェクトが共通に用いるデータ型であり，本仕様内に定義する。自動駐車における各列挙子の具体的な意味は，位置付けと機能の節に示す。

```
type AutomationOperatingStateType = enum {
    Off,    // 機能が無効化されており，作動していない状態
    Standby,    // 機能が有効であり，作動の条件が整えば作動できるが，現在は作動していない状態
    Activating, // 作動開始の要求を受理し，作動開始の処理を行っている過渡状態
    Active, // 作動中。車両の運動への介入または警報の出力を現に行っている状態
    Suspended,  // 作動が一時的に解除され，中断している状態。直前の作動時の設定値を保持しており，抑止する要因が解消すれば作動を再開できる
    Deactivating,   // 作動終了の要求を受理し，作動終了の処理を行っている過渡状態
    Fallback,   // 縮退運転を実行している状態
    MRC,    // 最小リスク状態へ移行し，これを維持している状態
    Fault   // 故障により機能を提供できない状態
```

```
};
```

各機能オブジェクトが現にどの作動状態にあるかを表す型である。機能によっては現れない列挙子がある。

```
type AutomationAvailabilityType = enum {
    Available,  // 作動を開始または継続できる
    TemporarilyUnavailable, // 一時的に条件を満たさないため，作動を開始または継続できない（条件が回復すれば作動できる）
    Unavailable // 故障等により，作動を開始または継続できない
};
```

その機能がいま作動を開始または継続できるかを表す型である。作動しているか否かは作動状態operatingStateが表すものであり，本型はこれを表さない。

```
type AutomationUnavailableReasonType = enum {
    ODDOutside, // 走行環境が運行設計領域の外にある
    SensorDegraded, // センサの性能低下・遮蔽により，必要な認識の性能が得られない
    DependentFunctionFault, // この機能が用いている他のAD/ADAS機能が故障している（blockingFunctionがその機能を指す）
    DriverMonitoringUnavailable,    // 運転者の監視が行えない
    VehicleStateNotReady,   // 車速・変速機の位置・制動等の車両の状態が作動の条件を満たさない
    FunctionFault,  // この機能自体が故障している
    ControlOverridden,  // 運転者の操作または他の経路による車両の運動の制御により，作動が抑止されている
    UserDisabled,   // 利用者の設定により機能が無効にされている
    SupersededByOtherFunction,  // 他のAD/ADAS機能がその運転操作を担っているため作動しない（blockingFunctionがその機能を指す）
    CalibrationRequired,    // 較正が必要で作動できない
    Other   // その他の理由
};
```

availabilityがAvailable以外の値である場合に，作動を開始または継続できない理由を表す型である。機能によっては現れない列挙子がある。

```
type AutomationOutputLevelType = enum {
    Detected,   // 事象を検知したが，運転者への提示はまだ行っていない
    Advisory,   // 注意喚起（表示等の弱い提示）
    Warning,    // 警報（音・振動を伴う提示）
    Critical    // 緊急（車両運動への介入を伴う）
};
```

機能の出力の段階を表す型である。Detected，Advisory，Warning，Criticalの順に強い段階を表す。機能によっては現れない段階がある。

```
type AutomationFunctionOutputType = struct {
```

```
    active : Bool,  // その機能が事象の検知または運転者への提示を現に行っているか
    level : AutomationOutputLevelType?  // 出力の段階。activeがtrueであり，かつ段階を
提供できる機能である場合にのみ値を持つ
};
```

機能が外部（運転者）に対して現に行っている出力を，有無と段階のみで表す型である。出力の判断の根拠（検知した車線の形状・物体の種別等）は本型に含まない。

5.8.8.6. AD/ADAS機能共通の故障の区分

以下のデータ型は，AD/ADAS機能オブジェクトが共通に用いるデータ型であり，本仕様内に定義する。

```
type AutomationFaultCategoryType = enum {
    SensorFault,    // センサの故障
    ActuatorFault,  // アクチュエータの故障
    ComputationFault,   // 演算装置の故障
    CommunicationFault, // 車内通信の故障
    PowerFault, // 電源の異常
    WarningDeviceFault, // 警報を提示する装置の故障
    Other   // その他の故障
};
```

機能を提供できない故障の区分を表す型である。機能によっては現れない列挙子がある。故障の区分は，各機能オブジェクトの状態faultCategoryおよびFaultDetectedイベントの関連情報により渡される。

5.8.8.7. AutomatedParkingの操作

```
type AutomatedParkingOperationType = enum {
    Activate,   // 作動の開始の要求（サービスコールactivate）
    Deactivate, // 作動の終了の要求（サービスコールdeactivate）
    StartParkIn,    // 入庫の実行の要求（サービスコールstartParkIn）
    StartParkOut    // 出庫の実行の要求（サービスコールstartParkOut）
};
```

アプリケーションから要求できる操作を表す型である。列挙子は，操作を行うサービスコールと1対1に対応する。すべてのアプリケーションが呼び出すことができるサービスコール（getConfig，getStatus，getParkingSlots，notify，unnotify，getEvent，lock，unlock）は，本型の対象としない。

5.8.8.8. 駐車の進捗

```
type AutomatedParkingPhaseType = enum {
    Searching,  // 駐車枠を探索しており，駐車枠の候補がない
    SlotDetected,   // 駐車枠を探索しており，駐車枠の候補が1つ以上ある
    Maneuvering,    // 入庫または出庫を実行している（一時中断を含む）
    Completed,  // 入庫または出庫を終え，車両が停止を保持している
    Interrupted // 入庫または出庫を終える前に駐車を中断し，車両が停止を保持している
};
```

駐車の手順のどの段階にあるかを表す型である。Searching，SlotDetectedは作動状態がStandbyである間に，Maneuveringは作動状態がActivating，ActiveまたはSuspendedである間に現れる。

```
type AutomatedParkingManeuverType = enum {
    ParkIn, // 入庫（駐車枠へ車両を入れる）
    ParkOut // 出庫（駐車枠から車両を出す）
};
```

```
type AutomatedParkingInterruptionReasonType = enum {
    CancelRequested,    // アプリケーションまたは運転者により作動の終了が要求された
    Overridden, // 運転者の運転操作または他の経路による車両の運動の制御により，駐車を継続できなくなった
    ObstacleDetected,   // 障害物により，駐車を継続できなくなった
    SlotUnavailable,    // 目標の駐車枠に入庫できなくなった
    DriverAbsent,   // 運転者が運転席を離れた，またはドアが開かれた
    FunctionFault,  // 自動駐車の機能が故障した
    Other   // その他の理由
};
```

入庫または出庫を終える前に駐車を中断した理由を表す型である。

5.8.8.9. 駐車枠の候補

```
type AutomatedParkingSlotKindType = enum {
    Parallel,   // 縦列駐車の駐車枠（車両の進行方向に沿った枠）
    Perpendicular,  // 並列駐車の駐車枠（車両の進行方向に直交する枠）
    Angled  // 斜め駐車の駐車枠（車両の進行方向に斜めに交わる枠）
};
```

```
type AutomatedParkingDirectionType = enum {
    Front,  // 自車の前方
    Rear,   // 自車の後方
    Left,   // 自車の左方
    Right   // 自車の右方
};
```

自車から見た方向を表す型である。駐車枠の候補の方向，および出庫する方向に用いる。

```
type AutomatedParkingSlotType = struct {
    slotId : IdType,    // 駐車枠の候補の識別子。入庫の要求（startParkIn）で指定する
    kind : AutomatedParkingSlotKindType,    // 駐車枠の種別
    direction : AutomatedParkingDirectionType   // 自車から見た駐車枠の方向
};
```

1つの駐車枠の候補を表す型である。駐車枠の位置・形状の座標は含まない。

5.8.8.10. AutomatedParkingの構成情報（getConfigが返すデータ型）

```
type AutomatedParkingConfigType = struct {
    capabilities : enumSet(AutomatedParkingCapabilityType),
    riskClasses : enumSet(AutomatedParkingRiskType),
    supportedOperations : enumSet(AutomatedParkingOperationType),
    displayName : String?,
    timestampAccuracy : UInt32
};
```

5.8.8.11. AutomatedParking状態

```
type AutomatedParkingStatusType = struct {
    timestamp : TimestampUsType,
    dataQualifier : DataQualifierType,
    operatingState : AutomationOperatingStateType,
    availability : AutomationAvailabilityType,
    unavailableReason : AutomationUnavailableReasonType?,
    blockingFunction : RelatedObjectType[],
    faultCategory : AutomationFaultCategoryType?,
    parkingPhase : AutomatedParkingPhaseType?,
    maneuver : AutomatedParkingManeuverType?,
    targetSlot : IdType?,
    interruptionReason : AutomatedParkingInterruptionReasonType?,
    lockStatus : LockStatusType
};
```

呼び出し時点の自動駐車の状態の冪等なスナップショットである。

5.8.8.12. AutomatedParkingイベント

```
type AutomatedParkingEventKindType = enum {
    OperatingStateChanged,
    AvailabilityChanged,
    ParkingPhaseChanged,
    ParkingSlotsChanged,
    FaultDetected,
    FaultRecovered,
    DataQualifierChanged,
    OwnLockRevoked,
    OtherLockReleased,
    EventOverflow,
    ConfigChanged
};
```

```
type AutomatedParkingEventType = struct {
    eventInfo : variant(AutomatedParkingEventKindType) {
```

```
        FaultDetected {
            faultCategory : AutomationFaultCategoryType
        },
        EventOverflow {
            noLostEvents : UInt32
        }
    }
};
```

5.8.9. サービスコールの詳細規定

getConfig

AutomatedParkingの構成情報の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
config	AutomatedParkingConfigType	AutomatedParkingの構成情報

【エラーコード】

なし

【機能】

その車両の自動駐車が持つ機能と，アプリケーションから要求できる操作の集合を返す。

アプリケーションは，本サービスコールにより，画面を構成する前に，どの操作の要素を提示すべきかを決める。

getStatus

AutomatedParkingの状態の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
status	AutomatedParkingStatusType	AutomatedParkingの状態

【エラーコード】

E_OBJECT_FAULT	本オブジェクトが故障しており，状態を提供できない
E_RISK_APPLICATION	アプリケーションがRiskPrivacyに対応していない
E_RISK_USER	ユーザがRiskPrivacyを承認していない

【機能】

自動駐車の現在の作動状態・利用可否・故障の区分・駐車の進捗・入庫と出庫の別・目標の駐車枠・中断の理由・ロック状態を返す。

呼び出しのたびに，その時点の状態の全体を返す冪等な応答である。値を読む前にdataQualifierを確認する。

getParkingSlots

駐車枠の候補の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
slots	AutomatedParkingSlotType[]	駐車枠の候補の一覧

【エラーコード】

E_OBJECT_FAULT	本オブジェクトが故障しており，駐車枠の候補を提供できない
E_RISK_APPLICATION	アプリケーションがRiskPrivacyに対応していない
E_RISK_USER	ユーザがRiskPrivacyを承認していない

【機能】

呼び出し時点の駐車枠の候補の一覧を返す。作動状態がStandby以外である場合，および候補が無い場合は，空の配列を返す。

アプリケーションは，返された候補の中から入庫する駐車枠を選び，そのslotIdを入庫の要求（startParkIn）で指定する。候補の追加・消失はParkingSlotsChangedイベントにより通知される。

activate

AutomatedParkingの作動の開始の要求

【パラメータ】

priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_PARAMETER	priorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_LOCKED	本オブジェクトが，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている
E_OBJECT_STATUS	作動を開始できる作動状態でない（既に有効である，またはavailabilityがUnavailableであり，その理由が故障以外である）
E_OBJECT_FAULT	本オブジェクトが故障しており，要求を受け付けられない（unavailableReasonがFunctionFaultである場合を含む）
E_INFEASIBLE	availabilityがTemporarilyUnavailableであり，作動の条件以外の一時的な理由（故障等）により，現在はAutomatedParkingを有効にできない
E_RISK_APPLICATION	アプリケーションがRiskCollisionに対応していない
E_RISK_USER	ユーザがRiskCollisionを承認していない

【機能】

自動駐車に作動の開始を要求する。本サービスコールは，自動駐車を有効にする操作に相当し，OffからStandbyへの遷移を要求する。遷移すると，車両は駐車枠の探索を始める。車両の運動の制御は開始されない。

正常終了は要求が受理されたことのみを意味し，作動が開始したことを意味しない。遷移の完了は，getStatusまたはOperatingStateChangedイベントにより確認する。

作動を開始できない場合の応答は，利用可否availabilityとその理由unavailableReasonにより定まる。availabilityがTemporarilyUnavailableである場合はE_INFEASIBLEエラー，Unavailableでありunavailable ReasonがFunctionFaultである場合はE_OBJECT_FAULTエラー，Unavailableでありそれ以外の理由である場合はE_OBJECT_STATUSエラーとなる。

本サービスコールは，構成情報supportedOperationsにActivateを含む場合にのみ用いることができる。

deactivate

AutomatedParkingの作動の終了の要求

【パラメータ】

priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_PARAMETER	priorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_LOCKED	本オブジェクトが，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている
E_OBJECT_STATUS	作動を終了できる作動状態でない（既にOffである等）
E_OBJECT_FAULT	本オブジェクトが故障しており，要求を受け付けられない
E_RISK_APPLICATION	アプリケーションがRiskCollisionに対応していない
E_RISK_USER	ユーザがRiskCollisionを承認していない

【機能】

自動駐車に作動の終了を要求する。本サービスコールは，自動駐車を無効にする操作に相当し，Offへの遷移を要求する。

駐車の実行中（作動状態がActiveまたはSuspended）に要求した場合，駐車は中断され，interruptionReasonはCancelRequestedとなる。

正常終了は要求が受理されたことのみを意味し，作動が終了したことを意味しない。遷移の完了は，getStatusまたはOperatingStateChangedイベントにより確認する。

本サービスコールは，構成情報supportedOperationsにDeactivateを含む場合にのみ用いることができる。



駐車の実行中に作動を終了させると，車両は入庫または出庫を終えていない位置で停止する。車両が通路等に停止したまま運転者に運転操作が委ねられるため，アプリケーションは，駐車を中断したことを運転者が認識できるようにしなければならない。

startParkIn

入庫の実行の要求

【パラメータ】

slotId	IdType	入庫する駐車枠の識別子。Standbyにおいては駐車枠の候補の識別子（getParkingSlotsで取得したもの），Suspendedにおいては中断している入庫の目標の駐車枠の識別子（状態targetSlot）
priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_ID	slotIdが有効範囲外，または作動状態がStandbyであり，slotIdが駐車枠の候補にない

E_INVALID_PARAMETER	priorityが有効範囲外，または作動状態がSuspendedであり，slotIdが中断している入庫の目標の駐車枠と異なる
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_LOCKED	本オブジェクトが，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている
E_OBJECT_STATUS	入庫を開始できる作動状態でない（Standby，または入庫を中断しているSuspended以外である），またはavailabilityがUnavailableであり，その理由が故障以外である
E_OBJECT_FAULT	本オブジェクトが故障しており，要求を受け付けられない（unavailableReasonがFunctionFaultである場合を含む）
E_INFEASIBLE	availabilityがTemporarilyUnavailableであり，現在の車両の状態または周囲の環境では入庫を開始できない
E_RISK_APPLICATION	アプリケーションがRiskCollisionに対応していない
E_RISK_USER	ユーザがRiskCollisionを承認していない

【機能】

slotIdで指定した駐車枠への入庫の実行を要求する。作動状態がStandbyである場合は入庫の開始を，入庫を中断しているSuspendedである場合は入庫の再開を，それぞれ要求する。再開の場合，slotIdには中断している入庫の目標の駐車枠（状態targetSlot）を指定する。再開の場合のslotIdは，駐車枠の候補の一覧ではなくtargetSlotとの一致により判定し，一致しない場合はE_INVALID_PARAMETERエラーとなる。

入庫の実行中のDDTの責任，および入庫を終えた後の作動状態は，位置付けと機能の節の規定による。運転者が運転席にいない場合は，E_INFEASIBLEエラーとなる。

正常終了は要求が受理されたことのみを意味し，入庫が開始したことを意味しない。駐車の進捗の変化はParkingPhaseChangedイベント，作動状態の変化はOperatingStateChangedイベントにより通知される。

本サービスコールは，構成情報supportedOperationsにStartParkInを含む場合にのみ用いることができる。

startParkOut

出庫の実行の要求

【パラメータ】

direction	AutomatedParkingDirectionType?	出庫する方向。省略した場合，車両が方向を決める
priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_PARAMETER	direction，priorityが有効範囲外

E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_LOCKED	本オブジェクトが，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている
E_OBJECT_STATUS	出庫を開始できる作動状態でない（Standby，または出庫を中断しているSuspended以外である），またはavailabilityがUnavailableであり，その理由が故障以外である
E_OBJECT_FAULT	本オブジェクトが故障しており，要求を受け付けられない（unavailableReasonがFunctionFaultである場合を含む）
E_INFEASIBLE	availabilityがTemporarilyUnavailableである，車両が駐車枠に停車していない，または指定された方向に出庫できない
E_RISK_APPLICATION	アプリケーションがRiskCollisionに対応していない
E_RISK_USER	ユーザがRiskCollisionを承認していない

【機能】

駐車枠に停車している車両の出庫の実行を要求する。作動状態がStandbyである場合は出庫の開始を，出庫を中断しているSuspendedである場合は出庫の再開を，それぞれ要求する。

出庫の実行中のDDTの責任，および出庫を終えた後の作動状態は，位置付けと機能の節の規定による。運転者が運転席にいない場合は，E_INFEASIBLEエラーとなる。

正常終了は要求が受理されたことのみを意味し，出庫が開始したことを意味しない。駐車の進捗の変化はParkingPhaseChangedイベント，作動状態の変化はOperatingStateChangedイベントにより通知される。

本サービスコールは，機能クラスParkOutを備え，かつ構成情報supportedOperationsにStartParkOutを含む場合にのみ用いることができる。

lock

AutomatedParkingのロックの取得

【パラメータ】

priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_PARAMETER	priorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_LOCKED	本オブジェクトが，priorityと同じかそれより高い優先度でロックされている
E_OBJECT_FAULT	本オブジェクトが故障しており，ロックを取得できない

【機能】

本オブジェクトを，指定した操作優先度でロックする。ロックを取得したアプリケーションは，状態を変化させるサービスコール（activate，deactivate，startParkIn，startParkOut）を，同じかそれより低い優先度の他のアプリケーションによる操作から保護する。

既存のロックより高い優先度でロックを取得した場合，既存のロックは強制解除され，そのロックを取得していたアプリケーションにはOwnLockRevokedイベントが通知される。

運転者による直接の操作は，ロックおよび操作優先度によって妨げられない。

unlock

AutomatedParkingのロックの解除

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_OBJECT_STATUS	自アプリケーションが，本オブジェクトをロックしていない

【機能】

自アプリケーションが取得した本オブジェクトのロックを解除する。

notify

AutomatedParkingイベントの通知要求

【パラメータ】

eventFilter	enumSet (AutomatedParkingEventKindType)?	通知するイベントの種類の集合

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【エラーコード】

E_INVALID_PARAMETER	eventFilterが有効範囲外
E_RISK_APPLICATION	アプリケーションがRiskPrivacyに対応していない
E_RISK_USER	ユーザがRiskPrivacyを承認していない

【機能】

AutomatedParkingで発生したeventFilterで指定したイベントの通知を要求する。イベントを通知するためのイベントキューを生成し，その通知ハンドルをnotifyHandleに返す。

eventFilterを省略した場合，すべての種類のイベントの通知を要求する。

駐車の開始・完了の観測からも運転者の行動を推定できるため，本サービスコールもRiskPrivacyのリスク制御の対象である。

unnotify

AutomatedParkingイベントの通知停止

【パラメータ】

notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正

【機能】

notifyHandleで指定した通知を停止する。指定した通知用のイベントキューは削除される。

getEvent

AutomatedParkingイベントの取得

【パラメータ】

notifyHandle	NotifyHandleType	イベントキューを識別するための通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
event	AutomatedParkingEventType	取得したイベント

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正
E_NO_DATA	イベントキューが空
E_RISK_APPLICATION	アプリケーションがRiskPrivacyに対応していない
E_RISK_USER	ユーザがRiskPrivacyを承認していない

【機能】

notifyHandleで指定した通知用のイベントキューから，先頭のイベントを取り出す。取り出したイベントは，イベントキューから削除される。

イベントの内容からも運転者の行動を推定できるため，本サービスコールもRiskPrivacyのリスク制御の対象である。



イベントがパラメータ等で渡されてくるプログラミング環境向けの物理APIでは，用意する必要がない。

5.9. Vehicle.ADADAS.BSM（略称：BSM）

5.9.1. 位置付けと機能

Vehicle.ADADAS.BSM（略称：BSM）は，車両に搭載された死角検知（BSM）を参照・操作するためのシングルインスタンスオブジェクトである。作動の開始・終了の要求と，現在の作動状態および警報の出力の状況の参照を提供する。

本オブジェクトは，Vehicle.ADADASの目録に，機能の種別BSM，提供するオブジェクトの完全修飾名Vehicle.ADADAS.BSMとして現れる。DDTの責任主体・運行設計領域の充足状態・縮退運転・最小リスク状態など，AD/ADASシステム全体について一つに定まる状態は，本オブジェクトではなくVehicle.ADADASが提供する。

5.9.1.1. BSMの位置付けと性質

BSMは，外界センサにより自車の側方および後側方の死角にいる車両を検知し，運転者に警報を出力する運転支援の機能である。BSMは車両の運動に介入しない。警報のみを行い，操舵・制動の介入は行わないため，BSMの作動中も車両の運動の制御は運転者が行う。

本オブジェクトは，BSMが用いるセンサの構成，検知の範囲，車両の検知と警報の判定のアルゴリズムを規定しない。検知した車両の位置・相対速度・台数といった判定の根拠は本オブジェクトでは扱わない。本オブジェクトが提供するのは，機能が作動しているか（operatingState），いま作動を継続できるか（availability），現に警報を出力しているか（functionOutput），左右それぞれの側で警報を出力しているか（leftOutput，rightOutput），および故障の区分（faultCategory）である。

5.9.1.2. 作動の開始・終了の操作

本オブジェクトは，作動の開始（activate）と終了（deactivate）をアプリケーションから要求できる。車両が備える設定の手段（車載の設定画面等）により運転者がBSMを有効化・無効化した場合も，同じ作動状態operatingStateに反映される。

アプリケーションが行える操作は，構成情報supportedOperationsが宣言する。supportedOperationsの列挙子は，操作を行うサービスコールと1対1に対応する。アプリケーションは，画面を構成する前にsupportedOperationsを参照し，宣言されていない操作の要素を提示してはならない。宣言されていない操作のサービスコールを呼び出した場合，E_NOT_SUPPORTEDエラーとなる。

supportedOperationsが操作の可否（恒久的な可否）を表すのに対し，availabilityは，いまその操作を受け付けられるか（一時的な可否）を表す。この2つは別の判定であり，アプリケーションは，前者により要素を提示するか否かを，後者により要素を操作可能とするか否かを決める。

警報を出力しないようにすることは，死角にいる車両に運転者が気付かない状態を生じうる。作動の終了

の要求に伴うリスクは，操作・参照に伴うリスクの節に規定する。

5.9.1.3. 作動状態

本オブジェクトの作動状態operatingStateおよび利用可否availabilityは，Vehicle.ADADASが規定するAD/ADAS機能共通の作動状態モデルに従う。本節では，BSMに固有の制約のみを規定する。

BSMにおける各作動状態の意味は，次のとおりである。

- Off … BSMが無効化されており，死角を監視していない状態。アプリケーションによる作動の終了の要求，および運転者の設定のいずれによっても，この状態となる。
- Standby … BSMが有効であるが，車速が警報の対象となる範囲の外にある等の理由で，死角を監視していない状態。
- Activating … 作動開始の要求（activate）を受理し，遷移の処理を行っている過渡状態。
- Active … BSMが有効であり，死角を監視している状態。この状態において，死角に車両を検知すると出力が生じ，functionOutputのactiveがtrueとなる。
- Deactivating … 作動終了の要求（deactivate）を受理し，Offへ遷移する処理を行っている過渡状態。
- Suspended … 運転操作の全体を担う他のAD/ADAS機能が作動しているため，死角の監視と警報を行わない状態。この場合，availabilityはTemporarilyUnavailable，unavailableReasonはSupersededByOtherFunctionとなり，blockingFunctionがその機能を指す。
- Fault … 故障により機能を提供できない状態。故障の区分は状態faultCategoryが表す。

BSMでは，FallbackおよびMRCの列挙子は現れない。縮退運転および最小リスク状態への移行は，DDTの責任を持つ機能が行うものであり，警報のみを行うBSMはこれを行わないためである。

BSMが有効である間に車速が警報の対象となる範囲を外れた場合は，Suspendedではなく，Standbyかつavailability がTemporarilyUnavailableで表す。

operatingStateがOffである間，availabilityは，車速によらず，BSMを有効にできるか（故障の有無等）のみを表す。車速の範囲等の，有効化した後にのみ意味を持つ作動の条件はavailabilityに反映せず，車速が警報の対象となる範囲の外にあることのみを理由として，availability がTemporarilyUnavailableとなることはない。アプリケーションは，車速によらず作動の開始（activate）を要求でき，受理された場合，作動の条件が整っていなければoperatingStateはStandbyへ遷移する。作動の条件を満たさないことは，Standbyへ遷移した後のavailabilityおよびunavailableReasonにより表す。

運転者の設定によりBSMが無効化されている場合も，operatingStateはOffとなる。このとき，BSMを有効にできればavailabilityはAvailableであり，アプリケーションは作動の開始（activate）を要求できる。この要求は，後勝ちの原則に従い，運転者の設定より後に行われた操作として受理される。アプリケーションは，作動状態の変化をOperatingStateChangedイベント，利用可否の変化をAvailabilityChangedイベントにより知ることができる。

車両が，アプリケーションからの作動の開始・終了そのものを認めない場合は，構成情報supportedOperationsにActivateおよびDeactivateを含めないことで表す。

警報を出力しているか否かは，作動状態ではなくfunctionOutputで表す。作動状態Activeは「監視しており，いつでも警報を出力できる状態」を表し，警報の出力そのものは示さない。

5.9.1.4. 警報の出力

警報の出力の状況は，左右の側ごとに状態leftOutputおよびrightOutputが表し，両側をまとめた状況を状態functionOutputが表す。左右は，自車の前方に向かっての左側・右側をいう。左側の死角にいる車両に対する出力はleftOutputに，右側の死角にいる車両に対する出力はrightOutputに現れる。

functionOutputは，leftOutputとrightOutputから次のように定まる。

- activeは，leftOutputとrightOutputの少なくとも一方のactiveがtrueである場合にtrueとなり，それ以外の場合にfalseとなる。
- levelは，activeがtrueである側の段階のうち，最も強い段階となる。

各出力のactiveは，その側の死角に車両を検知しているか，または運転者への提示を現に行っているかを表す。levelは出力の段階を表し，機能クラスWarningLevelReportedを備えている場合にのみ値を持つ。BSMにおける各段階の意味は，次のとおりである。

- Detected ... その側の死角に車両を検知したが，まだ運転者への提示を行っていない状態。
- Advisory ... その側の死角に車両がいることを，表示器の点灯等の表示により運転者に提示している状態。
- Warning ... その側の死角に車両がいる状態で衝突の危険が高まったと車両が判断した場合に，音・振動を伴う提示を行っている状態。典型的には，運転者がその側への方向指示器を操作した場合である。

BSMは車両の運動に介入しないため，Criticalは現れない。その機能が出しうる最も強い段階は，構成情報maximumWarningLevelが宣言する。

警報の提示の方法（表示・音・触覚）は，機能クラスVisualWarning，AudibleWarning，HapticWarningが宣言する。どの方法で提示するかは車両実装に依存し，警報1回ごとの提示の方法は本オブジェクトでは表さない。

functionOutput，leftOutputおよびrightOutputのいずれかの値が変化したときは，FunctionOutputChangedイベントが発生する。一方の側の出力が変化し，functionOutputが変化しない場合にも発生する。イベントは事象の発生のみを伝え，値そのものは運ばない。値は，イベントを受けたアプリケーションがgetStatusにより取得する。



検知した車両の位置・相対速度・台数は本オブジェクトでは提供しない。これらはVehicle.SurroundModelの担当である。

5.9.1.5. BSMに対する制御の調停

本オブジェクトに対する制御の調停には，標準的な調停機能（後勝ちの原則）が適用される。すなわち，あるアプリケーションが作動を開始した後に，別のアプリケーションが作動の終了を要求した場合，後の要求が有効となる。

本オブジェクトは，標準的なロック機能を持つLockableObjectである。状態を変化させるサービスコール（activate，deactivate）およびロックを取得するサービスコール（lock）は，操作優先度priorityを持つ。ロック中の本オブジェクトに対して，ロックと同じかそれより低い優先度で操作した場合はE_OBJECT_LOCKEDエラーとなり，より高い優先度で操作した場合は，既存のロックが強制解除されて操作が実行される。アプリケーションが使用できる最高値より高い優先度を指定した場合はE_DENIED_PRIORITYエラー，操作優先度が有効範囲外である場合はE_INVALID_PARAMETERエラーとなる。

自アプリケーションのロックが強制解除された場合はOwnLockRevokedイベントが，他のアプリケーション等によりロックが解除された場合はOtherLockReleasedイベントが発生する。作動状態がFaultとなった場合，ロックは解除され，その間は新たにロックを取得することができない。

運転者による直接の操作（車載の設定画面等による有効化・無効化）は，本APIを用いない一つのアプリケーションによる操作と見なし，後勝ちの原則に従う。この操作は，アプリケーションが取得したロックおよび操作優先度によって妨げられることはなく，常に有効となる。運転の責任を負うのは運転者であり，アプリケーションによる排他が運転者の操作に優先してはならないためである。

5.9.1.6. 操作・参照に伴うリスク

BSMの作動を終了させると，死角にいる車両の警報が出力されなくなり，運転者がその車両に気付かないまま車線を変更することにつながりうる。これをリスククラスRiskCollisionとして定義し，activateおよびdeactivateをリスク制御の対象とする。作動の開始も，警報の出力により運転者の注意を引くものであり，同じリスククラスの対象とする。

BSMの作動状態および警報の出力の履歴は，運転者がいつどのように運転していたか（車線の変更の操作等）を第三者が推定できる情報である。これをリスククラスRiskPrivacyとして定義し，状態を参照するサービスコール（getStatus），イベントの通知要求（notify）およびイベントの取得（getEvent）をリスク制御の対象とする。

いずれのリスククラスについても，当該リスクに対応していないアプリケーションからの呼び出しはE_RISK_APPLICATIONエラー，ユーザが当該リスクを承認していない場合はE_RISK_USERエラーとなる。

構成情報の参照（getConfig）は，車両に搭載されたBSMの静的な性質を返すものであり，リスク制御の対象としない。

5.9.2. 機能クラス

BSMは，以下の機能クラスを持つ。

WarningLevelReported	出力の段階（検知，注意喚起，警報）を提供することができる（状態functionOutput，leftOutputおよびrightOutputのlevel）
VisualWarning	表示による警報を提示することができる
AudibleWarning	音による警報を提示することができる
HapticWarning	触覚（ステアリングまたは座席の振動等）による警報を提示することができる

5.9.3. リスククラス

BSMは，以下のリスククラスを持つ。

RiskCollision	BSMの作動の開始・終了により警報の出力の有無が変わり，運転者が死角にいる車両に気付かないまま車線を変更して衝突を生じるリスク
RiskPrivacy	走行状態および運転者の運転への関与状態の開示により，個人の行動を推定できることに伴うプライバシのリスク

5.9.4. 構成情報

BSMは，以下の構成情報を持つ。

capabilities	enumSet (BSMCapabilityType)	その車両のBSMが持つ機能
riskClasses	enumSet(BSMRiskType)	車両およびビークルOSで対応できていないリスククラスの集合
supportedOperations	enumSet (BSMOperationType)	アプリケーションから要求できる操作の集合。空集合となることがある
displayName	String?	ユーザに表示する呼称を取得するためのキー。車両が呼称を公開する場合にのみ値を持つ
warningSpeedRange	BSMSpeedRangeType	警報の対象となる車速の範囲。BSMが有効である間，この範囲の外では，作動状態はStandbyとなる
maximumWarningLevel	AutomationOutputLevelType?	この機能が出しうる最も強い段階。機能クラスWarningLevelReportedを備えている場合にのみ値を持つ
timestampAccuracy	UInt32	この車両のタイムスタンプ誤差上限の宣言。単位はμs

上記は，YAML形式で記述した構成記述ファイル中に記述する形式である。本オブジェクトはシングルインスタンスオブジェクトであるため，構成記述では，車両を表すノードの下に置かれたADADASノードの下に，オブジェクトを表すノードBSMを置き，その下に上記の構成情報を記述する。このノードは，Vehicle.ADADASの目録において，providingObjectがVehicle.ADADAS.BSMである要素に対応する。

riskClassesには，車両およびビークルOSで対応できていない（危険を防止できていない）リスククラスの集合を記述する。

構成情報を参照するサービスコールは，この構成情報に対して，次の変換を行った構成情報を返す。

- displayNameに対しては，それを元に現在のロケールに対応する呼称の文字列を取得し，その文字列を返す。

5.9.5. 状態

BSMは，以下の状態を持つ。

timestamp	TimestampUsType	状態の測定時刻。単位はμs。車内共通時刻基盤に基づく
dataQualifier	DataQualifierType	このサンプルの品質。InvalidまたはUnavailableの場合，以下の状態の値を読んではならない。値の変化はDataQualifierChangedイベントにより通知される
operatingState	AutomationOperatingStateType	現在の作動状態。BSMではFallbackおよびMRCは現れない
availability	AutomationAvailabilityType	いま死角の監視を開始または継続できるか
unavailableReason	AutomationUnavailableReasonType?	監視を開始または継続できない理由。availabilityがAvailable以外の値である場合にのみ値を持つ

blockingFunction	RelatedObjectType[]	監視を妨げている機能オブジェクト。unavailableReasonがDependentFunctionFaultまたはSupersededByOtherFunctionである場合にのみ要素を持つ
faultCategory	AutomationFaultCategoryType?	故障の区分。operatingStateがFaultである場合にのみ値を持つ
functionOutput	AutomationFunctionOutputType	警報の出力の状況（左右の側をまとめたもの）。levelは機能クラスWarningLevelReportedを備えている場合にのみ値を持つ
leftOutput	AutomationFunctionOutputType	左側の死角にいる車両に対する警報の出力の状況。levelは機能クラスWarningLevelReportedを備えている場合にのみ値を持つ
rightOutput	AutomationFunctionOutputType	右側の死角にいる車両に対する警報の出力の状況。levelは機能クラスWarningLevelReportedを備えている場合にのみ値を持つ
lockStatus	LockStatusType	ロック状態

状態は，呼び出し時点のBSMの状態を返す冪等なスナップショットである。値を読む前にdataQualifierを確認する。

BSMが警報を出せる状態にあるかは，operatingStateにより判定する。いま死角に車両を検知しているか，警報を出しているか，それがどちらの側かは，functionOutput，leftOutputおよびrightOutputにより判定する。operatingStateがActive以外である間，これらのactiveは常にfalseである。

operatingStateとavailabilityの組合せの規則は，Vehicle.ADADASが規定するAD/ADAS機能共通の作動状態モデルによる。BSMが有効である間に車速が構成情報warningSpeedRangeの範囲の外にある場合，availabilityはTemporarilyUnavailable，unavailableReasonはVehicleStateNotReadyとなり，operatingStateはStandbyとなる。operatingStateがOffである間のavailabilityは，車速を反映しない。

故障により機能を提供できない場合，operatingStateはFaultとなり，faultCategoryがその区分を表す。これにより，FaultDetectedイベントを受け取っていないアプリケーションも，状態の参照のみで故障の区分を知ることができる。本オブジェクト自体が状態を提供できない場合は，getStatusがE_OBJECT_FAULTエラーを返す。

5.9.6. サービスコール一覧

BSMに対するサービスコールは以下の通り。

getConfig	BSMの構成情報の参照
getStatus	BSMの状態の参照
activate	BSMの作動の開始の要求
deactivate	BSMの作動の終了の要求
lock	BSMのロックの取得
unlock	BSMのロックの解除
notify	BSMイベントの通知要求
unnotify	BSMイベントの通知停止

getEvent	BSMイベントの取得

5.9.7. イベント

イベントは，事象が発生したことのみを伝え，状態の値そのものを運ばない。ただし，事象の発生時点の情報として有用なもの（故障の区分等）は，イベントの関連情報としても渡す。アプリケーションは，イベントを受けた後，必要に応じてgetStatusにより値を取得する。

状態の取得は，要求応答（getStatus）と，各状態項目の変化イベントによる。

BSMに対するイベント種別は以下の通り。

OperatingStateChanged	作動状態の変化（アプリケーションの要求および運転者の設定による有効化・無効化を含む）
AvailabilityChanged	死角の監視を開始または継続できるかの変化（監視を妨げている理由の変化を含む）
FunctionOutputChanged	死角にいる車両に対する警報の出力の状況（状態functionOutput，leftOutputおよびrightOutput）の変化。いずれかの側の出力の開始・段階の変化・終了のいずれでも発生する
FaultDetected	故障の検知（BSMの機能を提供できない状態）
FaultRecovered	故障からの回復
DataQualifierChanged	状態の品質（状態dataQualifier）の変化
OwnLockRevoked	自アプリケーションのロックの強制解除
OtherLockReleased	他のアプリケーション等によるロック解除
EventOverflow	イベントキューのオーバフロー
ConfigChanged	構成情報の変更（ソフトウェアの更新による対応する操作・警報の対象となる車速の範囲・出力の段階の変更）

この内，イベント種別がFaultDetectedの場合には，以下の関連情報を持つ。

faultCategory	AutomationFaultCategoryType	故障の区分

また，イベント種別がEventOverflowの場合には，以下の関連情報を持つ。

noLostEvents	UInt32	失われたイベントの数

5.9.8. データ型の定義

5.9.8.1. BSMの機能クラス

```
type BSMCapabilityType = enum {
    WarningLevelReported,
    VisualWarning,
    AudibleWarning,
    HapticWarning
```

```
};
```

5.9.8.2. BSMのリスククラス

```
type BSMRiskType = enum {
    RiskCollision,
    RiskPrivacy
};
```

5.9.8.3. タイムスタンプ

```
type TimestampUsType = UInt64;  // 車内共通時刻基盤の起点を基準とする測定時刻。単位はμs。複数のビークルAPIで共通に用いる共通データ型であり，本仕様内に定義する。
```

5.9.8.4. サンプル品質の種別

```
type DataQualifierType = enum {
    Normal, // 正常
    Unavailable,    // 利用不可
    Degraded,   // 劣化
    TestMode,   // テストモード
    Invalid // 無効（値を読んではならない）
};
```

複数のビークルAPIで共通に用いる共通データ型であり，本仕様内に定義する。

5.9.8.5. 速度

```
type SpeedType = Float32;   // 速度。単位はkm/h。正の値は前進，負の値は後退を示す。複数のビークルAPIで共通に用いる共通データ型であり，本仕様内に定義する。
```

5.9.8.6. AD/ADAS機能共通の作動状態

以下の5つのデータ型は，AD/ADAS機能オブジェクトが共通に用いるデータ型であり，本仕様内に定義する。BSMにおける各列挙子の具体的な意味は，位置付けと機能の節に示す。

```
type AutomationOperatingStateType = enum {
    Off,    // 機能が無効化されており，作動していない状態
    Standby,    // 機能が有効であり，作動の条件が整えば作動できるが，現在は作動していない状態
    Activating, // 作動開始の要求を受理し，作動開始の処理を行っている過渡状態
    Active, // 作動中。車両の運動への介入または警報の出力を現に行っている状態
    Suspended,  // 作動が一時的に解除され，中断している状態。直前の作動時の設定値を保持しており，抑止する要因が解消すれば作動を再開できる
```

```
    Deactivating,   // 作動終了の要求を受理し，作動終了の処理を行っている過渡状態
    Fallback,   // 縮退運転を実行している状態
    MRC,    // 最小リスク状態へ移行し，これを維持している状態
    Fault   // 故障により機能を提供できない状態
};
```

各機能オブジェクトが現にどの作動状態にあるかを表す型である。機能によっては現れない列挙子がある。

```
type AutomationAvailabilityType = enum {
    Available,  // 作動を開始または継続できる
    TemporarilyUnavailable, // 一時的に条件を満たさないため，作動を開始または継続できない（条件が回復すれば作動できる）
    Unavailable // 故障等により，作動を開始または継続できない
};
```

その機能がいま作動を開始または継続できるかを表す型である。作動しているか否かは作動状態operatingStateが表すものであり，本型はこれを表さない。

```
type AutomationUnavailableReasonType = enum {
    ODDOutside, // 走行環境が運行設計領域の外にある
    SensorDegraded, // センサの性能低下・遮蔽により，必要な認識の性能が得られない
    DependentFunctionFault, // この機能が用いている他のAD/ADAS機能が故障している（blockingFunctionがその機能を指す）
    DriverMonitoringUnavailable,    // 運転者の監視が行えない
    VehicleStateNotReady,   // 車速・変速機の位置・制動等の車両の状態が作動の条件を満たさない
    FunctionFault,  // この機能自体が故障している
    ControlOverridden,  // 運転者の操作または他の経路による車両の運動の制御により，作動が抑止されている
    UserDisabled,   // 利用者の設定により機能が無効にされている
    SupersededByOtherFunction,  // 他のAD/ADAS機能がその運転操作を担っているため作動しない（blockingFunctionがその機能を指す）
    CalibrationRequired,    // 較正が必要で作動できない
    Other   // その他の理由
};
```

availabilityがAvailable以外の値である場合に，作動を開始または継続できない理由を表す型である。機能によっては現れない列挙子がある。

```
type AutomationOutputLevelType = enum {
    Detected,   // 事象を検知したが，運転者への提示はまだ行っていない
    Advisory,   // 注意喚起（表示等の弱い提示）
    Warning,    // 警報（音・振動を伴う提示）
    Critical    // 緊急（車両運動への介入を伴う）
};
```

機能の出力の段階を表す型である。Detected，Advisory，Warning，Criticalの順に強い段階を表す。機

能によっては現れない段階がある。

```
type AutomationFunctionOutputType = struct {
    active : Bool,  // その機能が事象の検知または運転者への提示を現に行っているか
    level : AutomationOutputLevelType?  // 出力の段階。activeがtrueであり，かつ段階を
提供できる機能である場合にのみ値を持つ
};
```

機能が外部（運転者）に対して現に行っている出力を，有無と段階のみで表す型である。出力の判断の根拠（検知した車線の形状・物体の種別等）は本型に含まない。

5.9.8.7. AD/ADAS機能共通の故障の区分

以下のデータ型は，AD/ADAS機能オブジェクトが共通に用いるデータ型であり，本仕様内に定義する。BSMは車両の運動に介入しないため，アクチュエータの故障（ActuatorFault）は現れない。

```
type AutomationFaultCategoryType = enum {
    SensorFault,    // センサの故障
    ActuatorFault,  // アクチュエータの故障
    ComputationFault,   // 演算装置の故障
    CommunicationFault, // 車内通信の故障
    PowerFault, // 電源の異常
    WarningDeviceFault, // 警報を提示する装置の故障
    Other   // その他の故障
};
```

機能を提供できない故障の区分を表す型である。機能によっては現れない列挙子がある。故障の区分は，各機能オブジェクトの状態faultCategoryおよびFaultDetectedイベントの関連情報により渡される。

5.9.8.8. BSMの操作

```
type BSMOperationType = enum {
    Activate,   // 作動の開始の要求（サービスコールactivate）
    Deactivate  // 作動の終了の要求（サービスコールdeactivate）
};
```

アプリケーションから要求できる操作を表す型である。列挙子は，操作を行うサービスコールと1対1に対応する。すべてのアプリケーションが呼び出すことができるサービスコール（getConfig，getStatus，notify，unnotify，getEvent，lock，unlock）は，本型の対象としない。

5.9.8.9. 警報の対象となる車速の範囲

```
type BSMSpeedRangeType = struct {
    minimum : SpeedType,    // 警報の対象となる車速の下限。単位はkm/h
    maximum : SpeedType?    // 警報の対象となる車速の上限。単位はkm/h。上限を設けない
場合は値を持たない
};
```

5.9.8.10. BSMの構成情報（getConfigが返すデータ型）

```
type BSMConfigType = struct {
    capabilities : enumSet(BSMCapabilityType),
    riskClasses : enumSet(BSMRiskType),
    supportedOperations : enumSet(BSMOperationType),
    displayName : String?,
    warningSpeedRange : BSMSpeedRangeType,
    maximumWarningLevel : AutomationOutputLevelType?,
    timestampAccuracy : UInt32
};
```

5.9.8.11. BSM状態

```
type BSMStatusType = struct {
    timestamp : TimestampUsType,
    dataQualifier : DataQualifierType,
    operatingState : AutomationOperatingStateType,
    availability : AutomationAvailabilityType,
    unavailableReason : AutomationUnavailableReasonType?,
    blockingFunction : RelatedObjectType[],
    faultCategory : AutomationFaultCategoryType?,
    functionOutput : AutomationFunctionOutputType,
    leftOutput : AutomationFunctionOutputType,
    rightOutput : AutomationFunctionOutputType,
    lockStatus : LockStatusType
};
```

呼び出し時点のBSMの状態の冪等なスナップショットである。

5.9.8.12. BSMイベント

```
type BSMEventKindType = enum {
    OperatingStateChanged,
    AvailabilityChanged,
    FunctionOutputChanged,
    FaultDetected,
    FaultRecovered,
    DataQualifierChanged,
    OwnLockRevoked,
    OtherLockReleased,
    EventOverflow,
    ConfigChanged
};
```

```
type BSMEventType = struct {
    eventInfo : variant(BSMEventKindType) {
```

```
        FaultDetected {
            faultCategory : AutomationFaultCategoryType
        },
        EventOverflow {
            noLostEvents : UInt32
        }
    }
};
```

5.9.9. サービスコールの詳細規定

getConfig

BSMの構成情報の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
config	BSMConfigType	BSMの構成情報

【エラーコード】

なし

【機能】

その車両のBSMが持つ機能，アプリケーションから要求できる操作の集合，警報の対象となる車速の範囲，およびこの機能が出しうる最も強い段階を返す。

アプリケーションは，本サービスコールにより，画面を構成する前に，どの操作の要素を提示すべきかを決める。

getStatus

BSMの状態の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
status	BSMStatusType	BSMの状態

【エラーコード】

E_OBJECT_FAULT	本オブジェクトが故障しており，状態を提供できない
E_RISK_APPLICATION	アプリケーションがRiskPrivacyに対応していない
E_RISK_USER	ユーザがRiskPrivacyを承認していない

【機能】

BSMの現在の作動状態・利用可否・故障の区分・警報の出力の状況（左右の側ごとの状況を含む）・ロック状態を返す。

呼び出しのたびに，その時点の状態の全体を返す冪等な応答である。値を読む前にdataQualifierを確認する。

activate

BSMの作動の開始の要求

【パラメータ】

priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_PARAMETER	priorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_LOCKED	本オブジェクトが，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている
E_OBJECT_STATUS	作動を開始できる作動状態でない（既に有効である，またはavailabilityがUnavailableであり，その理由が故障以外である）
E_OBJECT_FAULT	本オブジェクトが故障しており，要求を受け付けられない（unavailableReasonがFunctionFaultである場合を含む）
E_INFEASIBLE	availabilityがTemporarilyUnavailableであり，作動の条件以外の一時的な理由（故障等）により，現在はBSMを有効にできない
E_RISK_APPLICATION	アプリケーションがRiskCollisionに対応していない
E_RISK_USER	ユーザがRiskCollisionを承認していない

【機能】

BSMに作動の開始を要求する。本サービスコールは，BSMを有効にする操作に相当し，Offからの遷移を要求する。作動の条件（車速が警報の対象となる範囲にある等）が整っている場合はActiveへ，整っていない場合はStandbyへ遷移する。

正常終了は要求が受理されたことのみを意味し，作動が開始したことを意味しない。遷移の完了は，

getStatusまたはOperatingStateChangedイベントにより確認する。

作動を開始できない場合の応答は，利用可否availabilityとその理由unavailableReasonにより定まる。availabilityがTemporarilyUnavailableである場合はE_INFEASIBLEエラー，Unavailableでありunavailable ReasonがFunctionFaultである場合はE_OBJECT_FAULTエラー，Unavailableでありそれ以外の理由である場合はE_OBJECT_STATUSエラーとなる。

本サービスコールは，構成情報supportedOperationsにActivateを含む場合にのみ用いることができる。

deactivate

BSMの作動の終了の要求

【パラメータ】

priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_PARAMETER	priorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_LOCKED	本オブジェクトが，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている
E_OBJECT_STATUS	作動を終了できる作動状態でない（既にOffである等）
E_OBJECT_FAULT	本オブジェクトが故障しており，要求を受け付けられない
E_RISK_APPLICATION	アプリケーションがRiskCollisionに対応していない
E_RISK_USER	ユーザがRiskCollisionを承認していない

【機能】

BSMに作動の終了を要求する。本サービスコールは，BSMを無効にする操作に相当し，Offへの遷移を要求する。

正常終了は要求が受理されたことのみを意味し，作動が終了したことを意味しない。遷移の完了は，getStatusまたはOperatingStateChangedイベントにより確認する。

本サービスコールは，構成情報supportedOperationsにDeactivateを含む場合にのみ用いることができる。



本サービスコールによりBSMが無効になると，死角にいる車両の警報は出力されなくなる。運転者がその車両に気付かないまま車線を変更することにつながるため，アプリケーションは，無効にしたことを運転者が認識できるようにしなければならない。

lock

BSMのロックの取得

【パラメータ】

priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_PARAMETER	priorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_LOCKED	本オブジェクトが，priorityと同じかそれより高い優先度でロックされている
E_OBJECT_FAULT	本オブジェクトが故障しており，ロックを取得できない

【機能】

本オブジェクトを，指定した操作優先度でロックする。ロックを取得したアプリケーションは，状態を変化させるサービスコール（activate，deactivate）を，同じかそれより低い優先度の他のアプリケーションによる操作から保護する。

既存のロックより高い優先度でロックを取得した場合，既存のロックは強制解除され，そのロックを取得していたアプリケーションにはOwnLockRevokedイベントが通知される。

運転者による直接の操作は，ロックおよび操作優先度によって妨げられない。

unlock

BSMのロックの解除

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_OBJECT_STATUS	自アプリケーションが，本オブジェクトをロックしていない

【機能】

自アプリケーションが取得した本オブジェクトのロックを解除する。

notify

BSMイベントの通知要求

【パラメータ】

eventFilter	enumSet (BSMEventKindType)?	通知するイベントの種類の集合

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【エラーコード】

E_INVALID_PARAMETER	eventFilterが有効範囲外
E_RISK_APPLICATION	アプリケーションがRiskPrivacyに対応していない
E_RISK_USER	ユーザがRiskPrivacyを承認していない

【機能】

BSMで発生したeventFilterで指定したイベントの通知を要求する。イベントを通知するためのイベントキューを生成し，その通知ハンドルをnotifyHandleに返す。

eventFilterを省略した場合，すべての種類のイベントの通知を要求する。

警報の発生の観測からも運転の傾向を推定できるため，本サービスコールもRiskPrivacyのリスク制御の対象である。

unnotify

BSMイベントの通知停止

【パラメータ】

notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正

【機能】

notifyHandleで指定した通知を停止する。指定した通知用のイベントキューは削除される。

getEvent

BSMイベントの取得

【パラメータ】

notifyHandle	NotifyHandleType	イベントキューを識別するための通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
event	BSMEventType	取得したイベント

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正
E_NO_DATA	イベントキューが空
E_RISK_APPLICATION	アプリケーションがRiskPrivacyに対応していない
E_RISK_USER	ユーザがRiskPrivacyを承認していない

【機能】

notifyHandleで指定した通知用のイベントキューから，先頭のイベントを取り出す。取り出したイベントは，イベントキューから削除される。

イベントの内容からも運転の傾向を推定できるため，本サービスコールもRiskPrivacyのリスク制御の対象である。



イベントがパラメータ等で渡されてくるプログラミング環境向けの物理APIでは，用意する必要がない。

5.10. Vehicle.ADADAS.FCW（略称：FCW）

5.10.1. 位置付けと機能

Vehicle.ADADAS.FCW（略称：FCW）は，車両に搭載された前方衝突警報（FCW）を参照・操作するためのシングルインスタンスオブジェクトである。作動の開始・終了の要求，警報のタイミングの設定，および現在の作動状態・警報のタイミング・警報の出力の状況の参照を提供する。

本オブジェクトは，Vehicle.ADADASの目録に，機能の種別FCW，提供するオブジェクトの完全修飾名Vehicle.ADADAS.FCWとして現れる。DDTの責任主体・運行設計領域の充足状態・縮退運転・最小リスク状態など，AD/ADASシステム全体について一つに定まる状態は，本オブジェクトではなくVehicle.ADADASが提供する。

5.10.1.1. FCWの位置付けと性質

FCWは，外界のセンサ（レーダ・カメラ等）により前方の車両・歩行者等を検知し，それらとの衝突の危険があると判定した場合に，運転者に警報を出力する運転支援の機能である。FCWは車両の運動に介入しない。警報のみを行い，制動・操舵の介入は行わないため，FCWの作動中も車両の運動の制御は運転者が行う。

前方の物体との衝突の危険に対して制動により介入する機能は，衝突被害軽減ブレーキ（AEB）であり，Vehicle.ADADAS.AEBが提供する。FCWとAEBは，同じ危険に対して異なる段階の対応を担う別の機能である。衝突の危険が高まる過程では，まずFCWが警報により運転者に回避の操作を促し，運転者が回避の操作を行わないまま危険がさらに高まった場合に，AEBが制動により介入する。FCWとAEBは，それぞれ独立したオブジェクトとして作動状態・利用可否・故障の区分を持つ。AEBの作動状態および介入の状況は，Vehicle.ADADAS.AEBにより参照する。

本オブジェクトは，FCWが用いるセンサの構成，前方の物体の検知と衝突の危険の判定のアルゴリズムを規定しない。検知した物体の種別・位置・相対速度，衝突までの予測時間といった判定の根拠は本オブジェクトでは扱わない。本オブジェクトが提供するのは，機能が作動しているか（operatingState），いま作動を継続できるか（availability），現に警報を出力しているか（functionOutput），警報のタイミング（warningTiming），および故障の区分（faultCategory）である。

5.10.1.2. 作動の開始・終了の操作

本オブジェクトは，作動の開始（activate）と終了（deactivate）をアプリケーションから要求できる。車両が備える設定の手段（車載の設定画面等）により運転者がFCWを有効化・無効化した場合も，同じ作動状態operatingStateに反映される。

アプリケーションが行える操作は，構成情報supportedOperationsが宣言する。supportedOperationsの列挙子は，操作を行うサービスコールと1対1に対応する。アプリケーションは，画面を構成する前にsupportedOperationsを参照し，宣言されていない操作の要素を提示してはならない。宣言されていない操作のサービスコールを呼び出した場合，E_NOT_SUPPORTEDエラーとなる。

supportedOperationsが操作の可否（恒久的な可否）を表すのに対し，availabilityは，いまその操作を受け付けられるか（一時的な可否）を表す。この2つは別の判定であり，アプリケーションは，前者により要素を提示するか否かを，後者により要素を操作可能とするか否かを決める。

警報を出力しないようにすることは，前方の衝突の危険に運転者が気付かない状態を生じうる。作動の終了の要求に伴うリスクは，操作・参照に伴うリスクの節に規定する。

5.10.1.3. 作動状態

本オブジェクトの作動状態operatingStateおよび利用可否availabilityは，Vehicle.ADADASが規定するAD/ADAS機能共通の作動状態モデルに従う。本節では，FCWに固有の制約のみを規定する。

FCWにおける各作動状態の意味は，次のとおりである。

- Off … FCWが無効化されており，前方の衝突の危険を監視していない状態。アプリケーションによる作動の終了の要求，および運転者の設定のいずれによっても，この状態となる。
- Standby … FCWが有効であるが，車速が警報の対象となる範囲の外にある等の理由で，前方の衝突の危険を監視していない状態。
- Activating … 作動開始の要求（activate）を受理し，遷移の処理を行っている過渡状態。
- Active … FCWが有効であり，前方の衝突の危険を監視している状態。この状態において，衝突の危険を検知すると出力が生じ，functionOutputのactiveがtrueとなる。
- Deactivating … 作動終了の要求（deactivate）を受理し，Offへ遷移する処理を行っている過渡状態。
- Suspended … 運転操作の全体を担う他のAD/ADAS機能が作動しているため，前方の衝突の危険の監視と警報を行わない状態。この場合，availabilityはTemporarilyUnavailable，unavailableReasonはSupersededByOtherFunctionとなり，blockingFunctionがその機能を指す。

- Fault ... 故障により機能を提供できない状態。故障の区分は状態faultCategoryが表す。

FCWでは，FallbackおよびMRCの列挙子は現れない。縮退運転および最小リスク状態への移行は，DDTの責任を持つ機能が行うものであり，警報のみを行うFCWはこれを行わないためである。

FCWが有効である間に車速が警報の対象となる範囲を外れた場合は，Suspendedではなく，Standbyかつavailabilityが TemporarilyUnavailableで表す。

operatingStateがOffである間，availabilityは，車速によらず，FCWを有効にできるか（故障の有無等）のみを表す。車速の範囲等の，有効化した後にのみ意味を持つ作動の条件はavailabilityに反映せず，車速が警報の対象となる範囲の外にあることのみを理由として，availabilityがTemporarilyUnavailableとなることはない。アプリケーションは，車速によらず作動の開始（activate）を要求でき，受理された場合，作動の条件が整っていなければoperatingStateはStandbyへ遷移する。作動の条件を満たさないことは，Standbyへ遷移した後のavailabilityおよびunavailableReasonにより表す。

運転者の設定によりFCWが無効化されている場合も，operatingStateはOffとなる。このとき，FCWを有効にできればavailabilityはAvailableであり，アプリケーションは作動の開始（activate）を要求できる。この要求は，後勝ちの原則に従い，運転者の設定より後に行われた操作として受理される。アプリケーションは，作動状態の変化をOperatingStateChangedイベント，利用可否の変化をAvailabilityChangedイベントにより知ることができる。

車両が，アプリケーションからの作動の開始・終了そのものを認めない場合は，構成情報supportedOperationsにActivateおよびDeactivateを含めないことで表す。

警報を出力しているか否かは，作動状態ではなくfunctionOutputで表す。作動状態Activeは「監視しており，いつでも警報を出力できる状態」を表し，警報の出力そのものは示さない。

5.10.1.4. 警報の出力

警報の出力の状況は，状態functionOutputが表す。activeは，前方の衝突の危険を検知しているか，または運転者への提示を現に行っているかを表す。levelは出力の段階を表し，機能クラスWarningLevelReportedを備えている場合にのみ値を持つ。

FCWにおける各段階の意味は，次のとおりである。

- Detected ... 前方の物体との衝突の危険を検知したが，まだ運転者への提示を行っていない状態。
- Advisory ... 前方の物体への接近を，表示等の弱い提示により運転者に知らせている状態（接近の注意）。
- Warning ... 衝突の危険が高く，直ちに回避の操作が必要であることを，音・振動を伴う提示により運転者に知らせている状態（衝突の警報）。

FCWは車両の運動に介入しないため，Criticalは現れない。車両によっては，Advisoryを経ずにWarningとなることがある。その機能が出しうる最も強い段階は，構成情報maximumWarningLevelが宣言する。

警報の提示の方法（表示・音・触覚）は，機能クラスVisualWarning，AudibleWarning，HapticWarningが宣言する。どの方法で提示するかは車両実装に依存し，警報1回ごとの提示の方法は本オブジェクトでは表さない。

functionOutputの値が変化したときは，FunctionOutputChangedイベントが発生する。衝突の危険の検知（Detected）の開始，提示の段階の変化（DetectedからAdvisory，AdvisoryからWarning等），および出力の終了のいずれでも発生する。イベントは事象の発生のみを伝え，値そのものは運ばない。値は，

イベントを受けたアプリケーションがgetStatusにより取得する。



衝突の危険の対象となった物体の種別・位置・相対速度，および衝突までの予測時間は本オブジェクトでは提供しない。前方の物体の情報はVehicle.SurroundModelの担当である。

5.10.1.5. 警報のタイミング

警報のタイミングは，衝突の危険を運転者に提示し始める時機の早さを表す。Normalは車両の標準の時機を，Earlyはそれより早い時機（衝突までの余裕が大きい時点）を，Lateはそれより遅い時機を表す。各タイミングに対応する判定の基準は車両実装に依存する。

機能クラスWarningTimingSelectableを備える車両では，警報のタイミングを選択でき，現在のタイミングを状態warningTimingが表す。選択できるタイミングの集合は，構成情報selectableWarningTimingsが宣言する。機能クラスWarningTimingSelectableを備えない車両では，警報のタイミングは車両が定める一つの時機に固定されており，warningTimingは値を持たず，selectableWarningTimingsは空集合となる。

機能クラスWarningTimingSelectableは，警報のタイミングを選択する手段を車両が備えることを表し，アプリケーションからの設定の可否は表さない。アプリケーションからの設定の可否は，構成情報supportedOperationsのSetWarningTimingが表す。supportedOperationsがSetWarningTimingを含むのは，機能クラスWarningTimingSelectableを備える場合に限る。運転者が車載の設定画面等によりタイミングを変更した場合も，同じ状態warningTimingに反映される。

警報のタイミングは，作動状態によらず保持される。FCWが無効化されている間も，次に作動したときに用いるタイミングとして設定できる。

5.10.1.6. FCWに対する制御の調停

本オブジェクトに対する制御の調停には，標準的な調停機能（後勝ちの原則）が適用される。すなわち，あるアプリケーションが作動を開始した後に，別のアプリケーションが作動の終了を要求した場合，後の要求が有効となる。

本オブジェクトは，標準的なロック機能を持つLockableObjectである。状態を変化させるサービスコール（activate，deactivate，setWarningTiming）およびロックを取得するサービスコール（lock）は，操作優先度priorityを持つ。ロック中の本オブジェクトに対して，ロックと同じかそれより低い優先度で操作した場合はE_OBJECT_LOCKEDエラーとなり，より高い優先度で操作した場合は，既存のロックが強制解除されて操作が実行される。アプリケーションが使用できる最高値より高い優先度を指定した場合はE_DENIED_PRIORITYエラー，操作優先度が有効範囲外である場合はE_INVALID_PARAMETERエラーとなる。

自アプリケーションのロックが強制解除された場合はOwnLockRevokedイベントが，他のアプリケーション等によりロックが解除された場合はOtherLockReleasedイベントが発生する。作動状態がFaultとなった場合，ロックは解除され，その間は新たにロックを取得することができない。

運転者による直接の操作（車載の設定画面等による有効化・無効化，警報のタイミングの変更）は，本APIを用いない一つのアプリケーションによる操作と見なし，後勝ちの原則に従う。この操作は，アプリケーションが取得したロックおよび操作優先度によって妨げられることはなく，常に有効となる。運転の責任を負うのは運転者であり，アプリケーションによる排他が運転者の操作に優先してはならないためである。

5.10.1.7. 操作・参照に伴うリスク

FCWの作動を終了させると，前方の衝突の危険の警報が出力されなくなり，運転者が危険に気付かないまま走行することにつながりうる。警報のタイミングを遅くすることも，運転者が回避の操作に使える時間

を短くする。これをリスククラスRiskCollisionとして定義し，activate，deactivateおよびsetWarningTimingをリスク制御の対象とする。作動の開始も，警報の出力により運転者の注意を引くものであり，同じリスククラスの対象とする。

FCWの作動状態および警報の出力の履歴は，運転者がいつどのように運転していたか（前方の車両との車間の詰め方等）を第三者が推定できる情報である。これをリスククラスRiskPrivacyとして定義し，状態を参照するサービスコール（getStatus），イベントの通知要求（notify）およびイベントの取得（getEvent）をリスク制御の対象とする。

いずれのリスククラスについても，当該リスクに対応していないアプリケーションからの呼び出しはE_RISK_APPLICATIONエラー，ユーザが当該リスクを承認していない場合はE_RISK_USERエラーとなる。

構成情報の参照（getConfig）は，車両に搭載されたFCWの静的な性質を返すものであり，リスク制御の対象としない。

5.10.2. 機能クラス

FCWは，以下の機能クラスを持つ。

WarningLevelReported	出力の段階（検知，接近の注意，衝突の警報）を提供することができる（状態functionOutputのlevel）
VisualWarning	表示による警報を提示することができる
AudibleWarning	音による警報を提示することができる
HapticWarning	触覚（ステアリング，座席またはペダルの振動等）による警報を提示することができる
WarningTimingSelectable	警報のタイミングを選択することができる（状態warningTiming，構成情報selectableWarningTimings）

5.10.3. リスククラス

FCWは，以下のリスククラスを持つ。

RiskCollision	FCWの作動の開始・終了および警報のタイミングの変更により警報の出力の有無・時機が変わり，運転者が前方の衝突の危険に気付かないまま，または気付くのが遅れて衝突を生じるリスク
RiskPrivacy	走行状態および運転者の運転への関与状態の開示により，個人の行動を推定できることに伴うプライバシのリスク

5.10.4. 構成情報

FCWは，以下の構成情報を持つ。

capabilities	enumSet (FCWCapabilityType)	その車両のFCWが持つ機能
riskClasses	enumSet(FCWRiskType)	車両およびビークルOSで対応できていないリスククラスの集合
supportedOperations	enumSet (FCWOperationType)	アプリケーションから要求できる操作の集合。空集合となることがある

displayName	String?	ユーザに表示する呼称を取得するためのキー。車両が呼称を公開する場合にのみ値を持つ
warningSpeedRange	FCWSpeedRangeType	警報の対象となる車速の範囲。FCWが有効である間，この範囲の外では，作動状態はStandbyとなる
maximumWarningLevel	AutomationOutputLevelType?	この機能が出しうる最も強い段階。機能クラスWarningLevelReportedを備えている場合にのみ値を持つ
selectableWarningTimings	enumSet (FCWWarningTimingType)	選択できる警報のタイミングの集合。機能クラスWarningTimingSelectableを備えていない場合は空集合となる
timestampAccuracy	UInt32	この車両のタイムスタンプ誤差上限の宣言。単位はμs

上記は，YAML形式で記述した構成記述ファイル中に記述する形式である。本オブジェクトはシングルインスタンスオブジェクトであるため，構成記述では，車両を表すノードの下に置かれたADADASノードの下に，オブジェクトを表すノードFCWを置き，その下に上記の構成情報を記述する。このノードは，Vehicle.ADADASの目録において，providingObjectがVehicle.ADADAS.FCWである要素に対応する。

riskClassesには，車両およびビークルOSで対応できていない（危険を防止できていない）リスククラスの集合を記述する。

構成情報を参照するサービスコールは，この構成情報に対して，次の変換を行った構成情報を返す。

- displayNameに対しては，それを元に現在のロケールに対応する呼称の文字列を取得し，その文字列を返す。

5.10.5. 状態

FCWは，以下の状態を持つ。

timestamp	TimestampUsType	状態の測定時刻。単位はμs。車内共通時刻基盤に基づく
dataQualifier	DataQualifierType	このサンプルの品質。InvalidまたはUnavailableの場合，以下の状態の値を読んではならない。値の変化はDataQualifierChangedイベントにより通知される
operatingState	AutomationOperatingStateType	現在の作動状態。FCWではFallbackおよびMRCは現れない
availability	AutomationAvailabilityType	いま前方の衝突の危険の監視を開始または継続できるか
unavailableReason	AutomationUnavailableReasonType?	監視を開始または継続できない理由。availabilityがAvailable以外の値である場合にのみ値を持つ
blockingFunction	RelatedObjectType[]	監視を妨げている機能オブジェクト。unavailableReasonがDependentFunctionFaultまたはSupersededByOtherFunctionである場合にのみ要素を持つ

faultCategory	AutomationFaultCategoryType?	故障の区分。operatingStateがFaultである場合にのみ値を持つ
functionOutput	AutomationFunctionOutputType	警報の出力の状況。levelは機能クラスWarningLevelReportedを備えている場合にのみ値を持つ
warningTiming	FCWWarningTimingType?	現在の警報のタイミング。機能クラスWarningTimingSelectableを備えている場合にのみ値を持つ
lockStatus	LockStatusType	ロック状態

状態は，呼び出し時点のFCWの状態を返す冪等なスナップショットである。値を読む前にdataQualifierを確認する。

FCWが警報を出せる状態にあるかは，operatingStateにより判定する。いま前方の衝突の危険を検知しているか，警報を出しているかは，functionOutputにより判定する。operatingStateがActive以外である間，functionOutputのactiveは常にfalseである。

operatingStateとavailabilityの組合せの規則は，Vehicle.ADADASが規定するAD/ADAS機能共通の作動状態モデルによる。FCWが有効である間に車速が構成情報warningSpeedRangeの範囲の外にある場合，availabilityはTemporarilyUnavailable，unavailableReasonはVehicleStateNotReadyとなり，operatingStateはStandbyとなる。operatingStateがOffである間のavailabilityは，車速を反映しない。

故障により機能を提供できない場合，operatingStateはFaultとなり，faultCategoryがその区分を表す。これにより，FaultDetectedイベントを受け取っていないアプリケーションも，状態の参照のみで故障の区分を知ることができる。本オブジェクト自体が状態を提供できない場合は，getStatusがE_OBJECT_FAULTエラーを返す。

5.10.6. サービスコール一覧

FCWに対するサービスコールは以下の通り。

getConfig	FCWの構成情報の参照
getStatus	FCWの状態の参照
activate	FCWの作動の開始の要求
deactivate	FCWの作動の終了の要求
setWarningTiming	FCWの警報のタイミングの設定
lock	FCWのロックの取得
unlock	FCWのロックの解除
notify	FCWイベントの通知要求
unnotify	FCWイベントの通知停止
getEvent	FCWイベントの取得

5.10.7. イベント

イベントは，事象が発生したことのみを伝え，状態の値そのものを運ばない。ただし，事象の発生時点の

情報として有用なもの（故障の区分等）は，イベントの関連情報としても渡す。アプリケーションは，イベントを受けた後，必要に応じてgetStatusにより値を取得する。

状態の取得は，要求応答（getStatus）と，各状態の変化を伝えるイベントによる。

FCWに対するイベント種別は以下の通り。

OperatingStateChanged	作動状態の変化（アプリケーションの要求および運転者の設定による有効化・無効化を含む）
AvailabilityChanged	前方の衝突の危険の監視を開始または継続できるかの変化（監視を妨げている理由の変化を含む）
SettingChanged	警報のタイミングの変化（アプリケーションの要求および運転者の設定による変更を含む）
FunctionOutputChanged	前方の衝突の危険の警報の出力の状況（状態functionOutput）の変化。出力の開始・段階の変化・終了のいずれでも発生する
FaultDetected	故障の検知（FCWの機能を提供できない状態）
FaultRecovered	故障からの回復
DataQualifierChanged	状態の品質（状態dataQualifier）の変化
OwnLockRevoked	自アプリケーションのロックの強制解除
OtherLockReleased	他のアプリケーション等によるロック解除
EventOverflow	イベントキューのオーバフロー
ConfigChanged	構成情報の変更（ソフトウェアの更新による対応する操作・警報の対象となる車速の範囲・出力の段階・選択できる警報のタイミングの変更）

この内，イベント種別がFaultDetectedの場合には，以下の関連情報を持つ。

faultCategory	AutomationFaultCategoryType	故障の区分

また，イベント種別がEventOverflowの場合には，以下の関連情報を持つ。

noLostEvents	UInt32	失われたイベントの数

5.10.8. データ型の定義

5.10.8.1. FCWの機能クラス

```
type FCWCapabilityType = enum {
    WarningLevelReported,
    VisualWarning,
    AudibleWarning,
    HapticWarning,
    WarningTimingSelectable
};
```

5.10.8.2. FCWのリスククラス

```
type FCWRiskType = enum {
    RiskCollision,
    RiskPrivacy
};
```

5.10.8.3. タイムスタンプ

```
type TimestampUsType = UInt64;  // 車内共通時刻基盤の起点を基準とする測定時刻。単位はμs。複数のビークルAPIで共通に用いる共通データ型であり，本仕様内に定義する。
```

5.10.8.4. サンプル品質の種別

```
type DataQualifierType = enum {
    Normal, // 正常
    Unavailable,    // 利用不可
    Degraded,   // 劣化
    TestMode,   // テストモード
    Invalid // 無効（値を読んではならない）
};
```

複数のビークルAPIで共通に用いる共通データ型であり，本仕様内に定義する。

5.10.8.5. 速度

```
type SpeedType = Float32;   // 速度。単位はkm/h。正の値は前進，負の値は後退を示す。複数のビークルAPIで共通に用いる共通データ型であり，本仕様内に定義する。
```

5.10.8.6. AD/ADAS機能共通の作動状態

以下の5つのデータ型は，AD/ADAS機能オブジェクトが共通に用いるデータ型であり，本仕様内に定義する。FCWにおける各列挙子の具体的な意味は，位置付けと機能の節に示す。

```
type AutomationOperatingStateType = enum {
    Off,    // 機能が無効化されており，作動していない状態
    Standby,    // 機能が有効であり，作動の条件が整えば作動できるが，現在は作動していない状態
    Activating, // 作動開始の要求を受理し，作動開始の処理を行っている過渡状態
    Active, // 作動中。車両の運動への介入または警報の出力を現に行っている状態
    Suspended,  // 作動が一時的に解除され，中断している状態。直前の作動時の設定値を保持しており，抑止する要因が解消すれば作動を再開できる
    Deactivating,   // 作動終了の要求を受理し，作動終了の処理を行っている過渡状態
    Fallback,   // 縮退運転を実行している状態
    MRC,    // 最小リスク状態へ移行し，これを維持している状態
```

```
    Fault   // 故障により機能を提供できない状態
};
```

各機能オブジェクトが現にどの作動状態にあるかを表す型である。機能によっては現れない列挙子がある。

```
type AutomationAvailabilityType = enum {
    Available,  // 作動を開始または継続できる
    TemporarilyUnavailable, // 一時的に条件を満たさないため，作動を開始または継続できない（条件が回復すれば作動できる）
    Unavailable // 故障等により，作動を開始または継続できない
};
```

その機能がいま作動を開始または継続できるかを表す型である。作動しているか否かは作動状態operatingStateが表すものであり，本型はこれを表さない。

```
type AutomationUnavailableReasonType = enum {
    ODDOutside, // 走行環境が運行設計領域の外にある
    SensorDegraded, // センサの性能低下・遮蔽により，必要な認識の性能が得られない
    DependentFunctionFault, // この機能が用いている他のAD/ADAS機能が故障している（blockingFunctionがその機能を指す）
    DriverMonitoringUnavailable,    // 運転者の監視が行えない
    VehicleStateNotReady,   // 車速・変速機の位置・制動等の車両の状態が作動の条件を満たさない
    FunctionFault,  // この機能自体が故障している
    ControlOverridden,  // 運転者の操作または他の経路による車両の運動の制御により，作動が抑止されている
    UserDisabled,   // 利用者の設定により機能が無効にされている
    SupersededByOtherFunction,  // 他のAD/ADAS機能がその運転操作を担っているため作動しない（blockingFunctionがその機能を指す）
    CalibrationRequired,    // 較正が必要で作動できない
    Other   // その他の理由
};
```

availabilityがAvailable以外の値である場合に，作動を開始または継続できない理由を表す型である。機能によっては現れない列挙子がある。

```
type AutomationOutputLevelType = enum {
    Detected,   // 事象を検知したが，運転者への提示はまだ行っていない
    Advisory,   // 注意喚起（表示等の弱い提示）
    Warning,    // 警報（音・振動を伴う提示）
    Critical    // 緊急（車両運動への介入を伴う）
};
```

機能の出力の段階を表す型である。Detected，Advisory，Warning，Criticalの順に強い段階を表す。機能によっては現れない段階がある。

```
type AutomationFunctionOutputType = struct {
    active : Bool,  // その機能が事象の検知または運転者への提示を現に行っているか
    level : AutomationOutputLevelType?  // 出力の段階。activeがtrueであり，かつ段階を
提供できる機能である場合にのみ値を持つ
};
```

機能が外部（運転者）に対して現に行っている出力を，有無と段階のみで表す型である。出力の判断の根拠（検知した車線の形状・物体の種別等）は本型に含まない。

5.10.8.7. AD/ADAS機能共通の故障の区分

以下のデータ型は，AD/ADAS機能オブジェクトが共通に用いるデータ型であり，本仕様内に定義する。FCWは車両の運動に介入しないため，アクチュエータの故障（ActuatorFault）は現れない。

```
type AutomationFaultCategoryType = enum {
    SensorFault,    // センサの故障
    ActuatorFault,  // アクチュエータの故障
    ComputationFault,   // 演算装置の故障
    CommunicationFault, // 車内通信の故障
    PowerFault, // 電源の異常
    WarningDeviceFault, // 警報を提示する装置の故障
    Other   // その他の故障
};
```

機能を提供できない故障の区分を表す型である。機能によっては現れない列挙子がある。故障の区分は，各機能オブジェクトの状態faultCategoryおよびFaultDetectedイベントの関連情報により渡される。

5.10.8.8. FCWの操作

```
type FCWOperationType = enum {
    Activate,   // 作動の開始の要求（サービスコールactivate）
    Deactivate, // 作動の終了の要求（サービスコールdeactivate）
    SetWarningTiming    // 警報のタイミングの設定（サービスコールsetWarningTiming）
};
```

アプリケーションから要求できる操作を表す型である。列挙子は，操作を行うサービスコールと1対1に対応する。すべてのアプリケーションが呼び出すことができるサービスコール（getConfig，getStatus，notify，unnotify，getEvent，lock，unlock）は，本型の対象としない。

5.10.8.9. 警報のタイミング

```
type FCWWarningTimingType = enum {
    Early,  // 標準より早い時機（衝突までの余裕が大きい時点）に提示を始める
    Normal, // 車両の標準の時機に提示を始める
    Late    // 標準より遅い時機に提示を始める
};
```

衝突の危険を運転者に提示し始める時機の早さを表す型である。各タイミングに対応する判定の基準は車両実装に依存する。

5.10.8.10. 警報の対象となる車速の範囲

```
type FCWSpeedRangeType = struct {
    minimum : SpeedType,    // 警報の対象となる車速の下限。単位はkm/h
    maximum : SpeedType?    // 警報の対象となる車速の上限。単位はkm/h。上限を設けない場合は値を持たない
};
```

5.10.8.11. FCWの構成情報（getConfigが返すデータ型）

```
type FCWConfigType = struct {
    capabilities : enumSet(FCWCapabilityType),
    riskClasses : enumSet(FCWRiskType),
    supportedOperations : enumSet(FCWOperationType),
    displayName : String?,
    warningSpeedRange : FCWSpeedRangeType,
    maximumWarningLevel : AutomationOutputLevelType?,
    selectableWarningTimings : enumSet(FCWWarningTimingType),
    timestampAccuracy : UInt32
};
```

5.10.8.12. FCW状態

```
type FCWStatusType = struct {
    timestamp : TimestampUsType,
    dataQualifier : DataQualifierType,
    operatingState : AutomationOperatingStateType,
    availability : AutomationAvailabilityType,
    unavailableReason : AutomationUnavailableReasonType?,
    blockingFunction : RelatedObjectType[],
    faultCategory : AutomationFaultCategoryType?,
    functionOutput : AutomationFunctionOutputType,
    warningTiming : FCWWarningTimingType?,
    lockStatus : LockStatusType
};
```

呼び出し時点のFCWの状態の冪等なスナップショットである。

5.10.8.13. FCWイベント

```
type FCWEventKindType = enum {
    OperatingStateChanged,
    AvailabilityChanged,
    SettingChanged,
```

```
    FunctionOutputChanged,
    FaultDetected,
    FaultRecovered,
    DataQualifierChanged,
    OwnLockRevoked,
    OtherLockReleased,
    EventOverflow,
    ConfigChanged
};
```

```
type FCWEventType = struct {
    eventInfo : variant(FCWEventKindType) {
        FaultDetected {
            faultCategory : AutomationFaultCategoryType
        },
        EventOverflow {
            noLostEvents : UInt32
        }
    }
};
```

5.10.9. サービスコールの詳細規定

getConfig

FCWの構成情報の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
config	FCWConfigType	FCWの構成情報

【エラーコード】

なし

【機能】

その車両のFCWが持つ機能，アプリケーションから要求できる操作の集合，警報の対象となる車速の範囲，この機能が出しうる最も強い段階，および選択できる警報のタイミングの集合を返す。

アプリケーションは，本サービスコールにより，画面を構成する前に，どの操作の要素を提示すべきかと，提示する警報のタイミングの選択肢を決める。

getStatus

FCWの状態の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
status	FCWStatusType	FCWの状態

【エラーコード】

E_OBJECT_FAULT	本オブジェクトが故障しており，状態を提供できない
E_RISK_APPLICATION	アプリケーションがRiskPrivacyに対応していない
E_RISK_USER	ユーザがRiskPrivacyを承認していない

【機能】

FCWの現在の作動状態・利用可否・故障の区分・警報の出力の状況・警報のタイミング・ロック状態を返す。

呼び出しのたびに，その時点の状態の全体を返す冪等な応答である。値を読む前にdataQualifierを確認する。

activate

FCWの作動の開始の要求

【パラメータ】

priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_PARAMETER	priorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_LOCKED	本オブジェクトが，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている
E_OBJECT_STATUS	作動を開始できる作動状態でない（既に有効である，またはavailabilityがUnavailableであり，その理由が故障以外である）
E_OBJECT_FAULT	本オブジェクトが故障しており，要求を受け付けられない（unavailableReasonがFunctionFaultである場合を含む）

E_INFEASIBLE	availabilityがTemporarilyUnavailableであり，作動の条件以外の一時的な理由（故障等）により，現在はFCWを有効にできない
E_RISK_APPLICATION	アプリケーションがRiskCollisionに対応していない
E_RISK_USER	ユーザがRiskCollisionを承認していない

【機能】

FCWに作動の開始を要求する。本サービスコールは，FCWを有効にする操作に相当し，Offからの遷移を要求する。作動の条件（車速が警報の対象となる範囲にある等）が整っている場合はActiveへ，整っていない場合はStandbyへ遷移する。

正常終了は要求が受理されたことのみを意味し，作動が開始したことを意味しない。遷移の完了は，getStatusまたはOperatingStateChangedイベントにより確認する。

作動を開始できない場合の応答は，利用可否availabilityとその理由unavailableReasonにより定まる。availabilityがTemporarilyUnavailableである場合はE_INFEASIBLEエラー，Unavailableでありunavailable ReasonがFunctionFaultである場合はE_OBJECT_FAULTエラー，Unavailableでありそれ以外の理由である場合はE_OBJECT_STATUSエラーとなる。

本サービスコールは，構成情報supportedOperationsにActivateを含む場合にのみ用いることができる。

deactivate

FCWの作動の終了の要求

【パラメータ】

priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_PARAMETER	priorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_LOCKED	本オブジェクトが，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている
E_OBJECT_STATUS	作動を終了できる作動状態でない（既にOffである等）
E_OBJECT_FAULT	本オブジェクトが故障しており，要求を受け付けられない
E_RISK_APPLICATION	アプリケーションがRiskCollisionに対応していない
E_RISK_USER	ユーザがRiskCollisionを承認していない

【機能】

FCWに作動の終了を要求する。本サービスコールは，FCWを無効にする操作に相当し，Offへの遷移を要求する。

正常終了は要求が受理されたことのみを意味し，作動が終了したことを意味しない。遷移の完了は，getStatusまたはOperatingStateChangedイベントにより確認する。

本サービスコールは，構成情報supportedOperationsにDeactivateを含む場合にのみ用いることができる。



本サービスコールによりFCWが無効になると，前方の衝突の危険の警報は出力されなくなる。運転者が危険に気付かないまま走行することにつながるため，アプリケーションは，無効にしたことを運転者が認識できるようにしなければならない。

setWarningTiming

FCWの警報のタイミングの設定

【パラメータ】

warningTiming	FCWWarningTimingType	設定する警報のタイミング。構成情報selectableWarningTimingsに含まれる値でなければならない
priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_PARAMETER	warningTimingが構成情報selectableWarningTimingsに含まれない，またはpriorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_LOCKED	本オブジェクトが，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている
E_OBJECT_STATUS	警報のタイミングを設定できる作動状態でない（過渡状態（Activating，Deactivating）にある）
E_OBJECT_FAULT	本オブジェクトが故障しており，要求を受け付けられない
E_RISK_APPLICATION	アプリケーションがRiskCollisionに対応していない
E_RISK_USER	ユーザがRiskCollisionを承認していない

【機能】

FCWの警報のタイミングを設定する。設定は，作動状態がFaultでなく，かつ過渡状態にない間，Offである間を含めていつでも行うことができる。

正常終了は要求が受理されたことのみを意味する。設定値の変化はSettingChangedイベントにより通知される。

過渡状態（Activating，Deactivating）にある間は，E_OBJECT_STATUSエラーとなる。要求は留保されないため，アプリケーションは，OperatingStateChangedイベントにより遷移の完了を確認した上で，あらためて要求する。

本サービスコールは，構成情報supportedOperationsにSetWarningTimingを含む場合にのみ用いることができる。



警報のタイミングを遅くすると，運転者が衝突の危険に気付いてから回避の操作を行うまでの時間が短くなる。アプリケーションは，変更後のタイミングを運転者が認識できるようにしなければならない。

lock

FCWのロックの取得

【パラメータ】

priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_PARAMETER	priorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_LOCKED	本オブジェクトが，priorityと同じかそれより高い優先度でロックされている
E_OBJECT_FAULT	本オブジェクトが故障しており，ロックを取得できない

【機能】

本オブジェクトを，指定した操作優先度でロックする。ロックを取得したアプリケーションは，状態を変化させるサービスコール（activate，deactivate，setWarningTiming）を，同じかそれより低い優先度の他のアプリケーションによる操作から保護する。

既存のロックより高い優先度でロックを取得した場合，既存のロックは強制解除され，そのロックを取得していたアプリケーションにはOwnLockRevokedイベントが通知される。

運転者による直接の操作は，ロックおよび操作優先度によって妨げられない。

unlock

FCWのロックの解除

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_OBJECT_STATUS	自アプリケーションが，本オブジェクトをロックしていない

【機能】

自アプリケーションが取得した本オブジェクトのロックを解除する。

notify

FCWイベントの通知要求

【パラメータ】

eventFilter	enumSet (FCWEventKindType)?	通知するイベントの種類の集合

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【エラーコード】

E_INVALID_PARAMETER	eventFilterが有効範囲外
E_RISK_APPLICATION	アプリケーションがRiskPrivacyに対応していない
E_RISK_USER	ユーザがRiskPrivacyを承認していない

【機能】

FCWで発生したeventFilterで指定したイベントの通知を要求する。イベントを通知するためのイベントキューを生成し，その通知ハンドルをnotifyHandleに返す。

eventFilterを省略した場合，すべての種類のイベントの通知を要求する。

警報の発生の観測からも運転の傾向を推定できるため，本サービスコールもRiskPrivacyのリスク制御の対象である。

unnotify

FCWイベントの通知停止

【パラメータ】

notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正

【機能】

notifyHandleで指定した通知を停止する。指定した通知用のイベントキューは削除される。

getEvent

FCWイベントの取得

【パラメータ】

notifyHandle	NotifyHandleType	イベントキューを識別するための通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
event	FCWEventType	取得したイベント

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正
E_NO_DATA	イベントキューが空
E_RISK_APPLICATION	アプリケーションがRiskPrivacyに対応していない
E_RISK_USER	ユーザがRiskPrivacyを承認していない

【機能】

notifyHandleで指定した通知用のイベントキューから，先頭のイベントを取り出す。取り出したイベントは，イベントキューから削除される。

イベントの内容からも運転の傾向を推定できるため，本サービスコールもRiskPrivacyのリスク制御の対象である。



イベントがパラメータ等で渡されてくるプログラミング環境向けの物理APIでは，用意する必要がない。

5.11. Vehicle.ADADAS.TSR（略称：TSR）

5.11.1. 位置付けと機能

Vehicle.ADADAS.TSR（略称：TSR）は，車両に搭載された標識認識（TSR）を参照・操作するためのシングルインスタンスオブジェクトである。作動の開始・終了の要求と，現在の作動状態および認識している標識の参照を提供する。

本オブジェクトは，Vehicle.ADADASの目録に，機能の種別TSR，提供するオブジェクトの完全修飾名Vehicle.ADADAS.TSRとして現れる。DDTの責任主体・運行設計領域の充足状態・縮退運転・最小リスク状態など，AD/ADASシステム全体について一つに定まる状態は，本オブジェクトではなく

Vehicle.ADADASが提供する。

5.11.1.1. TSRの位置付けと性質

TSRは，車載のカメラにより道路標識（速度規制，追越し禁止，一時停止等）を認識し，その結果を運転者に提示する運転支援の機能である。TSRは車両の運動に介入しない。認識した標識の提示のみを行い，TSRの作動中も車両の運動の制御は運転者が行う。

TSRは警報の機能ではない。TSRは，認識した標識と自車の車速・挙動との比較（速度超過の判定等）を行わず，それに基づく警報も出力しない。

本オブジェクトは，TSRが用いるカメラの構成，標識の認識のアルゴリズムを規定しない。本オブジェクトが提供するのは，機能が作動しているか（operatingState），いま作動を継続できるか（availability），現在有効であると認識している標識（recognizedSigns），および故障の区分（faultCategory）である。



本オブジェクトが提供するのは，車載のカメラが現に認識している標識に限る。地図の情報に基づく道路の規制（制限速度等）はVehicle.Navigationが，道路の論理構造としての交通規制はVehicle.SurroundModel.LogicalStructureが提供する。

5.11.1.2. 作動の開始・終了の操作

本オブジェクトは，作動の開始（activate）と終了（deactivate）をアプリケーションから要求できる。車両が備える設定の手段（車載の設定画面等）により運転者がTSRを有効化・無効化した場合も，同じ作動状態operatingStateに反映される。

アプリケーションが行える操作は，構成情報supportedOperationsが宣言する。supportedOperationsの列挙子は，操作を行うサービスコールと1対1に対応する。アプリケーションは，画面を構成する前にsupportedOperationsを参照し，宣言されていない操作の要素を提示してはならない。宣言されていない操作のサービスコールを呼び出した場合，E_NOT_SUPPORTEDエラーとなる。

supportedOperationsが操作の可否（恒久的な可否）を表すのに対し，availabilityは，いまその操作を受け付けられるか（一時的な可否）を表す。この2つは別の判定であり，アプリケーションは，前者により要素を提示するか否かを，後者により要素を操作可能とするか否かを決める。

標識の提示を行わないようにすることは，運転者が標識を見落としたことに気付かない状態を生じうる。作動の終了の要求に伴うリスクは，操作・参照に伴うリスクの節に規定する。

5.11.1.3. 作動状態

本オブジェクトの作動状態operatingStateおよび利用可否availabilityは，Vehicle.ADADASが規定するAD/ADAS機能共通の作動状態モデルに従う。本節では，TSRに固有の制約のみを規定する。

TSRにおける各作動状態の意味は，次のとおりである。

- Off … TSRが無効化されており，標識を認識していない状態。アプリケーションによる作動の終了の要求，および運転者の設定のいずれによっても，この状態となる。
- Standby … TSRが有効であるが，車両の状態またはカメラの状態が認識の条件を満たさないため，標識を認識していない状態。
- Activating … 作動開始の要求（activate）を受理し，遷移の処理を行っている過渡状態。
- Active … TSRが有効であり，標識を認識している状態。

- Deactivating … 作動終了の要求（deactivate）を受理し，Offへ遷移する処理を行っている過渡状態。
- Suspended … 運転操作の全体を担う他のAD/ADAS機能が作動しているため，標識の認識と提示を行わない状態。この場合，availabilityはTemporarilyUnavailable，unavailableReasonはSupersededByOtherFunctionとなり，blockingFunctionがその機能を指す。
- Fault … 故障により機能を提供できない状態。故障の区分は状態faultCategoryが表す。

TSRでは，FallbackおよびMRCの列挙子は現れない。TSRは車両の運転操作を担わず，縮退運転および最小リスク状態への移行を行わないためである。

TSRが有効である間に認識の条件を満たさなくなった場合は，Suspendedではなく，Standbyかつavailabilityが TemporarilyUnavailableで表す。

operatingStateがOffである間，availabilityは，車両の状態によらず，TSRを有効にできるか（故障の有無等）のみを表す。カメラの遮蔽・汚れ等による一時的な性能の低下はOffである間のavailabilityに反映せず，作動の開始（activate）が受理されてStandbyへ遷移した後のunavailableReason（SensorDegraded）により表す。車速等の，有効化した後にのみ意味を持つ作動の条件はavailabilityに反映せず，車両の状態が認識の条件を満たさないことのみを理由として，availabilityがTemporarilyUnavailableとなることはない。アプリケーションは，車両の状態によらず作動の開始（activate）を要求でき，受理された場合，認識の条件が整っていなければoperatingStateはStandbyへ遷移する。認識の条件を満たさないことは，Standbyへ遷移した後のavailabilityおよびunavailableReasonにより表す。

運転者の設定によりTSRが無効化されている場合も，operatingStateはOffとなる。このとき，TSRを有効にできればavailabilityはAvailableであり，アプリケーションは作動の開始（activate）を要求できる。この要求は，後勝ちの原則に従い，運転者の設定より後に行われた操作として受理される。アプリケーションは，作動状態の変化をOperatingStateChangedイベント，利用可否の変化をAvailabilityChangedイベントにより知ることができる。

車両が，アプリケーションからの作動の開始・終了そのものを認めない場合は，構成情報supportedOperationsにActivateおよびDeactivateを含めないことで表す。

5.11.1.4. 認識している標識

TSRが現在有効であると認識している標識は，状態recognizedSignsが表す。recognizedSignsの各要素は，1つの標識について，その種別（kind），速度規制の値（speedLimit），認識の品質（quality），および最後に認識した時刻（recognizedAt）を持つ。自車に対して同時に有効な標識が複数ある場合（速度規制と追越し禁止等），recognizedSignsは複数の要素を持つ。同じ種別の要素は高々1つである。

標識は，通過した後もその効力が続く間は有効である。ある標識をいつrecognizedSignsに含め，いつ取り除くか（効力が続く区間の判定，補助標識が示す時間帯・車種等の条件の判定）は車両実装に依存し，本オブジェクトでは規定しない。補助標識が示す条件そのものは，本オブジェクトでは提供しない。

speedLimitは，標識が示す速度の値を単位km/hで表す。マイル毎時の標識を認識した場合は，km/hに換算した値とする。

qualityは，その標識の認識の確からしさを表す。認識の確度が低い場合はDegradedとなる。認識できなかった標識は要素として現れないため，qualityがUnavailableまたはInvalidとなることはない。状態全体の品質は，状態dataQualifierが表す。

recognizedSignsが変化したときは，RecognitionChangedイベントが発生する。要素の追加・削除，および要素の種別・速度規制の値・品質の変化のいずれでも発生する。同じ標識を再び認識してrecognizedAtのみが更新された場合は発生しない。イベントは事象の発生のみを伝え，値そのものは運ば

ない。値は，イベントを受けたアプリケーションがgetStatusにより取得する。

認識した標識の運転者への提示の方法（表示・音）は，機能クラスVisualPresentation，AudiblePresentationが宣言する。どの方法で提示するかは車両実装に依存し，提示1回ごとの方法は本オブジェクトでは表さない。

5.11.1.5. 標識の種別

標識の種別は，データ型TSRSignKindTypeで表す。TSRSignKindTypeは，国・地域によらず共通に用いられる主要な標識の種別を列挙する。これらのいずれにも当たらない標識をTSRが認識し，現在有効であると判断した場合は，その要素の種別をOtherとする。

国・地域に固有の標識の種別を表すため，TSRSignKindTypeには列挙子が追加されることがある。アプリケーションは，定義を知らない値の要素を，種別がOtherである要素と同様に扱わなければならない。

その車両のTSRが認識しうる標識の種別は，構成情報recognizableSignKindsが宣言する。recognizedSignsには，recognizableSignKindsに含まれる種別の要素のみが現れる。

5.11.1.6. TSRに対する制御の調停

本オブジェクトに対する制御の調停には，標準的な調停機能（後勝ちの原則）が適用される。すなわち，あるアプリケーションが作動を開始した後に，別のアプリケーションが作動の終了を要求した場合，後の要求が有効となる。

本オブジェクトは，標準的なロック機能を持つLockableObjectである。状態を変化させるサービスコール（activate，deactivate）およびロックを取得するサービスコール（lock）は，操作優先度priorityを持つ。ロック中の本オブジェクトに対して，ロックと同じかそれより低い優先度で操作した場合はE_OBJECT_LOCKEDエラーとなり，より高い優先度で操作した場合は，既存のロックが強制解除されて操作が実行される。アプリケーションが使用できる最高値より高い優先度を指定した場合はE_DENIED_PRIORITYエラー，操作優先度が有効範囲外である場合はE_INVALID_PARAMETERエラーとなる。

自アプリケーションのロックが強制解除された場合はOwnLockRevokedイベントが，他のアプリケーション等によりロックが解除された場合はOtherLockReleasedイベントが発生する。作動状態がFaultとなった場合，ロックは解除され，その間は新たにロックを取得することができない。

運転者による直接の操作（車載の設定画面等による有効化・無効化）は，本APIを用いない一つのアプリケーションによる操作と見なし，後勝ちの原則に従う。この操作は，アプリケーションが取得したロックおよび操作優先度によって妨げられることはなく，常に有効となる。運転の責任を負うのは運転者であり，アプリケーションによる排他が運転者の操作に優先してはならないためである。

5.11.1.7. 操作・参照に伴うリスク

TSRの作動を終了させると，認識した標識が運転者に提示されなくなり，運転者が標識を見落としたまま走行して，速度規制・一時停止等の交通規則に違反することにつながりうる。これをリスククラスRiskViolationとして定義し，activateおよびdeactivateをリスク制御の対象とする。作動の開始も，標識の提示により運転者の注意を引くものであり，同じリスククラスの対象とする。

TSRの作動状態および認識した標識の履歴は，運転者がいつ，どのような規制の道路を走行していたかを第三者が推定できる情報である。これをリスククラスRiskPrivacyとして定義し，状態を参照するサービスコール（getStatus），イベントの通知要求（notify）およびイベントの取得（getEvent）をリスク制御の対象とする。

いずれのリスククラスについても，当該リスクに対応していないアプリケーションからの呼び出しはE_RISK_APPLICATIONエラー，ユーザが当該リスクを承認していない場合はE_RISK_USERエラーとな

る。

構成情報の参照（getConfig）は，車両に搭載されたTSRの静的な性質を返すものであり，リスク制御の対象としない。

5.11.2. 機能クラス

TSRは，以下の機能クラスを持つ。

VisualPresentation	認識した標識を表示により運転者に提示することができる
AudiblePresentation	認識した標識の変化を音により運転者に知らせることができる

5.11.3. リスククラス

TSRは，以下のリスククラスを持つ。

RiskViolation	TSRの作動の開始・終了により標識の提示の有無が変わり，運転者が標識を見落としたまま走行して交通規則に違反するリスク
RiskPrivacy	走行状態・走行した道路の規制および運転者の運転への関与状態の開示により，個人の行動を推定できることに伴うプライバシのリスク

5.11.4. 構成情報

TSRは，以下の構成情報を持つ。

capabilities	enumSet (TSRCapabilityType)	その車両のTSRが持つ機能
riskClasses	enumSet(TSRRiskType)	車両およびビークルOSで対応できていないリスククラスの集合
supportedOperations	enumSet (TSROperationType)	アプリケーションから要求できる操作の集合。空集合となることがある
displayName	String?	ユーザに表示する呼称を取得するためのキー。車両が呼称を公開する場合にのみ値を持つ
recognizableSignKinds	enumSet (TSRSignKindType)	その車両のTSRが認識しうる標識の種別の集合
timestampAccuracy	UInt32	この車両のタイムスタンプ誤差上限の宣言。単位はμs

上記は，YAML形式で記述した構成記述ファイル中に記述する形式である。本オブジェクトはシングルインスタンスオブジェクトであるため，構成記述では，車両を表すノードの下に置かれたADADASノードの下に，オブジェクトを表すノードTSRを置き，その下に上記の構成情報を記述する。このノードは，Vehicle.ADADASの目録において，providingObjectがVehicle.ADADAS.TSRである要素に対応する。

riskClassesには，車両およびビークルOSで対応できていない（危険を防止できていない）リスククラスの集合を記述する。

構成情報を参照するサービスコールは，この構成情報に対して，次の変換を行った構成情報を返す。

- displayNameに対しては，それを元に現在のロケールに対応する呼称の文字列を取得し，その文字列

を返す。

5.11.5. 状態

TSRは，以下の状態を持つ。

timestamp	TimestampUsType	状態の測定時刻。単位はμs。車内共通時刻基盤に基づく
dataQualifier	DataQualifierType	このサンプルの品質。InvalidまたはUnavailableの場合，以下の状態の値を読んではならない。値の変化はDataQualifierChangedイベントにより通知される
operatingState	AutomationOperatingStateType	現在の作動状態。TSRではFallbackおよびMRCは現れない
availability	AutomationAvailabilityType	いま標識の認識を開始または継続できるか
unavailableReason	AutomationUnavailableReasonType?	認識を開始または継続できない理由。availabilityがAvailable以外の値である場合にのみ値を持つ
blockingFunction	RelatedObjectType[]	認識を妨げている機能オブジェクト。unavailableReasonがDependentFunctionFaultまたはSupersededByOtherFunctionである場合にのみ要素を持つ
faultCategory	AutomationFaultCategoryType?	故障の区分。operatingStateがFaultである場合にのみ値を持つ
recognizedSigns	TSRRecognizedSignType[]	現在有効であると認識している標識。operatingStateがActiveである場合にのみ要素を持ちうる
lockStatus	LockStatusType	ロック状態

状態は，呼び出し時点のTSRの状態を返す冪等なスナップショットである。値を読む前にdataQualifierを確認する。

TSRが標識を認識できる状態にあるかは，operatingStateにより判定する。いま現在有効であると認識している標識は，recognizedSignsにより判定する。operatingStateがActive以外である間，recognizedSignsは常に空の配列である。operatingStateがActiveであっても，現在有効であると認識している標識が無い場合，recognizedSignsは空の配列となる。

operatingStateとavailabilityの組合せの規則は，Vehicle.ADADASが規定するAD/ADAS機能共通の作動状態モデルによる。TSRが有効である間に，車両の状態が認識の条件を満たさない場合，availabilityはTemporarilyUnavailable，unavailableReasonはVehicleStateNotReadyとなり，カメラの性能低下・遮蔽により認識の性能が得られない場合，unavailableReasonはSensorDegradedとなる。いずれの場合も，operatingStateはStandbyとなる。operatingStateがOffである間のavailabilityは，車両の状態およびカメラの一時的な性能の低下を反映しない。

故障により機能を提供できない場合，operatingStateはFaultとなり，faultCategoryがその区分を表す。これにより，FaultDetectedイベントを受け取っていないアプリケーションも，状態の参照のみで故障の区分を知ることができる。本オブジェクト自体が状態を提供できない場合は，getStatusが

E_OBJECT_FAULTエラーを返す。

5.11.6. サービスコール一覧

TSRに対するサービスコールは以下の通り。

getConfig	TSRの構成情報の参照
getStatus	TSRの状態の参照
activate	TSRの作動の開始の要求
deactivate	TSRの作動の終了の要求
lock	TSRのロックの取得
unlock	TSRのロックの解除
notify	TSRイベントの通知要求
unnotify	TSRイベントの通知停止
getEvent	TSRイベントの取得

5.11.7. イベント

イベントは，事象が発生したことのみを伝え，状態の値そのものを運ばない。ただし，事象の発生時点の情報として有用なもの（故障の区分等）は，イベントの関連情報としても渡す。アプリケーションは，イベントを受けた後，必要に応じてgetStatusにより値を取得する。

状態の取得は，要求応答（getStatus）と，各状態の変化を伝えるイベントによる。

TSRに対するイベント種別は以下の通り。

OperatingStateChanged	作動状態の変化（アプリケーションの要求および運転者の設定による有効化・無効化を含む）
AvailabilityChanged	標識の認識を開始または継続できるかの変化（認識を妨げている理由の変化を含む）
RecognitionChanged	現在有効であると認識している標識（状態recognizedSigns）の変化。要素の追加・削除，および要素の種別・速度規制の値・品質の変化のいずれでも発生する
FaultDetected	故障の検知（TSRの機能を提供できない状態）
FaultRecovered	故障からの回復
DataQualifierChanged	状態の品質（状態dataQualifier）の変化
OwnLockRevoked	自アプリケーションのロックの強制解除
OtherLockReleased	他のアプリケーション等によるロック解除
EventOverflow	イベントキューのオーバフロー
ConfigChanged	構成情報の変更（ソフトウェアの更新による対応する操作・認識しうる標識の種別の変更）

この内，イベント種別がFaultDetectedの場合には，以下の関連情報を持つ。

faultCategory	AutomationFaultCategoryType	故障の区分

また，イベント種別がEventOverflowの場合には，以下の関連情報を持つ。

noLostEvents	UInt32	失われたイベントの数

5.11.8. データ型の定義

5.11.8.1. TSRの機能クラス

```
type TSRCapabilityType = enum {
    VisualPresentation,
    AudiblePresentation
};
```

5.11.8.2. TSRのリスククラス

```
type TSRRiskType = enum {
    RiskViolation,
    RiskPrivacy
};
```

5.11.8.3. タイムスタンプ

```
type TimestampUsType = UInt64;  // 車内共通時刻基盤の起点を基準とする測定時刻。単位はμs。複数のビークルAPIで共通に用いる共通データ型であり，本仕様内に定義する。
```

5.11.8.4. サンプル品質の種別

```
type DataQualifierType = enum {
    Normal, // 正常
    Unavailable,    // 利用不可
    Degraded,   // 劣化
    TestMode,   // テストモード
    Invalid // 無効（値を読んではならない）
};
```

複数のビークルAPIで共通に用いる共通データ型であり，本仕様内に定義する。

5.11.8.5. 速度

```
type SpeedType = Float32;   // 速度。単位はkm/h。正の値は前進，負の値は後退を示す。複
```

数のビークルAPIで共通に用いる共通データ型であり，本仕様内に定義する。

5.11.8.6. AD/ADAS機能共通の作動状態

以下の3つのデータ型は，AD/ADAS機能オブジェクトが共通に用いるデータ型であり，本仕様内に定義する。TSRにおける各列挙子の具体的な意味は，位置付けと機能の節に示す。

```
type AutomationOperatingStateType = enum {
    Off,    // 機能が無効化されており，作動していない状態
    Standby,    // 機能が有効であり，作動の条件が整えば作動できるが，現在は作動していない状態
    Activating, // 作動開始の要求を受理し，作動開始の処理を行っている過渡状態
    Active, // 作動中。車両の運動への介入または警報の出力を現に行っている状態
    Suspended,  // 作動が一時的に解除され，中断している状態。直前の作動時の設定値を保持しており，抑止する要因が解消すれば作動を再開できる
    Deactivating,   // 作動終了の要求を受理し，作動終了の処理を行っている過渡状態
    Fallback,   // 縮退運転を実行している状態
    MRC,    // 最小リスク状態へ移行し，これを維持している状態
    Fault   // 故障により機能を提供できない状態
};
```

各機能オブジェクトが現にどの作動状態にあるかを表す型である。機能によっては現れない列挙子がある。

```
type AutomationAvailabilityType = enum {
    Available,  // 作動を開始または継続できる
    TemporarilyUnavailable, // 一時的に条件を満たさないため，作動を開始または継続できない（条件が回復すれば作動できる）
    Unavailable // 故障等により，作動を開始または継続できない
};
```

その機能がいま作動を開始または継続できるかを表す型である。作動しているか否かは作動状態operatingStateが表すものであり，本型はこれを表さない。

```
type AutomationUnavailableReasonType = enum {
    ODDOutside, // 走行環境が運行設計領域の外にある
    SensorDegraded, // センサの性能低下・遮蔽により，必要な認識の性能が得られない
    DependentFunctionFault, // この機能が用いている他のAD/ADAS機能が故障している（blockingFunctionがその機能を指す）
    DriverMonitoringUnavailable,    // 運転者の監視が行えない
    VehicleStateNotReady,   // 車速・変速機の位置・制動等の車両の状態が作動の条件を満たさない
    FunctionFault,  // この機能自体が故障している
    ControlOverridden,  // 運転者の操作または他の経路による車両の運動の制御により，作動が抑止されている
    UserDisabled,   // 利用者の設定により機能が無効にされている
    SupersededByOtherFunction,  // 他のAD/ADAS機能がその運転操作を担っているため作動しない（blockingFunctionがその機能を指す）
```

```
    CalibrationRequired,    // 較正が必要で作動できない
    Other   // その他の理由
};
```

availabilityがAvailable以外の値である場合に，作動を開始または継続できない理由を表す型である。機能によっては現れない列挙子がある。

5.11.8.7. AD/ADAS機能共通の故障の区分

以下のデータ型は，AD/ADAS機能オブジェクトが共通に用いるデータ型であり，本仕様内に定義する。TSRは車両の運動に介入しないため，アクチュエータの故障（ActuatorFault）は現れない。

```
type AutomationFaultCategoryType = enum {
    SensorFault,    // センサの故障
    ActuatorFault,  // アクチュエータの故障
    ComputationFault,   // 演算装置の故障
    CommunicationFault, // 車内通信の故障
    PowerFault, // 電源の異常
    WarningDeviceFault, // 警報を提示する装置の故障
    Other   // その他の故障
};
```

機能を提供できない故障の区分を表す型である。機能によっては現れない列挙子がある。故障の区分は，各機能オブジェクトの状態faultCategoryおよびFaultDetectedイベントの関連情報により渡される。

5.11.8.8. TSRの操作

```
type TSROperationType = enum {
    Activate,   // 作動の開始の要求（サービスコールactivate）
    Deactivate  // 作動の終了の要求（サービスコールdeactivate）
};
```

アプリケーションから要求できる操作を表す型である。列挙子は，操作を行うサービスコールと1対1に対応する。すべてのアプリケーションが呼び出すことができるサービスコール（getConfig，getStatus，notify，unnotify，getEvent，lock，unlock）は，本型の対象としない。

5.11.8.9. 標識の種別

```
type TSRSignKindType = enum {
    SpeedLimit, // 最高速度の規制
    NoOvertaking,   // 追越しの禁止
    Stop,   // 一時停止
    Yield,  // 他の交通の進行を妨げてはならない（譲れ）
    NoEntry,    // 車両の進入の禁止
    Other   // 上記のいずれにも当たらない標識
};
```

TSRが認識する標識の種別を表す型である。列挙子の追加と，定義を知らない値の扱いは，位置付けと機能の節（標識の種別）に示す。

5.11.8.10. 認識している標識

```
type TSRRecognizedSignType = struct {
    kind : TSRSignKindType, // 標識の種別
    speedLimit : SpeedType?,    // 標識が示す最高速度。単位はkm/h。0以上の値である。kindがSpeedLimitである場合にのみ値を持つ
    quality : DataQualifierType,    // この標識の認識の品質
    recognizedAt : TimestampUsType  // この標識を最後に認識した時刻。単位はμs
};
```

TSRが現在有効であると認識している1つの標識を表す型である。標識の位置，自車からの距離，補助標識が示す条件は本型に含まない。

5.11.8.11. TSRの構成情報（getConfigが返すデータ型）

```
type TSRConfigType = struct {
    capabilities : enumSet(TSRCapabilityType),
    riskClasses : enumSet(TSRRiskType),
    supportedOperations : enumSet(TSROperationType),
    displayName : String?,
    recognizableSignKinds : enumSet(TSRSignKindType),
    timestampAccuracy : UInt32
};
```

5.11.8.12. TSR状態

```
type TSRStatusType = struct {
    timestamp : TimestampUsType,
    dataQualifier : DataQualifierType,
    operatingState : AutomationOperatingStateType,
    availability : AutomationAvailabilityType,
    unavailableReason : AutomationUnavailableReasonType?,
    blockingFunction : RelatedObjectType[],
    faultCategory : AutomationFaultCategoryType?,
    recognizedSigns : TSRRecognizedSignType[],
    lockStatus : LockStatusType
};
```

呼び出し時点のTSRの状態の冪等なスナップショットである。

5.11.8.13. TSRイベント

```
type TSREventKindType = enum {
    OperatingStateChanged,
```

```
    AvailabilityChanged,
    RecognitionChanged,
    FaultDetected,
    FaultRecovered,
    DataQualifierChanged,
    OwnLockRevoked,
    OtherLockReleased,
    EventOverflow,
    ConfigChanged
};
```

```
type TSREventType = struct {
    eventInfo : variant(TSREventKindType) {
        FaultDetected {
            faultCategory : AutomationFaultCategoryType
        },
        EventOverflow {
            noLostEvents : UInt32
        }
    }
};
```

5.11.9. サービスコールの詳細規定

getConfig

TSRの構成情報の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
config	TSRConfigType	TSRの構成情報

【エラーコード】

なし

【機能】

その車両のTSRが持つ機能，アプリケーションから要求できる操作の集合，および認識しうる標識の種別の集合を返す。

アプリケーションは，本サービスコールにより，画面を構成する前に，どの操作の要素を提示すべきかを決める。

getStatus

TSRの状態の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
status	TSRStatusType	TSRの状態

【エラーコード】

E_OBJECT_FAULT	本オブジェクトが故障しており，状態を提供できない
E_RISK_APPLICATION	アプリケーションがRiskPrivacyに対応していない
E_RISK_USER	ユーザがRiskPrivacyを承認していない

【機能】

TSRの現在の作動状態・利用可否・故障の区分・認識している標識・ロック状態を返す。

呼び出しのたびに，その時点の状態の全体を返す冪等な応答である。値を読む前にdataQualifierを確認する。

activate

TSRの作動の開始の要求

【パラメータ】

priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_PARAMETER	priorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_LOCKED	本オブジェクトが，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている
E_OBJECT_STATUS	作動を開始できる作動状態でない（既に有効である，またはavailabilityがUnavailableであり，その理由が故障以外である）
E_OBJECT_FAULT	本オブジェクトが故障しており，要求を受け付けられない（unavailableReasonがFunctionFaultである場合を含む）

E_INFEASIBLE	availabilityがTemporarilyUnavailableであり，作動の条件以外の一時的な理由（故障等）により，現在はTSRを有効にできない
E_RISK_APPLICATION	アプリケーションがRiskViolationに対応していない
E_RISK_USER	ユーザがRiskViolationを承認していない

【機能】

TSRに作動の開始を要求する。本サービスコールは，TSRを有効にする操作に相当し，Offからの遷移を要求する。認識の条件が整っている場合はActiveへ，整っていない場合はStandbyへ遷移する。

正常終了は要求が受理されたことのみを意味し，作動が開始したことを意味しない。遷移の完了は，getStatusまたはOperatingStateChangedイベントにより確認する。

作動を開始できない場合の応答は，利用可否availabilityとその理由unavailableReasonにより定まる。availabilityがTemporarilyUnavailableである場合はE_INFEASIBLEエラー，Unavailableでありunavailable ReasonがFunctionFaultである場合はE_OBJECT_FAULTエラー，Unavailableでありそれ以外の理由である場合はE_OBJECT_STATUSエラーとなる。

本サービスコールは，構成情報supportedOperationsにActivateを含む場合にのみ用いることができる。

deactivate

TSRの作動の終了の要求

【パラメータ】

priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_PARAMETER	priorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_LOCKED	本オブジェクトが，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている
E_OBJECT_STATUS	作動を終了できる作動状態でない（既にOffである等）
E_OBJECT_FAULT	本オブジェクトが故障しており，要求を受け付けられない
E_RISK_APPLICATION	アプリケーションがRiskViolationに対応していない
E_RISK_USER	ユーザがRiskViolationを承認していない

【機能】

TSRに作動の終了を要求する。本サービスコールは，TSRを無効にする操作に相当し，Offへの遷移を要求する。

正常終了は要求が受理されたことのみを意味し，作動が終了したことを意味しない。遷移の完了は，getStatusまたはOperatingStateChangedイベントにより確認する。

本サービスコールは，構成情報supportedOperationsにDeactivateを含む場合にのみ用いることができる。



本サービスコールによりTSRが無効になると，認識した標識は運転者に提示されなくなる。運転者が標識を見落としたまま走行することにつながるため，アプリケーションは，無効にしたことを運転者が認識できるようにしなければならない。

lock

TSRのロックの取得

【パラメータ】

priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_PARAMETER	priorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_LOCKED	本オブジェクトが，priorityと同じかそれより高い優先度でロックされている
E_OBJECT_FAULT	本オブジェクトが故障しており，ロックを取得できない

【機能】

本オブジェクトを，指定した操作優先度でロックする。ロックを取得したアプリケーションは，状態を変化させるサービスコール（activate，deactivate）を，同じかそれより低い優先度の他のアプリケーションによる操作から保護する。

既存のロックより高い優先度でロックを取得した場合，既存のロックは強制解除され，そのロックを取得していたアプリケーションにはOwnLockRevokedイベントが通知される。

運転者による直接の操作は，ロックおよび操作優先度によって妨げられない。

unlock

TSRのロックの解除

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_OBJECT_STATUS	自アプリケーションが，本オブジェクトをロックしていない

【機能】

自アプリケーションが取得した本オブジェクトのロックを解除する。

notify

TSRイベントの通知要求

【パラメータ】

eventFilter	enumSet (TSREventKindType)?	通知するイベントの種類の集合

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【エラーコード】

E_INVALID_PARAMETER	eventFilterが有効範囲外
E_RISK_APPLICATION	アプリケーションがRiskPrivacyに対応していない
E_RISK_USER	ユーザがRiskPrivacyを承認していない

【機能】

TSRで発生したeventFilterで指定したイベントの通知を要求する。イベントを通知するためのイベントキューを生成し，その通知ハンドルをnotifyHandleに返す。

eventFilterを省略した場合，すべての種類のイベントの通知を要求する。

認識した標識の変化の観測からも走行した道路を推定できるため，本サービスコールもRiskPrivacyのリスク制御の対象である。

unnotify

TSRイベントの通知停止

【パラメータ】

notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正

【機能】

notifyHandleで指定した通知を停止する。指定した通知用のイベントキューは削除される。

getEvent

TSRイベントの取得

【パラメータ】

notifyHandle	NotifyHandleType	イベントキューを識別するための通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
event	TSREventType	取得したイベント

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正
E_NO_DATA	イベントキューが空
E_RISK_APPLICATION	アプリケーションがRiskPrivacyに対応していない
E_RISK_USER	ユーザがRiskPrivacyを承認していない

【機能】

notifyHandleで指定した通知用のイベントキューから，先頭のイベントを取り出す。取り出したイベントは，イベントキューから削除される。

イベントの内容からも走行した道路を推定できるため，本サービスコールもRiskPrivacyのリスク制御の対象である。



イベントがパラメータ等で渡されてくるプログラミング環境向けの物理APIでは，用意する必要がない。

5.12. Vehicle.ADADAS.RCTA（略称：RCTA）

5.12.1. 位置付けと機能

Vehicle.ADADAS.RCTA（略称：RCTA）は，車両に搭載された後退時交差車両警報（RCTA）を参照・操作するためのシングルインスタンスオブジェクトである。作動の開始・終了の要求と，現在の作動状態および警報の出力の状況の参照を提供する。

本オブジェクトは，Vehicle.ADADASの目録に，機能の種別RCTA，提供するオブジェクトの完全修飾名Vehicle.ADADAS.RCTAとして現れる。DDTの責任主体・運行設計領域の充足状態・縮退運転・最小リスク状態など，AD/ADASシステム全体について一つに定まる状態は，本オブジェクトではなく

Vehicle.ADADASが提供する。

5.12.1.1. RCTAの位置付けと性質

RCTAは，自車の後退中（駐車場所からの出庫等）に，自車の後方を左右から横切って接近する車両・歩行者等を外界のセンサにより検知し，運転者に警報を出力する運転支援の機能である。RCTAは車両の運動に介入しない。警報のみを行い，制動の介入は行わないため，RCTAの作動中も車両の運動の制御は運転者が行う。

本オブジェクトは，RCTAが用いるセンサの構成，接近する対象の認識と警報の判定のアルゴリズムを規定しない。検知した対象の種別・位置・速度，自車の後方に到達するまでの時間といった判定の根拠は本オブジェクトでは扱わない。本オブジェクトが提供するのは，機能が作動しているか（operatingState），いま作動を継続できるか（availability），現に警報を出力しているか（functionOutput），左右それぞれの側から接近する対象に対して警報を出力しているか（leftOutput，rightOutput），および故障の区分（faultCategory）である。

5.12.1.2. 作動の開始・終了の操作

本オブジェクトは，作動の開始（activate）と終了（deactivate）をアプリケーションから要求できる。車両が備える設定の手段（車載の設定画面等）により運転者がRCTAを有効化・無効化した場合も，同じ作動状態operatingStateに反映される。

アプリケーションが行える操作は，構成情報supportedOperationsが宣言する。supportedOperationsの列挙子は，操作を行うサービスコールと1対1に対応する。アプリケーションは，画面を構成する前にsupportedOperationsを参照し，宣言されていない操作の要素を提示してはならない。宣言されていない操作のサービスコールを呼び出した場合，E_NOT_SUPPORTEDエラーとなる。

supportedOperationsが操作の可否（恒久的な可否）を表すのに対し，availabilityは，いまその操作を受け付けられるか（一時的な可否）を表す。この2つは別の判定であり，アプリケーションは，前者により要素を提示するか否かを，後者により要素を操作可能とするか否かを決める。

警報を出力しないようにすることは，後退中に接近する車両・歩行者に運転者が気付かない状態を生じうる。作動の終了の要求に伴うリスクは，操作・参照に伴うリスクの節に規定する。

5.12.1.3. 作動状態

本オブジェクトの作動状態operatingStateおよび利用可否availabilityは，Vehicle.ADADASが規定するAD/ADAS機能共通の作動状態モデルに従う。本節では，RCTAに固有の制約のみを規定する。

RCTAにおける各作動状態の意味は，次のとおりである。

- Off … RCTAが無効化されており，接近する対象を監視していない状態。アプリケーションによる作動の終了の要求，および運転者の設定のいずれによっても，この状態となる。
- Standby … RCTAが有効であるが，変速機の位置が後退（R）でない等の理由で，接近する対象を監視していない状態。
- Activating … 作動開始の要求（activate）を受理し，遷移の処理を行っている過渡状態。
- Active … RCTAが有効であり，変速機の位置が後退（R）であって，自車の後方を横切って接近する対象を監視している状態。この状態において，接近する対象を検知すると出力が生じ，functionOutputのactiveがtrueとなる。
- Deactivating … 作動終了の要求（deactivate）を受理し，Offへ遷移する処理を行っている過渡状態。

- Suspended … 他のAD/ADAS機能が後退の運転操作を担っているため，接近する対象の監視と警報を行わない状態。この場合，availabilityはTemporarilyUnavailable，unavailableReasonはSupersededByOtherFunctionとなり，blockingFunctionがその機能を指す。
- Fault … 故障により機能を提供できない状態。故障の区分は状態faultCategoryが表す。

RCTAでは，FallbackおよびMRCの列挙子は現れない。縮退運転および最小リスク状態への移行は，DDTの責任を持つ機能が行うものであり，警報のみを行うRCTAはこれを行わないためである。

5.12.1.4. 後退の状態と作動状態の関係

RCTAが監視を行うのは，変速機の位置が後退（R）である間に限られる。RCTAが有効である間，変速機の位置と作動状態は次の関係を持つ。

- 変速機の位置が後退（R）以外である間は，operatingStateはStandby，availabilityはTemporarilyUnavailable，unavailableReasonはVehicleStateNotReadyとなる。
- 変速機の位置が後退（R）となり，他の作動の条件が満たされると，operatingStateはActiveへ遷移する。変速機の位置が後退（R）であっても，車速等の他の作動の条件が満たされない間は，Standbyのままとなる。
- 変速機の位置が後退（R）から他の位置となると，operatingStateはActiveからStandbyへ遷移する。

したがって，RCTAが有効である車両では，後退の開始と終了のたびにStandbyとActiveの間の遷移が生じ，OperatingStateChangedイベントおよびAvailabilityChangedイベントが発生する。

operatingStateがOffである間，availabilityは，RCTAを有効にできるか（故障の有無等）のみを表し，有効化した後にのみ意味を持つ作動の条件（変速機の位置，車速等）を反映しない。すなわち，変速機の位置が後退（R）以外であることのみを理由として，Offである間のavailabilityがTemporarilyUnavailableとなることはない。アプリケーションは，後退していない間も作動の開始（activate）を要求でき，受理された場合，operatingStateはStandbyへ遷移する。

運転者の設定によりRCTAが無効化されている場合も，operatingStateはOffとなり，availabilityは前述の規定に従う。このときアプリケーションが行う作動の開始の要求は，後勝ちの原則に従い，運転者の設定より後に行われた操作として受理される。

車両が，アプリケーションからの作動の開始・終了そのものを認めない場合は，構成情報supportedOperationsにActivateおよびDeactivateを含めないことで表す。

警報を出力しているか否かは，作動状態ではなくfunctionOutputで表す。作動状態Activeは「監視しており，いつでも警報を出力できる状態」を表し，警報の出力そのものは示さない。

5.12.1.5. 警報の出力

警報の出力の状況は，左右の側ごとに状態leftOutputおよびrightOutputが表し，両側をまとめた状況を状態functionOutputが表す。左右は，自車の前方に向かっての左側・右側をいう。左側から接近する対象に対する出力はleftOutputに，右側から接近する対象に対する出力はrightOutputに現れる。

functionOutputは，leftOutputとrightOutputから次のように定まる。

- activeは，leftOutputとrightOutputの少なくとも一方のactiveがtrueである場合にtrueとなり，それ以外の場合にfalseとなる。
- levelは，activeがtrueである側の段階のうち，最も強い段階となる。

各出力のactiveは，その側から接近する対象を検知しているか，または運転者への提示を現に行っているかを表す。levelは出力の段階を表し，機能クラスWarningLevelReportedを備えている場合にのみ値を持つ。

段階により，「接近する対象を検知したが，まだ運転者への提示を行っていない状態」（Detected）と，「警報を出力している状態」（Advisory，Warning）とを区別できる。Advisoryは表示等による弱い提示を，Warningは音・振動を伴う提示を表す。RCTAは車両の運動に介入しないため，Criticalは現れない。その機能が出しうる最も強い段階は，構成情報maximumWarningLevelが宣言する。

警報の提示の方法（表示・音・触覚）は，機能クラスVisualWarning，AudibleWarning，HapticWarningが宣言する。どの方法で提示するかは車両実装に依存し，警報1回ごとの提示の方法は本オブジェクトでは表さない。

functionOutput，leftOutputおよびrightOutputのいずれかの値が変化したときは，FunctionOutputChangedイベントが発生する。一方の側の出力が変化し，functionOutputが変化しない場合にも発生する。イベントは事象の発生のみを伝え，値そのものは運ばない。値は，イベントを受けたアプリケーションがgetStatusにより取得する。



接近する対象の種別・位置・速度・進行方向，および自車の後方に到達するまでの時間は本オブジェクトでは提供しない。これらはVehicle.SurroundModelの担当である。

5.12.1.6. RCTAに対する制御の調停

本オブジェクトに対する制御の調停には，標準的な調停機能（後勝ちの原則）が適用される。すなわち，あるアプリケーションが作動を開始した後に，別のアプリケーションが作動の終了を要求した場合，後の要求が有効となる。

本オブジェクトは，標準的なロック機能を持つLockableObjectである。状態を変化させるサービスコール（activate，deactivate）およびロックを取得するサービスコール（lock）は，操作優先度priorityを持つ。ロック中の本オブジェクトに対して，ロックと同じかそれより低い優先度で操作した場合はE_OBJECT_LOCKEDエラーとなり，より高い優先度で操作した場合は，既存のロックが強制解除されて操作が実行される。アプリケーションが使用できる最高値より高い優先度を指定した場合はE_DENIED_PRIORITYエラー，操作優先度が有効範囲外である場合はE_INVALID_PARAMETERエラーとなる。

自アプリケーションのロックが強制解除された場合はOwnLockRevokedイベントが，他のアプリケーション等によりロックが解除された場合はOtherLockReleasedイベントが発生する。作動状態がFaultとなった場合，ロックは解除され，その間は新たにロックを取得することができない。

運転者による直接の操作（車載の設定画面等による有効化・無効化）は，本APIを用いない一つのアプリケーションによる操作と見なし，後勝ちの原則に従う。この操作は，アプリケーションが取得したロックおよび操作優先度によって妨げられることはなく，常に有効となる。運転の責任を負うのは運転者であり，アプリケーションによる排他が運転者の操作に優先してはならないためである。

5.12.1.7. 操作・参照に伴うリスク

RCTAの作動を終了させると，後退中に接近する車両・歩行者の警報が出力されなくなり，運転者が接近に気付かないまま後退することにつながりうる。これをリスククラスRiskCollisionとして定義し，activateおよびdeactivateをリスク制御の対象とする。作動の開始も，警報の出力により運転者の注意を引くものであり，同じリスククラスの対象とする。

RCTAの作動状態および警報の出力の履歴は，運転者がいつ後退していたか（駐車場所への出入りの時刻等）を第三者が推定できる情報である。これをリスククラスRiskPrivacyとして定義し，状態を参照するサービスコール（getStatus），イベントの通知要求（notify）およびイベントの取得（getEvent）をリス

ク制御の対象とする。

いずれのリスククラスについても，当該リスクに対応していないアプリケーションからの呼び出しはE_RISK_APPLICATIONエラー，ユーザが当該リスクを承認していない場合はE_RISK_USERエラーとなる。

構成情報の参照（getConfig）は，車両に搭載されたRCTAの静的な性質を返すものであり，リスク制御の対象としない。

5.12.2. 機能クラス

RCTAは，以下の機能クラスを持つ。

WarningLevelReported	出力の段階（検知，注意喚起，警報）を提供することができる（状態functionOutput，leftOutputおよびrightOutputのlevel）
VisualWarning	表示による警報を提示することができる
AudibleWarning	音による警報を提示することができる
HapticWarning	触覚（ステアリングまたは座席の振動等）による警報を提示することができる

5.12.3. リスククラス

RCTAは，以下のリスククラスを持つ。

RiskCollision	RCTAの作動の開始・終了により警報の出力の有無が変わり，運転者が後退中に接近する車両・歩行者に気付かないまま後退して衝突を生じるリスク
RiskPrivacy	走行状態および運転者の運転への関与状態の開示により，個人の行動を推定できることに伴うプライバシのリスク

5.12.4. 構成情報

RCTAは，以下の構成情報を持つ。

capabilities	enumSet (RCTACapabilityType)	その車両のRCTAが持つ機能
riskClasses	enumSet(RCTARiskType)	車両およびビークルOSで対応できていないリスククラスの集合
supportedOperations	enumSet (RCTAOperationType)	アプリケーションから要求できる操作の集合。空集合となることがある
displayName	String?	ユーザに表示する呼称を取得するためのキー。車両が呼称を公開する場合にのみ値を持つ
maximumWarningLevel	AutomationOutputLevelType?	この機能が出しうる最も強い段階。機能クラスWarningLevelReportedを備えている場合にのみ値を持つ
timestampAccuracy	UInt32	この車両のタイムスタンプ誤差上限の宣言。単位はμs

上記は，YAML形式で記述した構成記述ファイル中に記述する形式である。本オブジェクトはシングルイ

ンスタンスオブジェクトであるため，構成記述では，車両を表すノードの下に置かれたADADASノードの下に，オブジェクトを表すノードRCTAを置き，その下に上記の構成情報を記述する。このノードは，Vehicle.ADADASの目録において，providingObjectがVehicle.ADADAS.RCTAである要素に対応する。

riskClassesには，車両およびビークルOSで対応できていない（危険を防止できていない）リスククラスの集合を記述する。

構成情報を参照するサービスコールは，この構成情報に対して，次の変換を行った構成情報を返す。

- displayNameに対しては，それを元に現在のロケールに対応する呼称の文字列を取得し，その文字列を返す。

5.12.5. 状態

RCTAは，以下の状態を持つ。

timestamp	TimestampUsType	状態の測定時刻。単位はμs。車内共通時刻基盤に基づく
dataQualifier	DataQualifierType	このサンプルの品質。InvalidまたはUnavailableの場合，以下の状態の値を読んではならない。値の変化はDataQualifierChangedイベントにより通知される
operatingState	AutomationOperatingStateType	現在の作動状態。RCTAではFallbackおよびMRCは現れない
availability	AutomationAvailabilityType	いま接近する対象の監視を開始または継続できるか
unavailableReason	AutomationUnavailableReasonType?	監視を開始または継続できない理由。availabilityがAvailable以外の値である場合にのみ値を持つ
blockingFunction	RelatedObjectType[]	監視を妨げている機能オブジェクト。unavailableReasonがDependentFunctionFaultまたはSupersededByOtherFunctionである場合にのみ要素を持つ
faultCategory	AutomationFaultCategoryType?	故障の区分。operatingStateがFaultである場合にのみ値を持つ
functionOutput	AutomationFunctionOutputType	警報の出力の状況（左右の側をまとめたもの）。levelは機能クラスWarningLevelReportedを備えている場合にのみ値を持つ
leftOutput	AutomationFunctionOutputType	左側から接近する対象に対する警報の出力の状況。levelは機能クラスWarningLevelReportedを備えている場合にのみ値を持つ
rightOutput	AutomationFunctionOutputType	右側から接近する対象に対する警報の出力の状況。levelは機能クラスWarningLevelReportedを備えている場合にのみ値を持つ
lockStatus	LockStatusType	ロック状態

状態は，呼び出し時点のRCTAの状態を返す冪等なスナップショットである。値を読む前にdataQualifierを確認する。

RCTAが警報を出せる状態にあるかは，operatingStateにより判定する。いま接近する対象を検知しているか，警報を出しているか，それがどちらの側かは，functionOutput，leftOutputおよびrightOutputにより判定する。operatingStateがActive以外である間，これらのactiveは常にfalseである。

operatingStateとavailabilityの組合せの規則は，Vehicle.ADADASが規定するAD/ADAS機能共通の作動状態モデルによる。変速機の位置と作動状態の関係は，後退の状態と作動状態の関係の節による。

故障により機能を提供できない場合，operatingStateはFaultとなり，faultCategoryがその区分を表す。これにより，FaultDetectedイベントを受け取っていないアプリケーションも，状態の参照のみで故障の区分を知ることができる。本オブジェクト自体が状態を提供できない場合は，getStatusがE_OBJECT_FAULTエラーを返す。

5.12.6. サービスコール一覧

RCTAに対するサービスコールは以下の通り。

getConfig	RCTAの構成情報の参照
getStatus	RCTAの状態の参照
activate	RCTAの作動の開始の要求
deactivate	RCTAの作動の終了の要求
lock	RCTAのロックの取得
unlock	RCTAのロックの解除
notify	RCTAイベントの通知要求
unnotify	RCTAイベントの通知停止
getEvent	RCTAイベントの取得

5.12.7. イベント

イベントは，事象が発生したことのみを伝え，状態の値そのものを運ばない。ただし，事象の発生時点の情報として有用なもの（故障の区分等）は，イベントの関連情報としても渡す。アプリケーションは，イベントを受けた後，必要に応じてgetStatusにより値を取得する。

状態の取得は，要求応答（getStatus）と，各状態の変化を伝えるイベントによる。

RCTAに対するイベント種別は以下の通り。

OperatingStateChanged	作動状態の変化（アプリケーションの要求および運転者の設定による有効化・無効化，後退の開始・終了に伴うStandbyとActiveの間の遷移を含む）
AvailabilityChanged	接近する対象の監視を開始または継続できるかの変化（監視を妨げている理由の変化を含む）
FunctionOutputChanged	接近する対象に対する警報の出力の状況（状態functionOutput，leftOutputおよびrightOutput）の変化。いずれかの側の出力の開始・段階の変化・終了のいずれでも発生する
FaultDetected	故障の検知（RCTAの機能を提供できない状態）
FaultRecovered	故障からの回復

DataQualifierChanged	状態の品質（状態dataQualifier）の変化
OwnLockRevoked	自アプリケーションのロックの強制解除
OtherLockReleased	他のアプリケーション等によるロック解除
EventOverflow	イベントキューのオーバフロー
ConfigChanged	構成情報の変更（ソフトウェアの更新による対応する操作・出力の段階・提供する機能の変更）

この内，イベント種別がFaultDetectedの場合には，以下の関連情報を持つ。

faultCategory	AutomationFaultCategoryType	故障の区分

また，イベント種別がEventOverflowの場合には，以下の関連情報を持つ。

noLostEvents	UInt32	失われたイベントの数

5.12.8. データ型の定義

5.12.8.1. RCTAの機能クラス

```
type RCTACapabilityType = enum {
    WarningLevelReported,
    VisualWarning,
    AudibleWarning,
    HapticWarning
};
```

5.12.8.2. RCTAのリスククラス

```
type RCTARiskType = enum {
    RiskCollision,
    RiskPrivacy
};
```

5.12.8.3. タイムスタンプ

```
type TimestampUsType = UInt64;  // 車内共通時刻基盤の起点を基準とする測定時刻。単位はμs。複数のビークルAPIで共通に用いる共通データ型であり，本仕様内に定義する。
```

5.12.8.4. サンプル品質の種別

```
type DataQualifierType = enum {
    Normal, // 正常
    Unavailable,    // 利用不可
```

```
    Degraded,   // 劣化
    TestMode,   // テストモード
    Invalid // 無効（値を読んではならない）
};
```

複数のビークルAPIで共通に用いる共通データ型であり，本仕様内に定義する。

5.12.8.5. AD/ADAS機能共通の作動状態

以下の5つのデータ型は，AD/ADAS機能オブジェクトが共通に用いるデータ型であり，本仕様内に定義する。RCTAにおける各列挙子の具体的な意味は，位置付けと機能の節に示す。

```
type AutomationOperatingStateType = enum {
    Off,    // 機能が無効化されており，作動していない状態
    Standby,    // 機能が有効であり，作動の条件が整えば作動できるが，現在は作動していない状態
    Activating, // 作動開始の要求を受理し，作動開始の処理を行っている過渡状態
    Active, // 作動中。車両の運動への介入または警報の出力を現に行っている状態
    Suspended,  // 作動が一時的に解除され，中断している状態。直前の作動時の設定値を保持しており，抑止する要因が解消すれば作動を再開できる
    Deactivating,   // 作動終了の要求を受理し，作動終了の処理を行っている過渡状態
    Fallback,   // 縮退運転を実行している状態
    MRC,    // 最小リスク状態へ移行し，これを維持している状態
    Fault   // 故障により機能を提供できない状態
};
```

各機能オブジェクトが現にどの作動状態にあるかを表す型である。機能によっては現れない列挙子がある。

```
type AutomationAvailabilityType = enum {
    Available,  // 作動を開始または継続できる
    TemporarilyUnavailable, // 一時的に条件を満たさないため，作動を開始または継続できない（条件が回復すれば作動できる）
    Unavailable // 故障等により，作動を開始または継続できない
};
```

その機能がいま作動を開始または継続できるかを表す型である。作動しているか否かは作動状態operatingStateが表すものであり，本型はこれを表さない。

```
type AutomationUnavailableReasonType = enum {
    ODDOutside, // 走行環境が運行設計領域の外にある
    SensorDegraded, // センサの性能低下・遮蔽により，必要な認識の性能が得られない
    DependentFunctionFault, // この機能が用いている他のAD/ADAS機能が故障している（blockingFunctionがその機能を指す）
    DriverMonitoringUnavailable,    // 運転者の監視が行えない
    VehicleStateNotReady,   // 車速・変速機の位置・制動等の車両の状態が作動の条件を満たさない
    FunctionFault,  // この機能自体が故障している
```

```
    ControlOverridden,  // 運転者の操作または他の経路による車両の運動の制御により，作動が抑止されている
    UserDisabled,   // 利用者の設定により機能が無効にされている
    SupersededByOtherFunction,  // 他のAD/ADAS機能がその運転操作を担っているため作動しない（blockingFunctionがその機能を指す）
    CalibrationRequired,    // 較正が必要で作動できない
    Other   // その他の理由
};
```

availabilityがAvailable以外の値である場合に，作動を開始または継続できない理由を表す型である。機能によっては現れない列挙子がある。

```
type AutomationOutputLevelType = enum {
    Detected,   // 事象を検知したが，運転者への提示はまだ行っていない
    Advisory,   // 注意喚起（表示等の弱い提示）
    Warning,    // 警報（音・振動を伴う提示）
    Critical    // 緊急（車両運動への介入を伴う）
};
```

機能の出力の段階を表す型である。Detected，Advisory，Warning，Criticalの順に強い段階を表す。機能によっては現れない段階がある。

```
type AutomationFunctionOutputType = struct {
    active : Bool,  // その機能が事象の検知または運転者への提示を現に行っているか
    level : AutomationOutputLevelType?  // 出力の段階。activeがtrueであり，かつ段階を提供できる機能である場合にのみ値を持つ
};
```

機能が外部（運転者）に対して現に行っている出力を，有無と段階のみで表す型である。出力の判断の根拠（検知した車線の形状・物体の種別等）は本型に含まない。

5.12.8.6. AD/ADAS機能共通の故障の区分

以下のデータ型は，AD/ADAS機能オブジェクトが共通に用いるデータ型であり，本仕様内に定義する。RCTAは車両の運動に介入しないため，アクチュエータの故障（ActuatorFault）は現れない。

```
type AutomationFaultCategoryType = enum {
    SensorFault,    // センサの故障
    ActuatorFault,  // アクチュエータの故障
    ComputationFault,   // 演算装置の故障
    CommunicationFault, // 車内通信の故障
    PowerFault, // 電源の異常
    WarningDeviceFault, // 警報を提示する装置の故障
    Other   // その他の故障
};
```

機能を提供できない故障の区分を表す型である。機能によっては現れない列挙子がある。故障の区分は，各機能オブジェクトの状態faultCategoryおよびFaultDetectedイベントの関連情報により渡される。

5.12.8.7. RCTAの操作

```
type RCTAOperationType = enum {
    Activate,   // 作動の開始の要求（サービスコールactivate）
    Deactivate  // 作動の終了の要求（サービスコールdeactivate）
};
```

アプリケーションから要求できる操作を表す型である。列挙子は，操作を行うサービスコールと1対1に対応する。すべてのアプリケーションが呼び出すことができるサービスコール（getConfig，getStatus，notify，unnotify，getEvent，lock，unlock）は，本型の対象としない。

5.12.8.8. RCTAの構成情報（getConfigが返すデータ型）

```
type RCTAConfigType = struct {
    capabilities : enumSet(RCTACapabilityType),
    riskClasses : enumSet(RCTARiskType),
    supportedOperations : enumSet(RCTAOperationType),
    displayName : String?,
    maximumWarningLevel : AutomationOutputLevelType?,
    timestampAccuracy : UInt32
};
```

5.12.8.9. RCTA状態

```
type RCTAStatusType = struct {
    timestamp : TimestampUsType,
    dataQualifier : DataQualifierType,
    operatingState : AutomationOperatingStateType,
    availability : AutomationAvailabilityType,
    unavailableReason : AutomationUnavailableReasonType?,
    blockingFunction : RelatedObjectType[],
    faultCategory : AutomationFaultCategoryType?,
    functionOutput : AutomationFunctionOutputType,
    leftOutput : AutomationFunctionOutputType,
    rightOutput : AutomationFunctionOutputType,
    lockStatus : LockStatusType
};
```

呼び出し時点のRCTAの状態の冪等なスナップショットである。

5.12.8.10. RCTAイベント

```
type RCTAEventKindType = enum {
    OperatingStateChanged,
    AvailabilityChanged,
    FunctionOutputChanged,
    FaultDetected,
```

```
    FaultRecovered,
    DataQualifierChanged,
    OwnLockRevoked,
    OtherLockReleased,
    EventOverflow,
    ConfigChanged
};
```

```
type RCTAEventType = struct {
    eventInfo : variant(RCTAEventKindType) {
        FaultDetected {
            faultCategory : AutomationFaultCategoryType
        },
        EventOverflow {
            noLostEvents : UInt32
        }
    }
};
```

5.12.9. サービスコールの詳細規定

getConfig

RCTAの構成情報の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
config	RCTAConfigType	RCTAの構成情報

【エラーコード】

なし

【機能】

その車両のRCTAが持つ機能，アプリケーションから要求できる操作の集合，およびこの機能が出しうる最も強い段階を返す。

アプリケーションは，本サービスコールにより，画面を構成する前に，どの操作の要素を提示すべきかを決める。

getStatus

RCTAの状態の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
status	RCTAStatusType	RCTAの状態

【エラーコード】

E_OBJECT_FAULT	本オブジェクトが故障しており，状態を提供できない
E_RISK_APPLICATION	アプリケーションがRiskPrivacyに対応していない
E_RISK_USER	ユーザがRiskPrivacyを承認していない

【機能】

RCTAの現在の作動状態・利用可否・故障の区分・警報の出力の状況（左右の側ごとの状況を含む）・ロック状態を返す。

呼び出しのたびに，その時点の状態の全体を返す冪等な応答である。値を読む前にdataQualifierを確認する。

activate

RCTAの作動の開始の要求

【パラメータ】

priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_PARAMETER	priorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_LOCKED	本オブジェクトが，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている
E_OBJECT_STATUS	作動を開始できる作動状態でない（既に有効である，またはavailabilityがUnavailableであり，その理由が故障以外である）
E_OBJECT_FAULT	本オブジェクトが故障しており，要求を受け付けられない（unavailableReasonがFunctionFaultである場合を含む）
E_INFEASIBLE	availabilityがTemporarilyUnavailableであり，作動の条件以外の一時的な理由（故障等）により，現在はRCTAを有効にできない
E_RISK_APPLICATION	アプリケーションがRiskCollisionに対応していない

E_RISK_USER	ユーザがRiskCollisionを承認していない

【機能】

RCTAに作動の開始を要求する。本サービスコールは，RCTAを有効にする操作に相当し，Offからの遷移を要求する。変速機の位置が後退（R）であり，作動の条件が整っている場合はActiveへ，それ以外の場合はStandbyへ遷移する。作動の条件が整っていないことは，本要求を受け付けられない理由とはならない。

正常終了は要求が受理されたことのみを意味し，作動が開始したことを意味しない。遷移の完了は，getStatusまたはOperatingStateChangedイベントにより確認する。

作動を開始できない場合の応答は，利用可否availabilityとその理由unavailableReasonにより定まる。availabilityがTemporarilyUnavailableである場合はE_INFEASIBLEエラー，Unavailableでありunavailable ReasonがFunctionFaultである場合はE_OBJECT_FAULTエラー，Unavailableでありそれ以外の理由である場合はE_OBJECT_STATUSエラーとなる。

本サービスコールは，構成情報supportedOperationsにActivateを含む場合にのみ用いることができる。

deactivate

RCTAの作動の終了の要求

【パラメータ】

priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_PARAMETER	priorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_LOCKED	本オブジェクトが，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている
E_OBJECT_STATUS	作動を終了できる作動状態でない（既にOffである等）
E_OBJECT_FAULT	本オブジェクトが故障しており，要求を受け付けられない
E_RISK_APPLICATION	アプリケーションがRiskCollisionに対応していない
E_RISK_USER	ユーザがRiskCollisionを承認していない

【機能】

RCTAに作動の終了を要求する。本サービスコールは，RCTAを無効にする操作に相当し，Offへの遷移を要求する。

正常終了は要求が受理されたことのみを意味し，作動が終了したことを意味しない。遷移の完了は，getStatusまたはOperatingStateChangedイベントにより確認する。

本サービスコールは，構成情報supportedOperationsにDeactivateを含む場合にのみ用いることができる。



本サービスコールによりRCTAが無効になると，後退中に接近する車両・歩行者の警報は出力されなくなる。運転者が接近に気付かないまま後退することにつながるため，アプリケーションは，無効にしたことを運転者が認識できるようにしなければならない。

lock

RCTAのロックの取得

【パラメータ】

priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_PARAMETER	priorityが有効範囲外
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_LOCKED	本オブジェクトが，priorityと同じかそれより高い優先度でロックされている
E_OBJECT_FAULT	本オブジェクトが故障しており，ロックを取得できない

【機能】

本オブジェクトを，指定した操作優先度でロックする。ロックを取得したアプリケーションは，状態を変化させるサービスコール（activate，deactivate）を，同じかそれより低い優先度の他のアプリケーションによる操作から保護する。

既存のロックより高い優先度でロックを取得した場合，既存のロックは強制解除され，そのロックを取得していたアプリケーションにはOwnLockRevokedイベントが通知される。

運転者による直接の操作は，ロックおよび操作優先度によって妨げられない。

unlock

RCTAのロックの解除

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_OBJECT_STATUS	自アプリケーションが，本オブジェクトをロックしていない

【機能】

自アプリケーションが取得した本オブジェクトのロックを解除する。

notify

RCTAイベントの通知要求

【パラメータ】

eventFilter	enumSet (RCTAEventKindType)?	通知するイベントの種類の集合

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【エラーコード】

E_INVALID_PARAMETER	eventFilterが有効範囲外
E_RISK_APPLICATION	アプリケーションがRiskPrivacyに対応していない
E_RISK_USER	ユーザがRiskPrivacyを承認していない

【機能】

RCTAで発生したeventFilterで指定したイベントの通知を要求する。イベントを通知するためのイベントキューを生成し，その通知ハンドルをnotifyHandleに返す。

eventFilterを省略した場合，すべての種類のイベントの通知を要求する。

作動状態の変化および警報の発生の観測からも後退の時刻を推定できるため，本サービスコールもRiskPrivacyのリスク制御の対象である。

unnotify

RCTAイベントの通知停止

【パラメータ】

notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正

【機能】

notifyHandleで指定した通知を停止する。指定した通知用のイベントキューは削除される。

getEvent

RCTAイベントの取得

【パラメータ】

notifyHandle	NotifyHandleType	イベントキューを識別するための通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
event	RCTAEventType	取得したイベント

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正
E_NO_DATA	イベントキューが空
E_RISK_APPLICATION	アプリケーションがRiskPrivacyに対応していない
E_RISK_USER	ユーザがRiskPrivacyを承認していない

【機能】

notifyHandleで指定した通知用のイベントキューから，先頭のイベントを取り出す。取り出したイベントは，イベントキューから削除される。

イベントの内容からも後退の時刻を推定できるため，本サービスコールもRiskPrivacyのリスク制御の対象である。



イベントがパラメータ等で渡されてくるプログラミング環境向けの物理APIでは，用意する必要がない。

5.13. Vehicle.CurrentLocation（略称：CurrentLocation）

5.13.1. 位置付けと機能

Vehicle.CurrentLocation（略称：CurrentLocation）は，車両の現在位置（自己位置）を提供するシングルインスタンスオブジェクトである。緯度・経度・高度・進行方位と，それぞれの推定誤差，および測定時刻とサンプル品質を状態として提供する。ローカル直交座標系における位置を併せて提供することもできる。

5.13.1.1. CurrentLocationの位置付けと性質

CurrentLocationの値は，車両としての自己位置のベスト推定値であり，算出手段（GNSS単独測位・ジャイロや車速等との複合測位・地図や周辺監視情報との融合など）は規定しない。算出手段や走行環境（トンネル・高層ビル街等）による精度の違いは，各推定誤差フィールドの実値として表現され，APIとしての精度の保証は行わない。latitude，longitude，altitudeが表す位置は，車両基準点（ISO 8855の車両座標系原点。通常は後軸中心の路面投影点）の位置である。基準点は構成情報referencePointで宣言され，独自の基準点は用いない。

CurrentLocationが提供するのは位置の幾何情報（緯度経度・高度・方位，ローカル直交座標系における位置，およびそれらの推定誤差）のみである。現在位置の住所（行政区域）や走行中の道路の種別など，位置を地図と照合して得られる情報はVehicle.Navigationが担当し，CurrentLocationは持たない。また，CurrentLocationは，特定の道路・車線等の地図オブジェクトに対する相対位置（車線ID，車線内の縦方向位置・横方向位置など）を提供しない。

5.13.1.2. 位置を表す座標系

CurrentLocationは，地理座標系における位置（緯度・経度・高度・進行方位）を常に提供する。地理座標系はWGS84とし，その実現版は構成情報geodeticDatumで宣言される。高度は楕円体高とし，進行方位は真北を0とする時計回りの角度とする。標高（ジオイド面を基準とする高さ）や磁北を基準として位置・方位を算出する実装では，車両側実装が楕円体高および真北基準へ換算して返す。

CurrentLocationは，これに加えて，ローカル直交座標系における位置（状態cartesianPose）をオプションとして併せて提供することができる。これは，地図に基づく自己位置推定を行う車両が，内部で用いる座標系での値を変換損失なく提供するための手段である。ローカル直交座標系における位置を提供する場合も，地理座標系における位置の提供を省略してはならない。これにより，ローカル直交座標系を解釈しない利用側でも，CurrentLocationを利用できる。

ローカル直交座標系は，構成情報localFrameで定義される。localFrameは，座標系の識別子に加えて，原点（地理座標系における緯度・経度・高度），軸のとり方，軸のとり方を基準とする水平面内の追加回転角，地理座標系に対するスケール，座標系が有効な範囲，および変換の許容残差を宣言する。この宣言により，ローカル直交座標系を解釈しない利用側でも，cartesianPoseを地理座標へ変換できる。地理座標系との変換は，原点における局所水平面（接平面）による近似とする。ローカル直交座標系は水平とみなし，回転は水平面内の1自由度のみとする。

5.13.1.3. 推定誤差の表現

各計測値の推定誤差（horizontalAccuracy，verticalAccuracy，headingAccuracy）が表す信頼水準は，95%に固定する。高度および進行方位については，真値が推定値±推定誤差の範囲に入る確率が95%（2σ相当）となる。水平位置については，真値が推定位置を中心とする半径horizontalAccuracyの円内に入る確率が95%となる。1σ等の他の信頼水準で誤差を算出する実装では，車両側実装が95%相当へ換算して返す。これは精度の保証ではなく，誤差申告の意味の統一である。

水平位置の推定誤差は，円の半径による表現（horizontalAccuracy）に加えて，誤差楕円（horizontalErrorEllipse）の形式でも表すことができる。誤差楕円は，半長軸・半短軸と長軸の方位で表し，その信頼水準も95%に固定する。horizontalErrorEllipseとhorizontalAccuracyは，同一の推定誤差分布から導かれなければならない。

推定誤差を算出できない実装や瞬間には，推定誤差フィールドは値なしとなる。

5.13.1.4. 測定時刻とサンプル品質

状態には，位置の測定時刻timestampとサンプル品質dataQualifierが含まれる。timestampは，車内共通時刻基盤に基づく測定時刻（センサ観測のエポック）であり，送信時刻ではない。同じ時刻基盤に基づく

他のビークルAPIのタイムスタンプと直接比較できる。タイムスタンプ自体の誤差上限は，構成情報timestampAccuracyで宣言される。

dataQualifierは，取得できたサンプルの品質を表す。故障（自己位置を推定できない状態。E_OBJECT_FAULTエラーおよびFaultDetectedイベント）が「推定値を出力できない状態」を表すのに対し，dataQualifierは「出力されたサンプルがどれだけ信用できるか」を表す。両者の使い分けの詳細は，状態の節に規定する。

5.13.1.5. 状態の更新と参照に伴うプライバシのリスク

状態の更新周期は車両実装に依存する。状態は，要求応答（getStatus）により取得できるほか，状態報告イベント（StatusReported）によって周期的に得ることができる。位置は連続更新される信号系データであるため，位置の更新のたびに発生するイベントは定義しない。位置の値は，StatusReportedイベントの関連情報statusにより周期的に得られる。

位置情報および位置の履歴は，個人の行動を推定できる高感度な情報である。そのため，状態を参照するサービスコール（getStatus）はリスククラスRiskPrivacyの対象であり，RiskPrivacyに対応していないアプリケーションに対してはE_RISK_APPLICATIONエラー，ユーザが状態の開示を承認していない場合はE_RISK_USERエラーとなる。イベントの通知要求（notify）により通知されるイベントには，報告時点の状態の全量を運ぶStatusReportedイベントが含まれることがあり，また，FaultDetected等のイベントの発生パターンは走行環境（トンネル等）と相関しうる。このため，notifyの呼び出しそのものを，eventFilterで指定するイベントの種類によらず，RiskPrivacyの対象とする。

5.13.2. 機能クラス

CurrentLocationは，以下の機能クラスを持つ。

AltitudeDetectable	高度を提供することができる
HeadingDetectable	進行方位を提供することができる
AccuracyDetectable	各計測値の推定誤差を提供することができる
ErrorEllipseDetectable	水平位置の推定誤差を誤差楕円の形式で提供することができる
CartesianPoseProvidable	ローカル直交座標系における位置を提供することができる

ErrorEllipseDetectableは，AccuracyDetectableを備えている場合にのみ備えることができる。

5.13.3. リスククラス

CurrentLocationは，以下のリスククラスを持つ。

RiskPrivacy	緯度・経度・高度・進行方位など自車の測位情報の開示に伴うプライバシのリスク（例：未対応アプリケーションへの位置の開示，ユーザが承認していない状態での開示）

5.13.4. 構成情報

CurrentLocationは，以下の構成情報を持つ。

capabilities	enumSet (CurrentLocationCapabilityType)	そのロケータが持つ機能
riskClasses	enumSet (CurrentLocationRiskType)	そのロケータのリスククラスの集合
geodeticDatum	String	位置（latitude，longitude，altitude）の測地系の宣言。地理座標系はWGS84とし，その実現版を宣言する
localFrame	CurrentLocationLocalFrameType?	ローカル直交座標系の定義。機能クラスCartesianPoseProvidableを備えている場合は必ず値を持ち，備えていない場合は値なしとする
referencePoint	String	位置の基準点の宣言（ISO 8855の車両座標系原点：後軸中心の路面投影点）
statusReportInterval	UInt32 ([1,4294967295])?	状態報告イベント（StatusReported）が発生する周期。単位はms。値を持つ場合は1以上。値なしの場合，この車両ではStatusReportedイベントは発生しない
timestampAccuracy	UInt32	この車両のタイムスタンプ誤差上限の宣言。単位はμs

5.13.5. 状態

CurrentLocationは，以下の状態を持つ。

timestamp	TimestampUsType	測定時刻。単位はμs。車内共通時刻基盤に基づく。dataQualifierがUnavailableのサンプルのtimestampは，サンプルの生成時刻を表す（測定時刻ではない）
dataQualifier	DataQualifierType	このサンプルの品質。InvalidまたはUnavailableの場合，位置（cartesianPoseを含む）・方位および推定誤差（horizontalErrorEllipseを含む）のいずれも読んではならない
latitude	Float64 ([-90.0,90.0])	緯度。単位はdeg（WGS84）。北緯が正
longitude	Float64 ([-180.0,180.0])	経度。単位はdeg（WGS84）。東経が正
altitude	Float64?	高度。単位はm（WGS84楕円体高）
heading	Float64 ([0.0,360.0))?	進行方位。単位はdeg。真北を0とし時計回り
horizontalAccuracy	Float64 ([0.0,+INF))?	水平位置（緯度経度）の推定誤差。推定位置を中心とし，真値を確率95%で含む円の半径。単位はm
verticalAccuracy	Float64 ([0.0,+INF))?	高度の推定誤差。単位はm。信頼水準は95%（2σ相当）
headingAccuracy	Float64 ([0.0,+INF))?	進行方位の推定誤差。単位はdeg。信頼水準は95%（2σ相当）

horizontalErrorEllipse	CurrentLocationErrorEllipseType?	水平位置の推定誤差の誤差楕円による表現。信頼水準は95%
cartesianPose	CurrentLocationCartesianPoseType?	構成情報localFrameが定義するローカル直交座標系における位置と進行方位

altitudeは，機能クラスAltitudeDetectableを備えている場合にのみ値を持つ。headingは，機能クラスHeadingDetectableを備え，かつ方位を推定できている場合にのみ値を持つ。停車中・極低速時には，進行方位を推定できず値なしとなる場合がある。

horizontalAccuracy，verticalAccuracy，headingAccuracyは，機能クラスAccuracyDetectableを備え，かつ対応する計測値の推定誤差を算出できている場合にのみ値を持つ。verticalAccuracyは機能クラスAltitudeDetectableを，headingAccuracyは機能クラスHeadingDetectableを併せて備えている場合にのみ値を持つ。horizontalErrorEllipseは，機能クラスErrorEllipseDetectableを備え，かつ誤差楕円を算出できている場合にのみ値を持つ。すべての推定誤差が表す信頼水準は，95%に固定する。高度および進行方位の推定誤差については，これは2σ相当である。horizontalErrorEllipseとhorizontalAccuracyは，同一の推定誤差分布から導かれなければならない。

cartesianPoseは，機能クラスCartesianPoseProvidableを備え，かつ車両基準点の位置が構成情報localFrameの原点からvalidRadius以内にある場合にのみ値を持つ。原点からの距離は，scaleを適用した水平方向の地上距離で判定する。cartesianPoseのzは機能クラスAltitudeDetectableを備えている場合にのみ，cartesianPoseのheadingは機能クラスHeadingDetectableを備え，かつ方位を推定できている場合にのみ値を持つ。

cartesianPoseが値を持つ場合，cartesianPoseと，latitude・longitude・altitudeは，同一の測定時刻における同一の車両基準点（構成情報referencePointが宣言する点）の位置を表す。両者は，構成情報localFrameが宣言する変換によって同一の位置を表さなければならない。headingとcartesianPoseのheadingがともに値を持つ場合も同様に，両者は，localFrameが宣言する軸のとり方と回転によって同一の方位を表さなければならない。

この一致は，次の基準によって判定する。cartesianPoseを構成情報localFrameの宣言に従い，原点における局所水平面（接平面）近似によって地理座標へ変換して得た水平位置と，latitude・longitudeが表す水平位置との差が，構成情報localFrameのconversionToleranceを超えないことをもって判定する。高度については，cartesianPoseのzとaltitudeがともに値を持つ場合に，変換して得た高度とaltitudeとの差が，conversionToleranceを超えないことをもって判定する。

cartesianPoseのframeIdは，その位置が基づくローカル直交座標系を表す。利用側は，サンプルのframeIdが，保持している構成情報localFrameのframeIdと一致しない場合，当該cartesianPoseを使用してはならない。この場合，利用側は構成情報を再取得しなければならない。

車両基準点の位置が構成情報localFrameの原点からvalidRadiusを超えた場合，cartesianPoseは値なしとなる。車両側実装が別のローカル直交座標系へ切り替える場合は，構成情報localFrameを更新し，ConfigChangedイベントを発生させる。切り替え後のcartesianPoseは，更新後のlocalFrameに基づき，そのframeIdは更新後のlocalFrameのframeIdと一致する。車両側実装は，validRadiusに達する前に座標系を切り替えることが望ましい。

緯度・経度・高度・進行方位は，validRadiusの影響を受けず，cartesianPoseの値の有無にかかわらず提供される。

timestampは，このサンプルの位置の測定時刻（センサ観測のエポック）であり，車内共通時刻基盤に基づく。タイムスタンプの誤差上限は，構成情報timestampAccuracyで宣言される。

dataQualifierは，このサンプルの品質を表す。Invalidは，推定器自身が自分の出力（推定誤差の値を含

む）を信用できない状態（起動直後の初期化中・フィルタ発散中など）であり，位置（cartesianPoseを含む）・方位および推定誤差（horizontalErrorEllipseを含む）のいずれも読んではならない。Invalidを大きな推定誤差の値で代用してはならない。Degradedは，値は使えるが通常品質でない状態（例：トンネル等でGNSSを長時間失い，デッドレコニング主体での推定が継続している場合）であり，精度そのものは推定誤差の実値で判断する。

自己位置を推定できない状態（推定値を出力できない状態）は故障と扱い，故障中の状態の参照（getStatus）はE_OBJECT_FAULTエラーとなる。故障中にStatusReportedイベントで報告される状態は，dataQualifierがUnavailableである。UnavailableのサンプルもInvalidと同様に，位置（cartesianPoseを含む）・方位および推定誤差（horizontalErrorEllipseを含む）のいずれも読んではならない（最終既知値として読むことも許されない）。Unavailableのサンプルのtimestampは，サンプルの生成時刻を表す（測定時刻ではない）。トンネル等で一時的にGNSSを利用できなくても，複合測位等により自己位置を推定できている間は故障とせず，精度の低下は推定誤差の実値で表現する。

5.13.6. サービスコール一覧

CurrentLocationに対するサービスコールは以下の通り。

getConfig	CurrentLocationの構成情報の参照
getStatus	現在位置の参照
notify	CurrentLocationイベントの通知要求
unnotify	CurrentLocationイベントの通知停止
getEvent	CurrentLocationイベントの取得

5.13.7. イベント

CurrentLocationに対するイベント種別は以下の通り。

FaultDetected	故障検知（自己位置を推定できない状態）
FaultRecovered	故障から回復
StatusReported	状態の報告（構成情報statusReportIntervalの周期ごとに発生する）
EventOverflow	イベントキューのオーバフロー
ConfigChanged	構成情報の変更

StatusReportedイベントの関連情報statusは，報告時点の現在位置の状態の全量であり，getStatusが返す状態と同じ型（CurrentLocationStatusType）である。関連情報timestamp（TimestampMsType）と，状態が持つtimestamp（TimestampUsType）とは単位が異なり，起点が同一であることも保証されないため，互いに比較することはできない。状態の更新周期と構成情報statusReportIntervalとは一致するとは限らない。dataQualifierがUnavailableでないサンプルどうしでは，連続する報告の関連情報statusが同じ測位結果に基づくか否かを，statusのtimestampにより判断できる。dataQualifierがUnavailableのサンプルのtimestampはサンプルの生成時刻を表すため，これにより判断することはできない。

この内，イベント種別がStatusReportedの場合には，以下の関連情報を持つ。

timestamp	TimestampMsType	報告が発生した時刻

status	CurrentLocationStatusType	報告時点の現在位置の状態（全量）

また，イベント種別がEventOverflowの場合には，以下の関連情報を持つ。

noLostEvents	UInt32	失われたイベントの数

5.13.8. データ型の定義

5.13.8.1. CurrentLocationの機能クラス

```
type CurrentLocationCapabilityType = enum {
    AltitudeDetectable,
    HeadingDetectable,
    AccuracyDetectable,
    ErrorEllipseDetectable,
    CartesianPoseProvidable
};
```

5.13.8.2. CurrentLocationに対するリスククラス

```
type CurrentLocationRiskType = enum {
    RiskPrivacy
};
```

5.13.8.3. ローカル直交座標系

```
type CurrentLocationAxisConventionType = enum {
    ENU,    // x軸を東・y軸を北・z軸を上とする軸のとり方
    NED // x軸を北・y軸を東・z軸を下とする軸のとり方
};
```

```
type CurrentLocationLocalFrameType = struct {
    frameId : String,   // ローカル直交座標系の識別子
    originLatitude : Float64 ([-90.0,90.0]),    // 原点の緯度。単位はdeg。北緯が正
    originLongitude : Float64 ([-180.0,180.0]), // 原点の経度。単位はdeg。東経が正
    originAltitude : Float64,   // 原点の高度。単位はm（WGS84楕円体高）
    rotation : Float64, // x軸の方位角に加える追加回転角。単位はdeg。axisConventionが定めるx軸の方位角にrotationを加えた値を，x軸の方位角とする。方位角は真北を0とし，東回り（上空から見て時計回り）に測る。0のときaxisConventionが定める軸のとり方そのものとする
    scale : Float64 ((0.0,+INF)),   // 地理座標系に対する水平方向のスケール。無次元。水平方向の地上距離[m]は，x・yの座標値の差にscaleを乗じて得られる。zには適用しない
    validRadius : Float64 ((0.0,+INF)), // この座標系が有効な，原点からの半径。単位はm。scaleを適用した水平方向の地上距離で判定する
```

```
    conversionTolerance : Float64 ((0.0,+INF)), // このローカル直交座標系と地理座標系
との相互変換において許容される残差。単位はm
    axisConvention : CurrentLocationAxisConventionType  // 座標軸のとり方
};
```

localFrameは，ローカル直交座標系の原点・軸のとり方・追加回転・スケール・有効範囲・変換の許容残差を宣言する。この宣言により，ローカル直交座標系を解釈しない利用側でも，cartesianPoseを地理座標へ変換できる。地理座標系との変換は，原点における局所水平面（接平面）による近似とする。ローカル直交座標系は水平とみなし，回転は水平面内の1自由度のみとする。ローカル直交座標系は，水平面内の直交座標と，原点の高度からの楕円体高の差により位置を表す。

scaleは，水平方向の地上距離[m]＝x・yの座標値の差×scaleとして適用する。scaleが1.0のとき，x・yの座標値の単位はmと一致する。zにはscaleを適用せず，その単位はmとする。

frameIdは，原点・軸のとり方・追加回転・スケール・有効範囲のいずれかが異なる座標系に対して，異なる値でなければならない。conversionToleranceは座標系の幾何を定めないため，一意性の判定には含めない。

cartesianPoseが表す位置は，原点からの水平方向の地上距離がvalidRadius以内でなければならない。validRadiusは，接平面近似による水平位置の誤差がconversionToleranceを超えない範囲で宣言しなければならない。利用側は，conversionToleranceが自らの要求に対して十分でないと判断する場合，cartesianPoseを用いず，地理座標系による位置を用いることができる。

車両基準点の位置が原点からvalidRadiusを超えた場合，cartesianPoseは値なしとなる。車両側実装が別のローカル直交座標系へ切り替える場合は，構成情報localFrameを更新し，ConfigChangedイベントを発生させる。切り替え後のcartesianPoseは，更新後のlocalFrameに基づき，そのframeIdは更新後のlocalFrameのframeIdと一致する。車両側実装は，validRadiusに達する前に座標系を切り替えることが望ましい。緯度・経度・高度・進行方位は，validRadiusの影響を受けず常に提供される。

5.13.8.4. CurrentLocationの構成情報（getConfigが返すデータ型）

```
type CurrentLocationConfigType = struct {
    capabilities : enumSet(CurrentLocationCapabilityType),
    riskClasses : enumSet(CurrentLocationRiskType),
    geodeticDatum : String,
    localFrame : CurrentLocationLocalFrameType?,
    referencePoint : String,
    statusReportInterval : UInt32 ([1,4294967295])?,
    timestampAccuracy : UInt32
};
```

5.13.8.5. タイムスタンプ

```
type TimestampUsType = UInt64;  // 車内共通時刻基盤の起点を基準とする測定時刻。単位は
μs。複数のビークルAPIで共通に用いる共通データ型であり，本仕様内に定義する。
```

5.13.8.6. サンプル品質の種別

```
type DataQualifierType = enum {
    Normal, // 正常
    Unavailable,    // 利用不可
    Degraded,   // 劣化
    TestMode,   // テストモード
    Invalid // 無効（値も誤差も読んではならない）
};
```

5.13.8.7. 水平位置の推定誤差

```
type CurrentLocationEllipseOrientationType = Float64 ([0.0,180.0)); // 誤差楕円の長軸が真北となす角。単位はdeg。時計回り。長軸の向きは180degを周期として同一であるため，値がとりうる範囲は[0,180)とする。
```

```
type CurrentLocationErrorEllipseType = struct {
    majorAxis : Float64 ([0.0,+INF)),   // 誤差楕円の半長軸。単位はm。信頼水準は95%
    minorAxis : Float64 ([0.0,+INF)),   // 誤差楕円の半短軸。単位はm。信頼水準は95%
    orientation : CurrentLocationEllipseOrientationType // 誤差楕円の長軸の方位
};
```

誤差楕円は，水平位置の推定誤差の分布を，半長軸majorAxis・半短軸minorAxisと長軸の方位orientationで表す。majorAxisはminorAxis以上でなければならない。誤差楕円とhorizontalAccuracyは，同一の推定誤差分布から導かれなければならない。

5.13.8.8. ローカル直交座標系における位置

```
type CurrentLocationLocalHeadingType = Float64 ([0.0,360.0));   // ローカル直交座標系のx軸から測った進行方位。単位はdeg。x軸からy軸へ向かう向きを正とする。
```

ローカル直交座標系における進行方位は，座標軸を基準とする方位であり，状態headingが表す真北基準の方位角とは基準および回転の向きが異なる。両者は，構成情報localFrameのaxisConventionおよびrotationにより，次の式で相互に変換する。ここで方位角は真北を0とし東回り（上空から見て時計回り）に測るものとし，結果は0以上360未満に正規化する。

axisConventionがENUの場合，方位角 ＝ 90 ＋ rotation − ローカル方位。axisConventionがNEDの場合，方位角 ＝ rotation ＋ ローカル方位。

```
type CurrentLocationCartesianPoseType = struct {
    frameId : String,   // この位置が基づくローカル直交座標系の識別子。構成情報localFrameのframeIdに対応する
    x : Float64,    // ローカル直交座標系のx座標。水平方向の地上距離[m]への換算は構成情報localFrameのscaleによる
    y : Float64,    // ローカル直交座標系のy座標。水平方向の地上距離[m]への換算は構成情報localFrameのscaleによる
```

```
    z : Float64?,   // ローカル直交座標系のz座標。単位はm。各点における楕円体高と構成情報localFrameのoriginAltitudeとの差を表す
    heading : CurrentLocationLocalHeadingType?  // ローカル直交座標系における進行方位
};
```

ローカル直交座標系における位置は，位置x・y・zと進行方位headingの4自由度で表す。姿勢（ロール角・ピッチ角）は表さない。zは機能クラスAltitudeDetectableを備えている場合にのみ，headingは機能クラスHeadingDetectableを備え，かつ方位を推定できている場合にのみ値を持つ。

frameIdは，この位置が基づくローカル直交座標系を表す。利用側は，frameIdが，保持している構成情報localFrameのframeIdと一致しない場合，当該の位置を使用してはならない。この場合，利用側は構成情報を再取得しなければならない。

x・yは水平面内の座標であり，水平方向の地上距離への換算にはscaleを適用する。zはscaleを適用せず，各点における楕円体高とoriginAltitudeとの差を単位mで表す（鉛直軸は各点の局所鉛直に従う）。

x・y・zが表す位置は，地理座標系における位置と同一の車両基準点の位置である。構成情報localFrameが宣言する原点・軸のとり方・追加回転・スケールによる変換によって，状態が示す緯度・経度・高度と同一の位置を表さなければならない。

この一致は，次の基準によって判定する。原点における局所水平面（接平面）近似によって地理座標へ変換して得た水平位置と，状態が示す緯度・経度が表す水平位置との差が，構成情報localFrameのconversionToleranceを超えないことをもって判定する。高度については，zと状態のaltitudeがともに値を持つ場合に，変換して得た高度と状態のaltitudeとの差が，conversionToleranceを超えないことをもって判定する。

5.13.8.9. CurrentLocation状態

```
type CurrentLocationStatusType = struct {
    timestamp : TimestampUsType,
    dataQualifier : DataQualifierType,
    latitude : Float64 ([-90.0,90.0]),
    longitude : Float64 ([-180.0,180.0]),
    altitude : Float64?,
    heading : Float64 ([0.0,360.0))?,
    horizontalAccuracy : Float64 ([0.0,+INF))?,
    verticalAccuracy : Float64 ([0.0,+INF))?,
    headingAccuracy : Float64 ([0.0,+INF))?,
    horizontalErrorEllipse : CurrentLocationErrorEllipseType?,
    cartesianPose : CurrentLocationCartesianPoseType?
};
```

地理座標系はWGS84とし，その実現版は構成情報geodeticDatumで宣言される。高度は楕円体高とし，進行方位は真北を0とする時計回りの角度とする。cartesianPoseは，構成情報localFrameが定義するローカル直交座標系における位置を表し，地理座標系における位置と対応していなければならない。各推定誤差が表す信頼水準は，95%に固定する。timestampは，車内共通時刻基盤に基づく測定時刻を表す。ただし，dataQualifierがUnavailableのサンプルのtimestampは，サンプルの生成時刻を表す（測定時刻ではない）。dataQualifierがInvalidまたはUnavailableの場合，位置（cartesianPoseを含む）・方位および推定誤差（horizontalErrorEllipseを含む）のいずれも読んではならない。

5.13.8.10. CurrentLocationイベント

```
type CurrentLocationEventKindType = enum {
    FaultDetected,
    FaultRecovered,
    StatusReported,
    EventOverflow,
    ConfigChanged
};
```

```
type CurrentLocationEventType = struct {
    eventInfo : variant(CurrentLocationEventKindType) {
        StatusReported {
            timestamp : TimestampMsType ,
            status : CurrentLocationStatusType
        },
        EventOverflow {
            noLostEvents : UInt32
        }
    }
};
```

5.13.9. サービスコールの詳細規定

getConfig

CurrentLocationの構成情報の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
config	CurrentLocationConfigType	CurrentLocationの構成情報

【エラーコード】

なし

【機能】

CurrentLocationに関する構成情報を返す。

getStatus

現在位置の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
status	CurrentLocationStatusType	車両の現在位置（自己位置のベスト推定値と推定誤差）

【エラーコード】

E_RISK_APPLICATION	アプリケーションが対応していないリスクがある
E_RISK_USER	ユーザがリスクを承認していない
E_OBJECT_FAULT	故障中（自己位置を推定できない状態）

【機能】

車両の現在位置に関するすべての状態を返す。返される値は自己位置のベスト推定値であり，精度は保証されない。取得した値の信頼度は，併せて返されるdataQualifierと推定誤差によって判断する。dataQualifierがInvalidまたはUnavailableの場合，位置（cartesianPoseを含む）・方位および推定誤差（horizontalErrorEllipseを含む）のいずれも読んではならない。

位置情報は個人の行動を推定できる高感度な情報であるため，本サービスコールはリスククラスRiskPrivacyの対象であり，RiskPrivacyに対応していないアプリケーションはE_RISK_APPLICATIONエラー，ユーザが状態の開示を承認していない場合はE_RISK_USERエラーとなる。

notify

CurrentLocationイベントの通知要求

【パラメータ】

eventFilter	enumSet (CurrentLocationEventKindType)?	通知するイベントの種類の集合

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【エラーコード】

E_RISK_APPLICATION	アプリケーションが対応していないリスクがある

E_RISK_USER	ユーザがリスクを承認していない
E_INVALID_PARAMETER	eventFilterが有効範囲外

【機能】

CurrentLocationで発生したeventFilterで指定したイベントの通知を要求する。イベントを通知するためのイベントキューを生成し，その通知ハンドルをnotifyHandleに返す。

eventFilterを省略した場合，すべての種類のイベントの通知を要求する。

通知されるイベントには，報告時点の状態の全量を運ぶStatusReportedイベントが含まれることがあり，また，FaultDetected等のイベントの発生パターンは走行環境（トンネル等）と相関しうる。このため，本サービスコールの呼び出しそのものが，eventFilterの内容によらず（eventFilterを省略した場合を含む），リスククラスRiskPrivacyの対象である。RiskPrivacyに対応していないアプリケーションはE_RISK_APPLICATIONエラー，ユーザが状態の開示を承認していない場合はE_RISK_USERエラーとなる。

unnotify

CurrentLocationイベントの通知停止

【パラメータ】

notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正

【機能】

notifyHandleで指定した通知を停止する。指定した通知用のイベントキューは削除される。

getEvent

CurrentLocationイベントの取得

【パラメータ】

notifyHandle	NotifyHandleType	イベントキューを識別するための通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
event	CurrentLocationEventType	取得したイベント

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正
E_NO_DATA	イベントキューが空

【機能】

notifyHandleで指定した通知用のイベントキューから，先頭のイベントを取り出す。取り出したイベントは，イベントキューから削除される。



イベントがパラメータ等で渡されてくるプログラミング環境向けの物理APIでは，用意する必要がない。

5.14. Vehicle.Driver（略称：Driver）

5.14.1. 位置付けと機能

Vehicle.Driver（略称：Driver）は，運転者に関する観測・状態推定・意図推定の機能を集約する入口となるシングルインスタンスオブジェクトである。個々の観測値・推定値は，下位のオブジェクト（物理的観測・生理量計測・認知状態推定・意図推定等）が提供し，本オブジェクトは，それらが提供する項目のメタデータ（種類・件数・データ型・座標系・品質モデル・更新周期・プライバシ分類等）の集約と，現在利用可能な観測・推定の可用性のスナップショットを提供する。

5.14.1.1. Driverの位置付けと性質

Driverは，運転者に関する機能の全体像をアプリケーションに提示する集約オブジェクトである。運転者の観測値・推定値そのもの（頭部姿勢・視線・心拍・疲労度・意図等の実値）は保持せず，それらを提供する下位のオブジェクトへの案内と，機能の一覧・可用性を提供する。したがって，本オブジェクトは測定時刻・サンプル品質・座標系といった実値に伴う語彙を持たない。実値およびその時刻・品質・座標系は，各項目を提供する下位のオブジェクトから取得する。

対象は現在の運転者1名である。運転者が誰であるかの同定や，個人のプロファイルの内容は本オブジェクトでは扱わない。生体認証の生テンプレート等の生データは，アプリケーションに公開しない。

5.14.1.2. 下位オブジェクト

運転者に関する構成・観測値・推定値は，次の下位オブジェクトが提供する。

- Vehicle.Driver.Device（略称：DriverDevice）：運転者の観測・計測に用いるセンサ装置の静的な構成（装置の種別・取付位置・取付姿勢，およびそれらの座標系と基準点）
- Vehicle.Driver.IdentityProfile（略称：IdentityProfile）：現在の運転者の同定情報，推定された基本属性（性別・年齢等），およびパーソナライズ用プロファイルへの参照
- Vehicle.Driver.PreferenceProfile（略称：PreferenceProfile）：経路選択・運転に対する長期的な嗜好（経路選択の重み・運転スタイルの嗜好）と選択の履歴
- Vehicle.Driver.PhysicalObservation（略称：PhysicalObservation）：頭部の位置・姿勢，目の状態と視線，口の状態と発話，手の位置とステアリング把持状態，胴体姿勢，足の位置とペダル操作状況，および体格の物理的観測
- Vehicle.Driver.PhysiologicalMeasurement（略称：PhysiologicalMeasurement）：心拍数，心拍変動およびRR間隔の生体計測値
- Vehicle.Driver.CognitiveState（略称：CognitiveState）：疲労・眠気・注意・注意散漫・感情・パ

ニック・応答性の認知状態の推定値

- Vehicle.Driver.IntentAndCompliance.Violation（略称：IntentAndComplianceViolation）：現在および次の運転意図と，法規違反の状態の推定値

DriverDevice以外の下位オブジェクトが提供する項目は，本オブジェクトの構成情報itemsに，提供元の下位オブジェクトの完全修飾名（providingObject）とともに列挙される。DriverDeviceは装置の静的な構成を提供するものであり，観測・推定の項目は持たない。

5.14.1.3. 構成情報と可用性

構成情報（getConfig）は，運転者に関して提供される観測信号・状態推定・意図推定・パーソナライズプロファイルの件数と，各項目の静的なメタデータ（提供元オブジェクト・分類・データ型・単位・座標系・品質モデル・動作範囲と前提条件・センシング方式と測定原理・更新周期と遅延・プライバシ分類とアクセス制御の有無）の集約を返す。これはアプリケーションが，どの項目がどの下位オブジェクトから，どの品質・周期で得られるかを一度に把握するためのものである。品質モデル・動作範囲・プライバシ分類・アクセス制御の有無は，アプリケーションが機械的に判定できる型で表される。

状態（getStatus）は，現在利用可能な観測・推定の可用性のスナップショットを返す。呼び出しのたびに，その時点の可用性の全体を返す冪等な応答であり，前回呼び出しからの差分や，変更された項目の通知は含まない。項目の可用性は，運転者の不在・センサの一時的な遮蔽・下位機能の故障等により時々刻々変化しうる。

状態は，要求応答（getStatus）により取得できるほか，状態報告イベント（StatusReported）によって周期的に得ることができる。可用性の変化をきっかけに発生するイベントは定義せず，可用性は，StatusReportedイベントの関連情報statusにより周期的に得られる。

5.14.1.4. アクセス制御

本オブジェクトが返す機能メタデータおよび可用性は，車両が運転者について何を観測・推定できるかを表す情報であり，それ自体が運転者のプライバシに関わる。そのため，構成情報を参照するサービスコール（getConfig）および状態を参照するサービスコール（getStatus）はアクセス制御の対象であり，アクセスが認められていないアプリケーションに対してはE_DENIED_ACCESSエラーとなる。イベントの通知要求（notify）により通知されるイベントには，報告時点の可用性の全量を運ぶStatusReportedイベントが含まれることがあり，また，イベントの発生の観測からも構成・可用性の変化を推定できる。このため，notifyの呼び出しそのものを，eventFilterで指定するイベントの種類によらず，アクセス制御の対象とする。

安全性およびプライバシに関連する項目は，ポリシによって制御される。個々の観測値・推定値へのアクセス制御は，各下位オブジェクトが規定する。

5.14.2. リスククラス

Driverは，車両の状態を変化させるサービスコールを持たないため，リスククラスを定義しない。

機能メタデータ・可用性の参照に伴うプライバシのリスクに対しては，構成情報・状態の参照（getConfig，getStatus）およびイベントの通知要求（notify）へのアクセス制御（E_DENIED_ACCESSエラー）により対応する。

5.14.3. 構成情報

Driverは，以下の構成情報を持つ。

observationSignalCount	UInt32	提供される観測信号の項目の数

stateEstimationCount	UInt32	提供される状態推定の項目の数
intentEstimationCount	UInt32	提供される意図推定の項目の数
personalizationProfileCount	UInt32	提供されるパーソナライズプロファイルの項目の数
statusReportInterval	UInt32 ([1,4294967295])?	状態報告イベント（StatusReported）が発生する周期。単位はms。値を持つ場合は1以上。値なしの場合，この車両ではStatusReportedイベントは発生しない
items	DriverCapabilityItemType[]	提供される各項目の静的なメタデータの配列

5.14.4. 状態

Driverは，以下の状態を持つ。

availableObservationSignalCount	UInt32	現在利用可能な観測信号の項目の数
availableStateEstimationCount	UInt32	現在利用可能な状態推定の項目の数
availableIntentEstimationCount	UInt32	現在利用可能な意図推定の項目の数
itemAvailabilities	DriverItemAvailabilityType[]	各項目の現在の可用性の配列

状態は，呼び出し時点の可用性の全体を返す冪等なスナップショットである。前回呼び出しからの差分や，変更された項目の通知は含まない。itemAvailabilitiesは，構成情報（getConfig）が列挙する項目に対応し，各項目が現時点で値を取得できるか（available）を示す。availableがfalseの項目は，運転者の不在・一時的な遮蔽・下位機能の故障等により，現時点で値を取得できないことを表す。

運転者監視機能の全体を提供できない故障中（FaultDetectedイベントからFaultRecoveredイベントまでの間）は，getStatusが返す状態およびStatusReportedイベントで報告される状態のいずれにおいても，すべての項目のavailableがfalseとなる。

5.14.5. サービスコール一覧

Driverに対するサービスコールは以下の通り。

getConfig	Driverの構成情報の参照
getStatus	観測・推定の可用性の参照
notify	Driverイベントの通知要求
unnotify	Driverイベントの通知停止
getEvent	Driverイベントの取得

5.14.6. イベント

Driverのイベント種別には，運転者監視機能全体の故障の検知・回復，状態の報告（StatusReported），および構成の変更がある。可用性の変化をきっかけに発生するイベントは定義しない。可用性は，StatusReportedイベントの関連情報statusにより周期的に得られる。

Driverに対するイベント種別は以下の通り。

FaultDetected	故障検知（運転者監視機能の全体を提供できない状態）
FaultRecovered	故障から回復
StatusReported	状態の報告（構成情報statusReportIntervalの周期ごとに発生する）
EventOverflow	イベントキューのオーバフロー
ConfigChanged	構成情報の変更（提供される項目の増減・メタデータの変更）

StatusReportedイベントの関連情報statusは，報告時点の観測・推定の可用性のスナップショットの全量であり，getStatusが返す状態と同じ型（DriverStatusType）である。個々の観測値・推定値そのものは含まない。

この内，イベント種別がStatusReportedの場合には，以下の関連情報を持つ。

timestamp	TimestampMsType	報告が発生した時刻
status	DriverStatusType	報告時点の観測・推定の可用性（全量）

また，イベント種別がEventOverflowの場合には，以下の関連情報を持つ。

noLostEvents	UInt32	失われたイベントの数

5.14.7. データ型の定義

5.14.7.1. 項目の分類

```
type DriverItemCategoryType = enum {
    ObservationSignal,  // 観測信号
    StateEstimation,    // 状態推定
    IntentEstimation,   // 意図推定
    PersonalizationProfile  // パーソナライズプロファイル
};
```

5.14.7.2. 項目のデータ型

```
type DriverItemDataKindType = enum {
    Boolean,    // 真偽値
    Integer,    // 整数
    Real,   // 実数
    String, // 文字列
    Enumeration,    // 列挙型
```

```
    EnumerationSet, // 列挙型の集合
    Structure,  // 構造体
    Array   // 配列
};
```

項目の値の種類を表す列挙型である。

```
type DriverItemDataFormatType = struct {
    typeName : String,  // 項目のデータ型の表記。提供元の下位オブジェクトが定める型名または型の表記（例：Float64 ([0.0,1.0])，CognitiveStateAttentionCategoryType）
    kind : DriverItemDataKindType   // 項目の値の種類
};
```

項目のデータ型を表す型である。typeNameにより提供元の下位オブジェクトが定める型を特定でき，kindにより値の種類を機械的に判定できる。

5.14.7.3. 項目の品質モデル

```
type DriverItemQualityModelType = enum {
    Uncertainty,    // 不確かさ（推定誤差の実値）が対になる。信頼水準は95%（2σ相当）
    ConfidenceLevel,    // 信頼度が対になる。Float64の0.0以上1.0以下の値で表し，1.0が最大信頼
    NormalizedUncertainty   // 正規化した不確かさが対になる。単位を持たず，Float64の0.0以上1.0以下の値で表し，0.0が最も確か，1.0が最も不確か
};
```

項目の値に対になって提供される品質指標の種類を表す列挙型である。不確かさ（Uncertainty）は，項目と同じ単位で表される推定誤差であり，真値が推定値±不確かさの範囲に入る確率が95%となる値である。信頼度（ConfidenceLevel）は，推定器が自己申告する確からしさであり，精度の保証ではない。正規化した不確かさ（NormalizedUncertainty）は，推定器が自己申告する推定の不確かさを，単位を持たない0.0以上1.0以下の値で表したものであり，精度の保証ではない。項目と同じ単位の推定誤差ではなく，信頼度とは値の向きが逆である。

1つの項目が，品質指標の種類の異なる複数の成分からなる場合（例：位置の成分には不確かさが，離散状態の判定の成分には正規化した不確かさが対になる場合）には，品質モデルは該当するすべての種類の集合となる。

5.14.7.4. 項目の動作範囲

```
type DriverItemOperatingRangeType = struct {
    parameterName : String, // 範囲を定める量の名称。項目自身の値の範囲を表す場合は項目の名称
    unit : String?, // 範囲を定める量の単位。無次元の量では値なし
    minimum : Float64?, // 範囲の下限（下限を含む）。下限の定めがない場合は値なし
    maximum : Float64?  // 範囲の上限（上限を含む）。上限の定めがない場合は値なし
};
```

項目が有効となる数値の範囲を1つの量について表す型である。項目は，範囲を定めるすべての量が，それぞれの下限以上かつ上限以下である場合に有効となる。数値の範囲で表せない条件は，前提条件（preconditions）で表す。

5.14.7.5. 項目のプライバシ分類

```
type DriverItemPrivacyClassType = enum {
    Identifying,    // 運転者個人を同定できる情報，または同定に結び付く個人属性
    HealthRelated,  // 運転者の健康状態・生理的状態を推定しうる情報
    MentalState,    // 運転者の認知状態・感情状態を推定した情報
    Behavioral, // 運転者の運転意図・運転行動・法規の遵守状況を推定した情報
    Preference, // 運転者の嗜好
    BodyObservation // 運転者の身体の外形的な観測（姿勢・視線・体格等）
};
```

項目がどの種類の個人情報に該当するかを表す列挙型である。1つの項目が複数の分類に該当する場合がある。

5.14.7.6. 機能メタデータの項目

```
type DriverCapabilityItemType = struct {
    parameterName : String, // 項目の名称
    providingObject : String,   // 項目を提供する下位オブジェクトの完全修飾名
    itemCategory : DriverItemCategoryType,  // 項目の分類
    dataType : DriverItemDataFormatType,    // 項目のデータ型
    unit : String?, // 項目の単位。無次元の項目では値なし
    coordinateFrame : String?,  // 項目の座標系。座標を持たない項目では値なし
    qualityModel : enumSet(DriverItemQualityModelType), // 項目の値に対になって提供される品質指標の種類の集合。品質指標が対にならない項目では空の集合
    operatingRanges : DriverItemOperatingRangeType[],   // 項目が有効となる数値の範囲の配列。定めがない場合は空の配列
    preconditions : String?,    // 項目が有効となる前提条件のうち，数値の範囲で表せないものの説明。定めがある場合にのみ値を持つ
    sensingModality : String?,  // 項目のセンシング方式と測定原理。定めがある場合にのみ値を持つ
    updatePeriod : UInt32?, // 項目の更新周期。単位はms。周期更新される項目でのみ値を持つ
    latency : UInt32?,  // 項目の提供遅延の代表値。単位はms。定めがある場合にのみ値を持つ
    privacyClassification : enumSet(DriverItemPrivacyClassType),    // 項目が該当するプライバシ分類の集合
    accessControlled : Bool // 項目の値の参照がアクセス制御の対象であるか
};
```

運転者に関して提供される1項目の静的なメタデータを表す型である。実値は本型には含まれず，providingObjectが指す下位オブジェクトから取得する。

アクセス制御は提供元の下位オブジェクトの単位で規定されるため，providingObjectが同じ項目は，accessControlledが同じ値をとる。accessControlledがtrueの項目では，アクセスが認められていないア

プリケーションに対して，提供元の下位オブジェクトの状態の参照（getStatus）およびイベントの通知要求（notify）がE_DENIED_ACCESSエラーとなりうる。

5.14.7.7. 項目の可用性

```
type DriverItemAvailabilityType = struct {
    parameterName : String, // 項目の名称
    providingObject : String,   // 項目を提供する下位オブジェクトの完全修飾名
    itemCategory : DriverItemCategoryType,  // 項目の分類
    available : Bool,   // 現時点で値を取得できるか
    currentUpdatePeriod : UInt32?   // 現在の更新周期。単位はms。周期更新される項目が利用可能な場合にのみ値を持つ
};
```

運転者に関して提供される1項目の現在の可用性を表す型である。availableがfalseの項目は，現時点で値を取得できないことを表す。

5.14.7.8. Driverの構成情報（getConfigが返すデータ型）

```
type DriverConfigType = struct {
    observationSignalCount : UInt32,
    stateEstimationCount : UInt32,
    intentEstimationCount : UInt32,
    personalizationProfileCount : UInt32,
    statusReportInterval : UInt32 ([1,4294967295])?,
    items : DriverCapabilityItemType[]
};
```

5.14.7.9. Driver状態

```
type DriverStatusType = struct {
    availableObservationSignalCount : UInt32,
    availableStateEstimationCount : UInt32,
    availableIntentEstimationCount : UInt32,
    itemAvailabilities : DriverItemAvailabilityType[]
};
```

現在利用可能な観測・推定の可用性の冪等なスナップショットである。itemAvailabilitiesは構成情報（getConfig）が列挙する項目に対応する。

5.14.7.10. Driverイベント

```
type DriverEventKindType = enum {
    FaultDetected,
    FaultRecovered,
    StatusReported,
    EventOverflow,
```

```
    ConfigChanged
};
```

```
type DriverEventType = struct {
    eventInfo : variant(DriverEventKindType) {
        StatusReported {
            timestamp : TimestampMsType ,
            status : DriverStatusType
        },
        EventOverflow {
            noLostEvents : UInt32
        }
    }
};
```

5.14.8. サービスコールの詳細規定

getConfig

Driverの構成情報の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
config	DriverConfigType	運転者に関して提供される項目の機能メタデータの集約

【エラーコード】

E_DENIED_ACCESS	機能メタデータの参照が認められていない

【機能】

運転者に関して提供される観測信号・状態推定・意図推定・パーソナライズプロファイルの件数と，各項目の静的な機能メタデータの集約を返す。生体認証の生テンプレート等の生データは公開しない。安全性およびプライバシに関連する項目は，ポリシによって制御される。

機能メタデータは，車両が運転者について何を観測・推定できるかを表す情報であるため，本サービスコールはアクセス制御の対象である。

getStatus

観測・推定の可用性の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
status	DriverStatusType	現在利用可能な観測・推定の可用性のスナップショット

【エラーコード】

E_DENIED_ACCESS	可用性の参照が認められていない

【機能】

現在利用可能な観測・推定の可用性のスナップショットを返す。呼び出しのたびに，その時点の可用性の全体を返す冪等な応答であり，前回呼び出しからの差分や，変更された項目の通知は含まない。個々の観測値・推定値そのものは返さず，各項目を提供する下位オブジェクトから取得する。

可用性は，車両が運転者について現に何を観測・推定できているかを表す情報であるため，本サービスコールはアクセス制御の対象である。

notify

Driverイベントの通知要求

【パラメータ】

eventFilter	enumSet (DriverEventKindType)?	通知するイベントの種類の集合

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【エラーコード】

E_DENIED_ACCESS	イベントの通知要求が認められていない
E_INVALID_PARAMETER	eventFilterが有効範囲外

【機能】

Driverで発生したeventFilterで指定したイベントの通知を要求する。イベントを通知するためのイベントキューを生成し，その通知ハンドルをnotifyHandleに返す。

eventFilterを省略した場合，すべての種類のイベントの通知を要求する。

通知されるイベントには，報告時点の可用性の全量を運ぶStatusReportedイベントが含まれることがあり，また，イベントの発生の観測からも構成・可用性の変化を推定できる。このため，本サービスコールの呼び出しそのものが，eventFilterの内容によらず（eventFilterを省略した場合を含む），アクセス制御の対象であり，アクセスが認められていないアプリケーションに対してはE_DENIED_ACCESSエラーとなる。

unnotify

Driverイベントの通知停止

【パラメータ】

notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正

【機能】

notifyHandleで指定した通知を停止する。指定した通知用のイベントキューは削除される。

getEvent

Driverイベントの取得

【パラメータ】

notifyHandle	NotifyHandleType	イベントキューを識別するための通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
event	DriverEventType	取得したイベント

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正
E_NO_DATA	イベントキューが空

【機能】

notifyHandleで指定した通知用のイベントキューから，先頭のイベントを取り出す。取り出したイベントは，イベントキューから削除される。



イベントがパラメータ等で渡されてくるプログラミング環境向けの物理APIでは，用意する必要がない。

5.15. Vehicle.Driver.Device（略称：DriverDevice）

5.15.1. 位置付けと機能

Vehicle.Driver.Device（略称：DriverDevice）は，運転者の観測・計測に用いるセンサ装置の静的な構成を提供するシングルインスタンスオブジェクトである。装置の数と，各装置の識別子・種別・主装置か否か・取付位置・取付姿勢，および取付位置の座標系と基準点を，構成情報として提供する。

5.15.1.1. DriverDeviceの位置付けと性質

DriverDeviceは，運転者を観測・計測する装置がどこにどのように取り付けられているかを，アプリケーションに提示する構成オブジェクトである。提供するのは静的な構成情報のみであり，装置の稼働状態・計測値・健全性といった時々刻々変化する状態は保持しない。運転者に関する観測値・推定値は，各観測・計測を行う下位のオブジェクトから取得する。

対象は現在の車両に取り付けられた運転者用センサ装置である。運転者が誰であるかの同定や，個人のプロファイルは扱わない。

5.15.1.2. 座標系と基準点

各装置の取付位置・取付姿勢が表す座標系は，ISO　8855の車両座標系（原点：後軸中心の路面投影点，X軸：車両前方，Y軸：車両左方，Z軸：上方の右手系）である。座標系と基準点は，構成情報coordinateSystemおよびreferencePointで宣言される。独自の基準点は用いない。

取付姿勢角（ロール・ピッチ・ヨー）の単位はdegであり，回転の正方向は右手系に従う。すなわち，ヨーはZ軸まわりの回転角（車両前方を0とし，左向きが正），ピッチはY軸まわりの回転角（前傾（下向き）が正），ロールはX軸まわりの回転角（右傾が正）である。取付位置の単位はmである。

5.15.1.3. アクセス制御

センサ装置の構成（種別・取付位置等）は，車両が運転者について何を，どこから観測・計測できるかを表す情報であり，それ自体が運転者のプライバシに関わる。そのため，構成情報を参照するサービスコール（getConfig）はアクセス制御の対象であり，アクセスが認められていないアプリケーションに対してはE_DENIED_ACCESSエラーとなる。イベントの発生の観測からも構成の変化を推定できるため，イベントの通知要求（notify）も同様にアクセス制御の対象である。



本オブジェクトは静的な構成情報のみを提供し，時々刻々変化する状態を持たないため，状態を参照するサービスコール（getStatus）を持たない。構成の変化はConfigChangedイベントにより取得する。

5.15.2. リスククラス

DriverDeviceは，車両の状態を変化させるサービスコールを持たないため，リスククラスを定義しない。

センサ装置の構成の参照に伴うプライバシのリスクに対しては，参照系サービスコールおよびイベントの通知要求（notify）へのアクセス制御（E_DENIED_ACCESSエラー）により対応する。

5.15.3. 構成情報

DriverDeviceは，以下の構成情報を持つ。

deviceCount	UInt32	運転者の観測・計測に用いるセンサ装置の数
devices	DriverDeviceInfoType[]	各センサ装置の構成の配列

coordinateSystem	String	取付位置・取付姿勢の座標系の宣言（ISO 8855の車両座標系）
referencePoint	String	取付位置の基準点の宣言（ISO 8855の車両座標系原点：後軸中心の路面投影点）

5.15.4. サービスコール一覧

DriverDeviceに対するサービスコールは以下の通り。

getConfig	DriverDeviceの構成情報の参照
notify	DriverDeviceイベントの通知要求
unnotify	DriverDeviceイベントの通知停止
getEvent	DriverDeviceイベントの取得

5.15.5. イベント

DriverDeviceのイベント種別は，構成の変更のみである。本オブジェクトは静的な構成情報のみを提供し，装置の稼働状態・計測値を保持しないため，故障の検知・回復や値の更新に関するイベントは定義しない。装置の増設・撤去・取付位置の変更等の構成の変化は，ConfigChangedイベントで通知される。

DriverDeviceに対するイベント種別は以下の通り。

EventOverflow	イベントキューのオーバフロー
ConfigChanged	構成情報の変更（装置の増減・取付位置等の変更）

この内，イベント種別がEventOverflowの場合には，以下の関連情報を持つ。

noLostEvents	UInt32	失われたイベントの数

5.15.6. データ型の定義

5.15.6.1. 装置の種別

```
type DriverDeviceKindType = enum {
    Camera, // カメラ
    NirCamera,  // 近赤外線カメラ
    FirCamera,  // 遠赤外線カメラ
    Microphone, // マイク
    WeightSensor,   // 体重センサ
    TouchSensor,    // タッチセンサ
    HapticsSensor,  // 触覚センサ
    Wearable,   // ウェアラブルデバイス（生体計測を行う装着型装置）
    PpgSensor,  // 光電式脈波センサ（PPG）
    Unknown // 上記のいずれにも当てはまらない装置
};
```

運転者の観測・計測に用いるセンサ装置の種別を表す型である。

5.15.6.2. 取付位置

```
type DriverDeviceMountingPositionType = struct {
    x : Float64,    // X座標（車両前方が正）。単位はm
    y : Float64,    // Y座標（車両左方が正）。単位はm
    z : Float64 // Z座標（上方が正）。単位はm
};
```

装置の取付位置を表す型である。座標系はISO 8855の車両座標系（原点：後軸中心の路面投影点）とし，構成情報coordinateSystemおよびreferencePointで宣言される。

5.15.6.3. 取付姿勢

```
type DriverDeviceMountingOrientationType = struct {
    roll : Float64, // ロール角（X軸まわり，右傾が正）。単位はdeg
    pitch : Float64,    // ピッチ角（Y軸まわり，前傾（下向き）が正）。単位はdeg
    yaw : Float64   // ヨー角（Z軸まわり，車両前方を0とし，左向きが正）。単位はdeg
};
```

装置の取付姿勢を表す型である。回転の正方向はISO 8855の車両座標系の右手系に従う。

5.15.6.4. センサ装置の構成

```
type DriverDeviceInfoType = struct {
    deviceId : IdType,  // 装置の識別子
    deviceType : DriverDeviceKindType,  // 装置の種別
    isPrimary : Bool,   // 同種の装置のうち主として用いる装置か
    mountingPosition : DriverDeviceMountingPositionType,    // 装置の取付位置
    mountingOrientation : DriverDeviceMountingOrientationType   // 装置の取付姿勢
};
```

1台のセンサ装置の静的な構成を表す型である。

5.15.6.5. DriverDeviceの構成情報（getConfigが返すデータ型）

```
type DriverDeviceConfigType = struct {
    deviceCount : UInt32,
    devices : DriverDeviceInfoType[],
    coordinateSystem : String,
    referencePoint : String
};
```

5.15.6.6. DriverDeviceイベント

```
type DriverDeviceEventKindType = enum {
```

```
    EventOverflow,
    ConfigChanged
};
```

```
type DriverDeviceEventType = struct {
    eventInfo : variant(DriverDeviceEventKindType) {
        EventOverflow {
            noLostEvents : UInt32
        }
    }
};
```

5.15.7. サービスコールの詳細規定

getConfig

DriverDeviceの構成情報の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
config	DriverDeviceConfigType	運転者用センサ装置の静的な構成

【エラーコード】

E_DENIED_ACCESS	センサ装置の構成の参照が認められていない

【機能】

運転者の観測・計測に用いるセンサ装置の数と，各装置の識別子・種別・主装置か否か・取付位置・取付姿勢，および取付位置の座標系と基準点を返す。返す値は静的な構成情報であり，装置の稼働状態・計測値は含まない。

センサ装置の構成は，車両が運転者について何を，どこから観測・計測できるかを表す情報であるため，本サービスコールはアクセス制御の対象である。

notify

DriverDeviceイベントの通知要求

【パラメータ】

eventFilter	enumSet (DriverDeviceEventKindType)?	通知するイベントの種類の集合

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【エラーコード】

E_DENIED_ACCESS	イベントの通知要求が認められていない
E_INVALID_PARAMETER	eventFilterが有効範囲外

【機能】

DriverDeviceで発生したeventFilterで指定したイベントの通知を要求する。イベントを通知するためのイベントキューを生成し，その通知ハンドルをnotifyHandleに返す。

eventFilterを省略した場合，すべての種類のイベントの通知を要求する。

イベントの発生の観測からも構成の変化を推定できるため，イベントの通知要求もアクセス制御の対象であり，アクセスが認められていないアプリケーションに対してはE_DENIED_ACCESSエラーとなる。

unnotify

DriverDeviceイベントの通知停止

【パラメータ】

notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正

【機能】

notifyHandleで指定した通知を停止する。指定した通知用のイベントキューは削除される。

getEvent

DriverDeviceイベントの取得

【パラメータ】

notifyHandle	NotifyHandleType	イベントキューを識別するための通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

event	DriverDeviceEventType	取得したイベント

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正
E_NO_DATA	イベントキューが空

【機能】

notifyHandleで指定した通知用のイベントキューから，先頭のイベントを取り出す。取り出したイベントは，イベントキューから削除される。



イベントがパラメータ等で渡されてくるプログラミング環境向けの物理APIでは，用意する必要がない。

5.16. Vehicle.Driver.IdentityProfile（略称：IdentityProfile）

5.16.1. 位置付けと機能

Vehicle.Driver.IdentityProfile（略称：IdentityProfile）は，現在の運転者の同定情報と，運転者に結び付いた準静的な個人プロファイルを提供するシングルインスタンスオブジェクトである。認証に基づく運転者の同定情報（IDトークンの各属性），推定された基本属性（性別・年齢等），および各種パーソナライズ用プロファイルへの参照（識別子）と，それぞれの推定の信頼度を状態として提供する。あわせて，プロファイルの最終更新時刻timestampとサンプル品質dataQualifierを提供する。

5.16.1.1. IdentityProfileの位置付けと性質

IdentityProfileが扱う情報は，運転者個人を特定できる最も機微な個人情報である。本オブジェクトは，運転者の同定に用いた生体テンプレート（顔・声・虹彩等の生の特徴量）を公開しない。保護対象の個人属性は，値を直接公開せず，パーソナライズ用のプロファイルへの参照（プロファイル識別子）によって間接的に扱う。座席・室内快適・経路嗜好・メディア・運転スタイル・特性の各プロファイルは，本オブジェクトが識別子のみを保持し，属性の実体は各プロファイルを管理する主体が扱う。

運転者の同定情報（identifier，identifierSubject，identifierIssuer）は，OAuth 2.0に基づく確定的な同定情報である。一方，性別・年齢・運転免許状態・言語嗜好等は推定値を含みうるものであり，確定値（グラウンドトゥルース）ではない。各推定値の判定基準・信頼度の算出モデルは車両実装に依存し，本APIでは規定しない。アプリケーションは，推定に基づく値を確定的な事実として扱ってはならず，併せて返される信頼度とサンプル品質により値の確からしさを判断するものとする。

IdentityProfileの対象は，現在運転席にいる運転者1名である。

5.16.1.2. 信頼度の表現

各推定値には，推定の信頼度（authenticationConfidenceLevel等の*ConfidenceLevelフィールド）が対になる。すべての信頼度の値域は[0.0,1.0]に統一し，1.0が最大信頼を表す。信頼度は精度の保証ではなく，推定器または認証器が自己申告する確からしさである。

5.16.1.3. 最終更新時刻とサンプル品質

状態には，プロファイルの最終更新時刻timestampとサンプル品質dataQualifierが含まれる。timestampは，車内共通時刻基盤に基づく，プロファイルが最後に更新・推定された時刻であり，送信時刻ではな

い。同じ時刻基盤に基づく他のビークルAPIのタイムスタンプと直接比較できる。タイムスタンプ自体の誤差上限は，構成情報timestampAccuracyで宣言される。IdentityProfileは準静的な情報であり，高頻度に連続更新されるものではない。

dataQualifierは，取得できたプロファイルの品質を表す。故障（同定・推定の機能を提供できない状態。E_OBJECT_FAULTエラーおよびFaultDetectedイベント）が「プロファイルを出力できない状態」を表すのに対し，dataQualifierは「出力されたプロファイルがどれだけ信用できるか」を表す。両者の使い分けの詳細は，状態の節に規定する。

5.16.1.4. IdentityProfileの機能

運転者の同定情報（identifier，identifierSubject，identifierIssuer）は，すべてのIdentityProfileが提供する基本の情報であり，機能クラスに依存しない。authenticationConfidenceLevelは，認証の信頼度を数値として算出できる場合にのみ値を持つ。

推定された基本属性（性別・年齢），運転免許状態，言語嗜好，アクセシビリティプロファイル，および各種パーソナライズ用プロファイルへの参照は，対応する機能クラスによって宣言され，当該機能クラスを備えている場合にのみ値を持つ。段階的な機能開示により，同定情報のみを提供する構成から，多数のプロファイル参照を提供する構成まで，同一の状態型で表現する。

5.16.1.5. 状態の更新とアクセス制御

IdentityProfileは準静的な情報であるため，状態報告イベント（StatusReported）は持たない。状態の取得には要求応答（getStatus）を用いる。プロファイルの更新は，離散的なイベント（ProfileUpdated）によって通知する。イベントを受けたアプリケーションは，getStatusにより最新の状態を取得する。

IdentityProfileが扱う情報は，運転者個人を特定できる最も機微な個人情報である。そのため，状態を参照するサービスコール（getStatus）はアクセス制御の対象であり，アクセスが認められていないアプリケーションに対してはE_DENIED_ACCESSエラーとなる。イベントの発生の観測からも運転者の交代等を推定できるため，イベントの通知要求（notify）も同様にアクセス制御の対象である。

5.16.2. 機能クラス

IdentityProfileは，以下の機能クラスを持つ。

DemographicEstimation	運転者の基本属性（性別・年齢）を推定することができる
DrivingLicenseEstimation	運転免許の状態を推定することができる
LanguagePreferenceAvailable	運転者の言語嗜好を提供することができる
AccessibilityProfileAvailable	HMI提示のためのアクセシビリティプロファイルを提供することができる
UserProfileReference	IVIパーソナライズ用のユーザプロファイルへの参照を提供することができる
SeatMemoryProfileReference	座席メモリプロファイルへの参照を提供することができる
CabinComfortProfileReference	室内快適プロファイルへの参照を提供することができる
RoutePreferenceProfileReference	経路嗜好プロファイルへの参照を提供することができる

MediaPreferenceProfileReference	メディア嗜好プロファイルへの参照を提供することができる
DrivingStyleProfileReference	運転スタイルプロファイルへの参照を提供することができる
TraitProfileReference	特性プロファイルへの参照を提供することができる

5.16.3. リスククラス

IdentityProfileは，車両の状態を変化させるサービスコールを持たないため，リスククラスを定義しない。

運転者の同定情報・個人プロファイルの参照に伴うプライバシのリスクに対しては，参照系サービスコールおよびイベントの通知要求（notify）へのアクセス制御（E_DENIED_ACCESSエラー）により対応する。生体テンプレートの非公開と，保護対象個人属性のプロファイル参照による間接化により，属性の実体の露出を抑える。

5.16.4. 構成情報

IdentityProfileは，以下の構成情報を持つ。

capabilities	enumSet (IdentityProfileCapabilityType)	そのIdentityProfileが持つ機能（推定・参照のうち提供できるもの）
timestampAccuracy	UInt32	この車両のタイムスタンプ誤差上限の宣言。単位はμs

5.16.5. 状態

IdentityProfileは，以下の状態を持つ。

timestamp	TimestampUsType	プロファイルの最終更新時刻。単位はμs。車内共通時刻基盤に基づく
dataQualifier	DataQualifierType	このプロファイルの品質。InvalidまたはUnavailableの場合，同定情報・推定値・プロファイル参照および信頼度のいずれも読んではならない
identifier	IdType	運転者の同定情報。OAuth 2.0に基づく認証のIDトークンを識別する情報
identifierSubject	String	認証の対象（Subject）。OAuth 2.0の主体識別子（例：利用者ID）
identifierIssuer	String	認証の発行者（Issuer）。OAuth 2.0の発行者識別子（例：発行者のURI）
authenticationConfidenceLevel	Float64 ([0.0,1.0])?	運転者の同定の信頼度。1.0が最大信頼。数値としての信頼度を算出できる場合にのみ値を持つ
sex	IdentityProfileSexType?	申告または推定された性別。機能クラスDemographicEstimationを備えている場合にのみ値を持つ

sexConfidenceLevel	Float64 ([0.0,1.0])?	性別推定の信頼度。1.0が最大信頼
ageEstimate	UInt8?	推定された運転者の年齢。単位は歳。機能クラスDemographicEstimationを備えている場合にのみ値を持つ
ageUncertainty	Float64?	年齢推定の不確かさ。単位は歳。信頼水準は95%（2σ相当）。数値としての不確かさを算出できる場合にのみ値を持つ
drivingLicenseStatus	IdentityProfileDrivingLicenseStatusType?	運転免許の状態。機能クラスDrivingLicenseEstimationを備えている場合にのみ値を持つ
drivingLicenseConfidenceLevel	Float64 ([0.0,1.0])?	運転免許状態推定の信頼度。1.0が最大信頼
userProfileId	IdType?	IVIパーソナライズ用のユーザプロファイルの識別子。機能クラスUserProfileReferenceを備えている場合にのみ値を持つ
userProfileConfidenceLevel	Float64 ([0.0,1.0])?	ユーザプロファイル同定の信頼度。1.0が最大信頼
languagePreference	String?	運転者の言語嗜好。言語タグ（例：ja，en-US）で表す。機能クラスLanguagePreferenceAvailableを備えている場合にのみ値を持つ
languagePreferenceConfidenceLevel	Float64 ([0.0,1.0])?	言語嗜好推定の信頼度。1.0が最大信頼
accessibilityProfile	String?	HMI提示のためのアクセシビリティプロファイル。実装が定める表現の識別子または記述。機能クラスAccessibilityProfileAvailableを備えている場合にのみ値を持つ
accessibilityProfileConfidenceLevel	Float64 ([0.0,1.0])?	アクセシビリティプロファイル推定の信頼度。1.0が最大信頼
seatMemoryProfileId	IdType?	座席メモリプロファイルの識別子。機能クラスSeatMemoryProfileReferenceを備えている場合にのみ値を持つ
seatMemoryProfileConfidenceLevel	Float64 ([0.0,1.0])?	座席メモリプロファイル同定の信頼度。1.0が最大信頼
cabinComfortProfileId	IdType?	室内快適プロファイルの識別子。機能クラスCabinComfortProfileReferenceを備えている場合にのみ値を持つ
cabinComfortProfileConfidenceLevel	Float64 ([0.0,1.0])?	室内快適プロファイル同定の信頼度。1.0が最大信頼
routePreferenceProfileId	IdType?	経路嗜好プロファイルの識別子。機能クラスRoutePreferenceProfileReferenceを備えている場合にのみ値を持つ

routePreferenceProfileConfidenceLevel	Float64 ([0.0,1.0])?	経路嗜好プロファイル同定の信頼度。1.0が最大信頼
mediaPreferenceProfileId	IdType?	メディア嗜好プロファイルの識別子。機能クラスMediaPreferenceProfileReferenceを備えている場合にのみ値を持つ
mediaPreferenceProfileConfidenceLevel	Float64 ([0.0,1.0])?	メディア嗜好プロファイル同定の信頼度。1.0が最大信頼
drivingStyleProfileId	IdType?	運転スタイルプロファイルの識別子。機能クラスDrivingStyleProfileReferenceを備えている場合にのみ値を持つ
drivingStyleProfileConfidenceLevel	Float64 ([0.0,1.0])?	運転スタイルプロファイル同定の信頼度。1.0が最大信頼
traitProfileConfidenceLevel	Float64 ([0.0,1.0])?	特性プロファイル同定の信頼度。1.0が最大信頼
traitProfileId	IdType?	特性プロファイルの識別子。機能クラスTraitProfileReferenceを備えている場合にのみ値を持つ

identifier，identifierSubject，identifierIssuerは，すべてのIdentityProfileが提供する基本の同定情報であり，機能クラスに依存しない。これらは，運転者の同定に用いた生体テンプレートを含まない。identifierSubjectおよびidentifierIssuerは，OAuth 2.0のSubjectおよびIssuerを表す文字列である。

authenticationConfidenceLevelは，認証の信頼度を数値として算出できる場合にのみ値を持つ。sex，sexConfidenceLevel，ageEstimate，ageUncertaintyは機能クラスDemographicEstimationを，drivingLicenseStatus，drivingLicenseConfidenceLevelは機能クラスDrivingLicenseEstimationを，languagePreference，languagePreferenceConfidenceLevelは機能クラスLanguagePreferenceAvailableを，accessibilityProfile，accessibilityProfileConfidenceLevelは機能クラスAccessibilityProfileAvailableを備えている場合にのみ値を持つ。各プロファイル参照の識別子（userProfileId，seatMemoryProfileId，cabinComfortProfileId，routePreferenceProfileId，mediaPreferenceProfileId，drivingStyleProfileId，traitProfileId）とその信頼度は，対応する機能クラス（UserProfileReference等）を備えている場合にのみ値を持つ。

保護対象の個人属性は，値を直接公開せず，プロファイル参照の識別子によって間接的に扱う。プロファイルの属性の実体は，各プロファイルを管理する主体が扱う。

すべての信頼度（*ConfidenceLevel）の値域は[0.0,1.0]であり，1.0が最大信頼を表す。ageUncertaintyは不確かさ（誤差）の実値であり，信頼水準は95%（2σ相当）である。

timestampは，このプロファイルが最後に更新・推定された時刻であり，車内共通時刻基盤に基づく。タイムスタンプの誤差上限は，構成情報timestampAccuracyで宣言される。

dataQualifierは，このプロファイルの品質を表す。Normalは，プロファイルが有効な状態である。Invalidは，同定・推定の結果を信用できない状態（起動直後の初期化中・認証に失敗している状態など）であり，同定情報・推定値・プロファイル参照および信頼度のいずれも読んではならない。Degradedは，プロファイルは使えるが一部の要素の確からしさが低下している状態であり，各要素の確からしさは信頼度の実値で判断する。

運転者を同定できていない状態（運転席に運転者がいない場合や，認証が完了していない場合）では，プ

ロファイルを出力できないため，dataQualifierがUnavailableのプロファイルとなる（故障とは扱わない）。同定・推定の機能そのものを提供できない状態（必要なセンサ・認証機構の故障等）は故障と扱い，故障中の状態の参照（getStatus）はE_OBJECT_FAULTエラーとなる。Unavailableのプロファイルでは，同定情報・推定値・プロファイル参照および信頼度のいずれも読んではならない（最終既知値として読むことも許されない）。

5.16.6. サービスコール一覧

IdentityProfileに対するサービスコールは以下の通り。

getConfig	IdentityProfileの構成情報の参照
getStatus	運転者の同定情報とプロファイルの参照
notify	IdentityProfileイベントの通知要求
unnotify	IdentityProfileイベントの通知停止
getEvent	IdentityProfileイベントの取得

5.16.7. イベント

IdentityProfileのイベントとして通知するのは，プロファイルの更新（ProfileUpdated）と，故障の検知・回復である。IdentityProfileは準静的な情報であるため，状態報告イベント（StatusReported）は持たず，更新はProfileUpdatedイベントによって通知する。

イベントの付随情報には最小限の情報のみを含め，状態のデータ本体は含めない。データ本体は，イベントを受けたアプリケーションがgetStatusにより取得する。

IdentityProfileに対するイベント種別は以下の通り。

ProfileUpdated	運転者の同定情報またはプロファイルが更新された（運転者の再同定，推定属性・プロファイル参照の更新等）
FaultDetected	故障検知（同定・推定の機能を提供できない状態）
FaultRecovered	故障から回復
EventOverflow	イベントキューのオーバフロー
ConfigChanged	構成情報の変更

この内，イベント種別がEventOverflowの場合には，以下の関連情報を持つ。

noLostEvents	UInt32	失われたイベントの数

5.16.8. データ型の定義

5.16.8.1. IdentityProfileの機能クラス

```
type IdentityProfileCapabilityType = enum {
    DemographicEstimation,
    DrivingLicenseEstimation,
    LanguagePreferenceAvailable,
```

```
    AccessibilityProfileAvailable,
    UserProfileReference,
    SeatMemoryProfileReference,
    CabinComfortProfileReference,
    RoutePreferenceProfileReference,
    MediaPreferenceProfileReference,
    DrivingStyleProfileReference,
    TraitProfileReference
};
```

5.16.8.2. IdentityProfileの構成情報（getConfigが返すデータ型）

```
type IdentityProfileConfigType = struct {
    capabilities : enumSet(IdentityProfileCapabilityType),
    timestampAccuracy : UInt32
};
```

5.16.8.3. タイムスタンプ

```
type TimestampUsType = UInt64;  // 車内共通時刻基盤の起点を基準とする時刻。単位はμs。
複数のビークルAPIで共通に用いる共通データ型であり，本仕様内に定義する。
```

5.16.8.4. サンプル品質の種別

```
type DataQualifierType = enum {
    Normal, // 正常
    Unavailable,    // 利用不可
    Degraded,   // 劣化
    TestMode,   // テストモード
    Invalid // 無効（値も信頼度も読んではならない）
};
```

5.16.8.5. 性別の分類

```
type IdentityProfileSexType = enum {
    Male,   // 男性
    Female, // 女性
    Other,  // その他
    Unknown // 分類を確定できない
};
```

5.16.8.6. 運転免許の状態

```
type IdentityProfileDrivingLicenseStatusType = enum {
```

```
    Valid,  // 有効
    Provisional,    // 仮免許・暫定的な免許
    Suspended,  // 停止中
    Expired,    // 失効
    Revoked,    // 取消
    None,   // 免許なし
    Unknown // 状態を確定できない
};
```

5.16.8.7. IdentityProfile状態

```
type IdentityProfileStatusType = struct {
    timestamp : TimestampUsType,
    dataQualifier : DataQualifierType,
    identifier : IdType,
    identifierSubject : String,
    identifierIssuer : String,
    authenticationConfidenceLevel : Float64 ([0.0,1.0])?,
    sex : IdentityProfileSexType?,
    sexConfidenceLevel : Float64 ([0.0,1.0])?,
    ageEstimate : UInt8?,
    ageUncertainty : Float64?,
    drivingLicenseStatus : IdentityProfileDrivingLicenseStatusType?,
    drivingLicenseConfidenceLevel : Float64 ([0.0,1.0])?,
    userProfileId : IdType?,
    userProfileConfidenceLevel : Float64 ([0.0,1.0])?,
    languagePreference : String?,
    languagePreferenceConfidenceLevel : Float64 ([0.0,1.0])?,
    accessibilityProfile : String?,
    accessibilityProfileConfidenceLevel : Float64 ([0.0,1.0])?,
    seatMemoryProfileId : IdType?,
    seatMemoryProfileConfidenceLevel : Float64 ([0.0,1.0])?,
    cabinComfortProfileId : IdType?,
    cabinComfortProfileConfidenceLevel : Float64 ([0.0,1.0])?,
    routePreferenceProfileId : IdType?,
    routePreferenceProfileConfidenceLevel : Float64 ([0.0,1.0])?,
    mediaPreferenceProfileId : IdType?,
    mediaPreferenceProfileConfidenceLevel : Float64 ([0.0,1.0])?,
    drivingStyleProfileId : IdType?,
    drivingStyleProfileConfidenceLevel : Float64 ([0.0,1.0])?,
    traitProfileConfidenceLevel : Float64 ([0.0,1.0])?,
    traitProfileId : IdType?
};
```

identifier，identifierSubject，identifierIssuerは，すべてのIdentityProfileが提供する基本の同定情報であり，生体テンプレートを含まない。推定された基本属性・運転免許状態・言語嗜好・アクセシビリティプロファイル・各種プロファイル参照は，対応する機能クラスを備えている場合にのみ値を持つ。すべての信頼度（*ConfidenceLevel）の値域は[0.0,1.0]であり，1.0が最大信頼を表す。timestampは，車内共通時刻基盤に基づくプロファイルの最終更新時刻を表す。dataQualifierがInvalidまたはUnavailableの場合，同定情報・推定値・プロファイル参照および信頼度のいずれも読んではならない。

5.16.8.8. IdentityProfileイベント

```
type IdentityProfileEventKindType = enum {
    ProfileUpdated,
    FaultDetected,
    FaultRecovered,
    EventOverflow,
    ConfigChanged
};
```

```
type IdentityProfileEventType = struct {
    eventInfo : variant(IdentityProfileEventKindType) {
        EventOverflow {
            noLostEvents : UInt32
        }
    }
};
```

5.16.9. サービスコールの詳細規定

getConfig

IdentityProfileの構成情報の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
config	IdentityProfileConfigType	IdentityProfileの構成情報

【エラーコード】

なし

【機能】

IdentityProfileに関する構成情報を返す。

getStatus

運転者の同定情報とプロファイルの参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
status	IdentityProfileStatusType	運転者の同定情報とプロファイル（推定値・参照と信頼度・品質・時刻）

【エラーコード】

E_DENIED_ACCESS	同定情報・プロファイルの参照が認められていない
E_OBJECT_FAULT	故障中（同定・推定の機能を提供できない状態）

【機能】

運転者の同定情報とプロファイルに関するすべての状態を返す。推定に基づく値は確定値（グラウンドトゥルース）ではなく，取得した値の確からしさは，併せて返されるdataQualifierと各信頼度によって判断する。dataQualifierがInvalidまたはUnavailableの場合，同定情報・推定値・プロファイル参照および信頼度のいずれも読んではならない。返される情報には，運転者の同定に用いた生体テンプレートは含まれない。

運転者の同定情報・個人プロファイルは運転者個人を特定できる最も機微な個人情報であるため，本サービスコールはアクセス制御の対象である。

notify

IdentityProfileイベントの通知要求

【パラメータ】

eventFilter	enumSet (IdentityProfileEventKindType)?	通知するイベントの種類の集合

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【エラーコード】

E_DENIED_ACCESS	イベントの通知要求が認められていない
E_INVALID_PARAMETER	eventFilterが有効範囲外

【機能】

IdentityProfileで発生したeventFilterで指定したイベントの通知を要求する。イベントを通知するためのイベントキューを生成し，その通知ハンドルをnotifyHandleに返す。

eventFilterを省略した場合，すべての種類のイベントの通知を要求する。

イベントの発生の観測からも運転者の交代等を推定できるため，イベントの通知要求もアクセス制御の対象であり，アクセスが認められていないアプリケーションに対してはE_DENIED_ACCESSエラーとなる。

unnotify

IdentityProfileイベントの通知停止

【パラメータ】

notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正

【機能】

notifyHandleで指定した通知を停止する。指定した通知用のイベントキューは削除される。

getEvent

IdentityProfileイベントの取得

【パラメータ】

notifyHandle	NotifyHandleType	イベントキューを識別するための通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
event	IdentityProfileEventType	取得したイベント

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正
E_NO_DATA	イベントキューが空

【機能】

notifyHandleで指定した通知用のイベントキューから，先頭のイベントを取り出す。取り出したイベントは，イベントキューから削除される。



イベントがパラメータ等で渡されてくるプログラミング環境向けの物理APIでは，用意する必要がない。

5.17. Vehicle.Driver.PreferenceProfile（略称：PreferenceProfile）

5.17.1. 位置付けと機能

Vehicle.Driver.PreferenceProfile（略称：PreferenceProfile）は，現在の運転者の経路選択・運転に対する長期的な嗜好を提供するシングルインスタンスオブジェクトである。経路選択の重み（安全・所要時間・快適・エネルギー効率・料金）と運転スタイルの嗜好，およびそれぞれの推定の信頼度と選択の履歴を状態として提供する。あわせて，プロファイルの最終更新時刻timestampとサンプル品質dataQualifierを提供する。

5.17.1.1. PreferenceProfileの位置付けと性質

PreferenceProfileが扱う嗜好は，運転者の反復的な行動から経時的に学習される長期的なパーソナライズ情報であり，確定値（グラウンドトゥルース）ではない。各嗜好の学習・推定のモデル，重みの正規化の方法，信頼度の算出モデルは車両実装に依存し，本APIでは規定しない。アプリケーションは，取得した値を確定的な事実として扱ってはならず，併せて返される信頼度とサンプル品質により値の確からしさを判断するものとする。

PreferenceProfileの対象は，現在運転席にいる運転者1名である。運転者の同定情報や各種プロファイルへの参照はVehicle.Driver.IdentityProfileが扱い，PreferenceProfileは嗜好の内容を扱う。

5.17.1.2. 信頼度の表現

各嗜好には，推定の信頼度（routePreferenceWeightsConfidenceLevel等の*ConfidenceLevelフィールド）が対になる。すべての信頼度の値域は[0.0,1.0]に統一し，1.0が最大信頼を表す。信頼度は精度の保証ではなく，推定器が自己申告する確からしさである。

5.17.1.3. 最終更新時刻とサンプル品質

状態には，プロファイルの最終更新時刻timestampとサンプル品質dataQualifierが含まれる。timestampは，車内共通時刻基盤に基づく，プロファイルが最後に更新・学習された時刻であり，送信時刻ではない。同じ時刻基盤に基づく他のビークルAPIのタイムスタンプと直接比較できる。タイムスタンプ自体の誤差上限は，構成情報timestampAccuracyで宣言される。PreferenceProfileは準静的な情報であり，高頻度に連続更新されるものではない。

dataQualifierは，取得できたプロファイルの品質を表す。故障（嗜好の学習・推定の機能を提供できない状態。E_OBJECT_FAULTエラーおよびFaultDetectedイベント）が「プロファイルを出力できない状態」を表すのに対し，dataQualifierは「出力されたプロファイルがどれだけ信用できるか」を表す。両者の使い分けの詳細は，状態の節に規定する。

5.17.1.4. PreferenceProfileの機能

経路選択の嗜好（routePreferenceWeights，routePreferenceWeightsConfidenceLevel，routeSelectionHistory）は機能クラスRoutePreferenceAvailableによって，運転スタイルの嗜好（drivingStylePreference，drivingStylePreferenceConfidenceLevel，drivingStylePreferenceHistory）は機能クラスDrivingStylePreferenceAvailableによって宣言され，対応する機能クラスを備えている場合にのみ値を持つ。

5.17.1.5. 状態の更新とアクセス制御

PreferenceProfileは準静的な情報であるため，状態報告イベント（StatusReported）は持たない。状態

の取得には要求応答（getStatus）を用いる。プロファイルの更新は，離散的なイベント（ProfileUpdated）によって通知する。イベントを受けたアプリケーションは，getStatusにより最新の状態を取得する。

PreferenceProfileが扱う嗜好は，運転者の行動から学習される高感度な個人情報である。そのため，状態を参照するサービスコール（getStatus）はアクセス制御の対象であり，アクセスが認められていないアプリケーションに対してはE_DENIED_ACCESSエラーとなる。イベントの通知要求（notify）も同様にアクセス制御の対象である。

5.17.2. 機能クラス

PreferenceProfileは，以下の機能クラスを持つ。

RoutePreferenceAvailable	経路選択の嗜好（重みと選択履歴）を提供することができる
DrivingStylePreferenceAvailable	運転スタイルの嗜好（分類と履歴）を提供することができる

5.17.3. リスククラス

PreferenceProfileは，車両の状態を変化させるサービスコールを持たないため，リスククラスを定義しない。

長期的な嗜好の参照に伴うプライバシのリスクに対しては，参照系サービスコールおよびイベントの通知要求（notify）へのアクセス制御（E_DENIED_ACCESSエラー）により対応する。

5.17.4. 構成情報

PreferenceProfileは，以下の構成情報を持つ。

capabilities	enumSet (PreferenceProfileCapabilityType)	そのPreferenceProfileが持つ機能（経路選択・運転スタイルのうち提供できるもの）
timestampAccuracy	UInt32	この車両のタイムスタンプ誤差上限の宣言。単位はμs

5.17.5. 状態

PreferenceProfileは，以下の状態を持つ。

timestamp	TimestampUsType	プロファイルの最終更新時刻。単位はμs。車内共通時刻基盤に基づく
dataQualifier	DataQualifierType	このプロファイルの品質。InvalidまたはUnavailableの場合，嗜好の推定値および信頼度のいずれも読んではならない
routePreferenceWeights	PreferenceProfileRouteWeightsType?	経路選択における重要要素の重み。機能クラスRoutePreferenceAvailableを備えている場合にのみ値を持つ

routePreferenceWeightsConfidenceLevel	Float64 ([0.0,1.0])?	経路選択の重み推定の信頼度。1.0が最大信頼
routeSelectionHistory	String?	経路選択の履歴。実装が定める形式で符号化された，過去の経路選択の記録。機能クラスRoutePreferenceAvailableを備えている場合にのみ値を持つ
drivingStylePreference	PreferenceProfileDrivingStyleType?	運転スタイルの嗜好の分類。機能クラスDrivingStylePreferenceAvailableを備えている場合にのみ値を持つ
drivingStylePreferenceConfidenceLevel	Float64 ([0.0,1.0])?	運転スタイルの嗜好推定の信頼度。1.0が最大信頼
drivingStylePreferenceHistory	String?	運転スタイルの嗜好の履歴。実装が定める形式で符号化された，過去の運転スタイルの嗜好の記録。機能クラスDrivingStylePreferenceAvailableを備えている場合にのみ値を持つ

routePreferenceWeights，routePreferenceWeightsConfidenceLevel，routeSelectionHistoryは機能クラスRoutePreferenceAvailableを，drivingStylePreference，drivingStylePreferenceConfidenceLevel，drivingStylePreferenceHistoryは機能クラスDrivingStylePreferenceAvailableを備えている場合にのみ値を持つ。

routePreferenceWeightsは，経路選択における安全・所要時間・快適・エネルギー効率・料金の各要素の重みからなる構造体である。各重みの値域は[0.0,1.0]であり，値が大きいほどその要素を重視することを表す。重みの正規化の方法は車両実装に依存する。

routeSelectionHistoryおよびdrivingStylePreferenceHistoryは，学習の基となる過去の記録であり，実装が定める形式で符号化された文字列として運ぶ。履歴の形式は本APIでは規定しない。

すべての信頼度（*ConfidenceLevel）の値域は[0.0,1.0]であり，1.0が最大信頼を表す。

timestampは，このプロファイルが最後に更新・学習された時刻であり，車内共通時刻基盤に基づく。タイムスタンプの誤差上限は，構成情報timestampAccuracyで宣言される。

dataQualifierは，このプロファイルの品質を表す。Normalは，プロファイルが有効な状態である。Invalidは，学習・推定の結果を信用できない状態（起動直後の初期化中・十分な学習が行われていない状態など）であり，嗜好の推定値および信頼度のいずれも読んではならない。Degradedは，プロファイルは使えるが一部の要素の確からしさが低下している状態であり，各要素の確からしさは信頼度の実値で判断する。

運転者を同定できていない状態（運転席に運転者がいない場合や，嗜好を対応付ける運転者が定まっていない場合）では，プロファイルを出力できないため，dataQualifierがUnavailableのプロファイルとなる（故障とは扱わない）。学習・推定の機能そのものを提供できない状態（必要な機構の故障等）は故障と扱い，故障中の状態の参照（getStatus）はE_OBJECT_FAULTエラーとなる。Unavailableのプロファイルでは，嗜好の推定値および信頼度のいずれも読んではならない（最終既知値として読むことも許されない）。

5.17.6. サービスコール一覧

PreferenceProfileに対するサービスコールは以下の通り。

getConfig	PreferenceProfileの構成情報の参照
getStatus	運転者の嗜好プロファイルの参照
notify	PreferenceProfileイベントの通知要求
unnotify	PreferenceProfileイベントの通知停止
getEvent	PreferenceProfileイベントの取得

5.17.7. イベント

PreferenceProfileのイベントとして通知するのは，プロファイルの更新（ProfileUpdated）と，故障の検知・回復である。PreferenceProfileは準静的な情報であるため，状態報告イベント（StatusReported）は持たず，更新はProfileUpdatedイベントによって通知する。

イベントの付随情報には最小限の情報のみを含め，状態のデータ本体は含めない。データ本体は，イベントを受けたアプリケーションがgetStatusにより取得する。

PreferenceProfileに対するイベント種別は以下の通り。

ProfileUpdated	嗜好のプロファイルが更新された（学習による嗜好・信頼度・履歴の更新等）
FaultDetected	故障検知（嗜好の学習・推定の機能を提供できない状態）
FaultRecovered	故障から回復
EventOverflow	イベントキューのオーバフロー
ConfigChanged	構成情報の変更

この内，イベント種別がEventOverflowの場合には，以下の関連情報を持つ。

noLostEvents	UInt32	失われたイベントの数

5.17.8. データ型の定義

5.17.8.1. PreferenceProfileの機能クラス

```
type PreferenceProfileCapabilityType = enum {
    RoutePreferenceAvailable,
    DrivingStylePreferenceAvailable
};
```

5.17.8.2. PreferenceProfileの構成情報（getConfigが返すデータ型）

```
type PreferenceProfileConfigType = struct {
    capabilities : enumSet(PreferenceProfileCapabilityType),
    timestampAccuracy : UInt32
};
```

5.17.8.3. タイムスタンプ

```
type TimestampUsType = UInt64;  // 車内共通時刻基盤の起点を基準とする時刻。単位はμs。
複数のビークルAPIで共通に用いる共通データ型であり，本仕様内に定義する。
```

5.17.8.4. サンプル品質の種別

```
type DataQualifierType = enum {
    Normal, // 正常
    Unavailable,    // 利用不可
    Degraded,   // 劣化
    TestMode,   // テストモード
    Invalid // 無効（値も信頼度も読んではならない）
};
```

5.17.8.5. 経路選択の重み

```
type PreferenceProfileRouteWeightType = Float64 ([0.0,1.0]);
```

経路選択における1つの要素の重みを表す型である。値域は[0.0,1.0]であり，1.0が最大を表す。

```
type PreferenceProfileRouteWeightsType = struct {
    safety : PreferenceProfileRouteWeightType,  // 安全を重視する度合いの重み。1.0が最
大
    travelTime : PreferenceProfileRouteWeightType,  // 所要時間の短さを重視する度合い
の重み。1.0が最大
    comfort : PreferenceProfileRouteWeightType, // 快適さを重視する度合いの重み。1.0が
最大
    energyEfficiency : PreferenceProfileRouteWeightType,    // エネルギー効率を重視す
る度合いの重み。1.0が最大
    tollCost : PreferenceProfileRouteWeightType // 料金の低さを重視する度合いの重み。
1.0が最大
};
```

経路選択における各要素の重みを表す型である。各重みは，値が大きいほどその要素を重視することを表す。重みの正規化の方法は車両実装に依存する。

5.17.8.6. 運転スタイルの嗜好の分類

```
type PreferenceProfileDrivingStyleType = enum {
    Smooth, // 滑らかな運転を好む
    Sporty, // スポーティな運転を好む
    Defensive,  // 防御的な運転を好む
    Efficient,  // 効率的な運転を好む
    Unknown // 分類を確定できない
```

```
};
```

5.17.8.7. PreferenceProfile状態

```
type PreferenceProfileStatusType = struct {
    timestamp : TimestampUsType,
    dataQualifier : DataQualifierType,
    routePreferenceWeights : PreferenceProfileRouteWeightsType?,
    routePreferenceWeightsConfidenceLevel : Float64 ([0.0,1.0])?,
    routeSelectionHistory : String?,
    drivingStylePreference : PreferenceProfileDrivingStyleType?,
    drivingStylePreferenceConfidenceLevel : Float64 ([0.0,1.0])?,
    drivingStylePreferenceHistory : String?
};
```

routePreferenceWeights，routePreferenceWeightsConfidenceLevel，routeSelectionHistoryは機能クラスRoutePreferenceAvailableを，drivingStylePreference，drivingStylePreferenceConfidenceLevel，drivingStylePreferenceHistoryは機能クラスDrivingStylePreferenceAvailableを備えている場合にのみ値を持つ。すべての信頼度（*ConfidenceLevel）の値域は[0.0,1.0]であり，1.0が最大信頼を表す。timestampは，車内共通時刻基盤に基づくプロファイルの最終更新時刻を表す。dataQualifierがInvalidまたはUnavailableの場合，嗜好の推定値および信頼度のいずれも読んではならない。

5.17.8.8. PreferenceProfileイベント

```
type PreferenceProfileEventKindType = enum {
    ProfileUpdated,
    FaultDetected,
    FaultRecovered,
    EventOverflow,
    ConfigChanged
};
```

```
type PreferenceProfileEventType = struct {
    eventInfo : variant(PreferenceProfileEventKindType) {
        EventOverflow {
            noLostEvents : UInt32
        }
    }
};
```

5.17.9. サービスコールの詳細規定

getConfig

PreferenceProfileの構成情報の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
config	PreferenceProfileConfigType	PreferenceProfileの構成情報

【エラーコード】

なし

【機能】

PreferenceProfileに関する構成情報を返す。

getStatus

運転者の嗜好プロファイルの参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
status	PreferenceProfileStatusType	運転者の長期的な嗜好（推定値と信頼度・履歴・品質・時刻）

【エラーコード】

E_DENIED_ACCESS	嗜好プロファイルの参照が認められていない
E_OBJECT_FAULT	故障中（嗜好の学習・推定の機能を提供できない状態）

【機能】

運転者の長期的な嗜好に関するすべての状態を返す。返される値は学習・推定に基づく値であり，確定値（グラウンドトゥルース）ではない。取得した値の確からしさは，併せて返されるdataQualifierと各信頼度によって判断する。dataQualifierがInvalidまたはUnavailableの場合，嗜好の推定値および信頼度のいずれも読んではならない。

長期的な嗜好は運転者の行動から学習される高感度な個人情報であるため，本サービスコールはアクセス制御の対象である。

notify

PreferenceProfileイベントの通知要求

【パラメータ】

eventFilter	enumSet (PreferenceProfileEventKindType)?	通知するイベントの種類の集合

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【エラーコード】

E_DENIED_ACCESS	イベントの通知要求が認められていない
E_INVALID_PARAMETER	eventFilterが有効範囲外

【機能】

PreferenceProfileで発生したeventFilterで指定したイベントの通知を要求する。イベントを通知するためのイベントキューを生成し，その通知ハンドルをnotifyHandleに返す。

eventFilterを省略した場合，すべての種類のイベントの通知を要求する。

イベントの通知要求もアクセス制御の対象であり，アクセスが認められていないアプリケーションに対してはE_DENIED_ACCESSエラーとなる。

unnotify

PreferenceProfileイベントの通知停止

【パラメータ】

notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正

【機能】

notifyHandleで指定した通知を停止する。指定した通知用のイベントキューは削除される。

getEvent

PreferenceProfileイベントの取得

【パラメータ】

notifyHandle	NotifyHandleType	イベントキューを識別するための通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
event	PreferenceProfileEventType	取得したイベント

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正
E_NO_DATA	イベントキューが空

【機能】

notifyHandleで指定した通知用のイベントキューから，先頭のイベントを取り出す。取り出したイベントは，イベントキューから削除される。



イベントがパラメータ等で渡されてくるプログラミング環境向けの物理APIでは，用意する必要がない。

5.18. Vehicle.Driver.PhysicalObservation（略称：PhysicalObservation）

5.18.1. 位置付けと機能

Vehicle.Driver.PhysicalObservation（略称：PhysicalObservation）は，現在の運転者の物理的観測を提供するシングルインスタンスオブジェクトである。頭部の位置・姿勢と顔バウンディングボックス，目の状態と視線，アイウェアの装用状態，口の状態と発話，手の位置，ステアリング把持状態，胴体姿勢，足の位置とペダル操作状況，および体格（身長・体重）の推定値を状態として提供する。各観測値には不確かさまたは信頼度が対になって提供され，状態には測定時刻とサンプル品質が含まれる。

5.18.1.1. PhysicalObservationの位置付けと性質

PhysicalObservationの値は，車室内のセンサによる観測に基づく推定値であり，確定値（グラウンドトゥルース）ではない。観測手段（カメラ・近赤外線カメラ・タッチセンサ等）とその構成は規定しない。観測手段や環境（照度・遮蔽・運転者の姿勢等）による精度の違いは，各不確かさ・信頼度フィールドの実値として表現され，APIとしての精度の保証は行わない。

対象は現在の運転者1名である。運転者が誰であるかの同定や，個人のプロファイルは扱わない。身長・体重の推定値は，個人の登録情報ではなく，観測からの推定として扱う。

視線に関する用語は，ISO 15007の語彙に整合させる。

5.18.1.2. 座標系と基準点

位置・姿勢が表す座標系は，ISO　8855の車両座標系（原点：後軸中心の路面投影点，X軸：車両前方，Y軸：車両左方，Z軸：上方の右手系）である。座標系と基準点は，構成情報coordinateSystemおよびreferencePointで宣言される。独自の基準点は用いない。

姿勢角（ヨー・ピッチ・ロール）の単位はdegであり，回転の正方向は右手系に従う。すなわち，ヨーはZ軸まわりの回転角（車両前方を0とし，左向きが正），ピッチはY軸まわりの回転角（前傾（下向き）が正），ロールはX軸まわりの回転角（右傾が正）である。

5.18.1.3. 不確かさと信頼度の表現

連続量の観測値には不確かさ（〜Uncertainty）が，離散状態の観測値には信頼度（〜ConfidenceLevel）が対になる。

推定誤差を表す不確かさ（観測値と同じ単位で表す不確かさ）の信頼水準は，95%（2σ相当）に固定する。すなわち，真値が推定値±不確かさの範囲に入る確率が95%となる。1σ等の他の信頼水準で誤差を算出する実装では，車両側実装が95%相当へ換算して返す。これは精度の保証ではなく，誤差申告の意味の統一である。不確かさを算出できない実装や瞬間には，不確かさの当該成分は値なしとなる。

信頼度は，判定された離散状態に対する信頼の度合いをFloat64の[0.0,1.0]で表し，1.0が最大信頼である。

一方，複合量に含まれる離散成分（発話の有無・ペダル操作状況等）の判定には，対応するUncertainty struct内で正規化した不確かさが対になる。正規化した不確かさは，推定器が自己申告する判定の不確かさを，単位を持たないFloat64の0.0以上1.0以下の値で表したものであり，0.0が最も確か，1.0が最も不確かである。正規化した不確かさは推定誤差ではなく，信頼水準を持たない。また，独立した観測値に対する信頼度（1.0が最大信頼）とは値の向きが逆であることに注意する。

5.18.1.4. 測定時刻とサンプル品質

状態には，観測の測定時刻timestampとサンプル品質dataQualifierが含まれる。timestampは，車内共通時刻基盤に基づく測定時刻（センサ観測のエポック）であり，送信時刻ではない。同じ時刻基盤に基づく他のビークルAPIのタイムスタンプと直接比較できる。タイムスタンプ自体の誤差上限は，構成情報timestampAccuracyで宣言される。

dataQualifierは，取得できたサンプルの品質を表す。故障（運転者の物理的観測を提供できない状態。E_OBJECT_FAULTエラーおよびFaultDetectedイベント）が「観測値を出力できない状態」を表すのに対し，dataQualifierは「出力されたサンプルがどれだけ信用できるか」を表す。両者の使い分けの詳細は，状態の節に規定する。

5.18.1.5. PhysicalObservationの機能

頭部の位置・姿勢と顔バウンディングボックス（headPoseAndFaceBox）およびステアリング把持状態（steeringWheelGripState）は，それぞれの不確かさ・信頼度とともに，すべてのPhysicalObservationが提供する基本の観測値である。

目の状態と視線，アイウェアの装用状態，口の状態と発話，手の位置，胴体姿勢，足の位置とペダル操作状況，体格の推定値は，機能クラスによって宣言され，対応する機能クラスを備えている場合にのみ提供される。

5.18.1.6. 状態の更新とアクセス制御

状態の更新周期は車両実装に依存する。状態は，要求応答（getStatus）により取得できるほか，状態報告イベント（StatusReported）によって周期的に得ることができる。頭部姿勢・視線等の観測値は連続更新される信号系データであるため，その変化をきっかけに発生するイベントは定義しない。観測値は，StatusReportedイベントの関連情報statusにより周期的に得られる。状態の変化をきっかけに発生するイベントとして通知するのは，ステアリング把持状態の変化（SteeringWheelGripChanged）と，故障の検知・回復のみである。

運転者の物理的観測は，運転者の身体的特徴・挙動・体調を推定できる高感度な情報である。そのため，状態を参照するサービスコール（getStatus）はアクセス制御の対象であり，アクセスが認められていないアプリケーションに対してはE_DENIED_ACCESSエラーとなる。イベントの通知要求（notify）により通知されるイベントには，報告時点の状態の全量を運ぶStatusReportedイベントが含まれることがあり，また，把持状態の変化等のイベントの発生の観測からも運転者の挙動を推定できる。このため，notifyの呼び出しそのものを，eventFilterで指定するイベントの種類によらず，アクセス制御の対象とする。

5.18.2. 機能クラス

PhysicalObservationは，以下の機能クラスを持つ。

EyeStateDetectable	目の状態と視線（eyeStateAndGaze，eyeStateAndGazeUncertainty）を提供することができる
EyewearDetectable	アイウェアの装用状態（eyewearState，eyewearConfidenceLevel）を提供することができる
MouthStateDetectable	口の状態と発話（mouthStateAndSpeechActivity，mouthStateAndSpeechActivityUncertainty）を提供することができる
HandPositionDetectable	手の位置（handPosition，handPositionUncertainty）を提供することができる
TorsoPoseDetectable	胴体姿勢（torsoPose，torsoPoseUncertainty）を提供することができる
LegPositionDetectable	足の位置とペダル操作状況（legPositionAndPedalOperation，legPositionAndPedalOperationUncertainty）を提供することができる
BodySizeEstimable	体格の推定値（heightEstimate，weightEstimate）とその不確かさを提供することができる

5.18.3. リスククラス

PhysicalObservationは，車両の状態を変化させるサービスコールを持たないため，リスククラスを定義しない。

運転者の物理的観測の参照に伴うプライバシのリスクに対しては，状態の参照（getStatus）およびイベントの通知要求（notify）へのアクセス制御（E_DENIED_ACCESSエラー）により対応する。

5.18.4. 構成情報

PhysicalObservationは，以下の構成情報を持つ。

capabilities	enumSet (PhysicalObservationCapabilityType)	そのPhysicalObservationが持つ機能
coordinateSystem	String	位置・姿勢の座標系の宣言（ISO 8855の車両座標系）
referencePoint	String	位置の基準点の宣言（ISO 8855の車両座標系原点：後軸中心の路面投影点）

statusReportInterval	UInt32 ([1,4294967295])?	状態報告イベント（StatusReported）が発生する周期。単位はms。値を持つ場合は1以上。値なしの場合，この車両ではStatusReportedイベントは発生しない
timestampAccuracy	UInt32	この車両のタイムスタンプ誤差上限の宣言。単位はμs

5.18.5. 状態

PhysicalObservationは，以下の状態を持つ。

timestamp	TimestampUsType	測定時刻。単位はμs。車内共通時刻基盤に基づく
dataQualifier	DataQualifierType	このサンプルの品質。InvalidまたはUnavailableの場合，観測値・不確かさ・信頼度のいずれも読んではならない
headPoseAndFaceBox	PhysicalObservationHeadPoseAndFaceBoxType	頭部の位置・姿勢と顔バウンディングボックス
headPoseAndFaceBoxUncertainty	PhysicalObservationHeadPoseAndFaceBoxUncertaintyType	頭部の位置・姿勢と顔バウンディングボックスの成分別の不確かさ
eyeStateAndGaze	PhysicalObservationEyeStateAndGazeType?	目の状態（目位置・瞳位置・瞼開閉度）と視線方向
eyeStateAndGazeUncertainty	PhysicalObservationEyeStateAndGazeUncertaintyType?	目の状態と視線方向の成分別の不確かさ
eyewearState	PhysicalObservationEyewearStateType?	アイウェア（メガネ・サングラス・コンタクトレンズ・眼帯）の装用状態
eyewearConfidenceLevel	Float64 ([0.0,1.0])?	アイウェアの装用状態の信頼度。1.0が最大信頼
mouthStateAndSpeechActivity	PhysicalObservationMouthStateAndSpeechActivityType?	口の状態（口位置・開口度）と発話状況
mouthStateAndSpeechActivityUncertainty	PhysicalObservationMouthStateAndSpeechActivityUncertaintyType?	口の状態と発話状況の成分別の不確かさ
handPosition	PhysicalObservationHandPositionType?	左右の手の位置
handPositionUncertainty	PhysicalObservationHandPositionUncertaintyType?	左右の手の位置の成分別の不確かさ
steeringWheelGripState	PhysicalObservationSteeringWheelGripStateType	ステアリング把持状態

steeringWheelGripStateConfidenceLevel	Float64 ([0.0,1.0])	ステアリング把持状態の信頼度。1.0が最大信頼
torsoPose	PhysicalObservationTorsoPoseType?	胴体姿勢（中心位置と向き）
torsoPoseUncertainty	PhysicalObservationTorsoPoseUncertaintyType?	胴体姿勢の成分別の不確かさ
legPositionAndPedalOperation	PhysicalObservationLegPositionAndPedalOperationType?	左右の足の位置とペダル操作状況
legPositionAndPedalOperationUncertainty	PhysicalObservationLegPositionAndPedalOperationUncertaintyType?	左右の足の位置とペダル操作状況の成分別の不確かさ
heightEstimate	Float64?	身長の推定値。単位はcm
heightUncertainty	Float64?	身長の推定値の不確かさ。単位はcm
weightEstimate	Float64?	体重の推定値。単位はkg
weightUncertainty	Float64?	体重の推定値の不確かさ。単位はkg

eyeStateAndGazeとeyeStateAndGazeUncertaintyは機能クラスEyeStateDetectableを，eyewearStateとeyewearConfidenceLevelは機能クラスEyewearDetectableを，mouthStateAndSpeechActivityとmouthStateAndSpeechActivityUncertaintyは機能クラスMouthStateDetectableを，handPositionとhandPositionUncertaintyは機能クラスHandPositionDetectableを，torsoPoseとtorsoPoseUncertaintyは機能クラスTorsoPoseDetectableを，legPositionAndPedalOperationとlegPositionAndPedalOperationUncertaintyは機能クラスLegPositionDetectableを，heightEstimate・heightUncertainty・weightEstimate・weightUncertaintyは機能クラスBodySizeEstimableを備え，かつ当該観測値を推定できている場合にのみ値を持つ。

不確かさ・信頼度のフィールドは，対応する観測値が値を持ち，かつ不確かさ・信頼度を算出できている場合にのみ値を持つ。不確かさを表す構造体の各成分は，当該成分の不確かさを算出できている場合にのみ値を持つ（headPoseAndFaceBoxUncertaintyは常にフィールドとしては値を持つが，その成分は算出できているものだけが値を持つ）。

不確かさ・正規化した不確かさ・信頼度の意味と値域は，「不確かさと信頼度の表現」に規定するとおりである。

timestampは，このサンプルの観測の測定時刻（センサ観測のエポック）であり，車内共通時刻基盤に基づく。1つのサンプルを構成する観測値は，同一の測定時刻に対応する観測から生成する（複数のセンサを併用する実装では，車両側実装が同一時刻へ整列させる）。タイムスタンプの誤差上限は，構成情報timestampAccuracyで宣言される。

dataQualifierは，このサンプルの品質を表す。Invalidは，推定器自身が自分の出力（不確かさ・信頼度の値を含む）を信用できない状態（起動直後の初期化中など）であり，観測値・不確かさ・信頼度のいずれも読んではならない。Invalidを大きな不確かさの値や低い信頼度で代用してはならない。Degradedは，値は使えるが通常品質でない状態（例：逆光・部分的な遮蔽により観測品質が低下している場合）であり，精度そのものは不確かさ・信頼度の実値で判断する。

運転者を観測できていない状態（運転席に運転者がいない場合や，視野の遮蔽等により一時的に観測できない場合）では，観測値を出力できないため，dataQualifierがUnavailableのサンプルとなる（故障とは扱わない）。観測機能そのものを提供できない状態（センサの故障等）は故障と扱い，故障中の状態の参照（getStatus）はE_OBJECT_FAULTエラーとなる。故障中にStatusReportedイベントで報告される状態は，dataQualifierがUnavailableである。Unavailableのサンプルでは，観測値・不確かさ・信頼度のいずれも読んではならない（最終既知値として読むことも許されない）。Unavailableのサンプルのtimestampは，サンプルの生成時刻を表す（測定時刻ではない）。

5.18.6. サービスコール一覧

PhysicalObservationに対するサービスコールは以下の通り。

getConfig	PhysicalObservationの構成情報の参照
getStatus	運転者の物理的観測の参照
notify	PhysicalObservationイベントの通知要求
unnotify	PhysicalObservationイベントの通知停止
getEvent	PhysicalObservationイベントの取得

5.18.7. イベント

PhysicalObservationのイベント種別には，ステアリング把持状態の変化，故障の検知・回復，および状態の報告（StatusReported）がある。頭部姿勢・視線等の観測値は連続更新される信号系データであるため，その変化をきっかけに発生するイベントは定義しない。観測値は，StatusReportedイベントの関連情報statusにより周期的に得られる。

イベントの関連情報には，遷移後の状態などの最小限の情報のみを含める。状態の全体は，イベントを受けたアプリケーションがgetStatusにより取得する。ただし，StatusReportedイベントは，報告時点の状態の全量（PhysicalObservationStatusType）を関連情報statusとして含む。

SteeringWheelGripChangedイベントは，ステアリング把持状態の判定値が変化したときに発生する。推定の揺らぎによる短時間の判定の往復でイベントが連発しないよう，判定の平滑化（最小保持時間の設定等）は車両側実装の責務とする。

PhysicalObservationに対するイベント種別は以下の通り。

SteeringWheelGripChanged	ステアリング把持状態（steeringWheelGripState）の変化
FaultDetected	故障検知（運転者の物理的観測を提供できない状態）
FaultRecovered	故障から回復
StatusReported	状態の報告（構成情報statusReportIntervalの周期ごとに発生する）
EventOverflow	イベントキューのオーバフロー
ConfigChanged	構成情報の変更

StatusReportedイベントの関連情報statusは，報告時点の運転者の物理的観測の状態の全量であり，getStatusが返す状態と同じ型（PhysicalObservationStatusType）である。関連情報timestamp（TimestampMsType）と，状態が持つtimestamp（TimestampUsType）とは単位が異なり，起点が同一であることも保証されないため，互いに比較することはできない。状態の更新周期と構成情報

statusReportIntervalとは一致するとは限らない。dataQualifierがUnavailableでないサンプルどうしでは，連続する報告の関連情報statusが同じ観測に基づくか否かを，statusのtimestampにより判断できる。dataQualifierがUnavailableのサンプルのtimestampはサンプルの生成時刻を表すため，これにより判断することはできない。

この内，イベント種別がSteeringWheelGripChangedの場合には，以下の関連情報を持つ。

gripState	PhysicalObservationSteeringWheelGripStateType	遷移後の把持状態

また，イベント種別がStatusReportedの場合には，以下の関連情報を持つ。

timestamp	TimestampMsType	報告が発生した時刻
status	PhysicalObservationStatusType	報告時点の運転者の物理的観測の状態（全量）

また，イベント種別がEventOverflowの場合には，以下の関連情報を持つ。

noLostEvents	UInt32	失われたイベントの数

5.18.8. データ型の定義

5.18.8.1. PhysicalObservationの機能クラス

```
type PhysicalObservationCapabilityType = enum {
    EyeStateDetectable,
    EyewearDetectable,
    MouthStateDetectable,
    HandPositionDetectable,
    TorsoPoseDetectable,
    LegPositionDetectable,
    BodySizeEstimable
};
```

5.18.8.2. PhysicalObservationの構成情報（getConfigが返すデータ型）

```
type PhysicalObservationConfigType = struct {
    capabilities : enumSet(PhysicalObservationCapabilityType),
    coordinateSystem : String,
    referencePoint : String,
    statusReportInterval : UInt32 ([1,4294967295])?,
    timestampAccuracy : UInt32
};
```

5.18.8.3. 3次元の点

```
type DriverPoint3DType = struct {
    x : Float64,    // X座標（車両前方が正）。単位はm
    y : Float64,    // Y座標（車両左方が正）。単位はm
    z : Float64 // Z座標（上方が正）。単位はm
};
```

運転者に関する観測の3次元の点（位置）を表す型である。

座標系はISO　8855の車両座標系（原点：後軸中心の路面投影点）とし，構成情報coordinateSystemおよびreferencePointで宣言される。

不確かさの表現にも本型を用いる。その場合，各成分は対応する軸方向の推定誤差を表す不確かさであり，その意味は「不確かさと信頼度の表現」に規定するとおりである。

5.18.8.4. 頭部の位置・姿勢と顔バウンディングボックス

```
type PhysicalObservationHeadPoseAndFaceBoxType = struct {
    headPosition : DriverPoint3DType,   // 頭部中心の位置。単位はm
    yaw : Float64,  // 頭部のヨー角。単位はdeg。車両前方を0とし，左向きが正
    pitch : Float64,    // 頭部のピッチ角。単位はdeg。前傾（下向き）が正
    roll : Float64, // 頭部のロール角。単位はdeg。右傾が正
    faceBoxHeight : Float64,    // 顔バウンディングボックスの高さ。単位はm
    faceBoxWidth : Float64  // 顔バウンディングボックスの幅。単位はm
};
```

顔バウンディングボックスは，顔を囲む矩形の寸法を表す。画像上の画素数ではなく，実空間における寸法（m）で表す。

```
type PhysicalObservationHeadPoseAndFaceBoxUncertaintyType = struct {
    headPositionUncertainty : DriverPoint3DType?,   // 頭部位置の軸方向別の不確かさ。単位はm
    yawUncertainty : Float64?,  // 頭部のヨー角の不確かさ。単位はdeg
    pitchUncertainty : Float64?,    // 頭部のピッチ角の不確かさ。単位はdeg
    rollUncertainty : Float64?, // 頭部のロール角の不確かさ。単位はdeg
    faceBoxHeightUncertainty : Float64?,    // 顔バウンディングボックスの高さの不確かさ。単位はm
    faceBoxWidthUncertainty : Float64?  // 顔バウンディングボックスの幅の不確かさ。単位はm
};
```

5.18.8.5. 目の状態と視線

```
type PhysicalObservationGazeDirectionType = struct {
    yaw : Float64,  // 視線方向のヨー角。単位はdeg。車両前方を0とし，左向きが正
    pitch : Float64 // 視線方向のピッチ角。単位はdeg。下向きが正
};
```

```
type PhysicalObservationEyeStateType = struct {
    eyePosition : DriverPoint3DType,    // 目（眼球中心）の位置。単位はm
    eyeOpenness : Float64 ([0.0,1.0]),  // 瞼の開閉度。0.0が全閉，1.0が通常の全開
    pupilPosition : DriverPoint3DType?  // 瞳（瞳孔中心）の位置。単位はm。推定できている場合にのみ値を持つ
};
```

```
type PhysicalObservationEyeStateAndGazeType = struct {
    leftEye : PhysicalObservationEyeStateType?, // 左目の状態。観測できている場合にのみ値を持つ
    rightEye : PhysicalObservationEyeStateType?,    // 右目の状態。観測できている場合にのみ値を持つ
    gazeDirection : PhysicalObservationGazeDirectionType?   // 両眼の観測から合成した視線方向。推定できている場合（閉眼中等でない場合）にのみ値を持つ
};
```

```
type PhysicalObservationEyeStateAndGazeUncertaintyType = struct {
    eyePositionUncertainty : DriverPoint3DType?,    // 目位置の軸方向別の不確かさ（左右共通）。単位はm
    pupilPositionUncertainty : DriverPoint3DType?,  // 瞳位置の軸方向別の不確かさ（左右共通）。単位はm
    gazeDirectionUncertainty : Float64?,    // 視線方向の角度の不確かさ。単位はdeg
    eyeOpennessUncertainty : Float64?   // 瞼開閉度の不確かさ
};
```

5.18.8.6. アイウェアの装用状態

```
type PhysicalObservationEyewearStateType = enum {
    NoEyewear,  // 装用なし
    Glasses,    // メガネ
    Sunglasses, // サングラス
    ContactLenses,  // コンタクトレンズ
    Eyepatch,   // 眼帯
    Unknown // 不明
};
```

複数のアイウェアを同時に装用している場合は，目の観測への影響が最も大きいものを返す（影響の大きい順に，眼帯，サングラス，メガネ，コンタクトレンズとする）。

装用の有無を判定できない場合はUnknownとなる。

5.18.8.7. 口の状態と発話

```
type PhysicalObservationMouthStateAndSpeechActivityType = struct {
    mouthPosition : DriverPoint3DType,  // 口（口中心）の位置。単位はm
    mouthOpenness : Float64 ([0.0,1.0]),    // 口の開き具合（開口度）。0.0が全閉，1.0
```

```
が全開
    speechActivity : Bool   // 発話中か
};
```

```
type PhysicalObservationMouthStateAndSpeechActivityUncertaintyType = struct {
    mouthPositionUncertainty : DriverPoint3DType?,  // 口位置の軸方向別の不確かさ。単位はm
    mouthOpennessUncertainty : Float64?,    // 開口度の不確かさ
    speechActivityUncertainty : Float64?    // 発話判定（発話中か否か）に対する正規化した不確かさ。単位を持たない0.0（最も確か）から1.0（最も不確か）の値をとる
};
```

5.18.8.8. 手の位置

```
type PhysicalObservationHandPositionType = struct {
    leftHandPosition : DriverPoint3DType?,  // 左手の位置。単位はm。観測できている場合にのみ値を持つ
    rightHandPosition : DriverPoint3DType?  // 右手の位置。単位はm。観測できている場合にのみ値を持つ
};
```

```
type PhysicalObservationHandPositionUncertaintyType = struct {
    leftHandPositionUncertainty : DriverPoint3DType?,   // 左手位置の軸方向別の不確かさ。単位はm
    rightHandPositionUncertainty : DriverPoint3DType?   // 右手位置の軸方向別の不確かさ。単位はm
};
```

5.18.8.9. ステアリング把持状態

```
type PhysicalObservationSteeringWheelGripStateType = enum {
    NotGripped, // 両手とも把持していない
    LeftHandOnly,   // 左手のみ把持している
    RightHandOnly,  // 右手のみ把持している
    BothHands,  // 両手で把持している
    Unknown // 不明
};
```

把持とは，手がステアリングホイールに掛かっており，操舵入力を与えられる状態をいう。触れているだけの状態を把持と判定するかは観測手段に依存するため，判定の基準は車両側実装が定め，判定の確からしさは信頼度（steeringWheelGripStateConfidenceLevel）で申告する。

把持の有無を判定できない場合はUnknownとなる。

5.18.8.10. 胴体姿勢

```
type PhysicalObservationTorsoPoseType = struct {
    torsoCenterPosition : DriverPoint3DType,    // 胴体中心の位置。単位はm
    yaw : Float64,  // 胴体のヨー角。単位はdeg。車両前方を0とし，左向きが正
    pitch : Float64,    // 胴体のピッチ角。単位はdeg。前傾（下向き）が正
    roll : Float64  // 胴体のロール角。単位はdeg。右傾が正
};
```

```
type PhysicalObservationTorsoPoseUncertaintyType = struct {
    torsoCenterPositionUncertainty : DriverPoint3DType?,    // 胴体中心位置の軸方向別
の不確かさ。単位はm
    yawUncertainty : Float64?,  // 胴体のヨー角の不確かさ。単位はdeg
    pitchUncertainty : Float64?,    // 胴体のピッチ角の不確かさ。単位はdeg
    rollUncertainty : Float64?  // 胴体のロール角の不確かさ。単位はdeg
};
```

5.18.8.11. 足の位置とペダル操作状況

```
type PhysicalObservationPedalOperationStateType = enum {
    NotOnPedal, // いずれのペダルにも足が掛かっていない
    OnAccelerator,  // アクセルペダルに足が掛かっている
    OnBrake,    // ブレーキペダルに足が掛かっている
    OnMultiplePedals,   // 複数のペダルに同時に足が掛かっている
    OnOtherPedal,   // その他のペダル（クラッチペダル等）に足が掛かっている
    Unknown // 不明
};
```

```
type PhysicalObservationLegPositionAndPedalOperationType = struct {
    leftLegPosition : DriverPoint3DType?,   // 左足（足先）の位置。単位はm。観測できて
いる場合にのみ値を持つ
    rightLegPosition : DriverPoint3DType?,  // 右足（足先）の位置。単位はm。観測できて
いる場合にのみ値を持つ
    pedalOperationState : PhysicalObservationPedalOperationStateType    // ペダル操作
状況
};
```

```
type PhysicalObservationLegPositionAndPedalOperationUncertaintyType = struct {
    leftLegPositionUncertainty : DriverPoint3DType?,    // 左足位置の軸方向別の不確か
さ。単位はm
    rightLegPositionUncertainty : DriverPoint3DType?,   // 右足位置の軸方向別の不確か
さ。単位はm
    pedalOperationStateUncertainty : Float64?   // ペダル操作状況の判定に対する正規化
した不確かさ。単位を持たない0.0（最も確か）から1.0（最も不確か）の値をとる
};
```

5.18.8.12. タイムスタンプ

```
type TimestampUsType = UInt64;  // 車内共通時刻基盤の起点を基準とする測定時刻。単位はμs。複数のビークルAPIで共通に用いる共通データ型であり，本仕様内に定義する。
```

5.18.8.13. サンプル品質の種別

```
type DataQualifierType = enum {
    Normal, // 正常
    Unavailable,    // 利用不可
    Degraded,   // 劣化
    TestMode,   // テストモード
    Invalid // 無効（値も誤差も読んではならない）
};
```

5.18.8.14. PhysicalObservation状態

```
type PhysicalObservationStatusType = struct {
    timestamp : TimestampUsType,
    dataQualifier : DataQualifierType,
    headPoseAndFaceBox : PhysicalObservationHeadPoseAndFaceBoxType,
    headPoseAndFaceBoxUncertainty :
PhysicalObservationHeadPoseAndFaceBoxUncertaintyType,
    eyeStateAndGaze : PhysicalObservationEyeStateAndGazeType?,
    eyeStateAndGazeUncertainty : PhysicalObservationEyeStateAndGazeUncertaintyType?,
    eyewearState : PhysicalObservationEyewearStateType?,
    eyewearConfidenceLevel : Float64 ([0.0,1.0])?,
    mouthStateAndSpeechActivity : PhysicalObservationMouthStateAndSpeechActivityType?,
    mouthStateAndSpeechActivityUncertainty :
PhysicalObservationMouthStateAndSpeechActivityUncertaintyType?,
    handPosition : PhysicalObservationHandPositionType?,
    handPositionUncertainty : PhysicalObservationHandPositionUncertaintyType?,
    steeringWheelGripState : PhysicalObservationSteeringWheelGripStateType,
    steeringWheelGripStateConfidenceLevel : Float64 ([0.0,1.0]),
    torsoPose : PhysicalObservationTorsoPoseType?,
    torsoPoseUncertainty : PhysicalObservationTorsoPoseUncertaintyType?,
    legPositionAndPedalOperation :
PhysicalObservationLegPositionAndPedalOperationType?,
    legPositionAndPedalOperationUncertainty :
PhysicalObservationLegPositionAndPedalOperationUncertaintyType?,
    heightEstimate : Float64?,
    heightUncertainty : Float64?,
    weightEstimate : Float64?,
    weightUncertainty : Float64?
};
```

位置・姿勢の座標系はISO 8855の車両座標系（原点：後軸中心の路面投影点）とする。不確かさ・正規化

した不確かさ・信頼度の意味と値域は，「不確かさと信頼度の表現」に規定するとおりである。
timestampは，車内共通時刻基盤に基づく測定時刻を表す。dataQualifierがInvalidまたはUnavailableの場合，観測値・不確かさ・信頼度のいずれも読んではならない。

5.18.8.15. PhysicalObservationイベント

```
type PhysicalObservationEventKindType = enum {
    SteeringWheelGripChanged,
    FaultDetected,
    FaultRecovered,
    StatusReported,
    EventOverflow,
    ConfigChanged
};
```

```
type PhysicalObservationEventType = struct {
    eventInfo : variant(PhysicalObservationEventKindType) {
        SteeringWheelGripChanged {
            gripState : PhysicalObservationSteeringWheelGripStateType
        },
        StatusReported {
            timestamp : TimestampMsType ,
            status : PhysicalObservationStatusType
        },
        EventOverflow {
            noLostEvents : UInt32
        }
    }
};
```

5.18.9. サービスコールの詳細規定

getConfig

PhysicalObservationの構成情報の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
config	PhysicalObservationConfigType	PhysicalObservationの構成情報

【エラーコード】

なし

【機能】

PhysicalObservationに関する構成情報を返す。

getStatus

運転者の物理的観測の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
status	PhysicalObservationStatusType	運転者の物理的観測（推定値と不確かさ・信頼度）

【エラーコード】

E_DENIED_ACCESS	運転者の物理的観測の参照が認められていない
E_OBJECT_FAULT	故障中（運転者の物理的観測を提供できない状態）

【機能】

運転者の物理的観測に関するすべての状態を返す。返される値は観測に基づく推定値であり，精度は保証されない。取得した値の信頼度は，併せて返されるdataQualifierと各不確かさ・信頼度によって判断する。dataQualifierがInvalidまたはUnavailableの場合，観測値・不確かさ・信頼度のいずれも読んではならない。

運転者の物理的観測は運転者の身体的特徴・挙動・体調を推定できる高感度な情報であるため，本サービスコールはアクセス制御の対象である。

notify

PhysicalObservationイベントの通知要求

【パラメータ】

eventFilter	enumSet (PhysicalObservationEventKindType)?	通知するイベントの種類の集合

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【エラーコード】

E_DENIED_ACCESS	運転者の物理的観測の参照が認められていない
E_INVALID_PARAMETER	eventFilterが有効範囲外

【機能】

PhysicalObservationで発生したeventFilterで指定したイベントの通知を要求する。イベントを通知するためのイベントキューを生成し，その通知ハンドルをnotifyHandleに返す。

eventFilterを省略した場合，すべての種類のイベントの通知を要求する。

通知されるイベントには，報告時点の状態の全量を運ぶStatusReportedイベントが含まれることがあり，また，ステアリング把持状態の変化等のイベントの発生の観測からも運転者の挙動を推定できる。このため，本サービスコールの呼び出しそのものが，eventFilterの内容によらず（eventFilterを省略した場合を含む），アクセス制御の対象であり，アクセスが認められていないアプリケーションに対してはE_DENIED_ACCESSエラーとなる。

unnotify

PhysicalObservationイベントの通知停止

【パラメータ】

notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正

【機能】

notifyHandleで指定した通知を停止する。指定した通知用のイベントキューは削除される。

getEvent

PhysicalObservationイベントの取得

【パラメータ】

notifyHandle	NotifyHandleType	イベントキューを識別するための通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
event	PhysicalObservationEventType	取得したイベント

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正
E_NO_DATA	イベントキューが空

【機能】

notifyHandleで指定した通知用のイベントキューから，先頭のイベントを取り出す。取り出したイベントは，イベントキューから削除される。



イベントがパラメータ等で渡されてくるプログラミング環境向けの物理APIでは，用意する必要がない。

5.19. Vehicle.Driver.PhysiologicalMeasurement（略称：PhysiologicalMeasurement）

5.19.1. 位置付けと機能

Vehicle.Driver.PhysiologicalMeasurement（略称：PhysiologicalMeasurement）は，現在の運転者の生体計測値を提供するシングルインスタンスオブジェクトである。心拍数を基本の計測値として提供し，心拍変動（HRV）指標とRR間隔（拍動間隔）のビート列を，機能クラスによって段階的に提供する。各計測値には測定時刻timestampとサンプル品質dataQualifierが対になって提供される。

5.19.1.1. PhysiologicalMeasurementの位置付けと性質

PhysiologicalMeasurementが提供する値は，生体センサによる計測に基づく推定値であり，確定値（グラウンドトゥルース）ではない。計測手段・装着部位は規定せず，構成情報sensingModalityによって測定原理を宣言する。計測手段や環境（体動・接触状態等）による精度の違いは，サンプル品質dataQualifierと（算出できる場合の）不確かさの実値として表現され，APIとしての精度の保証は行わない。光電式脈波法（PPG）による測定では，環境に依存しない定量的な精度の保証は原理上行えず，心拍変動・RR間隔は手首等の末梢での測定において安静時に限り信頼でき，運転中はサンプル品質が劣化（Degraded）または無効（Invalid）となりうる。

対象は現在の運転者1名である。運転者が誰であるかの同定や，個人のプロファイルは扱わない。

本APIは，心拍数・心拍変動・RR間隔の医学的な判断（頻脈・徐脈・心拍変動の急低下等の判定）を行わない。本APIは値・品質・時刻の提供に徹し，値の解釈と判断は利用側に委ねる。

5.19.1.2. 測定時刻とサンプル品質

状態には，計測の測定時刻timestampとサンプル品質dataQualifierが含まれる。timestampは，車内共通時刻基盤に基づく測定時刻（計測のエポック）であり，送信時刻ではない。同じ時刻基盤に基づく他のビークルAPIのタイムスタンプと直接比較できる。タイムスタンプ自体の誤差上限は，構成情報timestampAccuracyで宣言される。

dataQualifierは，取得できたサンプルの品質を表す。故障（生体計測機能を提供できない状態。E_OBJECT_FAULTエラーおよびFaultDetectedイベント）が「計測値を出力できない状態」を表すのに対し，dataQualifierは「出力されたサンプルがどれだけ信用できるか」を表す。両者の使い分けの詳細は，状態の節に規定する。

5.19.1.3. サンプル品質の写像

dataQualifierは，計測時の接触状態と精度状態に基づいて定める。センサが体表に接触しておらず値を取得できない状態（非接触・オフボディ・取得開始中を含む）はUnavailableとする。値は使えるが精度が低い状態はDegradedとする。値が信頼できず使用に耐えない状態，および精度を判断できない状態はInvalidとする。精度が十分な状態はNormalとする。dataQualifierがInvalidまたはUnavailableのサンプルでは，心拍数・不確かさ・心拍変動・RR間隔のいずれも読んではならない。

心拍数の値域は20〜300　　bpmであり，心停止など計測範囲外の状態は心拍数の値（0等）ではなく，dataQualifierがUnavailableまたはInvalidのサンプルによって表す。

5.19.1.4. PhysiologicalMeasurementの機能

心拍数（heartRate）は，すべてのPhysiologicalMeasurementが提供する基本の計測値であり，機能クラスに依存しない。心拍数の不確かさ（heartRateUncertainty）は，心拍数の数値としての誤差を算出できている場合にのみ値を持つ。

心拍変動（heartRateVariability）は機能クラスHeartRateVariabilityDetectableによって，RR間隔（rrIntervals）は機能クラスRrIntervalDetectableによって宣言される。これらが値を持つ条件は，状態の節に規定する。心拍数だけを出力する計測装置では，これらの機能クラスをいずれも備えず，心拍変動・RR間隔は値なしとなる。

5.19.1.5. 状態の更新とアクセス制御

状態の更新周期は車両実装に依存する。状態は，要求応答（getStatus）により取得できるほか，状態報告イベント（StatusReported）によって周期的に得ることができる。心拍数・心拍変動・RR間隔は連続更新される信号系データであるため，その変化をきっかけに発生するイベントは定義しない。計測値は，StatusReportedイベントの関連情報statusにより周期的に得られる。また，計測値のしきい値横断（頻脈・徐脈・心拍変動の急低下等）を判定するイベントも定義しない。状態の変化をきっかけに発生するイベントとして通知するのは，故障の検知・回復のみである。

心拍数・心拍変動・RR間隔は，運転者の身体状態・体調を推定できる極めて高感度な個人情報である。とりわけ心拍変動は，自律神経の活動やストレスなどの健康状態を強く推定しうるため，一段機微である。そのため，状態を参照するサービスコール（getStatus）はアクセス制御の対象であり，アクセスが認められていないアプリケーションに対してはE_DENIED_ACCESSエラーとなる。イベントの通知要求（notify）により通知されるイベントには，報告時点の状態の全量を運ぶStatusReportedイベントが含まれることがあり，また，イベントの発生の観測からも運転者の状態を推定できる。このため，notifyの呼び出しそのものを，eventFilterで指定するイベントの種類によらず，アクセス制御の対象とする。

5.19.2. 機能クラス

PhysiologicalMeasurementは，以下の機能クラスを持つ。

HeartRateVariabilityDetectable	心拍変動（heartRateVariability）の指標を提供することができる
RrIntervalDetectable	RR間隔（rrIntervals）のビート列を提供することができる

5.19.3. リスククラス

PhysiologicalMeasurementは，車両の状態を変化させるサービスコールを持たないため，リスククラスを定義しない。

生体計測値の参照に伴うプライバシのリスクに対しては，状態の参照（getStatus）およびイベントの通知要求（notify）へのアクセス制御（E_DENIED_ACCESSエラー）により対応する。

5.19.4. 構成情報

PhysiologicalMeasurementは，以下の構成情報を持つ。

capabilities	enumSet (PhysiologicalMeasurementCapabilityType)	そのPhysiologicalMeasurementが持つ機能（心拍変動・RR間隔のうち提供できるもの）
sensingModality	String	生体計測の測定原理の宣言（光電式脈波法等）
statusReportInterval	UInt32 ([1,4294967295])?	状態報告イベント（StatusReported）が発生する周期。単位はms。値を持つ場合は1以上。値なしの場合，この車両ではStatusReportedイベントは発生しない
timestampAccuracy	UInt32	この車両のタイムスタンプ誤差上限の宣言。単位はμs

5.19.5. 状態

PhysiologicalMeasurementは，以下の状態を持つ。

timestamp	TimestampUsType	測定時刻。単位はμs。車内共通時刻基盤に基づく
dataQualifier	DataQualifierType	このサンプルの品質。InvalidまたはUnavailableの場合，心拍数・不確かさ・心拍変動・RR間隔のいずれも読んではならない
heartRate	UInt16 ([20,300])	心拍数の推定値。単位はbpm。値域は20〜300。心停止など計測範囲外の状態はdataQualifier（Unavailable／Invalid）で表す
heartRateUncertainty	Float64?	心拍数の推定値の不確かさ。単位はbpm。信頼水準は95%（2σ相当）。数値としての誤差を算出できない計測手段では値なしとなり，その場合の品質判断の主軸はdataQualifierである
heartRateVariability	PhysiologicalMeasurementHrvType?	心拍変動（HRV）の指標。機能クラスHeartRateVariabilityDetectableを備え，かつ当該値を算出できている場合にのみ値を持つ
rrIntervals	UInt32[]?	前サンプル以降に観測されたRR間隔（拍動間隔）の配列。各要素は1拍の間隔を表し，単位はms。機能クラスRrIntervalDetectableを備え，かつ当該値を算出できている場合にのみ値を持つ

heartRateは，すべてのPhysiologicalMeasurementが提供する基本の計測値であり，機能クラスに依存しない。heartRateUncertaintyは，心拍数の数値としての誤差を算出できている場合にのみ値を持つ。heartRateVariabilityは機能クラスHeartRateVariabilityDetectableを，rrIntervalsは機能クラスRrIntervalDetectableを備え，かつ当該値を算出できている場合にのみ値を持つ。

heartRateVariabilityは，複数の指標からなる構造体である。連続する正常拍間隔の差の二乗平均平方根（rmssd）を基本の指標とし，拍間隔の標準偏差（sdnn）・隣接拍間隔差が50msを超える割合（pnn50）・周波数領域の指標（frequencyDomain）・指標の算出区間長（windowSeconds）は，算出できている場合にのみ値を持つ。心拍変動の指標には不確かさを対で付けず，品質はdataQualifierで表す。

rrIntervalsは，前サンプル以降に観測されたRR間隔を配列として運ぶ。状態の1回の更新の間に複数の拍が入りうるため，最新のサンプルが持つ配列が，前サンプル以降に観測されたすべてのRR間隔を表す。StatusReportedイベントおよびgetStatusで得られるのは報告・読み出しの時点の最新のサンプルだけであるため，報告または読み出しの間隔が状態の更新間隔より長い場合，受け取らなかったサンプルが持っていたRR間隔は得られない。このため，本項目は，抜けのないRR間隔の系列を保証しない。観測された異常な拍は補正・除外せず観測値のまま配列に含め，サンプル全体の品質はdataQualifierで示す。

timestampは，このサンプルの計測の測定時刻（計測のエポック）であり，車内共通時刻基盤に基づく。タイムスタンプの誤差上限は，構成情報timestampAccuracyで宣言される。

dataQualifierは，このサンプルの品質を表す。Invalidは，計測値を信用できない状態（値が信頼できず使用に耐えない状態，精度を判断できない状態，起動直後の初期化中など）であり，心拍数・不確かさ・心拍変動・RR間隔のいずれも読んではならない。Degradedは，値は使えるが通常品質でない状態（体動等により計測品質が低下している場合）であり，精度そのものは（算出できる場合の）不確かさの実値で判断する。

センサが体表に接触しておらず計測値を取得できない状態（非接触・オフボディ・取得開始中）では，計測値を出力できないため，dataQualifierがUnavailableのサンプルとなる（故障とは扱わない）。生体計測機能そのものを提供できない状態（センサの故障等）は故障と扱い，故障中の状態の参照（getStatus）はE_OBJECT_FAULTエラーとなる。故障中にStatusReportedイベントで報告される状態は，dataQualifierがUnavailableである。Unavailableのサンプルでは，心拍数・不確かさ・心拍変動・RR間隔のいずれも読んではならない（最終既知値として読むことも許されない）。Unavailableのサンプルのtimestampは，サンプルの生成時刻を表す（測定時刻ではない）。

5.19.6. サービスコール一覧

PhysiologicalMeasurementに対するサービスコールは以下の通り。

getConfig	PhysiologicalMeasurementの構成情報の参照
getStatus	生体計測値の参照
notify	PhysiologicalMeasurementイベントの通知要求
unnotify	PhysiologicalMeasurementイベントの通知停止
getEvent	PhysiologicalMeasurementイベントの取得

5.19.7. イベント

PhysiologicalMeasurementのイベント種別には，故障の検知・回復，および状態の報告（StatusReported）がある。心拍数・心拍変動・RR間隔は連続更新される信号系データであるため，その変化をきっかけに発生するイベントは定義しない。計測値は，StatusReportedイベントの関連情報statusにより周期的に得られる。StatusReportedイベントは，報告時点の状態の全量（PhysiologicalMeasurementStatusType）を関連情報statusとして含む。計測値のしきい値横断（頻脈・徐脈・心拍変動の急低下等）を判定するイベントも定義しない。本APIは値・品質・時刻の提供に徹し，値の解釈と判断は利用側に委ねる。

PhysiologicalMeasurementに対するイベント種別は以下の通り。

FaultDetected	故障検知（生体計測機能を提供できない状態）
FaultRecovered	故障から回復
StatusReported	状態の報告（構成情報statusReportIntervalの周期ごとに発生する）
EventOverflow	イベントキューのオーバフロー
ConfigChanged	構成情報の変更

StatusReportedイベントの関連情報statusは，報告時点の運転者の生体計測値の状態の全量であり，getStatusが返す状態と同じ型（PhysiologicalMeasurementStatusType）である。関連情報timestamp（TimestampMsType）と，状態が持つtimestamp（TimestampUsType）とは単位が異なり，起点が同一であることも保証されないため，互いに比較することはできない。状態の更新周期と構成情報statusReportIntervalとは一致するとは限らない。dataQualifierがUnavailableでないサンプルどうしでは，連続する報告の関連情報statusが同じ計測に基づくか否かを，statusのtimestampにより判断できる。dataQualifierがUnavailableのサンプルのtimestampはサンプルの生成時刻を表すため，これにより判断することはできない。

この内，イベント種別がStatusReportedの場合には，以下の関連情報を持つ。

timestamp	TimestampMsType	報告が発生した時刻
status	PhysiologicalMeasurementStatusType	報告時点の運転者の生体計測値の状態（全量）

また，イベント種別がEventOverflowの場合には，以下の関連情報を持つ。

noLostEvents	UInt32	失われたイベントの数

5.19.8. データ型の定義

5.19.8.1. PhysiologicalMeasurementの機能クラス

```
type PhysiologicalMeasurementCapabilityType = enum {
    HeartRateVariabilityDetectable,
    RrIntervalDetectable
};
```

5.19.8.2. PhysiologicalMeasurementの構成情報（getConfigが返すデータ型）

```
type PhysiologicalMeasurementConfigType = struct {
    capabilities : enumSet(PhysiologicalMeasurementCapabilityType),
    sensingModality : String,
    statusReportInterval : UInt32 ([1,4294967295])?,
    timestampAccuracy : UInt32
};
```

5.19.8.3. タイムスタンプ

```
type TimestampUsType = UInt64;  // 車内共通時刻基盤の起点を基準とする測定時刻。単位はμs。複数のビークルAPIで共通に用いる共通データ型であり，本仕様内に定義する。
```

5.19.8.4. サンプル品質の種別

```
type DataQualifierType = enum {
    Normal, // 正常
    Unavailable,    // 利用不可
    Degraded,   // 劣化
    TestMode,   // テストモード
    Invalid // 無効（値も誤差も読んではならない）
};
```

5.19.8.5. 心拍変動（HRV）

```
type PhysiologicalMeasurementHrvFrequencyDomainType = struct {
    lfPower : Float64,  // 低周波帯（LF）のパワー。単位はms²
    hfPower : Float64,  // 高周波帯（HF）のパワー。単位はms²
    lfHfRatio : Float64 // 低周波帯と高周波帯のパワー比（LF/HF比）。無次元
};
```

心拍変動の周波数領域の指標を表す型である。周波数帯の境界の定義は車両実装に依存する。

```
type PhysiologicalMeasurementHrvType = struct {
    rmssd : Float64,    // 連続する正常拍間隔の差の二乗平均平方根（RMSSD）。単位はms
    sdnn : Float64?,    // 拍間隔の標準偏差（SDNN）。単位はms。算出できている場合にのみ値を持つ
    pnn50 : Float64 ([0,100])?, // 隣接する拍間隔の差が50msを超える割合（pNN50）。単位は%。算出できている場合にのみ値を持つ
    frequencyDomain : PhysiologicalMeasurementHrvFrequencyDomainType?,  // 周波数領域の指標。算出できている場合にのみ値を持つ
    windowSeconds : UInt32? // 指標の算出に用いた区間長。単位はs。指定できている場合にのみ値を持つ
};
```

心拍変動（HRV）の指標を表す型である。連続する正常拍間隔の差の二乗平均平方根（rmssd）を基本の指標とし，その他の指標は算出できている場合にのみ値を持つ。心拍変動の指標には不確かさを対で付けず，品質はサンプルのdataQualifierで表す。

5.19.8.6. PhysiologicalMeasurement状態

```
type PhysiologicalMeasurementStatusType = struct {
    timestamp : TimestampUsType,
    dataQualifier : DataQualifierType,
```

```
    heartRate : UInt16 ([20,300]),
    heartRateUncertainty : Float64?,
    heartRateVariability : PhysiologicalMeasurementHrvType?,
    rrIntervals : UInt32[]?
};
```

heartRateはすべてのPhysiologicalMeasurementが提供する基本の計測値である。heartRateVariabilityとrrIntervalsが値を持つ条件は，状態の節に規定する。heartRateの値域は20〜300 bpmである。timestampは，車内共通時刻基盤に基づく測定時刻を表す。dataQualifierがInvalidまたはUnavailableの場合，心拍数・不確かさ・心拍変動・RR間隔のいずれも読んではならない。

5.19.8.7. PhysiologicalMeasurementイベント

```
type PhysiologicalMeasurementEventKindType = enum {
    FaultDetected,
    FaultRecovered,
    StatusReported,
    EventOverflow,
    ConfigChanged
};
```

```
type PhysiologicalMeasurementEventType = struct {
    eventInfo : variant(PhysiologicalMeasurementEventKindType) {
        StatusReported {
            timestamp : TimestampMsType ,
            status : PhysiologicalMeasurementStatusType
        },
        EventOverflow {
            noLostEvents : UInt32
        }
    }
};
```

5.19.9. サービスコールの詳細規定

getConfig

PhysiologicalMeasurementの構成情報の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

config	PhysiologicalMeasurementConfigType	PhysiologicalMeasurementの構成情報

【エラーコード】

なし

【機能】

PhysiologicalMeasurementに関する構成情報を返す。

getStatus

生体計測値の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
status	PhysiologicalMeasurementStatusType	運転者の生体計測値（心拍数・心拍変動・RR間隔と品質・時刻）

【エラーコード】

E_DENIED_ACCESS	生体計測値の参照が認められていない
E_OBJECT_FAULT	故障中（生体計測機能を提供できない状態）

【機能】

運転者の生体計測値に関するすべての状態を返す。返される値は計測に基づく推定値であり，精度は保証されない。取得した値の確からしさは，併せて返されるdataQualifierと（算出できる場合の）不確かさによって判断する。dataQualifierがInvalidまたはUnavailableの場合，心拍数・不確かさ・心拍変動・RR間隔のいずれも読んではならない。

心拍数・心拍変動・RR間隔は運転者の身体状態・体調を推定できる極めて高感度な個人情報であるため，本サービスコールはアクセス制御の対象である。

notify

PhysiologicalMeasurementイベントの通知要求

【パラメータ】

eventFilter	enumSet (PhysiologicalMeasurementEventKindType)?	通知するイベントの種類の集合

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【エラーコード】

E_DENIED_ACCESS	生体計測値の参照が認められていない
E_INVALID_PARAMETER	eventFilterが有効範囲外

【機能】

PhysiologicalMeasurementで発生したeventFilterで指定したイベントの通知を要求する。イベントを通知するためのイベントキューを生成し，その通知ハンドルをnotifyHandleに返す。

eventFilterを省略した場合，すべての種類のイベントの通知を要求する。

通知されるイベントには，報告時点の状態の全量を運ぶStatusReportedイベントが含まれることがあり，また，イベントの発生の観測からも運転者の状態を推定できる。このため，本サービスコールの呼び出しそのものが，eventFilterの内容によらず（eventFilterを省略した場合を含む），アクセス制御の対象であり，アクセスが認められていないアプリケーションに対してはE_DENIED_ACCESSエラーとなる。

unnotify

PhysiologicalMeasurementイベントの通知停止

【パラメータ】

notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正

【機能】

notifyHandleで指定した通知を停止する。指定した通知用のイベントキューは削除される。

getEvent

PhysiologicalMeasurementイベントの取得

【パラメータ】

notifyHandle	NotifyHandleType	イベントキューを識別するための通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
event	PhysiologicalMeasurementEventType	取得したイベント

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正
E_NO_DATA	イベントキューが空

【機能】

notifyHandleで指定した通知用のイベントキューから，先頭のイベントを取り出す。取り出したイベントは，イベントキューから削除される。



イベントがパラメータ等で渡されてくるプログラミング環境向けの物理APIでは，用意する必要がない。

5.20. Vehicle.Driver.CognitiveState（略称：CognitiveState）

5.20.1. 位置付けと機能

Vehicle.Driver.CognitiveState（略称：CognitiveState）は，運転者（現在のドライバ）の認知状態の推定値を提供するシングルインスタンスオブジェクトである。疲労・眠気・注意・注意散漫・感情・パニック・応答性の推定値と，それぞれの推定の信頼度，および測定時刻とサンプル品質を状態として提供する。

5.20.1.1. CognitiveStateの位置付けと性質

CognitiveStateが提供する各状態は，車両が運転者の観測（車内カメラ・生理計測・車両挙動等）から推定した値であり，確定値（グラウンドトゥルース）ではない。推定手段は規定しない。各推定値の定義の細部・分類の判定基準（閾値）・信頼度の算出モデルは車両実装に依存し，本APIでは規定しない。アプリケーションは，取得した値を確定的な事実として扱ってはならず，併せて返される信頼度とサンプル品質により値の確からしさを判断するものとする。

CognitiveStateの対象は，現在運転席にいる運転者1名である。運転者の身体の観測値（頭部姿勢・視線・手足の位置等）はVehicle.Driver.PhysicalObservationが担当し，CognitiveStateは持たない。

5.20.1.2. 信頼度の表現

各推定値には，推定の信頼度（fatigueConfidenceLevel等の*ConfidenceLevelフィールド）が対になる。すべての信頼度の値域は[0.0,1.0]に統一し，1.0が最大信頼を表す。信頼度は精度の保証ではなく，推定器が自己申告する確からしさである。

5.20.1.3. 測定時刻とサンプル品質

状態には，推定の測定時刻timestampとサンプル品質dataQualifierが含まれる。timestampは，車内共通時刻基盤に基づく測定時刻（推定の基になった観測のエポック）であり，送信時刻ではない。同じ時刻基盤に基づく他のビークルAPIのタイムスタンプと直接比較できる。タイムスタンプ自体の誤差上限は，構

成情報timestampAccuracyで宣言される。

dataQualifierは，取得できたサンプルの品質を表す。故障（認知状態の推定機能を提供できない状態。E_OBJECT_FAULTエラーおよびFaultDetectedイベント）が「推定値を出力できない状態」を表すのに対し，dataQualifierは「出力されたサンプルがどれだけ信用できるか」を表す。両者の使い分けの詳細は，状態の節に規定する。

5.20.1.4. 状態の更新とアクセス制御

状態の更新周期は車両実装に依存する。状態は，要求応答（getStatus）により取得できるほか，状態報告イベント（StatusReported）によって周期的に得ることができる。疲労度・眠気レベル等の推定値は連続更新される信号系データであるため，その変化をきっかけに発生するイベントは定義しない。推定値は，StatusReportedイベントの関連情報statusにより周期的に得られる。状態の変化をきっかけに発生するイベントとして通知するのは，応答性分類の遷移（ResponsivenessChanged）と，故障の検知・回復のみである。

認知状態・感情状態は，運転者の内面を推定できる極めて高感度な個人情報である。そのため，状態を参照するサービスコール（getStatus）はアクセス制御の対象であり，アクセスが認められていないアプリケーションに対してはE_DENIED_ACCESSエラーとなる。イベントの通知要求（notify）により通知されるイベントには，報告時点の状態の全量を運ぶStatusReportedイベントが含まれることがあり，また，ResponsivenessChanged等のイベントの発生パターンの観測からも運転者の状態を推定できる。このため，notifyの呼び出しそのものを，eventFilterで指定するイベントの種類によらず，アクセス制御の対象とする。

5.20.2. 機能クラス

CognitiveStateは，以下の機能クラスを持つ。

AttentionDetectable	注意状態（注意度レベル・注意状態分類）を推定することができる
EmotionDetectable	感情状態（感情分類・感情強度・快不快度・覚醒度・支配度・推定情報源）を推定することができる
PanicDetectable	パニック状態を推定することができる

5.20.3. リスククラス

CognitiveStateは，車両の状態を変化させるサービスコールを持たないため，リスククラスを定義しない。

認知状態・感情状態の参照に伴うプライバシのリスクに対しては，状態の参照（getStatus）およびイベントの通知要求（notify）へのアクセス制御（E_DENIED_ACCESSエラー）により対応する。

5.20.4. 構成情報

CognitiveStateは，以下の構成情報を持つ。

capabilities	enumSet (CognitiveStateCapabilityType)	その車両の認知状態推定が持つ機能

statusReportInterval	UInt32 ([1,4294967295])?	状態報告イベント（StatusReported）が発生する周期。単位はms。値を持つ場合は1以上。値なしの場合，この車両ではStatusReportedイベントは発生しない
timestampAccuracy	UInt32	この車両のタイムスタンプ誤差上限の宣言。単位はμs

5.20.5. 状態

CognitiveStateは，以下の状態を持つ。

timestamp	TimestampUsType	測定時刻。単位はμs。車内共通時刻基盤に基づく
dataQualifier	DataQualifierType	このサンプルの品質。InvalidまたはUnavailableの場合，認知状態の推定値および信頼度のいずれも読んではならない
fatigueLevel	Float64 ([0.0,1.0])	長期的な疲労度の推定値。0.0が疲労なし，1.0が最大の疲労を表す
fatigueConfidenceLevel	Float64 ([0.0,1.0])	疲労度推定の信頼度。1.0が最大信頼
drowsiness	Float64 ([0.0,1.0])	眠気レベルの推定値。0.0が眠気なし，1.0が最大の眠気を表す
drowsinessConfidenceLevel	Float64 ([0.0,1.0])	眠気レベル推定の信頼度。1.0が最大信頼
attentionLevel	Float64 ([0.0,1.0])?	運転への注意度レベルの推定値。0.0が注意なし，1.0が最大の注意を表す
attentionCategory	CognitiveStateAttentionCategoryType?	注意状態の分類の推定値
attentionConfidenceLevel	Float64 ([0.0,1.0])?	注意状態推定の信頼度。1.0が最大信頼
distractionCategory	CognitiveStateDistractionCategoryType	注意散漫の分類の推定値
distractionConfidenceLevel	Float64 ([0.0,1.0])	注意散漫状態推定の信頼度。1.0が最大信頼
emotionCategory	CognitiveStateEmotionCategoryType?	感情の分類の推定値
emotionIntensity	Float64 ([0.0,1.0])?	支配的な感情（emotionCategory）の強度の推定値。0.0が最小，1.0が最大の強度を表す
valence	Float64 ([-1.0,1.0])?	感情の快・不快度の推定値。負が不快，正が快，0.0が中立を表す
arousal	Float64 ([0.0,1.0])?	覚醒・興奮の度合いの推定値。0.0が最小，1.0が最大の覚醒を表す
dominance	Float64 ([0.0,1.0])?	支配感（状況を制御できている感覚）の推定値。0.0が最小，1.0が最大の支配感を表す

affectiveStateSource	CognitiveStateAffectiveStateSourceType?	感情状態の推定に用いた観測情報源
affectiveStateConfidenceLevel	Float64 ([0.0,1.0])?	感情状態推定の信頼度。1.0が最大信頼
panicLevel	Float64 ([0.0,1.0])?	パニック度の推定値。0.0がパニックなし，1.0が最大のパニックを表す
panicConfidenceLevel	Float64 ([0.0,1.0])?	パニック度推定の信頼度。1.0が最大信頼
responsivenessLevel	Float64 ([0.0,1.0])	運転者の応答性（運転タスクへの応答可能性）の推定レベル。0.0が応答不能，1.0が完全に応答可能であることを表す
responsivenessConfidenceLevel	Float64 ([0.0,1.0])	応答性推定の信頼度。1.0が最大信頼
responsivenessCategory	CognitiveStateResponsivenessCategoryType	運転者の応答性の分類の推定値

attentionLevel，attentionCategory，attentionConfidenceLevelは，機能クラスAttentionDetectableを備えている場合にのみ値を持つ。emotionCategory，emotionIntensity，valence，arousal，dominance，affectiveStateSource，affectiveStateConfidenceLevelは，機能クラスEmotionDetectableを備えている場合にのみ値を持つ（この機能クラスを備えていても，emotionIntensity・valence・arousal・dominance・affectiveStateSourceのうち算出できない項目は値なしとなる）。panicLevel，panicConfidenceLevelは，機能クラスPanicDetectableを備えている場合にのみ値を持つ。

分類の推定値（attentionCategory，distractionCategory，emotionCategory，responsivenessCategory）において，機能を備えているが当該サンプルで分類を確定できない場合には，値なしではなくUnknownとなる。

responsivenessCategory（離散分類）とresponsivenessLevel（連続量）は役割を分担する。responsivenessCategoryは，安全ロジックの判定やイベント通知（ResponsivenessChanged）に用いる離散状態である。responsivenessLevelは，応答性の連続的な監視・傾向把握に用いる連続量である。分類と連続量の対応（どのレベル範囲がどの分類に当たるか）は車両実装に依存し，本APIでは規定しない。

すべての信頼度（*ConfidenceLevel）の値域は[0.0,1.0]であり，1.0が最大信頼を表す。

timestampは，このサンプルの推定の測定時刻（推定の基になった観測のエポック）であり，車内共通時刻基盤に基づく。複数の観測を統合して推定する実装では，代表となる観測の測定時刻を用いる。タイムスタンプの誤差上限は，構成情報timestampAccuracyで宣言される。

dataQualifierは，このサンプルの品質を表す。Invalidは，推定器自身が自分の出力（信頼度の値を含む）を信用できない状態（起動直後の初期化中・運転者の検出直後の学習中など）であり，認知状態の推定値および信頼度のいずれも読んではならない。Invalidを小さな信頼度の値で代用してはならない。Degradedは，値は使えるが通常品質でない状態（例：カメラの部分的な遮蔽等により観測が不足し，一部の観測に基づく推定が継続している場合）であり，推定の確からしさそのものは信頼度の実値で判断する。

運転者を観測できていない状態（運転席に運転者がいない場合や，視野の遮蔽等により一時的に観測できない場合）では，推定値を出力できないため，dataQualifierがUnavailableのサンプルとなる（故障とは扱わない）。認知状態の推定機能そのものを提供できない状態（観測に必要なセンサの故障等）は故障と

扱い，故障中の状態の参照（getStatus）はE_OBJECT_FAULTエラーとなる。故障中にStatusReportedイベントで報告される状態は，dataQualifierがUnavailableである。Unavailableのサンプルでは，認知状態の推定値および信頼度のいずれも読んではならない（最終既知値として読むことも許されない）。Unavailableのサンプルのtimestampは，サンプルの生成時刻を表す（測定時刻ではない）。

5.20.6. サービスコール一覧

CognitiveStateに対するサービスコールは以下の通り。

getConfig	CognitiveStateの構成情報の参照
getStatus	認知状態の推定値の参照
notify	CognitiveStateイベントの通知要求
unnotify	CognitiveStateイベントの通知停止
getEvent	CognitiveStateイベントの取得

5.20.7. イベント

CognitiveStateのイベント種別には，応答性分類の遷移（ResponsivenessChanged），故障の検知・回復，および状態の報告（StatusReported）がある。連続量（疲労度・眠気レベル等）は連続更新される信号系データであるため，その変化をきっかけに発生するイベントは定義しない。連続量の値は，StatusReportedイベントの関連情報statusにより周期的に得られる。

イベントの関連情報には遷移の内容などの最小限の情報のみを含め，状態のデータ本体は含めない。データ本体は，イベントを受けたアプリケーションがgetStatusにより取得する。ただし，StatusReportedイベントは，報告時点の状態の全量（CognitiveStateStatusType）を関連情報statusとして含む。

CognitiveStateに対するイベント種別は以下の通り。

ResponsivenessChanged	運転者の応答性分類（responsivenessCategory）の遷移。分類の判定基準は車両実装に依存するが，実装は，推定の揺らぎにより本イベントが高頻度に発生しないよう，分類の判定にヒステリシスを持たせるものとする
FaultDetected	故障検知（認知状態の推定機能を提供できない状態）
FaultRecovered	故障から回復
StatusReported	状態の報告（構成情報statusReportIntervalの周期ごとに発生する）
EventOverflow	イベントキューのオーバフロー
ConfigChanged	構成情報の変更

StatusReportedイベントの関連情報statusは，報告時点の認知状態の全量であり，getStatusが返す状態と同じ型（CognitiveStateStatusType）である。関連情報timestamp（TimestampMsType）と，状態が持つtimestamp（TimestampUsType）とは単位が異なり，起点が同一であることも保証されないため，互いに比較することはできない。状態の更新周期と構成情報statusReportIntervalとは一致するとは限らない。dataQualifierがUnavailableでないサンプルどうしでは，連続する報告の関連情報statusが同じ推定結果に基づくか否かを，statusのtimestampにより判断できる。dataQualifierがUnavailableのサンプルのtimestampはサンプルの生成時刻を表すため，これにより判断することはできない。

この内，イベント種別がResponsivenessChangedの場合には，以下の関連情報を持つ。

previousCategory	CognitiveStateResponsivenessCategoryType	遷移前の応答性分類
newCategory	CognitiveStateResponsivenessCategoryType	遷移後の応答性分類

また，イベント種別がStatusReportedの場合には，以下の関連情報を持つ。

timestamp	TimestampMsType	報告が発生した時刻
status	CognitiveStateStatusType	報告時点の認知状態（全量）

また，イベント種別がEventOverflowの場合には，以下の関連情報を持つ。

noLostEvents	UInt32	失われたイベントの数

5.20.8. データ型の定義

5.20.8.1. CognitiveStateの機能クラス

```
type CognitiveStateCapabilityType = enum {
    AttentionDetectable,
    EmotionDetectable,
    PanicDetectable
};
```

5.20.8.2. CognitiveStateの構成情報（getConfigが返すデータ型）

```
type CognitiveStateConfigType = struct {
    capabilities : enumSet(CognitiveStateCapabilityType),
    statusReportInterval : UInt32 ([1,4294967295])?,
    timestampAccuracy : UInt32
};
```

5.20.8.3. タイムスタンプ

```
type TimestampUsType = UInt64;  // 車内共通時刻基盤の起点を基準とする測定時刻。単位はμs。複数のビークルAPIで共通に用いる共通データ型であり，本仕様内に定義する。
```

5.20.8.4. サンプル品質の種別

```
type DataQualifierType = enum {
    Normal, // 正常
    Unavailable,    // 利用不可
    Degraded,   // 劣化
    TestMode,   // テストモード
```

```
    Invalid // 無効（値も誤差も読んではならない）
};
```

5.20.8.5. 注意状態の分類

```
type CognitiveStateAttentionCategoryType = enum {
    Attentive,  // 運転に注意を向けている
    Inattentive,    // 運転への注意が不足している
    Distracted, // 注意散漫である（内訳はdistractionCategoryが表す）
    Unknown // 分類を確定できない
};
```

```
type CognitiveStateDistractionCategoryType = enum {
    None,   // 注意散漫でない
    Visual, // 視覚的な注意散漫（視線が運転タスクから逸れている）
    Manual, // 手操作による注意散漫（手が運転タスク以外の操作に使われている）
    Cognitive,  // 認知的な注意散漫（考え事等により注意が運転から逸れている）
    Auditory,   // 聴覚的な注意散漫（会話・音声等に注意が向いている）
    Multiple,   // 複数の種類の注意散漫が併発している
    Unknown // 分類を確定できない
};
```

5.20.8.6. 感情状態の分類

```
type CognitiveStateEmotionCategoryType = enum {
    Neutral,    // 中立
    Joy,    // 喜び
    Anger,  // 怒り
    Sadness,    // 悲しみ
    Fear,   // 恐れ
    Surprise,   // 驚き
    Disgust,    // 嫌悪
    Other,  // その他の感情
    Unknown // 分類を確定できない
};
```

```
type CognitiveStateAffectiveStateSourceType = enum {
    FacialExpression,   // 表情
    Voice,  // 音声
    Gaze,   // 視線
    HeadPose,   // 頭部姿勢
    BodyPose,   // 身体姿勢
    PhysiologicalSignal,    // 生理信号
    VehicleBehavior,    // 車両挙動
    MultimodalFusion,   // 複数の情報源の統合
    Unknown // 情報源を特定できない
```

```
};
```

affectiveStateSourceは，感情状態の推定に用いた観測情報源を1つの値で表す。複数の情報源を統合して推定した場合はMultimodalFusionとなる。

5.20.8.7. 応答性の分類

```
type CognitiveStateResponsivenessCategoryType = enum {
    Responsive, // 応答可能（運転タスクに正常に応答できる）
    DelayedResponse,    // 応答遅延（応答できるが遅れがある）
    Unresponsive,   // 応答不能（運転タスクに応答できない）
    Unknown // 分類を確定できない
};
```

5.20.8.8. CognitiveState状態

```
type CognitiveStateStatusType = struct {
    timestamp : TimestampUsType,
    dataQualifier : DataQualifierType,
    fatigueLevel : Float64 ([0.0,1.0]),
    fatigueConfidenceLevel : Float64 ([0.0,1.0]),
    drowsiness : Float64 ([0.0,1.0]),
    drowsinessConfidenceLevel : Float64 ([0.0,1.0]),
    attentionLevel : Float64 ([0.0,1.0])?,
    attentionCategory : CognitiveStateAttentionCategoryType?,
    attentionConfidenceLevel : Float64 ([0.0,1.0])?,
    distractionCategory : CognitiveStateDistractionCategoryType,
    distractionConfidenceLevel : Float64 ([0.0,1.0]),
    emotionCategory : CognitiveStateEmotionCategoryType?,
    emotionIntensity : Float64 ([0.0,1.0])?,
    valence : Float64 ([-1.0,1.0])?,
    arousal : Float64 ([0.0,1.0])?,
    dominance : Float64 ([0.0,1.0])?,
    affectiveStateSource : CognitiveStateAffectiveStateSourceType?,
    affectiveStateConfidenceLevel : Float64 ([0.0,1.0])?,
    panicLevel : Float64 ([0.0,1.0])?,
    panicConfidenceLevel : Float64 ([0.0,1.0])?,
    responsivenessLevel : Float64 ([0.0,1.0]),
    responsivenessConfidenceLevel : Float64 ([0.0,1.0]),
    responsivenessCategory : CognitiveStateResponsivenessCategoryType
};
```

attentionLevel，attentionCategory，attentionConfidenceLevelは機能クラスAttentionDetectableを，emotionCategory，emotionIntensity，valence，arousal，dominance，affectiveStateSource，affectiveStateConfidenceLevelは機能クラスEmotionDetectableを，panicLevel，panicConfidenceLevelは機能クラスPanicDetectableを備えている場合にのみ値を持つ。すべての信頼度（*ConfidenceLevel）の値域は[0.0,1.0]であり，1.0が最大信頼を表す。timestampは，車内共通時刻基盤に基づく測定時刻を表

す。dataQualifierがInvalidまたはUnavailableの場合，認知状態の推定値および信頼度のいずれも読んではならない。

5.20.8.9. CognitiveStateイベント

```
type CognitiveStateEventKindType = enum {
    ResponsivenessChanged,
    FaultDetected,
    FaultRecovered,
    StatusReported,
    EventOverflow,
    ConfigChanged
};
```

```
type CognitiveStateEventType = struct {
    eventInfo : variant(CognitiveStateEventKindType) {
        ResponsivenessChanged {
            previousCategory : CognitiveStateResponsivenessCategoryType ,
            newCategory : CognitiveStateResponsivenessCategoryType
        },
        StatusReported {
            timestamp : TimestampMsType ,
            status : CognitiveStateStatusType
        },
        EventOverflow {
            noLostEvents : UInt32
        }
    }
};
```

5.20.9. サービスコールの詳細規定

getConfig

CognitiveStateの構成情報の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
config	CognitiveStateConfigType	CognitiveStateの構成情報

【エラーコード】

なし

【機能】

CognitiveStateに関する構成情報を返す。

getStatus

認知状態の推定値の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
status	CognitiveStateStatusType	運転者の認知状態（推定値と信頼度）

【エラーコード】

E_DENIED_ACCESS	認知状態の参照が認められていない
E_OBJECT_FAULT	故障中（認知状態の推定機能を提供できない状態）

【機能】

運転者の認知状態に関するすべての状態を返す。返される値は車両による推定値であり，確定値（グラウンドトゥルース）ではない。取得した値の確からしさは，併せて返されるdataQualifierと各信頼度によって判断する。dataQualifierがInvalidの場合，認知状態の推定値および信頼度を読んではならない。

認知状態・感情状態は運転者の内面を推定できる極めて高感度な個人情報であるため，本サービスコールはアクセス制御の対象である。

notify

CognitiveStateイベントの通知要求

【パラメータ】

eventFilter	enumSet (CognitiveStateEventKindType)?	通知するイベントの種類の集合

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【エラーコード】

E_DENIED_ACCESS	認知状態の参照が認められていない
E_INVALID_PARAMETER	eventFilterが有効範囲外

【機能】

CognitiveStateで発生したeventFilterで指定したイベントの通知を要求する。イベントを通知するためのイベントキューを生成し，その通知ハンドルをnotifyHandleに返す。

eventFilterを省略した場合，すべての種類のイベントの通知を要求する。

通知されるイベントには，報告時点の状態の全量を運ぶStatusReportedイベントが含まれることがあり，また，ResponsivenessChanged等のイベントの発生パターンの観測からも運転者の状態を推定できる。このため，本サービスコールの呼び出しそのものが，eventFilterの内容によらず（eventFilterを省略した場合を含む），アクセス制御の対象であり，アクセスが認められていないアプリケーションに対してはE_DENIED_ACCESSエラーとなる。

unnotify

CognitiveStateイベントの通知停止

【パラメータ】

notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正

【機能】

notifyHandleで指定した通知を停止する。指定した通知用のイベントキューは削除される。

getEvent

CognitiveStateイベントの取得

【パラメータ】

notifyHandle	NotifyHandleType	イベントキューを識別するための通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
event	CognitiveStateEventType	取得したイベント

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正
E_NO_DATA	イベントキューが空

【機能】

notifyHandleで指定した通知用のイベントキューから，先頭のイベントを取り出す。取り出したイベントは，イベントキューから削除される。



イベントがパラメータ等で渡されてくるプログラミング環境向けの物理APIでは，用意する必要がない。

5.21. Vehicle.Driver.IntentAndCompliance.Violation（略称：IntentAndComplianceViolation）

5.21.1. 位置付けと機能

Vehicle.Driver.IntentAndCompliance.Violation（略称：IntentAndComplianceViolation）は，現在の運転者の運転意図と法規違反の状態に関する推定値を提供するシングルインスタンスオブジェクトである。現在の運転意図と次の運転意図の推定，運転意図推定の不確かさ，法規違反の状態の推定，およびその推定の信頼度を状態として提供する。各状態には測定時刻timestampとサンプル品質dataQualifierが対になって提供される。

5.21.1.1. IntentAndComplianceViolationの位置付けと性質

IntentAndComplianceViolationが提供する各状態は，車両が運転者の観測（車内カメラ・車両挙動・操作入力等）から推定した値であり，確定値（グラウンドトゥルース）ではない。推定手段・分類の判定基準・信頼度の算出モデルは車両実装に依存し，本APIでは規定しない。アプリケーションは，取得した値を確定的な事実として扱ってはならず，併せて返される信頼度・不確かさとサンプル品質により値の確からしさを判断するものとする。

本APIは，運転意図および法規違反状態の推定値の提供に徹する。法規違反の状態は推定値であり，本APIは違反の断定・確定や，運転者に対する法的な判断を行わない。法規（交通規則）の適用と解釈は各管轄区域に依存するため，complianceViolationStateの分類が具体的にどの規則にどの条件で対応するかは車両実装および適用地域に依存する。利用側は，本APIの推定値を根拠に法的な確定判断を下してはならない。

運転意図および法規違反状態の推定は，車両を操作する直接的なアクチュエーション（操作）コマンドとは明確に分離される。本APIは参照専用であり，運転操作を指示するサービスコールを持たない。

本APIの対象は，現在運転席にいる運転者1名である。運転者の認知状態（疲労・注意・感情等）はVehicle.Driver.CognitiveStateが，身体の観測値はVehicle.Driver.PhysicalObservationが担当し，本オブジェクトは持たない。

5.21.1.2. 不確かさと信頼度の表現

運転意図推定には不確かさintentEstimationUncertaintyが対になる。この不確かさは，単位を持たない正規化した値であり，推定誤差の実値ではない。値域は[0.0,1.0]であり，0.0が最も確か（不確かさなし），1.0が最も不確かであることを表す。

法規違反状態の推定には信頼度complianceViolationConfidenceLevelが対になる。信頼度の値域は[0.0,1.0]であり，1.0が最大信頼を表す。信頼度・不確かさは精度の保証ではなく，推定器が自己申告する確からしさである。

5.21.1.3. 測定時刻とサンプル品質

状態には，推定の測定時刻timestampとサンプル品質dataQualifierが含まれる。timestampは，車内共通

時刻基盤に基づく測定時刻（推定の基になった観測のエポック）であり，送信時刻ではない。同じ時刻基盤に基づく他のビークルAPIのタイムスタンプと直接比較できる。タイムスタンプ自体の誤差上限は，構成情報timestampAccuracyで宣言される。

dataQualifierは，取得できたサンプルの品質を表す。故障（意図・法規違反の推定機能を提供できない状態。E_OBJECT_FAULTエラーおよびFaultDetectedイベント）が「推定値を出力できない状態」を表すのに対し，dataQualifierは「出力されたサンプルがどれだけ信用できるか」を表す。両者の使い分けの詳細は，状態の節に規定する。

5.21.1.4. IntentAndComplianceViolationの機能

運転意図の推定（intentEstimation・intentEstimationUncertainty）は，機能クラスIntentEstimationDetectableによって宣言され，これを備えている場合にのみ値を持つ。法規違反状態の推定（complianceViolationState・complianceViolationConfidenceLevel）は，機能クラスComplianceViolationDetectableによって宣言され，これを備えている場合にのみ値を持つ。いずれの機能クラスも備えていない実装は，本オブジェクトを提供しない。

5.21.1.5. 状態の更新とアクセス制御

状態の更新周期は車両実装に依存する。状態は，要求応答（getStatus）により取得できるほか，状態報告イベント（StatusReported）によって周期的に得ることができる。運転意図・法規違反状態の推定値は連続更新される信号系データであるため，その変化をきっかけに発生するイベントは定義しない。推定値は，StatusReportedイベントの関連情報statusにより周期的に得られる。状態の変化をきっかけに発生するイベントとして通知するのは，故障の検知・回復のみである。

運転意図・法規違反状態は，運転者の運転行動を推定できる高感度な個人情報である。そのため，状態を参照するサービスコール（getStatus）はアクセス制御の対象であり，アクセスが認められていないアプリケーションに対してはE_DENIED_ACCESSエラーとなる。イベントの通知要求（notify）により通知されるイベントには，報告時点の状態の全量を運ぶStatusReportedイベントが含まれることがあり，また，イベントの発生の観測からも運転者の状態を推定できる。このため，notifyの呼び出しそのものを，eventFilterで指定するイベントの種類によらず，アクセス制御の対象とする。

5.21.2. 機能クラス

IntentAndComplianceViolationは，以下の機能クラスを持つ。

IntentEstimationDetectable	運転意図（現在の運転意図と次の運転意図）を推定することができる
ComplianceViolationDetectable	法規違反の状態を推定することができる

5.21.3. リスククラス

IntentAndComplianceViolationは，車両の状態を変化させるサービスコールを持たないため，リスククラスを定義しない。

運転意図・法規違反状態の参照に伴うプライバシのリスクに対しては，状態の参照（getStatus）およびイベントの通知要求（notify）へのアクセス制御（E_DENIED_ACCESSエラー）により対応する。

5.21.4. 構成情報

IntentAndComplianceViolationは，以下の構成情報を持つ。

capabilities	enumSet (IntentAndComplianceViolationCapabilityType)	そのIntentAndComplianceViolationが持つ機能（運転意図の推定・法規違反状態の推定のうち提供できるもの）
statusReportInterval	UInt32 ([1,4294967295])?	状態報告イベント（StatusReported）が発生する周期。単位はms。値を持つ場合は1以上。値なしの場合，この車両ではStatusReportedイベントは発生しない
timestampAccuracy	UInt32	この車両のタイムスタンプ誤差上限の宣言。単位はμs

5.21.5. 状態

IntentAndComplianceViolationは，以下の状態を持つ。

timestamp	TimestampUsType	測定時刻。単位はμs。車内共通時刻基盤に基づく
dataQualifier	DataQualifierType	このサンプルの品質。InvalidまたはUnavailableの場合，運転意図・法規違反状態の推定値および不確かさ・信頼度のいずれも読んではならない
intentEstimation	IntentAndComplianceViolationIntentEstimationType?	現在の運転意図と次の運転意図の推定値。機能クラスIntentEstimationDetectableを備えている場合にのみ値を持つ
intentEstimationUncertainty	Float64 ([0.0,1.0])?	運転意図推定の不確かさ。単位を持たない正規化した値で，0.0が最も確か，1.0が最も不確かであることを表す。機能クラスIntentEstimationDetectableを備えている場合にのみ値を持つ
complianceViolationConfidenceLevel	Float64 ([0.0,1.0])?	法規違反状態推定の信頼度。1.0が最大信頼。機能クラスComplianceViolationDetectableを備えている場合にのみ値を持つ
complianceViolationState	IntentAndComplianceViolationComplianceViolationStateType?	法規違反の状態の推定値。機能クラスComplianceViolationDetectableを備えている場合にのみ値を持つ

intentEstimation，intentEstimationUncertaintyは，機能クラスIntentEstimationDetectableを備えている場合にのみ値を持つ。complianceViolationState，complianceViolationConfidenceLevelは，機能クラスComplianceViolationDetectableを備えている場合にのみ値を持つ。

intentEstimationは，現在の運転意図（currentIntent）と次の運転意図（nextIntent）からなる構造体である。機能を備えているが当該サンプルで運転意図を確定できない場合には，各分類はUnknownとなる。intentEstimationUncertaintyの値域は[0.0,1.0]であり，0.0が最も確か（不確かさなし），1.0が最も不確かであることを表す。

complianceViolationStateは，法規違反の状態の推定値であり，確定値ではない。機能を備えているが当該サンプルで分類を確定できない場合には，値なしではなくUnknownとなる。
complianceViolationConfidenceLevelの値域は[0.0,1.0]であり，1.0が最大信頼を表す。
complianceViolationStateは推定値であって違反の断定ではなく，法規の適用・解釈は各管轄区域に依存する。

timestampは，このサンプルの推定の測定時刻（推定の基になった観測のエポック）であり，車内共通時刻基盤に基づく。複数の観測を統合して推定する実装では，代表となる観測の測定時刻を用いる。タイムスタンプの誤差上限は，構成情報timestampAccuracyで宣言される。

dataQualifierは，このサンプルの品質を表す。Invalidは，推定器自身が自分の出力（不確かさ・信頼度の値を含む）を信用できない状態（起動直後の初期化中・運転者の検出直後の学習中など）であり，運転意図・法規違反状態の推定値および不確かさ・信頼度のいずれも読んではならない。Degradedは，値は使えるが通常品質でない状態（観測が不足し，一部の観測に基づく推定が継続している場合等）であり，推定の確からしさそのものは不確かさ・信頼度の実値で判断する。

運転者を観測できていない状態（運転席に運転者がいない場合や，一時的に観測できない場合）では，推定値を出力できないため，dataQualifierがUnavailableのサンプルとなる（故障とは扱わない）。意図・法規違反の推定機能そのものを提供できない状態（観測に必要なセンサの故障等）は故障と扱い，故障中の状態の参照（getStatus）はE_OBJECT_FAULTエラーとなる。故障中にStatusReportedイベントで報告される状態は，dataQualifierがUnavailableである。Unavailableのサンプルでは，運転意図・法規違反状態の推定値および不確かさ・信頼度のいずれも読んではならない（最終既知値として読むことも許されない）。Unavailableのサンプルのtimestampは，サンプルの生成時刻を表す（測定時刻ではない）。

5.21.6. サービスコール一覧

IntentAndComplianceViolationに対するサービスコールは以下の通り。

getConfig	IntentAndComplianceViolationの構成情報の参照
getStatus	運転意図・法規違反状態の推定値の参照
notify	IntentAndComplianceViolationイベントの通知要求
unnotify	IntentAndComplianceViolationイベントの通知停止
getEvent	IntentAndComplianceViolationイベントの取得

5.21.7. イベント

IntentAndComplianceViolationのイベント種別には，故障の検知・回復，および状態の報告（StatusReported）がある。運転意図・法規違反状態の推定値は連続更新される信号系データであるため，その変化をきっかけに発生するイベントは定義しない。推定値は，StatusReportedイベントの関連情報statusにより周期的に得られる。本APIは推定値・品質・時刻の提供に徹し，値の解釈と判断は利用側に委ねる。

IntentAndComplianceViolationに対するイベント種別は以下の通り。

FaultDetected	故障検知（意図・法規違反の推定機能を提供できない状態）
FaultRecovered	故障から回復
StatusReported	状態の報告（構成情報statusReportIntervalの周期ごとに発生する）
EventOverflow	イベントキューのオーバフロー
ConfigChanged	構成情報の変更

StatusReportedイベントの関連情報statusは，報告時点の運転意図・法規違反状態の全量であり，getStatusが返す状態と同じ型（IntentAndComplianceViolationStatusType）である。関連情報timestamp（TimestampMsType）と，状態が持つtimestamp（TimestampUsType）とは単位が異なり，起点が同一であることも保証されないため，互いに比較することはできない。状態の更新周期と構成

情報statusReportIntervalとは一致するとは限らない。dataQualifierがUnavailableでないサンプルどうしでは，連続する報告の関連情報statusが同じ推定結果に基づくか否かを，statusのtimestampにより判断できる。dataQualifierがUnavailableのサンプルのtimestampはサンプルの生成時刻を表すため，これにより判断することはできない。

この内，イベント種別がStatusReportedの場合には，以下の関連情報を持つ。

timestamp	TimestampMsType	報告が発生した時刻
status	IntentAndComplianceViolationStatusType	報告時点の運転意図・法規違反状態（全量）

また，イベント種別がEventOverflowの場合には，以下の関連情報を持つ。

noLostEvents	UInt32	失われたイベントの数

5.21.8. データ型の定義

5.21.8.1. IntentAndComplianceViolationの機能クラス

```
type IntentAndComplianceViolationCapabilityType = enum {
    IntentEstimationDetectable,
    ComplianceViolationDetectable
};
```

5.21.8.2. IntentAndComplianceViolationの構成情報（getConfigが返すデータ型）

```
type IntentAndComplianceViolationConfigType = struct {
    capabilities : enumSet(IntentAndComplianceViolationCapabilityType),
    statusReportInterval : UInt32 ([1,4294967295])?,
    timestampAccuracy : UInt32
};
```

5.21.8.3. タイムスタンプ

```
type TimestampUsType = UInt64;  // 車内共通時刻基盤の起点を基準とする測定時刻。単位はμs。複数のビークルAPIで共通に用いる共通データ型であり，本仕様内に定義する。
```

5.21.8.4. サンプル品質の種別

```
type DataQualifierType = enum {
    Normal, // 正常
    Unavailable,    // 利用不可
    Degraded,   // 劣化
    TestMode,   // テストモード
    Invalid // 無効（値も誤差も読んではならない）
```

```
};
```

5.21.8.5. 運転意図の分類

```
type IntentAndComplianceViolationDrivingIntentType = enum {
    KeepLane,   // 現在の車線を維持する
    Accelerate, // 加速する
    Decelerate, // 減速する
    Stop,   // 停止する
    LaneChangeLeft, // 左への車線変更を行う
    LaneChangeRight,    // 右への車線変更を行う
    TurnLeft,   // 左折する
    TurnRight,  // 右折する
    UTurn,  // Uターンする
    Merge,  // 本線等へ合流する
    Overtake,   // 追い越しを行う
    Park,   // 駐車する
    Reverse,    // 後退する
    Unknown // 分類を確定できない
};
```

運転意図の分類を1つの値で表す。currentIntentは現在の運転意図を，nextIntentは次に予測される運転意図を表す。分類の判定基準は車両実装に依存する。

5.21.8.6. 法規違反の状態の分類

```
type IntentAndComplianceViolationComplianceViolationStateType = enum {
    Compliant,  // 法規違反は推定されない（法規に従っていると推定される）
    Speeding,   // 速度超過が推定される
    LaneViolation,  // 車線に関する違反（車線のはみ出し・逆走等）が推定される
    SignalViolation,    // 信号無視が推定される
    SignViolation,  // 標識・標示に関する違反（一時停止・進入禁止等）が推定される
    FollowingTooClose,  // 車間距離の不足が推定される
    ProhibitedManeuver, // 禁止された操作（禁止区間での転回・追い越し等）が推定される
    Other,  // その他の法規違反が推定される
    Unknown // 分類を確定できない
};
```

法規違反の状態の推定値を1つの値で表す。各分類は推定であって違反の断定ではなく，どの分類がどの交通規則にどの条件で対応するかは，法規の適用・解釈が各管轄区域に依存するため車両実装および適用地域に依存する。

5.21.8.7. 運転意図の推定

```
type IntentAndComplianceViolationIntentEstimationType = struct {
    currentIntent : IntentAndComplianceViolationDrivingIntentType,  // 現在の運転意図の推定値
    nextIntent : IntentAndComplianceViolationDrivingIntentType  // 次に予測される運転
```

```
意図の推定値
};
```

現在の運転意図と次に予測される運転意図を表す型である。運転意図を確定できない場合，各分類はUnknownとなる。

5.21.8.8. IntentAndComplianceViolation状態

```
type IntentAndComplianceViolationStatusType = struct {
    timestamp : TimestampUsType,
    dataQualifier : DataQualifierType,
    intentEstimation : IntentAndComplianceViolationIntentEstimationType?,
    intentEstimationUncertainty : Float64 ([0.0,1.0])?,
    complianceViolationConfidenceLevel : Float64 ([0.0,1.0])?,
    complianceViolationState :
IntentAndComplianceViolationComplianceViolationStateType?
};
```

intentEstimation，intentEstimationUncertaintyは機能クラスIntentEstimationDetectableを，complianceViolationState，complianceViolationConfidenceLevelは機能クラスComplianceViolationDetectableを備えている場合にのみ値を持つ。intentEstimationUncertaintyの値域は[0.0,1.0]であり0.0が最も確か，complianceViolationConfidenceLevelの値域は[0.0,1.0]であり1.0が最大信頼を表す。timestampは，車内共通時刻基盤に基づく測定時刻を表す。dataQualifierがInvalidまたはUnavailableの場合，運転意図・法規違反状態の推定値および不確かさ・信頼度のいずれも読んではならない。

5.21.8.9. IntentAndComplianceViolationイベント

```
type IntentAndComplianceViolationEventKindType = enum {
    FaultDetected,
    FaultRecovered,
    StatusReported,
    EventOverflow,
    ConfigChanged
};
```

```
type IntentAndComplianceViolationEventType = struct {
    eventInfo : variant(IntentAndComplianceViolationEventKindType) {
        StatusReported {
            timestamp : TimestampMsType ,
            status : IntentAndComplianceViolationStatusType
        },
        EventOverflow {
            noLostEvents : UInt32
        }
    }
};
```

5.21.9. サービスコールの詳細規定

getConfig

IntentAndComplianceViolationの構成情報の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
config	IntentAndComplianceViolationConfigType	IntentAndComplianceViolationの構成情報

【エラーコード】

なし

【機能】

IntentAndComplianceViolationに関する構成情報を返す。

getStatus

運転意図・法規違反状態の推定値の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
status	IntentAndComplianceViolationStatusType	運転者の運転意図・法規違反状態の推定値（推定値と不確かさ・信頼度）

【エラーコード】

E_DENIED_ACCESS	運転意図・法規違反状態の参照が認められていない
E_OBJECT_FAULT	故障中（意図・法規違反の推定機能を提供できない状態）

【機能】

運転者の運転意図・法規違反状態に関するすべての状態を返す。返される値は車両による推定値であり，確定値（グラウンドトゥルース）ではない。取得した値の確からしさは，併せて返されるdataQualifierと不確かさ・信頼度によって判断する。dataQualifierがInvalidまたはUnavailableの場合，運転意図・法規違反状態の推定値および不確かさ・信頼度を読んではならない。法規違反の状態は推定値であり，本サービスコールは違反の断定・法的判断を行わない。

運転意図・法規違反状態は運転者の運転行動を推定できる高感度な個人情報であるため，本サービスコー

ルはアクセス制御の対象である。

notify

IntentAndComplianceViolationイベントの通知要求

【パラメータ】

eventFilter	enumSet (IntentAndComplianceViolationEventKindType)?	通知するイベントの種類の集合

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【エラーコード】

E_DENIED_ACCESS	運転意図・法規違反状態の参照が認められていない
E_INVALID_PARAMETER	eventFilterが有効範囲外

【機能】

IntentAndComplianceViolationで発生したeventFilterで指定したイベントの通知を要求する。イベントを通知するためのイベントキューを生成し，その通知ハンドルをnotifyHandleに返す。

eventFilterを省略した場合，すべての種類のイベントの通知を要求する。

通知されるイベントには，報告時点の状態の全量を運ぶStatusReportedイベントが含まれることがあり，また，イベントの発生の観測からも運転者の状態を推定できる。このため，本サービスコールの呼び出しそのものが，eventFilterの内容によらず（eventFilterを省略した場合を含む），アクセス制御の対象であり，アクセスが認められていないアプリケーションに対してはE_DENIED_ACCESSエラーとなる。

unnotify

IntentAndComplianceViolationイベントの通知停止

【パラメータ】

notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正

【機能】

notifyHandleで指定した通知を停止する。指定した通知用のイベントキューは削除される。

getEvent

IntentAndComplianceViolationイベントの取得

【パラメータ】

notifyHandle	NotifyHandleType	イベントキューを識別するための通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
event	IntentAndComplianceViolationEventType	取得したイベント

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正
E_NO_DATA	イベントキューが空

【機能】

notifyHandleで指定した通知用のイベントキューから，先頭のイベントを取り出す。取り出したイベントは，イベントキューから削除される。



イベントがパラメータ等で渡されてくるプログラミング環境向けの物理APIでは，用意する必要がない。

5.22. Vehicle.Navigation（略称：Navi）

5.22.1. 位置付けと機能

Vehicle.Navigation（略称：Navi）は，車両のナビゲーション機能を扱うシングルインスタンスオブジェクトである。目的地の設定と経路の計算・案内，経路計算時の回避エリアの指定，設定されている経路・目的地・経由地の参照，現在位置の住所（行政区域）の判定，走行中の道路の種別・属性の提供を行う。状態には，測定時刻とサンプル品質が含まれる。

5.22.1.1. ナビゲーションの位置付けと性質

Naviは，車両が保持する地図データに基づいて，経路・住所・道路情報などの地図知識に由来する情報を提供するオブジェクトである。自己位置の幾何情報（緯度経度・高度・方位とそれらの推定誤差）はVehicle.CurrentLocationが提供し，Naviはそれらを重複して持たない。現在位置の住所や走行中の道路の種別のように，位置を地図と照合して得られる情報は，Naviが担当する。経路上の目的地・経由地における車両の向き（heading）は，自己位置の方位ではない（「目的地・経由地における車両の向き」の節を参照）。

車両が扱うアクティブな経路は常に1本である。複数経路の保存・管理は扱わない。

地図データの由来（商用地図・OpenStreetMap等）や経路計算のアルゴリズムは規定しない。地図データの版は，構成情報mapDataVersionで宣言される。

Naviが扱う座標（NaviGeoPointType）は，地図上の地点座標（目的地・経由地・経路の座標列等）である。その座標系はWGS84とし，構成情報coordinateSystemで宣言される。Naviは自車基準点の幾何位置を提供しないため，基準点（referencePoint）の宣言は行わない（現在位置の幾何情報はVehicle.CurrentLocationが担当する）。



経路の座標列や住所の判定結果の絶対精度は，地図データの整備品質に依存するため保証されない。

5.22.1.2. Naviの機能

Naviが提供する機能は，機能クラスによって宣言される。経路計算機能（機能クラスNaviRouting）を持たない車両でも，運転者がHMIから設定した経路・目的地の参照のみ，あるいは住所判定（機能クラスNaviAddress）・道路情報（機能クラスNaviRoadInfo）のみを提供する構成が可能である。

経路計算は非同期に行われる。setDestinationは経路計算の開始のみを行って直ちに返り，計算の完了はRouteCalculatedイベント，失敗はRouteCalculationFailedイベントで通知される。

経路の座標列・区間情報はデータ量が大きくなりうるため，状態には要約（routeSummary）のみを持ち，経路の全体はgetRouteで取得する。

5.22.1.3. 測定時刻とサンプル品質

状態には，測定時刻timestampとサンプル品質dataQualifierが含まれる。timestampは，車内共通時刻基盤に基づく測定時刻であり，送信時刻ではない。Naviでは，位置照合に由来する状態（routeSummary，roadInfo，address）の算出の基になった自車位置の測定時刻を基本とし，位置照合に由来する状態を持たないサンプルでは，timestampはサンプルの生成時刻を表す。同じ時刻基盤に基づく他のビークルAPIのタイムスタンプと直接比較できる。タイムスタンプ自体の誤差上限は，構成情報timestampAccuracyで宣言される。

dataQualifierは，取得できたサンプルの品質を表す。故障（ナビゲーション機能を提供できない状態。E_OBJECT_FAULTエラーおよびFaultDetectedイベント）が「位置照合の結果（有効なサンプル）を出力できない状態」を表すのに対し，dataQualifierは「出力されたサンプルがどれだけ信用できるか」を表す。両者の使い分けの詳細は，状態の節に規定する。

5.22.1.4. 状態の取得と更新

Naviにおける状態の更新とは，自車位置の更新に伴うrouteSummary・roadInfo・address等の再評価をいう。状態の更新周期は車両実装に依存する。状態は，要求応答（getStatus）により取得できるほか，状態報告イベント（StatusReported）によって周期的に得ることができる。routeSummary（残距離・残所要時間・遅延）は，自車位置の更新に伴って連続更新される信号系データであるため，その変化をきっかけに発生するイベントは定義しない。routeSummaryの値は，StatusReportedイベントの関連情報statusにより周期的に得られる。目的地・経路・行政区域・道路情報などの離散的な変化は，イベント（DestinationChanged，RouteChanged，AreaChanged，RoadInfoChanged等）で通知される。

5.22.1.5. 経由地の消化状態

経路に経由地が設定されている場合，各経由地を通過したか否かは，状態waypointsの各要素が持つ通過状態（未通過・通過済み）で表される。通過状態は，経路が設定された時点ではすべて未通過であり，経由地を通過すると通過済みになる。次に向かう経由地は，未通過の経由地のうち最初のものであり，未通

過の経由地が無い場合，次に向かう地点は目的地である。

経由地の通過は，WaypointPassedイベントによっても通知される。イベントは通過という離散遷移の発生のみを伝え，通過状態の値そのものは運ばない。値は，イベントを受けたアプリケーションがgetStatusから取得する。各経由地の通過状態は，StatusReportedイベントの関連情報statusによっても得られる。状態とイベントの双方を持つことにより，通過の時点で通知を受けていたアプリケーションは即応でき，その後に参照を開始したアプリケーションも，その時点の通過状態を取得できる。

通過状態は，経路の再計算（リルート）の範囲を定める役割も持つ。すなわち，リルートによって，既に通過した経由地へ引き返す経路が計算されることはない。再計算の対象の規定は，状態の節に示す。

残距離・残所要時間（routeSummary）が自車位置の更新に伴って連続的に変化する信号系のデータであるのに対し，経由地の通過状態は通過の確定時にのみ変化する離散的な状態である。両者は別の状態として提供される。

経由地の指定（setDestinationのwaypointsパラメータ）と，設定されている経由地の参照（状態waypoints）とでは，用いるデータ型が異なる。指定にはNaviDestinationType（位置・表示名）を，参照にはNaviWaypointProgressType（位置・表示名・通過状態等）を用いる。通過状態は案内の進行によって定まるものであり，アプリケーションが指定するものではない。

5.22.1.6. 目的地・経由地における車両の向き

Naviは，計算済みの経路上の目的地・経由地のそれぞれについて，その地点における車両の向きの計画値（heading）を提供する（機能クラスNaviRouteHeading）。headingは，車両がその地点を通過するとき，または次にその地点から発進するときの，車両の向きを表す。車両の向きとは車両の前方が向く方位であり，前進している車両では進行方位と一致する。headingは，自車の現在の進行方位（Vehicle.CurrentLocationの状態heading）ではない。値の表し方（単位はdeg，真北を0とし時計回り）はVehicle.CurrentLocationの状態headingと同じである。

目的地のheadingはgetRouteが返す地点列のうちkindがEndである要素で，経由地のheadingは状態waypointsの各要素（およびkindがViaである要素）で得られる。headingは，機能クラスNaviRouteHeadingを備え，かつその地点における車両の向きを提供できる場合にのみ値を持つ。



headingは計画値であり，その精度，および値が得られることは保証されない。また，その地点において車両が実際にその向きをとることを保証するものではない。

5.22.1.7. ナビゲーションに対する制御の調停

Naviに対する制御の調停には，標準的な調停機能（後勝ちの原則）が適用される。すなわち，経路が設定されている状態でsetDestinationが呼び出されると，既存の経路・目的地・経由地は上書きされる。

また，Naviは標準的なロック機能を持つLockableObjectである。目的地・経路はユーザの移動意図を表す重要な状態であり，ロックにより，他のアプリケーションからの無断の上書きを防ぐことができる。

5.22.1.8. ナビゲーションの操作・参照に伴うリスク

走行中に目的地・経由地・経路が変更されると，案内が急変して運転者の混乱・注意散漫を招くリスクがある。これをリスククラスRiskRouteChangeとして定義し，経路を変更するサービスコール（setDestination，cancelRouting）をリスク制御の対象とする。ロックの取得（lock）は経路変更の書き込み権を予約する操作であるため，同じくRiskRouteChangeの対象とする。

また，位置・住所・目的地は個人の行動を推定できる高感度な情報であり，これらの開示にはプライバシのリスクが伴う。これをリスククラスRiskPrivacyとして定義し，位置・状態を開示するサービスコール

（getStatus，getRoute，getAddress）をリスク制御の対象とする。イベントの通知要求（notify）により通知されるイベントには，報告時点の状態の全量を運ぶStatusReportedイベントが含まれることがあり，また，位置から派生するイベント（Arrived，WaypointPassed，AreaChanged等）の発生の観測からも位置情報を推定できる。このため，notifyの呼び出しそのものを，eventFilterで指定するイベントの種類によらず，RiskPrivacyの対象とする。

RiskRouteChange・RiskPrivacyのいずれについても，当該リスクに対応していないアプリケーションからの呼び出しはE_RISK_APPLICATIONエラー，ユーザが当該リスクを承認していない場合はE_RISK_USERエラーとなる。



経路の変更直後に急な案内（至近距離での右左折指示等）を出さないための工夫は，ナビゲーション実装側の責務であり，本APIでは規定しない。

5.22.2. 機能クラス

Naviは，以下の機能クラスを持つ。

NaviRouting	目的地を設定して経路を計算・案内することができる（setDestination，cancelRouting）
NaviIntermediateWaypoints	経路計算時に経由地を指定することができる（setDestinationのwaypointsパラメータ）
NaviPreference	経路探索の優先条件（最短時間・最短距離等）を指定することができる（setDestinationのpreferenceパラメータ）
NaviVehicleRestrictions	経路計算において車両の高さ・重量を考慮することができる（構成情報vehicleHeight，vehicleWeight）
NaviRouteDetails	経路の詳細（座標列・区間情報）を提供することができる（getRoute）
NaviGuidanceSteps	経路の区間ごとの案内ステップ（道路名・右左折指示等）を提供することができる（NaviRouteTypeのsegments）
NaviAddress	現在位置の住所（行政区域）を判定することができる（getAddress，状態address，AreaChangedイベント）
NaviRoadInfo	走行中の道路の種別・属性を判定することができる（状態roadInfo，RoadInfoChangedイベント）
NaviAvoidAreas	経路計算時に回避するエリア（回避エリア）を指定することができる（setDestinationのavoidAreasパラメータ）
NaviRouteHeading	経路上の目的地・経由地について，その地点における車両の向きの計画値を提供することができる（状態waypointsの各要素のheading，getRouteが返すwaypointsの各要素のheading）

5.22.3. リスククラス

Naviは，以下のリスククラスを持つ。

RiskRouteChange	走行中に目的地・経由地・経路が変更されると，案内が急変して運転者の混乱・注意散漫を招くリスク（例：変更直後に至近距離での右左折指示が発生する）

RiskPrivacy		位置・住所・道路情報など自車状態の開示に伴うプライバシのリスク（例：未対応アプリケーションへの位置の開示，ユーザが承認していない状態での開示）

5.22.4. 構成情報

Naviは，以下の構成情報を持つ。

capabilities	enumSet (NaviCapabilityType)	そのナビゲーションが持つ機能
riskClasses	enumSet(NaviRiskType)	そのナビゲーションのリスククラスの集合
supportedPreferences	enumSet (NaviPreferenceType)?	対応する経路探索の優先条件の集合。機能クラスNaviPreferenceを備えている場合にのみ値を持つ
supportedAvoidModes	enumSet (NaviAvoidModeType)?	対応する回避モード（Prohibit／Prefer）の集合。機能クラスNaviAvoidAreasを備えている場合にのみ値を持つ
supportedAvoidShapes	enumSet (NaviAvoidShapeKindType)?	対応する回避エリアの形状（ポリゴン・円・矩形）の集合。機能クラスNaviAvoidAreasを備えている場合にのみ値を持つ
vehicleHeight	Float32?	経路計算で考慮する車両の高さ。単位はm。機能クラスNaviVehicleRestrictionsを備えている場合にのみ値を持つ
vehicleWeight	Float32?	経路計算で考慮する車両の重量。単位はt。機能クラスNaviVehicleRestrictionsを備えている場合にのみ値を持つ
mapDataVersion	String?	地図データの版の識別文字列。地図データが更新されると値が変わる
coordinateSystem	String	位置（NaviGeoPointTypeの座標）の座標系の宣言（WGS84）
statusReportInterval	UInt32 ([1,4294967295])?	状態報告イベント（StatusReported）が発生する周期。単位はms。値を持つ場合は1以上。値なしの場合，この車両ではStatusReportedイベントは発生しない
timestampAccuracy	UInt32	この車両のタイムスタンプ誤差上限の宣言。単位はμs

5.22.5. 状態

Naviは，以下の状態を持つ。

timestamp	TimestampUsType	測定時刻。単位はμs。車内共通時刻基盤に基づく。dataQualifierがUnavailableのサンプルのtimestampは，サンプルの生成時刻を表す（測定時刻ではない）

dataQualifier	DataQualifierType	このサンプルの品質。InvalidまたはUnavailableの場合，位置照合に由来する状態（routeSummary，roadInfo，address）を読んではならない
mainStatus	NaviMainStatusType	案内の主状態
lockStatus	LockStatusType	ロック状態
destination	NaviDestinationType?	設定されている目的地
waypoints	NaviWaypointProgressType[]?	設定されている経由地の列（通過順）。各要素は，位置・表示名と通過状態（未通過・通過済み），およびその経由地における車両の向きを持つ
routeSummary	NaviRouteSummaryType?	経路の要約（残距離・残所要時間・遅延）
roadInfo	NaviRoadInfoType?	走行中の道路の情報（種別・属性）
address	NaviAddressType?	現在位置の住所（行政区域）

destinationおよびwaypointsは，経路が設定されている場合（mainStatusがCalculating，Guiding，Rerouting，Arrivedのいずれかの場合）にのみ値を持つ。waypointsは，経由地が設定されていない場合には空の配列となる。

waypointsの各要素は，経由地の位置・表示名と，その経由地の通過状態（state），およびその経由地における車両の向き（heading）を持つ。headingについては「目的地・経由地における車両の向き」の節に規定する。通過状態は，setDestinationにより経路が設定された時点ではすべてNotPassed（未通過）であり，当該経由地の通過が確定するとPassed（通過済み）となる。Passedとなった経由地の通過状態は，同一の経路設定の下でNotPassedに戻ることはない。経由地は先頭から順に通過され，先行するすべての経由地がPassedとなる前に，後続の経由地がPassedとなることはない。次に向かう経由地は，通過状態がNotPassedである最初の要素であり，NotPassedの要素が無い場合，次に向かう地点はdestinationである。通過の確定条件は，イベントの節に規定する。

リルート（mainStatusがRerouting）では，経由地の集合は変化せず，各経由地の通過状態も維持される。経路の再計算の対象となるのは，通過状態がNotPassedである経由地と目的地であり，通過状態がPassedである経由地は再計算の対象に含まれない。したがって，リルートによって，既に通過した経由地へ引き返す経路が計算されることはない。

routeSummaryは，経路計算が完了している場合（mainStatusがGuiding，Rerouting，Arrivedのいずれかの場合）にのみ値を持つ。経路計算中（Calculating）は経路が確定していないため，値なしとなる。リルート中（Rerouting）は，再計算開始前の経路に基づく値を保持する。

mainStatusがNoRouteまたはFaultの場合，destination，waypoints，routeSummaryはいずれも値なしとなる。

故障（mainStatusがFault）から回復した場合，回復の時点で経路が保持されていれば，destination，waypointsは再び値を持ち，各経由地の通過状態も故障前の値が維持される。回復の時点で経路が保持されていない場合は，mainStatusはNoRouteとなる。故障中に経路を保持するか否かは実装依存であり，本APIでは規定しない。

roadInfoは，機能クラスNaviRoadInfoを備え，かつ走行中の道路を特定（マップマッチング）できている場合にのみ値を持つ。addressは，機能クラスNaviAddressを備え，かつ現在位置の行政区域を判定できている場合にのみ値を持つ。

経路の座標列・区間情報などの経路の全体は，データ量が大きくなりうるため状態には含めず，getRoute

で取得する。

timestampは，このサンプルの測定時刻であり，車内共通時刻基盤に基づく。位置照合に由来する状態（routeSummary，roadInfo，address）を1つ以上持つサンプルでは，それらの算出の基になった自車位置の測定時刻を表す。位置照合に由来する状態を1つも持たないサンプル（mainStatusがNoRouteであってroadInfo・addressがいずれも値なしの場合や，dataQualifierがUnavailableの場合等）では，timestampはサンプルの生成時刻を表す（測定時刻ではない）。dataQualifierがInvalidのサンプルでは，位置照合に由来する状態が形式上値を持っていても算出の基になった自車位置を確定できないことがあるため，timestampは位置照合が使用した自車位置の測定時刻またはサンプルの生成時刻を表す。タイムスタンプの誤差上限は，構成情報timestampAccuracyで宣言される。

dataQualifierは，このサンプルの品質を表す。Invalidは，位置照合（マップマッチング）自身がその結果を信用できない状態（起動直後の初期化中・自車位置の確定前など）であり，位置照合に由来する状態（routeSummary，roadInfo，address）を読んではならない。mainStatus，lockStatus，destination，waypoints（各経由地の通過状態およびheadingを含む）は位置照合に由来しないため，Invalidの場合も読むことができる。Degradedは，値は使えるが通常品質でない状態（例：自車位置の品質低下により位置照合の信頼度が低下している場合）である。

ナビゲーション機能を提供できない状態（地図データの破損等）は故障と扱い，故障中の状態の参照（getStatus）はE_OBJECT_FAULTエラーとなる。故障中にStatusReportedイベントで報告される状態は，dataQualifierがUnavailableであり，そのtimestampはサンプルの生成時刻を表す（測定時刻ではない）。Unavailableのサンプルでは，位置照合に由来する状態を読んではならない（最終既知値として読むことも許されない）。

5.22.6. サービスコール一覧

Naviに対するサービスコールは以下の通り。

getConfig	ナビゲーションの構成情報の参照
getStatus	ナビゲーションの状態の参照
setDestination	目的地の設定と経路計算の開始
cancelRouting	経路案内の中止
getRoute	経路の全体の参照
getAddress	現在位置の住所（行政区域）の参照
lock	ナビゲーションのロックの取得
unlock	ナビゲーションのロックの解除
notify	ナビゲーションイベントの通知要求
unnotify	ナビゲーションイベントの通知停止
getEvent	ナビゲーションイベントの取得

5.22.7. イベント

Naviに対するイベント種別には，LockableObjectの持つイベントに加えて，経路計算・案内・住所・道路情報に関するイベント，および状態の報告に関するイベント（StatusReported）がある。

イベントの関連情報には発生原因などの最小限の情報のみを含め，経路・住所等のデータ本体は含めない。データ本体は，イベントを受けたアプリケーションがgetRoute，getStatus，getAddressにより取得

する。ただし，StatusReportedイベントは，報告時点の状態の全量（NaviStatusType）を関連情報statusとして含む。

Naviに対するイベント種別は以下の通り。

RouteCalculated	経路計算が完了し，案内を開始
RouteCalculationFailed	経路計算に失敗
RouteChanged	経路の変化（リルート・setDestinationによる経路の再設定等）
DestinationChanged	目的地の変化
WaypointsChanged	経由地の変化
Arrived	目的地に到着
AreaChanged	現在位置の行政区域の変化（国・都道府県・市区町村・区の境界越え）
RoadInfoChanged	走行中の道路の情報（種別・属性）の変化
FaultDetected	故障検知（地図データの破損等によりナビゲーション機能を提供できない状態）
FaultRecovered	故障から回復
WaypointPassed	経由地の通過（通過が確定した経由地ごとに1回発生する）
StatusReported	状態の報告（構成情報statusReportIntervalの周期ごとに発生する）
OwnLockRevoked	自アプリケーションのロックの強制解除
OtherLockReleased	他のアプリケーション等によるロック解除
EventOverflow	イベントキューのオーバフロー
ConfigChanged	構成情報の変更

WaypointPassedイベントは，設定されている経由地の通過が確定したときに，通過が確定した経由地ごとに1回発生する。通過が確定するのは，自車位置を経路上に投影した進行位置が，当該経由地に対応する経路上の地点に到達し，これを経路の進行方向に越えた場合である。後退または逆走により同じ地点を逆向きに越えた場合には発生しない。

通過の判定は，dataQualifierがInvalidまたはUnavailableでないサンプル（以下，有効なサンプルという）に基づく場合にのみ確定する。また，経路からの逸脱等により自車位置を経路上の一つの地点に投影できない場合，経路上への投影は有効でなく，投影が有効でない間は，通過は確定しない。

通過を確定できない期間（有効でないサンプルが続いた期間，または経路上への投影が有効でない期間）が生じた場合，その後の最初の有効なサンプルであって経路上への投影が有効であるものにおいて，進行位置が経由地に対応する経路上の地点を経路の進行方向に越えていることが確認された経由地は，その時点で通過が確定する。この場合，複数の経由地の通過が同一のサンプルで確定することがある。

近接する複数の経由地の通過が同一のサンプルで確定した場合は，通過順にそれぞれのイベントが発生する。本イベントの発生により当該経由地の通過状態はNotPassedからPassedに遷移し，同一の経路設定の下でNotPassedに戻ることはないため，同一の経路設定の下では，同じ経由地について本イベントが再び発生することはない。

WaypointPassedイベントは，通過という離散遷移の発生のみを伝え，通過状態の値そのものは運ばない。イベントを受けたアプリケーションは，getStatusにより，その時点の各経由地の通過状態（状態waypoints）を取得する。通過状態は，StatusReportedイベントの関連情報statusによっても得られる。通過状態の変化は経由地の集合の変化ではないため，本イベントによってWaypointsChangedイベントが

発生することはない。目的地への到着はArrivedイベントで通知され，本イベントの対象ではない。最後の経由地の通過と目的地への到着が同一のサンプルで確定した場合は，WaypointPassedイベントの後にArrivedイベントが発生する。

StatusReportedイベントの関連情報statusは，報告時点のナビゲーションの状態の全量であり，getStatusが返す状態と同じ型（NaviStatusType）である。関連情報timestamp（TimestampMsType）と，状態が持つtimestamp（TimestampUsType）とは単位が異なり，起点が同一であることも保証されないため，互いに比較することはできない。状態の更新周期と構成情報statusReportIntervalとは一致するとは限らない。timestampが測定時刻を表すサンプル（位置照合に由来する状態を1つ以上持ち，dataQualifierがInvalidまたはUnavailableでないもの）どうしでは，連続する報告の関連情報statusが同じ位置照合の結果に基づくか否かを，statusのtimestampにより判断できる。それ以外のサンプル（timestampが生成時刻を表しうるもの）では，これにより判断することはできない。ただし，同じ位置照合の結果に基づく報告であっても，経路の変化（RouteChangedイベント）により，routeSummaryの値は異なりうる。

この内，イベント種別がRouteCalculationFailedの場合には，以下の関連情報を持つ。

reason	NaviRouteFailReasonType	失敗の原因

また，イベント種別がRouteChangedの場合には，以下の関連情報を持つ。

sourceApplication	ApplicationIdType?	変更を行ったアプリケーション。リルート等のシステム起因の変更や車両のHMIからの操作の場合は値なし

また，イベント種別がDestinationChangedの場合には，以下の関連情報を持つ。

sourceApplication	ApplicationIdType?	変更を行ったアプリケーション。車両のHMIからの操作の場合は値なし

また，イベント種別がWaypointsChangedの場合には，以下の関連情報を持つ。

sourceApplication	ApplicationIdType?	変更を行ったアプリケーション。車両のHMIからの操作の場合は値なし

また，イベント種別がAreaChangedの場合には，以下の関連情報を持つ。

level	NaviAreaLevelType	越えた境界のレベル。複数レベルの境界を同時に越えた場合は最も上位のレベル。新しい住所はgetStatusまたはgetAddressで取得する

また，イベント種別がStatusReportedの場合には，以下の関連情報を持つ。

timestamp	TimestampMsType	報告が発生した時刻
status	NaviStatusType	報告時点のナビゲーションの状態（全量）

また，イベント種別がEventOverflowの場合には，以下の関連情報を持つ。

noLostEvents	UInt32	失われたイベントの数

5.22.8. データ型の定義

5.22.8.1. Naviの機能クラス

```
type NaviCapabilityType = enum {
    NaviRouting,
    NaviIntermediateWaypoints,
    NaviPreference,
    NaviVehicleRestrictions,
    NaviRouteDetails,
    NaviGuidanceSteps,
    NaviAddress,
    NaviRoadInfo,
    NaviAvoidAreas,
    NaviRouteHeading
};
```

5.22.8.2. Naviに対するリスククラス

```
type NaviRiskType = enum {
    RiskRouteChange,
    RiskPrivacy
};
```

5.22.8.3. 地理座標

```
type NaviGeoPointType = struct {
    latitude : Float64 ([-90.0,90.0]),  // 緯度。単位はdeg（WGS84）。北緯が正
    longitude : Float64 ([-180.0,180.0]),   // 経度。単位はdeg（WGS84）。東経が正
    altitude : Float64? // 高度。単位はm（WGS84楕円体高）。値を提供できない場合は値な
し
};
```

地理座標（NaviGeoPointType）は，複数のオブジェクトで共通に用いうる型であるが，本版ではNaviの中で定義する。

座標系はWGS84とし，構成情報coordinateSystemで宣言される。

5.22.8.4. Naviの構成情報（getConfigが返すデータ型）

```
type NaviConfigType = struct {
    capabilities : enumSet(NaviCapabilityType),
    riskClasses : enumSet(NaviRiskType),
    supportedPreferences : enumSet(NaviPreferenceType)?,
    supportedAvoidModes : enumSet(NaviAvoidModeType)?,
    supportedAvoidShapes : enumSet(NaviAvoidShapeKindType)?,
    vehicleHeight : Float32?,
```

```
    vehicleWeight : Float32?,
    mapDataVersion : String?,
    coordinateSystem : String,
    statusReportInterval : UInt32 ([1,4294967295])?,
    timestampAccuracy : UInt32
};
```

5.22.8.5. 目的地と経由地

目的地および経由地の指定にはNaviDestinationType（位置・表示名）を用いる。設定されている経由地の参照にはNaviWaypointProgressType（位置・表示名・通過状態・車両の向き）を用い，通過状態はNaviWaypointStateTypeで表す。

経路の地点列（始点・経由地・終点の列）にはNaviWaypointTypeを用い，各地点の種別はNaviWaypointKindTypeで表す。

```
type NaviDestinationType = struct {
    position : NaviGeoPointType,    // 位置（緯度経度）
    name : String?  // 表示名
};
```

```
type NaviWaypointStateType = enum {
    NotPassed,  // 未通過（これから向かう経由地）
    Passed  // 通過済み
};
```

```
type NaviWaypointProgressType = struct {
    position : NaviGeoPointType,    // 位置（緯度経度）
    heading : Float64 ([0.0,360.0))?,   // この経由地における車両の向きの計画値。「目的地・経由地における車両の向き」の節を参照
    name : String?, // 表示名
    state : NaviWaypointStateType   // 通過状態
};
```

通過状態は案内の進行によって，車両の向き（heading）は計算された経路によって定まるものであり，いずれも指定できるものではないため，指定に用いる型（NaviDestinationType）はこれらを持たない。

```
type NaviWaypointKindType = enum {
    Start,  // 始点（経路の計算を開始した地点）
    Via,    // 経由地
    End // 終点（目的地）
};
```

```
type NaviWaypointType = struct {
```

```
    kind : NaviWaypointKindType,    // 地点の種別
    position : NaviGeoPointType,    // 位置（緯度経度）
    heading : Float64 ([0.0,360.0))?,   // その地点における車両の向きの計画値。kindがStartの地点では値なし。「目的地・経由地における車両の向き」の節を参照
    name : String?  // 表示名
};
```

5.22.8.6. 経路

```
type NaviPreferenceType = enum {
    Fastest,    // 最短時間の経路を優先
    Shortest    // 最短距離の経路を優先
};
```

```
type NaviInstructionType = enum {
    Continue,   // 直進（そのまま進む）
    TurnLeft,   // 左折
    TurnRight,  // 右折
    Other   // その他の案内
};
```

```
type NaviSegmentType = struct {
    endPosition : NaviGeoPointType, // 区間の終端の位置
    length : UInt32,    // 区間の距離。単位はm
    duration : UInt32?, // 区間の所要時間の推定値。単位はs
    roadName : String?, // 区間の道路名
    speedLimit : UInt16 ([1,300] | UNKNOWN(0)), // 区間の制限速度。単位はkm/h固定
    instruction : NaviInstructionType?  // 区間の終端における案内指示
};
```

```
type NaviRouteType = struct {
    destination : NaviDestinationType,  // 目的地（経路の定義情報）
    waypoints : NaviWaypointType[], // 始点・経由地・終点の列（通過順）。経路の定義情報であり，経由地の通過状態は含まない
    preference : NaviPreferenceType?,   // 経路探索の優先条件（経路の定義情報）
    avoidAreas : NaviAvoidAreaType[]?,  // 経路計算時に指定された回避エリアの列（経路の定義情報）。機能クラスNaviAvoidAreasを備え，かつ回避エリアが指定された経路である場合にのみ値を持つ
    geometry : NaviGeoPointType[],  // 経路の座標列
    totalLength : UInt32 ([0,4294967294] | UNKNOWN(4294967295)),    // 経路全体の距離。単位はm
    totalDuration : UInt32?,    // 経路全体の所要時間の推定値。単位はs
    segments : NaviSegmentType[]?   // 区間情報の列。機能クラスNaviGuidanceStepsを備えている場合にのみ値を持つ
};
```

```
type NaviRouteSummaryType = struct {
    remainingLength : UInt32,   // 目的地までの残距離。単位はm
    remainingDuration : UInt32?,    // 目的地までの残所要時間の推定値。単位はs
    delay : Int32?  // 交通状況等による遅延の推定値。単位はs。負値は早着を表す
};
```

到着予定は，絶対時刻（ETA）ではなく残所要時間（remainingDuration）で表す。所要時間・遅延は交通状況に依存する推定値であり，保証されない。

5.22.8.7. 回避エリア

```
type NaviAvoidModeType = enum {
    Prohibit,   // 禁止（ハード回避）。当該エリアを通過する経路を返してはならない。回避した経路が存在しない場合，経路計算は失敗する
    Prefer  // 回避推奨（ソフト回避）。当該エリアをできるだけ避けるが，他に経路がなければ通過してよい。回避しきれない場合も経路計算は失敗しない
};
```

回避モードは，禁止（Prohibit）と回避推奨（Prefer）の二値である。ProhibitとPreferの本質的な違いは，回避しきれない場合の挙動にある。Prohibitは，当該エリアを通過しない経路が存在しなければ経路計算を失敗させる（明示的失敗）。Preferは，当該エリアを避けられない場合でも当該エリアを通過する経路を返し，経路計算を失敗させない。回避の強さは数値では表さない。

```
type NaviAvoidShapeKindType = enum {
    Polygon,    // 多角形（外周点列）
    Circle, // 円（中心と半径）
    Rectangle   // 矩形（南西隅と北東隅）
};
```

```
type NaviGeoPolygonType = struct {
    exterior : NaviGeoPointType[]   // 多角形の外周を成す点列。3点以上を指定する。先頭と末尾の点は自動的に連結され，閉じた領域を成す
};
```

```
type NaviGeoCircleType = struct {
    center : NaviGeoPointType,  // 円の中心の位置（緯度経度）
    radius : Float64    // 円の半径。単位はm。0より大きい値。上限は実装依存であり，範囲外はE_INVALID_PARAMETERとなる
};
```

```
type NaviGeoRectangleType = struct {
    southwest : NaviGeoPointType,   // 矩形の南西隅の位置（緯度経度）
    northeast : NaviGeoPointType    // 矩形の北東隅の位置（緯度経度）
};
```

```
type NaviAvoidShapeType = variant(NaviAvoidShapeKindType) {
    Polygon {
        polygon : NaviGeoPolygonType
    },
    Circle {
        circle : NaviGeoCircleType
    },
    Rectangle {
        rectangle : NaviGeoRectangleType
    }
};
```

回避エリアの形状は，多角形・円・矩形のいずれかである。形状の種別はNaviAvoidShapeKindTypeで判別され，種別ごとに対応する形状情報を持つ。対応する形状は，構成情報supportedAvoidShapesで車両ごとに宣言される。座標系はWGS84とし，構成情報coordinateSystemで宣言される。

```
type NaviAvoidAreaType = struct {
    shape : NaviAvoidShapeType, // 回避エリアの形状（多角形・円・矩形のいずれか）
    mode : NaviAvoidModeType    // 回避モード（禁止／回避推奨）。省略できない
};
```

回避エリアは，形状（shape）と回避モード（mode）の対である。回避モードはエリアごとに指定し，同一の経路計算のなかで禁止エリアと回避推奨エリアを混在させることができる。modeは省略できず，既定値は設けない。

5.22.8.8. 道路情報

```
type NaviRoadClassType = enum {
    Motorway,   // 高速道路（日本では高速自動車国道が典型）
    Trunk,  // 最上位の幹線道路（日本では国道が典型）
    Primary,    // 主要な幹線道路（日本では主要地方道が典型）
    Secondary,  // 幹線道路（日本では一般都道府県道が典型）
    Tertiary,   // 地域の主要道路（日本では市町村道の主要路線が典型）
    Unclassified,   // 上記に該当しない一般道路
    Residential,    // 住宅地内の生活道路
    Service,    // 施設内・敷地内の通路（駐車場内の通路等）
    LivingStreet,   // 歩行者優先の道路
    Other,  // その他（自動車の通常走行を想定しない道路等）
    Unknown // 不明
};
```

道路の機能分類は，OpenStreetMapのhighwayタグの語彙に基づく国際的な分類を用い，日本の法的種別との対応はあくまで典型例として示す。地図データが未知の分類を持つ場合はUnknownと扱う。

```
type NaviRoadInfoType = struct {
    roadClass : NaviRoadClassType,  // 道路の機能分類
    speedLimit : UInt16 ([1,300] | UNKNOWN(0)), // 制限速度。単位はkm/h固定
```

```
    isRamp : Bool?, // ランプ・連結路か
    isMotorroad : Bool?,    // 自動車専用道路か
    inSpeedZone : Bool? // 速度規制ゾーン（ゾーン30等）の内か
};
```

「高速道か一般道か」の二値判定は，roadClassがMotorwayであるか，またはisMotorroadが真であるかによって導出できる。

速度規制ゾーン（ゾーン30等）は道路の機能分類と直交する属性であり，inSpeedZoneで表す。ゾーンの規制速度はspeedLimitが持つ。

制限速度の単位はkm/hに固定し，単位フィールドは持たない。マイル表示圏の道路では換算して返す。

5.22.8.9. 住所（行政区域）

```
type NaviAddressType = struct {
    countryCode : String,   // 国コード（ISO 3166-1 alpha-2。例：JP）
    country : String?,  // 国名
    prefecture : String?,   // 都道府県名
    prefectureCode : String?,   // 都道府県コード（ISO 3166-2。例：JP-23）
    county : String?,   // 郡・振興局名
    municipality : String?, // 市区町村名（特別区を含む）
    municipalityCode : String?, // 全国地方公共団体コード（JIS X 0402）
    ward : String?, // 政令指定都市の区名
    district : String?  // 町域名（大字・町字）
};
```

住所は，全粒度をオプショナルフィールドで持つ固定の構造体で表し，利用側が必要な粒度のフィールドのみを読む。地図データから得られない粒度のフィールドは値なしとなる。

市区町村の境界越え判定などの同一性比較には，名称の文字列比較ではなく，municipalityCode（JIS X 0402）の比較を用いることを推奨する。

各名称の文字列は，多言語対応の規定に従い，現在のロケールに対応する呼称を返す。

```
type NaviAreaLevelType = enum {
    Country,    // 国
    Prefecture, // 都道府県
    Municipality,   // 市区町村
    Ward    // 政令指定都市の区
};
```

5.22.8.10. タイムスタンプ

```
type TimestampUsType = UInt64;  // 車内共通時刻基盤の起点を基準とする測定時刻。単位はμs。複数のビークルAPIで共通に用いる共通データ型であり，本仕様内に定義する。
```

5.22.8.11. サンプル品質の種別

```
type DataQualifierType = enum {
    Normal, // 正常
    Unavailable,    // 利用不可
    Degraded,   // 劣化
    TestMode,   // テストモード
    Invalid // 無効（値も誤差も読んではならない）
};
```

5.22.8.12. Navi状態

```
type NaviMainStatusType = enum {
    NoRoute,    // 経路未設定
    Calculating,    // 経路計算中
    Guiding,    // 案内中
    Rerouting,  // 経路再計算中（リルート）
    Arrived,    // 目的地到着
    Fault   // 故障中
};
```

目的地に到着するとmainStatusはArrivedに遷移し，Arrivedイベントが発生する。その後，ナビゲーション実装の定める条件（一定時間の経過またはユーザ操作等）によりNoRouteに遷移し，経路・目的地・経由地は破棄される。ArrivedからNoRouteへの遷移条件は実装依存であり，本APIでは規定しない。

案内は，未通過の経由地を通過するように行われるが，通過状態がNotPassedである経由地が残っている場合であっても，目的地への到着をもってmainStatusはArrivedに遷移する。この場合，残った経由地の通過状態はNotPassedのまま保持されるため，アプリケーションは，どの経由地を通過しなかったかを知ることができる。

```
type NaviStatusType = struct {
    timestamp : TimestampUsType,
    dataQualifier : DataQualifierType,
    mainStatus : NaviMainStatusType,
    lockStatus : LockStatusType,
    destination : NaviDestinationType?,
    waypoints : NaviWaypointProgressType[]?,
    routeSummary : NaviRouteSummaryType?,
    roadInfo : NaviRoadInfoType?,
    address : NaviAddressType?
};
```

destination，waypointsは経路が設定されている場合（mainStatusがCalculating，Guiding，Rerouting，Arrivedのいずれかの場合）に，routeSummaryは経路計算が完了している場合（mainStatusがGuiding，Rerouting，Arrivedのいずれかの場合）にのみ値を持つ。mainStatusがNoRouteまたはFaultの場合，これらはいずれも値なしとなる。waypointsの各要素は，経由地の位置・表示名と通過状態（NotPassed，Passed），およびその経由地における車両の向き（heading）を持ち，通過状態は通過の確

定時にのみ変化する。roadInfo，addressは，それぞれ機能クラスNaviRoadInfo，NaviAddressを備え，かつ判定できている場合にのみ値を持つ。

timestampは，車内共通時刻基盤に基づく測定時刻を表す。位置照合に由来する状態（routeSummary，roadInfo，address）を持つサンプルではそれらの算出の基になった自車位置の測定時刻を，位置照合に由来する状態を持たないサンプルではサンプルの生成時刻を，dataQualifierがInvalidのサンプルでは位置照合が使用した自車位置の測定時刻またはサンプルの生成時刻を表す。dataQualifierがInvalidまたはUnavailableの場合，位置照合に由来する状態（routeSummary，roadInfo，address）を読んではならない。

5.22.8.13. Naviイベント

```
type NaviRouteFailReasonType = enum {
    NoRouteFound,   // 到達可能な経路が見つからない
    MapUnavailable, // 地図データが利用できない
    AvoidAreaInfeasible,    // 禁止（Prohibit）の回避エリアを回避する経路が存在しない
    Other   // その他の原因
};
```

```
type NaviEventKindType = enum {
    RouteCalculated,
    RouteCalculationFailed,
    RouteChanged,
    DestinationChanged,
    WaypointsChanged,
    Arrived,
    AreaChanged,
    RoadInfoChanged,
    FaultDetected,
    FaultRecovered,
    WaypointPassed,
    StatusReported,
    OwnLockRevoked,
    OtherLockReleased,
    EventOverflow,
    ConfigChanged
};
```

```
type NaviEventType = struct {
    eventInfo : variant(NaviEventKindType) {
        RouteCalculationFailed {
            reason : NaviRouteFailReasonType
        },
        RouteChanged {
            sourceApplication : ApplicationIdType?
        },
        DestinationChanged {
            sourceApplication : ApplicationIdType?
```

```
        },
        WaypointsChanged {
            sourceApplication : ApplicationIdType?
        },
        AreaChanged {
            level : NaviAreaLevelType
        },
        StatusReported {
            timestamp : TimestampMsType ,
            status : NaviStatusType
        },
        EventOverflow {
            noLostEvents : UInt32
        }
    }
};
```

5.22.9. サービスコールの詳細規定

getConfig

ナビゲーションの構成情報の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
config	NaviConfigType	ナビゲーションの構成情報

【エラーコード】

なし

【機能】

ナビゲーションに関する構成情報を返す。

getStatus

ナビゲーションの状態の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

status	NaviStatusType	ナビゲーションの状態

【エラーコード】

E_RISK_APPLICATION	アプリケーションが対応していないリスクがある
E_RISK_USER	ユーザがリスクを承認していない
E_OBJECT_FAULT	故障中（ナビゲーション機能を提供できない状態）

【機能】

ナビゲーションに関するすべての状態を返す。取得した状態の信頼度は，併せて返されるdataQualifierによって判断する。dataQualifierがInvalidの場合，位置照合に由来する状態（routeSummary，roadInfo，address）を読んではならない。

状態には目的地・住所などの高感度な情報が含まれるため，本サービスコールはリスククラスRiskPrivacyの対象であり，RiskPrivacyに対応していないアプリケーションはE_RISK_APPLICATIONエラー，ユーザが状態の開示を承認していない場合はE_RISK_USERエラーとなる。

setDestination

目的地の設定と経路計算の開始

【パラメータ】

destination	NaviDestinationType	目的地
waypoints	NaviDestinationType[]?	経由地の列（通過順）
preference	NaviPreferenceType?	経路探索の優先条件。省略した場合はFastest
avoidAreas	NaviAvoidAreaType[]?	経路計算時に回避するエリアの列。各要素は形状（多角形・円・矩形）と回避モード（禁止／回避推奨）を持つ。省略した場合は回避エリアなし
priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_PARAMETER	destination，waypoints，preference，avoidAreas，priorityが有効範囲外（回避エリアの形状の不正，個数・半径・頂点数が実装の許容範囲外，対応していない形状・回避モードの指定を含む）
E_RISK_APPLICATION	アプリケーションが対応していないリスクがある
E_RISK_USER	ユーザがリスクを承認していない
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_INFEASIBLE	到達可能な経路がないと即時に判定できる場合（禁止（Prohibit）の回避エリアを回避する経路が存在しないと即時に判定できる場合を含む）

E_OBJECT_FAULT	ナビゲーションが故障中
E_OBJECT_LOCKED	ナビゲーションが，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている

【機能】

目的地をdestinationに設定し，経路計算を開始する。本サービスコールは経路計算の開始のみを行って直ちに返り，計算の完了はRouteCalculatedイベント，失敗はRouteCalculationFailedイベントで通知される。

本サービスコールが成功すると，destination・waypointsが更新され，すべての経由地の通過状態はNotPassed（未通過）となってmainStatusはCalculatingに遷移し，DestinationChangedイベントが発生する（waypointsを指定した場合や，経由地つきの既存経路を経由地なしで上書きした場合など，経由地の集合が変化する場合にはWaypointsChangedイベントも発生する）。経路が設定されている状態で呼び出された場合，後勝ちの原則により，既存の経路・目的地・経由地はこの時点で上書きされる。

計算が完了すると案内を開始し（mainStatusはGuidingに遷移する），RouteCalculatedイベントおよびRouteChangedイベントが発生する。計算に失敗した場合はRouteCalculationFailedイベントが発生し，目的地・経由地は破棄されてmainStatusはNoRouteに遷移する（呼び出し前に既存の経路が設定されていた場合も，元の経路は復元されない）。

waypointsには通過順に経由地を指定する（機能クラスNaviIntermediateWaypoints）。preferenceには経路探索の優先条件を指定する（機能クラスNaviPreference。対応する条件は構成情報supportedPreferencesで宣言される）。

avoidAreasには，経路計算時に回避するエリアの列を指定する（機能クラスNaviAvoidAreas）。各回避エリアは形状（多角形・円・矩形）と回避モード（mode）を持つ。回避モードには禁止（Prohibit）と回避推奨（Prefer）があり，各エリアに個別に指定する。禁止（Prohibit）のエリアは，これを通過しない経路が計算される。禁止のエリアを回避する経路が存在しない場合，経路計算は失敗する。すなわち，これを即時に判定できる場合はE_INFEASIBLEエラーを返し，経路計算中に判明した場合はRouteCalculationFailedイベント（reasonはAvoidAreaInfeasible）が発生する。回避推奨（Prefer）のエリアは，これをできるだけ避ける経路が計算されるが，他に経路がない場合はこれを通過する経路を返し，経路計算は失敗しない。対応する形状は構成情報supportedAvoidShapesで，対応する回避モードは構成情報supportedAvoidModesで宣言される。回避エリアの個数・半径・頂点数の上限はAPIでは規定せず，実装の許容範囲外の指定はE_INVALID_PARAMETERエラーとなる。回避エリアの指定は，禁止（Prohibit）であっても，地図データに反映された道路網の範囲での回避を表すものであり，地図データに未反映の実世界の通行可否までを保証するものではない。

機能クラスNaviVehicleRestrictionsを備えている場合，構成情報vehicleHeight，vehicleWeightを考慮した経路を計算する。

指定された座標は，マップマッチングにより最寄りの道路上の地点に丸められることがある。

機能クラスNaviRoutingを持たない車両では，本サービスコールはE_NOT_SUPPORTEDエラーとなる。機能クラスNaviIntermediateWaypointsを持たない車両に対してwaypointsを指定した場合，本サービスコールはE_NOT_SUPPORTEDエラーとなる（経由地を反映しない経路を返すことはしない）。機能クラスNaviAvoidAreasを持たない車両に対してavoidAreasを指定した場合，本サービスコールはE_NOT_SUPPORTEDエラーとなる（回避エリアを反映しない経路を返すことはしない）。

走行中の経路の変更はリスククラスRiskRouteChangeの対象であり，対応できていない場合には，E_RISK_APPLICATIONエラーまたはE_RISK_USERエラーとなる。

cancelRouting

経路案内の中止

【パラメータ】

priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_PARAMETER	priorityが有効範囲外
E_RISK_APPLICATION	アプリケーションが対応していないリスクがある
E_RISK_USER	ユーザがリスクを承認していない
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_FAULT	ナビゲーションが故障中
E_OBJECT_STATUS	経路が設定されていない（mainStatusがNoRoute）
E_OBJECT_LOCKED	ナビゲーションが，他のアプリケーションにより，priorityと同じかそれより高い優先度でロックされている

【機能】

経路案内を中止し，設定されている経路・目的地・経由地を破棄する（mainStatusはNoRouteに遷移する）。中止するとDestinationChangedイベントおよびRouteChangedイベントが発生する（経由地が設定されていた場合は，その破棄により経由地の集合が変化するため，WaypointsChangedイベントも発生する）。イベントを受けたアプリケーションは，mainStatusがNoRouteに遷移していること（destination・waypoints・routeSummaryが値なしであること）により，経路が破棄されたことを知ることができる。

機能クラスNaviRoutingを持たない車両では，本サービスコールはE_NOT_SUPPORTEDエラーとなる。

走行中の案内の中止も案内の急変となるため，リスククラスRiskRouteChangeの対象であり，対応できていない場合には，E_RISK_APPLICATIONエラーまたはE_RISK_USERエラーとなる。

getRoute

経路の全体の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
route	NaviRouteType	設定されている経路の全体

【エラーコード】

E_NO_DATA	経路が設定されていない（mainStatusがNoRoute），または経路の計算が完了していない（mainStatusがCalculating）
E_RISK_APPLICATION	アプリケーションが対応していないリスクがある
E_RISK_USER	ユーザがリスクを承認していない
E_OBJECT_FAULT	ナビゲーションが故障中

【機能】

設定されている経路の全体を返す。経路には，定義情報（destination，waypoints，preference，avoidAreas），経路の座標列（geometry），距離・所要時間，および区間情報（segments）が含まれる。avoidAreasには，経路計算時に指定された回避エリアの列がそのまま返される（機能クラスNaviAvoidAreasを備え，かつ回避エリアが指定された経路である場合）。目的地・経由地における車両の向きは，waypointsのうちkindがEnd，Viaである要素のheadingで得られる（「目的地・経由地における車両の向き」の節を参照）。リルート中（mainStatusがRerouting）は，再計算開始前の経路を返す。

本サービスコールが返す経路は計画を表すものであり，経由地の通過状態は含まない。返されるwaypointsは，設定されているすべての経由地を，通過済みのものを含めて通過順に列挙する。したがって，kindがViaである要素は，状態waypointsの要素と，同じ順序で1対1に対応する。各経由地の通過状態は，状態waypointsで参照する。

リルートによって再計算されるのは経路の線形であり，経由地の列は変化しない。kindがStartである要素は，その経路の計算を開始した地点を表す。すなわち，リルートにより計算された経路では，kindがStartである要素は再計算を開始した地点であり，経路を最初に計算したときの出発地ではない。

リルートにより計算された経路のgeometryは，通過済みの経由地を通らない。すなわち，返されるwaypointsのうち通過状態がPassedである経由地は，経路の定義情報として列挙されるが，geometry上の地点に対応しないことがある。geometryに沿って経路を描画する場合は，waypointsの各地点を順に結ぶのではなく，geometryを用いる。

segmentsは，機能クラスNaviGuidanceStepsを備えている場合にのみ値を持つ。

機能クラスNaviRouteDetailsを持たない車両では，本サービスコールはE_NOT_SUPPORTEDエラーとなる。

経路には目的地・走行予定経路などの高感度な情報が含まれるため，本サービスコールはリスククラスRiskPrivacyの対象であり，RiskPrivacyに対応していないアプリケーションはE_RISK_APPLICATIONエラー，ユーザが経路の開示を承認していない場合はE_RISK_USERエラーとなる。



経路の座標列の絶対精度は地図データに依存するため，保証されない。

getAddress

現在位置の住所（行政区域）の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
address	NaviAddressType	現在位置の住所（行政区域）

【エラーコード】

E_NO_DATA	住所を判定できない（地図範囲外・マップマッチング不能等）
E_RISK_APPLICATION	アプリケーションが対応していないリスクがある
E_RISK_USER	ユーザがリスクを承認していない
E_OBJECT_FAULT	ナビゲーションが故障中

【機能】

現在位置（自車位置）の住所（行政区域）を，Naviが保持する地図データに基づいて判定して返す。

機能クラスNaviAddressを持たない車両では，本サービスコールはE_NOT_SUPPORTEDエラーとなる。

住所は個人の行動を推定できる高感度な情報であるため，本サービスコールはリスククラスRiskPrivacyの対象であり，RiskPrivacyに対応していないアプリケーションはE_RISK_APPLICATIONエラー，ユーザが住所の開示を承認していない場合はE_RISK_USERエラーとなる。



本版では自車位置の住所のみを判定できる。将来版では，オプショナルの位置パラメータの追加により，任意地点の住所判定へ後方互換に拡張しうる。

lock

ナビゲーションのロックの取得

【パラメータ】

priority	PriorityType?	操作優先度

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_PARAMETER	priorityが有効範囲外
E_RISK_APPLICATION	アプリケーションが対応していないリスクがある
E_RISK_USER	ユーザがリスクを承認していない
E_DENIED_PRIORITY	priorityが，アプリケーションが使用できない値
E_OBJECT_LOCKED	ナビゲーションが，priorityと同じかそれより高い優先度でロックされている
E_OBJECT_FAULT	ナビゲーションが故障中

【機能】

ナビゲーションを，指定した操作優先度でロックする。ロックすることにより，他のアプリケーションによる目的地・経由地・経路の変更（setDestination，cancelRouting）を抑止できる。

ロックの取得は経路変更の書き込み権を予約する操作であるため，リスククラスRiskRouteChangeの対象であり，対応できていない場合には，E_RISK_APPLICATIONエラーまたはE_RISK_USERエラーとなる。

unlock

ナビゲーションのロックの解除

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_OBJECT_STATUS	自アプリケーションが，ナビゲーションをロックしていない

【機能】

ナビゲーションをロック解除する。

notify

ナビゲーションイベントの通知要求

【パラメータ】

eventFilter	enumSet (NaviEventKindType)?	通知するイベントの種類の集合

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【エラーコード】

E_INVALID_PARAMETER	eventFilterが有効範囲外
E_RISK_APPLICATION	アプリケーションが対応していないリスクがある
E_RISK_USER	ユーザがリスクを承認していない

【機能】

ナビゲーションで発生したeventFilterで指定したイベントの通知を要求する。イベントを通知するための

イベントキューを生成し，その通知ハンドルをnotifyHandleに返す。

eventFilterを省略した場合，すべての種類のイベントの通知を要求する。

通知されるイベントには，報告時点の状態の全量を運ぶStatusReportedイベントが含まれることがあり，また，位置から派生するイベント（Arrived，WaypointPassed，AreaChanged等）の発生の観測からも位置情報を推定できる。このため，本サービスコールの呼び出しそのものが，eventFilterの内容によらず（eventFilterを省略した場合を含む），リスククラスRiskPrivacyの対象である。RiskPrivacyに対応していないアプリケーションはE_RISK_APPLICATIONエラー，ユーザが位置・状態の開示を承認していない場合はE_RISK_USERエラーとなる。

unnotify

ナビゲーションイベントの通知停止

【パラメータ】

notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正

【機能】

notifyHandleで指定した通知を停止する。指定した通知用のイベントキューは削除される。

getEvent

ナビゲーションイベントの取得

【パラメータ】

notifyHandle	NotifyHandleType	イベントキューを識別するための通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
event	NaviEventType	取得したイベント

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正
E_NO_DATA	イベントキューが空

【機能】

notifyHandleで指定した通知用のイベントキューから，先頭のイベントを取り出す。取り出したイベントは，イベントキューから削除される。



イベントがパラメータ等で渡されてくるプログラミング環境向けの物理APIでは，用意する必要がない。

5.23. Vehicle.SurroundModel（略称：SurroundModel）

5.23.1. 位置付けと機能

Vehicle.SurroundModel（略称：SurroundModel）は，自車周辺の環境モデルのうち，自車周辺で検出された可動物（車両・人・障害物など，動きうる物体）のリストを提供するシングルインスタンスオブジェクトである。各物体の位置・速度・加速度・回転・形状・分類・追跡状態・存在確率・品質・車線との紐付け，およびそれぞれの推定誤差を状態として提供する。状態には，物体リスト全体に共通する測定時刻・サンプル品質・有効外界センサ数・現在の認識性能とその変化理由・有効物体数が含まれる。

5.23.1.1. SurroundModelの位置付けと性質

SurroundModelが提供する各値は，自車の外界センサによる観測に基づく推定値であり，確定値（グラウンドトゥルース）ではない。観測手段（カメラ・ミリ波レーダ・LiDAR・それらの融合など）とその構成は規定しない。観測手段や走行環境（悪天候・逆光・遮蔽・混雑等）による精度の違いは，各推定誤差フィールドおよび品質フィールド（存在確率・追跡品質・サンプル品質）の実値として表現され，APIとしての精度の保証は行わない。取得した値の信頼度は，これらの品質フィールドと推定誤差によって判断する。

SurroundModelが提供するのは可動物のリストであり，道路（走行可能領域・車線・路面等），路側物（ランドマーク・交通標識・交通信号），自由空間の情報は持たない。道路の論理構造（レーン・エリアとそれらの接続関係，交通規制）は，Vehicle.SurroundModel.LogicalStructureが提供する。

5.23.1.2. 座標系と基準点

物体の位置・速度・加速度・回転・向きが表す座標系は，ISO 8855の車両座標系（原点：後軸中心の路面投影点，X軸：車両前方，Y軸：車両左方，Z軸：上方の右手系）である。すなわち各物体の位置は，自車を基準とする相対座標で表される。座標系と基準点は，構成情報coordinateSystemおよびreferencePointで宣言される。独自の基準点は用いない。

角度（ヨー・ピッチ・ロール・方位角・仰角）の単位はdegであり，回転の正方向は右手系に従う。すなわち，ヨーはZ軸まわりの回転角（車両前方を0とし，左向きが正），ピッチはY軸まわりの回転角（前傾（下向き）が正），ロールはX軸まわりの回転角（右傾が正）である。方位角は，向きをXY平面に投影した向きがX軸となす角（車両前方を0とし，左向きが正）である。仰角は，向きがXY平面となす角であり，XY平面から上向き（Z軸の正の向き）を正とする。仰角の符号の向きは，ピッチ（前傾（下向き）が正）とは逆である。

各物体の位置がその物体のどの点を表すかは，物体ごとの参照点（referencePoint）で示される。

5.23.1.3. 推定誤差と品質の表現

位置・速度・加速度・回転・向き・サイズには，それぞれの推定誤差が対になって提供される。位置の誤差は成分別誤差（x，y，z）または誤差楕円（半長軸・半短軸・長軸の角度）の形式で，速度の誤差はベクトル形式（x，y，z）または極形式（大きさ・方位角・仰角）の形式で表現される。提供側は，算出できる形式の誤差を返す（センサにより得意な表現が異なるため，両形式のいずれか一方または双方が値を

持ちうる）。

すべての推定誤差が表す信頼水準は，95%に固定する。誤差楕円以外の推定誤差については，真値が推定値±推定誤差の範囲に入る確率が95%（2σ相当）となる。誤差楕円については，真値が推定位置を中心とする誤差楕円の内側に入る確率が95%となる。1σ等の他の信頼水準で誤差を算出する実装では，車両側実装が95%相当へ換算して返す。これは精度の保証ではなく，誤差申告の意味の統一である。推定誤差を算出できない実装や瞬間には，当該の誤差フィールドは値なしとなる。

物体ごとの品質は，存在確率（existenceProbability，その物体が実在する確からしさ）と追跡品質（trackQuality，追跡の安定性の指標）で表す。いずれもFloat64の[0.0,1.0]で表し，値が大きいほど確からしい（安定している）ことを示す。

5.23.1.4. 測定時刻とサンプル品質

状態には，物体リスト全体の測定時刻timestampとサンプル品質dataQualifierが含まれる。timestampは，車内共通時刻基盤に基づく測定時刻（センサ観測のエポック）であり，送信時刻ではない。同じ時刻基盤に基づく他のビークルAPIのタイムスタンプと直接比較できる。1つのサンプル（物体リスト）を構成する物体は，同一の測定時刻に対応する観測から生成する（複数のセンサを併用する実装では，車両側実装が同一時刻へ整列させる）。trackingStatusがPredicted（Predicted〜）の物体の値は，状態のtimestampの時点に合わせて予測した値である。タイムスタンプ自体の誤差上限は，構成情報timestampAccuracyで宣言される。

dataQualifierは，取得できたサンプルの品質を表す。故障（外界センサ系の異常により周辺環境モデルを提供できない状態。E_OBJECT_FAULTエラーおよびFaultDetectedイベント）が「物体リストを出力できない状態」を表すのに対し，dataQualifierは「出力されたサンプルがどれだけ信用できるか」を表す。物体を1つも検出していない状態は故障ではなく通常の状態であり，物体の配列が空のサンプル（またはdataQualifierがNormalで有効物体数が0のサンプル）として表される。両者の使い分けの詳細は，状態の節に規定する。

5.23.1.5. 物体リストの構造と状態の取得

SurroundModelの状態は，物体リスト全体に共通する情報（測定時刻・サンプル品質・有効外界センサ数・現在の認識性能とその変化理由・有効物体数）と，可動物の配列（movingObjects）から構成される。物体は走行に伴い動的に増減するため，配列として提供される。個々の物体は物体ごとのID（objectId）で識別されるが，個別の物体をID指定で取得するサービスコールは設けず，物体リスト全体を単位として取得・報告する。

状態の更新周期は車両実装に依存する。状態は，要求応答（getStatus）により取得できるほか，状態報告イベント（StatusReported）によって周期的に得ることができる。物体リストは連続更新される信号系データであるため，物体の出現・消滅や物体リストの更新をきっかけに発生するイベントは定義しない。物体リストは，StatusReportedイベントの関連情報statusにより周期的に得られる。

5.23.1.6. 公称の認識性能と現在の認識性能

構成情報の視野（fieldOfViews）・空間分解能（recognitionResolution）・認識性能（recognitionPerformances）・処理能力（maxRecognizableObjectCount，maxProcessedObjectCount，processingTime）は，搭載時の公称値であり，稼働中の天候やセンサの汚れ等による変化を反映しない。getConfigは，これらの公称値を返す。

稼働中の実際の値は，状態の現在の認識性能（currentPerformance）として，公称値と同じ型で表される。公称値から変化している理由は，状態の変化理由（performanceChangeReasons）で表される。変化理由が変化したとき，または現在の認識性能が一定以上変化したときは，PerformanceChangedイベン

トが発生する（発生条件の詳細は，イベントの節に規定する）。PerformanceChangedイベントは変化の内容を運ばない。利用側は，PerformanceChangedイベントを受けたら，getStatusにより状態を取得し直す。現在の認識性能と変化理由は状態の一部であるため，StatusReportedイベントの関連情報statusにも含まれる。

5.23.1.7. 車線との紐付け

各物体は，その物体が位置する車線との紐付け（laneAssociation）を持ちうる。紐付けは，Vehicle.SurroundModel.LogicalStructureの車線の要素IDによる形式と，自車の車線を基準とする相対車線番号による形式の2つで表され，提供側は判定できる形式の値を返す（いずれか一方または双方が値を持ちうる）。いずれの形式を提供できるかは，機能クラスで宣言される。紐付けの判定は車両実装が行い，その結果と帰属の確からしさを返す。

車線の要素IDによる形式を用いる場合，利用側は次の点に注意する。可動物の検出範囲とLogicalStructureの提供範囲は一致するとは限らない。また，可動物の位置は車両座標系で表されるのに対し，LogicalStructureの幾何はWGS84で表されることがある。紐付けの判定はこれらの違いを車両実装が解消したうえで行うものであり，利用側が可動物の位置とLogicalStructureの幾何を直接比較して紐付けを求め直すことは想定しない。さらに，要素IDが指す車線は，可動物の状態と同じ時点のLogicalStructureの状態であっても，それに含まれているとは限らない（LogicalStructureの提供範囲の外にある車線や，提供範囲の移動により状態から削除された車線を指すことがある）。利用側が保持するLogicalStructureの状態に当該の要素IDの車線が含まれない場合，その紐付けは車線の要素IDによっては解決できないものとして扱う。

5.23.1.8. アクセス制御

周辺環境モデルは，自車周辺の歩行者・車両等の位置・挙動を含み，これらの人・車を推定できる情報である。そのため，状態を参照するサービスコール（getStatus）はアクセス制御の対象であり，アクセスが認められていないアプリケーションに対してはE_DENIED_ACCESSエラーとなる。イベントの通知要求（notify）により通知されるイベントには，報告時点の状態の全量を運ぶStatusReportedイベントが含まれることがある。このため，notifyの呼び出しそのものを，eventFilterで指定するイベントの種類によらず，アクセス制御の対象とする。

SurroundModelは，車両の状態を変化させるサービスコールを持たない参照専用のオブジェクトであるため，リスククラスを定義しない。

5.23.2. 機能クラス

SurroundModelは，以下の機能クラスを持つ。

DetectionStatisticsProvidable	検出統計（detectionCount，missDetectionCount，freezeCount，currentUsedSensor）を提供することができる
TrackQualityProvidable	追跡品質（trackQuality）および有効観測数（numberOfValidObservations）を提供することができる
PositionErrorEllipseProvidable	位置誤差を誤差楕円の形式（positionErrorEllipse）で提供することができる
VelocityPolarFormProvidable	速度およびその誤差を極形式（velocityPolar，velocityPolarError）で提供することができる
AccelerationProvidable	物体の加速度（acceleration）とその誤差を提供することができる
RotationProvidable	物体の回転中心・回転速度（rotationCentre，rotationRateYaw）とその誤差を提供することができる

VehicleAttributeProvidable	車両である物体の属性（灯火状態・V2X情報）を提供することができる
PersonAttributeProvidable	人である物体の属性（姿勢・付帯物・人数）を提供することができる
LaneIdAssociationProvidable	物体と車線との紐付けを，Vehicle.SurroundModel.LogicalStructureの車線の要素ID（laneAssociationのlaneId）で提供することができる
RelativeLaneAssociationProvidable	物体と車線との紐付けを，自車の車線を基準とする相対車線番号（laneAssociationのrelativeLaneNumber）で提供することができる

5.23.3. リスククラス

SurroundModelは，車両の状態を変化させるサービスコールを持たない参照専用のオブジェクトであるため，リスククラスを定義しない。

周辺の人・車の情報の参照に伴うプライバシのリスクに対しては，状態の参照（getStatus）およびイベントの通知要求（notify）へのアクセス制御（E_DENIED_ACCESSエラー）により対応する。

5.23.4. 構成情報

SurroundModelは，以下の構成情報を持つ。

capabilities	enumSet (SurroundModelCapabilityType)	このSurroundModelが持つ機能
coordinateSystem	String	物体の位置・速度・向きの座標系の宣言（ISO 8855の車両座標系）
referencePoint	String	車両座標系の基準点の宣言（ISO 8855の車両座標系原点：後軸中心の路面投影点）
timestampAccuracy	UInt32	この車両のタイムスタンプ誤差上限の宣言。単位はμs
statusReportInterval	UInt32 ([1,4294967295])?	状態報告イベント（StatusReported）が発生する周期。単位はms。値を持つ場合は1以上。値なしの場合，この車両ではStatusReportedイベントは発生しない
fieldOfViews	SurroundModelFieldOfViewType[]	認識可能な視野（FOV）の公称値の一覧。配列の要素数がFOVの数を表す。各FOVは車両座標系で定義される
recognitionResolution	SurroundModelResolutionType	観測領域における認識の空間分解能の公称値
recognitionPerformances	SurroundModelRecognitionPerformanceType[]	物体種別ごと・距離ごとの認識性能（正認識率・誤認識率・識別できる分類数）の公称値の一覧
maxRecognizableObjectCount	UInt32	同時に認識できる物体の最大数の公称値
maxProcessedObjectCount	UInt32	1周期に処理できる物標の最大数の公称値

processingTime	UInt32	物体リストの認識処理に要する標準的な処理時間の公称値。単位はms

fieldOfViews，recognitionResolution，recognitionPerformances，maxRecognizableObjectCount，maxProcessedObjectCount，processingTimeは，搭載時の公称値であり，稼働中の天候やセンサの汚れ等による変化を反映しない。稼働中の実際の値は，状態currentPerformanceで表される。

5.23.5. 状態

SurroundModelは，以下の状態を持つ。

timestamp	TimestampUsType	物体リストの測定時刻。単位はμs。車内共通時刻基盤に基づく。dataQualifierがUnavailableのサンプルのtimestampは，サンプルの生成時刻を表す（測定時刻ではない）
dataQualifier	DataQualifierType	このサンプル（物体リスト）の品質。InvalidまたはUnavailableの場合，物体の配列（movingObjects）およびその要素数（numberOfValidMovingObjects）を読んではならない。Unavailableの場合は，有効外界センサ数（numberOfValidSensors），現在の認識性能（currentPerformance）および変化理由（performanceChangeReasons）も読んではならない
numberOfValidSensors	UInt32	この物体リストの生成に寄与した有効な外界センサの数。dataQualifierがUnavailableの場合，読んではならない
currentPerformance	SurroundModelPerformanceType	現在の認識性能（視野・空間分解能・認識性能・処理能力）。dataQualifierがUnavailableの場合，読んではならない
performanceChangeReasons	enumSet (SurroundModelPerformanceChangeReasonType)	現在の認識性能が公称値から変化している理由の集合。いずれかの対象項目の基準の値が公称値から判定の幅以上離れている場合に限り空でない集合となり，それ以外の場合は空の集合となる（対象項目・基準の値・判定の幅はイベントの節に規定する）。dataQualifierがUnavailableの場合，読んではならない
numberOfValidMovingObjects	UInt32	有効な可動物の数。movingObjectsの配列の要素数と一致する
movingObjects	SurroundModelMovingObjectType[]	検出された可動物の配列。物体を1つも検出していない場合は空の配列となる

timestampは，この物体リストの測定時刻（センサ観測のエポック）であり，車内共通時刻基盤に基づく。タイムスタンプの誤差上限は，構成情報timestampAccuracyで宣言される。

dataQualifierは，このサンプルの品質を表す。Invalidは，認識器自身が自分の出力を信用できない状態（起動直後の初期化中・フィルタ発散中など）であり，物体の配列（movingObjects）およびその要素数（numberOfValidMovingObjects）を読んではならない。Degradedは，物体リストは使えるが通常品質でない状態（例：悪天候・逆光により認識品質が低下している場合）であり，個々の物体の確からしさは

存在確率・追跡品質・各推定誤差の実値で判断する。

物体を1つも検出していない状態は故障ではなく通常の状態であり，movingObjectsが空の配列（numberOfValidMovingObjectsが0）となる。外界センサ系の異常により周辺環境モデルそのものを提供できない状態は故障と扱い，故障中の状態の参照（getStatus）はE_OBJECT_FAULTエラーとなる。故障中にStatusReportedイベントで報告される状態は，dataQualifierがUnavailableである。Unavailableのサンプルも Invalidと同様に，物体の配列（movingObjects）およびその要素数（numberOfValidMovingObjects）を読んではならない（最終既知の物体リストとして読むことも許されない）。Unavailableのサンプルでは，有効外界センサ数（numberOfValidSensors），現在の認識性能（currentPerformance）および変化理由（performanceChangeReasons）も読んではならない。Unavailableのサンプルのtimestampは，サンプルの生成時刻を表す（測定時刻ではない）。一部の外界センサが利用できなくても，残りの外界センサにより物体リストを生成できている間は故障とせず，品質の低下はdataQualifier（Degraded）および各物体の存在確率・追跡品質・推定誤差の実値で表現する。

currentPerformanceは，構成情報の公称値（fieldOfViews，recognitionResolution，recognitionPerformances，maxRecognizableObjectCount，maxProcessedObjectCount，processingTime）に対する，稼働中の実際の値である。currentPerformanceの視野の配列（fieldOfViews）と認識性能の配列（recognitionPerformances）は，構成情報の対応する配列と同じ要素数・同じ順序とし，各要素は構成情報の同じ位置の要素に対応する実際の値を表す。要素を識別する項目，すなわち視野の形状（shape），および認識性能の対象となる物体の大分類（objectMacroClassification）と距離範囲（distanceRangeMin，distanceRangeMax）は，構成情報の同じ位置の要素と等しい値とする。視野がまったく使えない状態は，その視野の到達距離（range）を0とすることで表す。構成情報が変更された場合（ConfigChangedイベント），currentPerformanceは変更後の構成情報に追従する（配列の要素数・順序・要素を識別する項目を含む）。

performanceChangeReasonsは，currentPerformanceが公称値から変化している理由の集合である。performanceChangeReasonsは，いずれかの対象項目の基準の値が，公称値から判定の幅以上離れている場合に限り，空でない集合となる（理由を特定できない場合はUnknownを含める）。すべての対象項目について基準の値と公称値との差が判定の幅未満である場合，performanceChangeReasonsは空の集合となる。対象項目，基準の値および判定の幅は，イベントの節に規定する。基準の値が変わるのは，PerformanceChangedイベントを発生させたとき，稼働を開始したとき，故障から回復したとき，および構成情報が変更されたときだけである。したがって，performanceChangeReasonsが空の集合であるか否かが変わるのも，これらの時点だけである。車両実装は，同じ理由の追加・削除を短時間に繰り返さないよう，理由の判定を安定させる。

currentPerformanceとdataQualifierとは，それぞれ独立に定まる。currentPerformanceは観測できる範囲・能力の変化を表し，dataQualifierはサンプルの品質を表す。認識性能が公称値から低下していても，低下後の視野の中で生成した物体リストが通常の品質であれば，dataQualifierはNormalでありうる。dataQualifierがInvalidのサンプルでも，有効外界センサ数（numberOfValidSensors），現在の認識性能（currentPerformance）および変化理由（performanceChangeReasons）は読んでよい。これらを読んではならないのは，dataQualifierがUnavailableの場合だけである。

各物体の任意項目（検出統計・追跡品質・誤差楕円・速度の極形式・加速度・回転・車両属性・人属性・車線との紐付け）は，対応する機能クラスを備え，かつ当該物体について当該値を算出できている場合にのみ値を持つ。

5.23.6. サービスコール一覧

SurroundModelに対するサービスコールは以下の通り。

getConfig	SurroundModelの構成情報の参照

getStatus	周辺物体リストの参照
notify	SurroundModelイベントの通知要求
unnotify	SurroundModelイベントの通知停止
getEvent	SurroundModelイベントの取得

5.23.7. イベント

SurroundModelのイベント種別には，故障の検知・回復，認識性能の変化（PerformanceChanged），および状態の報告（StatusReported）がある。物体リストは連続更新される信号系データであるため，物体の出現・消滅や物体リストの更新をきっかけに発生するイベントは定義しない。物体リストは，StatusReportedイベントの関連情報statusにより周期的に得られる。

イベントの関連情報には，診断的な最小限の情報のみを含める。PerformanceChangedイベントは変化の内容を運ばず，変化後の現在の認識性能と変化理由は，getStatusにより取得する。ただし，StatusReportedイベントは，報告時点の状態の全量（SurroundModelStatusType）を関連情報statusとして含む。

SurroundModelに対するイベント種別は以下の通り。

FaultDetected	故障検知（外界センサ系の異常により周辺環境モデルを提供できない状態）
FaultRecovered	故障から回復
StatusReported	状態の報告（構成情報statusReportIntervalの周期ごとに発生する）
PerformanceChanged	現在の認識性能または変化理由の変化（発生条件は下記）
EventOverflow	イベントキューのオーバフロー
ConfigChanged	構成情報の変更

PerformanceChangedイベントは，getStatusが返す状態について，次のいずれかが成り立ったときに発生する。

- 変化理由（performanceChangeReasons）の集合が変化した（理由が追加された，または削除された）。
- 現在の認識性能（currentPerformance）の対象項目のいずれかについて，現在の値と基準の値との差の絶対値が，その項目の判定の幅以上となった。

対象項目は，視野の各要素のhorizontalFieldOfView，verticalFieldOfView，range，空間分解能のlateralResolution，longitudinalResolution，verticalResolution，認識性能の各要素のtruePositiveRate，falsePositiveRate，distinguishableClassCount，およびmaxRecognizableObjectCount，maxProcessedObjectCount，processingTimeである。要素を識別する項目（shape，objectMacroClassification，distanceRangeMin，distanceRangeMax）は対象項目ではない。各対象項目の公称値は，構成情報の同じ項目（配列の場合は同じ位置の要素の同じ項目）の値である。

各対象項目の判定の幅は，truePositiveRateとfalsePositiveRateでは，公称値によらず0.1（値域[0.0,1.0]の10%）とする。それ以外の対象項目では，その項目の公称値の絶対値の10%とする。ただし，truePositiveRateとfalsePositiveRate以外の対象項目のうち公称値が0のものは，PerformanceChangedイベントの発生条件および変化理由の判定の対象としない。判定の幅は，変化理由（

performanceChangeReasons）が空でない集合となる条件の判定にも同じく用いる。

各対象項目の基準の値は，直近にPerformanceChangedイベントを発生させた時点のその項目の値である。PerformanceChangedイベントを発生させたときは，いずれの条件によって発生させた場合も，その時点のすべての対象項目の値を新しい基準の値とする。稼働を開始したとき，故障から回復したとき，および構成情報が変更されたときは，その時点の値を基準の値とする。基準の値を新しくしたときは，performanceChangeReasonsも新しい基準の値に基づいて定める。したがって，基準の値からの差が判定の幅に満たない値の揺れによっては，PerformanceChangedイベントは発生しない。
PerformanceChangedイベントは，変化理由の集合が変化しない場合でも，対象項目の値の変化によって発生しうる。その間の最新の値は，getStatusまたはStatusReportedイベントの関連情報statusにより取得する。同時に複数の条件が成り立った場合は，1つのイベントとして発生させてよい。
PerformanceChangedイベントは，変化を反映した状態がgetStatusにより取得できるようになった後に発生する。故障中（dataQualifierがUnavailable）には発生しない。故障から回復したときの現在の認識性能と変化理由は，FaultRecoveredイベントを受けてgetStatusにより取得する。

構成情報の視野・認識性能・処理能力は搭載時の公称値であり，稼働中の認識性能の変化によっては変化しない。したがって，稼働中の認識性能の変化によってConfigChangedイベントは発生しない。構成情報が変更され，それに追従してcurrentPerformanceが変わった場合は，ConfigChangedイベントが発生し，PerformanceChangedイベントは発生しない。構成情報の変更に伴う変化理由の変化によっても，PerformanceChangedイベントは発生しない。利用側は，ConfigChangedイベントを受けたら，getConfigとgetStatusにより構成情報と状態を取得し直す。

StatusReportedイベントの関連情報statusは，報告時点の周辺物体リストの状態の全量であり，getStatusが返す状態と同じ型（SurroundModelStatusType）である。関連情報timestamp（TimestampMsType）と，状態が持つtimestamp（TimestampUsType）とは単位が異なり，起点が同一であることも保証されないため，互いに比較することはできない。状態の更新周期と構成情報statusReportIntervalとは一致するとは限らない。dataQualifierがUnavailableでないサンプルどうしでは，連続する報告の関連情報statusが同じ観測に基づく物体リストであるか否かを，statusのtimestampにより判断できる。dataQualifierがUnavailableのサンプルのtimestampはサンプルの生成時刻を表すため，これにより判断することはできない。

この内，イベント種別がStatusReportedの場合には，以下の関連情報を持つ。

timestamp	TimestampMsType	報告が発生した時刻
status	SurroundModelStatusType	報告時点の周辺物体リストの状態（全量）

また，イベント種別がEventOverflowの場合には，以下の関連情報を持つ。

noLostEvents	UInt32	失われたイベントの数

5.23.8. データ型の定義

5.23.8.1. SurroundModelの機能クラス

```
type SurroundModelCapabilityType = enum {
    DetectionStatisticsProvidable,
    TrackQualityProvidable,
    PositionErrorEllipseProvidable,
    VelocityPolarFormProvidable,
```

```
    AccelerationProvidable,
    RotationProvidable,
    VehicleAttributeProvidable,
    PersonAttributeProvidable,
    LaneIdAssociationProvidable,
    RelativeLaneAssociationProvidable
};
```

5.23.8.2. 視野（FOV）

```
type SurroundModelFovShapeType = enum {
    Unknown,    // 不明
    Conical,    // 円錐状
    Rectangular,    // 矩形状
    Sector, // 扇形状
    Other   // その他
};
```

```
type SurroundModelFieldOfViewType = struct {
    shape : SurroundModelFovShapeType,  // 視野の形状
    horizontalFieldOfView : Float64,    // 水平方向の視野角。単位はdeg
    verticalFieldOfView : Float64,  // 垂直方向の視野角。単位はdeg
    range : Float64 // 視野の到達距離。単位はm
};
```

視野（FOV）は車両座標系で定義される。認識可能な観測領域を表す。

5.23.8.3. 認識の空間分解能

```
type SurroundModelResolutionType = struct {
    lateralResolution : Float64,    // 横方向（Y軸方向）の分解能。単位はm
    longitudinalResolution : Float64,   // 奥行き方向（X軸方向）の分解能。単位はm
    verticalResolution : Float64    // 縦方向（Z軸方向）の分解能。単位はm
};
```

5.23.8.4. 認識性能

```
type SurroundModelRecognitionPerformanceType = struct {
    objectMacroClassification : SurroundModelObjectMacroClassificationType, // 認識性
能の対象となる物体の大分類
    distanceRangeMin : Float64, // 認識性能が適用される距離範囲の下限。単位はm
    distanceRangeMax : Float64, // 認識性能が適用される距離範囲の上限。単位はm
    truePositiveRate : Float64 ([0.0,1.0]), // 正認識率（TruePositive）。1.0が最良
    falsePositiveRate : Float64 ([0.0,1.0]),    // 誤認識率（FalsePositive）。0.0が最
良
    distinguishableClassCount : UInt32  // この距離範囲で識別できる分類の数
```

```
};
```

5.23.8.5. 現在の認識性能

```
type SurroundModelPerformanceType = struct {
    fieldOfViews : SurroundModelFieldOfViewType[],  // 現在の視野（FOV）の一覧。構成情報fieldOfViewsと同じ要素数・同じ順序
    recognitionResolution : SurroundModelResolutionType,    // 現在の認識の空間分解能
    recognitionPerformances : SurroundModelRecognitionPerformanceType[],    // 現在の物体種別ごと・距離ごとの認識性能の一覧。構成情報recognitionPerformancesと同じ要素数・同じ順序
    maxRecognizableObjectCount : UInt32,    // 現在，同時に認識できる物体の最大数
    maxProcessedObjectCount : UInt32,   // 現在，1周期に処理できる物標の最大数
    processingTime : UInt32 // 現在の物体リストの認識処理に要する標準的な処理時間。単位はms
};
```

稼働中の実際の視野・空間分解能・認識性能・処理能力を表す。各フィールドは，構成情報の同名の項目（公称値）と同じ型・同じ単位である。

```
type SurroundModelPerformanceChangeReasonType = enum {
    Unknown,    // 不明（認識性能が変化しているが，理由を特定できない）
    Rain,   // 降雨
    Snow,   // 降雪
    Fog,    // 霧
    Glare,  // 逆光・強い光源（太陽光・対向車の前照灯等）
    Clutter,    // クラッタ（路面・構造物等からの不要な反射）
    FlyingDebris,   // 飛散物（舞い上がる落ち葉・砂塵・水しぶき等）
    SensorContamination,    // センサの汚れ（泥・雪・氷・虫等のセンサ表面への付着）
    SensorBlockage, // センサの遮蔽（センサの直前にある物・積載物・取付物による視野の遮り）
    SensorFault,    // 一部の外界センサの故障
    Other   // その他の理由
};
```

認識性能が公称値から変化している理由を表す。変化理由は複数が同時に成り立ちうるため，状態では集合（enumSet）で表す。SensorBlockageは，周辺の交通参加者や構造物による一時的な隠蔽を含まない（これらによって見えない物体は，物体ごとの追跡状態で表される）。SensorFaultは，残りの外界センサにより物体リストを生成できている間の一部の外界センサの故障を表し，周辺環境モデルそのものを提供できない故障（FaultDetectedイベント）とは区別される。

5.23.8.6. SurroundModelの構成情報（getConfigが返すデータ型）

```
type SurroundModelConfigType = struct {
    capabilities : enumSet(SurroundModelCapabilityType),
    coordinateSystem : String,
    referencePoint : String,
    timestampAccuracy : UInt32,
```

```
    statusReportInterval : UInt32 ([1,4294967295])?,
    fieldOfViews : SurroundModelFieldOfViewType[],
    recognitionResolution : SurroundModelResolutionType,
    recognitionPerformances : SurroundModelRecognitionPerformanceType[],
    maxRecognizableObjectCount : UInt32,
    maxProcessedObjectCount : UInt32,
    processingTime : UInt32
};
```

5.23.8.7. 3次元の点

```
type SurroundModelPoint3DType = struct {
    x : Float64,    // X座標（車両前方が正）。単位はm
    y : Float64,    // Y座標（車両左方が正）。単位はm
    z : Float64 // Z座標（上方が正）。単位はm
};
```

物体の相対位置を表す3次元の点である。座標系はISO　　8855の車両座標系（原点：後軸中心の路面投影点）とする。

```
type SurroundModelPoint3DErrorType = struct {
    xError : Float64,   // X成分の推定誤差。単位はm。信頼水準は95%（2σ相当）
    yError : Float64,   // Y成分の推定誤差。単位はm。信頼水準は95%（2σ相当）
    zError : Float64    // Z成分の推定誤差。単位はm。信頼水準は95%（2σ相当）
};
```

```
type SurroundModelEllipseAngleType = Float64 ([0.0,180.0]); // 誤差楕円の長軸がX軸となす角。単位はdeg。X軸を0とし，左向き（X軸からY軸へ向かう向き）を正とする。長軸の向きは180degを周期として同一であるため，値がとりうる範囲は[0,180)とする。
```

```
type SurroundModelErrorEllipseType = struct {
    majorAxisError : Float64,   // 誤差楕円の半長軸。単位はm。信頼水準は95%
    minorAxisError : Float64,   // 誤差楕円の半短軸。単位はm。信頼水準は95%
    ellipseAngle : SurroundModelEllipseAngleType    // 誤差楕円の長軸の角度
};
```

誤差楕円は，XY平面内における位置の推定誤差の分布を，半長軸majorAxisError・半短軸minorAxisErrorと長軸の角度ellipseAngleで表す。majorAxisErrorおよびminorAxisErrorは，楕円の中心から楕円の周までの長さであり，軸の全長（長径・短径）の2分の1である。majorAxisErrorはminorAxisError以上でなければならない。

```
type SurroundModelPoint2DType = struct {
    x : Float64,    // X座標（車両前方が正）。単位はm
    y : Float64 // Y座標（車両左方が正）。単位はm
```

```
};
```

```
type SurroundModelPoint2DErrorType = struct {
    xError : Float64,   // X成分の推定誤差。単位はm。信頼水準は95%（2σ相当）
    yError : Float64    // Y成分の推定誤差。単位はm。信頼水準は95%（2σ相当）
};
```

5.23.8.8. 物体の位置

```
type SurroundModelObjectPositionType = struct {
    referencePoint : SurroundModelReferencePointType,   // 位置が表す物体上の参照点
    position : SurroundModelPoint3DType,    // 参照点の相対位置。単位はm
    positionError : SurroundModelPoint3DErrorType?, // 位置の成分別の推定誤差。単位はm
    positionErrorEllipse : SurroundModelErrorEllipseType?,  // 位置の誤差を誤差楕円で
表したもの。機能クラスPositionErrorEllipseProvidableを備える場合にのみ値を持つ
    sameRoadLevel : Bool?   // 自車と同じ路面上にある物体か。判定できている場合にのみ
値を持つ
};
```

5.23.8.9. 速度・加速度・回転

```
type SurroundModelVelocityVectorType = struct {
    vx : Float64,   // X方向の速度（車両前方が正）。単位はm/s
    vy : Float64,   // Y方向の速度（車両左方が正）。単位はm/s
    vz : Float64    // Z方向の速度（上方が正）。単位はm/s
};
```

```
type SurroundModelVelocityVectorErrorType = struct {
    vxError : Float64,  // X方向速度の推定誤差。単位はm/s。信頼水準は95%（2σ相当）
    vyError : Float64,  // Y方向速度の推定誤差。単位はm/s。信頼水準は95%（2σ相当）
    vzError : Float64   // Z方向速度の推定誤差。単位はm/s。信頼水準は95%（2σ相当）
};
```

```
type SurroundModelElevationType = Float64 ([-90.0,90.0]);   // 向きがXY平面となす角
（仰角）。単位はdeg。XY平面を0とし，上向き（Z軸の正の向き）を正とする。
```

```
type SurroundModelVelocityPolarType = struct {
    speed : Float64,    // 速度の大きさ。単位はm/s
    azimuth : Float64,  // 速度の向きの方位角。単位はdeg。車両前方を0とし，左向きが正
    elevation : SurroundModelElevationType? // 速度の向きの仰角。推定できている場合に
のみ値を持つ
};
```

方位角azimuthは，速度ベクトルをXY平面に投影した向きがX軸となす角である。仰角elevationは，速度ベクトルがXY平面となす角であり，XY平面から上向き（Z軸の正の向き）を正とする（ピッチ角とは符号の向きが逆である）。elevationが値を持つ場合，speedは3次元の速度ベクトルの大きさである。elevationが値を持たない場合，speedはXY平面内の速度（速度ベクトルのXY平面への投影）の大きさである。

```
type SurroundModelVelocityPolarErrorType = struct {
    speedError : Float64,   // 速度の大きさの推定誤差。単位はm/s。信頼水準は95%（2σ相当）
    azimuthError : Float64, // 速度の向きの方位角の推定誤差。単位はdeg。信頼水準は95%（2σ相当）
    elevationError : Float64?   // 速度の向きの仰角の推定誤差。単位はdeg。信頼水準は95%（2σ相当）。velocityPolarのelevationが値を持つ場合にのみ値を持ちうる
};
```

```
type SurroundModelAccelerationType = struct {
    ax : Float64,   // X方向の加速度（車両前方が正）。単位はm/s^2
    ay : Float64,   // Y方向の加速度（車両左方が正）。単位はm/s^2
    az : Float64    // Z方向の加速度（上方が正）。単位はm/s^2
};
```

```
type SurroundModelAccelerationErrorType = struct {
    axError : Float64,  // X方向加速度の推定誤差。単位はm/s^2。信頼水準は95%（2σ相当）
    ayError : Float64,  // Y方向加速度の推定誤差。単位はm/s^2。信頼水準は95%（2σ相当）
    azError : Float64   // Z方向加速度の推定誤差。単位はm/s^2。信頼水準は95%（2σ相当）
};
```

```
type SurroundModelObjectDynamicsType = struct {
    velocityVector : SurroundModelVelocityVectorType?,  // 速度のベクトル形式
    velocityVectorError : SurroundModelVelocityVectorErrorType?,    // 速度のベクトル形式の推定誤差
    velocityPolar : SurroundModelVelocityPolarType?,    // 速度の極形式。機能クラスVelocityPolarFormProvidableを備える場合にのみ値を持つ
    velocityPolarError : SurroundModelVelocityPolarErrorType?,  // 速度の極形式の推定誤差
    acceleration : SurroundModelAccelerationType?,  // 加速度。機能クラスAccelerationProvidableを備える場合にのみ値を持つ
    accelerationError : SurroundModelAccelerationErrorType?,    // 加速度の推定誤差
    rotationCentre : SurroundModelPoint2DType?, // 瞬間回転中心。機能クラスRotationProvidableを備える場合にのみ値を持つ
    rotationCentreError : SurroundModelPoint2DErrorType?,   // 瞬間回転中心の推定誤差
    rotationRateYaw : Float64?, // 回転速度（ヨー方向）。単位はdeg/s
    rotationRateYawError : Float64? // 回転速度（ヨー方向）の推定誤差。単位はdeg/s。信頼水準は95%（2σ相当）
};
```

速度はベクトル形式・極形式のいずれか一方または双方が値を持ちうる。提供側が算出できる形式が値を持つ。

5.23.8.10. 物体の向きとサイズ

```
type SurroundModelOrientation3DType = struct {
    yaw : Float64,  // ヨー角。単位はdeg。車両前方を0とし，左向きが正
    pitch : Float64,    // ピッチ角。単位はdeg。前傾（下向き）が正
    roll : Float64  // ロール角。単位はdeg。右傾が正
};
```

```
type SurroundModelOrientation3DErrorType = struct {
    yawError : Float64, // ヨー角の推定誤差。単位はdeg。信頼水準は95%（2σ相当）
    pitchError : Float64,   // ピッチ角の推定誤差。単位はdeg。信頼水準は95%（2σ相当）
    rollError : Float64 // ロール角の推定誤差。単位はdeg。信頼水準は95%（2σ相当）
};
```

```
type SurroundModelSizeType = struct {
    length : Float64,   // 物体の長さ（前後方向）。単位はm
    width : Float64,    // 物体の幅（左右方向）。単位はm
    height : Float64    // 物体の高さ（上下方向）。単位はm
};
```

```
type SurroundModelSizeErrorType = struct {
    lengthError : Float64,  // 長さの推定誤差。単位はm。信頼水準は95%（2σ相当）
    widthError : Float64,   // 幅の推定誤差。単位はm。信頼水準は95%（2σ相当）
    heightError : Float64   // 高さの推定誤差。単位はm。信頼水準は95%（2σ相当）
};
```

5.23.8.11. 物体の形状把握状態

```
type SurroundModelShapeAmbiguityType = enum {
    ObjectOrientationUnknown,   // 物体の向きが不明
    ObjectFrontBackRightLeftUnknown,    // 前後・左右の向きが不明
    ObjectFrontBackUnknown, // 前後の向きが不明
    ObjectOrientationKnown  // 物体の向きが既知
};
```

5.23.8.12. 物体の形状

```
type SurroundModelObjectShapeType = struct {
    shapeAmbiguity : SurroundModelShapeAmbiguityType,   // 物体の向きの把握状態
    orientation : SurroundModelOrientation3DType?,  // 物体の向き（方位角）。推定できている場合にのみ値を持つ
```

```
    orientationError : SurroundModelOrientation3DErrorType?,    // 物体の向きの推定誤
差
    size : SurroundModelSizeType?,  // 物体のサイズ（長さ・幅・高さ）。推定できている
場合にのみ値を持つ
    sizeError : SurroundModelSizeErrorType?,    // 物体のサイズの推定誤差
    objectClearance : Float64?, // 物体の路面からのクリアランス。単位はm。推定できてい
る場合にのみ値を持つ
    sizeIncludesMirror : Bool?  // サイズにサイドミラー等の突起を含むか。判定できてい
る場合にのみ値を持つ
};
```

5.23.8.13. 物体の大分類

```
type SurroundModelObjectMacroClassificationType = enum {
    Unknown,    // 不明
    PotentiallyMovingObject,    // 可動物
    RoadObject, // 道路
    StaticObject,   // 路側物
    FreeSpaceObject // 自由空間
};
```

可動物の配列（movingObjects）の要素の大分類（classificationのobjectMacroClassification）は，PotentiallyMovingObjectである。可動物の配列の要素が，Unknown・RoadObject・StaticObject・FreeSpaceObjectの値をとることはない。RoadObject・StaticObject・FreeSpaceObjectの値は，構成情報recognitionPerformancesおよび状態currentPerformanceのrecognitionPerformancesで認識性能の対象となる物体の大分類を示すために用いうる。

5.23.8.14. 可動物の分類（階層分類）

```
type SurroundModelPotentiallyMovingObjectClassType = enum {
    Unknown,    // 不明
    Car,    // 車
    NonCar  // 車以外
};
```

```
type SurroundModelCarClassType = enum {
    Unknown,    // 不明
    FourWheelCar,   // 四輪車
    OtherCar    // その他の車
};
```

```
type SurroundModelFourWheelCarType = enum {
    Unknown,    // 不明
    BigSizeCar, // 大型車
    MiddleSizeCar,  // 中型車
    SmallSizeCar    // 小型車
```

```
};
```

```
type SurroundModelBigSizeCarType = enum {
    Unknown,    // 不明
    Truck,  // トラック
    Trailer,    // トレーラ
    Train,  // 鉄道車両
    Bus,    // バス
    Other   // その他の大型四輪車
};
```

```
type SurroundModelMiddleSizeCarType = enum {
    Unknown,    // 不明
    Truck,  // トラック
    Trailer,    // トレーラ
    Bus,    // バス
    Van,    // バン
    PassengerCar,   // 乗用車
    Other   // その他の中型四輪車
};
```

```
type SurroundModelSmallSizeCarType = enum {
    Unknown,    // 不明
    Truck,  // トラック
    Van,    // バン
    PassengerCar,   // 乗用車
    Other   // その他の小型四輪車
};
```

```
type SurroundModelTruckType = enum {
    Unknown,    // 不明
    HeavyDuty,  // 大型トラック
    MediumDuty, // 中型トラック
    LightDuty   // 小型トラック
};
```

```
type SurroundModelTrailerType = enum {
    Unknown,    // 不明
    Trailer,    // トレーラ
    SemiTrailer // セミトレーラ
};
```

```
type SurroundModelTrainType = enum {
```

```
    Unknown,    // 不明
    Train,  // 鉄道車両
    Tram    // 路面電車
};
```

```
type SurroundModelBusType = enum {
    Unknown,    // 不明
    Large,  // 大型バス
    Midsize,    // 中型バス
    Small,  // 小型バス
    DoubleDeckerBus,    // 二階建てバス
    ArticulatedBus, // 連節バス
    SchoolBus,  // スクールバス
    OpenTopBus  // オープントップバス
};
```

```
type SurroundModelPassengerCarType = enum {
    Unknown,    // 不明
    Large,  // 大型乗用車
    Midsize,    // 中型乗用車
    Compact,    // 小型乗用車
    SubCompact  // 軽・準小型乗用車
};
```

```
type SurroundModelEmergencyVehicleType = enum {
    Unknown,    // 不明
    PoliceCar,  // パトカー
    FireApparatus,  // 消防車
    Ambulance,  // 救急車
    DoctorCar,  // ドクターカー
    HighwayPatrolCar    // 高速道路パトロールカー
};
```

```
type SurroundModelAgriculturalVehicleType = enum {
    Unknown,    // 不明
    Tractor,    // トラクタ
    SemiTractor,    // セミトラクタ
    CombineHarvester,   // コンバイン
    RiceTransplanter    // 田植機
};
```

```
type SurroundModelHeavyEquipmentType = enum {
    Unknown,    // 不明
    WheelLoader,    // ホイールローダ
```

```
    Bulldozer,  // ブルドーザ
    MobileCrane,    // 移動式クレーン
    Excavator,  // 油圧ショベル
    RoadRoller, // ロードローラ
    ConcreteMixerTruck, // コンクリートミキサー車
    CrawlerVehicle  // クローラ車
};
```

```
type SurroundModelOtherFourWheelCarType = enum {
    Unknown,    // 不明
    CarCarrier, // 車両運搬車
    ForkLift,   // フォークリフト
    Carriage,   // 荷車
    AgriculturalVehicle,    // 農業機械
    EmergencyVehicle,   // 緊急車両
    HeavyEquipment  // 重機
};
```

```
type SurroundModelOtherCarType = enum {
    Unknown,    // 不明
    Bike,   // 自動二輪車
    Bicycle,    // 自転車
    Tricycle,   // 三輪車
    OtherTwoWheels  // その他の二輪車
};
```

```
type SurroundModelBikeType = enum {
    Unknown,    // 不明
    BigSizeBike,    // 大型自動二輪車
    SmallSizeBike,  // 小型自動二輪車
    SideCar,    // サイドカー付き
    Group   // 集団
};
```

```
type SurroundModelBicycleType = enum {
    Unknown,    // 不明
    Bicycle,    // 自転車
    Tandem, // タンデム自転車
    Trailer,    // トレーラ付き自転車
    Group   // 集団
};
```

```
type SurroundModelTricycleType = enum {
    Unknown,    // 不明
```

```
    TricycleMotorized,  // 原動機付き三輪車
    TricycleNonMotorized,   // 人力三輪車
    Group   // 集団
};
```

```
type SurroundModelOtherTwoWheelType = enum {
    Unknown,    // 不明
    HandCart,   // 手押し車
    RickShaw,   // 人力車
    Scooter,    // キックスクータ
    ElectricScooter,    // 電動キックスクータ
    Group   // 集団
};
```

```
type SurroundModelNonCarClassType = enum {
    Unknown,    // 不明
    LivingBeing,    // 生物
    NonLivingBeing  // 非生物
};
```

```
type SurroundModelLivingBeingType = enum {
    Unknown,    // 不明
    Human,  // 人
    Animal  // 動物
};
```

```
type SurroundModelHumanType = enum {
    Unknown,    // 不明
    Pedestrian, // 歩行者
    ChildPedestrian,    // 子供の歩行者
    WheelChair, // 車椅子利用者
    Stroller,   // ベビーカー
    MobilityScooter,    // 電動車椅子・シニアカー
    PoliceMan,  // 警察官
    RoadWorker, // 道路作業員
    Group   // 集団
};
```

```
type SurroundModelAnimalType = enum {
    Unknown,    // 不明
    Dog,    // 犬
    Cat,    // 猫
    Bird,   // 鳥
    Weasel, // イタチ
```

```
    Cow,    // 牛
    Horse,  // 馬
    Deer,   // 鹿
    Bear,   // 熊
    Monkey, // 猿
    WildBoar,   // イノシシ
    Turtle, // 亀
    Group   // 集団
};
```

```
type SurroundModelNonLivingBeingType = enum {
    Unknown,    // 不明
    OnRoadObject,   // 路上物体
    GeneralLandmarkClassification   // 一般ランドマーク分類
};
```

```
type SurroundModelOnRoadObjectType = enum {
    Unknown,    // 不明
    GarbageBag, // ゴミ袋
    LostCargo,  // 落下物
    Tire,   // タイヤ
    BallOrToy,  // ボール・玩具
    BranchOrTree,   // 枝・倒木
    Rock,   // 岩・石
    FallenLeaves,   // 落ち葉
    Pothole,    // 路面の穴
    OverHangingObject   // 頭上の張り出し物
};
```

5.23.8.15. 物体の分類

```
type SurroundModelObjectClassificationType = struct {
    objectMacroClassification : SurroundModelObjectMacroClassificationType, // 物体の大分類。可動物の配列（movingObjects）の要素では，PotentiallyMovingObjectである
    movingObjectClass : SurroundModelPotentiallyMovingObjectClassType,  // 可動物の分類ツリーの最上位（車／車以外／不明）
    carClass : SurroundModelCarClassType?,  // 車（Car）配下の分類。movingObjectClassがCarのときに値を持ちうる
    fourWheelCarClass : SurroundModelFourWheelCarType?, // 四輪車（FourWheelCar）配下の分類
    bigSizeCarClass : SurroundModelBigSizeCarType?, // 大型四輪車配下の分類
    middleSizeCarClass : SurroundModelMiddleSizeCarType?,   // 中型四輪車配下の分類
    smallSizeCarClass : SurroundModelSmallSizeCarType?, // 小型四輪車配下の分類
    truckClass : SurroundModelTruckType?,   // トラック配下の分類
    trailerClass : SurroundModelTrailerType?,   // トレーラ配下の分類
    trainClass : SurroundModelTrainType?,   // 鉄道車両配下の分類
    busClass : SurroundModelBusType?,   // バス配下の分類
```

```
    passengerCarClass : SurroundModelPassengerCarType?, // 乗用車配下の分類
    emergencyVehicleClass : SurroundModelEmergencyVehicleType?, // 緊急車両配下の分類
    agriculturalVehicleClass : SurroundModelAgriculturalVehicleType?,   // 農業機械配
下の分類
    heavyEquipmentClass : SurroundModelHeavyEquipmentType?, // 重機配下の分類
    otherFourWheelCarClass : SurroundModelOtherFourWheelCarType?,   // その他の四輪車
配下の分類
    otherCarClass : SurroundModelOtherCarType?, // その他の車（OtherCar）配下の分類
    bikeClass : SurroundModelBikeType?, // 自動二輪車配下の分類
    bicycleClass : SurroundModelBicycleType?,   // 自転車配下の分類
    tricycleClass : SurroundModelTricycleType?, // 三輪車配下の分類
    otherTwoWheelClass : SurroundModelOtherTwoWheelType?,   // その他の二輪車配下の分
類
    nonCarClass : SurroundModelNonCarClassType?,    // 車以外（NonCar）配下の分類。
movingObjectClassがNonCarのときに値を持ちうる
    livingBeingClass : SurroundModelLivingBeingType?,   // 生物（LivingBeing）配下の分
類
    humanClass : SurroundModelHumanType?,   // 人（Human）配下の分類
    animalClass : SurroundModelAnimalType?, // 動物（Animal）配下の分類
    nonLivingBeingClass : SurroundModelNonLivingBeingType?, // 非生物（NonLivingBeing
）配下の分類
    onRoadObjectClass : SurroundModelOnRoadObjectType?  // 路上物体（OnRoadObject）配
下の分類
};
```

物体の分類は階層構造で表される。上位の階層から順に，判別できた階層の分類を値として持ち，判別できない階層はUnknownまたは値なしとする。各階層はUnknownを備えるため，「車であるが車種は不明」「大型車であるが詳細は不明」といった状態を表現できる。下位階層のフィールドは，その物体が該当する上位分類（movingObjectClass・carClass等）に対応するもののみが値を持つ。

5.23.8.16. 車両属性

```
type SurroundModelLampType = enum {
    Unknown,    // 不明
    Headlight,  // 前照灯
    BrakeLight, // 制動灯
    TurnSignalLeft, // 左方向指示器
    TurnSignalRight,    // 右方向指示器
    HazardLight,    // 非常点滅灯
    ReverseLight,   // 後退灯
    FogLight,   // 前部霧灯・後部霧灯
    DaytimeRunningLight,    // 昼間走行灯
    Other   // その他の灯火
};
```

```
type SurroundModelLampOnStateType = enum {
    Unknown,    // 不明
    Off,    // 消灯
    On  // 点灯
```

```
};
```

```
type SurroundModelLampStateType = struct {
    lampType : SurroundModelLampType,   // 灯火の種類
    confidence : Float64 ([0.0,1.0])?,  // 灯火状態の判定の信頼度。1.0が最大信頼。算出できている場合にのみ値を持つ
    lampOnState : SurroundModelLampOnStateType  // 灯火の点灯状態
};
```

```
type SurroundModelV2xInfoType = struct {
    steeringAngle : Float64?,   // V2Xで得られた操舵角。単位はdeg。左向きが正
    longitudinalAcceleration : Float64?,    // V2Xで得られた前後加減速度。単位はm/s^2。加速方向が正
    vehicleLength : Float64?,   // V2Xで得られた車両の長さ。単位はm
    vehicleWidth : Float64?,    // V2Xで得られた車両の幅。単位はm
    vehicleHeight : Float64?    // V2Xで得られた車両の高さ。単位はm
};
```

```
type SurroundModelVehicleAttributeType = struct {
    lamps : SurroundModelLampStateType[]?,  // 灯火ごとの状態の一覧。観測できている場合にのみ値を持つ
    v2xInfo : SurroundModelV2xInfoType? // V2Xで得られた車両情報。取得できている場合にのみ値を持つ
};
```

5.23.8.17. 人属性

```
type SurroundModelPersonPostureType = enum {
    Unknown,    // 不明
    Standing,   // 起立
    Walking,    // 歩行
    Running,    // 走行
    Sitting,    // 着座・しゃがみ
    LyingDown,  // 横臥
    Other   // その他の姿勢
};
```

```
type SurroundModelPersonAccessoryType = enum {
    Unknown,    // 不明
    NoAccessory,    // 付帯物なし
    Umbrella,   // 傘
    Stroller,   // ベビーカー
    Wheelchair, // 車椅子
    WhiteCane,  // 白杖
```

```
    Bicycle,    // 自転車
    Other   // その他の付帯物
};
```

```
type SurroundModelPersonAttributeType = struct {
    personCount : UInt32?,  // 集団の場合の人数。推定できている場合にのみ値を持つ
    posture : SurroundModelPersonPostureType?,  // 人の姿勢。推定できている場合にのみ
値を持つ
    accessory : SurroundModelPersonAccessoryType?   // 人の付帯物。推定できている場合
にのみ値を持つ
};
```

5.23.8.18. 物体の追跡状態

```
type SurroundModelTrackingStatusType = enum {
    Invalid,    // 異常（観測が有効でない）
    MeasuredTracked,    // 追跡中・検出中
    MeasuredPartiallyMeasured,  // 追跡中・一部検出中
    MeasuredDivided,    // 追跡中・分割した
    MeasuredConnectedMain,  // 追跡中・結合した（本体側）
    PredictedOccluded,  // 照合なし・隠蔽
    PredictedPartiallyMeasured, // 照合なし・一部検出中
    PredictedConnectedRemainder,    // 照合なし・結合した（本体以外）
    PredictedVanished,  // 照合なし・視野内消滅
    PredictedOutOfView  // 照合なし・視野外
};
```

追跡状態は，観測に基づく状態（Measured～：検出により更新されている），予測に基づく状態（Predicted～：検出との照合がなく予測で維持されている），および異常（Invalid）を表す。追跡の生成・維持・消去の段階は，追跡ライフサイクル（trackingLifecycle）で別に表す。

```
type SurroundModelTrackingLifecycleType = enum {
    New,    // 生成（新規に追跡を開始した）
    Tracked,    // 追跡中（追跡が維持されている）
    PlanToBeDeleted,    // 消去予定（追跡の終了が見込まれる）
    Deleted // 消去（追跡を終了した）
};
```

5.23.8.19. 物体上の参照点

```
type SurroundModelReferencePointType = enum {
    Unknown,    // 不明
    Other,  // その他
    FrontLeftTop,   // 前・左・上
    FrontMidwidthTop,   // 前・幅中央・上
    FrontRightTop,  // 前・右・上
```

```
    FrontLeftMidheight, // 前・左・高さ中央
    FrontMidwidthMidheight, // 前・幅中央・高さ中央
    FrontRightMidheight,    // 前・右・高さ中央
    FrontLeftBottom,    // 前・左・下
    FrontMidwidthBottom,    // 前・幅中央・下
    FrontRightBottom,   // 前・右・下
    MidsideLeftTop, // 中央・左・上
    MidsideMidwidthTop, // 中央・幅中央・上
    MidsideRightTop,    // 中央・右・上
    MidsideLeftMidheight,   // 中央・左・高さ中央
    MidsideMidwidthMidheight,   // 中央・幅中央・高さ中央
    MidsideRightMidheight,  // 中央・右・高さ中央
    MidsideLeftBottom,  // 中央・左・下
    MidsideMidwidthBottom,  // 中央・幅中央・下
    MidsideRightBottom, // 中央・右・下
    RearLeftTop,    // 後・左・上
    RearMidwidthTop,    // 後・幅中央・上
    RearRightTop,   // 後・右・上
    RearLeftMidheight,  // 後・左・高さ中央
    RearMidwidthMidheight,  // 後・幅中央・高さ中央
    RearRightMidheight, // 後・右・高さ中央
    RearLeftBottom, // 後・左・下
    RearMidwidthBottom, // 後・幅中央・下
    RearRightBottom,    // 後・右・下
    CenterRearAxle, // 後軸中心
    CenterRearAxleGround,   // 後軸中心の路面投影点
    CenterObject    // 物体の中心
};
```

参照点は，物体を囲む直方体（バウンディングボックス）の格子点（前後×左右×上下の27点），および物体の代表点（後軸中心・後軸中心の路面投影点・物体中心）を表す。位置（position）は，ここで示される点の相対位置である。

5.23.8.20. 車線との紐付け

```
type SurroundModelLaneIdType = String;  // Vehicle.SurroundModel.LogicalStructureの車線の要素ID
```

Vehicle.SurroundModel.LogicalStructureの状態に含まれる車線を識別する要素IDと同じ値をとる文字列である。空文字列であってはならない。

```
type SurroundModelLaneAssociationType = struct {
    associationProbability : Float64 ([0.0,1.0])?,  // 物体がこの紐付けの車線に位置する確からしさ。1.0が最大。算出できている場合にのみ値を持つ
    laneId : SurroundModelLaneIdType?,  // 物体が位置する車線の，LogicalStructureにおける要素ID。機能クラスLaneIdAssociationProvidableを備え，かつ判定できている場合にのみ値を持つ
    relativeLaneNumber : Int32? // 物体が位置する車線の相対車線番号。自車の車線を0とし，左が正。機能クラスRelativeLaneAssociationProvidableを備え，かつ判定できている場合
```

```
にのみ値を持つ
};
```

物体が位置する車線を表す。物体が複数の車線にまたがる場合は，車両実装が判定した主たる1つの車線とする。laneIdとrelativeLaneNumberは同じ1つの車線を表し，いずれか一方または双方が値を持つ（双方が値なしとなることはない）。associationProbabilityは，物体がこの車線に位置する確からしさを[0.0,1.0]で表し，値が大きいほど確からしい。

relativeLaneNumberは，自車が走行している車線を0とし，自車の前方（X軸の正の向き）に向かって左隣の車線を1，右隣の車線を-1として，車線を1つ越えるごとに1ずつ増減する番号である。数えるのは車両が走行できる車線だけであり，路肩・中央帯は数えない。進行方向が自車と異なる車線も，車両が走行できる車線であれば数える。自車の位置と物体の位置との間で車線が増減する区間では，物体の位置における道路の横断方向で，自車の車線に接続する車線を0として数える。接続する車線が無い場合または複数ある場合は，車両実装が判定する。判定できない場合，relativeLaneNumberは値を持たない。自車が車線変更中などで複数の車線にまたがる場合，自車の車線は車両実装が判定した主たる1つの車線とする。自車の車線を判定できない場合，relativeLaneNumberは値なしとなる。

laneIdが指す車線は，LogicalStructureの同じ時点の状態に含まれているとは限らない。利用側は，laneIdをLogicalStructureの状態に含まれる車線の要素IDと照合して用い，含まれない場合はその車線を特定できないものとして扱う。

物体が車線上にない場合（歩道上・路外など）や，車線を判定できない場合，物体のlaneAssociationは値なしとなる。

5.23.8.21. 可動物の状態

```
type SurroundModelMovingObjectType = struct {
    objectId : UInt32,  // 物体ごとの識別子（追跡ID）
    trackingStatus : SurroundModelTrackingStatusType,   // 物体の追跡状態（観測に基づくか予測に基づくか）
    trackingLifecycle : SurroundModelTrackingLifecycleType, // 物体の追跡ライフサイクル（生成・追跡中・消去予定・消去）
    age : UInt32,   // 物体が検出されてからの経過時間。単位はms
    existenceProbability : Float64 ([0.0,1.0]), // 物体が実在する確からしさ（存在確率）。1.0が最大
    trackQuality : Float64 ([0.0,1.0])?,    // 追跡の安定性の指標。1.0が最良。機能クラスTrackQualityProvidableを備える場合にのみ値を持つ
    numberOfValidObservations : UInt32?,    // この物体に寄与した有効観測の数。機能クラスTrackQualityProvidableを備える場合にのみ値を持つ
    detectionCount : UInt32?,   // 累積の検出回数。機能クラスDetectionStatisticsProvidableを備える場合にのみ値を持つ
    missDetectionCount : UInt32?,   // 連続した未検出の回数。機能クラスDetectionStatisticsProvidableを備える場合にのみ値を持つ
    freezeCount : UInt32?,  // 静止状態が継続した回数。機能クラスDetectionStatisticsProvidableを備える場合にのみ値を持つ
    currentUsedSensor : String?,    // この物体の最新情報を得たセンサの識別名。機能クラスDetectionStatisticsProvidableを備える場合にのみ値を持つ
    position : SurroundModelObjectPositionType, // 物体の相対位置と参照点および位置の推定誤差
    laneAssociation : SurroundModelLaneAssociationType?,    // 物体と車線との紐付け。
```

```
機能クラスLaneIdAssociationProvidableまたはRelativeLaneAssociationProvidableを備え，かつ当該物体について紐付けを判定できている場合にのみ値を持つ
    dynamics : SurroundModelObjectDynamicsType, // 物体の速度・加速度・回転とそれぞれの推定誤差
    shape : SurroundModelObjectShapeType,   // 物体の向き・サイズ・形状把握状態とそれぞれの推定誤差
    classification : SurroundModelObjectClassificationType, // 物体の階層分類
    vehicleAttribute : SurroundModelVehicleAttributeType?,  // 物体が車両の場合の属性（灯火状態・V2X情報）。機能クラスVehicleAttributeProvidableを備え，かつ物体が車両の場合にのみ値を持つ
    personAttribute : SurroundModelPersonAttributeType? // 物体が人の場合の属性（姿勢・付帯物・人数）。機能クラスPersonAttributeProvidableを備え，かつ物体が人の場合にのみ値を持つ
};
```

5.23.8.22. タイムスタンプ

```
type TimestampUsType = UInt64;  // 車内共通時刻基盤の起点を基準とする測定時刻。単位はμs。複数のビークルAPIで共通に用いる共通データ型であり，本仕様内に定義する。
```

5.23.8.23. サンプル品質の種別

```
type DataQualifierType = enum {
    Normal, // 正常
    Unavailable,    // 利用不可
    Degraded,   // 劣化
    TestMode,   // テストモード
    Invalid // 無効（値も誤差も読んではならない）
};
```

5.23.8.24. SurroundModel状態

```
type SurroundModelStatusType = struct {
    timestamp : TimestampUsType,
    dataQualifier : DataQualifierType,
    numberOfValidSensors : UInt32,
    currentPerformance : SurroundModelPerformanceType,
    performanceChangeReasons : enumSet(SurroundModelPerformanceChangeReasonType),
    numberOfValidMovingObjects : UInt32,
    movingObjects : SurroundModelMovingObjectType[]
};
```

物体の位置・速度・向きの座標系はISO 8855の車両座標系（原点：後軸中心の路面投影点）とする。各推定誤差が表す信頼水準は，95%に固定する（誤差楕円以外の推定誤差については2σ相当）。存在確率・追跡品質はFloat64の[0.0,1.0]で表す。timestampは，車内共通時刻基盤に基づく測定時刻を表す。ただし，dataQualifierがUnavailableのサンプルのtimestampは，サンプルの生成時刻を表す（測定時刻ではな

い）。dataQualifierがInvalidまたはUnavailableの場合，物体の配列（movingObjects）およびその要素数（numberOfValidMovingObjects）を読んではならない。dataQualifierがUnavailableの場合は，有効外界センサ数（numberOfValidSensors），現在の認識性能（currentPerformance）および変化理由（performanceChangeReasons）も読んではならない。

5.23.8.25. SurroundModelイベント

```
type SurroundModelEventKindType = enum {
    FaultDetected,
    FaultRecovered,
    StatusReported,
    PerformanceChanged,
    EventOverflow,
    ConfigChanged
};
```

```
type SurroundModelEventType = struct {
    eventInfo : variant(SurroundModelEventKindType) {
        StatusReported {
            timestamp : TimestampMsType ,
            status : SurroundModelStatusType
        },
        EventOverflow {
            noLostEvents : UInt32
        }
    }
};
```

5.23.9. サービスコールの詳細規定

getConfig

SurroundModelの構成情報の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
config	SurroundModelConfigType	SurroundModelの構成情報（座標系・状態報告の周期・視野・認識性能・処理能力の公称値）

【エラーコード】

なし

【機能】

SurroundModelに関する構成情報（座標系・状態報告の周期，および搭載時の公称の視野・認識性能・処理能力）を返す。稼働中の実際の視野・認識性能・処理能力は，getStatusが返す状態currentPerformanceで表される。

getStatus

周辺物体リストの参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
status	SurroundModelStatusType	自車周辺の物体リスト（全体共通情報，現在の認識性能と変化理由，および可動物の配列）

【エラーコード】

E_DENIED_ACCESS	周辺環境モデルの参照が認められていない
E_OBJECT_FAULT	故障中（外界センサ系の異常により周辺環境モデルを提供できない状態）

【機能】

自車周辺の物体リストに関するすべての状態を返す。返される各値は観測に基づく推定値であり，精度は保証されない。取得した値の信頼度は，併せて返されるdataQualifierと，物体ごとの存在確率・追跡品質・各推定誤差によって判断する。dataQualifierがInvalidまたはUnavailableの場合，物体の配列（movingObjects）およびその要素数（numberOfValidMovingObjects）を読んではならない。物体を1つも検出していない状態は故障ではなく，物体の配列が空となる。返される状態には，稼働中の実際の視野・認識性能・処理能力（currentPerformance）と，それが公称値から変化している理由（performanceChangeReasons）が含まれる。

周辺環境モデルは周辺の人・車を推定できる情報であるため，本サービスコールはアクセス制御の対象である。

notify

SurroundModelイベントの通知要求

【パラメータ】

eventFilter	enumSet (SurroundModelEventKindType)?	通知するイベントの種類の集合

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【エラーコード】

E_DENIED_ACCESS	周辺環境モデルの参照が認められていない
E_INVALID_PARAMETER	eventFilterが有効範囲外

【機能】

SurroundModelで発生したeventFilterで指定したイベントの通知を要求する。イベントを通知するためのイベントキューを生成し，その通知ハンドルをnotifyHandleに返す。

eventFilterを省略した場合，すべての種類のイベントの通知を要求する。

通知されるイベントには，報告時点の状態の全量を運ぶStatusReportedイベントが含まれることがある。
周辺環境モデルは周辺の人・車を推定できる情報であるため，本サービスコールの呼び出しそのものが，eventFilterの内容によらず（eventFilterを省略した場合を含む），アクセス制御の対象であり，アクセスが認められていないアプリケーションに対してはE_DENIED_ACCESSエラーとなる。

unnotify

SurroundModelイベントの通知停止

【パラメータ】

notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正

【機能】

notifyHandleで指定した通知を停止する。指定した通知用のイベントキューは削除される。

getEvent

SurroundModelイベントの取得

【パラメータ】

notifyHandle	NotifyHandleType	イベントキューを識別するための通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

event	SurroundModelEventType	取得したイベント

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正
E_NO_DATA	イベントキューが空

【機能】

notifyHandleで指定した通知用のイベントキューから，先頭のイベントを取り出す。取り出したイベントは，イベントキューから削除される。



イベントがパラメータ等で渡されてくるプログラミング環境向けの物理APIでは，用意する必要がない。

5.24. Vehicle.SurroundModel.LogicalStructure（略称：LogicalStructure）

5.24.1. 位置付けと機能

Vehicle.SurroundModel.LogicalStructure（略称：LogicalStructure）は，自車周辺の道路の論理構造を提供するシングルインスタンスオブジェクトである。走行可能領域，道路・道路区間・車線群・車線，車線の境界，交差点等のエリア，交通規制，道路端・中央帯・路肩，およびそれらの接続関係と経路構造を状態として提供する。状態には，構造全体に共通する時刻・サンプル品質・版番号が含まれる。

5.24.1.1. LogicalStructureの位置付けと性質

LogicalStructureが表すのは，道路本線部を中心とした静的な道路構造である。走行可能領域は，道路として走行可能な範囲を表し，可動物（車両・歩行者等）の存在や死角の有無とは無関係に定まる。LogicalStructureは，可動物の情報を含まない。

道路構造の情報源は規定しない。高精度道路地図に由来する情報，外界センサによる認識のみに基づく推定結果，両者を融合した結果のいずれも提供できる。主要な要素は，その要素の情報の由来（source）と信頼度（confidence）を保持でき，由来の異なる要素が1つの状態の中に共存してよい。情報源や走行環境による精度の違いは，これらの値として表現され，APIとしての精度の保証は行わない。

5.24.1.2. 道路論理構造の3層

道路論理構造は，次の3層で表す。

- 物理層：幾何形状を表す。点（LogicalStructurePointType），点列（LogicalStructurePolyline3DType），多角形（LogicalStructurePolygonType）からなる。
- 関係層：車線・エリア・規制の意味構造を表す。道路（Road），道路区間（RoadSection），車線群（LaneGroup），車線（Lane），車線境界（LaneBoundary），エリア（Area），規制要素（RegulatoryElement），道路端（RoadEdge），中央帯（Median），路肩（Shoulder），走行可能領域（DrivableArea）からなる。
- トポロジ層：接続関係のグラフを表す。エリアを経由しない車線の先行・後続および左右の隣接の関係（車線の項目）と，エリアを経由する車線間の接続（状態laneConnections）からなる。エリアを経由する接続は，laneConnectionsだけで表す。

関係層の要素は，次の包含関係をもつ。道路区間に属する要素は，その道路区間の中に配列として含まれる。

- Road（道路）
  - RoadSection（道路区間）
    - LaneGroup（車線群）
      - Lane（車線）
    - LaneBoundary（車線境界），Area（エリア），RegulatoryElement（規制要素），RoadEdge（道路端），Median（中央帯），Shoulder（路肩）
- DrivableArea（走行可能領域）

これらに加えて，機能クラスPathStructureProvidableを備えている場合は，目安となる経路とその連結関係を表す経路構造（PathStructure）を提供する。

5.24.1.3. 要素の識別と参照

各要素は，要素ID（LogicalStructureElementIdType）で識別される。要素IDは，1つの状態の中で，要素の種類によらず一意である。要素間の関係（車線から車線境界への参照，車線の先行・後続，規制要素の適用対象など）は，要素IDによる参照で表す。参照先の要素は，同じ状態の中に含まれていなければならない。ただし，車線の先行・後続および隣接の参照は，提供範囲の外にある車線を指してもよい（後述）。

同一の要素は，状態に含まれ続ける間，同一の要素IDを保つ。情報源の都合で要素の同一性を保てない場合は，元の要素が削除され，新しい要素IDをもつ要素が追加されたものと扱う。異なる要素に同一の要素IDを付けてはならない。一度状態に含まれなくなった要素の要素IDを，別の要素に付けてはならない（時間をまたいでも要素IDを再利用しない）。

5.24.1.4. 座標系

状態に含まれるすべての幾何は，構成情報coordinateSystemが宣言する単一の座標系で表す。座標系は次のいずれかであり，WGS84を標準とする。車両座標系を用いるかどうかは実装に依存する。

- WGS84：点のxは経度（deg，東経が正），yは緯度（deg，北緯が正），zは楕円体高（m）を表す。WGS84の実現版は構成情報geodeticDatumで宣言される。
- 車両座標系：ISO　8855の車両座標系（原点：後軸中心の路面投影点，X軸：車両前方，Y軸：車両左方，Z軸：上方の右手系）とし，点のx・y・zはそれぞれの軸の座標（m）を表す。座標の基準となる車両の位置と向きは，状態のtimestampの時点のものとする。

車両座標系の場合，自車が移動すると，道路構造が変化していなくても，すべての座標値が変化する。この自車の移動に伴う座標値の変化は，道路構造の変化とは扱わない（版番号revisionを更新せず，StructureChangedイベントを発生させない）。getStatusは，呼び出しのたびに，その時点で最新のtimestampの車両の位置と向きを基準とする座標を返す。利用側が，取得した状態を後の時刻に用いる場合は，timestampの時点からの自車の移動量を用いて座標を換算する。

車両基準点は構成情報referencePointで宣言され，独自の基準点は用いない。

構成情報coordinateSystemとgeodeticDatumは，稼働中に変化しない。利用側は，一度取得したこれらの値を，以後に取得するすべての状態の解釈に用いることができる。

5.24.1.5. 道路参照系による位置の表現

道路に沿った位置は，道路の基準線に沿った距離sによる線形参照でも表すことができる。sは，道路の基準線上に定めた起点からの，基準線に沿った距離であり，単位はmとする。sの起点は道路の基準線上の点であり，その道路が状態に含まれ続ける間は変わらない。道路区間の範囲は，sの区間（startS以上endS以下）で表す。道路区間の位置は，座標による幾何と，sによる線形参照を併用して表すことが望ましい。

状態に含まれる道路の基準線（referencePath）は，道路の基準線のうち，含まれる道路区間を覆う部分である。referencePathの最初の点は，sectionsの最初の道路区間のstartSの位置に対応する。sの起点は道路が状態に含まれ続ける間は変わらないため，提供範囲が移動して含まれる道路区間が変わっても，同じ道路区間のstartSとendSは変わらない。

5.24.1.6. 提供範囲と構造の変化の通知

getStatusが返すのは，自車周辺の一定範囲（提供範囲）の道路構造である。提供範囲は，車両基準点を中心とし，構成情報coverageRadiusを半径とする水平面上の円である。状態には，提供範囲と一部でも重なる要素が，情報源から得られる限りすべて含まれる。提供範囲の外にある要素を含んでもよい。含まれる要素の幾何は，提供範囲の境界で切り取らない。ただし，道路（Road）については，提供範囲と重なる道路区間のみをsectionsに含め，基準線referencePathは，含まれる道路区間の全体を覆う部分とする。

自車が走行すると提供範囲は移動し，それに伴って状態に含まれる要素が追加・削除される。また，地図の更新や認識結果の確定などにより，提供範囲内の要素が追加・削除され，または要素の属性が変化する。これらの道路構造の変化は，StructureChangedイベントで通知される。StructureChangedイベントは変化の内容を運ばない。利用側は，StructureChangedイベントを受けたら，getStatusにより状態を取得し直す。LogicalStructureは，状態を周期的に報告するイベントを持たない。

状態は版番号revisionを持つ。revisionは，StructureChangedイベントを発生させるたびに1ずつ増加し，StructureChangedイベントの関連情報revisionは，そのイベントに対応する変化を反映した状態のrevisionを表す。利用側は，保持している状態のrevisionとイベントのrevisionを比較することで，保持している状態が変化を反映済みかどうかを判断できる。

幾何形状（点の座標，s，高さ，幅，車線幅の変化）と信頼度（confidence）の値の変化は，StructureChangedイベントの発生条件に該当しない。これらは再推定によって少しずつ変化する連続量であり，その最新値はgetStatusによって取得する。それ以外の項目の変化（速度制限の値や車線の並び順など，数値の項目を含む）は，StructureChangedイベントで通知される。

5.24.1.7. 時刻とサンプル品質

状態には，道路構造全体の時刻timestampとサンプル品質dataQualifierが含まれる。timestampは，車内共通時刻基盤に基づく時刻であり，同じ時刻基盤に基づく他のビークルAPIのタイムスタンプと直接比較できる。タイムスタンプ自体の誤差上限は，構成情報timestampAccuracyで宣言される。

dataQualifierは，取得できた状態の品質を表す。故障（道路構造を提供できない状態。E_OBJECT_FAULTエラーおよびFaultDetectedイベント）が「道路構造を出力できない状態」を表すのに対し，dataQualifierは「出力された道路構造がどれだけ信用できるか」を表す。地図の整備範囲外など，情報源から道路構造が得られない区域にいることは故障ではない。この場合，dataQualifierはUnavailableとなり，要素の配列は空となる。

5.24.1.8. 状態の参照に伴うプライバシのリスク

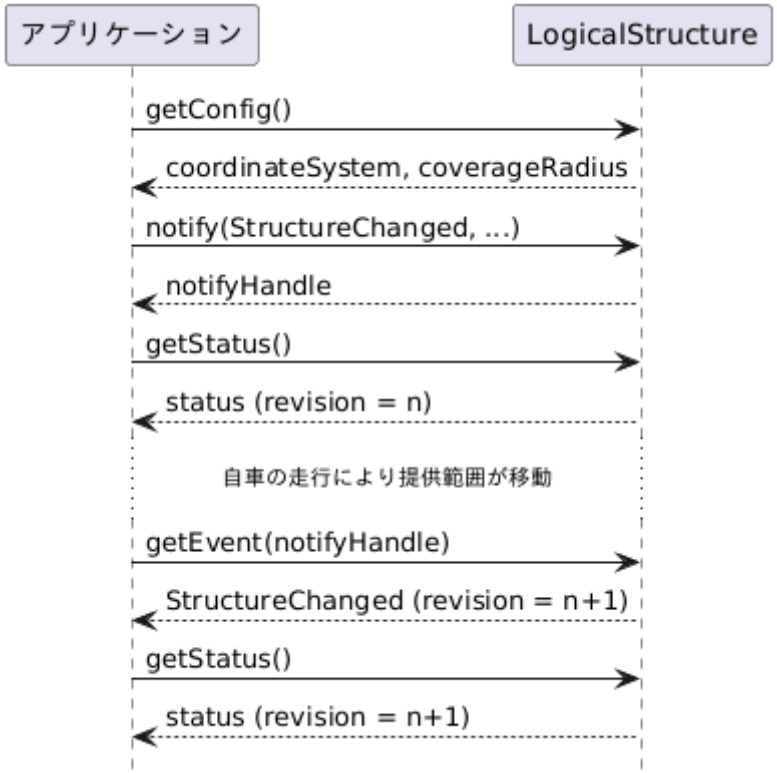
自車周辺の道路構造は，提供範囲の中心が自車であるため，自車の現在位置を推定できる情報である。座標系がWGS84の場合は座標から自車の位置がほぼ直接わかり，車両座標系の場合も，道路の名称や形状を

地図と照合することで位置を特定できる。そのため，状態を参照するサービスコール（getStatus）はリスククラスRiskPrivacyの対象であり，RiskPrivacyに対応していないアプリケーションに対してはE_RISK_APPLICATIONエラー，ユーザが状態の開示を承認していない場合はE_RISK_USERエラーとなる。イベントの通知要求（notify）により通知されるイベントは状態の内容を運ばないが，StructureChangedイベントの発生の時刻と理由は自車の走行と相関し，また，状態を取得し直す契機となる。このため，notifyの呼び出しそのものを，eventFilterで指定するイベントの種類によらず，RiskPrivacyの対象とする。

5.24.1.9. 使用例

自車周辺の車線構成を取得し，道路構造が変化したら取得し直す例を示す。アプリケーションは，まず構成情報で座標系と提供範囲を確認し，イベントの通知を要求した上で状態を取得する。通知を先に要求するのは，通知の要求までの間に起きた変化を取りこぼさないためである。

```
[01] config = getConfig()             // 座標系・提供範囲の確認
[02] frame  = config.coordinateSystem // Wgs84 または VehicleFrame
[03] handle = notify({StructureChanged, FaultDetected, FaultRecovered})
[04] status = getStatus()             // 周辺の道路構造
[05] use(status, frame)
[06] loop:
[07]   event = getEvent(handle)
[08]   if (event is StructureChanged
[09]       and event.revision != status.revision)
[10]     or event is FaultRecovered:    // 故障からの回復後も
[11]     status = getStatus()          // 取得し直す
[12]     use(status, frame)
```



アプリケーションは，StructureChangedイベントまたはFaultRecoveredイベントを受けたときだけ状態を取得し直せばよく，道路構造を周期的に取得し直す必要はない。ただし，幾何形状や信頼度の再推定に

よる値の変化はイベントで通知されないため，最新の値が必要な場合はgetStatusを呼び出す。座標系が車両座標系の場合，取得した状態の座標は状態のtimestampの時点の車両の位置と向きを基準とするため，後の時刻に用いる場合は自車の移動量で換算する。

5.24.2. 機能クラス

LogicalStructureは，以下の機能クラスを持つ。

PathStructureProvidable	経路構造（pathStructure）を提供することができる

5.24.3. リスククラス

LogicalStructureは，以下のリスククラスを持つ。

RiskPrivacy	自車周辺の道路構造から自車の位置が推定されることに伴うプライバシのリスク（例：未対応アプリケーションへの道路構造の開示，ユーザが承認していない状態での開示）

5.24.4. 構成情報

LogicalStructureは，以下の構成情報を持つ。

capabilities	enumSet (LogicalStructureCapabilityType)	このLogicalStructureが持つ機能
riskClasses	enumSet (LogicalStructureRiskType)	このLogicalStructureのリスククラスの集合
coordinateSystem	LogicalStructureCoordinateSystemType	状態に含まれるすべての幾何の座標系の宣言（WGS84または車両座標系）
geodeticDatum	String?	WGS84の実現版の宣言。coordinateSystemがWgs84の場合は必ず値を持ち，それ以外の場合は値なしとする
referencePoint	String	車両基準点の宣言（ISO 8855の車両座標系原点：後軸中心の路面投影点）。車両座標系の原点，および提供範囲の中心となる
coverageRadius	Float64	提供範囲の半径。単位はm。0より大きい値をとる。車両基準点を中心とする水平面上の円の半径である
timestampAccuracy	UInt32	この車両のタイムスタンプ誤差上限の宣言。単位はμs

coordinateSystemとgeodeticDatumは，稼働中に変化しない。coverageRadiusが変化した場合は，ConfigChangedイベントが発生する。

5.24.5. 状態

LogicalStructureは，以下の状態を持つ。

timestamp	TimestampUsType	道路構造の時刻。単位はμs。車内共通時刻基盤に基づく
dataQualifier	DataQualifierType	この状態（道路構造全体）の品質。InvalidまたはUnavailableの場合，要素を読んではならない
revision	UInt32	道路構造の版番号。StructureChangedイベントを発生させるたびに1ずつ増加する
drivableAreas	LogicalStructureDrivableAreaType[]	走行可能領域の配列
roads	LogicalStructureRoadType[]	道路の配列。道路区間・車線群・車線・車線境界・エリア・規制要素・道路端・中央帯・路肩は，道路の中に含まれる
laneConnections	LogicalStructureConnectionType[]	エリア（交差点等）を経由する車線間の接続関係の配列
pathStructure	LogicalStructurePathStructureType?	経路構造

timestampは，この道路構造が表す時点の時刻であり，車内共通時刻基盤に基づく。外界センサによる認識に基づく要素を含む場合は，その認識に用いた観測の測定時刻とする。座標系が車両座標系の場合は，座標の基準となる車両の位置と向きの時刻でもある。dataQualifierがUnavailableの状態のtimestampは，状態の生成時刻を表す（測定時刻ではない）。タイムスタンプの誤差上限は，構成情報timestampAccuracyで宣言される。

dataQualifierは，この状態の品質を表す。Invalidは，道路構造を生成する処理自身が自分の出力を信用できない状態（起動直後の初期化中・自己位置が定まらず地図と照合できない状態など）であり，要素を読んではならない。Degradedは，道路構造は使えるが通常品質でない状態（例：地図の鮮度が低下している場合，悪天候により認識品質が低下している場合）であり，個々の要素の確からしさは，要素ごとのsourceとconfidenceで判断する。Unavailableの状態でも，要素を読んではならない。

道路構造を提供できない状態は故障と扱い，故障中の状態の参照（getStatus）はE_OBJECT_FAULTエラーとなる。故障から回復した場合（FaultRecoveredイベント），利用側はgetStatusにより状態を取得し直す。地図の整備範囲外など，情報源から道路構造が得られない区域にいることは故障ではない。この場合，dataQualifierはUnavailableとし，要素の配列（drivableAreas，roads，laneConnections）は空とし，pathStructureは値なしとする。

revisionは，StructureChangedイベントを発生させるたびに1ずつ増加し，最大値の次は0に戻る。StructureChangedイベントを発生させない変化では，revisionは変化しない。

pathStructureは，機能クラスPathStructureProvidableを備え，かつ経路構造を生成できている場合にのみ値を持つ。

各要素の任意項目は，情報源から当該の値が得られている場合にのみ値を持つ。配列の任意項目では，値なしは「情報が得られていない」ことを，空の配列は「該当するものがない」ことを表す。

5.24.6. サービスコール一覧

LogicalStructureに対するサービスコールは以下の通り。

getConfig	LogicalStructureの構成情報の参照
getStatus	道路論理構造の参照

notify	LogicalStructureイベントの通知要求
unnotify	LogicalStructureイベントの通知停止
getEvent	LogicalStructureイベントの取得

5.24.7. イベント

LogicalStructureのイベント種別には，故障の検知・回復と，道路構造の変化がある。道路構造の変化を通知するStructureChangedイベントは，変化の内容を運ばない。変化後の道路構造は，getStatusにより取得する。FaultRecoveredイベントを受けた場合も，利用側はgetStatusにより状態を取得し直す。

LogicalStructureに対するイベント種別は以下の通り。

FaultDetected	故障検知（道路構造を提供できない状態）
FaultRecovered	故障から回復
StructureChanged	道路構造の変化（発生条件は下記）
EventOverflow	イベントキューのオーバフロー
ConfigChanged	構成情報の変更

StructureChangedイベントは，getStatusが返す状態について，次のいずれかの変化が生じたときに発生する。

- 提供範囲の移動に伴い，状態に含まれる要素が追加または削除された（理由CoverageShifted）。
- 提供範囲内の要素が追加または削除された，または要素の項目のうち，幾何形状と信頼度を除く項目の値が変化した（理由ContentUpdated）。数値の項目（速度制限の値，車線の並び順など）の変化もこれに含む。
- dataQualifierが変化した（理由DataQualifierChanged）。

ここで，幾何形状と信頼度とは，点の座標（点列・多角形・停止線・経路の形状を含む），道路区間のstartSとendS，走行可能領域の有効高さ範囲，車線境界の高さと幅，車線幅の変化（widthProfile），および信頼度（confidence）をいう。これらの値の変化のみでは，StructureChangedイベントは発生しない。座標系が車両座標系の場合，自車の移動に伴う座標値の変化のみでも発生しない。

同時に複数の理由に該当する変化が生じた場合は，それらの理由をreasonsに含む1つのイベントとして発生させてよい。StructureChangedイベントを1つ発生させるたびに，状態のrevisionは1ずつ増加する。

この内，イベント種別がStructureChangedの場合には，以下の関連情報を持つ。

reasons	enumSet (LogicalStructureChangeReasonType)	変化の理由の集合
revision	UInt32	変化後の状態の版番号

また，イベント種別がEventOverflowの場合には，以下の関連情報を持つ。

noLostEvents	UInt32	失われたイベントの数

5.24.8. データ型の定義

5.24.8.1. LogicalStructureの機能クラス

```
type LogicalStructureCapabilityType = enum {
    PathStructureProvidable
};
```

5.24.8.2. LogicalStructureに対するリスククラス

```
type LogicalStructureRiskType = enum {
    RiskPrivacy
};
```

5.24.8.3. 座標系

```
type LogicalStructureCoordinateSystemType = enum {
    Wgs84,  // WGS84（経度・緯度・楕円体高）
    VehicleFrame    // ISO 8855の車両座標系（原点：後軸中心の路面投影点）
};
```

5.24.8.4. LogicalStructureの構成情報（getConfigが返すデータ型）

```
type LogicalStructureConfigType = struct {
    capabilities : enumSet(LogicalStructureCapabilityType),
    riskClasses : enumSet(LogicalStructureRiskType),
    coordinateSystem : LogicalStructureCoordinateSystemType,
    geodeticDatum : String?,
    referencePoint : String,
    coverageRadius : Float64,
    timestampAccuracy : UInt32
};
```

5.24.8.5. 要素ID

```
type LogicalStructureElementIdType = String;    // 要素ID
```

道路論理構造の要素を識別するIDである。空文字列であってはならない。1つの状態の中で，要素の種類（道路・道路区間・車線群・車線・車線境界・エリア・規制要素・道路端・中央帯・路肩・走行可能領域・経路構造・経路要素）によらず一意である。同一の要素は，状態に含まれ続ける間，同一の要素IDを保つ。異なる要素に同一の要素IDを付けてはならない。

5.24.8.6. 幾何形状（物理層）

```
type LogicalStructurePointType = struct {
    x : Float64,    // WGS84の場合は経度（deg，東経が正），車両座標系の場合はX座標（m，車両前方が正）
    y : Float64,    // WGS84の場合は緯度（deg，北緯が正），車両座標系の場合はY座標（m，車両左方が正）
    z : Float64 // WGS84の場合は楕円体高（m），車両座標系の場合はZ座標（m，上方が正）
};
```

3次元の点である。x・y・zの意味は，構成情報coordinateSystemが宣言する座標系による。WGS84の場合，xは[-180.0,180.0]，yは[-90.0,90.0]の範囲の値をとる。

```
type LogicalStructurePolyline3DType = LogicalStructurePointType[];  // 3次元の点列
```

3次元の点列（折れ線）である。2点以上からなり，隣り合う点を線分で結んだ形状を表す。点の並びは形状の向きを表す。

```
type LogicalStructurePolygonType = LogicalStructurePointType[]; // 多角形の頂点列
```

多角形の外形を，頂点の並びで表す。3点以上からなり，最後の頂点と最初の頂点は線分で結ばれているものとする（最初の頂点を末尾に繰り返さない）。頂点の並びの向き（時計回り・反時計回り）は問わない。穴をもつ多角形は表さない。

```
type LogicalStructureHeightRangeType = struct {
    minimum : Float64,  // 有効高さの下限。単位はm
    maximum : Float64   // 有効高さの上限。単位はm
};
```

要素が有効である高さの範囲を，座標系のz（WGS84の場合は楕円体高，車両座標系の場合はZ座標）で表す。立体交差など，同じ水平位置に複数の要素が重なる場合に，要素を区別するために用いる。maximumはminimum以上でなければならない。

5.24.8.7. 情報の由来

```
type LogicalStructureSourceType = enum {
    Unknown,    // 不明
    Perception, // 外界センサによる認識
    HdMap,  // 高精度道路地図
    Fused,  // 高精度道路地図と外界センサによる認識の融合
    Inferred    // 他の要素からの推論による補完
};
```

要素の情報の由来を表す。主要な要素は，由来sourceと信頼度confidenceを保持できる。confidenceは，要素が実在し記述どおりであることの確からしさを0.0から1.0の値で表し，値が大きいほど確からしいことを示す。confidenceの算出方法は実装に依存する。

5.24.8.8. 走行可能領域

```
type LogicalStructureDrivableAreaType = struct {
    id : LogicalStructureElementIdType, // 走行可能領域の要素ID
    geometry : LogicalStructurePolygonType, // 走行可能領域の外形
    roadId : LogicalStructureElementIdType?,    // 所属する道路の要素ID
    heightRange : LogicalStructureHeightRangeType?, // 有効高さ範囲
    confidence : Float64 ([0.0,1.0])?,  // 信頼度
    source : LogicalStructureSourceType?    // 情報の由来
};
```

道路として走行可能な領域を表す。可動物の存在や死角の有無とは無関係に，道路として走行可能な範囲を示す。roadIdが値を持つ場合，参照先の道路は同じ状態のroadsに含まれていなければならない。

5.24.8.9. 道路

```
type LogicalStructureRoadClassType = enum {
    Unknown,    // 不明
    Motorway,   // 自動車専用の高速道路
    Trunk,  // 幹線道路
    Primary,    // 主要道路
    Secondary,  // 補助的な幹線道路
    UrbanRoad,  // 市街地の一般道路
    ServiceRoad,    // 施設への出入りや側道などのサービス道路
    Ramp    // ランプ（出入路・連絡路）
};
```

```
type LogicalStructureDrivingSideType = enum {
    Unknown,    // 不明
    Left,   // 左側通行
    Right   // 右側通行
};
```

```
type LogicalStructureRoadType = struct {
    id : LogicalStructureElementIdType, // 道路の要素ID
    name : String?, // 道路の名称
    classification : LogicalStructureRoadClassType, // 道路種別
    drivingSide : LogicalStructureDrivingSideType,  // 左側通行または右側通行
    referencePath : LogicalStructurePolyline3DType, // 道路の基準線
    sections : LogicalStructureRoadSectionType[]    // 道路を構成する道路区間
};
```

論理的にひとまとまりの道路の単位を表す。道路に沿った距離sは，道路の基準線上に定めた起点からの，基準線に沿った距離（m）である。sの起点は道路の基準線上の点であり，その道路が状態に含まれ続ける間は変わらない。

sectionsには，提供範囲と重なる道路区間のみを，sの小さい順に含める。referencePathは，道路の基準線のうち，sectionsに含まれる道路区間の全体を覆う部分とする。referencePathの最初の点は，sectionsの最初の道路区間のstartSの位置に対応し，referencePath上の点のsは，その点までのreferencePathに沿った距離にこのstartSを加えた値となる。提供範囲が移動して含まれる道路区間が変わっても，同じ道路区間のstartSとendSは変わらない。

5.24.8.10. 道路区間

```
type LogicalStructureRoadSectionType = struct {
    id : LogicalStructureElementIdType, // 道路区間の要素ID
    roadId : LogicalStructureElementIdType, // 所属する道路の要素ID
    startS : Float64,   // 区間の開始位置（s）。単位はm
    endS : Float64, // 区間の終了位置（s）。単位はm
    geometry : LogicalStructurePolyline3DType?, // 区間全体の幾何
    laneGroups : LogicalStructureLaneGroupType[],   // 車線群
    boundaries : LogicalStructureLaneBoundaryType[],    // 区間内の車線境界の一覧
    areas : LogicalStructureAreaType[]?,    // 交差点等のエリア
    regulatoryElements : LogicalStructureRegulatoryElementType[]?,  // 規制要素
    roadEdges : LogicalStructureRoadEdgeType[]?,    // 道路端
    medians : LogicalStructureMedianType[]?,    // 中央帯
    shoulders : LogicalStructureShoulderType[]?,    // 路肩
    confidence : Float64 ([0.0,1.0])?,  // 信頼度
    source : LogicalStructureSourceType?    // 情報の由来
};
```

車線構成・境界構成・主要な属性がおおむね一定である道路の区間を表す。roadIdは，この道路区間を含む道路の要素IDと一致する。区間の範囲は，所属する道路の基準線に沿った距離sの区間（startS以上endS以下）で表す。endSはstartSより大きくなければならない。

boundariesには，この道路区間の車線およびエリアが参照するすべての車線境界を含める。

5.24.8.11. 車線群

```
type LogicalStructureDirectionType = enum {
    Unknown,    // 不明
    WithReference,  // 道路の基準線と同じ向き
    AgainstReference,   // 道路の基準線と逆の向き
    Bidirectional   // 双方向
};
```

```
type LogicalStructureLaneGroupFunctionType = enum {
    Unknown,    // 不明
    Mainline,   // 本線
    Ramp,   // ランプ
    Auxiliary,  // 付加車線（登坂車線・加減速車線等）
    Service,    // 側道等のサービス用
    ShoulderRunning // 路肩を走行車線として用いる区間
```

```
};
```

```
type LogicalStructureLaneGroupType = struct {
    id : LogicalStructureElementIdType, // 車線群の要素ID
    sectionId : LogicalStructureElementIdType,  // 所属する道路区間の要素ID
    direction : LogicalStructureDirectionType,  // 道路の基準線に対する進行方向
    function : LogicalStructureLaneGroupFunctionType,   // 車線群の機能
    lanes : LogicalStructureLaneType[]  // 所属する車線
};
```

同一方向の，または機能的にまとまった車線の集合を表す。sectionIdは，この車線群を含む道路区間の要素IDと一致する。lanesは，車線の並び順（index）の小さい順に並べる。

5.24.8.12. 車線

```
type LogicalStructureLaneKindType = enum {
    Unknown,    // 不明
    Driving,    // 走行車線
    Exit,   // 流出車線
    Entry,  // 流入車線
    Merging,    // 合流車線
    Turn,   // 右左折車線
    Shoulder,   // 路肩
    Parking,    // 駐車帯
    Bicycle,    // 自転車通行帯
    Bus,    // バス専用車線
    Emergency,  // 緊急用の車線
    Restricted  // 通行が制限された車線
};
```

```
type LogicalStructureLaneType = struct {
    id : LogicalStructureElementIdType, // 車線の要素ID
    laneGroupId : LogicalStructureElementIdType,    // 所属する車線群の要素ID
    index : UInt8,  // 車線群内の並び順。1から始まる
    type : LogicalStructureLaneKindType,    // 車線種別
    centerline : LogicalStructurePolyline3DType?,   // 車線の中心線
    leftBoundaryId : LogicalStructureElementIdType, // 左側の車線境界の要素ID
    rightBoundaryId : LogicalStructureElementIdType,    // 右側の車線境界の要素ID
    predecessorLaneIds : LogicalStructureElementIdType[]?,  // 先行する（接続元の）車線の要素ID
    successorLaneIds : LogicalStructureElementIdType[]?,    // 後続する（接続先の）車線の要素ID
    leftAdjacentLaneId : LogicalStructureElementIdType?,    // 左隣に隣接する車線の要素ID
    rightAdjacentLaneId : LogicalStructureElementIdType?,   // 右隣に隣接する車線の要素ID
    widthProfile : LogicalStructureWidthProfileType?,   // 車線幅の変化
```

```
    speedLimit : LogicalStructureSpeedLimitType?,   // 速度制限
    accessRules : LogicalStructureAccessRuleType[]?,    // 通行制約
    maneuverRestrictions : enumSet(LogicalStructureManeuverRestrictionType)?,   // 操
作制約
    laneChange : LogicalStructureLaneChangeType?,   // 左右への車線変更の可否
    isDrivable : Bool?, // 走行可能か
    isTemporary : Bool?,    // 工事等による暫定的な車線か
    confidence : Float64 ([0.0,1.0])?,  // 信頼度
    source : LogicalStructureSourceType?    // 情報の由来
};
```

走行の単位として扱う，道路論理構造の中心となる要素である。車線の左右は，車線の進行方向に向かって定める。進行方向は，所属する車線群のdirectionがWithReferenceの場合は道路の基準線と同じ向き，AgainstReferenceの場合は逆の向きとし，BidirectionalまたはUnknownの場合は道路の基準線と同じ向きとみなす。indexは，同じ進行方向に向かって左端の車線を1とし，右へ順に1ずつ増える。

laneGroupIdは，この車線を含む車線群の要素IDと一致する。leftBoundaryIdとrightBoundaryIdは，この車線を含む道路区間のboundariesに含まれる車線境界を参照する。

先行・後続の関係は，車線の進行方向に沿った縦方向の接続を表す。車線は0個以上の先行車線と後続車線を持つことができる。1対1の接続は通常の連続する車線を，1対多の接続は分岐を，多対1の接続は合流を表す。predecessorLaneIdsとsuccessorLaneIdsは，エリアを経由しない直接の接続のみを表す。エリア（交差点等）を経由する接続は状態laneConnectionsだけで表し，predecessorLaneIdsとsuccessorLaneIdsには含めない。左右の隣接の関係は，車線変更の対象となりうる横方向の隣接を表す。predecessorLaneIds，successorLaneIds，leftAdjacentLaneId，rightAdjacentLaneIdは，状態に含まれない（提供範囲の外にある）車線を参照してもよい。

speedLimitは，この車線に適用される速度制限を表す。規制要素（RegulatoryElement）による速度制限がこの車線に適用される場合，両者は同じ制限を表さなければならない。

maneuverRestrictionsは，この車線に適用される操作制約の集合であり，次の規則に従って，他の項目・規制要素と整合しなければならない。

- NoLaneChangeは，左右の両側とも車線変更が禁止されている場合に用いる。maneuverRestrictionsがNoLaneChangeを含み，laneChangeが値を持つ場合，laneChangeのleftとrightはともにProhibitedでなければならない。
- この車線を適用対象とする規制種別LaneChangeProhibitedの規制要素が状態に含まれる場合，laneChangeが値を持つならば，leftとrightの少なくとも一方がProhibitedでなければならない。
- この車線を適用対象とする規制種別NoEntryの規制要素が状態に含まれる場合，maneuverRestrictionsが値を持つならば，NoEntryを含まなければならない。maneuverRestrictionsがNoEntryを含む場合も，同じ進入禁止を表す。

5.24.8.13. 車線幅の変化

```
type LogicalStructureWidthSampleType = struct {
    width : Float64 ([0.0,+INF)),   // 車線幅。単位はm
    distance : Float64  // 車線の始端からの車線に沿った距離。単位はm。0以上
};
```

```
type LogicalStructureWidthProfileType = LogicalStructureWidthSampleType[];  // 車線幅の変化
```

車線幅の変化を，車線に沿った位置ごとの車線幅の並びで表す。要素はdistanceの小さい順に並べる。distanceは，車線の始端（車線の進行方向における始点）からの，車線の中心線に沿った距離である。隣り合う要素の間の車線幅は，線形に補間する。

5.24.8.14. 速度制限

```
type LogicalStructureSpeedUnitType = enum {
    Unknown,    // 不明
    KmPerHour,  // km/h
    MilesPerHour    // mph（マイル毎時）
};
```

```
type LogicalStructureSpeedLimitType = struct {
    value : Float64 ([0.0,+INF)),   // 制限速度の値。単位はunitによる
    unit : LogicalStructureSpeedUnitType,   // 制限速度の単位
    source : LogicalStructureSourceType?    // 情報の由来
};
```

速度制限（最高速度）を，値と単位の組で表す。車両側実装は，単位を換算せず，情報源が表す単位のまま返してよい。

5.24.8.15. 通行制約

```
type LogicalStructureUserType = enum {
    Unknown,    // 不明
    PassengerCar,   // 乗用車
    Truck,  // トラック
    Bus,    // バス
    Motorcycle, // 二輪自動車
    Bicycle,    // 自転車
    Pedestrian, // 歩行者
    EmergencyVehicle    // 緊急車両
};
```

```
type LogicalStructureAccessRuleValueType = enum {
    Unknown,    // 不明
    Permitted,  // 通行可
    Prohibited, // 通行禁止
    Restricted  // 条件付きで通行可
};
```

```
type LogicalStructureAccessRuleType = struct {
```

```
    appliesTo : enumSet(LogicalStructureUserType),  // 制約の対象となる道路利用者の種
別
    rule : LogicalStructureAccessRuleValueType, // 通行の可否
    condition : String?,    // 通行の条件（時間帯等）
    source : LogicalStructureSourceType?    // 情報の由来
};
```

車線の通行制約を，対象となる道路利用者の種別ごとに表す。ruleがRestrictedの場合，conditionに条件を示す。conditionは，人が読むための記述である。

5.24.8.16. 操作制約

```
type LogicalStructureManeuverRestrictionType = enum {
    Unknown,    // 不明
    StraightOnly,   // 直進のみ可
    LeftOnly,   // 左折のみ可
    RightOnly,  // 右折のみ可
    NoLaneChange,   // 車線変更禁止
    NoEntry,    // 進入禁止
    Reversible  // 時間帯等により進行方向が切り替わる可変車線
};
```

5.24.8.17. 車線変更の可否

```
type LogicalStructureLaneChangeValueType = enum {
    Unknown,    // 不明
    Allowed,    // 車線変更可
    Discouraged,    // 車線変更は可能だが避けるべき
    Prohibited  // 車線変更禁止
};
```

```
type LogicalStructureLaneChangeType = struct {
    left : LogicalStructureLaneChangeValueType, // 左側の車線への車線変更の可否
    right : LogicalStructureLaneChangeValueType // 右側の車線への車線変更の可否
};
```

5.24.8.18. 車線境界

```
type LogicalStructureBoundaryKindType = enum {
    Unknown,    // 不明
    PaintedLine,    // 路面に描かれた線
    Curb,   // 縁石
    Wall,   // 壁
    Guardrail,  // ガードレール
    MedianEdge, // 中央帯の縁
    RoadEdge,   // 道路の端
```

```
    Virtual // 物理的な表示のない仮想の境界
};
```

```
type LogicalStructureMarkingType = enum {
    Unknown,    // 不明
    Solid,  // 実線
    Dashed, // 破線
    DoubleSolid,    // 二重の実線
    SolidDashed,    // 実線と破線の組（左が実線，右が破線）
    DashedSolid,    // 破線と実線の組（左が破線，右が実線）
    BottsDots,  // 道路鋲の列
    None    // 線なし
};
```

組になった線種（SolidDashed，DashedSolid）の左右は，車線境界の幾何（geometry）の点列の向きに向かって定める。

```
type LogicalStructureColorType = enum {
    Unknown,    // 不明
    White,  // 白
    Yellow, // 黄
    Blue,   // 青
    Red // 赤
};
```

```
type LogicalStructureCrossabilityType = enum {
    Unknown,    // 不明
    Crossable,  // 横断可能
    NonCrossable,   // 横断不可
    ConditionallyCrossable  // 条件付きで横断可能
};
```

```
type LogicalStructureSemanticRoleType = enum {
    Unknown,    // 不明
    LaneDivider,    // 同方向の車線の区切り
    CenterDivider,  // 対向する方向の区切り（中央線）
    EdgeDivider,    // 車道の外側の区切り
    ShoulderDivider,    // 車線と路肩の区切り
    RestrictedAreaBorder    // 進入が制限された区域の境界
};
```

```
type LogicalStructureLaneBoundaryType = struct {
    id : LogicalStructureElementIdType, // 車線境界の要素ID
    geometry : LogicalStructurePolyline3DType,  // 境界の幾何
    boundaryType : LogicalStructureBoundaryKindType,    // 境界種別
```

```
    marking : LogicalStructureMarkingType?, // 線種
    color : LogicalStructureColorType?, // 線の色
    physicalCrossability : LogicalStructureCrossabilityType?,   // 物理的な横断の可否
    semanticRole : LogicalStructureSemanticRoleType?,   // 意味上の役割
    height : Float64?,  // 路面からの境界の高さ。単位はm
    width : Float64?,   // 境界の幅。単位はm
    confidence : Float64 ([0.0,1.0])?,  // 信頼度
    source : LogicalStructureSourceType?    // 情報の由来
};
```

車線と車線の間，または車線と道路の端の間の境界を表す。1つの車線境界を，隣り合う2つの車線が共有してよい。physicalCrossabilityは，境界を物理的に横断できるかどうかを表し，交通規則上の可否は表さない。

5.24.8.19. エリア

```
type LogicalStructureAreaKindType = enum {
    Unknown,    // 不明
    Intersection,   // 交差点
    Merge,  // 合流区間
    Diverge,    // 分岐区間
    Roundabout, // 環状交差点
    PedestrianCrossing  // 横断歩道
};
```

```
type LogicalStructureAreaType = struct {
    id : LogicalStructureElementIdType, // エリアの要素ID
    sectionId : LogicalStructureElementIdType,  // 所属する道路区間の要素ID
    type : LogicalStructureAreaKindType,    // エリア種別
    geometry : LogicalStructurePolygonType, // 領域の外形
    innerBoundaries : LogicalStructureElementIdType[]?, // 内部の境界となる車線境界の
要素ID
    outerBoundaries : LogicalStructureElementIdType[],  // 外周の境界となる車線境界の
要素ID
    confidence : Float64 ([0.0,1.0])?,  // 信頼度
    source : LogicalStructureSourceType?    // 情報の由来
};
```

交差点・合流区間など，車線では表しにくい領域を表す。sectionIdは，このエリアを含む道路区間の要素IDと一致する。innerBoundariesとouterBoundariesは，このエリアを含む道路区間のboundariesに含まれる車線境界を参照する。エリアを経由する車線間の接続は，状態laneConnectionsだけで表す。

5.24.8.20. 規制要素

```
type LogicalStructureRegulatoryKindType = enum {
    Unknown,    // 不明
    SpeedLimit, // 速度制限
```

```
    NoEntry,    // 進入禁止
    Stop,   // 一時停止
    Yield,  // 他の交通の進行を妨げてはならない（譲れ）
    NoParking,  // 駐車禁止
    LaneChangeProhibited    // 車線変更禁止
};
```

```
type LogicalStructureRegulatoryParameterType = struct {
    speedLimit : LogicalStructureSpeedLimitType?,   // 制限速度
    stopLine : LogicalStructurePolyline3DType?, // 停止位置を表す線
    userTypes : enumSet(LogicalStructureUserType)?, // 規制の対象となる道路利用者の種
別
    condition : String? // 規制が適用される条件（時間帯等）
};
```

規制の内容を補う値を表す。speedLimitは規制種別がSpeedLimitの場合に，stopLineは規制種別がStopまたはYieldの場合に値を持ちうる。userTypesが値なしの場合，規制はすべての道路利用者に適用される。conditionは，人が読むための記述である。

```
type LogicalStructureRegulatoryElementType = struct {
    id : LogicalStructureElementIdType, // 規制要素の要素ID
    sectionId : LogicalStructureElementIdType,  // 所属する道路区間の要素ID
    type : LogicalStructureRegulatoryKindType,  // 規制種別
    appliesTo : LogicalStructureElementIdType[],    // 規制が適用される車線またはエリ
アの要素ID
    parameters : LogicalStructureRegulatoryParameterType?,  // 規制の内容を補う値
    confidence : Float64 ([0.0,1.0])?,  // 信頼度
    source : LogicalStructureSourceType?    // 情報の由来
};
```

速度制限・進入禁止・一時停止などの交通規制を表す。sectionIdは，この規制要素を含む道路区間の要素IDと一致する。appliesToは，規制が適用される車線またはエリアを参照し，参照先は同じ状態に含まれていなければならない。1つの規制要素を，複数の車線・エリアに適用してよい。

5.24.8.21. 道路端・中央帯・路肩

```
type LogicalStructureEdgeKindType = enum {
    Unknown,    // 不明
    PavementEdge,   // 舗装の端
    CurbEdge,   // 縁石による端
    BarrierEdge,    // 防護柵・壁等による端
    SlopeEdge   // 法面による端
};
```

```
type LogicalStructureRoadEdgeType = struct {
    id : LogicalStructureElementIdType, // 道路端の要素ID
```

```
    geometry : LogicalStructurePolyline3DType,  // 幾何形状
    edgeType : LogicalStructureEdgeKindType,    // 端部種別
    confidence : Float64 ([0.0,1.0])?,  // 信頼度
    source : LogicalStructureSourceType?    // 情報の由来
};
```

道路の有効な外縁，または路面の端を表す。

```
type LogicalStructureMedianKindType = enum {
    Unknown,    // 不明
    Painted,    // 路面標示による中央帯
    Raised, // 縁石等で路面より高くした中央帯
    BarrierSeparated,   // 防護柵等で分離された中央帯
    Vegetated   // 植栽のある中央帯
};
```

```
type LogicalStructureMedianType = struct {
    id : LogicalStructureElementIdType, // 中央帯の要素ID
    geometry : LogicalStructurePolyline3DType,  // 幾何形状
    medianType : LogicalStructureMedianKindType,    // 中央帯種別
    isCrossable : Bool?,    // 横断できるか
    confidence : Float64 ([0.0,1.0])?,  // 信頼度
    source : LogicalStructureSourceType?    // 情報の由来
};
```

中央帯または中央分離帯を表す。

```
type LogicalStructureSideType = enum {
    Unknown,    // 不明
    Left,   // 左
    Right   // 右
};
```

```
type LogicalStructureShoulderKindType = enum {
    Unknown,    // 不明
    Shoulder,   // 路肩
    EmergencyLane,  // 緊急時に用いる帯
    StoppingLane    // 停車帯
};
```

```
type LogicalStructureShoulderType = struct {
    id : LogicalStructureElementIdType, // 路肩の要素ID
    geometry : LogicalStructurePolyline3DType,  // 幾何形状
    side : LogicalStructureSideType,    // 道路の左右のいずれにあるか
    type : LogicalStructureShoulderKindType,    // 路肩種別
```

```
    isDrivable : Bool?, // 緊急時等を含めて走行可能か
    confidence : Float64 ([0.0,1.0])?,  // 信頼度
    source : LogicalStructureSourceType?    // 情報の由来
};
```

路肩・停車帯・緊急時に用いる帯などを表す。sideの左右は，道路の基準線の向きに向かって定める。

5.24.8.22. 接続関係（トポロジ層）

```
type LogicalStructureConnectionType = struct {
    fromId : LogicalStructureElementIdType, // 接続元の要素ID
    toId : LogicalStructureElementIdType,   // 接続先の要素ID
    viaAreaId : LogicalStructureElementIdType?  // 経由するエリアの要素ID
};
```

要素間の向きをもつ接続を表す。車両が接続元から接続先へ進むことができることを示す。viaAreaIdが値を持つ場合，その接続は当該のエリアを経由する。

状態laneConnectionsでは，接続元・接続先は車線を参照し，viaAreaIdは必ず値を持つ。エリアを経由する車線間の接続はlaneConnectionsだけで表し，車線のpredecessorLaneIdsとsuccessorLaneIdsには含めない。経路構造のconnectionsでは，接続元・接続先は経路要素を参照する。

5.24.8.23. 経路構造

```
type LogicalStructurePathType = struct {
    id : LogicalStructureElementIdType, // 経路要素の要素ID
    geometry : LogicalStructurePolyline3DType,  // 目安となる走行経路の形状
    elementIds : LogicalStructureElementIdType[]?   // この経路要素が通る車線・エリアの要素ID（通過順）
};
```

目安となる経路を構成する経路要素を表す。geometryは，車両が走行する目安となる経路の形状であり，点列の向きが進行方向を表す。

```
type LogicalStructurePathStructureType = struct {
    id : LogicalStructureElementIdType, // 経路構造の要素ID
    path : LogicalStructurePathType[],  // 経路要素の配列
    connections : LogicalStructureConnectionType[], // 経路要素間の接続関係
    confidence : Float64 ([0.0,1.0])?,  // 信頼度
    source : LogicalStructureSourceType?    // 情報の由来
};
```

目安となる経路と，その接続関係を表す。connectionsの接続元・接続先は，pathに含まれる経路要素を参照する。

5.24.8.24. タイムスタンプ

```
type TimestampUsType = UInt64;  // 車内共通時刻基盤の起点を基準とする測定時刻。単位はμs。複数のビークルAPIで共通に用いる共通データ型であり，本仕様内に定義する。
```

5.24.8.25. サンプル品質の種別

```
type DataQualifierType = enum {
    Normal, // 正常
    Unavailable,    // 利用不可
    Degraded,   // 劣化
    TestMode,   // テストモード
    Invalid // 無効（値も誤差も読んではならない）
};
```

複数のビークルAPIで共通に用いる共通データ型であり，本仕様内に定義する。

5.24.8.26. LogicalStructure状態

```
type LogicalStructureStatusType = struct {
    timestamp : TimestampUsType,
    dataQualifier : DataQualifierType,
    revision : UInt32,
    drivableAreas : LogicalStructureDrivableAreaType[],
    roads : LogicalStructureRoadType[],
    laneConnections : LogicalStructureConnectionType[],
    pathStructure : LogicalStructurePathStructureType?
};
```

状態に含まれるすべての幾何は，構成情報coordinateSystemが宣言する単一の座標系で表す。状態には，提供範囲と一部でも重なる要素が，情報源から得られる限りすべて含まれる。dataQualifierがInvalidまたはUnavailableの場合，要素を読んではならない。

5.24.8.27. LogicalStructureイベント

```
type LogicalStructureChangeReasonType = enum {
    CoverageShifted,    // 提供範囲の移動に伴う要素の追加・削除
    ContentUpdated, // 提供範囲内の要素の追加・削除，または幾何形状と信頼度を除く項目の変化
    DataQualifierChanged    // dataQualifierの変化
};
```

```
type LogicalStructureEventKindType = enum {
    FaultDetected,
    FaultRecovered,
```

```
    StructureChanged,
    EventOverflow,
    ConfigChanged
};
```

```
type LogicalStructureEventType = struct {
    eventInfo : variant(LogicalStructureEventKindType) {
        StructureChanged {
            reasons : enumSet(LogicalStructureChangeReasonType) ,
            revision : UInt32
        },
        EventOverflow {
            noLostEvents : UInt32
        }
    }
};
```

5.24.9. サービスコールの詳細規定

getConfig

LogicalStructureの構成情報の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
config	LogicalStructureConfigType	LogicalStructureの構成情報

【エラーコード】

なし

【機能】

LogicalStructureに関する構成情報を返す。

getStatus

道路論理構造の参照

【パラメータ】

なし

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
status	LogicalStructureStatusType	自車周辺の提供範囲の道路論理構造

【エラーコード】

E_RISK_APPLICATION	アプリケーションが対応していないリスクがある
E_RISK_USER	ユーザがリスクを承認していない
E_OBJECT_FAULT	故障中（道路構造を提供できない状態）

【機能】

自車周辺の提供範囲の道路論理構造に関するすべての状態を返す。返される道路構造は，情報源に基づく推定値を含み，精度は保証されない。取得した値の信頼度は，併せて返されるdataQualifierと，要素ごとのsourceおよびconfidenceによって判断する。dataQualifierがInvalidまたはUnavailableの場合，要素を読んではならない。

座標系が車両座標系の場合，返される座標は，状態のtimestampの時点の車両の位置と向きを基準とする。

自車周辺の道路構造は自車の現在位置を推定できる情報であるため，本サービスコールはリスククラスRiskPrivacyの対象であり，RiskPrivacyに対応していないアプリケーションはE_RISK_APPLICATIONエラー，ユーザが状態の開示を承認していない場合はE_RISK_USERエラーとなる。

notify

LogicalStructureイベントの通知要求

【パラメータ】

eventFilter	enumSet (LogicalStructureEventKindType)?	通知するイベントの種類の集合

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【エラーコード】

E_RISK_APPLICATION	アプリケーションが対応していないリスクがある
E_RISK_USER	ユーザがリスクを承認していない
E_INVALID_PARAMETER	eventFilterが有効範囲外

【機能】

LogicalStructureで発生したeventFilterで指定したイベントの通知を要求する。イベントを通知するためのイベントキューを生成し，その通知ハンドルをnotifyHandleに返す。

eventFilterを省略した場合，すべての種類のイベントの通知を要求する。

通知されるイベントは状態の内容を運ばないが，StructureChangedイベントの発生の時刻と理由は自車の走行と相関し，また，状態を取得し直す契機となる。このため，本サービスコールの呼び出しそのものが，eventFilterの内容によらず（eventFilterを省略した場合を含む），リスククラスRiskPrivacyの対象である。RiskPrivacyに対応していないアプリケーションはE_RISK_APPLICATIONエラー，ユーザが状態の開示を承認していない場合はE_RISK_USERエラーとなる。

unnotify

LogicalStructureイベントの通知停止

【パラメータ】

notifyHandle	NotifyHandleType	操作対象の通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正

【機能】

notifyHandleで指定した通知を停止する。指定した通知用のイベントキューは削除される。

getEvent

LogicalStructureイベントの取得

【パラメータ】

notifyHandle	NotifyHandleType	イベントキューを識別するための通知ハンドル

【リターンパラメータ】

returnValue	ReturnType	正常終了またはエラーコード
event	LogicalStructureEventType	取得したイベント

【エラーコード】

E_INVALID_HANDLE	notifyHandleが不正
E_NO_DATA	イベントキューが空

【機能】

notifyHandleで指定した通知用のイベントキューから，先頭のイベントを取り出す。取り出したイベントは，イベントキューから削除される。



イベントがパラメータ等で渡されてくるプログラミング環境向けの物理APIでは，用意する必要がない。

Appendix A: 車両状態の定義

＜TBD＞

Appendix B: C言語向け物理API（C言語バインディング）

B.1. データ型

ここでは，Section 2.1.2で規定した論理APIにおけるデータ型を，どのようにC言語のデータ型で実現するかを規定する。なお，物理APIにおけるデータ型は，状況毎に最適なものとし，論理APIから機械的に決まるとは限らないものとする。

(1) 基本型

素直に対応するデータ型で実現する。

特殊値の定数名は，先頭に名前の衝突を避けるための文字列を付加することを原則とし，#defineを用いて対応する数値に定義する。

(2) 文字列型

`char *`で実現する。

(3) 固定長配列

配列で実現する。

(4) 可変長配列

配列を置いた領域へのポインタと要素数をフィールドに持つ構造体で実現することを原則とする。

(5) 列挙型

C言語の列挙型で実現する。

列挙子を表す定数名は，列挙子の名前の前に，列挙型を区別するための文字列を付加した名称とすることを原則とする。列挙子には，原則として，1から連続する正の数値を明示的に割り付ける。

ただし，ReturnTypeはint32_t型で実現し，E_OKに0，その他の列挙子には負の数値を割り付ける。ReturnTypeの列挙子を表す定数名は，列挙子の名前とし，#defineを用いて割り付けられた数値に定義する。

(6) 列挙型の集合型

各列挙子が1つのビットに対応する個別に定めるサイズの符号なし整数型（ビットマップ）で実現する。

空集合を表すマクロ（emptyEnumSet）と，列挙子からそれに対応するビットを求めるためのマクロ（enumSetBit）を用意する。これらのマクロの定義は次の通り。

```
#define emptyEnumSet            0U
#define enumSetBit(Enumerator)  (1U << ((Enumerator) - 1))
```

(7) 構造体

構造体で実現する。

(8) バリアント（タグ付き共用体）

パラメータの列挙型と，各ペイロードのデータ型（原則として構造体）の共用体をフィールドに持つ構造体で実現することを原則とする。

(9) オプショナル型

基礎となるデータ型に無効を示す値がある場合には，その値でnullを表すことを原則とする。

そうでない場合には，基礎となるデータ型へのポインタで実現し，NULLポインタでnullを表すことを原則とする。

B.2. 共通データ型

(1) 戻り値のデータ型

```
typedef int32_t     ReturnType;

#define E_OK                    0       /* 正常終了 */
#define E_INVALID_ID            -1      /* IDが不正 */
#define E_INVALID_HANDLE        -2      /* ハンドルが不正 */
#define E_INVALID_PARAMETER     -3      /* その他のパラメータが不正 */
#define E_RISK_APPLICATION      -4      /* アプリケーションが対応していないリ
                                           スクがある */
#define E_RISK_USER             -5      /* ユーザがリスク対応責任の移転を承認
                                           していない */
#define E_DENIED_PRIORITY       -6      /* 使用できない操作優先度 */
#define E_DENIED_ACCESS         -7      /* アクセスが認められていない */
#define E_OBJECT_FAULT          -8      /* 対象オブジェクトが故障中 */
#define E_OBJECT_LOCKED         -9      /* 対象オブジェクトがロックされている */
#define E_OBJECT_STATUS         -10     /* 対象オブジェクトが指定された操作を
                                           できない状態にある */
#define E_INFEASIBLE            -11     /* 指定された操作が現在の状態では実現
                                           できない */
#define E_NO_DATA               -12     /* データがない */
#define E_NO_MEMORY             -13     /* メモリ不足 */
#define E_NOT_SUPPORTED         -14     /* サポートされていない機能 */
#define E_COMMUNICATION         -15     /* 通信エラー */
#define E_NOT_OK                -16     /* その他のエラー */
```

(2) インスタンスIDのデータ型

```
typedef uint16_t    IdType;

#define ID_NULL     0           /* オプショナル型でnullを示す */
```

(3) 操作優先度のデータ型

```
typedef uint8_t     PriorityType;

#define PRIORITY_NULL   0       /* オプショナル型でnullを示す */
```

(4) アプリケーションIDのデータ型

アプリケーションIDのデータ型は，OS/プラットフォーム依存に規定する。

(5) オブジェクトを表すデータ型

```
typedef enum {
    ObjectKindVehicle = 1,
    ObjectKindBody = 2,
    ObjectKindCabin = 3,
    ObjectKindDoor = 4,
    ObjectKindWindow = 5
} ObjectKindType;

typedef struct {
    ObjectKindType  objectKind;
    IdType          instanceId;
} RelatedObjectType;
```

シングルトンの場合は，instanceIdをID_NULLとする。

(6) オブジェクトの場所を表すデータ型

```
typedef enum {
    ZoneLeft = 1,           /* 左 */
    ZoneRight = 2,          /* 右 */
    ZoneCenter = 3,         /* 中央 */
    ZoneTop = 4,            /* 上 */
    ZoneBottom = 5          /* 下 */
} ZoneType;

typedef struct {
    uint8_t     row;        /* 座席の何列目か */
    ZoneType    zone;       /* 左右上下方向の位置 */
} PlacementType;
```

(7) タイムスタンプのデータ型

```
typedef int64_t     TimestampNsType;
```

(8) ロック状態のデータ型

```
typedef struct {
    bool                locked;
    ApplicationIdType   lockApplication;
    PriorityType        lockPriority;
} LockStatusType;
```

lockApplicationとlockPriorityは，lockedがtrueの場合のみ有効。

(9) 位置を表すデータ型

```
typedef int8_t PositionType;

#define POSITION_UNKNOWN    -1
#define POSITION_NONZERO    -2
```

B.3. コアビークルAPIのデータ型

B.3.1. WindowのAPIのデータ型

(1) Windowの機能クラスのデータ型

```
typedef enum {
    WindowMovable = 1,              /* APIにより移動できる */
    WindowClosedDetectable = 2,     /* 全閉状態を検知できる */
    WindowHasMove2 = 3,             /* 第二次元方向に移動できる */
    WindowMoveProfileSupported = 4  /* 移動プロファイルをサポートしている */
} WindowCapabilityType;

typedef uint32_t WindowCapabilitySetType;
```

(2) Windowに対するリスククラスのデータ型

```
typedef enum {
    WindowRiskPinch = 1,        /* 挟み込みのリスク */
    WindowRiskOpen = 2          /* 開動作に伴う諸々のリスク */
} WindowRiskType;

typedef uint32_t WindowRiskSetType;
```

(3) Windowの構成情報のデータ型

```
typedef struct {
```

```
    char                    *instanceName;
    IdType                  instanceId;
    PlacementType           placement;
    WindowCapabilitySetType capabilities;
    WindowRiskSetType       riskClasses;
    RelatedObjectType       mountedOn;
    char                    *displayName;
} WindowConfigType;
```

(4) Window状態のデータ型

```
typedef enum {
    WindowStopped = 1,          /* 停止中 */
    WindowOpening = 2,          /* 開動作中 */
    WindowClosing = 3,          /* 閉動作中 */
    WindowFault = 4,            /* 故障中 */
    WindowPinchAvoiding = 5     /* 挟み込み回避中 */
} WindowMainStatusType;

typedef struct {
    WindowMainStatusType    mainStatus;
    WindowMainStatusType    mainStatus2;
    LockStatusType          lockStatus;
    PositionType            position;
    PositionType            targetPosition;
    PositionType            position2;
    PositionType            targetPosition2;
} WindowStatusType;
```

mainStatus2，position2，targetPosition2は，Windowが第二次元方向に移動　する機能を持つ場合のみ有効。

(5) Windowの移動プロファイルのデータ型

```
typedef enum {
    WindowMoveProfileStandard = 0,  /* 標準的な移動 */
    WindowMoveProfileFast = 1,      /* 高速で移動 */
    WindowMoveProfileSlow = 2       /* 低速で移動 */
} WindowMoveProfileType;
```

(6) Windowイベントのデータ型

```
typedef enum {
    WindowControlledBySelf = 1,     /* 自アプリケーションによる制御 */
    WindowControlledByOther = 2,    /* 他のアプリケーションによる制御 */
    WindowTargetReached = 3,        /* 目標位置到達，移動停止完了 */
    WindowFaultDetected = 4,        /* 故障検知 */
```

```
    WindowFaultRecovered = 5,       /* 故障回復 */
    WindowPinchDetected = 6,        /* 挟み込み検知 */
    WindowPinchRecovered = 7,       /* 挟み込み回復 */
    WindowOwnLockRevoked = 25,      /* 自アプリケーションのロックの強制解除 */
    WindowOtherLockReleased = 26,   /* 他のアプリケーション等によるロック解除 */
    WindowRiskApplicationRejected = 29, /* アプリケーションがリスクに未対応 */
    WindowRiskUserRejected = 30,    /* ユーザがリスク対応責任の移転を未承認 */
    WindowEventOverflow = 31,       /* イベントキューのオーバフロー */
    WindowConfigChanged = 32        /* 構成情報の変更 */
} WindowEventKindType;

typedef uint32_t WindowEventKindSetType;

typedef struct {
    IdType              instanceId;
    WindowEventKindType eventKind;
    union {
        struct {
            ApplicationIdType   sourceApplication;
        } controlledByOther;
        struct {
            ApplicationIdType   sourceApplication;
        } targetReached;
        struct {
            uint32_t    noLostEvents;
        } eventOverflow;
    } eventInfo;
} WindowEventType;
```

sourceApplicationは，eventKindがControlledByOtherかTargetReachedの場合， noLostEventsは，eventKindがEventOverflowの場合のみ有効。