Unmanned Maritime Autonomy Architecture (UMAA) Software Development Kit (SDK) Systems Model

Component Specification

For

UMAA SDK Mission Executor

Component Model Element Server Id: 777e6c4c-c6f7-4425-9f37-6ba56680d951

Published on: October 7, 2024

by: Ryan D. Jescovitch, ARL PSU

Source Data

From Unmanned Maritime Systems Autonomy Model

Release branch version commit: 213

CONTENTS

1 Scope

1.1 Identification

This document describes the Component Specification for the UMAA SDK Mission Executor Component.

1.2 Unmanned Autonomy Architecture (UMAA) Component Overview

For an overview of UMAA, see UMAA-INF-ADD "Unmanned Maritime Autonomy Architecture (UMAA) Architecture Design Description (ADD)."

A Component is defined as a set of one or more Service Providers and/or Consumers along with any non-UMAA defined Interfaces that collectively meet the needs to perform a specific function or capability. Service Providers describe which UMAA Services must be implemented in accordance with the UMAA ICDs as part of the Component. Service Consumers describe software on the Component that utilizes a Service Provider either to obtain data or to effect change within a system utilizing interfaces described in the UMAA ICDs. In order to support an acquisition strategy that focuses on rapid development and integration of scoped software components, the Architecture Framework Working Group (AFWG), under the direction of the Chief Unmanned Autonomy Framework Architect (CUAFA), has established a reference architecture that defines several components for implementing functions within an autonomy system with standardized interfaces. The model-based reference architecture is contained within a SysML model hosted by Naval IME and maintained by PEO USC, PMS406.

The reference architecture describes Components in three abstraction layers: Component Definition, Component Specification, and Component Design. All three abstraction layers are represented in SysML, with the Component Definitions managed by the UMAA Board and the Component Specification and Design managed as part of the Autonomy Baseline under the CUAFA. Figure 1 summarizes the content described in each abstraction layer.

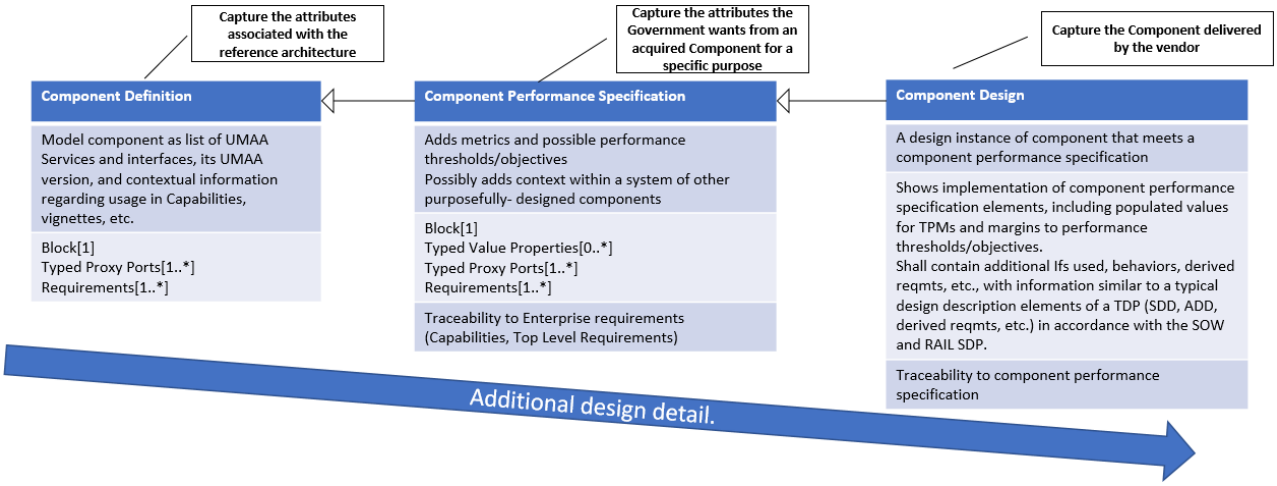


Figure 1. Model-based Components

The Component Definition is the Component's highest level of abstraction represented in the reference architecture. It provides context for the need of the Component, identifies the Service Providers and Service Consumers and their interfaces, requirements to conform with the reference architecture, and metrics that may be used in the Component Specification to monitor performance during the acquisition process.

The Component Specification is a representation of the Component that captures the Government's desired attributes for the Component in a specific acquisition. In addition to the attributes from the Component Definition, the Component Specification identifies any metrics the performer must report on during the contract and any additional requirements (threshold/objective) derived for the specific acquisition. Derived requirements show traceability to parent requirements contained in platform specifications or architecture elements in the PMS 406 UxV Portfolio Model.

The Component Design represents a Component that is implemented by a developer and delivered to the Government as part of the Autonomy Baseline. The Component Design shows implementation of the Component Specification, including traceability of design elements to the requirements, metrics, and Service Participant elements they satisfy. Component Designs provide similar information to Software Description documents and Algorithm Description Documents and may be utilized as evidence when assessing UMAA compliance. The "PMS 406 Autonomy Software Development Process" contains additional guidance in the development of Component Designs.

The three abstraction layers are purposely defined to support various phases of the acquisition process and for controlling and evaluating Components for re-use. The figure below illustrates one example for how each is used.

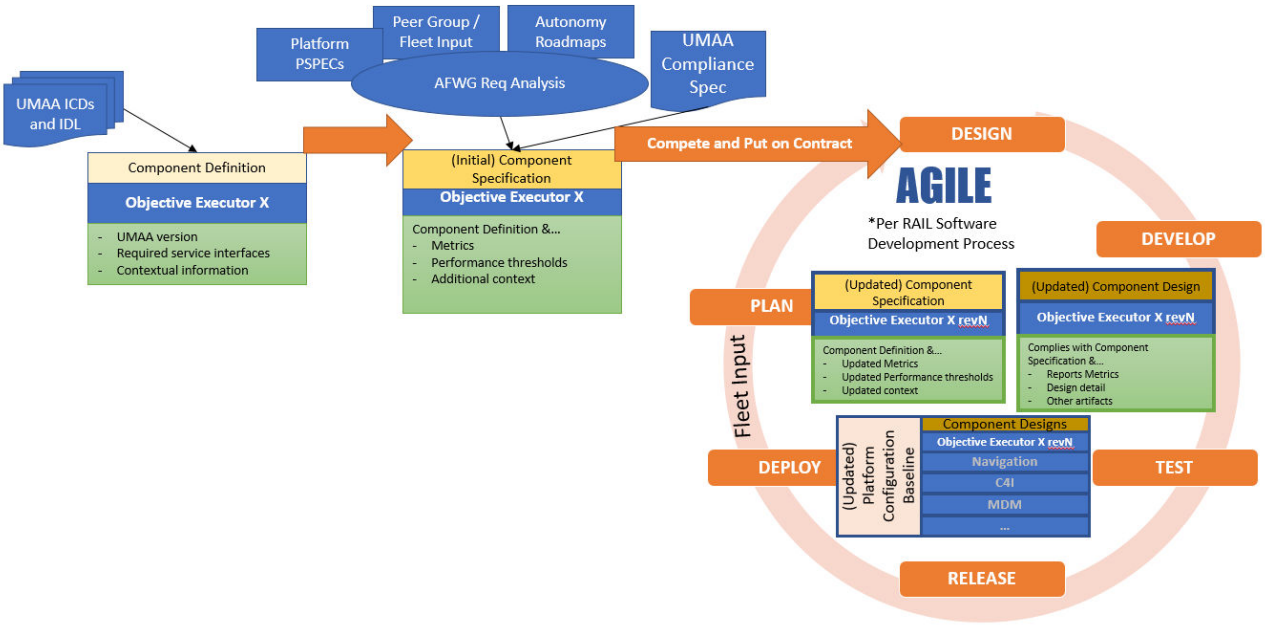


Figure 2. Example Use of Model-based Component Definition, Specification, and Design

1.2.1 Component Reference Architecture Overview

A set of UMAA Components form the basis of a reference architecture. Components satisfying the needs for new capabilities must be integrated into this reference architecture. Due to sometimes conflicting needs, there are two "notional" platform configurations that the AFWG uses to describe the reference architecture: one for unmanned surface vehicles and another for unmanned underwater vehicles. The figure below illustrates one aspect of the reference architecture and its componentization. Blocks in the diagram represent released Component Definitions. Clouds in the diagram represent areas of capability the AFWG is working to componentize at this time.

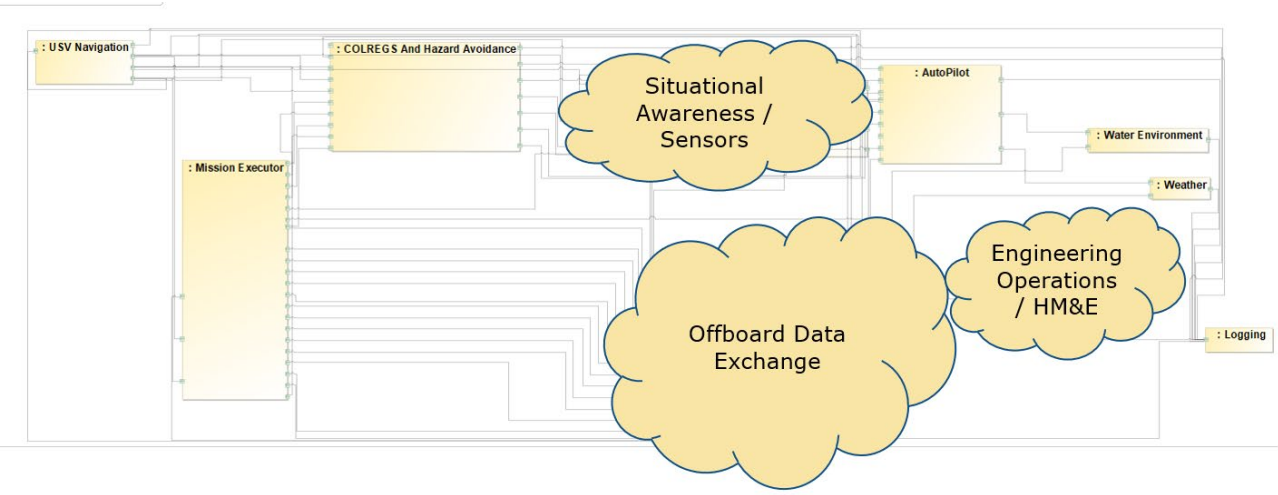


Figure 3. Component Reference Architecture

1.3 Document Overview

This document represents an export of the model-based Component Definition for this Component.

This document is organized in the following sections:

- **Scope** – Provides a brief overview of UMAA, definition of autonomy components, and the organization of this document.
- **References** – Lists the documents or other artifacts referenced in this document.
- **Component Description** – Provides context for this specific component and introduces its interfaces.
- **Component Requirements** – Identifies requirements associated with the Component Definition.
- **Component Metrics** – Identifies metrics that may be associated with the component in its Component Specification.
- **Component Interfaces** – Provides additional detail on the component's interfaces.

This document is auto-generated from a model and should not be altered in its document-based form. Source data for this published document is contained in the Unmanned Maritime Systems Autonomy Model hosted on Naval IME's Teamwork cloud server. Details on model version and component identification within the model are contained on the cover page of this document. For access to the model-based component specification contact NAVSEA, PEO USC, ATTN: PMS 406 via the contact information on the cover page of this document.

2 References

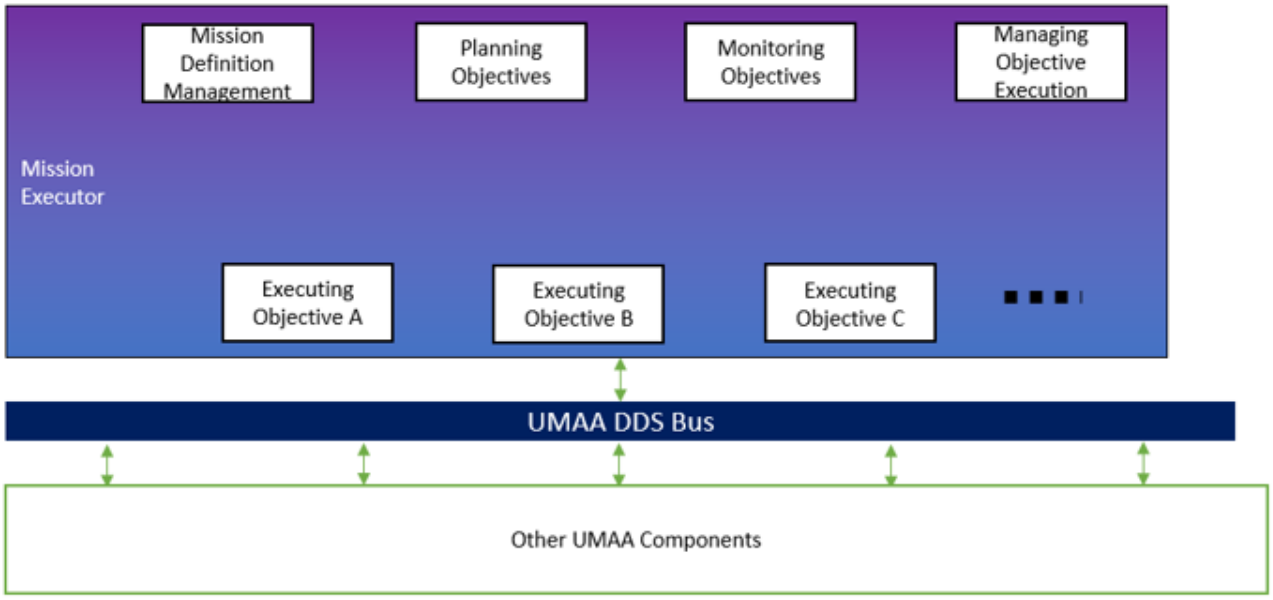
Number	Version	Title
UMAA-INF-ADD	1.1a	Unmanned Maritime Autonomy Architecture (UMAA) Architecture Design Description (ADD)
T0300-BE-IDS-010	1	Unmanned Maritime Autonomy Architecture (UMAA) Compliance Specification
UMAA-SPEC-EOICD UMAA-SPEC-MOICD UMAA-SPEC-MMICD UMAA-SPEC-SEMICD UMAA-SPEC-SAICD	6.0	UMAA Interface Control Documents (ICD)

Number	Version	Title
UMAA-SPEC-SOICD		
N/A	Draft (v0.3)	PMS 406 Autonomy Software Development Process

3 Component Description

3.1 Component Summary

The Mission Executor Component performs functions to: maintain the Mission Plan, decompose the Mission Plan into actionable autonomous objectives, manage execution of objectives, execute specific objectives, and monitor objective execution. The figure below illustrates Mission Executor Component and its interaction with other autonomy Components within the architecture.



The Mission Executor maintains the Mission Plan by tracking of the master list of conditionals and active constraints, as well as the MissionPlanReport associated with the overall set of missions loaded on the platform. This set of missions may contain more than just what is running at any given time.

The Mission Executor fully decomposes and allocates a given set of objectives without violating any constraints. The Mission Executor does not decompose missions or tasks.

The Mission Executor executes the given plan by managing one or more objectives. The Mission Executor will evaluate when an objective needs to be started, stopped, or restarted based on the associated StateTriggers and availability of the associated resources. Functions to achieve active objectives are part of this Component, using UMAA service interfaces to control other Components or consume data from other Components as needed. The UMAA SDK Mission Executor is capable of executing the RouteObjective type.

The Mission Executor evaluates the current running plan to determine if it ever becomes invalid. If the plan becomes invalid, the Component is capable of re-planning in an attempt to generate a new valid plan. Additionally, it provides execution status of all objectives being run as part of the Mission Plan.

3.2 Component Interfaces

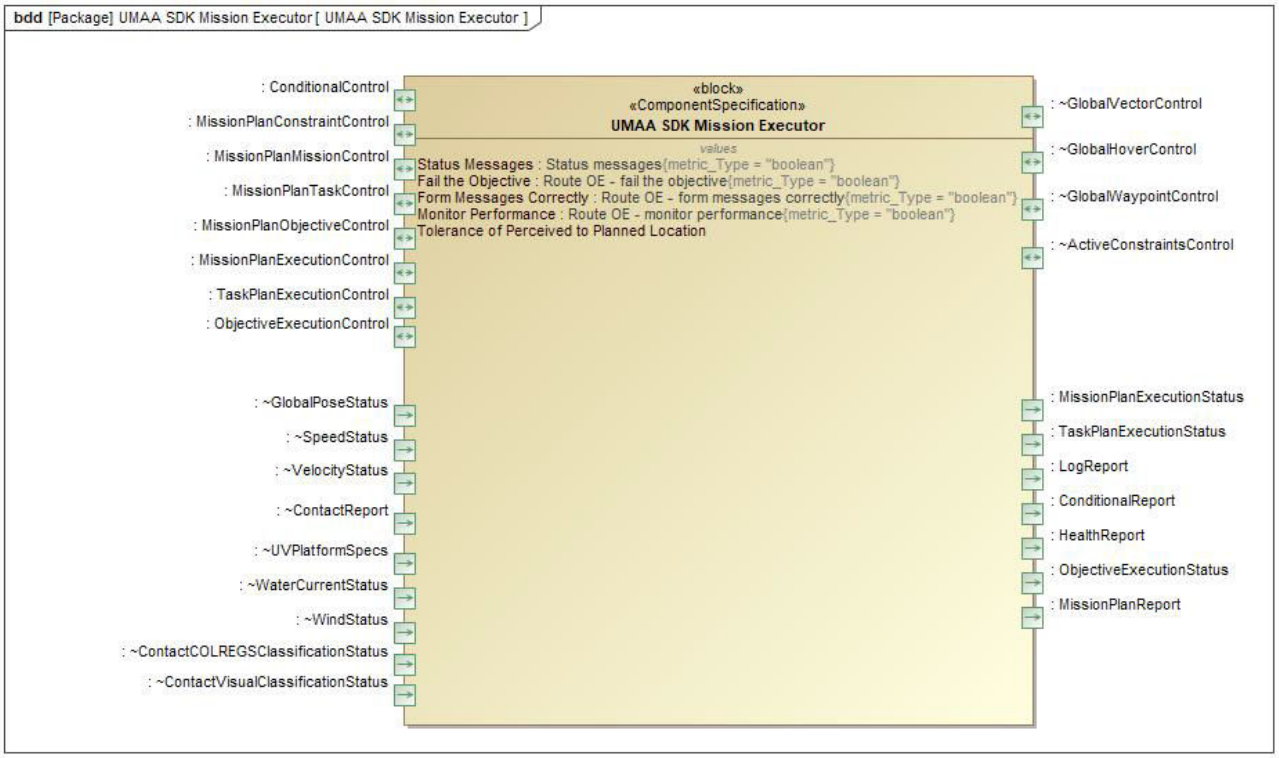


Figure 4. UMAA SDK Mission Executor

The above block definition diagram summarizes the Component Specification. Ports on the Component identify the Service Providers and Service Consumers for the Component. Conjugated ports, annotated with "~" prior to the defined port Type, represent usage of the interface as a consumer. Otherwise, the port represents usage of the interface as a provider of the service. The usage of a Service for the particular component is also identified by its color per the legend. Each port is typed by an Interface Block representing a UMAA Service, with nested proxy ports representing the operations of the Service. Additional details on the interfaces are described in section 6.

Note: Conjugated proxy ports, annotated with "~" prior to the defined Type, represent usage of the interface as a consumer. This results in the flows defined in the typed InterfaceBlock to be reversed when used by the component. Proxy ports that are not conjugated represent usage of the interface as a provider and the flows defined in the Typed InterfaceBlock are not reversed. The Interface Blocks representing Services have nested proxy ports with directivity (in / out) defined from the perspective of a Service Provider. This is because the nested proxy ports' flow properties, representing the Topics, are defined with directivity representing a message being published or subscribed by a Service Provider in relation to the DDS bus. This is why a conjugated port is used to represent a Service Consumer; the directivity of each flow property is reversed. This is consistent with the UMAA ICD definition of Services, with operations similarly categorized as Service Requests (inputs) and Service Responses (outputs).

3.3 Component Behavior

No additional behavior diagrams are included in this specification.

3.4 UMAA SDK Mission Executor Component Requirements [UMAASDK-MM-1]

3.4.1 Functional Requirements [UMAASDK-MM-1.1]

3.4.1.1 Maintain Mission Plan [UMAASDK-MM-1.1.1]

3.4.1.1.1 [UMAASDK-MM-1.1.1.1]

The component shall maintain the Mission Plan by tracking a master list of conditionals and active constraints, as well as the MissionPlanReport associated with the overall set of missions loaded on the platform.

3.4.1.1.2 Initiate Mission Plan [UMAASDK-MM-1.1.1.2]

The Mission Executor shall initiate execution of the mission plan upon command.

3.4.1.2 Monitor Mission Plan [UMAASDK-MM-1.1.2]

3.4.1.2.1 [UMAASDK-MM-1.1.2.1]

The component shall monitor the currently running Mission Plan to determine if it becomes invalid and monitor execution status of all objectives executing in the currently running Mission Plan.

3.4.1.2.2 Failed Objectives [UMAASDK-MM-1.1.2.2]

The Mission Executor shall detect failed objectives. Actions to be taken when an objective fails is implemented as part of the mission plan using conditionals based on objective type and state.

3.4.1.2.3 Objective Completion [UMAASDK-MM-1.1.2.3]

The Mission Executor shall determine achievement of objectives.

3.4.1.3 Decompose Objectives [UMAASDK-MM-1.1.3]

3.4.1.3.1 [UMAASDK-MM-1.1.3.1]

The component shall decompose and allocate a given set of objectives without violating any constraints.

3.4.1.4 Assign Resources [UMAASDK-MM-1.1.4]

3.4.1.4.1 [UMAASDK-MM-1.1.4.1]

The component shall assign resources to execute a given objective, managing when the objective is started, stopped, or restarted based on the associated StateTriggers and availability of associated resources.

3.4.1.4.2 Objective Execution [UMAASDK-MM-1.1.4.2]

The Mission Executor shall execute objectives as defined by the mission plan.

3.4.1.5 Provide Updated Mission Plan [UMAASDK-MM-1.1.5]

3.4.1.5.1 [UMAASDK-MM-1.1.5.1]

The component shall provide updated mission plan information using MissionPlanExecutionStatus, TaskPlanExecutionStatus, ObjectiveExecutionStatus, ConditionalReport, and MissionPlanReport service interfaces.

3.4.1.6 Update Mission Plan Reports [UMAASDK-MM-1.1.6]

3.4.1.6.1 [UMAASDK-MM-1.1.6.1]

The Mission Executor shall update the ConditionalReport and MissionPlanReport in accordance with the Adds and Deletes from the Control services.

3.4.1.7 Provide Vehicle Maneuvering Commands [UMAASDK-MM-1.1.8]

3.4.1.7.1 [UMAASDK-MM-1.1.8.1]

The component shall send desired vehicle maneuvering commands using GlobalVectorControl, GlobalWaypointControl, and GlobalHoverControl service interfaces.

3.4.1.8 Consume Externally Received Mission Plan [UMAASDK-MM-1.1.9]

3.4.1.8.1 [UMAASDK-MM-1.1.9.1]

The component shall consume the externally received mission plan using ConditionalControl, MissionPlanConstraintControl, MissionPlanMissionControl, MissionPlanTaskControl, MissionPlanObjectiveControl, TaskPlanExecutionControl, MissionPlanExecutionControl, and ObjectiveExecutionControl service interfaces.

3.4.1.8.2 Validate Mission Plan [UMAASDK-MM-1.1.9.2]

The Mission Executor shall validate the initial Mission Plan. The component compares objectives in the mission plan to capabilities of the autonomy, and fails any objectives the autonomy cannot support.

3.4.1.9 Consume Own-ship Navigation Information [UMAASDK-MM-1.1.10]

3.4.1.9.1 [UMAASDK-MM-1.1.10.1]

The component shall consume own-ship navigation information using the GlobalPoseStatus, SpeedStatus, and VelocityStatus service interfaces.

3.4.1.10 Consume Contact Report [UMAASDK-MM-1.1.11]

3.4.1.10.1 [UMAASDK-MM-1.1.11.1]

The Mission Executor shall consume contact reports using the ContactReport service interface.

3.4.1.11 Perform Route Objective [UMAASDK-MM-1.1.12]

3.4.1.11.1 Plan Route Objective [UMAASDK-MM-1.1.12.1]

The Mission Executor shall plan a route to the destination specified in the mission objective parameters.

3.4.1.11.1.1 Fail the Objective - Objective Parameters [UMAASDK-MM-1.1.12.1.2]

The Mission Executor shall execute the commanded route objective and fail the objective if the platform goes outside the tolerance values specified in the given route objective.

3.4.1.11.1.2 Safety in Planning [UMAASDK-MM-1.1.12.1.4]

The Mission Executor shall plan a route to the destination specified in the mission objective parameters which satisfies minimum safe depth, minimum safe altitude specified in the mission parameters.

3.4.1.11.1.3 Running Objective With Constraints [UMAASDK-MM-1.1.12.1.5]

The Mission Executor shall trigger objective failure if it cannot plan a route within constraints specified in ActiveConstraintsControl.

3.4.1.11.2 Execute Route Objective [UMAASDK-MM-1.1.12.2]

The Mission Executor shall execute the commanded route objective and fail the objective if the platform goes outside the tolerance values specified in the given route objective.

3.4.1.11.2.1 Waypoint Capture Radius [UMAASDK-MM-1.1.12.2.1]

The Mission Executor shall achieve the waypoint capture radius specified in the mission objective parameters while executing the planned transit.

3.4.1.11.2.2 Maneuvering Commands - GlobalVector Service [UMAASDK-MM-1.1.12.2.2]

The Mission Executor shall be capable of outputting vehicle maneuvering commands to execute the route objective using the GlobalVectorControl service.

3.4.1.11.2.3 Maneuvering Commands - GobalWaypoint Service [UMAASDK-MM-1.1.12.2.3]

The Mission Executor shall be capable of outputting vehicle maneuvering commands to execute the route objective using the GlobalWaypointControl service.

3.4.1.11.2.4 Tolerance of Perceived to Planned Location [UMAASDK-MM-1.1.12.2.4]

The Mission Executor shall maintain a distance from perceived to planned location less than the threshold specified in the mission objective parameters while executing planned transit.

3.4.1.12 Health and Status [UMAASDK-MM-1.1.13]

The Mission Execution Manager shall be capable of reporting UUV health and status while in mission.

3.4.1.12.1 Report Mission Cancel [UMAASDK-MM-1.1.13.1]

During execution of the mission plan, the Mission Executor shall be capable of reporting mission cancel.

3.4.1.12.2 Report Mission Complete [UMAASDK-MM-1.1.13.3]

During execution of the mission plan, the Mission Executor shall be capable of reporting mission complete.

3.4.2 Interface Requirements [UMAASDK-MM-1.2]

The Mission Executor is required to implement all interfaces described in Figure 4 of this specification in accordance with T0300-BE-IDS-010 "Unmanned Maritime Autonomy Architecture (UMAA) Compliance Specification." The subsections below include additional specification of select UMAA interface details.

3.4.2.1 GlobalVectorControl [UMAASDK-MM-1.2.1]

3.4.2.1.1 SpeedRequirementVariantType [UMAASDK-MM-1.2.1.1]

The Mission Executor shall be capable of providing at least one of the following SpeedRequirementVariantType enumerations: GROUNDSPEEDREQUIREMENTVARIANT_D, WATERSPEEDREQUIREMENTVARIANT_D, and ENGINERPMSPEEDREQUIREMENTVARIANT_D in GlobalVectorControl.

3.4.2.1.2 DirectionRequirementVariantType [UMAASDK-MM-1.2.1.2]

The Mission Executor shall provide DirectionRequirementVariantType DIRECTIONTRUENORTHREQUIREMENTVARIANT_D in GlobalVectorControl.

3.4.2.1.3 ElevationRequirementVariantType [UMAASDK-MM-1.2.1.3]

The Mission Executor shall be capable of providing both ElevationRequirementVariantType enumerations ALTITUDEASFREQUIREMENTVARIANT_D and DEPTHREQUIREMENTVARIANT_D in GlobalVectorControl.

3.4.2.1.4 DirectionModeEnumType [UMAASDK-MM-1.2.1.4]

The Mission Executor shall be capable of handling the DirectionModeEnumType enumeration HEADING_D for directionMode.

3.4.2.2 GlobalPoseStatus [UMAASDK-MM-1.2.2]

3.4.2.2.1 depth [UMAASDK-MM-1.2.2.1]

The Mission Executor shall be capable of handling depth in GlobalPoseReport.

3.4.2.2.2 altitudeASF [UMAASDK-MM-1.2.2.2]

The Mission Executor shall be capable of handling altitudeASF in GlobalPoseReport.

3.4.2.2.3 attitude [UMAASDK-MM-1.2.2.3]

The Mission Executor shall be capable of handling attitude in GlobalPoseReport.

3.4.2.3 SpeedStatus [UMAASDK-MM-1.2.3]

3.4.2.3.1 speedOverGround [UMAASDK-MM-1.2.3.1]

The Mission Executor shall be capable of handling speedOverGround in SpeedStatus.

3.4.2.3.2 speedThroughWater [UMAASDK-MM-1.2.3.2]

The Mission Executor shall be capable of handling speedThroughWater in SpeedStatus.

3.4.3 Performance Requirements [UMAASDK-MM-1.3]

3.4.3.1 Autonomous Mission Management [UMAASDK-MM-1.3.3]

The Mission Executor shall be capable of executing objectives defined in the mission objective list without the aid of RF communications (i.e., operator intervention).

3.4.3.2 Conditional Statements [UMAASDK-MM-1.3.4]

The Mission Executor shall be capable of accepting the following conditional statements that limit its operation: DepthConditionalType, DepthRateConditionalType, HeadingSectorConditionalType, LogicalANDConditionalType, LogicalNOTConditionalType, LogicalORConditionalType, TimeConditionalType, PitchRateConditionalType, RelativeSpeedConditionalType,

RollRateConditionalType, SpeedConditionalType, TimeConditionalType, WaterZoneConditionalType, and YawRateConditionalType.

3.4.4 Design Constraints [UMAASDK-MM-1.4]

3.4.4.1 UMAA 6.0 [UMAASDK-MM-1.4.1]

The Mission Executor shall meet requirements of UMAA release 6.0.

3.4.4.2 Configurable QoS [UMAASDK-MM-1.4.2]

Quality of Service (QoS) for the Mission Executor (domain participants), service interfaces (DDS endpoints), and Topics shall be configurable without changes to or recompiling of the implementation source code.

3.4.4.3 Configurable Domain ID and Partitions [UMAASDK-MM-1.4.3]

Domain ID and partitions for the Mission Executor and its services shall be configurable without changes to or recompiling of the implementation source code.

3.4.4.4 Configurable Source and Destination GUIDs [UMAASDK-MM-1.4.4]

Source and destination IdentifierTypes shall be configurable using without changes to or recompiling of the implementation source code.

4 Component performance metrics

The following are metrics that must be measured for this component or the system that contains this component. Both component-level and system-level metrics must be captured during component development in accordance with the PMS 406 Autonomy Software Development Process. The system-level metrics are captured using the defined test bed / simulation tools as determined by the PAA (platform-specific system) or ABM (reference architecture system).

Metric Name	Metric Description
Fail the Objective	The Objective Executor fails its objective under the specified conditions. The Objective Executor does not fail its objective when the specified conditions aren't met.
Form Messages Correctly	The Component forms UMAA service messages correctly. This is per the UMAA standard and per the Component Specification (e.g., specified VariantType, optionals, etc.).
Monitor Performance	The Objective Executor monitors its progress or status towards meeting the specified objective, reporting this in its status message.
Status Messages	The Objective Executor reports its status at some rate.
Tolerance of Perceived to Planned Location	The Objective Executor measures the difference in its perceived location (provided by external Component via UMAA service) and its planned location, reporting result in its status message.

5 Component interfaces

The sections below describe the interfaces used by the component. Each Service interface is modeled as an Interface Block with nested proxy ports for the Service's operations. Each nested proxy port has a flow property typed by a signal representing the Topic associated with the operation. The directivity of this flow property is set from the perspective of a Service Provider in relation to the DDS bus, by definition. Each signal representing a Topic has one attribute representing its Data Type. The Topic and Data Type is the finest detail for the interface described in the Unmanned Maritime Systems Autonomy Model. For further details on the interface data, such as message and structure definitions, refer to the UMAA ICDs. Example use of the Services as part of a system can also be found in either the UMAA ICDs or in the Unmanned Maritime Systems Autonomy Model.

5.1 Conditional Control Svc IF

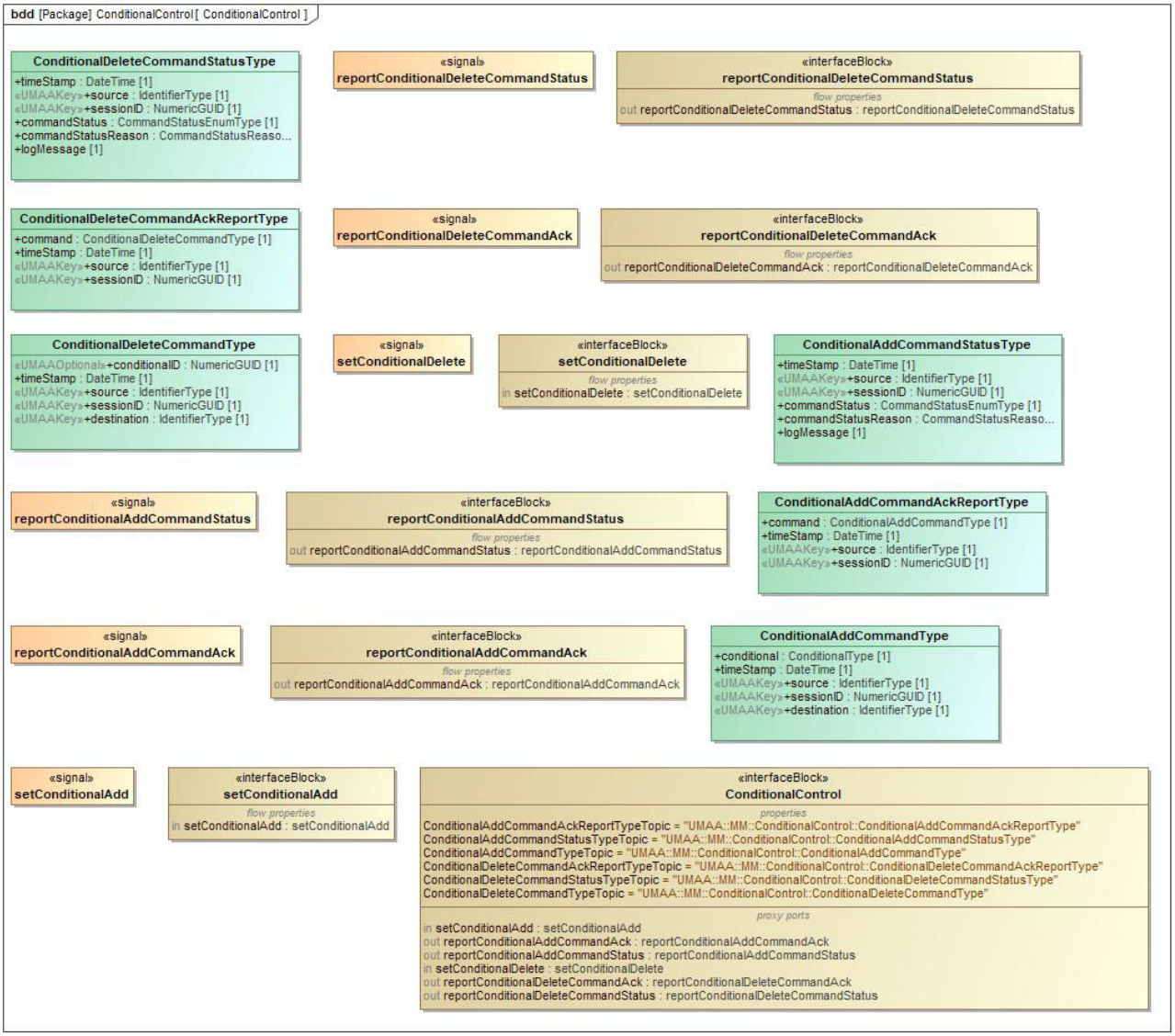


Figure 5. ConditionalControl

5.2 Mission Plan Constraint Control Svc IF

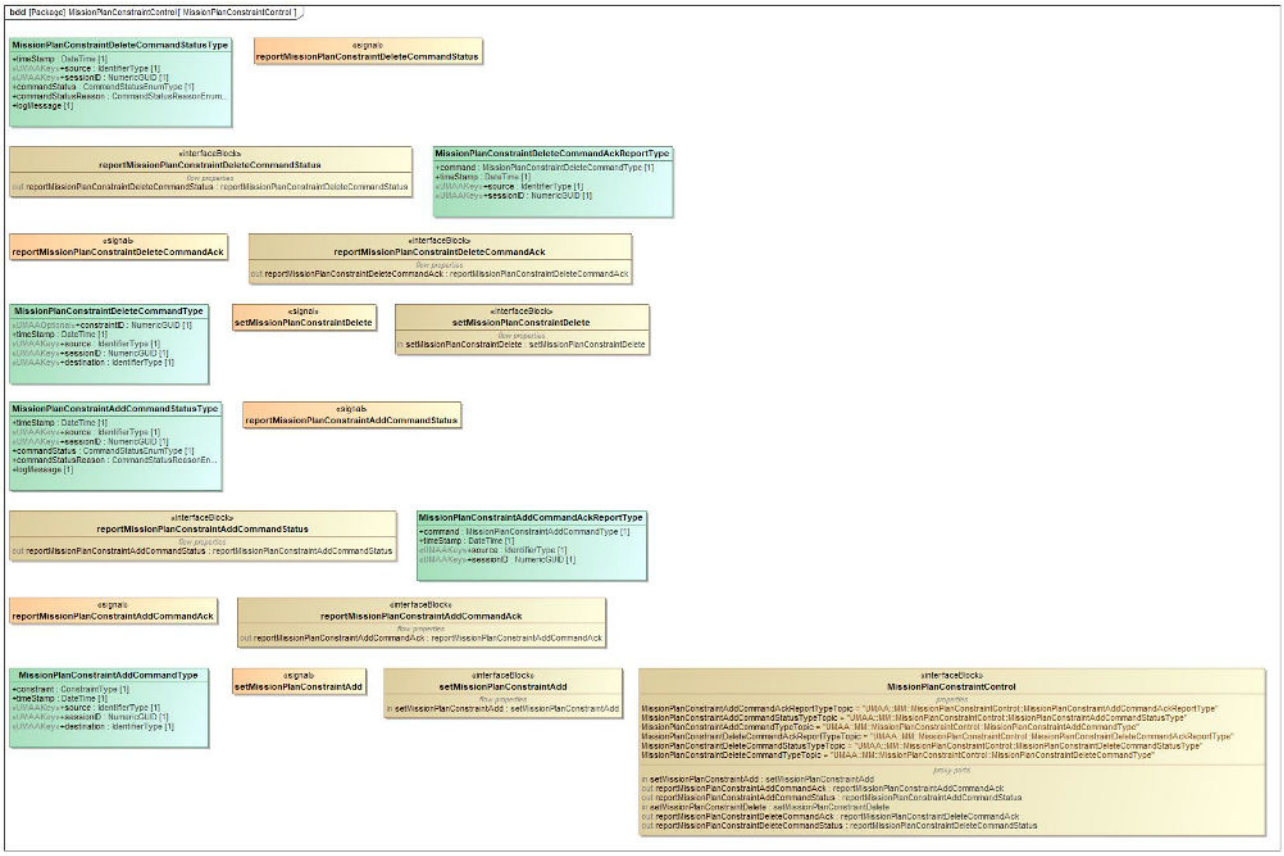


Figure 6. MissionPlanConstraintControl

5.3 Mission Plan Mission Control Svc IF

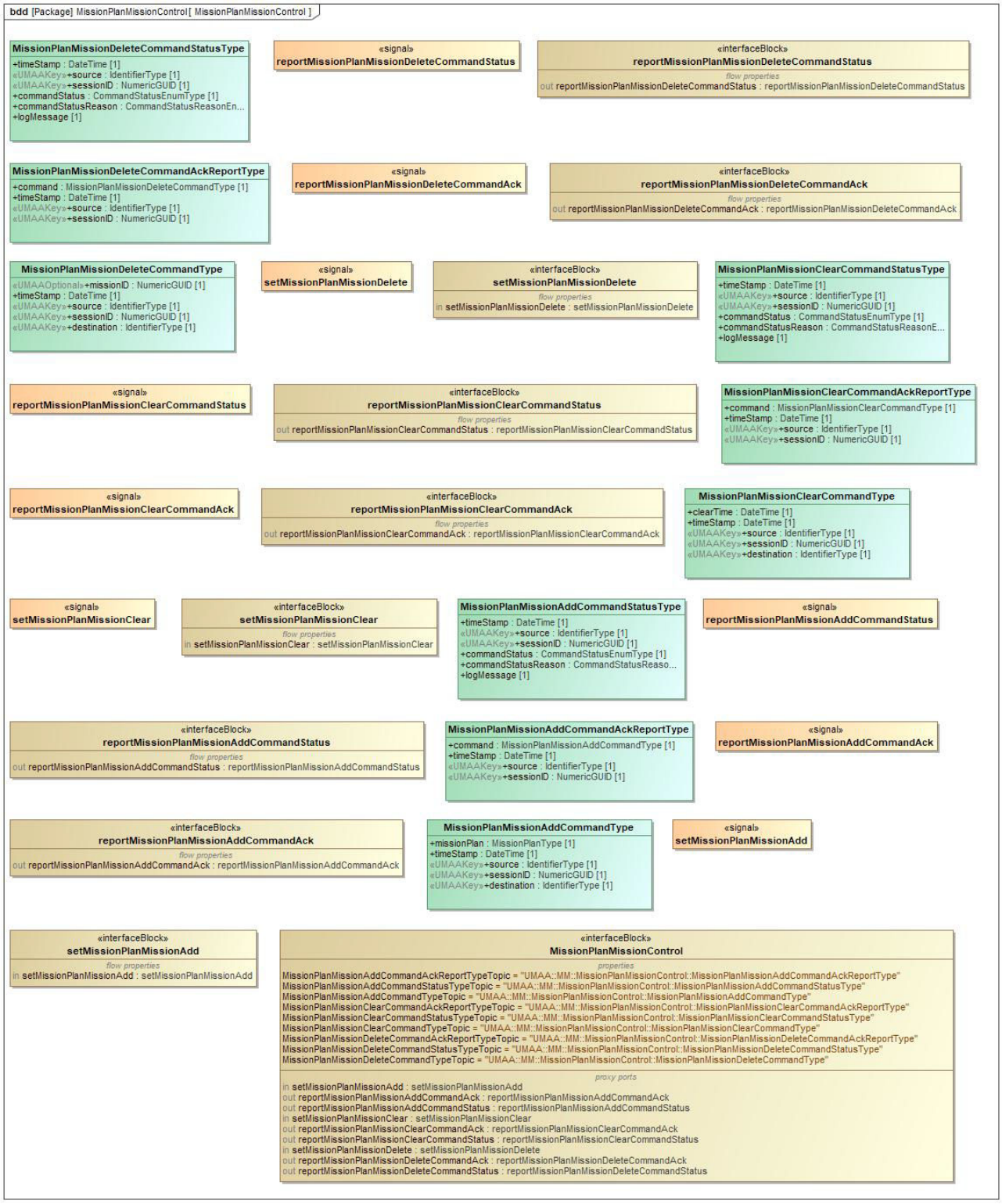


Figure 7. MissionPlanMissionControl

5.4 Mission Plan Task Control Svc IF

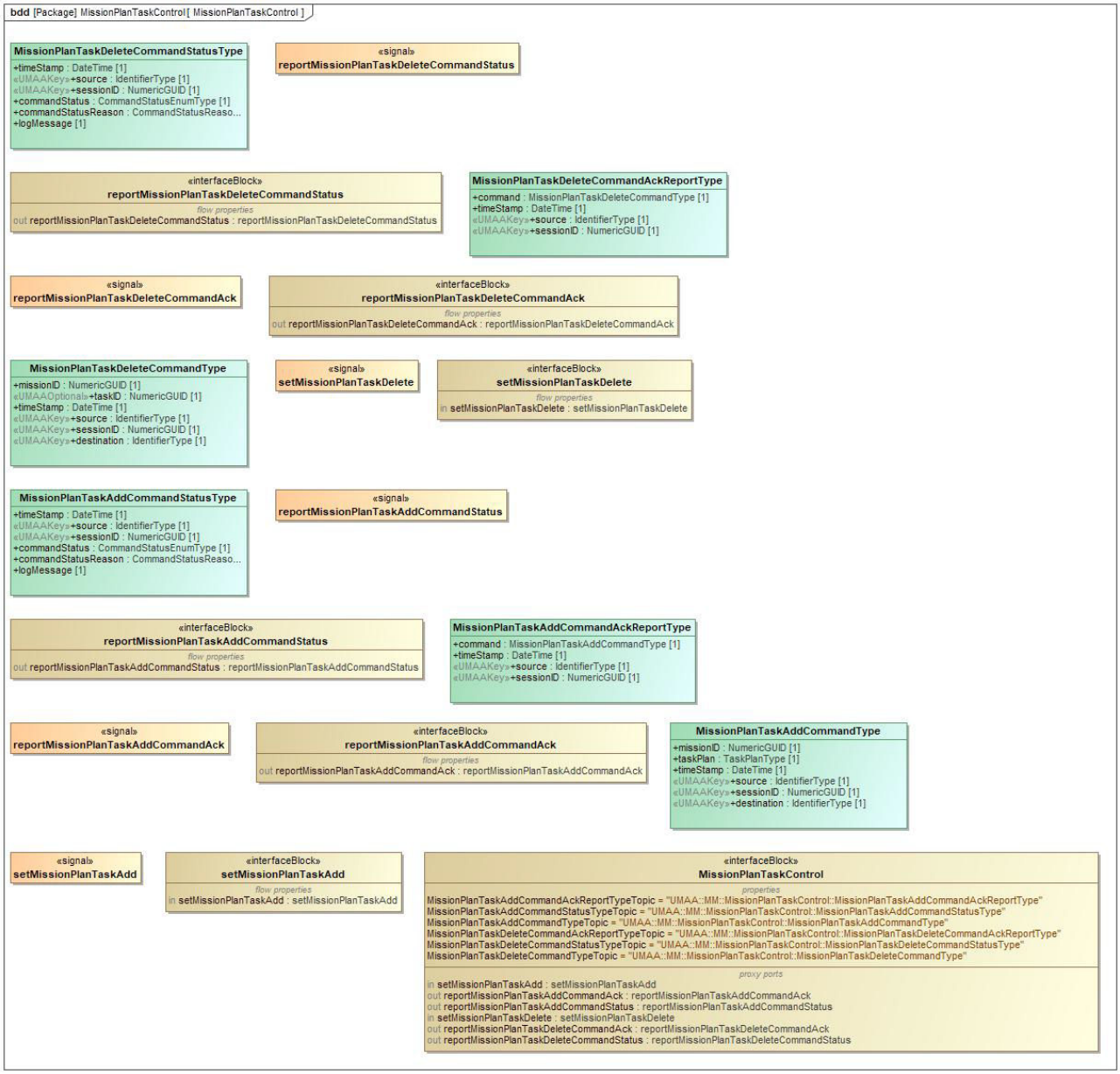


Figure 8. MissionPlanTaskControl

5.5 Mission Plan Objective Control Svc IF

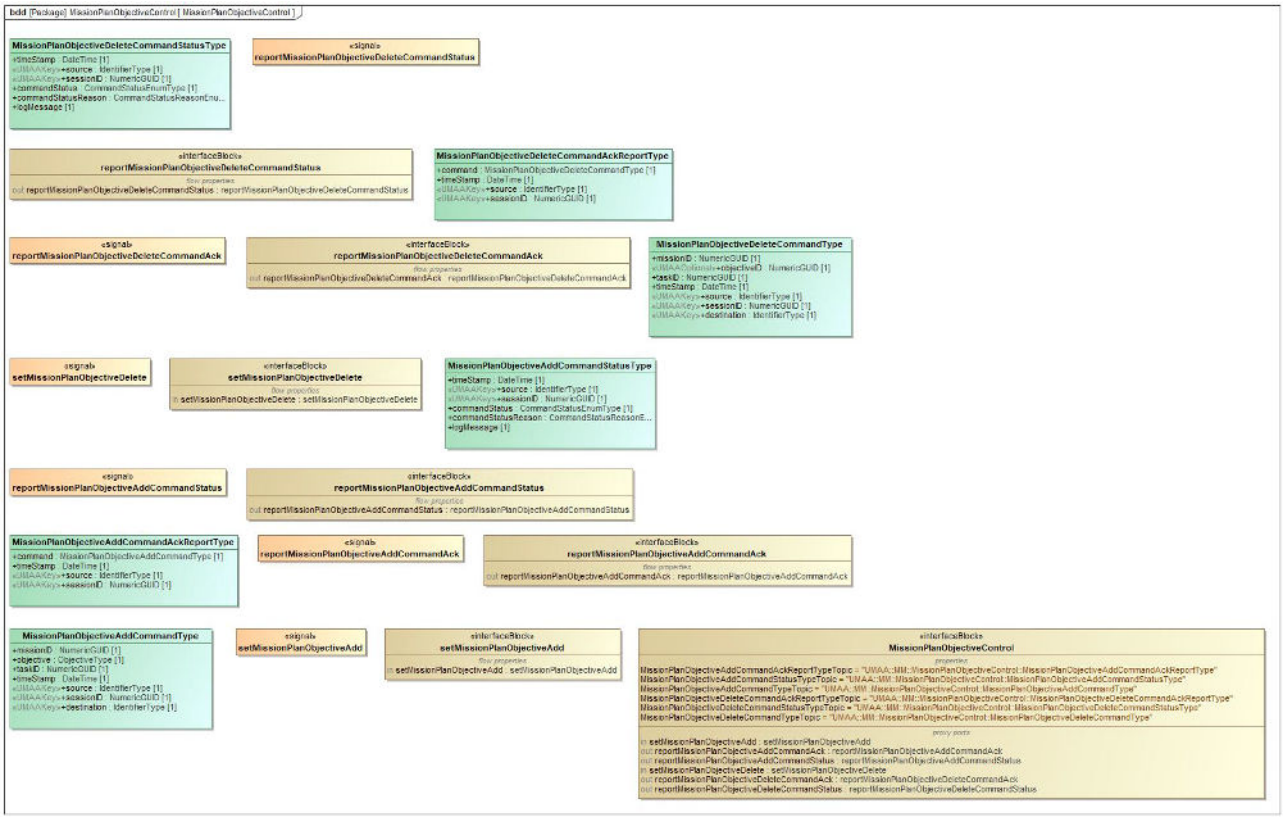


Figure 9. MissionPlanObjectiveControl

5.6 Mission Plan Execution Status Svc IF

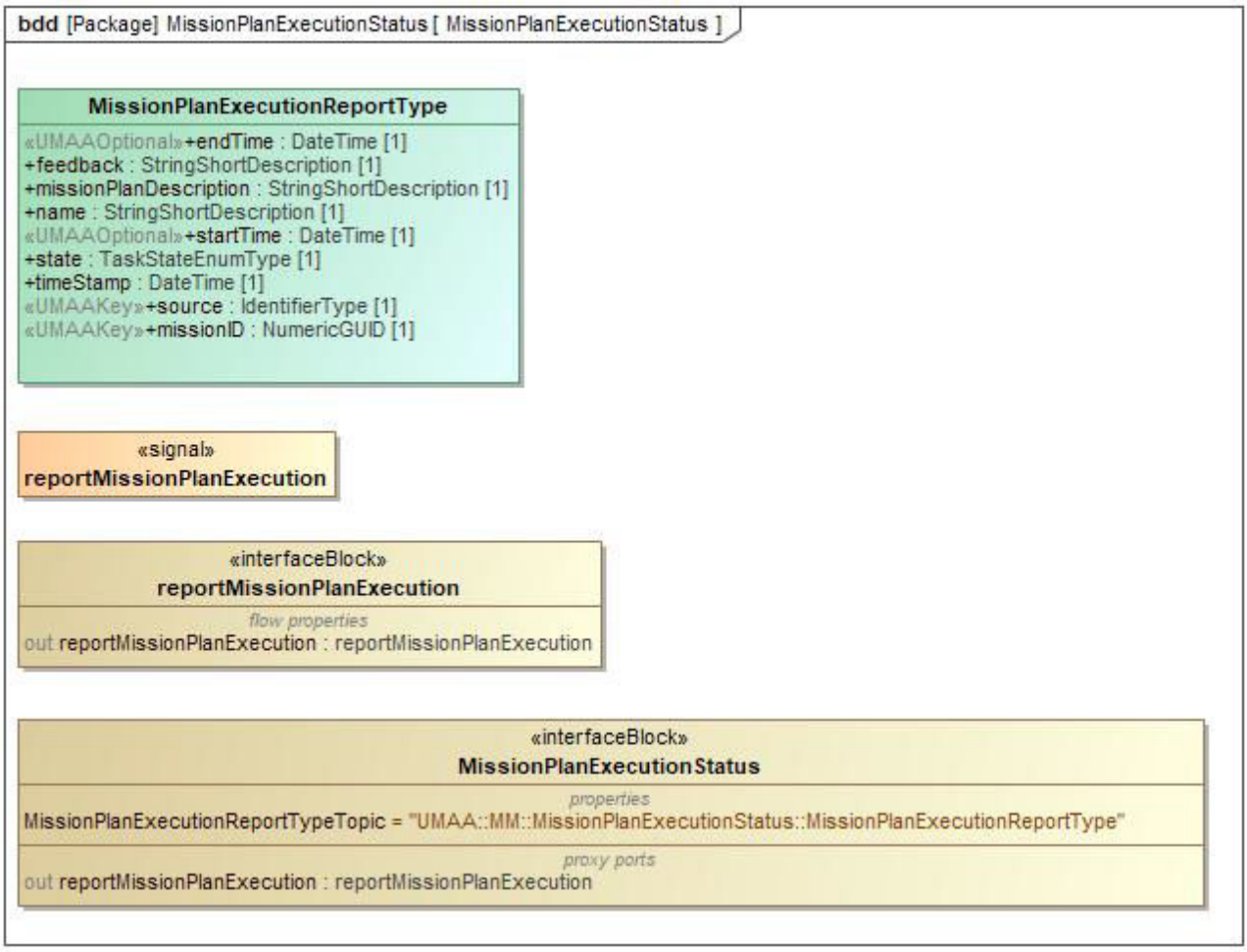


Figure 10. MissionPlanExecutionStatus

5.7 Task Plan Execution Status Svc

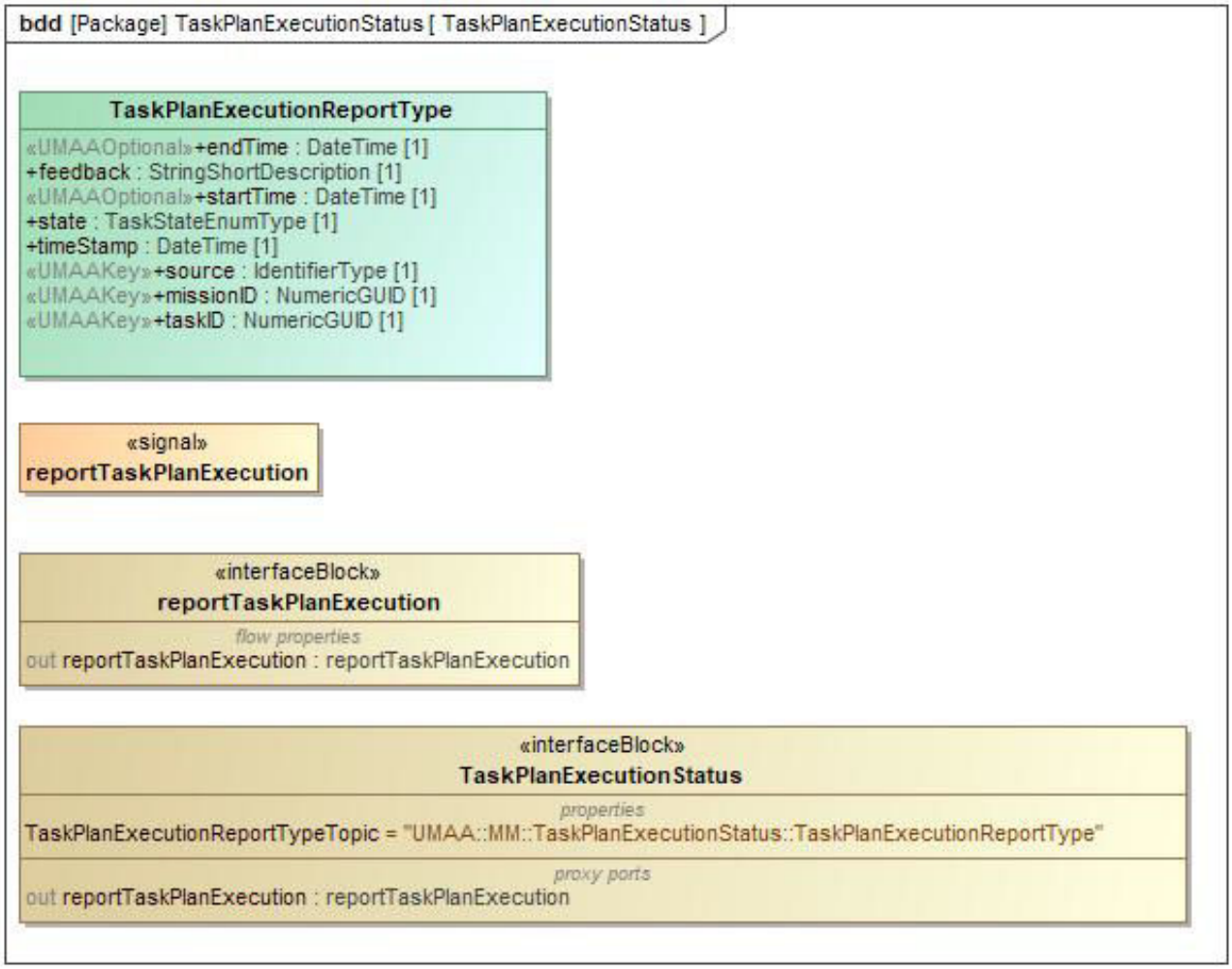


Figure 11. TaskPlanExecutionStatus

5.8 Mission Plan Execution Control Svc IF

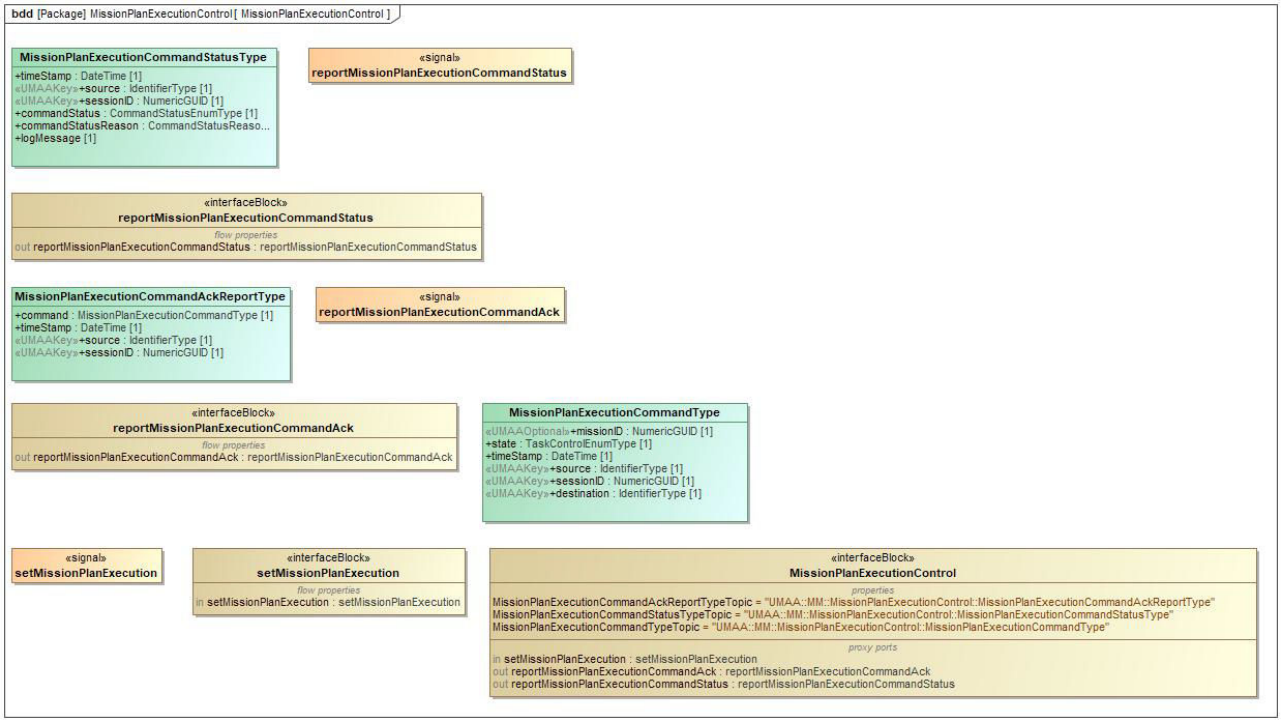


Figure 12. MissionPlanExecutionControl

5.9 Task Plan Execution Control Svc IF

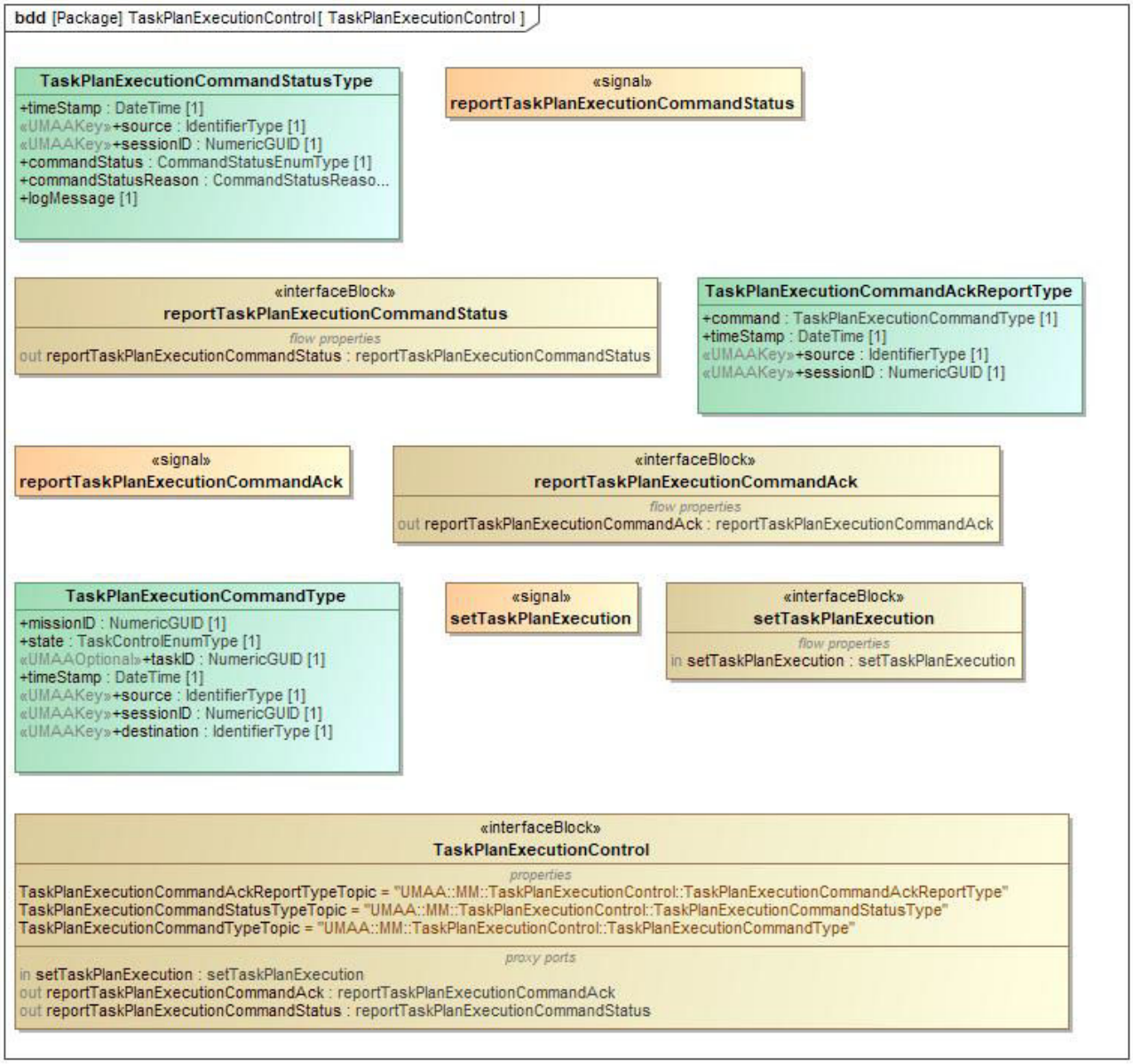


Figure 13. TaskPlanExecutionControl

5.10 Log Report Svc IF

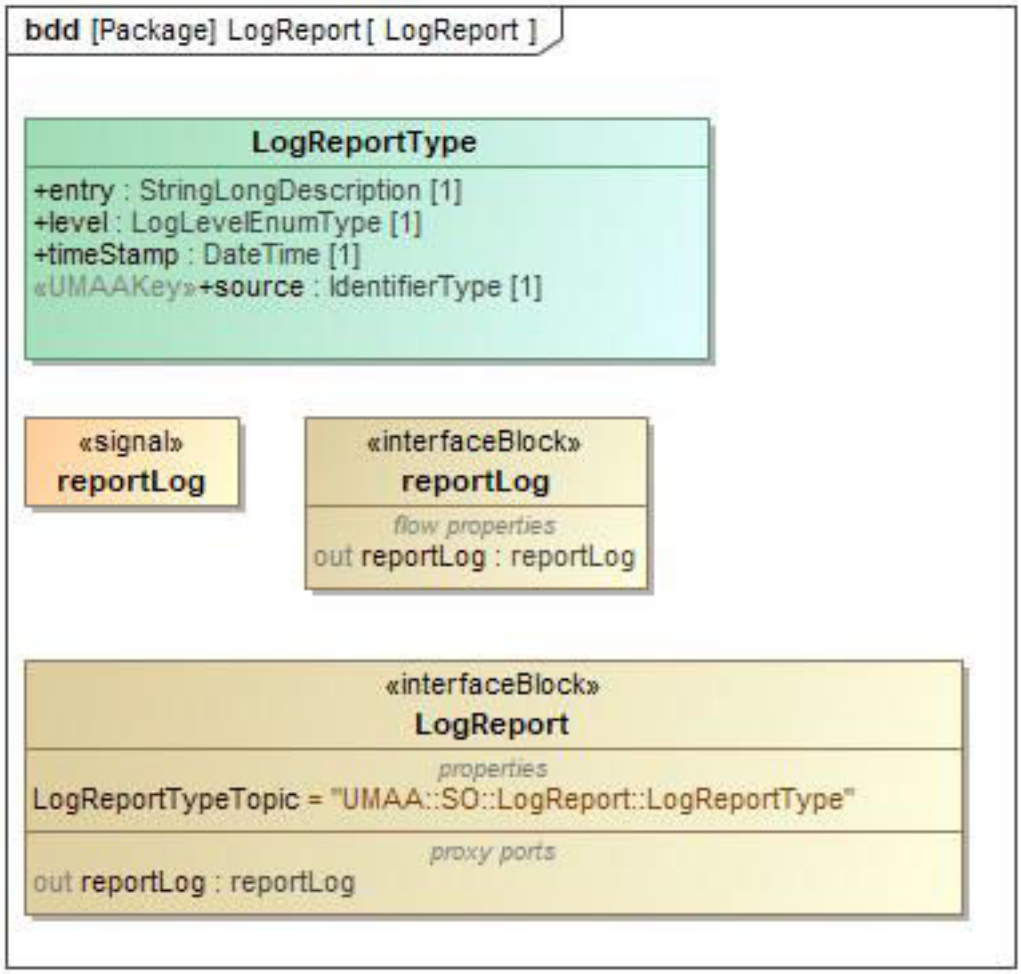


Figure 14. LogReport

5.11 Active Constraints Control Svc Consumer IF

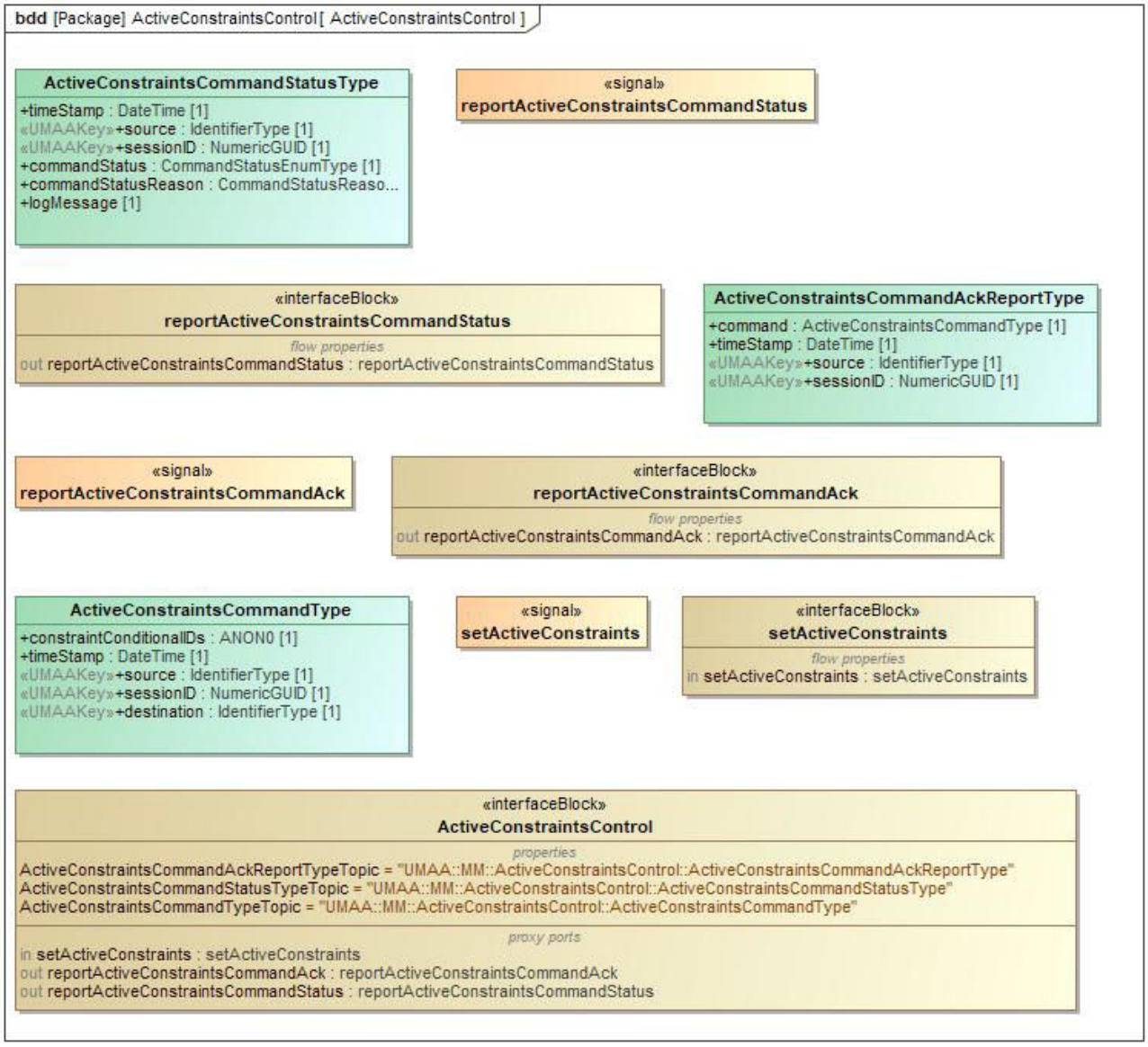


Figure 15. ActiveConstraintsControl

5.12 Conditional Report Svc IF

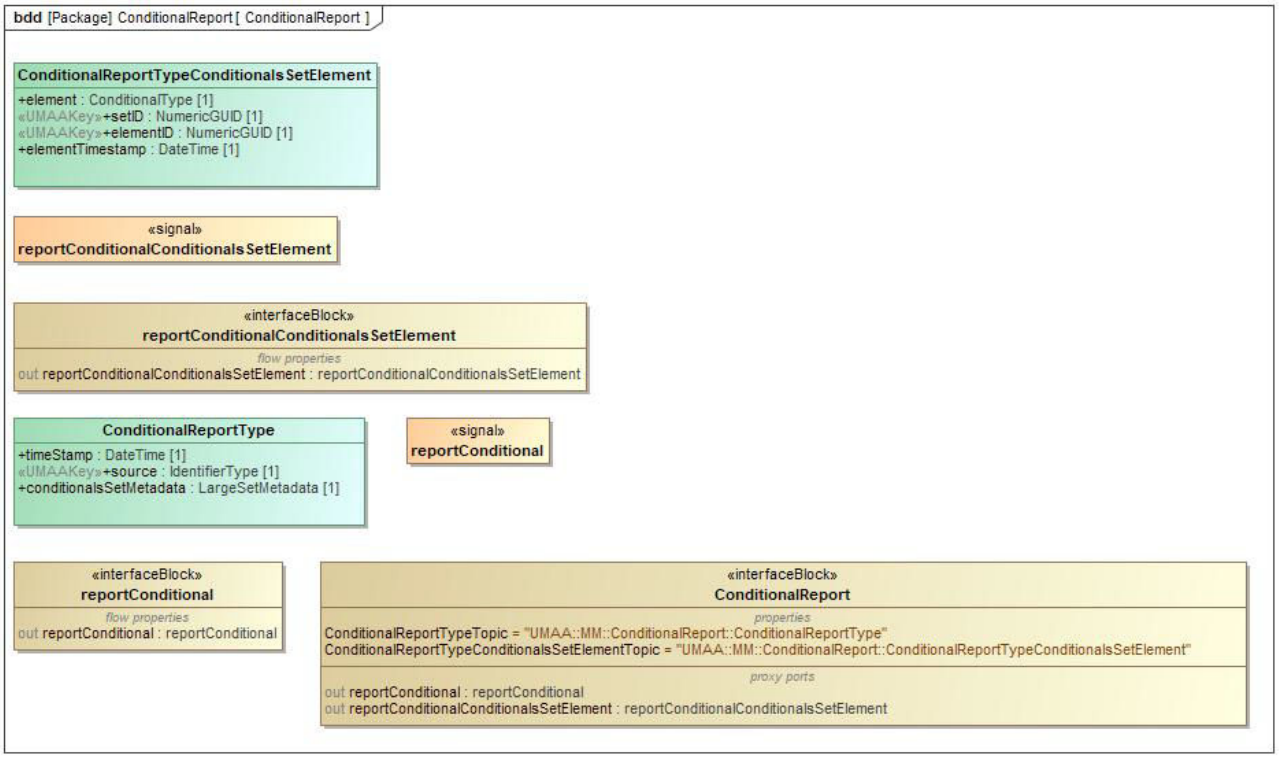


Figure 16. ConditionalReport

5.13 Health Report Svc IF

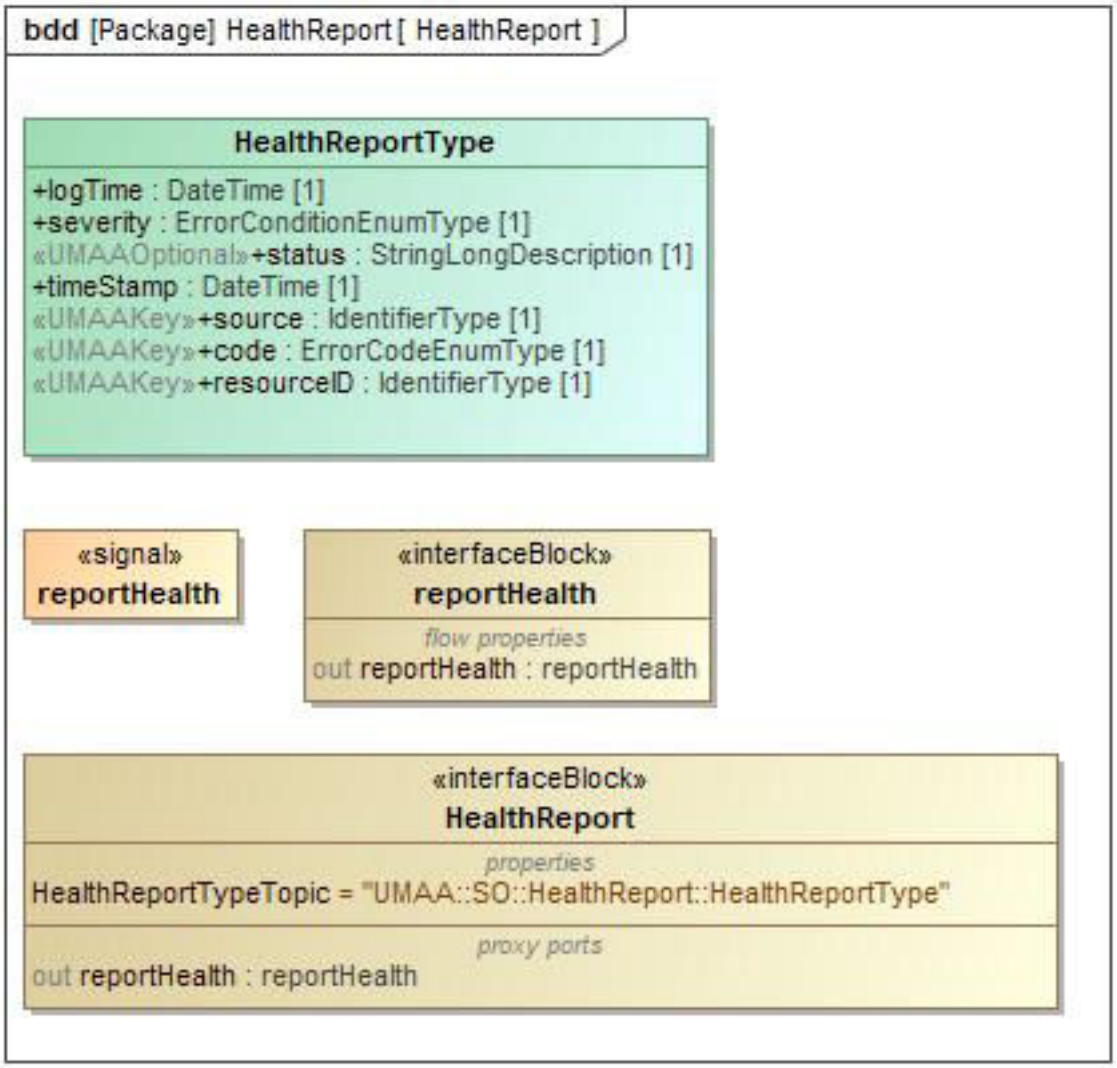


Figure 17. HealthReport

5.14 Objective Execution Status Svc IF

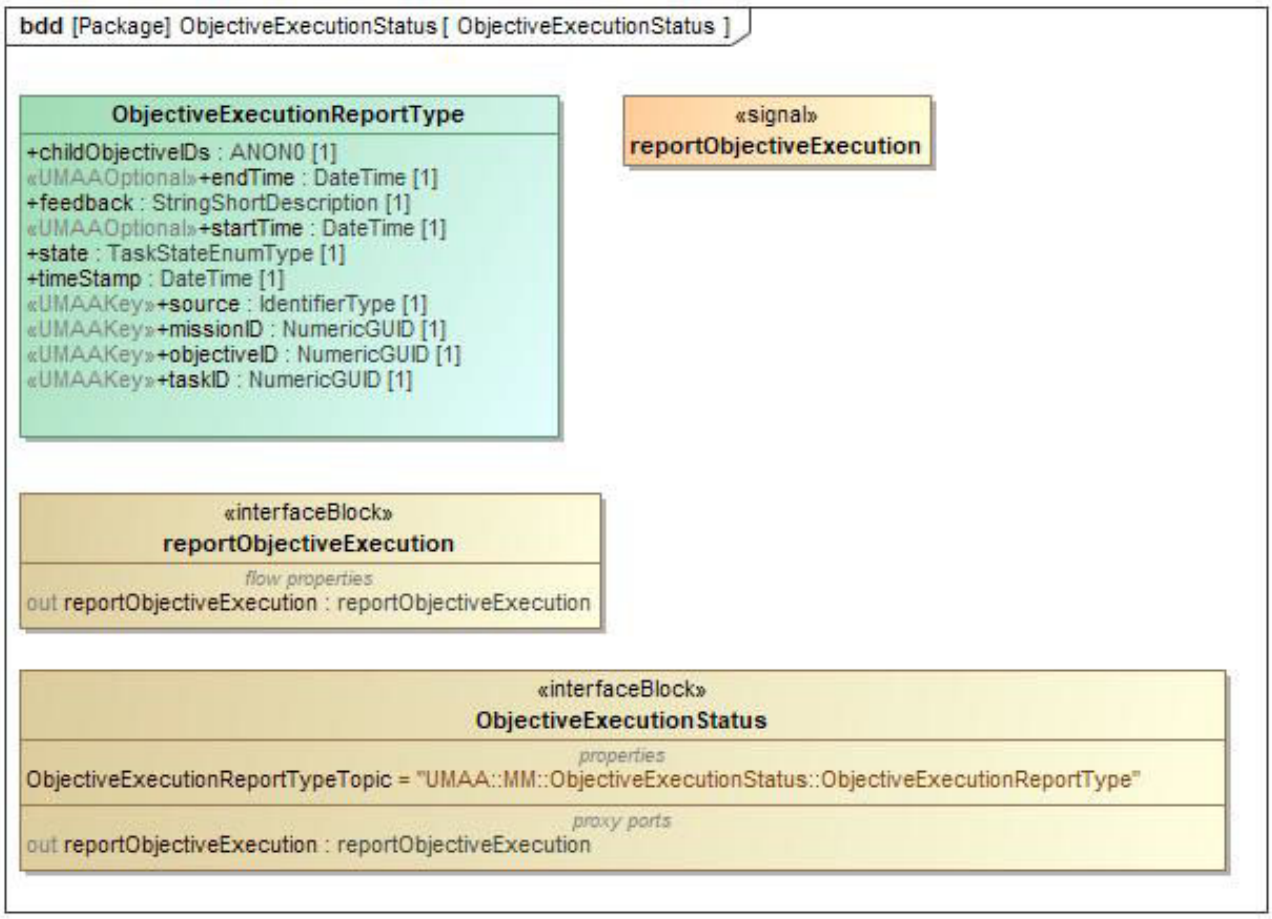


Figure 18. ObjectiveExecutionStatus

5.15 Global Pose Status Svc IF

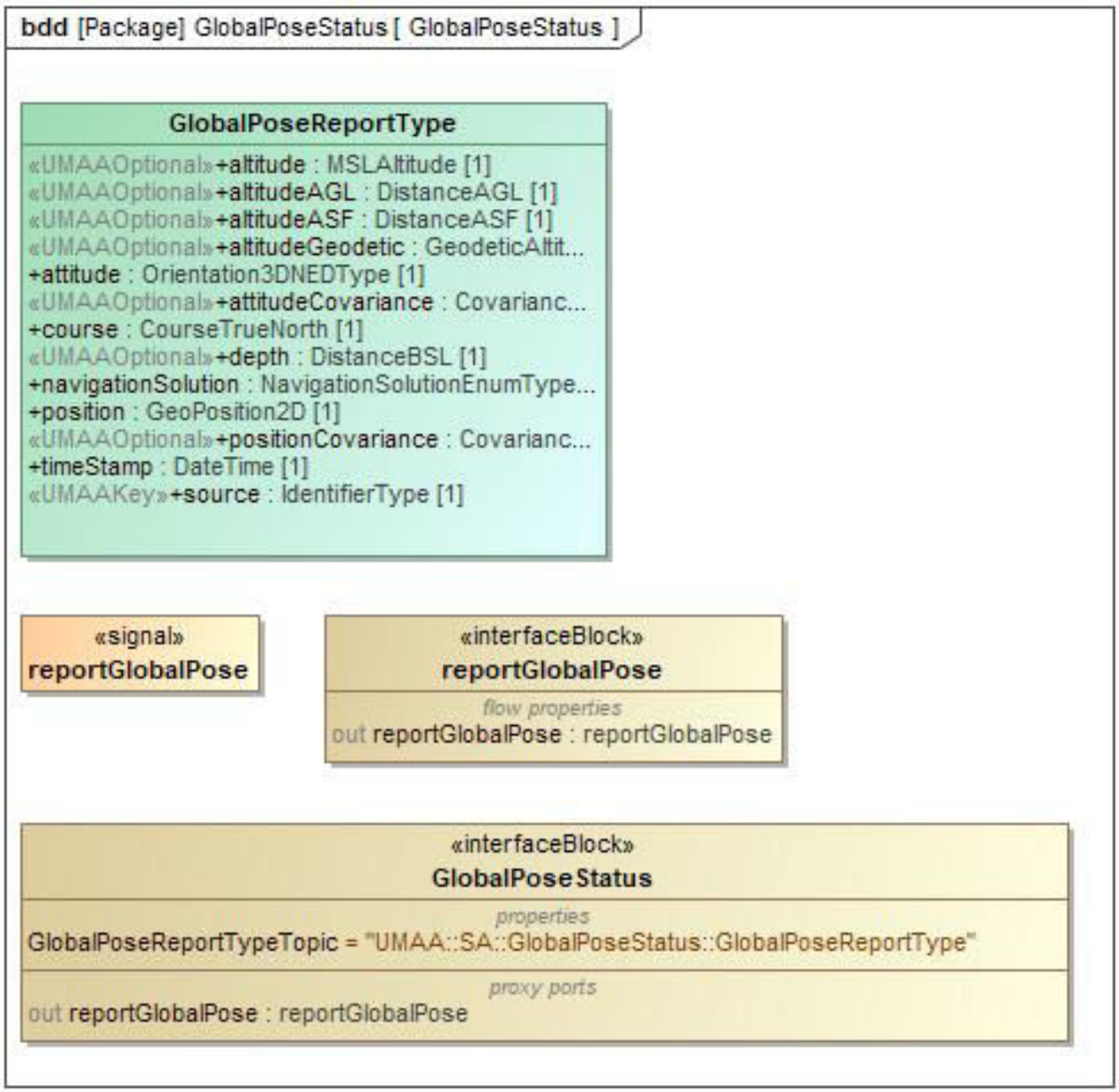


Figure 19. GlobalPoseStatus

5.16 Speed Status Svc IF

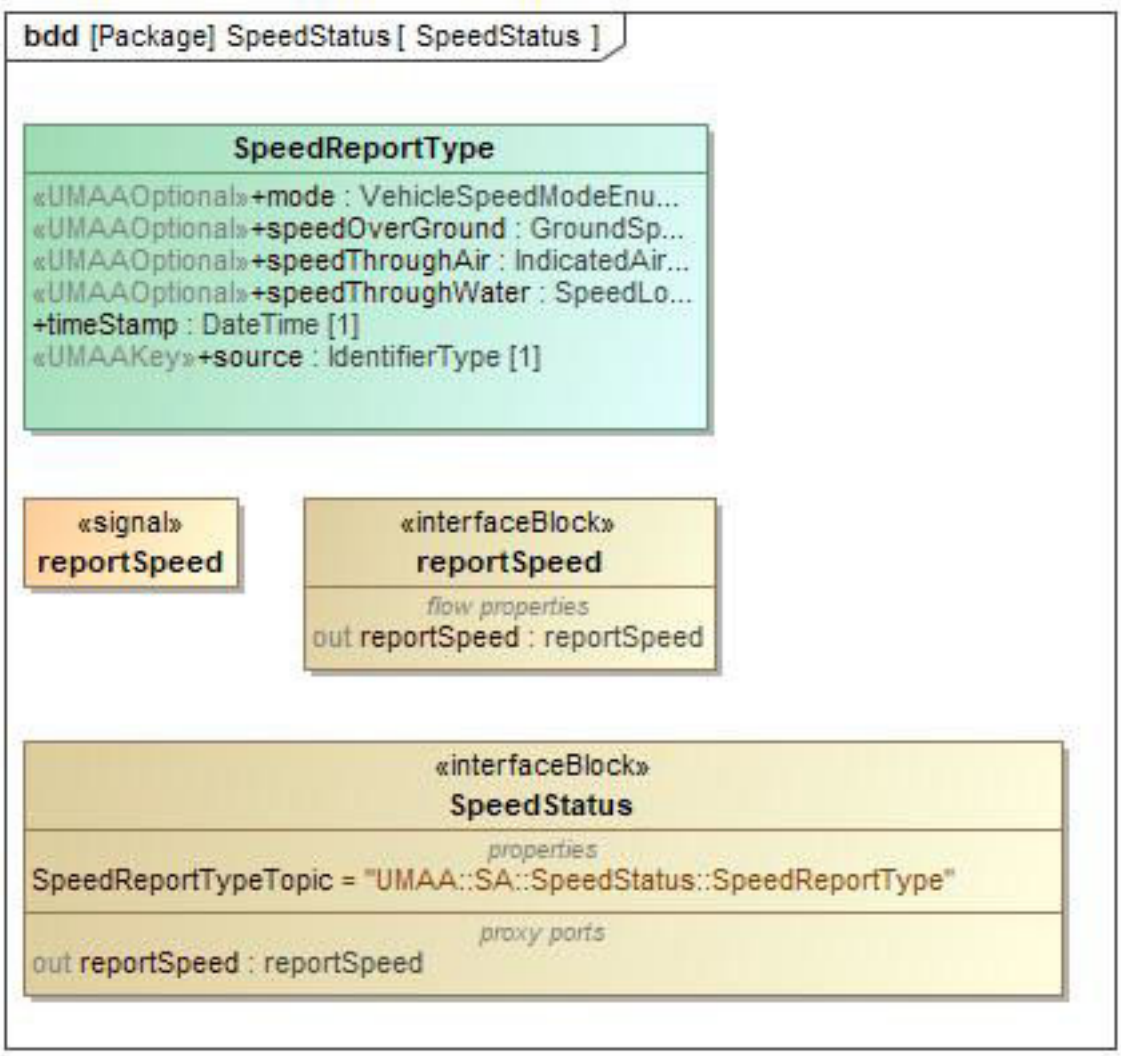


Figure 20. SpeedStatus

5.17 Global Vector Control Svc IF

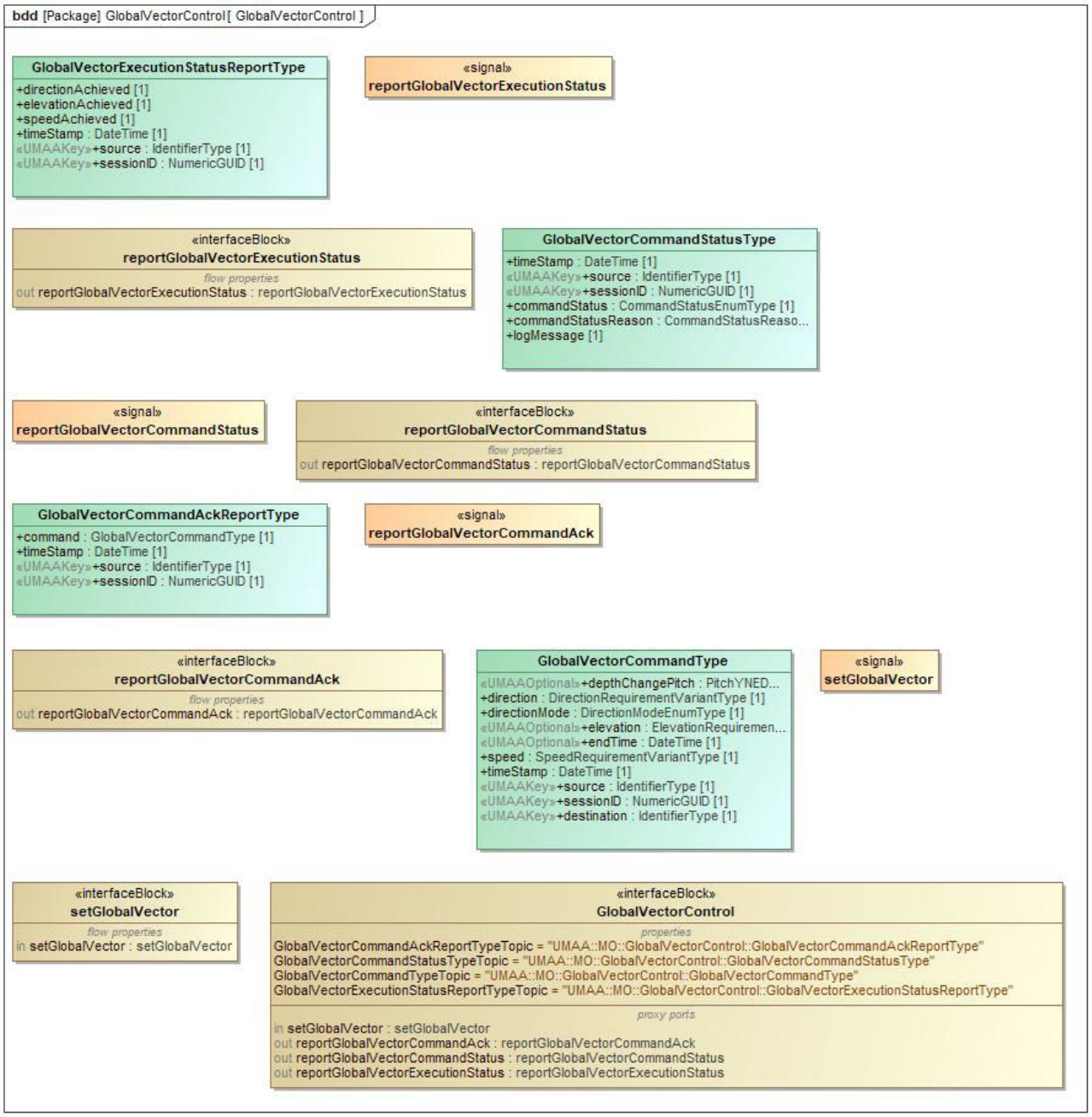


Figure 21. GlobalVectorControl

5.18 Global Waypoint Control Svc IF

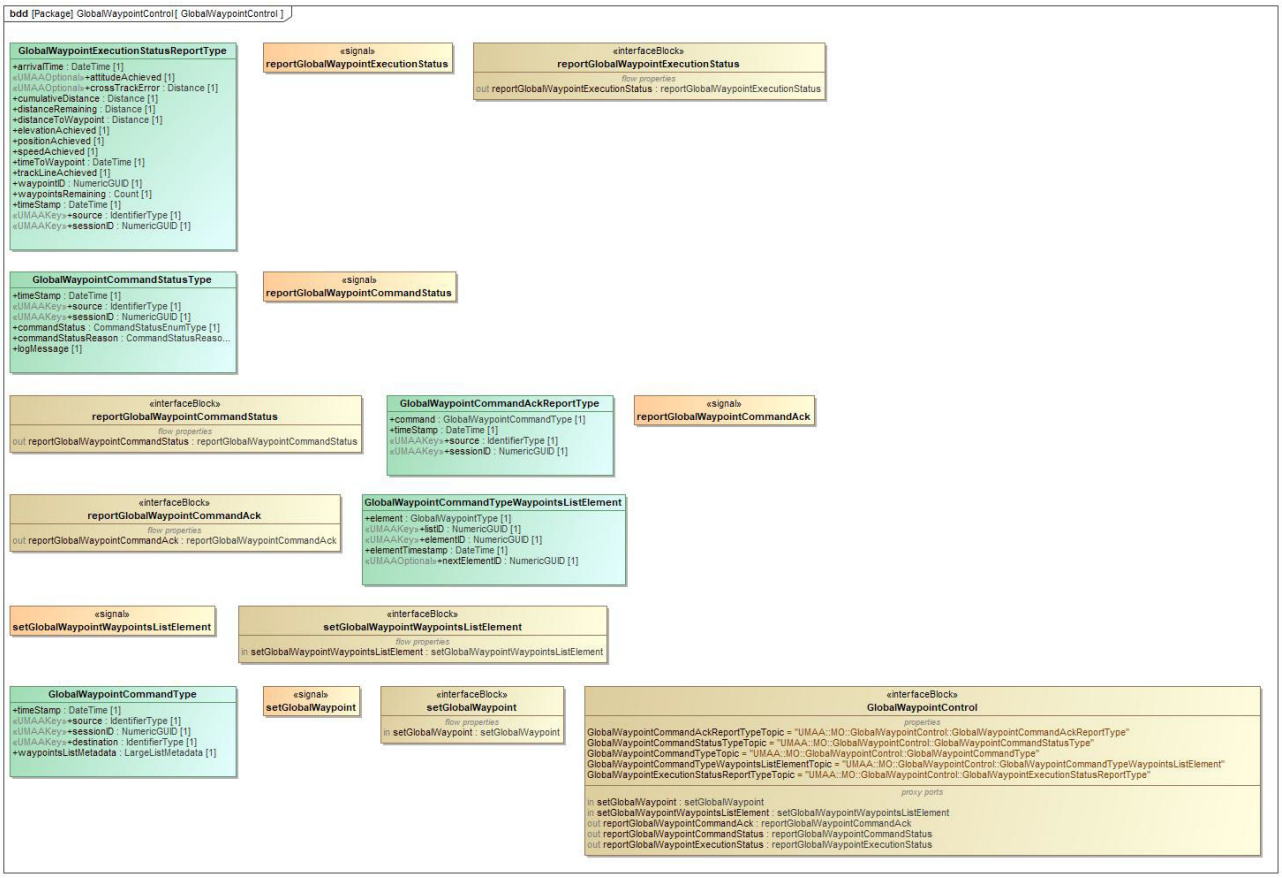


Figure 22. GlobalWaypointControl

5.19 Global Hover Control Svc IF

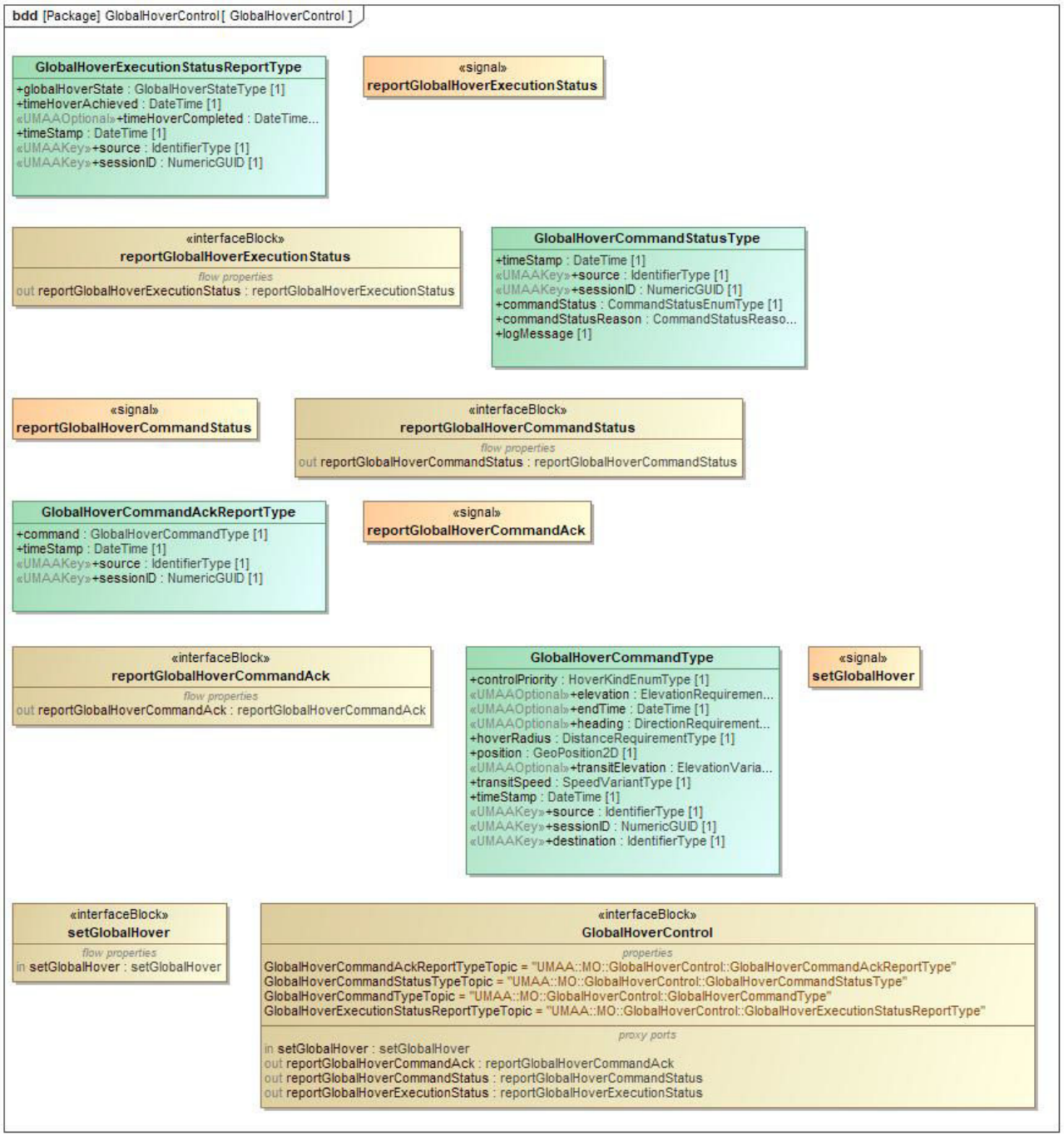


Figure 23. GlobalHoverControl

5.20 Velocity Status Svc IF

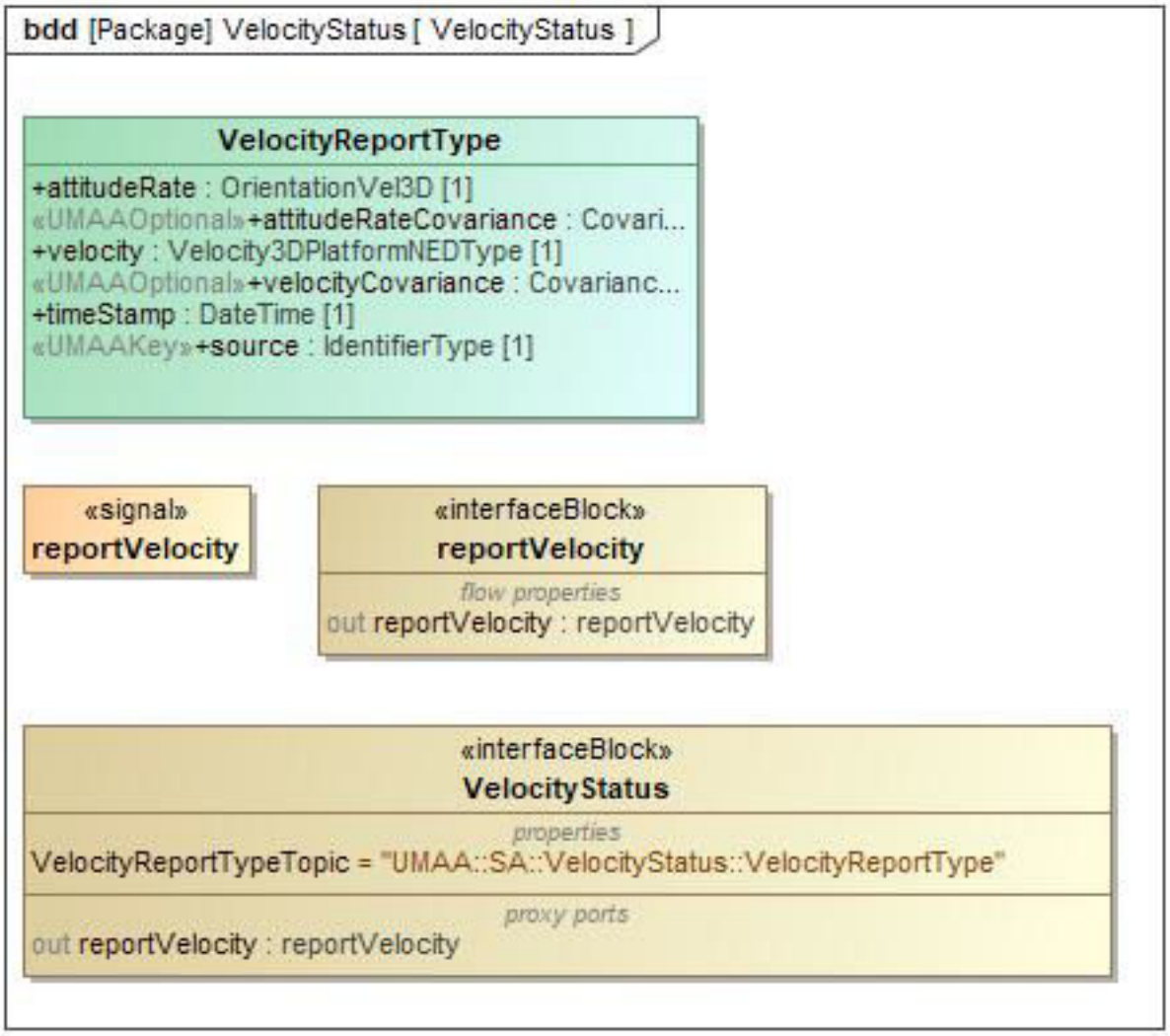


Figure 24. VelocityStatus

5.21 Contact Report Svc IF

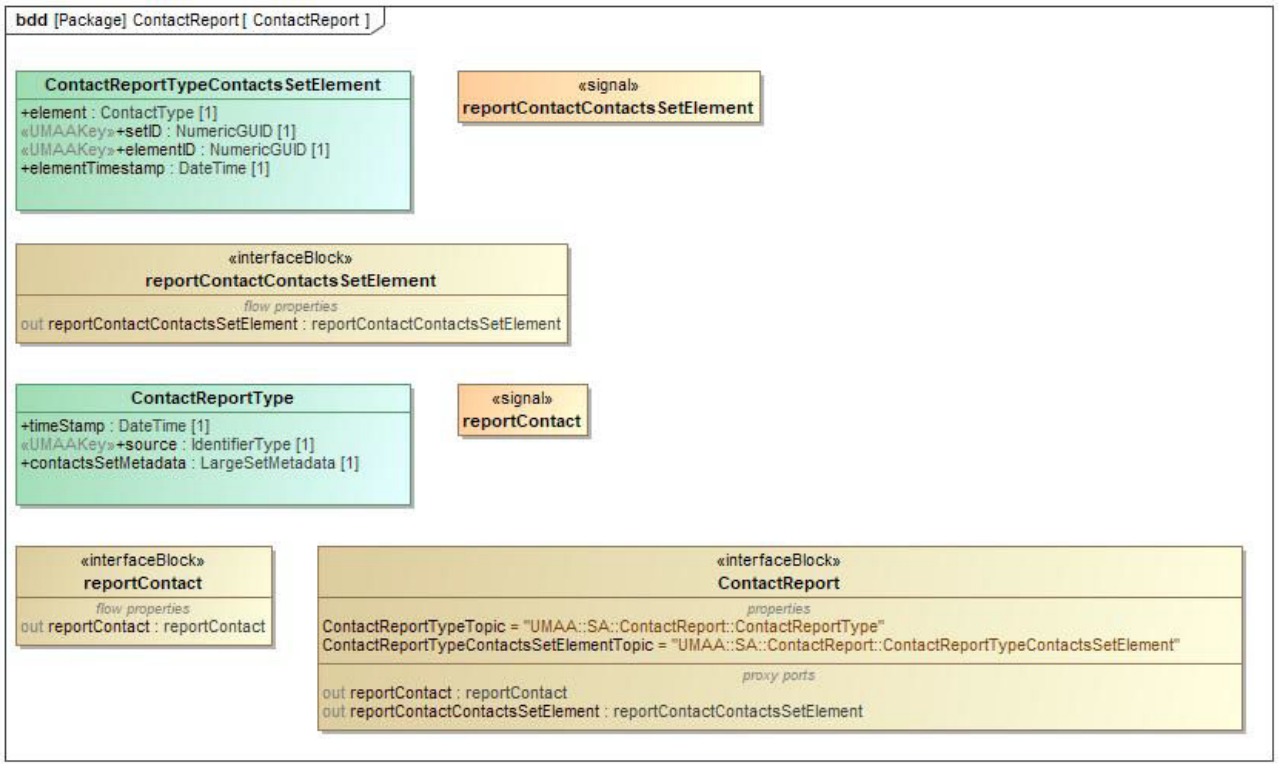


Figure 25. ContactReport

5.22 UV Platform Specs Svc IF

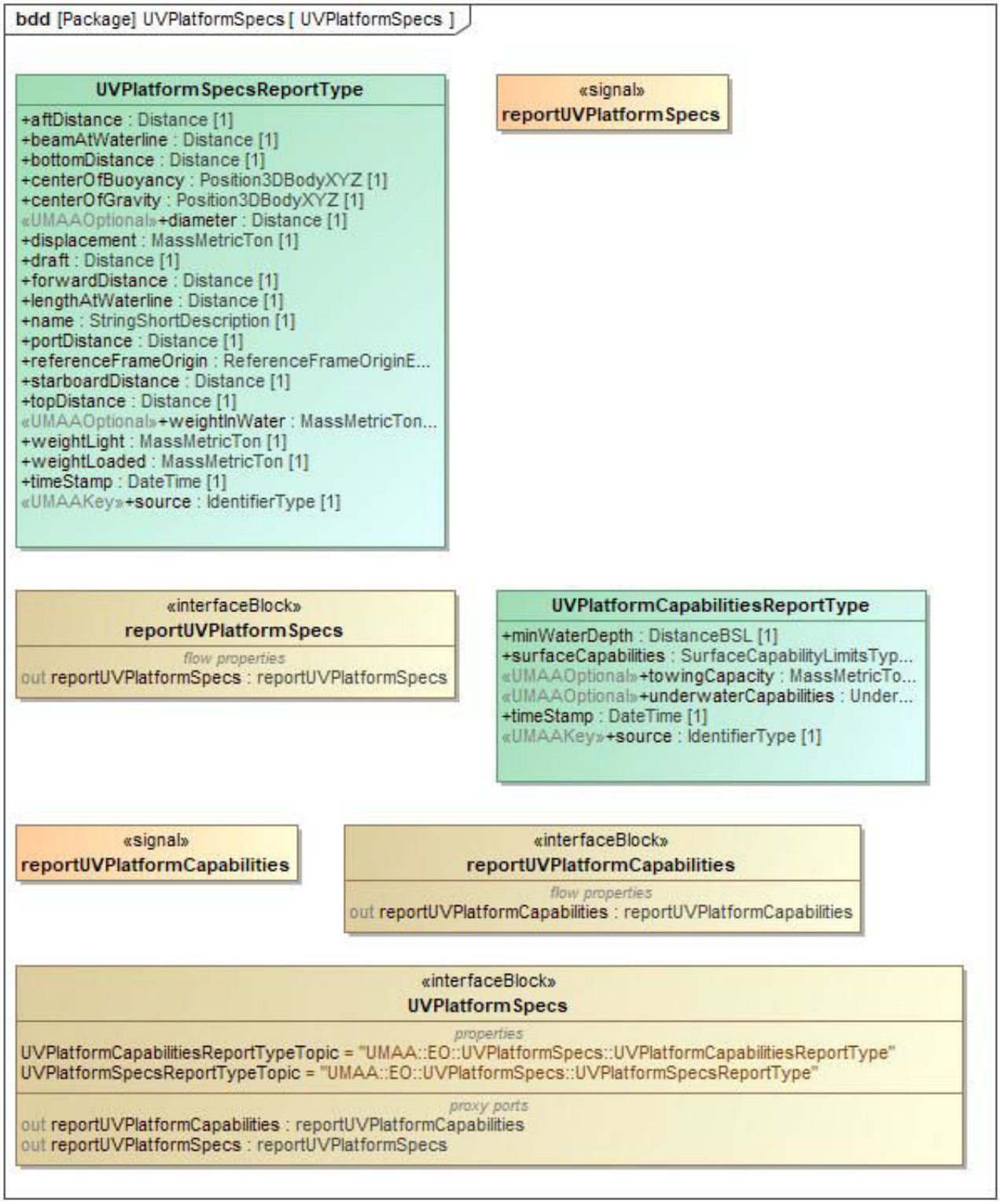


Figure 26. UVPlatformSpecs

5.23 Mission Plan Report Svc IF

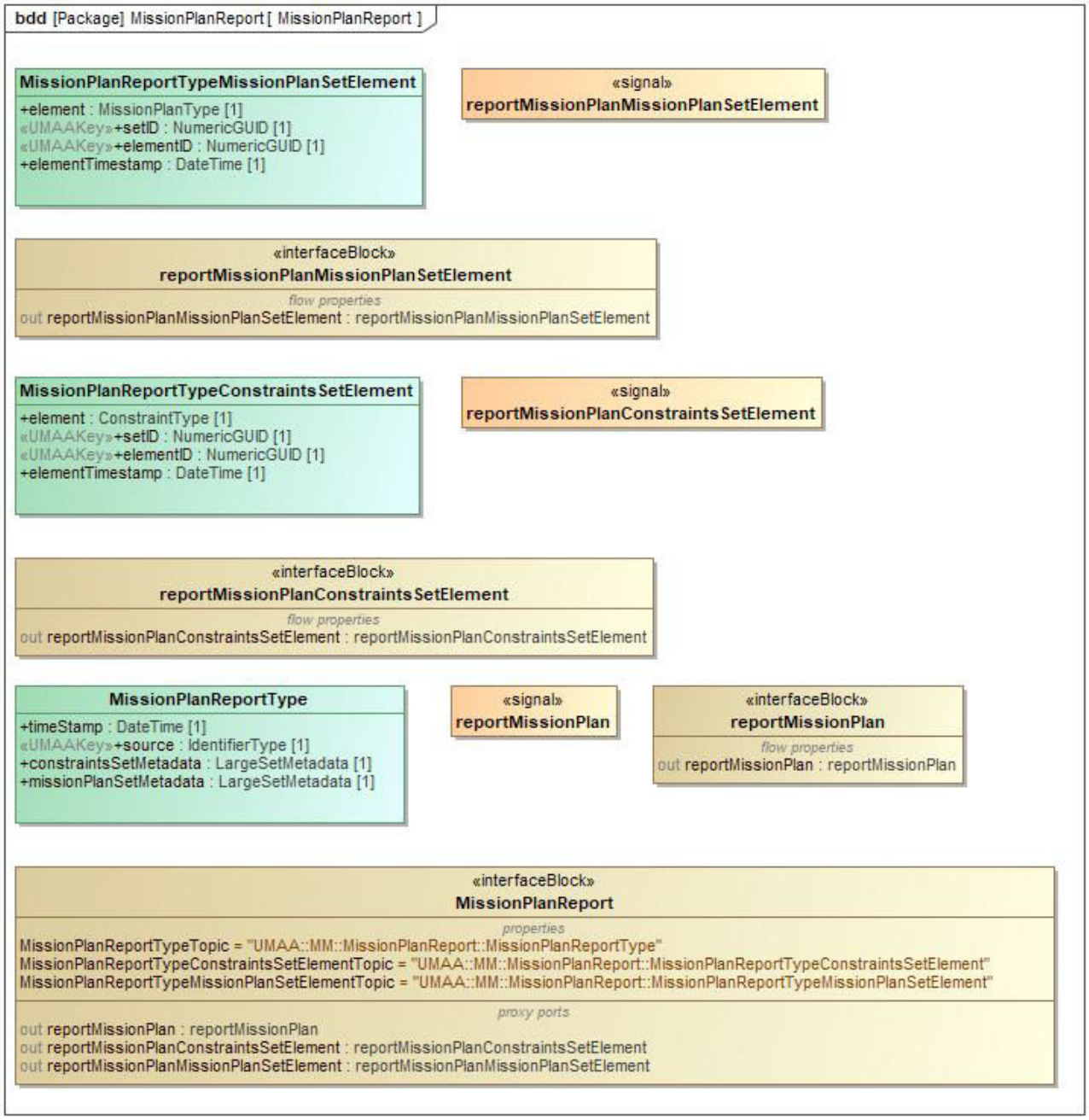


Figure 27. MissionPlanReport

5.24 Objective Execution Control Svc IF

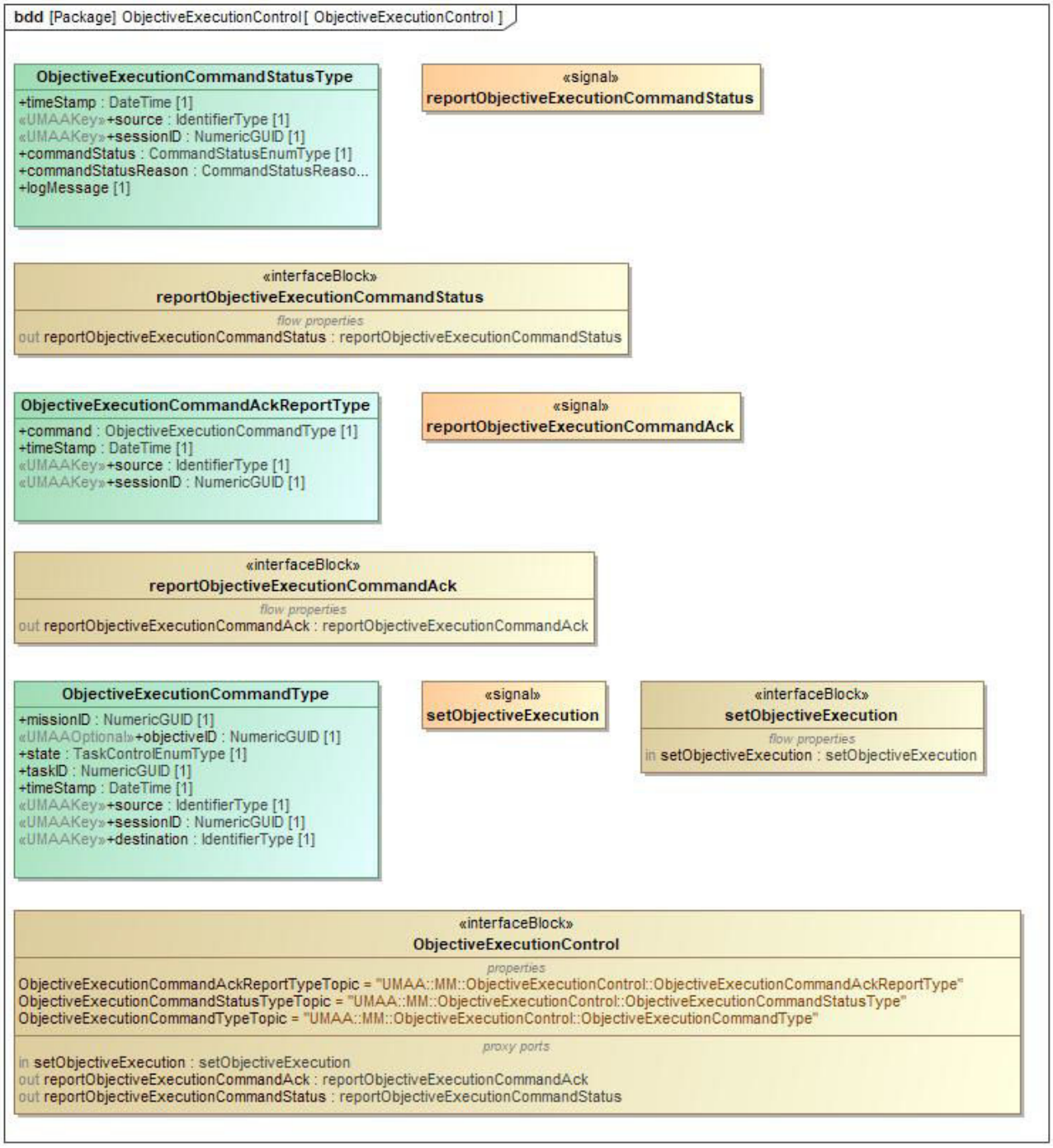


Figure 28. ObjectiveExecutionControl

5.25 Contact COLREGS Classification Status Svc IF

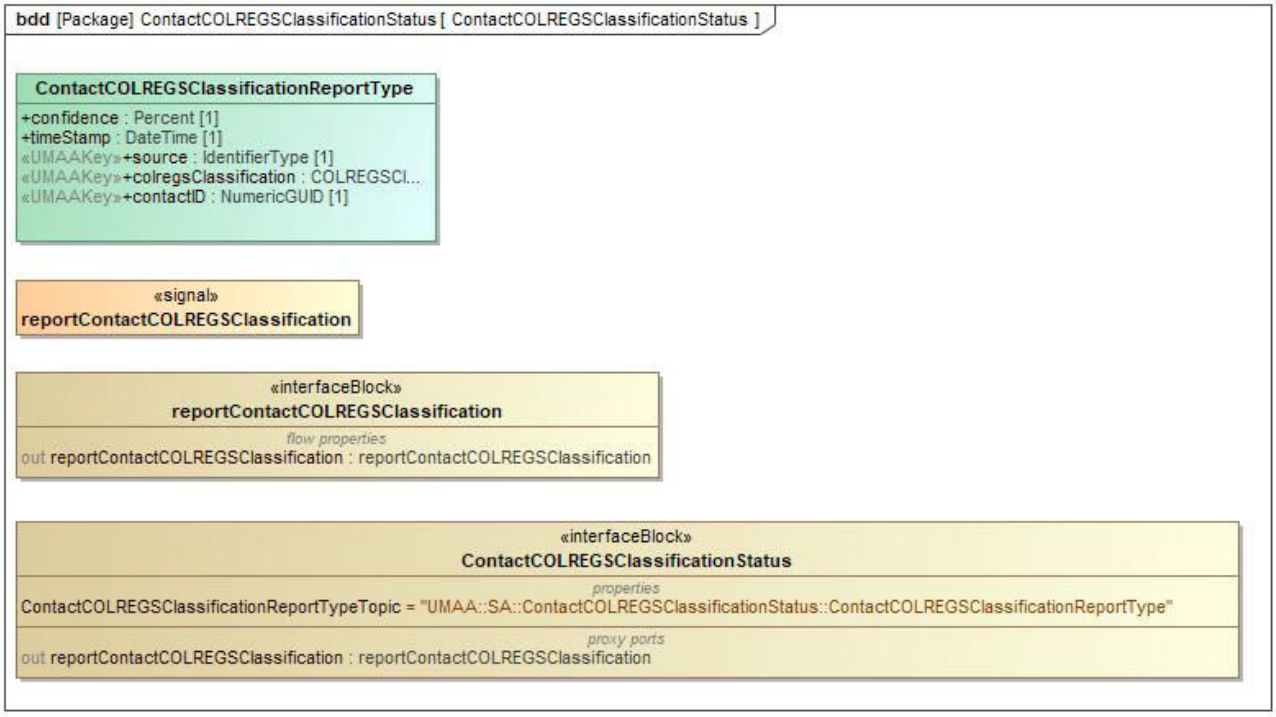


Figure 29. ContactCOLREGSClassificationStatus

5.26 Contact Visual Classification Status Svc IF

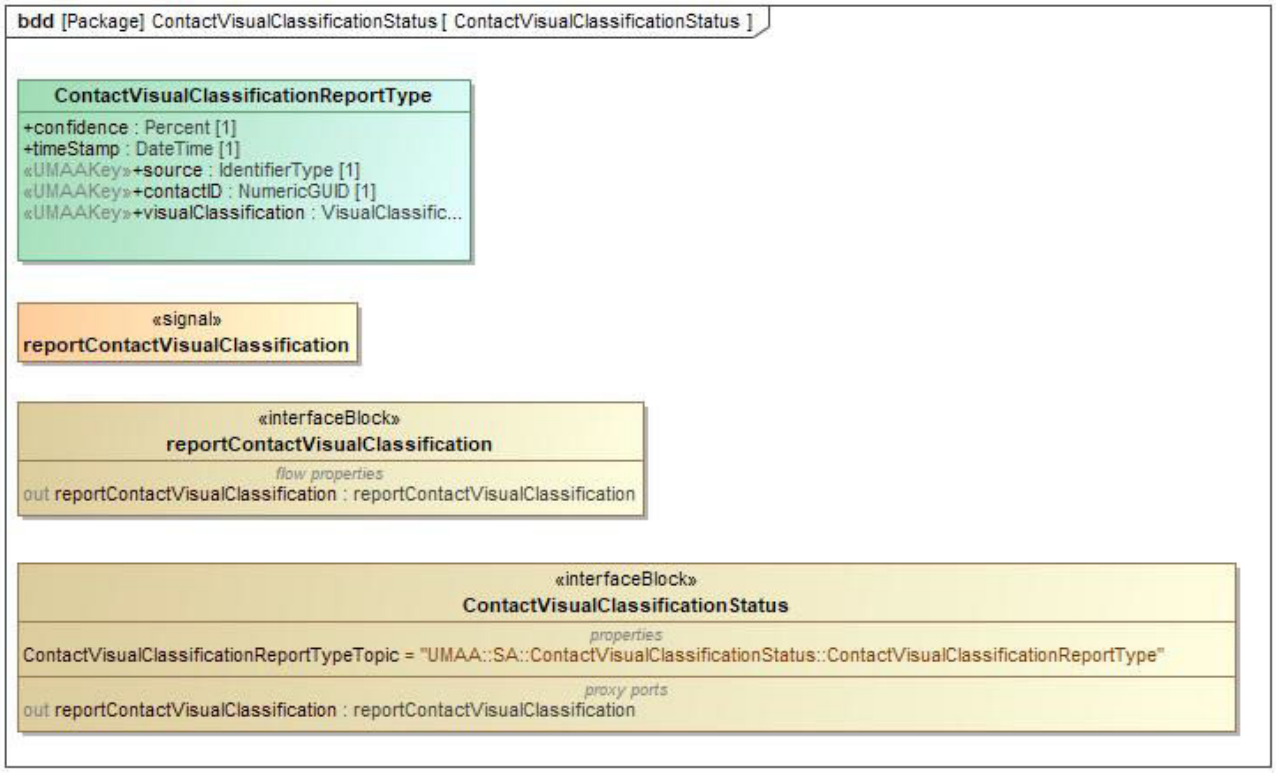


Figure 30. ContactVisualClassificationStatus

5.27 Water Current Status Svc IF

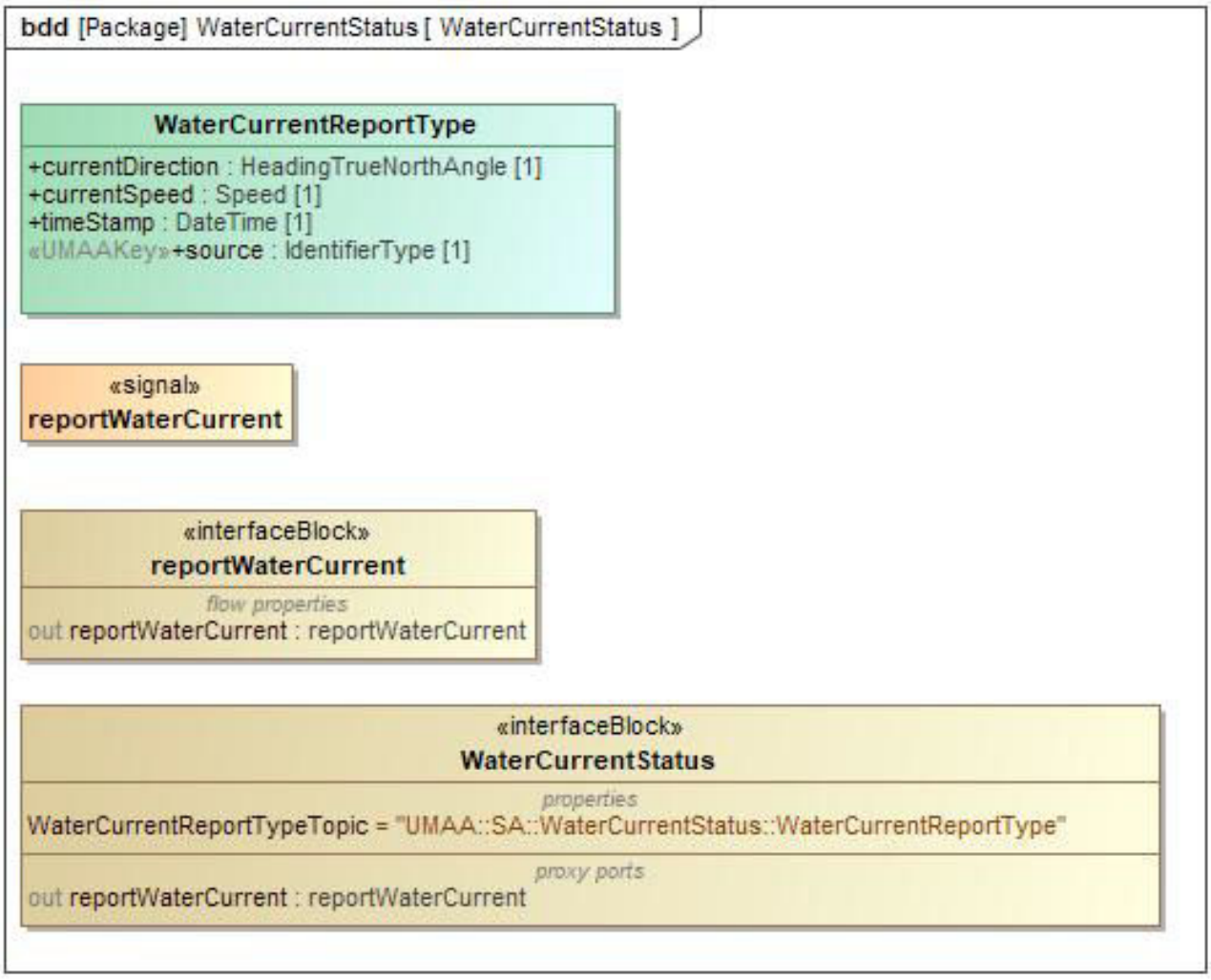


Figure 31. WaterCurrentStatus

5.28 Wind Status Svc IF

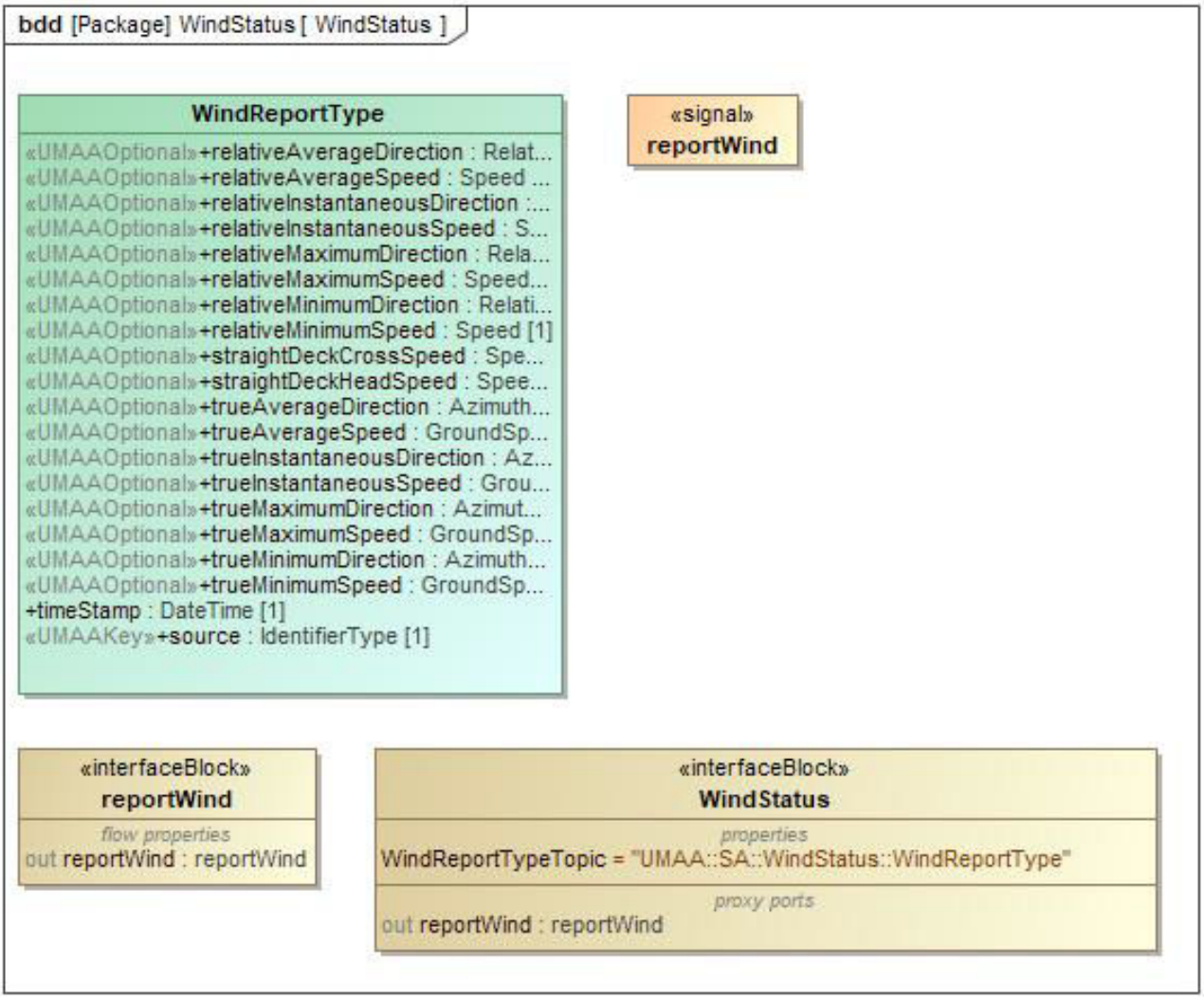


Figure 32. WindStatus

6 Traceability

Additional detail on traceability is available in the Unmanned Maritime Systems Autonomy Model.

END OF DOCUMENT