Date: ~~May 2023~~August 2024

Open Architecture Radar Interface Standard

FTF-1~~Final Submission~~ - 3.0

OMG Document Number: c4i/23-07~~5~~-01

Standard document URL: https://www.omg.org/spec/OARIS/3.0

Machine Consumable File(s): https://www.omg.org/OARIS/~~20230201~~20240802

Associated Normative Machine Consumable Files:

https://www.omg.org/spec/OARIS/~~20230501~~20240802/oaris.xmi/
https://www.omg.org/spec/OARIS/~~20230501~~20240802/oaris.graphqls/
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/TimeBase.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Common_Types.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/System_Track.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Coordinates_and_Positions.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Shape_Model.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Requests.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Encyclopaedic_Support.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Extended_Subsystem_Control.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Recording_and_Replay.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Simulation_Support.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Subsystem_Control.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Clutter_Reporting.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Media_Streaming.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Sensor_Assessment.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Supplementary_Measurement.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Search.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Plot_Reporting.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Sensor_Control.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Sensor_Performance.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Track_Reporting.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Tracking_Control.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Air_Engagement_Support.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Engagement_Support.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Missile_Guidance.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Surface_Engagement_Support.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Receive_Encyclopaedic_Data.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Manage_Network_Participation.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Manage_Physical_Configuration.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Perform_Offline_Test.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Provide_Networking_Statistics.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Restart.idl

https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Shutdown.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Startup.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Control_Recording.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Control_Replay.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Define_Simulation_Scenario.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Control_Simulation.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Define_Fault_Scripts.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Control_Fault_Scripts.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Manage_Technical_State.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Provide_Subsystem_Identification.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Provide_Health_State.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Manage_Operational_Mode.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Control_Battle_Override.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Manage_Subsystem_Parameters.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Provide_Subsystem_Services.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Manage_Mastership.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Register_Interest.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Provide_Area_with_Plot_Concentration.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Provide_Clutter_Assessment.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Allocate_Tracks_To_Stream.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Perform_Cued_Search.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Assess_Sensor_Track.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Filter_Plots.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Provide_Sensor_Characteristics.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Manage_Frequency_Usage.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Manage_Transmission_Sectors.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Control_Emissions.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Define_Test_Target_Scenario.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Test_Target_Facility.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Provide_Interference_Reports.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Provide_Nominal_Performance.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Provide_Performance_Assessment.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Provide_Jammer_Assessment.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Filter_Tracks.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Label_Tracks.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Provide_Sensor_Tracks.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Delete_Sensor_Track.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Receive_Track_Information.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Initiate_Track.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Manage_Tracking_Zones.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Provide_Projectile_Positional_Information.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Process_Target_Designation.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Support_Kill_Assessment.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Support_Surface_Target_Engagement.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Perform_Illumination.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Perform_Missile_Downlink.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Perform_Missile_Uplink.idl
https://www.omg.org/spec/OARIS/~~20230501~~20240802/IDL/Perform_Splash_Spotting.idl

OMG's Issue Reporting Procedure

All OMG specifications are subject to continuous review and improvement. As part of this process we encourage readers to report any ambiguities, inconsistencies, or inaccuracies they may find by completing the Issue Reporting Form listed on the main web page http://www.omg.org, under Documents, Report a Bug/Issue (http://www.omg.org/report_issue).

Table of Contents

Preface

OMG

Founded in 1989, the Object Management Group, Inc. (OMG) is an open membership, not-for-profit computer industry standards consortium that produces and maintains computer industry specifications for interoperable, portable, and reusable enterprise applications in distributed, heterogeneous environments. Membership includes Information Technology vendors, end users, government agencies, and academia.

OMG member companies write, adopt, and maintain its specifications following a mature, open process. OMG's specifications implement the Model Driven Architecture® (MDA®), maximizing ROI through a full-lifecycle approach to enterprise integration that covers multiple operating systems, programming languages, middleware and networking infrastructures, and software development environments. OMG's specifications include: UML® (Unified Modeling Language™); CORBA® (Common Object Request Broker Architecture); CWM™ (Common Warehouse Metamodel); and industry-specific standards for dozens of vertical markets.

More information on the OMG is available at http://www.omg.org/.

OMG Specifications

As noted, OMG specifications address middleware, modeling and vertical domain frameworks. All OMG Specifications are available from the OMG website at:
http://www.omg.org/spec

Specifications are organized by the following categories:

Business Modeling Specifications

Middleware Specifications

1 **CORBA/IIOP**

2 **Data Distribution Services**

3 **Specialized CORBA**

1. IDL/Language Mapping Specifications

2. Modeling and Metadata Specifications

1 **UML, MOF, CWM, XMI**

2 **UML Profile**

3. Modernization Specifications

4. Platform Independent Model (PIM), Platform Specific Model (PSM), Interface Specifications

1 **CORBAServices**

2 **CORBAFacilities**

OMG Domain Specifications

CORBA Embedded Intelligence Specifications

CORBA Security Specifications

Signal and Image Processing

All of OMG's formal specifications may be downloaded without charge from our website. (Products implementing OMG specifications are available from individual suppliers.) Copies of specifications, available in PostScript and PDF format, may be obtained from the Specifications Catalog cited above or by contacting the Object Management Group, Inc. at:

OMG Headquarters
109 Highland Avenue
Needham, MA 02494
USA
Tel: +1-781-444-0404
Fax: +1-781-444-0320
Email: *pubs@omg.org*

Certain OMG specifications are also available as ISO standards. Please consult http://www.iso.org

Typographical Conventions

The type styles shown below are used in this document to distinguish programming statements from ordinary English. However, these conventions are not used in tables or section headings where no distinction is necessary.
Times/Times New Roman - 10 pt.: Standard body text

Helvetica/Arial - 10 pt. Bold: OMG Interface Definition Language (OMG IDL) and syntax elements.

Courier/Courier New - 10 pt. Bold: Programming language elements.

Helvetica/Arial - 10 pt: Exceptions

1 Scope

This specification defines the interface between the Combat Management System (CMS) and a Sensor system (especially a Radar system) within a modular combat system architecture for naval platforms. It is structured to align with the objective of dividing the interface into three categories, namely subsystem services (interfaces applicable to any module within a combat system), sensor services (interfaces applicable to any sensor component within a combat system) and radar services (interfaces applicable to complex radar components within a combat system).

Version 1.0 of the specification addressed the scope required for radar integration specifically.

Version 2.0 of the specification expanded the scope of the sensor services such that it provides the necessary interfaces for the integration of other combat system sensors and subsystems.

Version 3.0 of the specification expanded the scope of the sensor services such that plots and other measurements can be shared between cooperating platform units within a joint operation.

2 Conformance

In order to support utilization by a range of sensors from simple navigation radars and electro-optic systems to complex multi-function radars, sonars or electronic warfare systems the RFP defines the following compliance levels:

1. Level 1
   The simplest sensor operation providing just plots and tracks
2. Level 2
   Basic sensor operation, but a complete interface supporting control and essential system configuration for a combat system context
3. Level 3A
   In addition to basic operation (level 2), interfaces for training support
4. Level 3B
   In addition to basic operation (level 2), full system configuration interfaces
5. Level 3C
   In addition to basic operation (level 2), the full track and plot reporting interfaces
6. Level 3D
   In addition to basic operation (level 2), the engagement support interface
7. Level 3E
   In addition to basic operation (level 2), the advanced radar interfaces
8. Level 3F
   In addition to basic operation (level 2), compliance with C2INav
9. Level 3G
   In addition to basic operation (level 2), compliance with METOC (To be defined in a future version of OARIS)
10. Level 3H
    In addition to basic operation (level 2), the full parameter measurement and identification assessment interfaces
11. Level 3I
    In addition to basic operation (level 2), the interfaces to support cooperative plot sharing

Radars conforming to this specification shall indicate which compliance levels are supported. The following options are possible:

1 Level 1

2 Level 2

3 Any combination of levels 3A to 3E (in addition to level 2)

Further detail on service interfaces contained within each conformance level is presented in section 7.2.

3 Normative References

The following normative documents contain provisions which, through reference in this text, constitute provisions of this specification. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply.

Title (Acronym)	Version / Date	Organization	Reference / URL
Data Distribution Service (DDS)	1.4 / March 2015	OMG	formal/2015-04-10 www.omg.org/spec/DDS
Interface Definition Language (IDL)	4.2 / January 2018	OMG	formal/2018-01-05 www.omg.org/spec/IDL
Extended View of Time (EVOT)	2.0 / August 2008	OMG	formal/2008-08-01 www.omg.org/spec/EVOT
Unified Modeling Language (UML)	2.0 / July 2005	OMG	formal/05-07-04 www.omg.org/spec/UML
XML Metadata Interchange (XMI)	2.1 / December 2007	OMG	formal/07-12-01 www.omg.org/spec/XMI
Meta Object Facility (MOF)	2.0 / January 2006	OMG	formal/06-01-01 www.omg.org/spec/MOF
Graph Query Language (GraphQL)	June 2018	Facebook	spec.graphql.org/June2018
NATO Tactical Data Exchange – Link 16	Edition 6	NATO	STANAG 5516
Joint C3 Information Exchange Data Model (JC3IEDM)	v3.1.4	NATO	STANAG 5525
NATO Joint Military Symbology (APP-6(B))	June 2008	NATO	
NATO Joint Military Symbology (APP-6(C))	May 2011	NATO	
Common Warfighting Symbology (MIL-STD-2525C)	November 2008	DoD	MIL-STD-2525C
Joint Military Symbology (MIL-STD-2525D)	June 2014	DoD	MIL-STD-2525C
Distributed Interactive Simulation (DIS)	1A / 1998	IEEE	1278.1A
World Geodetic System 1984 (WGS-84)	N/A	US National Geospatial Intelligence Agency	https://earth-info.nga.mil/?dir=wgs84&action=wgs84

4 Terms and Definitions

For the purposes of this specification, the following terms and definitions apply.

- AAW (Anti-Air Warfare)
- AB (Architecture Board)
- API (Application Programming Interface)
- APP (Allied Procedural Publication)
- ASuW (Anti-Surface Warfare)
- ATC (Air Traffic Control)
- BC (Business Committee)
- BCQ (Business Committee Questionnaire)
- BoD (Board of Directors)
- CMS (Combat Management System)
- DDS (Data Distribution Service)
- EVOT (Enhanced View of Time)
- FTF (Finalization Task Force)
- IDL (Interface Definition Language)
- IEC (International Electrotechnical Commission)
- IFF (Interrogation, Friend or Foe)
- IPR (Intellectual Property Right)
- ISO (International Organization for Standardization)
- LOI (Letter of Intent)
- MDA (Model Driven Architecture)
- METOC (Meteorological and Oceanographic)
- MOF (Meta Object Facility)
- NNSI (Naval Navigation System Interface)
- NS (Naming Service)
- OARIS (Open Architecture Radar Interface Standard)
- OASIS (Organization for Advancement of Structured Information Standards)
- OCL (Object Constraint Language)
- ODF (Open Document Format)
- OMA (Object Management Architecture)
- OMG (Object Management Group)

- PIM (Platform Independent Model)
- PSM (Platform Specific Model)
- P&P (Policies and Procedures of the OMG Technical Process)
- RFC (Request For Comment)
- RFP (Request For Proposal)
- RTF (Revision Task Force)
- SIDC (Symbol Identification Code)
- SOA (Service Oriented Architecture)
- SoaML (Service oriented architecture Modeling Language)
- SOLAS (Safety Of Life At Sea)
- TC (Technology Committee)
- TF (Task Force)
- TOS (Trading Object Service)
- UML (Unified Modeling Language)
- XMI (XML Metadata Interchange)
- XML (eXtensible Markup Language)

5 Symbols

No special symbols are introduced in this specification.

6 Additional Information

6.1 Acknowledgements

The following companies submitted this specification:

1 BAE Systems

6.2 Specification Generation

The specification is captured as a UML version 2.1 model, with this document being automatically generated as a report from the model.

1

7 Open Architecture Radar Information Specification

7.1 Introduction

7.1.1 Background

A Combat System on a naval warship (or platform) typically consists of, amongst other things, a Combat Management System (CMS) interfacing with a number of sensors and communication systems (e.g. DataLink network), together providing the user with a tactical picture of all the real world entities that have been detected. These are then passed to other ship systems (e.g. comms and weapon systems) to support ongoing warfighting activities.

In OARIS, sensors and these other ship systems are generalized as subsystems. OARIS partitions its data model and services into abstraction layers that are applicable to subsystems (most general), sensor and radars (most specific).

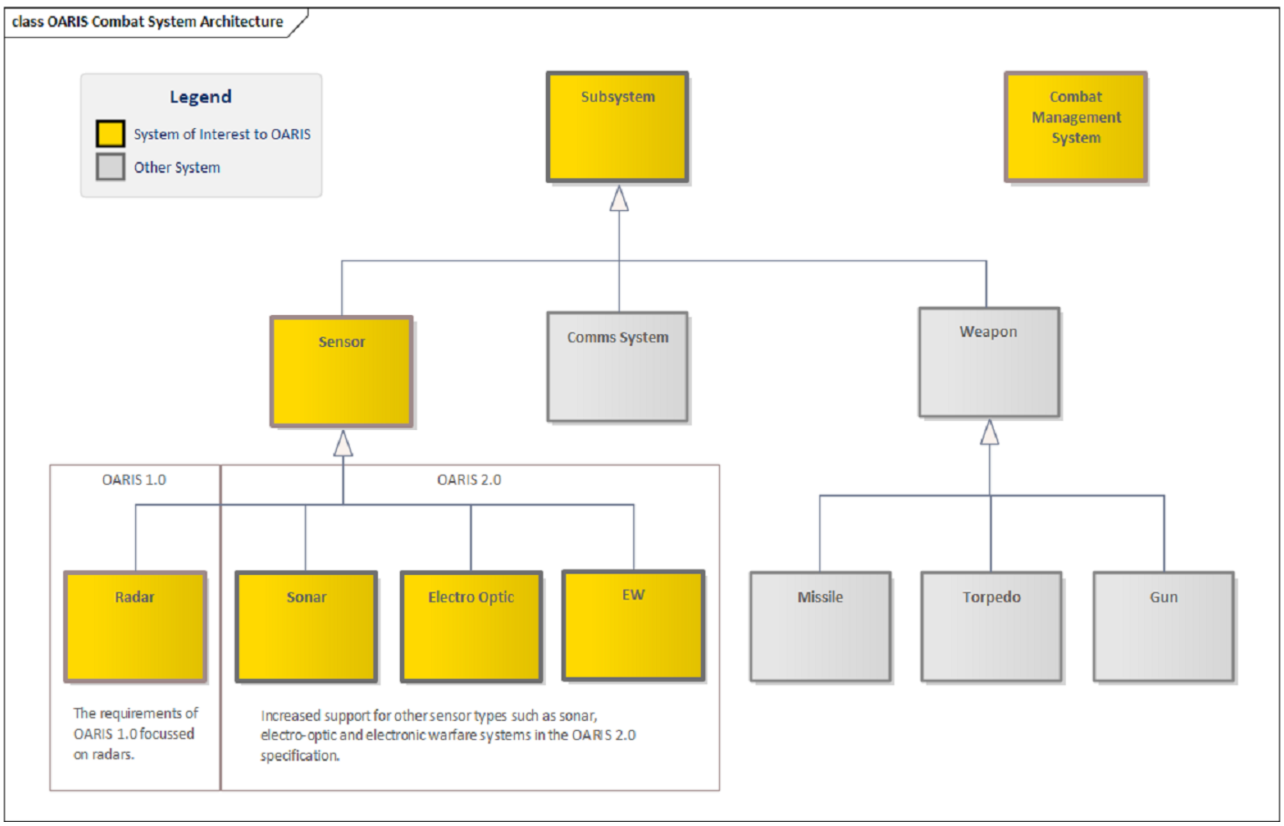


Figure 1.1 - The OARIS specification is applicable to Sensor systems within the Combat System Architecture

Sensors typically operate by recording detections of whatever physical property they are sensing (e.g. acoustic or electromagnetic events). These detections are called plots. A sensor may analyze the plots it is detecting over a period of time and make decisions about whether each plot has come from a real world object of interest, or whether the plot has been received as clutter from the environment (e.g. returns from the crests of waves). If the sensor has confidence that a number of received plots correspond to a real world object then the sensor will form a track based on those plots which is then sent to the CMS; a track being a sensor-view representation of a real-world object over time. The CMS maintains a track list which in general has contributions from all the sensor on the platform.

Where a platform is working as part of a task group then historically the platform is able to share tracks from its track list with other platforms via a DataLink network. Version 3.0 of OARIS expands the scope such that it provides the necessary interfaces for coherent sharing of plots and other measurements between cooperating platform units in a joint operation. Version 2.0 of OARIS extended the scope of services and the data model to cover the functionality and capabilities of sensors in addition to radars, whilst version 1.0 of OARIS focused on radars more specifically.

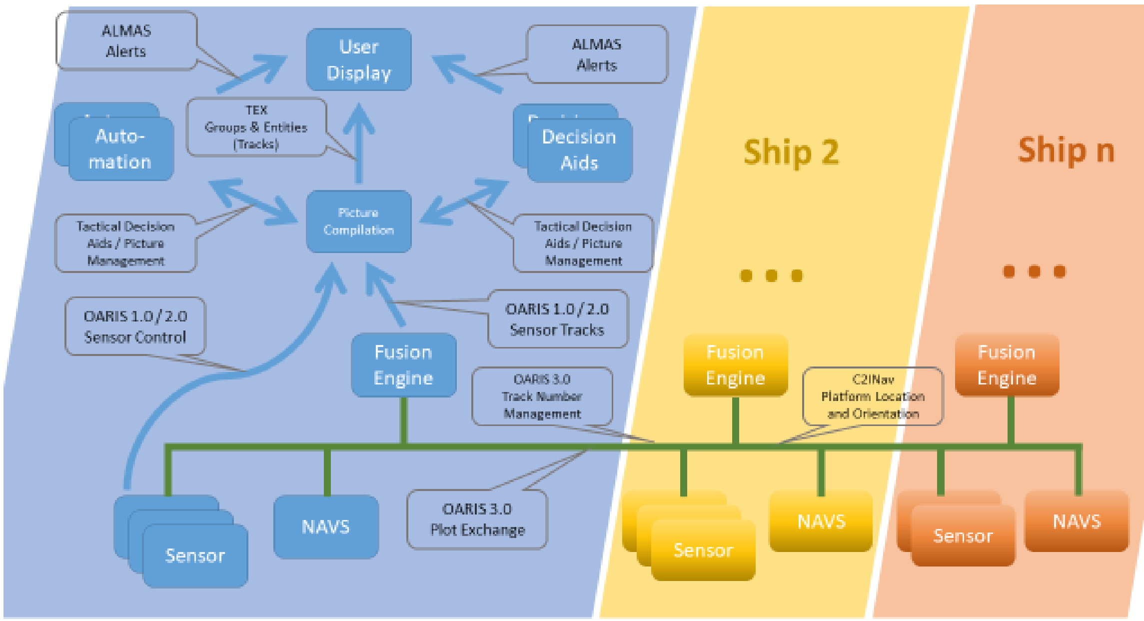


Figure 1.2 - The OARIS 3.0 specification in context with other C4I specifications and other platforms (e.g. ships) implementing the standard

7.1.2 Section Structure

This section of the document is organized as follows:

7.1 (this section) Introduces the OARIS specification and gives some context to the use of OARIS within a naval environment.

7.2 Provides an overview of how the various interfaces (later described in sections 7.7 to 7.9) are used by nominal components to achieve a particular level of compliance with the OARIS specification

7.3 Identifies all the common data types used within the specification

7.4 Identifies all the data types that are applicable to the Subsystem domain interfaces (described in 7.7)

7.5 Identifies all the data types that are applicable to the Sensor domain interfaces (described in 7.8)

7.6 Identifies all the data types that are applicable to the Radar domain interfaces (described in 7.9)

7.7 Identifies all the interfaces that are applicable to the Subsystem domain

7.8 Identifies all the interfaces that are applicable to the Sensor domain

7.9 Identifies all the interfaces that are applicable to the Radar domain

7.2 Document Structure

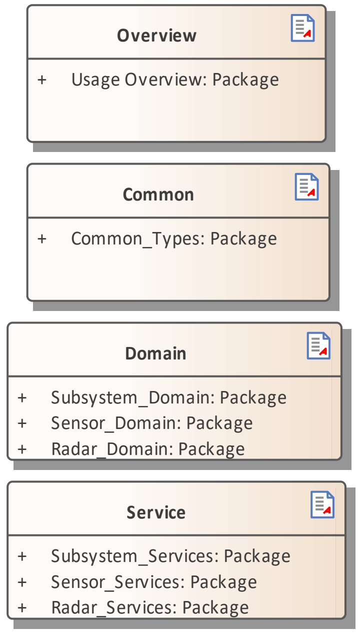


Figure 7.1 -Specification Master (Documentation diagram)

This specification is presented as:

- An overview of how the services are used to achieve levels of conformance to the standard
- Common data types used throughout
- Domain specific data types for the three domains (Subsystem, Sensor and Radar)

Service interfaces for the three domains (Subsystem, Sensor and Radar)

7.3 Usage Overview

Parent Package: Analysis Model (PIM)

OARIS defines compliance levels as follows:

- Level 1: A simple radar which provides just plots and tracks
- Level 2: Basic radar operation, but a complete interface supporting control and essential system configuration for a combat system context
- Level 3A: In addition to basic operation (level 2), interfaces for training support
- Level 3B: In addition to basic operation (level 2), full system configuration interfaces
- Level 3C: In addition to basic operation (level 2), the full track and plot reporting interfaces
- Level 3D: In addition to basic operation (level 2), the engagement support interface
- Level 3E: In addition to basic operation (level 2), the advanced radar interfaces
- Level 3F (compliance with C2INav) and Level 3G (compliance with METOC). are outside the scope of this response
- Level 3H In addition to basic operation (level 2), the full parameter measurement and identification assessment interfaces.

Sensors conforming to this specification shall indicate which compliance levels are supported. The following options are possible:

- Level 1
- Level 2
- Any combination of levels 3A to 3E or 3H (in addition to level 2)

In order to comply with the specification levels the following respective interfaces shall be supported in full, with the exception of level 3C where at least one of the environment types (Space/Air/Land/Surface) shall be supported and appropriately qualified, e.g. level 3C Air and Surface:
This section continues below with activity and component diagrams that show how the interfaces relate to achieve the difference compliance levels. The activity diagrams capture pre-requisites for interface usage, whilst the component diagrams illustrate non-normative functionality enabled by the interfaces within a compliance level. The component diagrams contain non-normative components representing subsets of a typical functional decomposition of the Subsystem and CMS interface abstractions used by the normative sections of this specification. The interfaces entailed by each conformance level are defined in sections 7.7 to 7.9 describing the subsystem, sensor and radar services.

7.3.1 Compliance Level 1

Parent Package: Usage Overview
The Compliance Level 1 required interfaces are:
- Register Interest
- Track Reporting
- Plot Reporting

This compliance level is aimed at the simplest integration use cases and provides an entry-point for initial adoption of the standard by implementers.

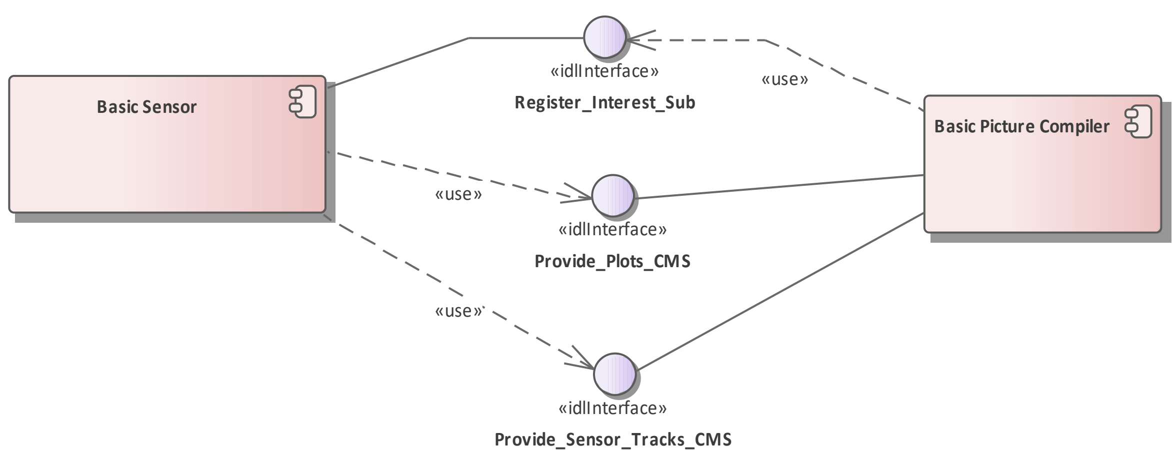


Figure 7.2 Compliance Level 1 (Component diagram)

This component diagram shows the interfaces realized and used by CMS and Subsystem components integrated at compliance level 1.

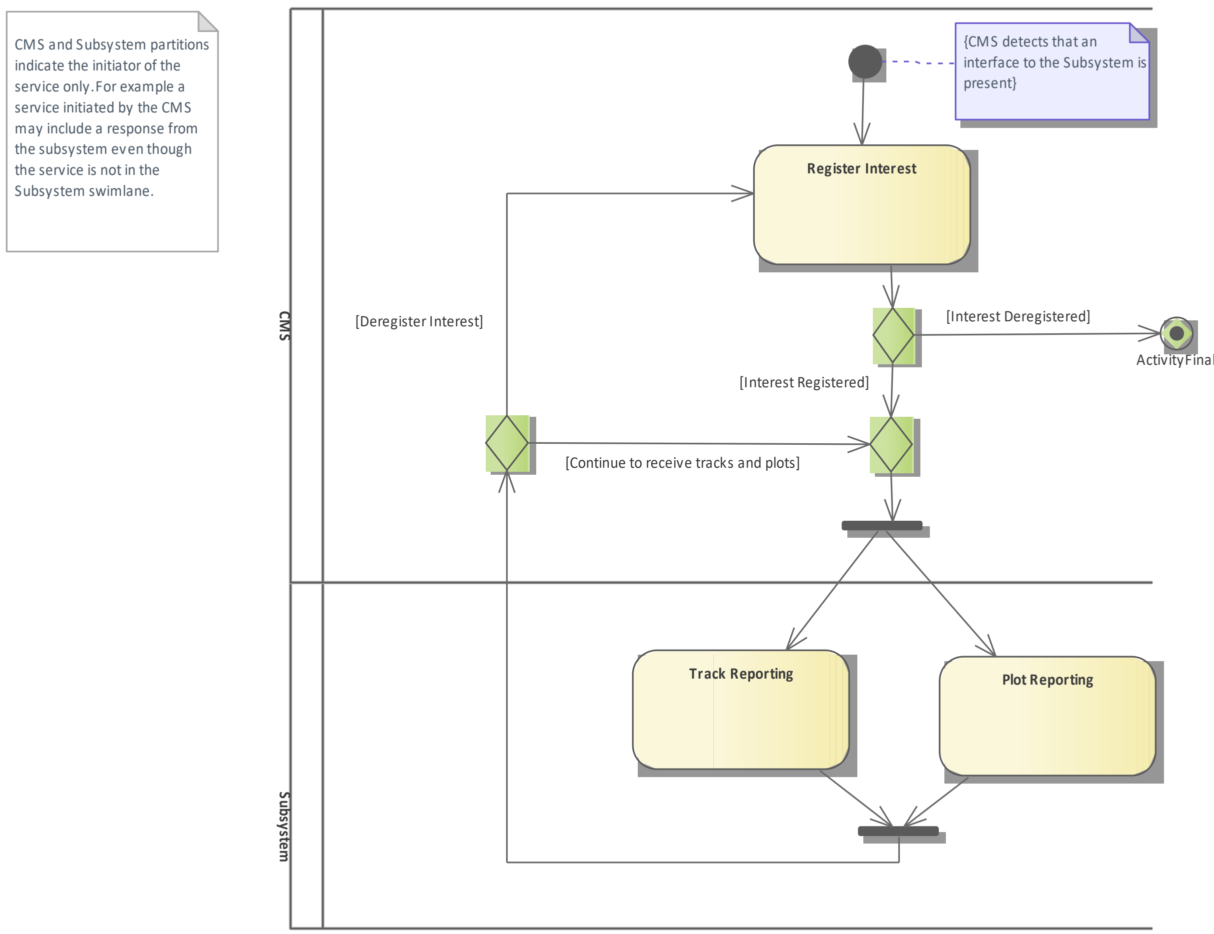


Figure 7.3 Compliance Level 1 (Activity diagram)

For compliance level 1, the radar powers up and commences track and plot reporting either without intervention or using an out of scope facility, such as a maintainer interface. The CMS detects the presence of the interface, registers interest then processes the incoming track and plot streams.

7.3.1.1 Basic Picture Compiler

A non-normative minimal example of the picture compilation function realizing the abstraction of a CMS.

7.3.1.2 Basic Sensor

A non-normative minimal example of a sensor realizing the abstraction of a Subsystem.

7.3.2 Compliance Level 2

Parent Package: Usage Overview

The Compliance Level 2 required interfaces are:

- Control Interface Connection
- Provide Subsystem Identification
- Provide Subsystem Services
- Manage Subsystem Parameters

- Provide Health State
- Manage Mastership
- Manage Technical State
- Exchange Heartbeat
- Register Interest
- Track Reporting
- Plot Reporting
- Manage Operational Mode
- Manage Tracking Zones
- Manage Frequency Usage
- Manage Transmission Sectors
- Control Battle Override
- Control Emissions

This compliance level supports core functionality required for operational usage in a fully integrated combat system.

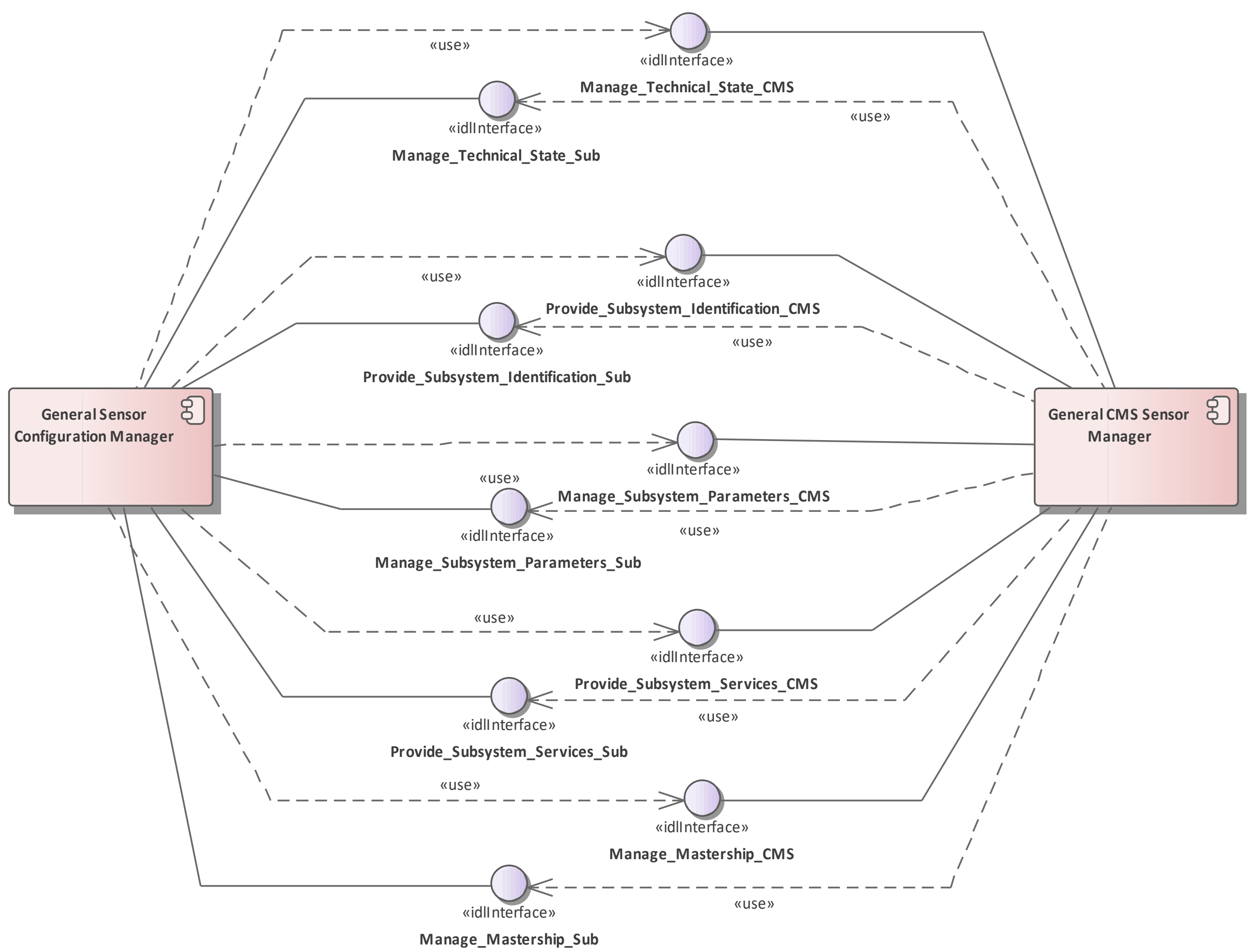
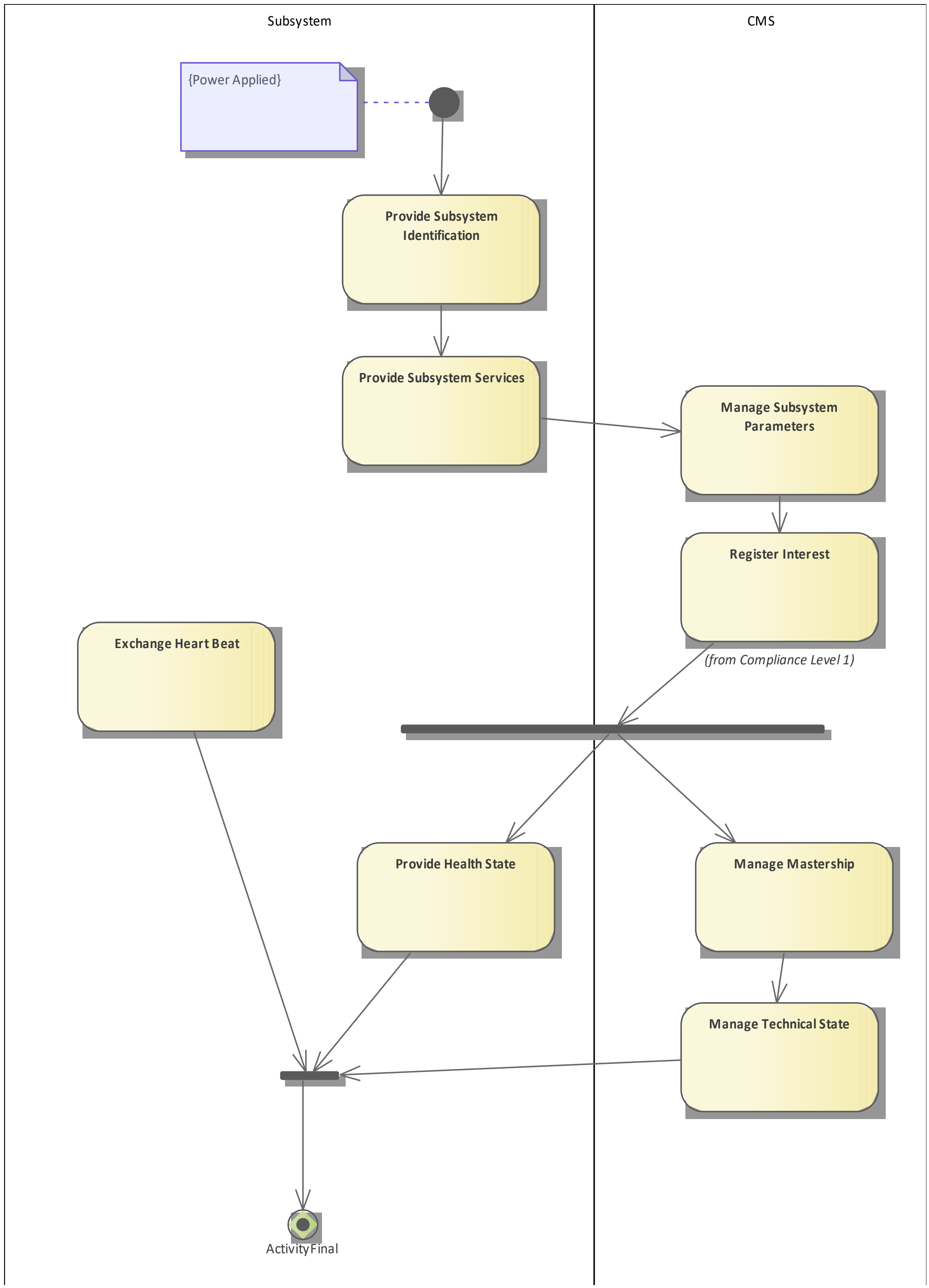


Figure 7.4 Compliance Level 2 - Initialization (Component diagram)

This component diagram shows the interfaces realized and used by CMS and Subsystem components, to achieve initialization, integrated at compliance level 2.



Subsystem
CMS
{Power Applied}
Provide Subsystem Identification
Provide Subsystem Services
Manage Subsystem Parameters
Register Interest
(from Compliance Level 1)
Exchange Heart Beat
Provide Health State
Manage Mastership
Manage Technical State
ActivityFinal

Figure 7.5 Compliance Level 2 - Initialization (Activity diagram)

For compliance level 2 a more versatile startup sequence is supported, with the subsystem and CMS going through a negotiation and configuration stage followed by more detailed interface control and reporting, including management of reversionary modes.

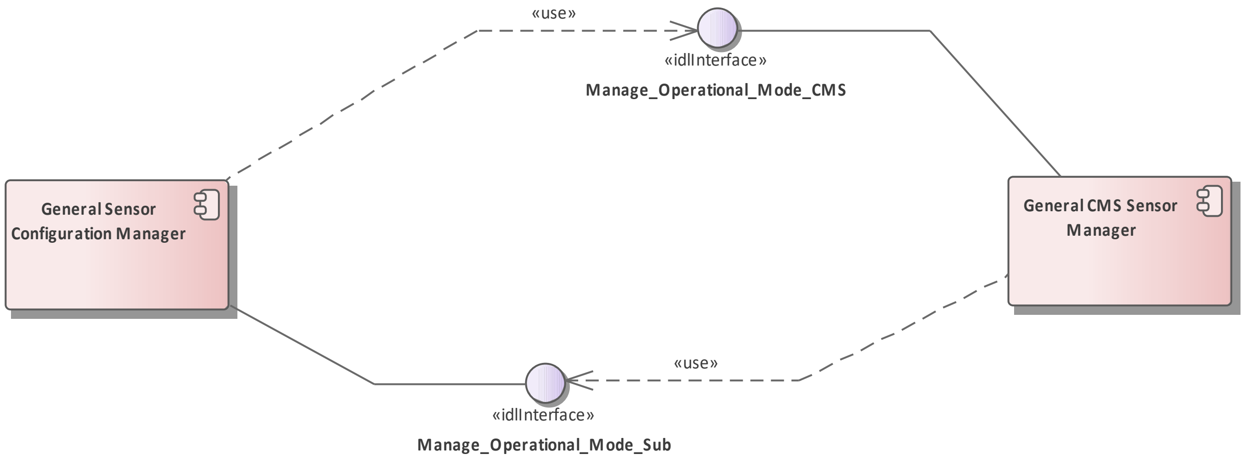


Figure 7.6 Compliance Level 2 - Operational Mode (Component diagram)

This component diagram shows the interfaces realized and used by CMS and Subsystem components, to manage the operational mode of the subsystem, integrated at compliance level 2.

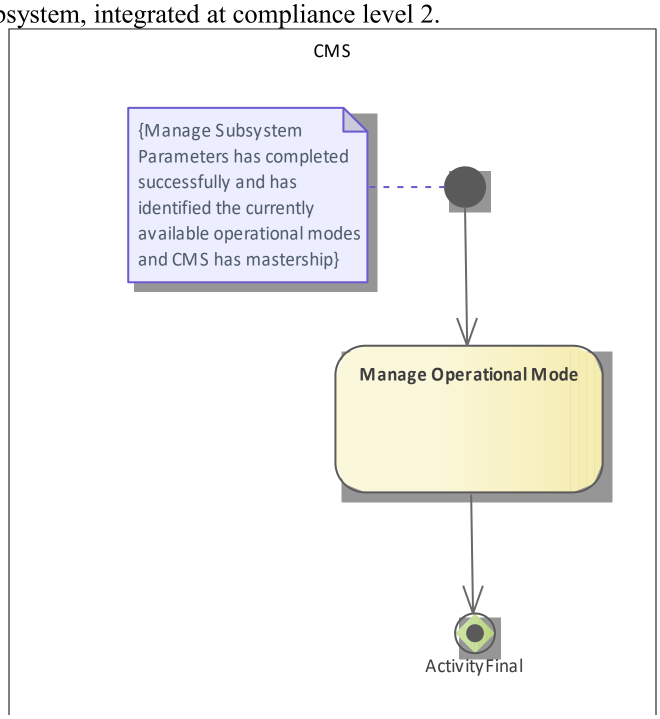


Figure 7.7 Compliance Level 2 - Operational Mode (Activity diagram)

Level 2 continues to manage the operational mode while the CMS has mastership.

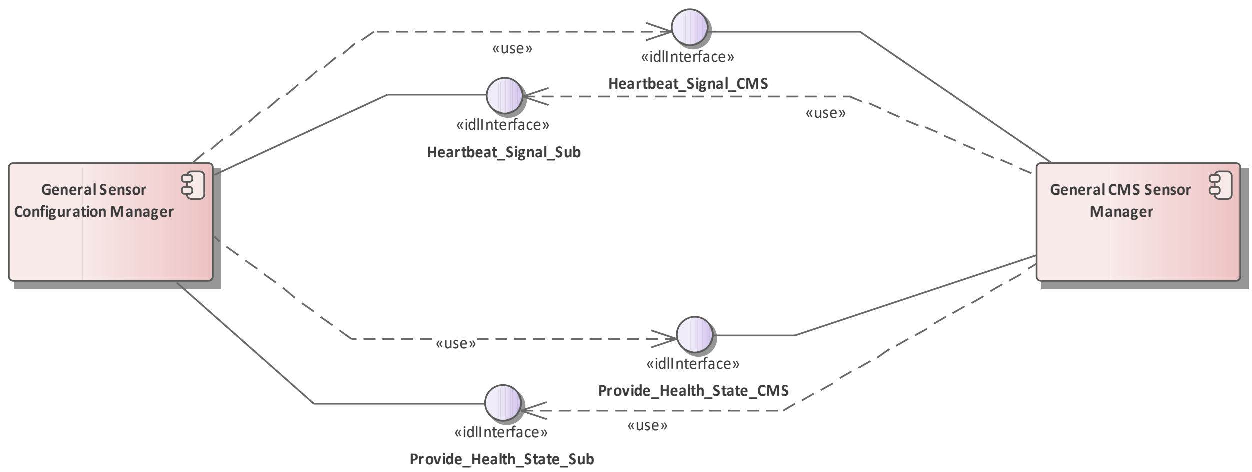


Figure 7.8 Compliance Level 2 - Status Monitoring (Component diagram)

This component diagram shows the interfaces realized and used by CMS and Subsystem components, to achieve status monitoring of the subsystem, integrated at compliance level 2.

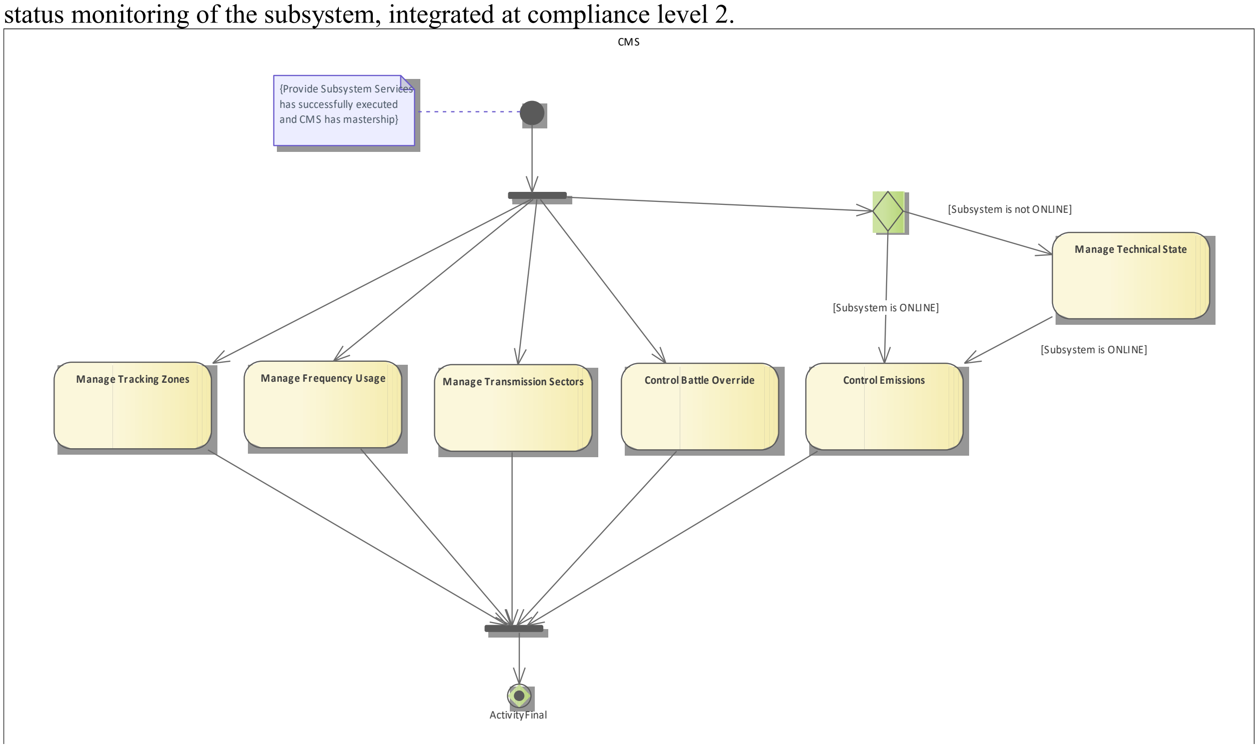


Figure 7.9 Compliance Level 2 - Subsystem Setup (Activity diagram)

Level 2 caters for continuous management of sensor configuration when the CMS has mastership.

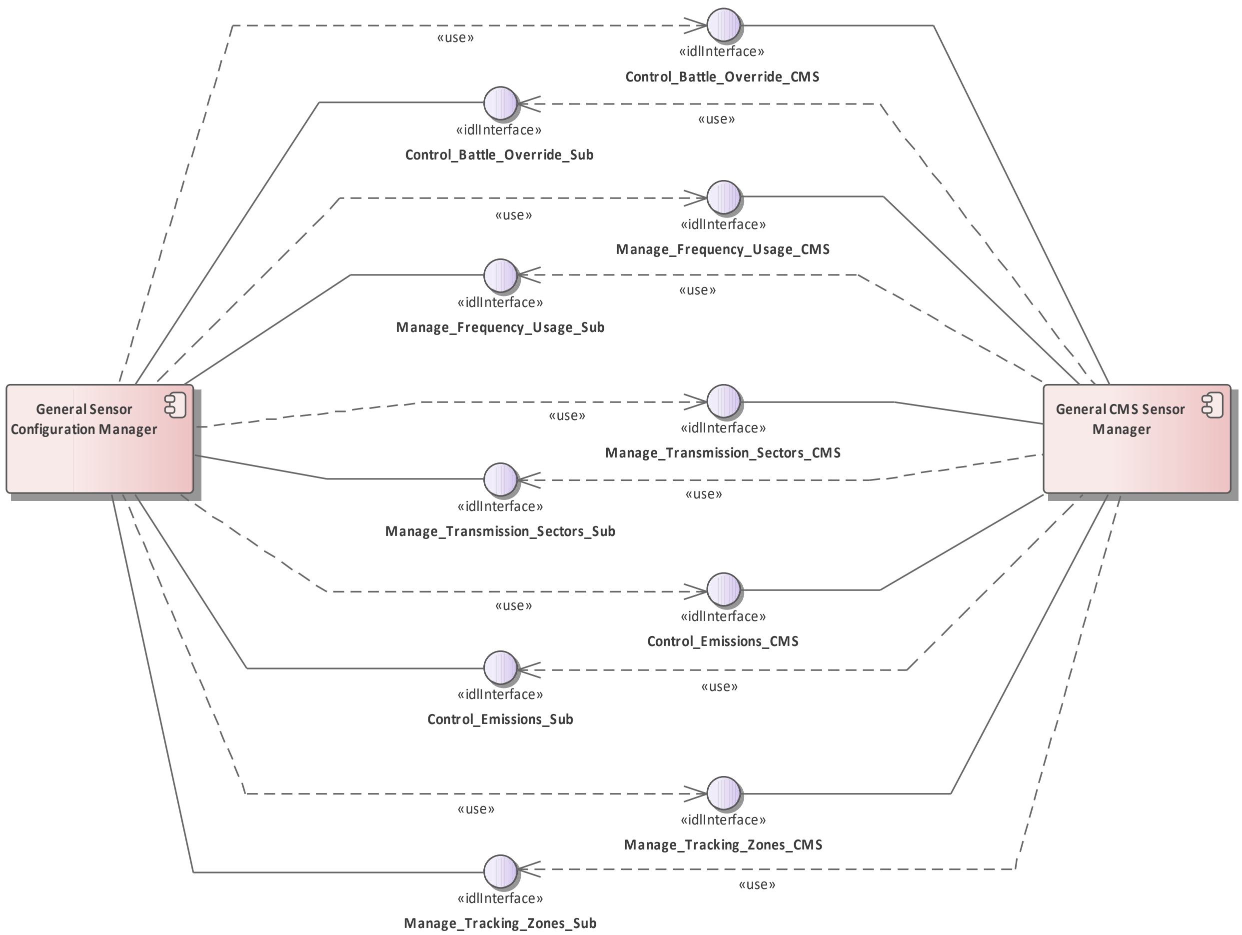


Figure 7.10 Compliance Level 2 - Subsystem Setup (Component diagram)

This component diagram shows the interfaces realized and used by CMS and Sensor Subsystem components, to achieve subsystem setup, integrated at compliance level 2. The CMS sets up the sensor subsystem such that it has the required configuration to perform the necessary operational role for the current task or mission assigned to the platform.

7.3.2.1 General CMS Sensor Manager

A non-normative example of sensor management function within a CMS; contains functionality to support system users in configuring combat system sensors to support their tasking and mission objectives.

7.3.2.2 General Sensor Configuration Manager

A non-normative example of a sensor configuration management function that allows the sensor to be configured to best perform the tasks to which it is allocated.

7.3.3 Compliance Level 3A

Parent Package: Usage Overview

The Compliance Level 3A required interfaces are:

- Define Test Target Scenario

- Define Fault Scripts
- Control Simulation
- Control Fault Script
- Control Test Target Facility
- Control Recording
- Control Replay
- Provide Simulation Data
- Perform Offline Test

This compliance level supports specialized functionality relating to simulation, online test and analysis. This level is applicable to all types of subsystem.

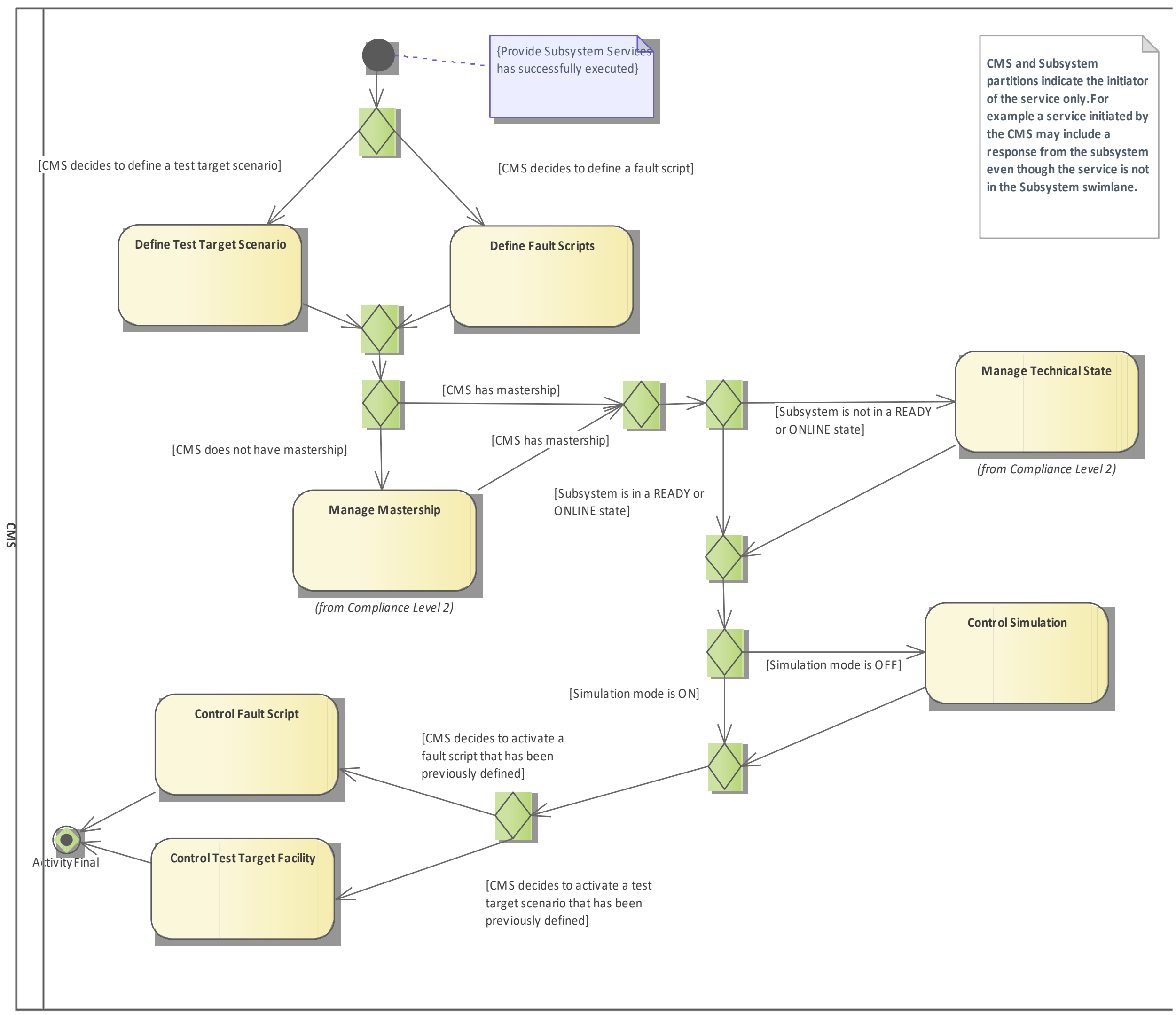


Figure 7.11 Compliance Level 3A - Fault Scripts and Test Targets (Activity diagram)

Level 3 provides for the simulation of faults and targets for test and training purposes.

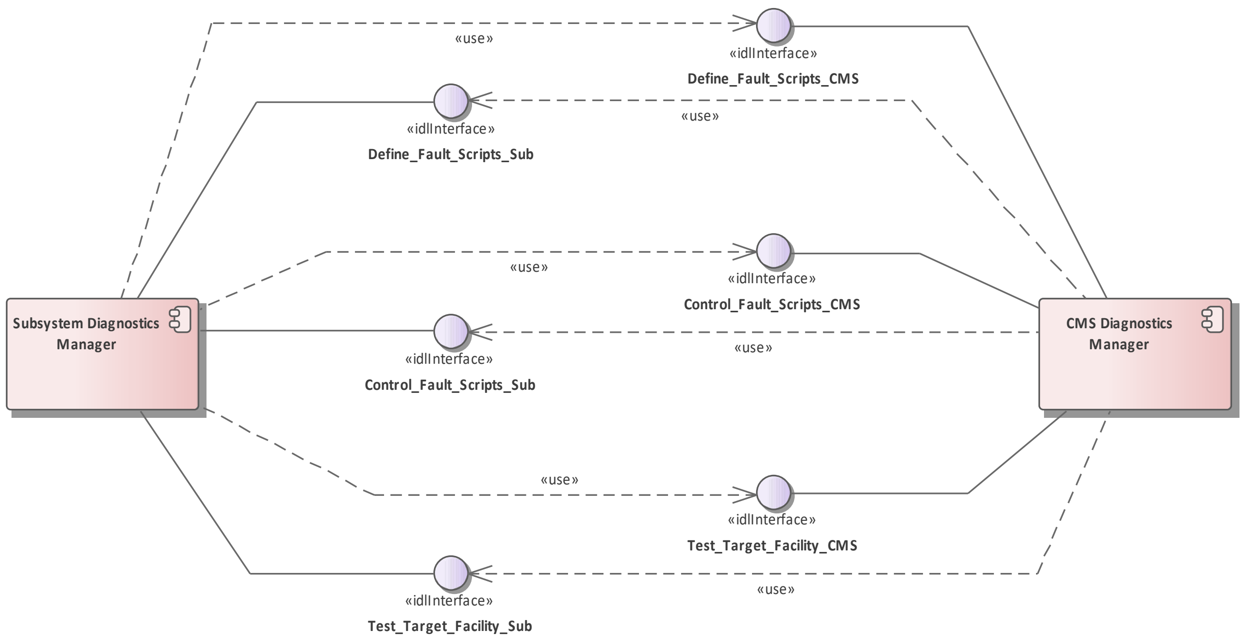


Figure 7.12 Compliance Level 3A - Fault Scripts and Test Targets (Component diagram)

This component diagram shows the interfaces realized and used by CMS and Subsystem components, to support online diagnostic analysis, integrated at compliance level 3A.

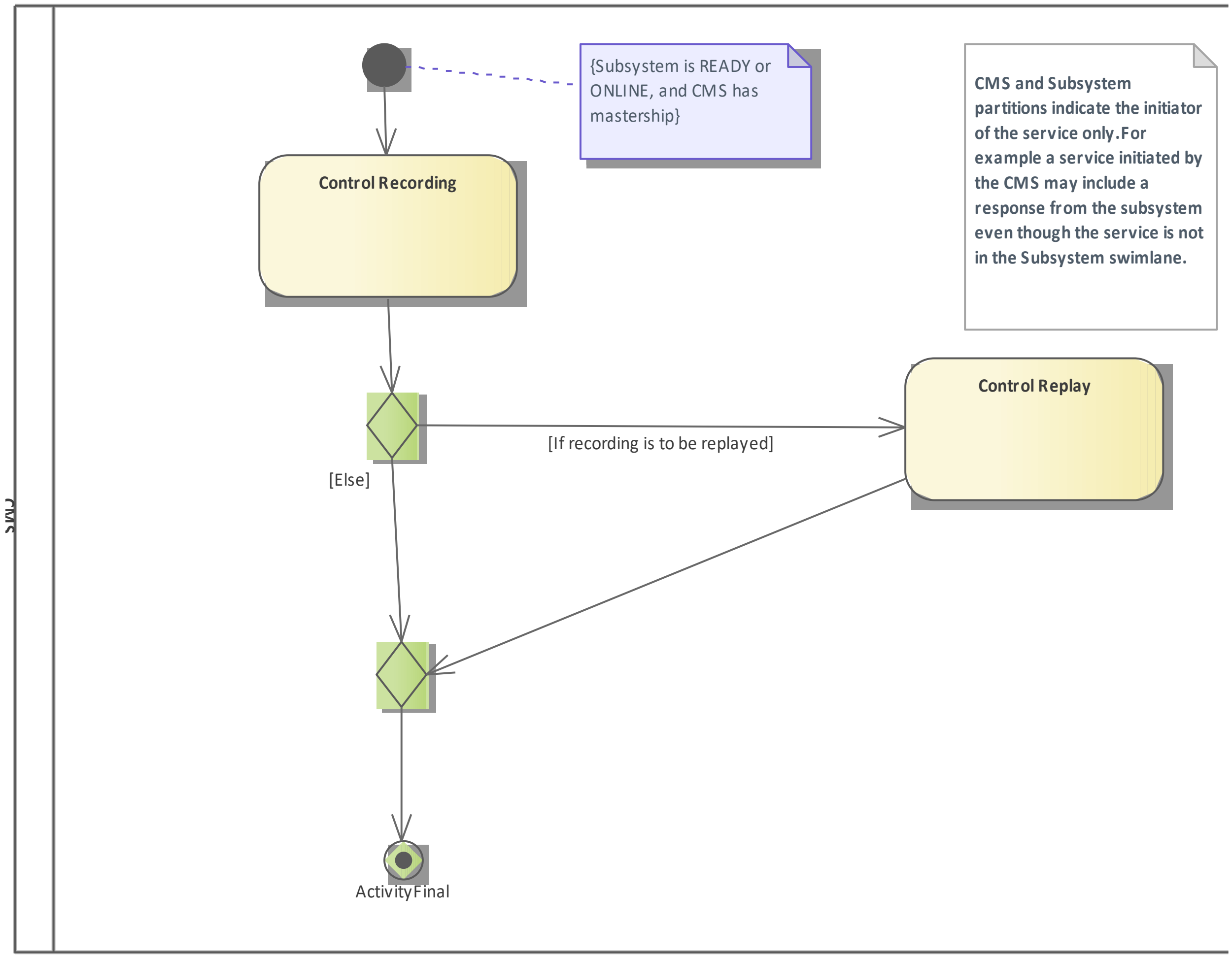


Figure 7.13 Compliance Level 3A - Recording/Replay (Activity diagram)

Recording and replay facilities support recording and replay of subsystem parameters for the purposes of training and/or post exercise review.

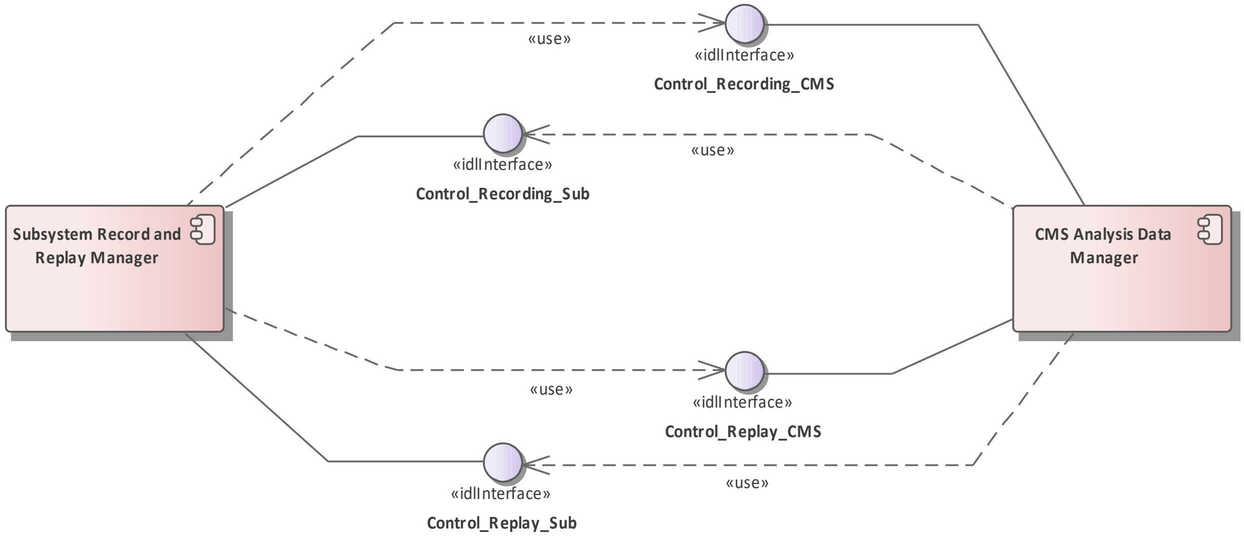


Figure 7.14 Compliance Level 3A - Recording/Replay (Component diagram)

This component diagram shows the interfaces realized and used by CMS and Subsystem components, to support record and replay for analysis, integrated at compliance level 3A.

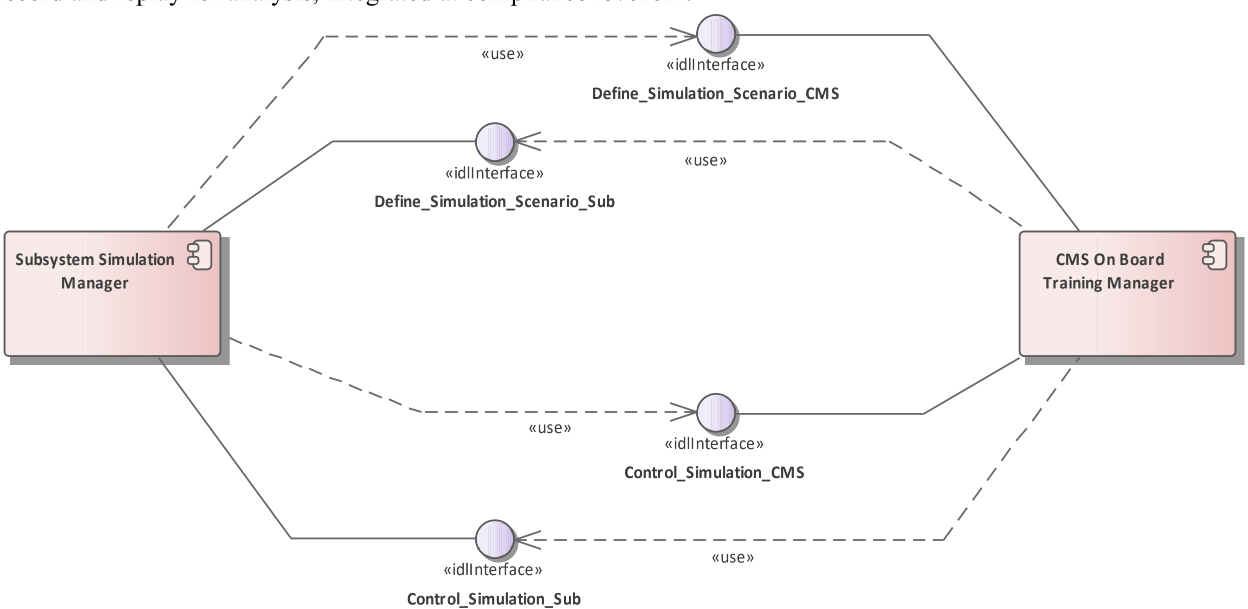


Figure 7.15 Compliance Level 3A - Simulation (Component diagram)

This component diagram shows the interfaces realized and used by CMS and Subsystem components, to support simulation, integrated at compliance level 3A.

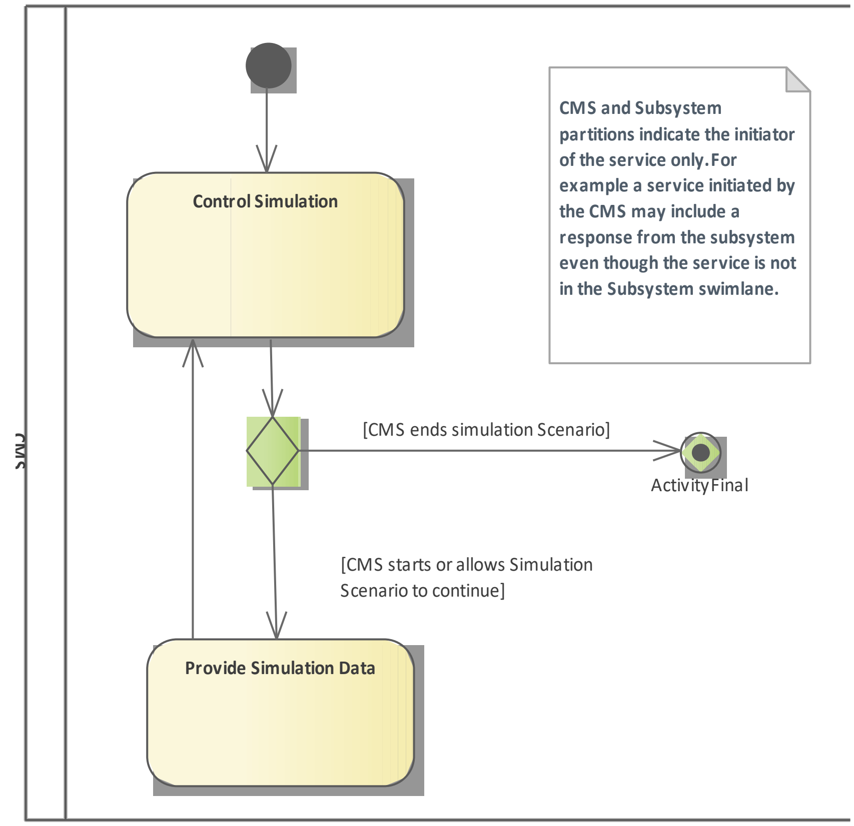


Figure 7.16 Compliance Level 3A - Simulation (Activity diagram)

The simulation interfaces are used to support training.

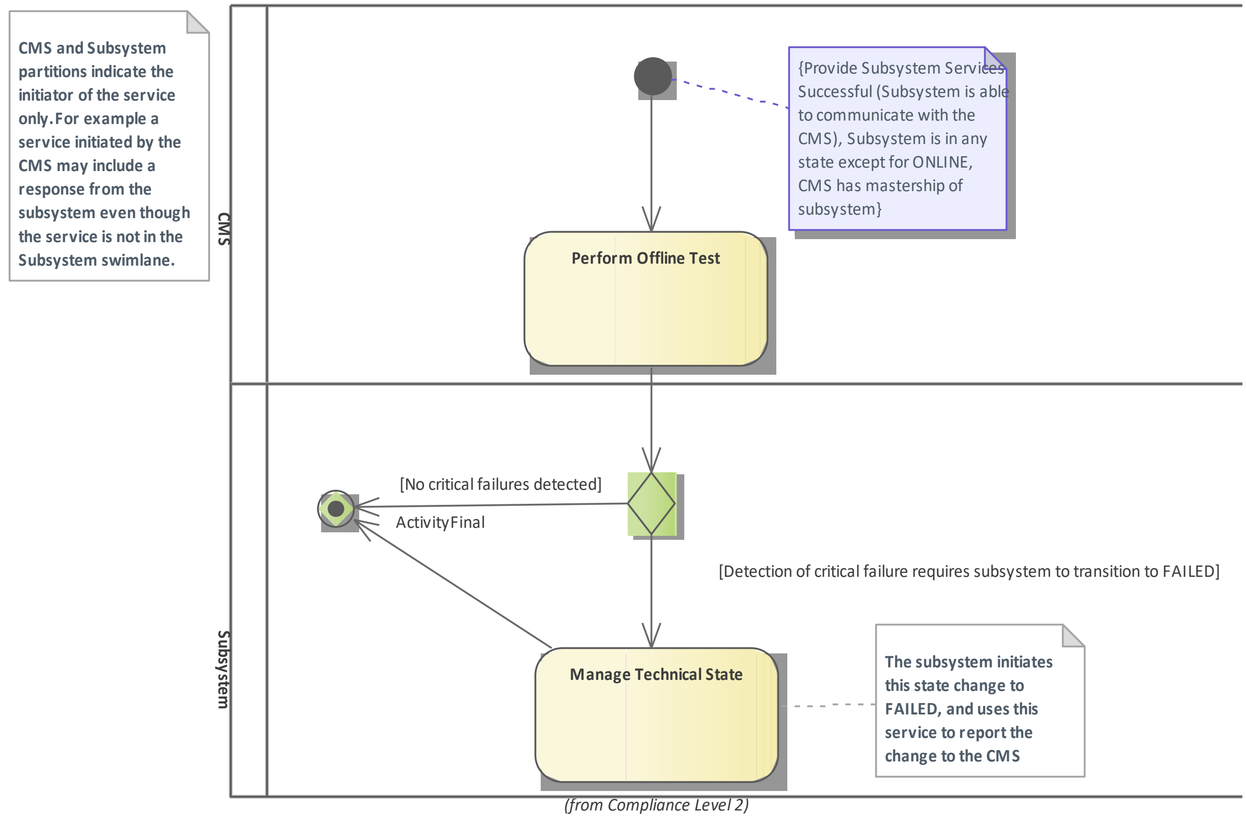


Figure 7.17 Compliance Level 3A - Perform Offline Test (Activity diagram)

Offline test provides a mechanism for diagnosing subsystem failures, after which the subsystem's technical state is adjusted accordingly.

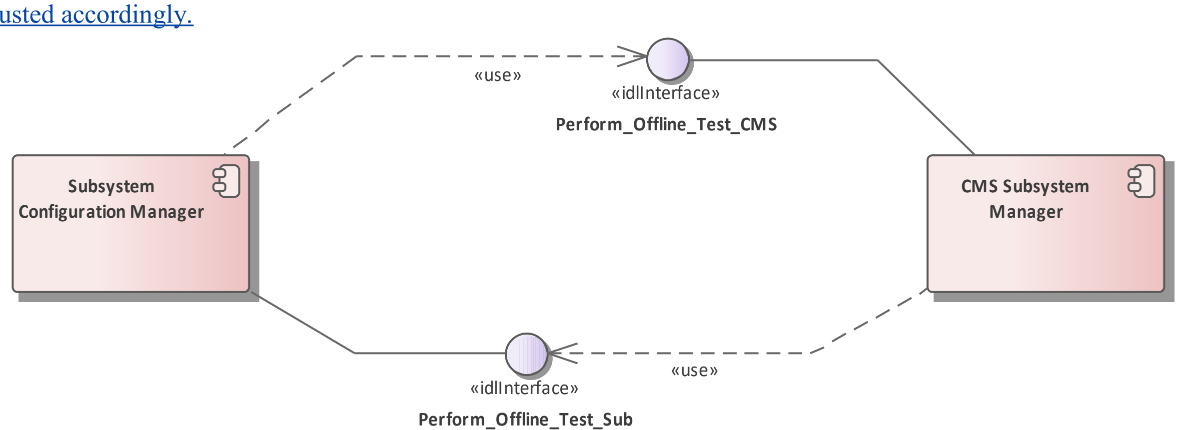


Figure 7.18 Compliance Level 3A - Perform Offline Test (Component diagram)

This component diagram shows the interfaces realized and used by CMS and Subsystem components, to perform offline tests, integrated at compliance level 3A.

7.3.3.1 CMS Analysis Data Manager

A non-normative example of an analysis data management function within a CMS. This component would support CMS users in learning from tasks recently undertaken.

7.3.3.2 CMS Diagnostics Manager

A non-normative example of a diagnostics management function within a CMS. This component would enable CMS users to investigate potential faults in the combat system.

7.3.3.3 CMS On Board Training Manager

A non-normative example of a simulation management function within a CMS. A plan rehearsal function would be another example.

7.3.3.4 Subsystem Diagnostics Manager

A non-normative example of a diagnostic function within a subsystem. Such a function enables the generation of diagnostic tests on the subsystem's other components.

7.3.3.5 Subsystem Record and Replay Manager

A non-normative example of a combined record and replay function within a subsystem to manage recording and later replay of the data the subsystem generates.

7.3.3.6 Subsystem Simulation Manager

A non-normative example of a simulation function within a subsystem enabling the subsystem to take part in federated simulated operations such as on-board training.

7.3.3.7 CMS Subsystem Manager

A non-normative example of a CMS function to manage the state of subsystems in the combat system.

7.3.3.8 Subsystem Configuration Manager

A non-normative example of a subsystem function to manage its configuration and state.

7.3.4 Compliance Level 3B

Parent Package: Usage Overview
The Compliance Level 3B required interfaces are:

- Shutdown
- Restart
- Startup
- Manage Physical Configuration
- ~~Perform Offline Test~~
- Receive Encyclopedic Data

This compliance level supports specialized configuration and state management of the subsystem (and applies to subsystems in general).

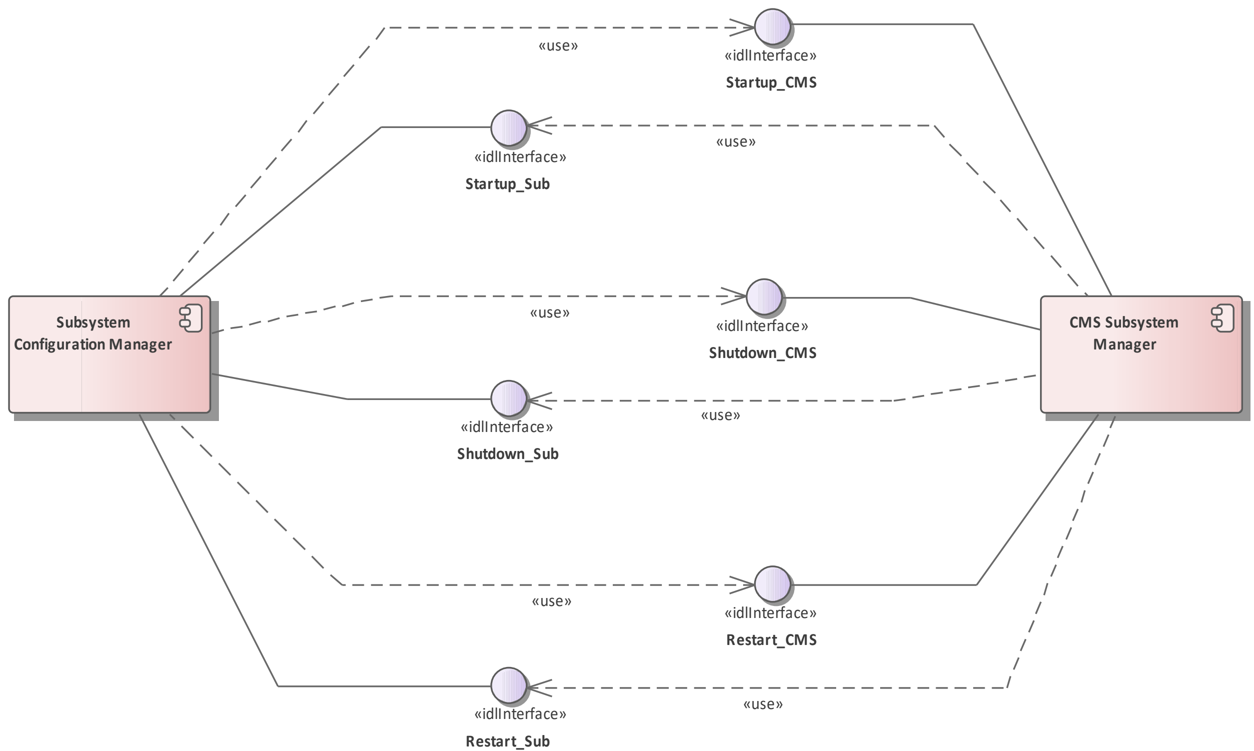


Figure 7.19 Compliance Level 3B - Macro State Management (Component diagram)

This component diagram shows the interfaces realized and used by CMS and Subsystem components, to achieve macro state management, integrated at compliance level 3B.

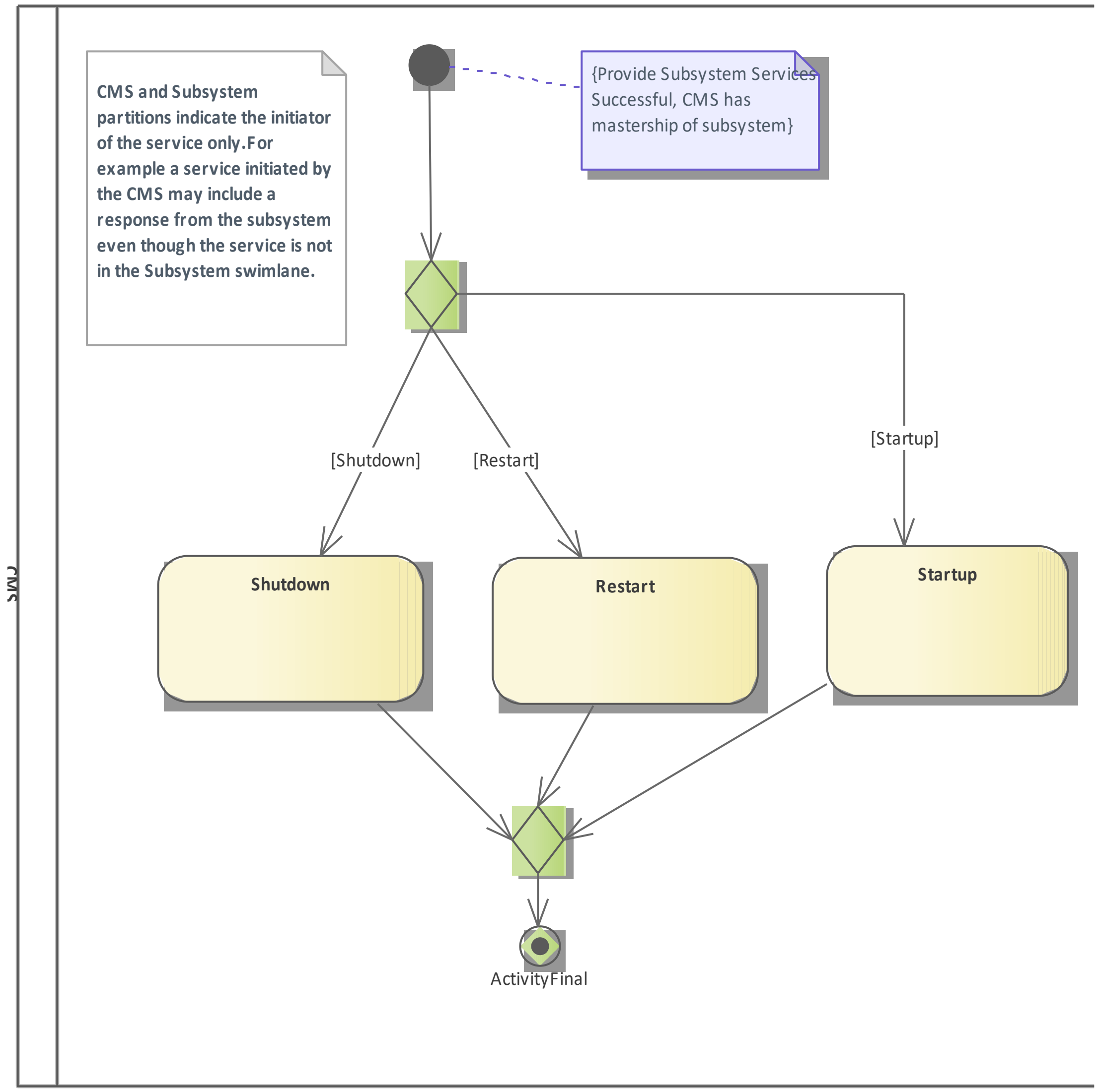


Figure 7.20 Compliance Level 3B - Macro State Management (Activity diagram)

These interfaces provide for more finely grained control of startup and shutdown.

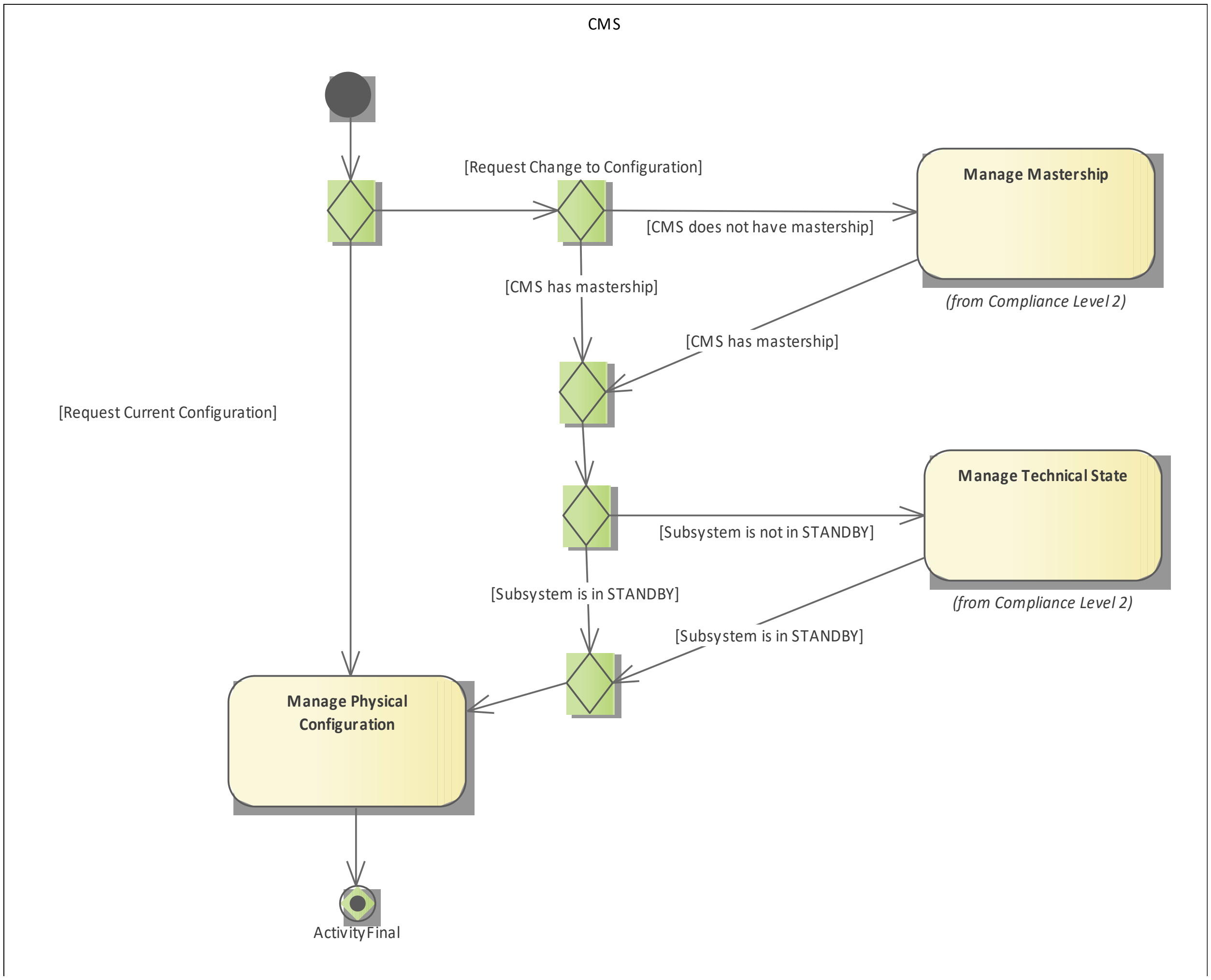


Figure 7.21 Compliance Level 3B - Manage Physical Configuration (Activity diagram)

These interfaces support more detailed control of the subsystem configuration.

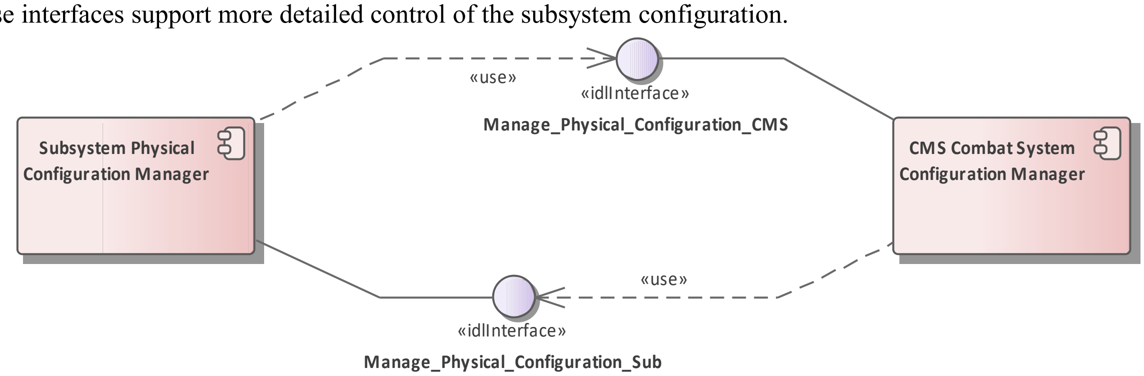
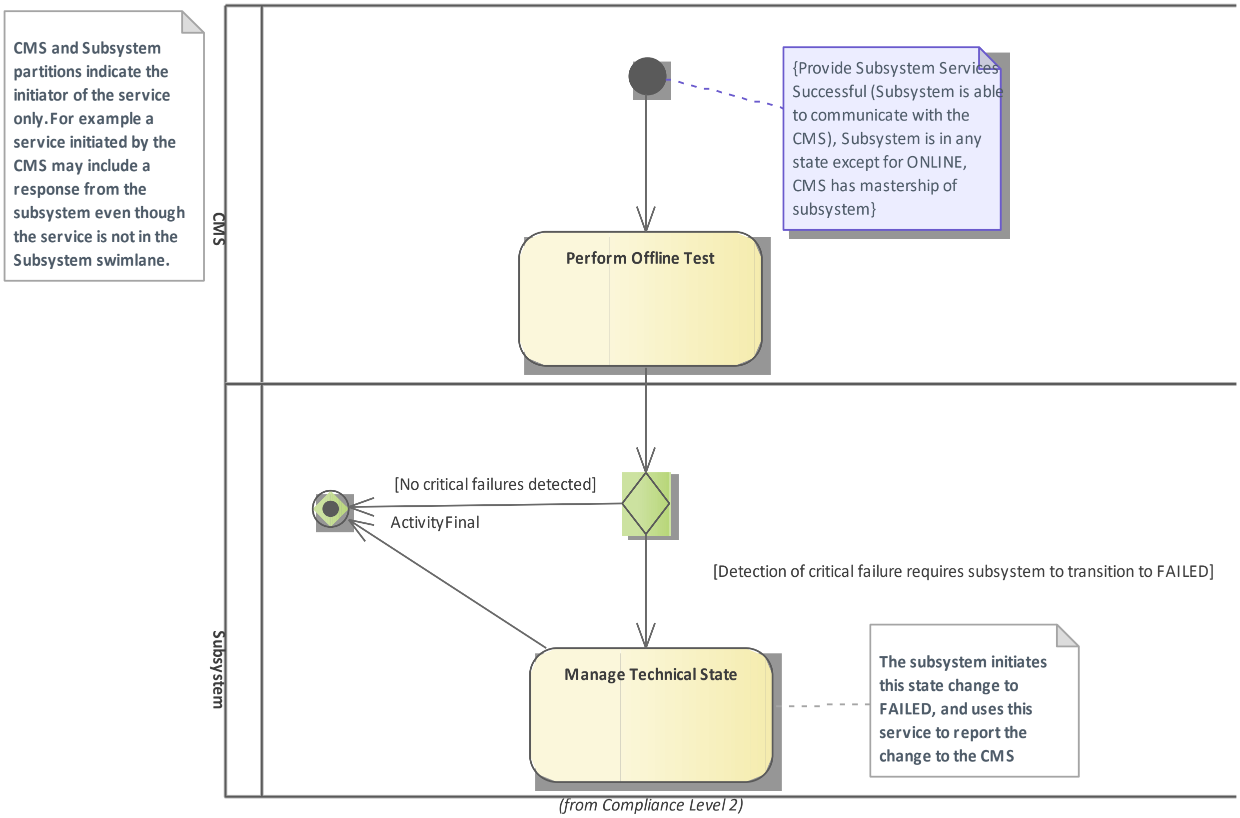


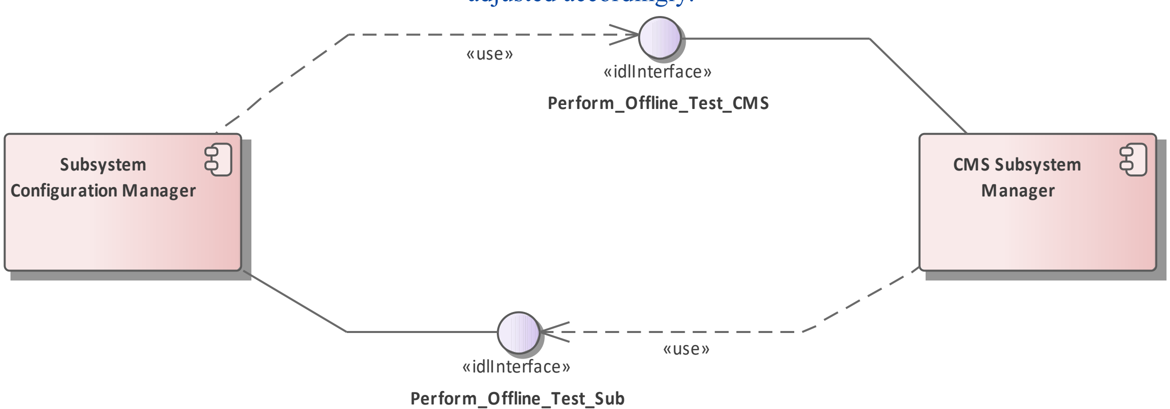
Figure 7.22 Compliance Level 3B - Manage Physical Configuration (Component diagram)

This component diagram shows the interfaces realized and used by CMS and Subsystem components, to manage the subsystem's physical configuration, integrated at compliance level 3B.



~~Figure 7.~~23 ~~Compliance Level 3B - Perform Offline Test (Activity diagram)~~

~~Offline test provides a mechanism for diagnosing subsystem failures, after which the subsystem's technical state is adjusted accordingly.~~



~~Figure 7.~~24 ~~Compliance Level 3B - Perform Offline Test (Component diagram)~~

~~This component diagram shows the interfaces realized and used by CMS and Subsystem components, to perform offline tests, integrated at compliance level 3B.~~

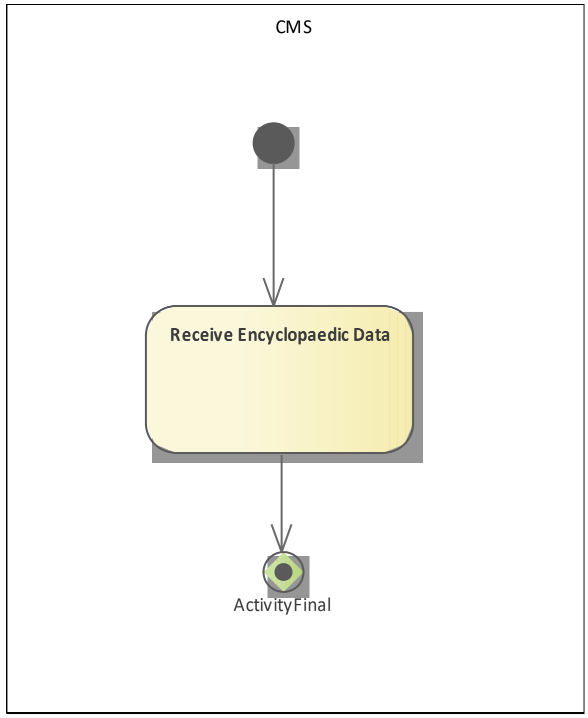


Figure 7.25 Compliance Level 3B - Receive Encyclopedic Data (Activity diagram)

The subsystem is able to receive relevant encyclopedic data from the CMS.

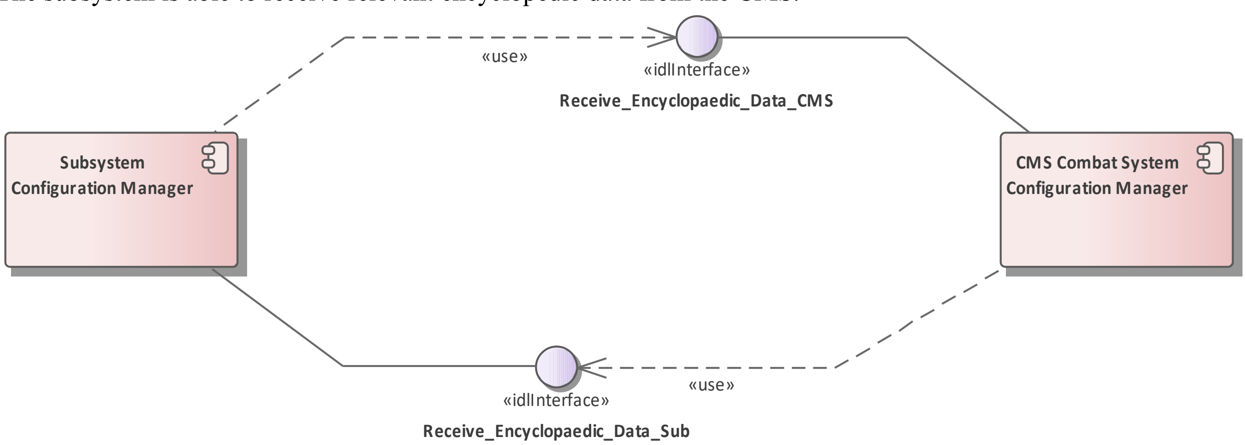


Figure 7.26 Compliance Level 3B - Receive Encyclopedic Data (Component diagram)

This component diagram shows the interfaces realized and used by CMS and Subsystem components, to transfer encyclopedic data, integrated at compliance level 3B.

7.3.4.1 CMS Combat System Configuration Manager

A non-normative example of a CMS function to manage the configuration of the Combat System.

~~**CMS Subsystem Manager**~~

7.3.4.2 ~~A non-normative example of a CMS function to manage the state of subsystems in the combat system.~~

7.3.4.3

~~**Subsystem Configuration Manager**~~

7.3.4.4 ~~A non-normative example of a subsystem function to manage its configuration and state.~~

7.3.4.5 Subsystem Physical Configuration Manager

A non-normative example of a subsystem function to manage its physical configuration (i.e. state of hardware and associated mechanical aspects and devices).

7.3.5 Compliance Level 3C

Parent Package: Usage Overview

The Compliance Level 3C required interfaces are:

- Receive Track Information
- Delete Sensor Track
- Initiate Track
- Perform Cued Search
- Provide Space Plots
- Provide Land Plots
- Provide Surface Plots
- Provide Air Plots
- Provide Sensor Space Tracks
- Provide Sensor Land Tracks
- Provide Sensor Surface Tracks
- Provide Sensor Air Tracks

This compliance level supports specialized provision and management of tracks and plots; it applies to sensors in general.

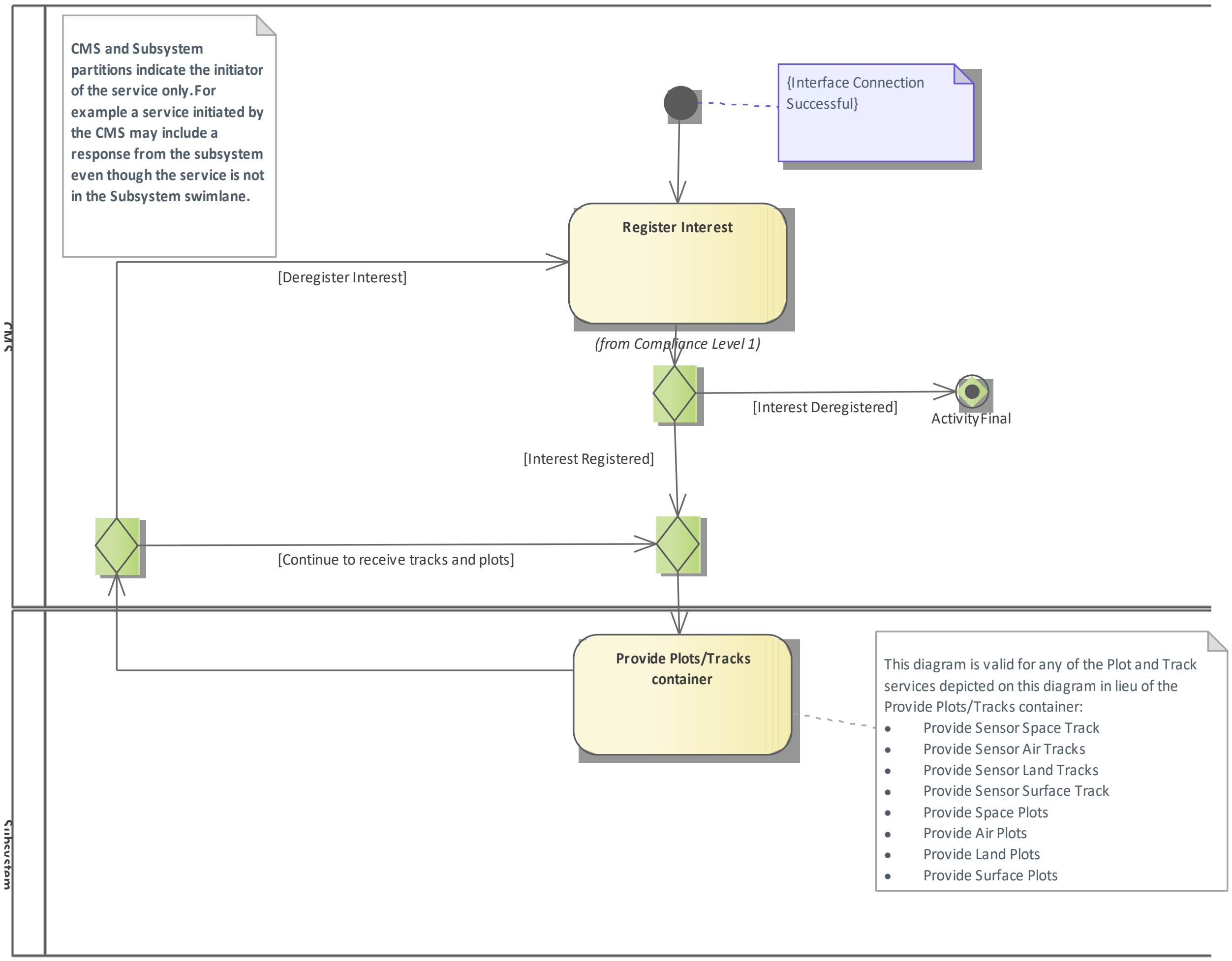


Figure 7.27 Compliance Level 3C - Advanced Track and Plot Reporting (Activity diagram)

The sensor supports reporting tracks and plots selectively based on the operational environment (space/air/land/surface).

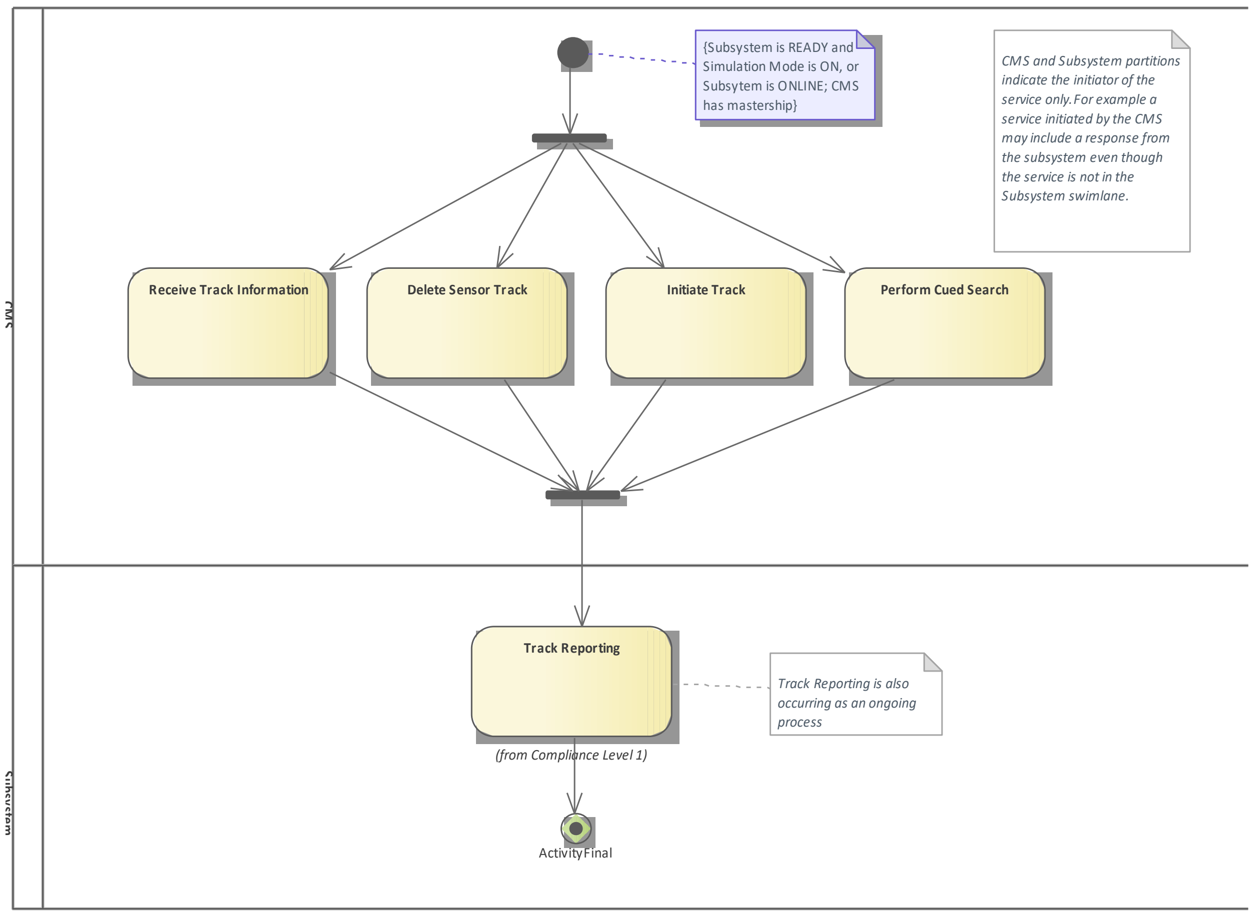


Figure 7.28 Compliance Level 3C - Advanced Track Management (Activity diagram)

The sensor supports detailed track management.

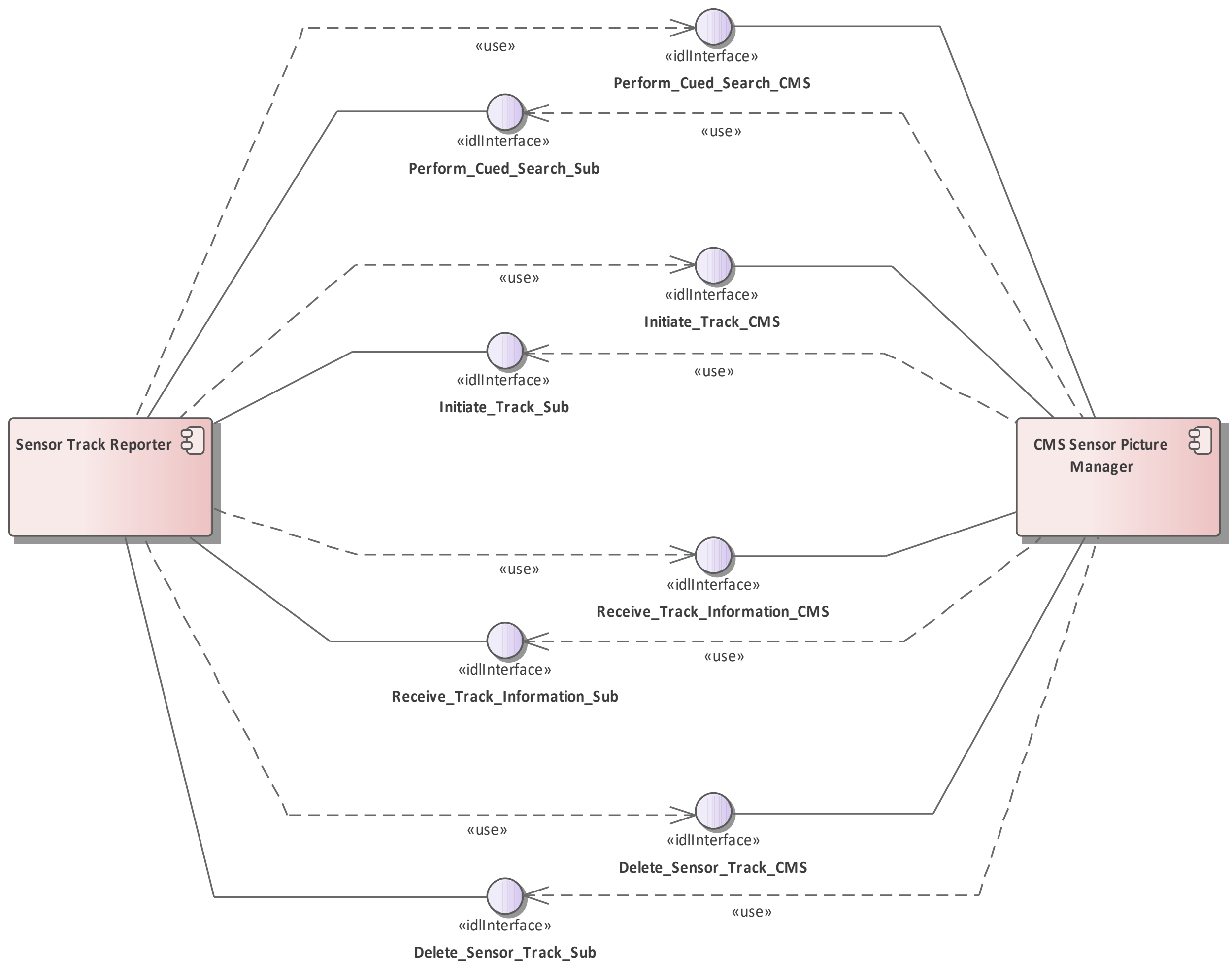


Figure 7.29 Compliance Level 3C - Advanced Track Management (Component diagram)

This component diagram shows the interfaces realized and used by CMS and Sensor Subsystem components integrated at compliance level 3C.

7.3.5.1 CMS Sensor Picture Manager

A non-normative example of a CMS picture management function.

7.3.5.2 Sensor Track Reporter

A non-normative example of a sensor function to manage track reporting.

7.3.6 Compliance Level 3D

Parent Package: Usage Overview

The Compliance Level 3D required interfaces are:

- Process Target Designation
- Provide Projectile Positional Information
- Perform Missile Downlink

- Perform Missile Uplink
- Kill Assessment
- Support Surface Engagement
- Perform Splash Plotting

This compliance level supports specialized engagement related radar functionality; it is specific to radar sensors.

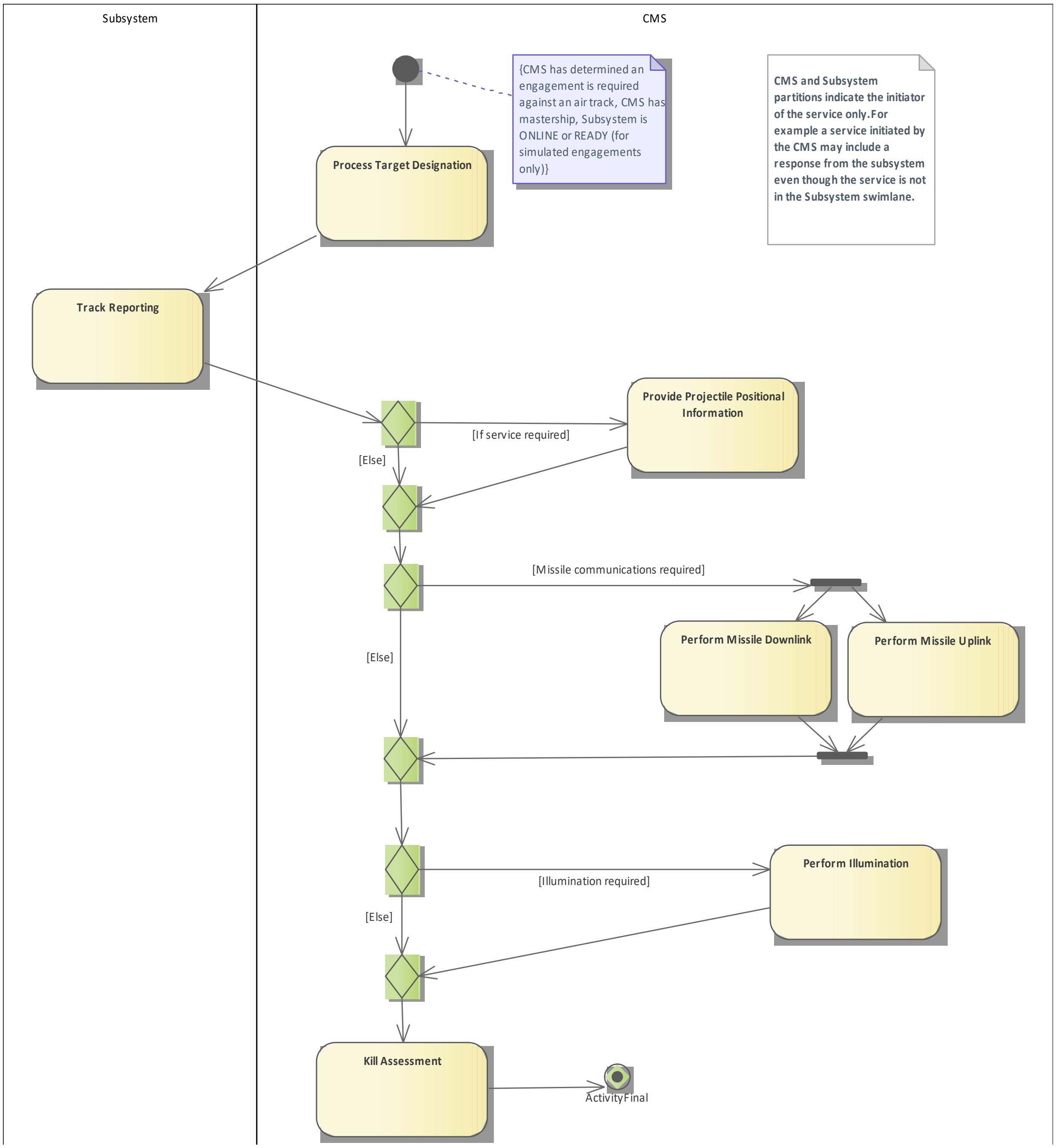


Figure 7.30 Compliance Level 3D - Air Engagement Support (Activity diagram)

Level 3D provides additional information to support air engagements, including missile links and kill assessment.

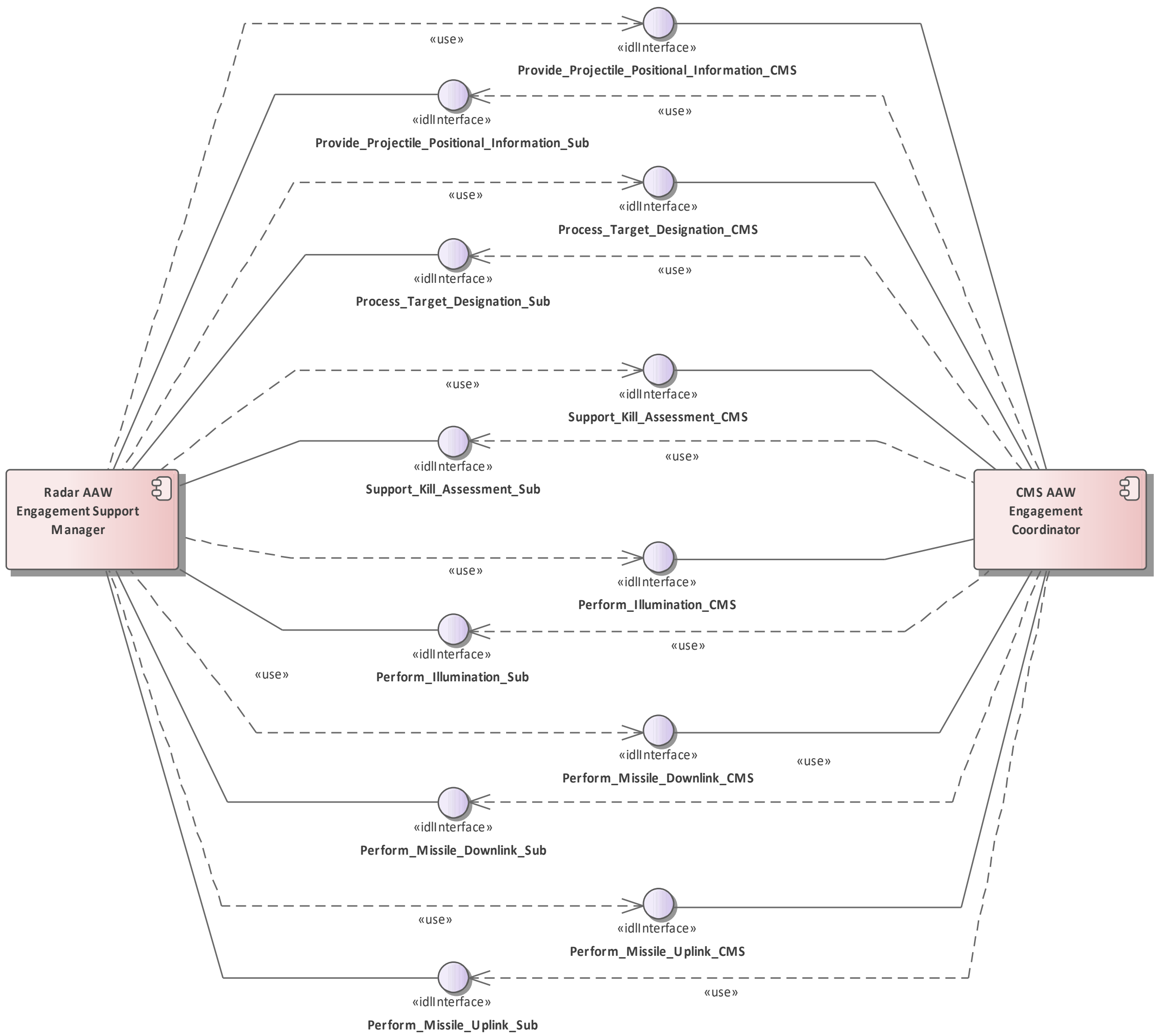


Figure 7.31 Compliance Level 3D - Air Engagement Support (Component diagram)

This component diagram shows the interfaces realized and used by CMS and Radar Subsystem components, to support air engagements, integrated at compliance level 3D.

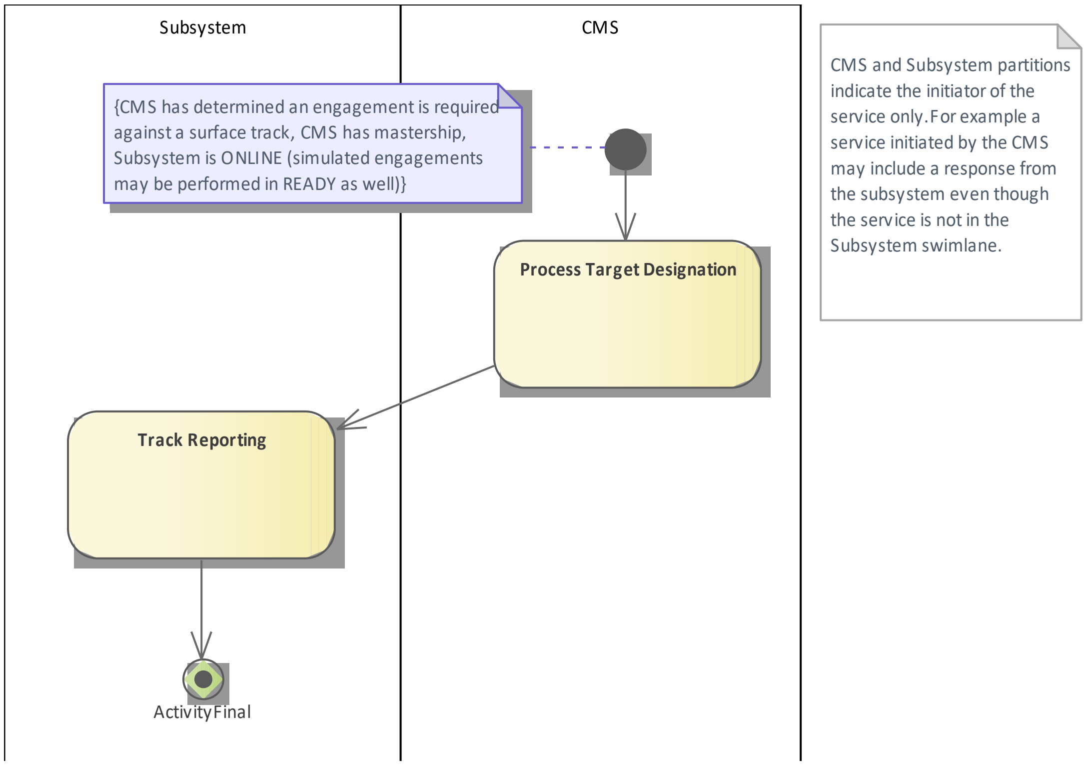


Figure 7.32 Compliance Level 3D - Surface Engagement Support - Fire Control Radar (Activity diagram)

This provides additional surface engagement support for fire control.

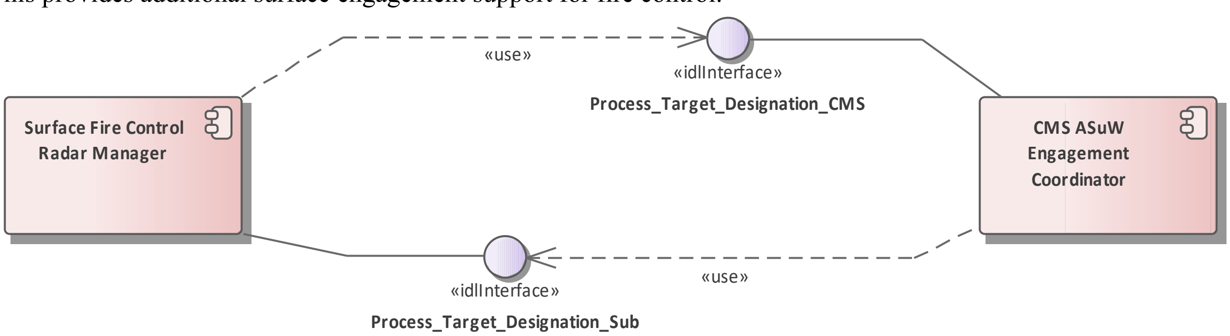


Figure 7.33 Compliance Level 3D - Surface Engagement Support - Fire Control Radar (Component diagram)

This component diagram shows the interfaces realized and used by CMS and Fire Control Radar Subsystem components, to support surface engagements integrated at compliance level 3D.

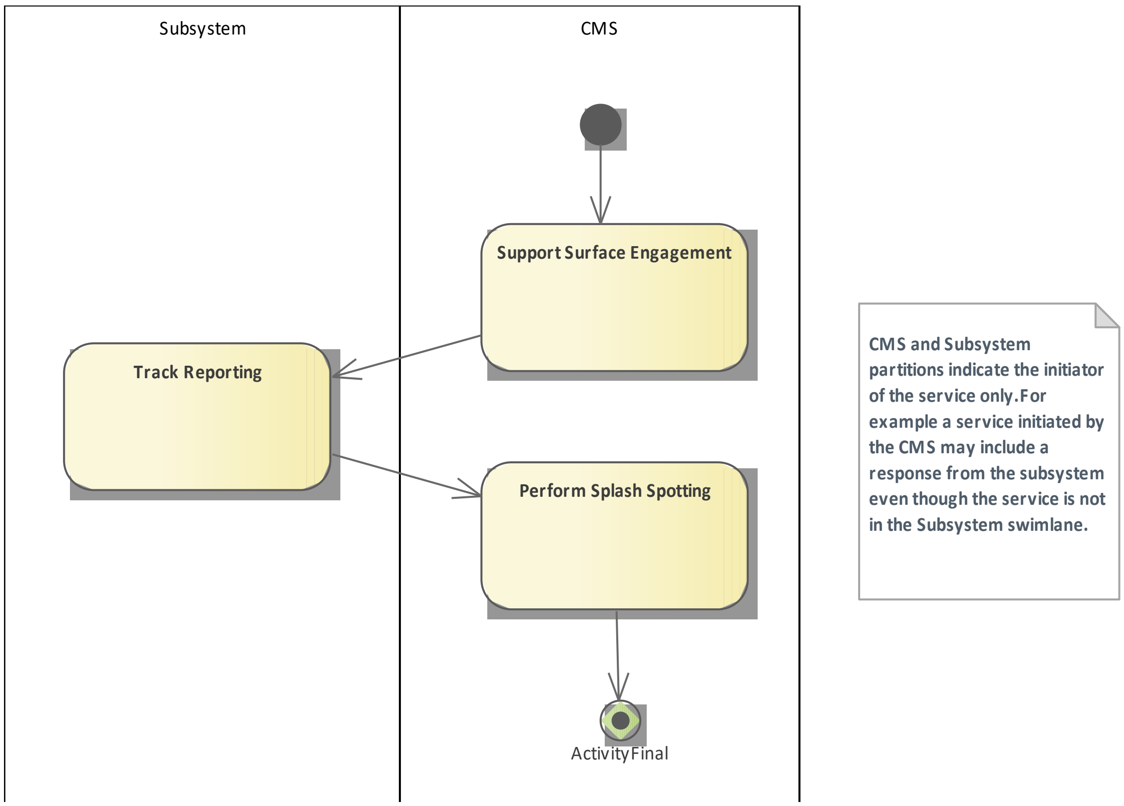


Figure 7.34 Compliance Level 3D - Surface Engagement Support - Surveillance Radar (Activity diagram)

This provides additional surface engagement support for surveillance purposes.

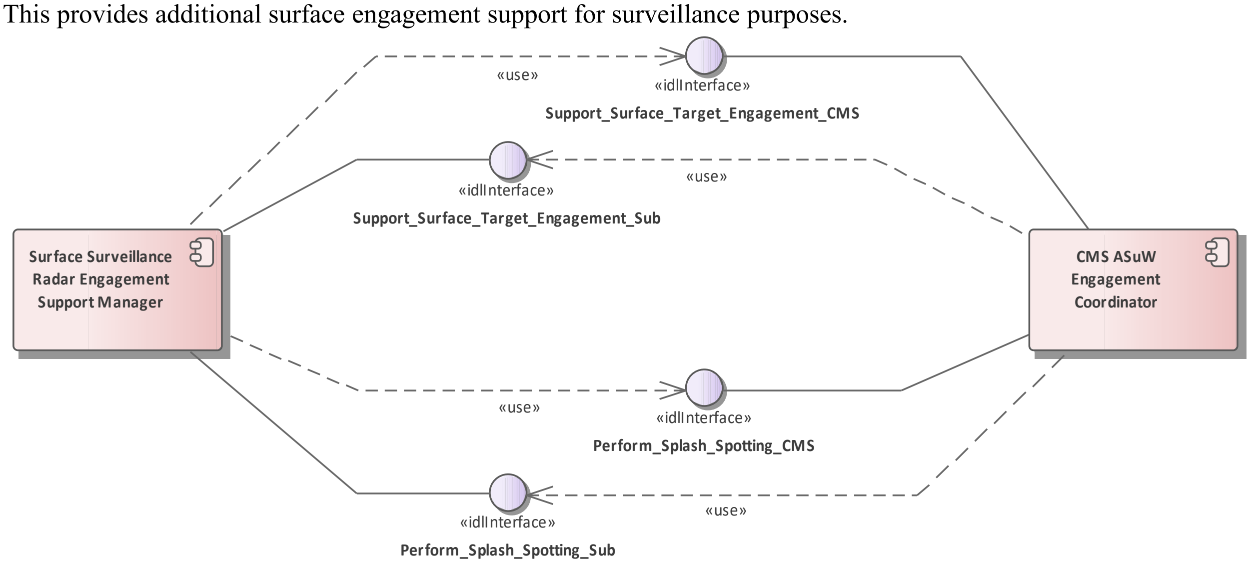


Figure 7.35 Compliance Level 3D - Surface Engagement Support - Surveillance Radar (Component diagram)

This component diagram shows the interfaces realized and used by CMS and Surveillance Radar Subsystem components, to support surface engagements integrated at compliance level 3D.

7.3.6.1 CMS AAW Engagement Coordinator

A non-normative example of CMS functionality to coordinate anti-air warfare engagements.

7.3.6.2 CMS ASuW Engagement Coordinator

A non-normative example of CMS functionality to coordinate anti-surface warfare engagements.

7.3.6.3 Radar AAW Engagement Support Manager

A non-normative example of Radar Sensor functionality providing anti-air warfare engagement support.

7.3.6.4 Surface Fire Control Radar Manager

A non-normative example of Fire-Control Radar Sensor functionality providing anti-surface warfare engagement support.

7.3.6.5 Surface Surveillance Radar Engagement Support Manager

A non-normative example of Surveillance Radar Sensor functionality providing anti-surface warfare engagement support.

7.3.7 Compliance Level 3E

Parent Package: Usage Overview

The Compliance Level 3E required interfaces are:

- Provide Interference Reports
- Provide Jammer Strobes
- Provide Jammer Tracks
- Provide Area with Plot Concentration
- Provide Clutter Assessment
- Provide Jamming Effect Assessment
- Provide Performance Assessment
- Provide Nominal Performance

This compliance level is for the provision and management of specialized services to manage sensor functional performance and mitigate jamming; it is applicable to sensors in general.

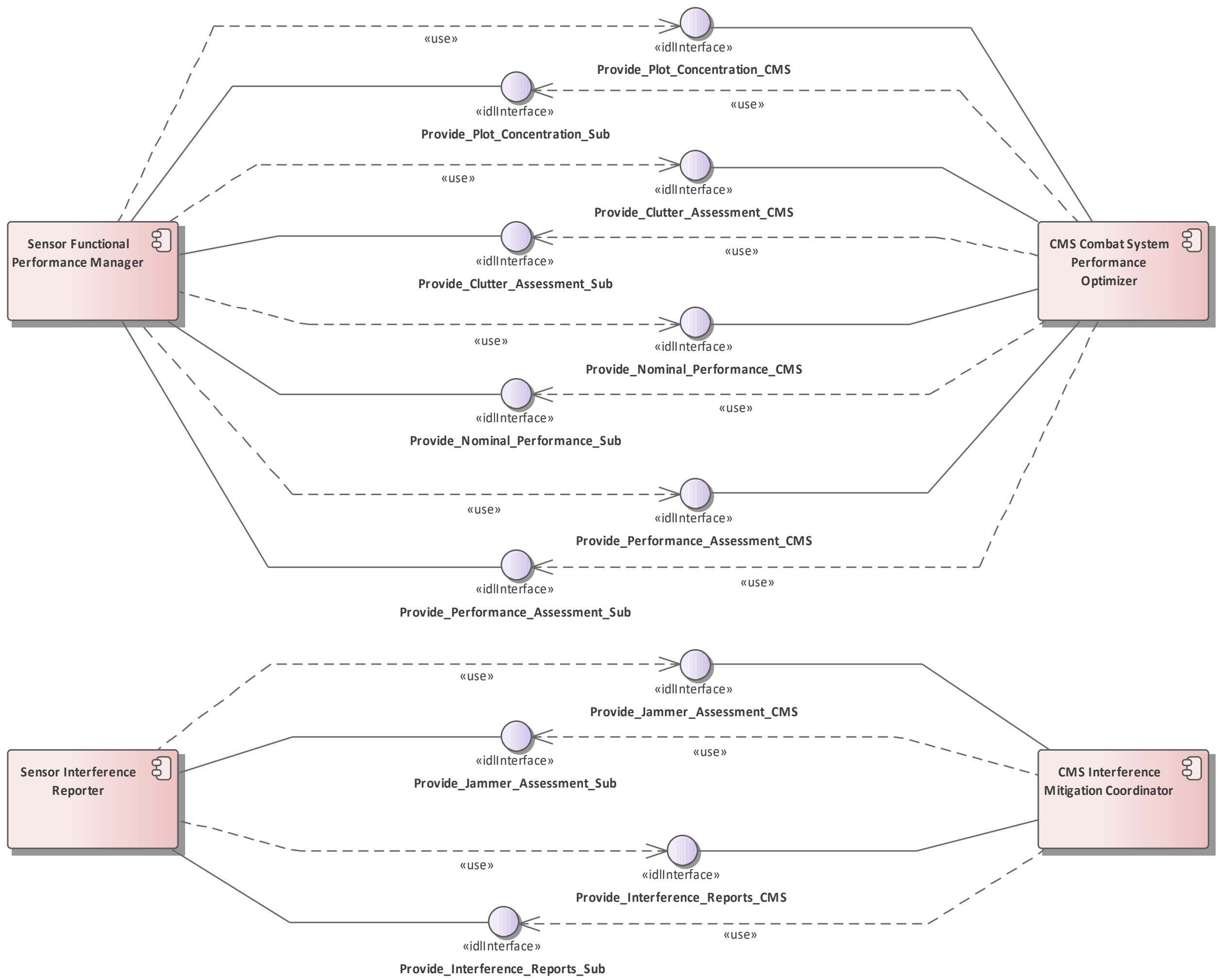


Figure 7.36 Compliance Level 3E (Component diagram)

This component diagram shows the interfaces realized and used by CMS and Subsystem components integrated at compliance level 3E.

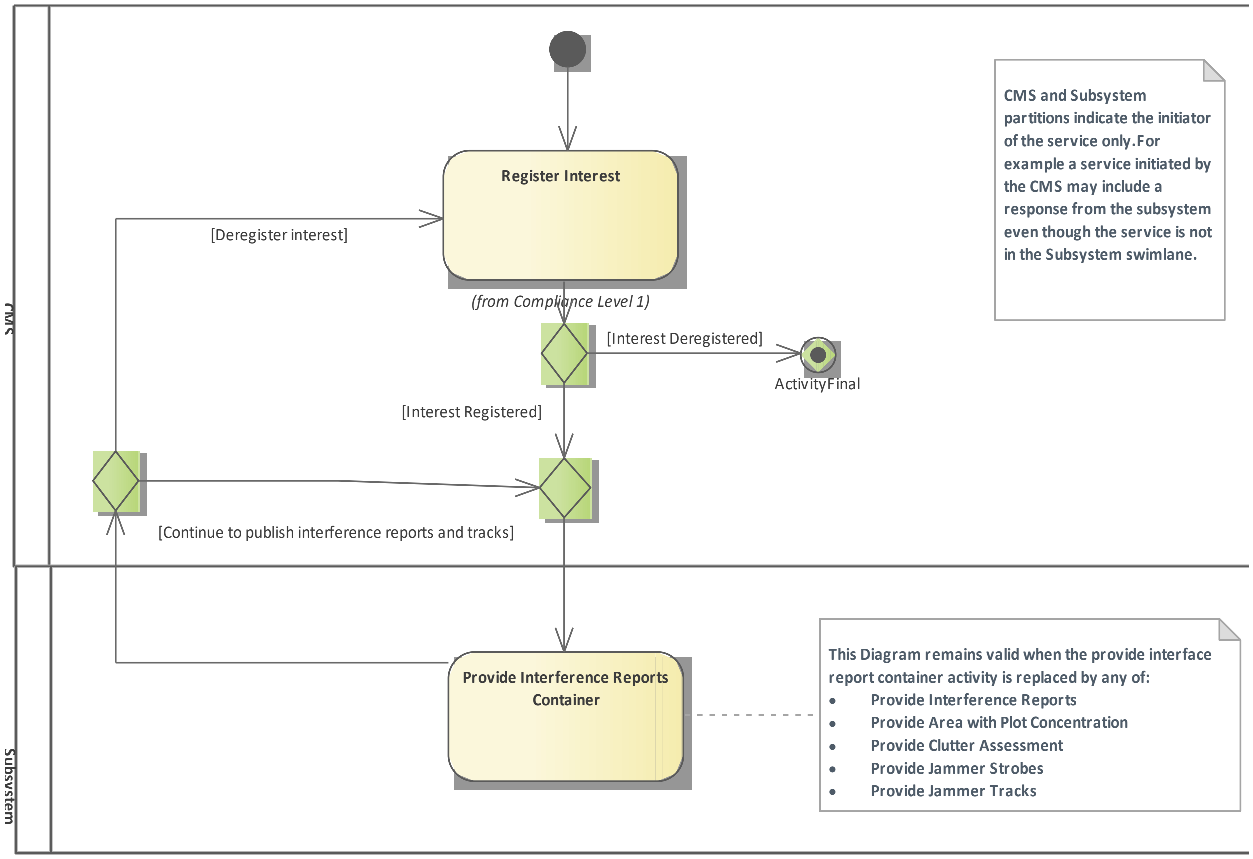


Figure 7.37 Compliance Level 3E - Automatic Interference Reporting (Activity diagram)

Level 3E provides for detailed interference reporting, including jammers.

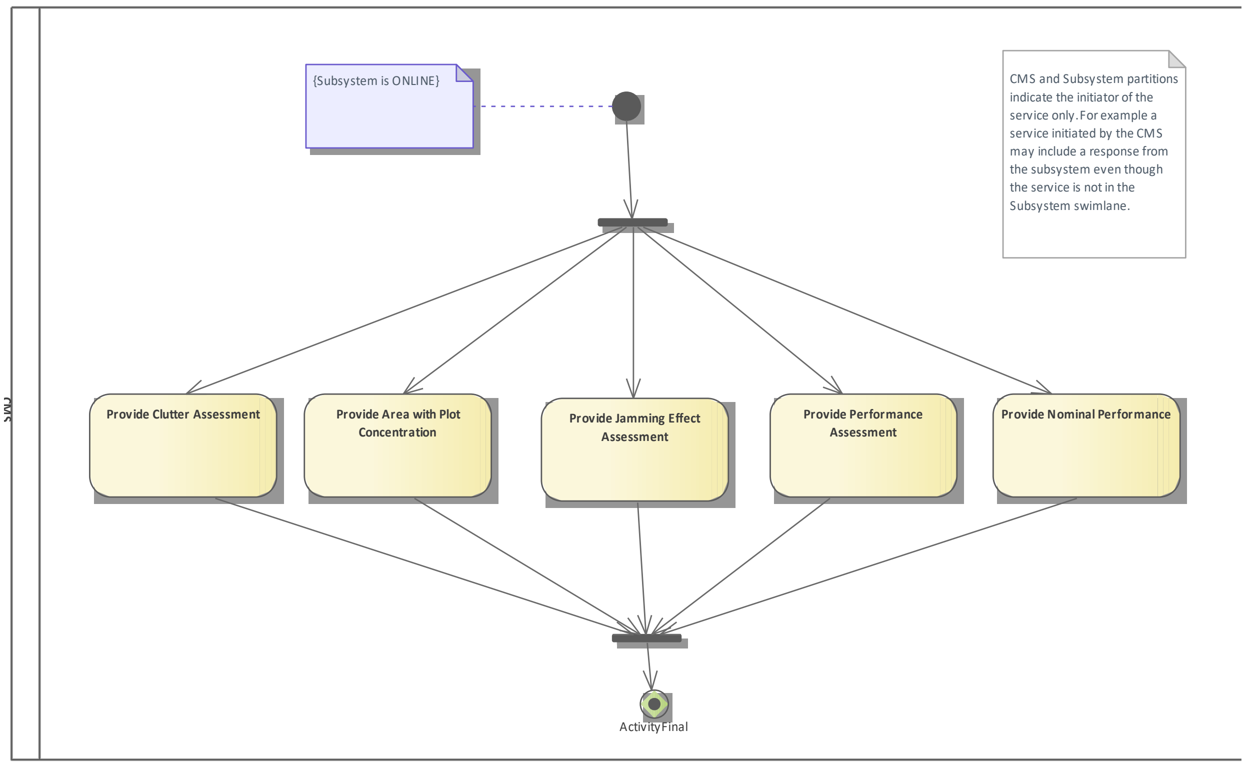


Figure 7.38 Compliance Level 3E - Requested Interference Reports (Activity diagram)

These interfaces provide for reporting sensor specified and actual performance in addition to interference related information.

7.3.7.1 CMS Combat System Performance Optimizer

A non-normative example of CMS functionality to understand and hence optimize the performance of the combat system.

7.3.7.2 CMS Interference Mitigation Coordinator

A non-normative example of CMS functionality to coordinate mitigation with respect to active interference in the environment - e.g. jamming.

7.3.7.3 Sensor Functional Performance Manager

A non-normative example of sensor functionality to manage, interrogate and publish its own functional performance.

7.3.7.4 Sensor Interference Reporter

A non-normative example of sensor functionality to report interference detected in the external environment.

7.3.8 Compliance Level 3H

Parent Package: Usage Overview

The Compliance Level 3H required interfaces are:

- Allocate_Tracks_To_Stream
- Configure_Media_Streams
- Assess_Sensor_Plot
- Assess_Sensor_Track
- Configure_Measurement_Parameters
- Provide_Sensor_Plot_Parameters
- Provide_Sensor_Track_Parameters

This compliance level is for the integration of sensors other than radars and in particular the publication of parametric data, assessment of identify and classification, and to relate media streams to tracks.

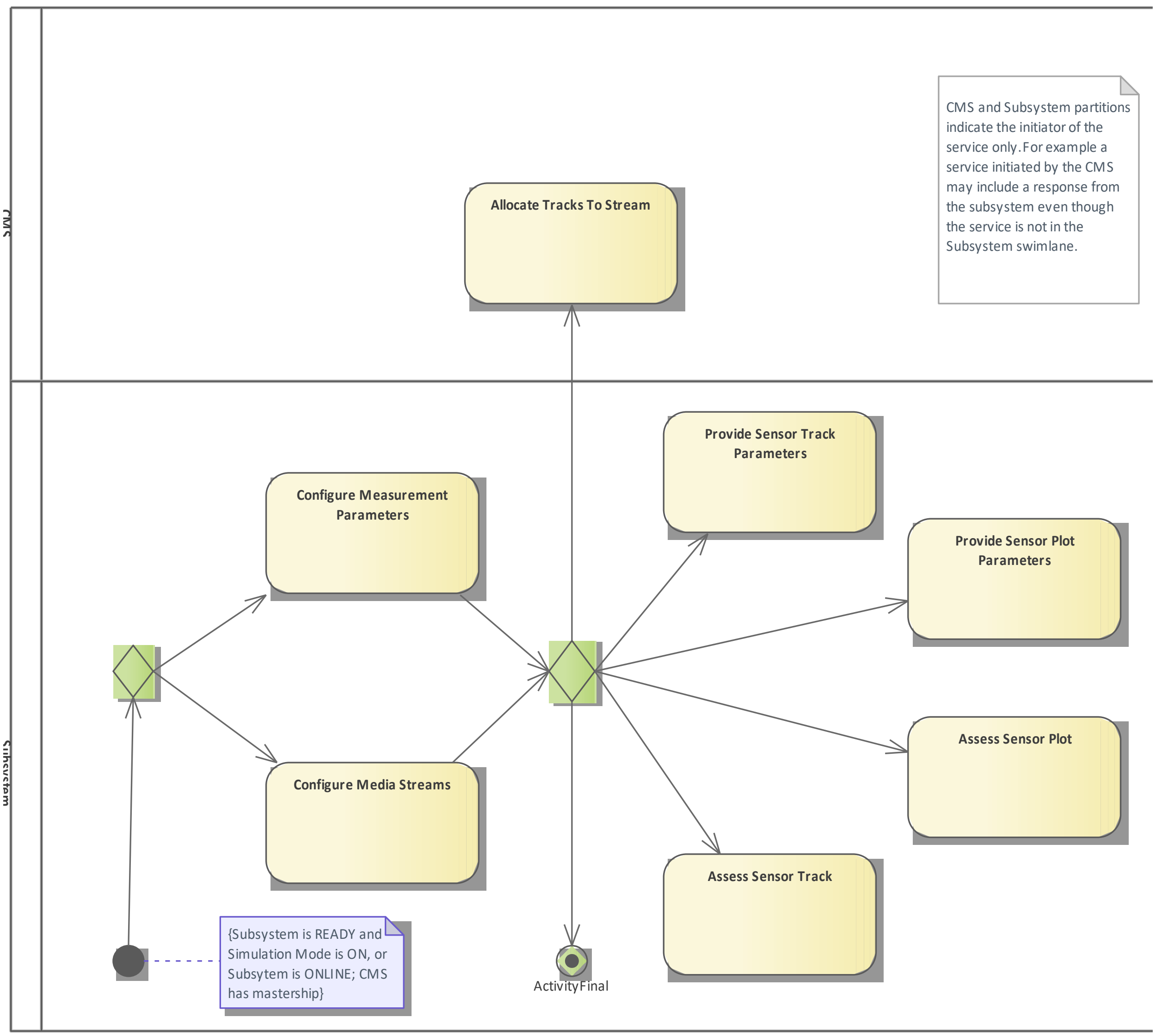


Figure 7.39 Compliance Level 3H - Measurement and Identification Assessment (Activity diagram)

These interfaces support the processing and assessment of information derived from the sensor's processing chain (especially detailed parametric data) to aid the identification and classification processes within the CMS.

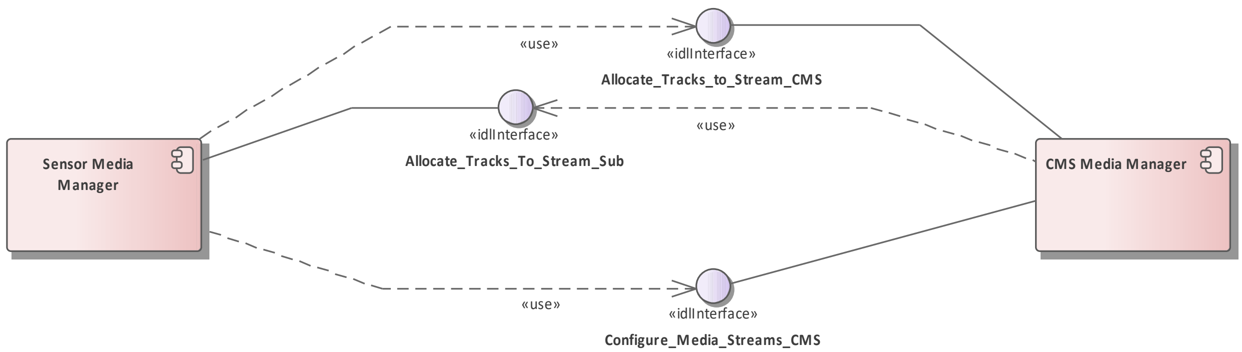


Figure 7.40 Compliance Level 3H - Media Streaming (Component diagram)

This component diagram shows the interfaces realized and used by CMS and Subsystem components, to relate media streams to the tactical picture, integrated at compliance level 3H.

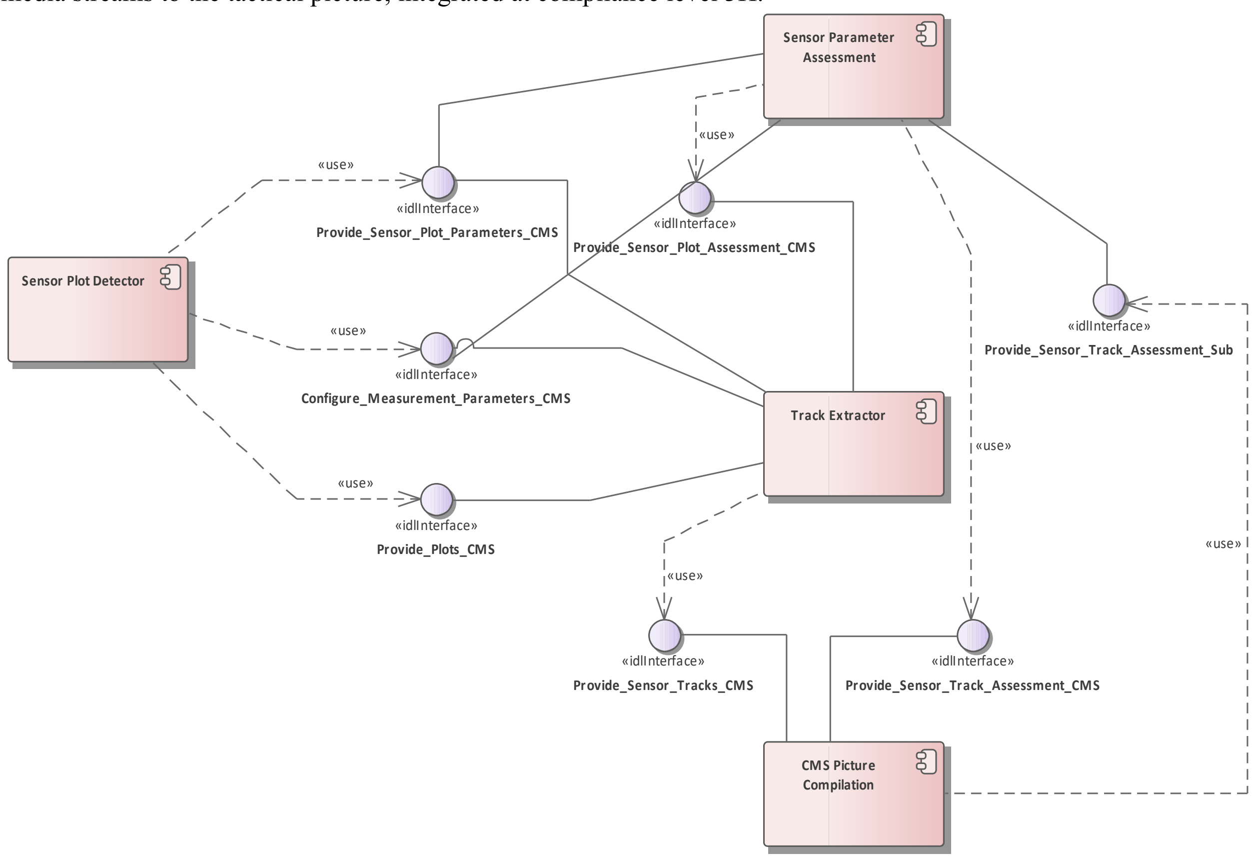


Figure 7.41 Compliance Level 3H - Picture Compilation From Plots (Component diagram)

This component diagram shows the interfaces realized and used by CMS and Subsystem components, to utilize plot-level parametric data and assessment functions, integrated at compliance level 3H.

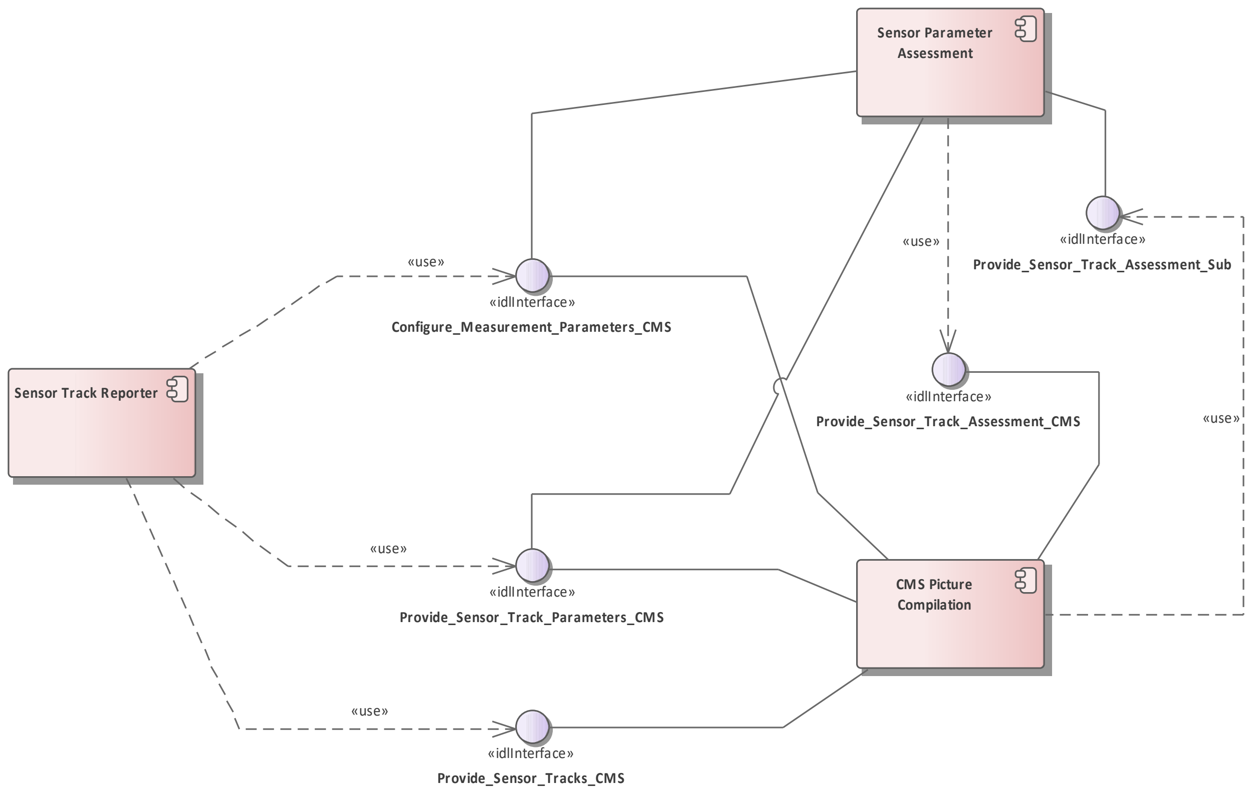


Figure 7.42 Compliance Level 3H - Picture Compilation From Tracks (Component diagram)

This component diagram shows the interfaces realized and used by CMS and Subsystem components, to utilize track-level parametric data and assessment functions, integrated at compliance level 3H.

7.3.8.1 CMS Media Manager

A non=normative example of CMS functionality to manage media streams derived from or otherwise available to the combat system.

7.3.8.2 CMS Picture Compilation

A non-normative example of CMS tactical picture management functionality.

7.3.8.3 Sensor Media Manager

A non-normative example of sensor functionality for providing a media stream.

7.3.8.4 Sensor Parameter Assessment

A non-normative example of functionality for the assessment of sensor parametric data.

7.3.8.5 Sensor Plot Detector

A non-normative example of sensor plot detection and reporting functionality.

7.3.8.6 Sensor Track Reporter

A non-normative example of functionality to report sensor tracks to the combat system.

7.3.8.7 Track Extractor

A non-normative example of functionality to extract tracks from a stream of plots.

7.3.9 Compliance Level 3I

Parent Package: Usage Overview

The Compliance Level 3I required interfaces are:

- Manage_Network_Participation
- Provide_Networking_Statistics
- Filter_Plots
- Provide_Sensor_Characteristics
- Filter_Tracks
- Label_Tracks

This compliance level is for the sharing and exploitation of plots from multiple sensors across multiple platforms (e.g. ships) cooperating within a task group.

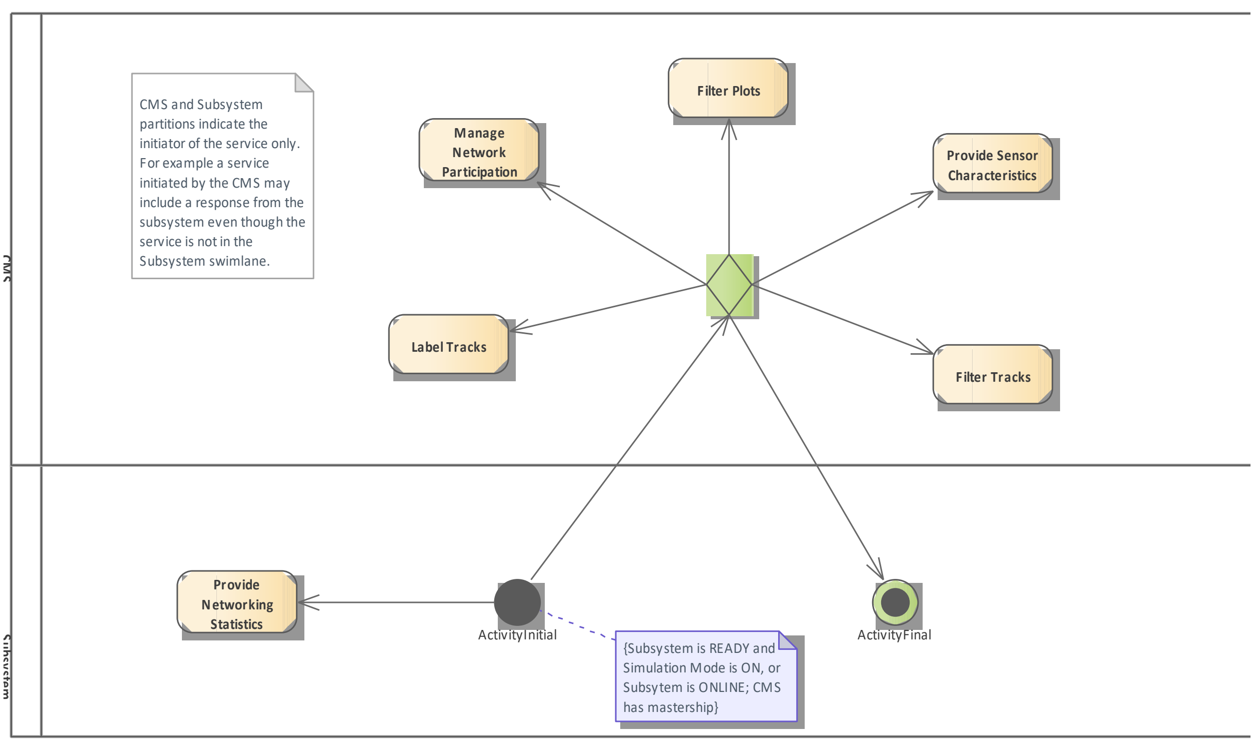


Figure 7.43 Compliance Level 3I (Activity diagram)

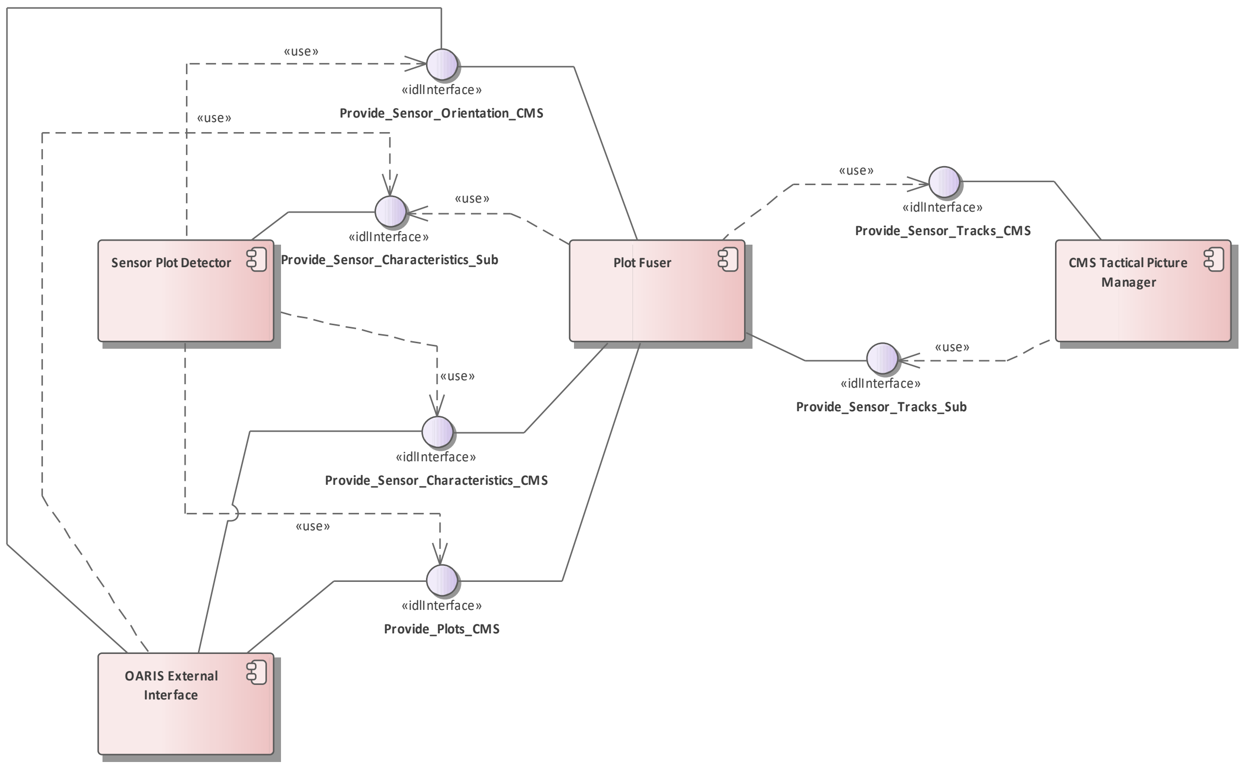


Figure 7.44 Compliance Level 3I - Plots (Component diagram)

This component diagram shows the interfaces realized and used by CMS and Subsystem components, to share plots within a platform, integrated at compliance level 3I.

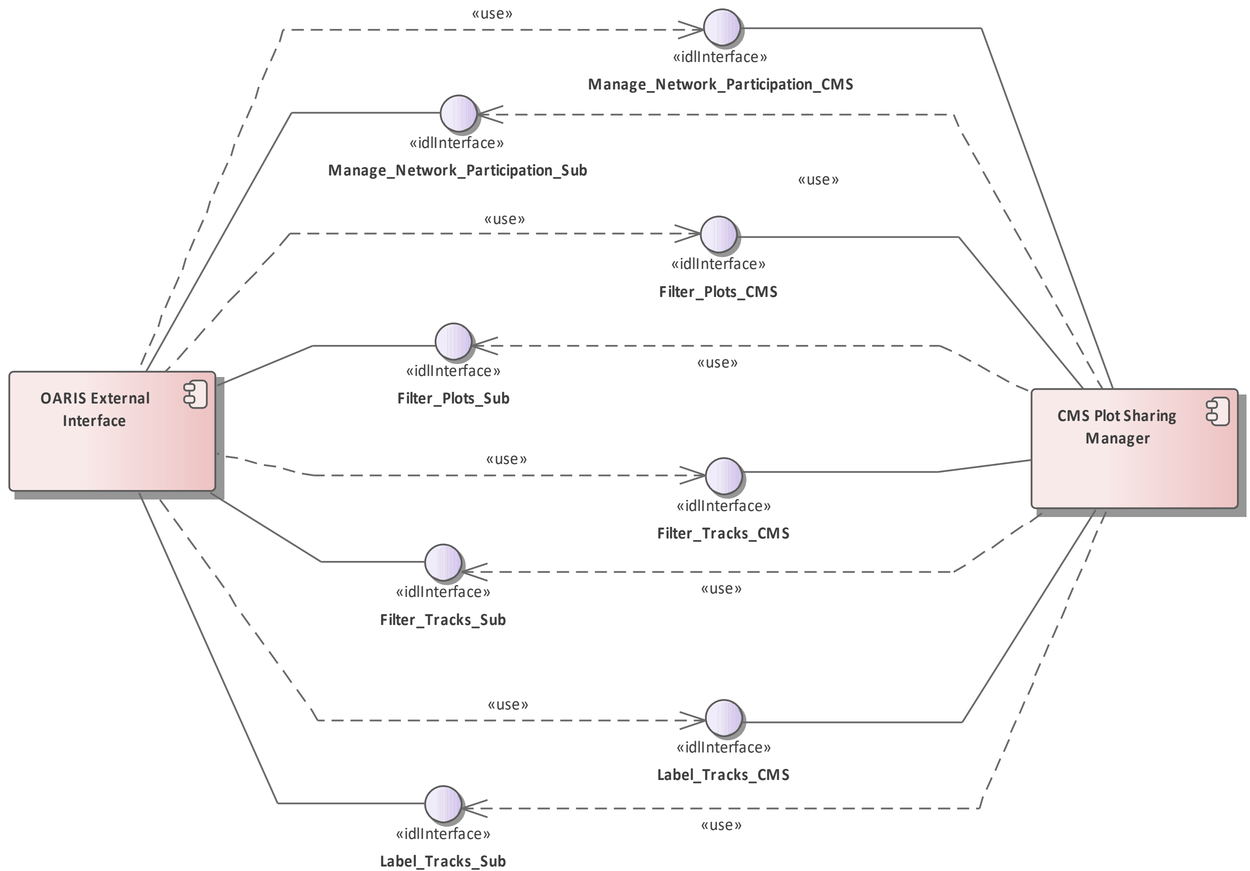


Figure 7.45 Compliance Level 3I - Picture Management (Component diagram)

This component diagram shows the interfaces realized and used by CMS and Subsystem components, to manage the sharing of tactical picture information, integrated at compliance level 3I.

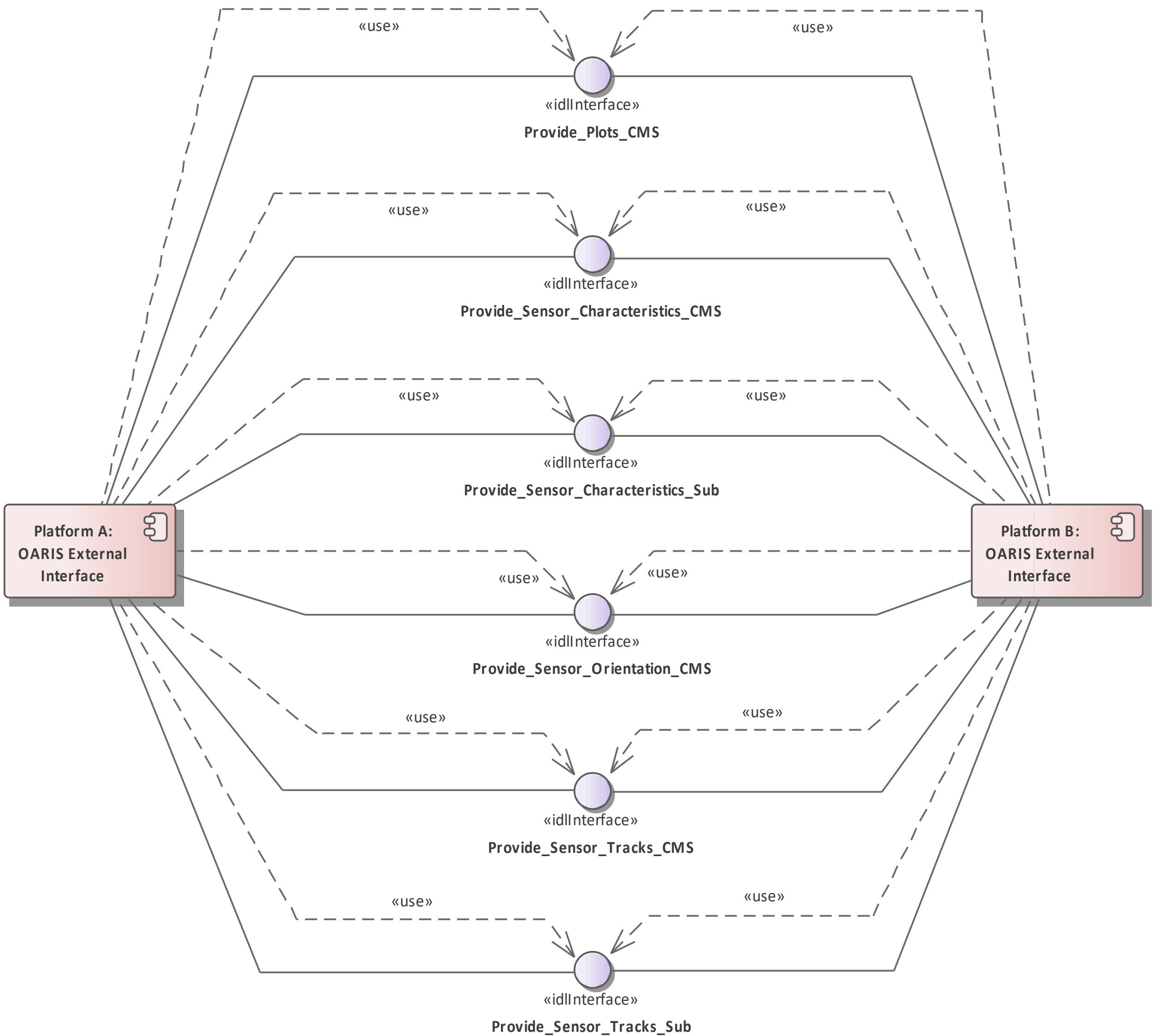


Figure 7.46 Compliance Level 3I - External Information Exchange (Component diagram)

This component diagram shows the interfaces realized and used by CMS and Subsystem components, to exchange information between platforms, integrated at compliance level 3I.

7.3.9.1 CMS Plot Sharing Manager

A non-normative example of CMS functionality to manage the sharing of plot-level information.

7.3.9.2 CMS Tactical Picture Manager

A non-normative example of CMS functionality to produce and manage the tactical picture.

7.3.9.3 OARIS External Interface

A non-normative example of functionality to provide the external-to-platform interface for distributing plots and other information to enable the exploitation of plots across multiple platforms. This component expects to interface peer-to-peer with other equivalent components using a symmetric, bidirectional interface.

7.3.9.4 Platform A

A non-normative instance of the component, notionally resident on a nominal platform A.

7.3.9.5 Platform B

A non-normative instance of the component, notionally resident on a nominal platform B.

7.3.9.6 Plot Fuser

A non-normative example of functionality to fuse plots from multiple sensors (and platforms) into continuous tracks.

7.3.9.7 Sensor Plot Detector

A non-normative example of sensor plot detection and reporting functionality.

7.4 Common_Types

Parent Package: Domain_Model

This package contains the types that are common to several areas of the model. Most of the content is in three sub-packages: Coordinates_and_Positions, Shape_Model and Requests. General types are captured at the top level.

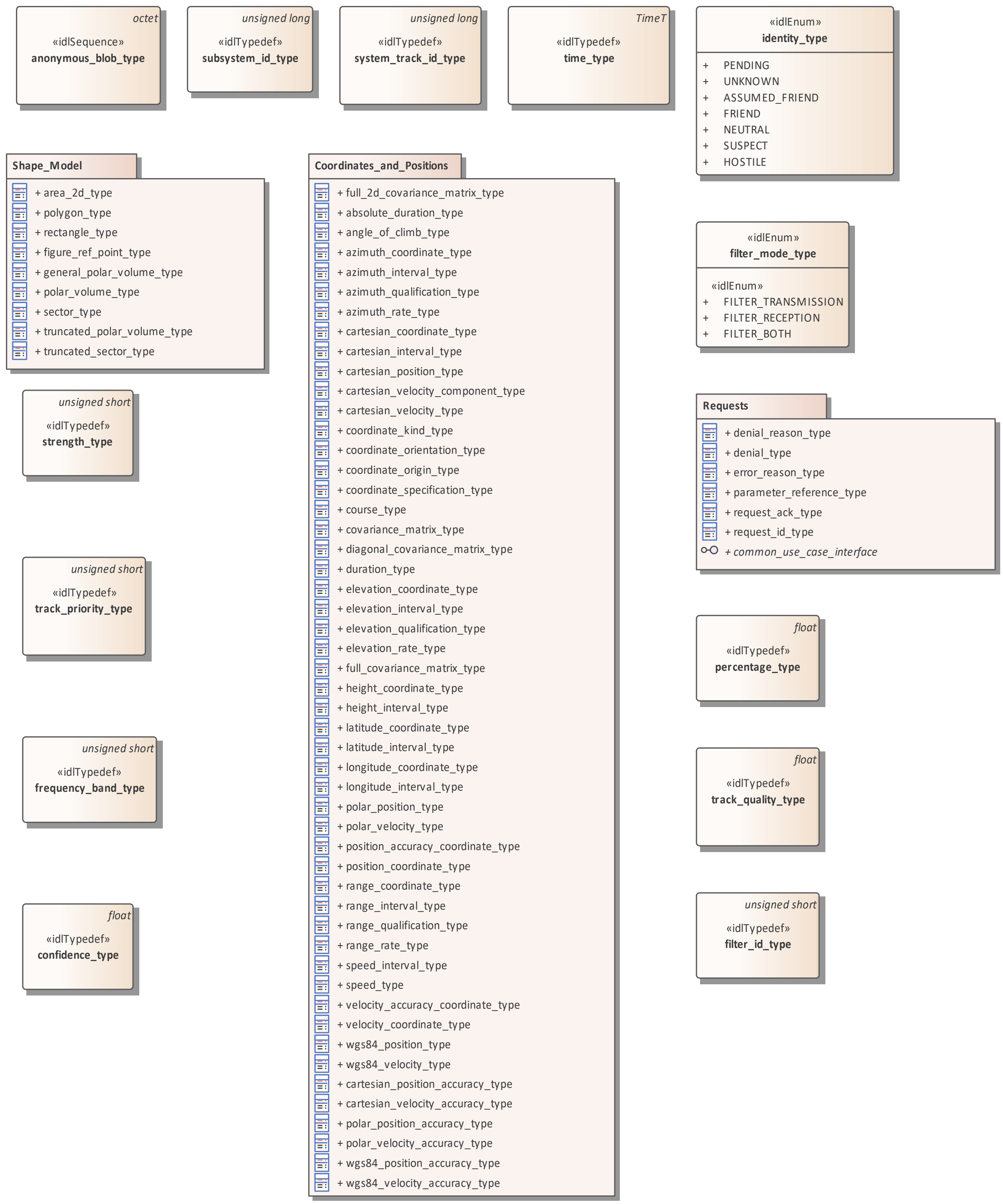


Figure 7.47 Domain Model (Class diagram)

7.4.1 anonymous_blob_type

Type: Class

Package: Common_Types
Representation for a general binary type
ElementTag: Length = 1024

7.4.2 confidence_interval_type

Type: IDLStruct
Package: Common_Types
An abstraction for a range of confidence values.

Table 7.1 - Attributes of IDLStruct confidence_interval_type

Attribute	Notes
minimum confidence_type	The minimum inclusive value for the interval.
maximum confidence_type	The maximum inclusive value for the interval.

7.4.3 confidence_type

Type: Class
Package: Common_Types
The confidence in the measurement or assessment expressed as a probability. This is the result of a hypothesis test that the data is a measurement of real-world phenomenon corresponding to its label. For an assessment it is the hypothesis that the assessment describes the real-world.
ElementTag: Range = 0 .. 1

7.4.4 filter_id_type

Type: IDLTypeDef
Package: Common_Types

7.4.5 filter_mode_type

Type: IDLEnum
Package: Common_Types
This class encapsulates the possible modes in which a filter can operate.

Table 7.2 - Attributes of IDLEnum filter_mode_type

Attribute	Notes
«idlEnum» **FILTER_TRANSMISSION**	The filter is applied to transmission of data (plots or tracks).
«idlEnum» **FILTER_RECEPTION**	The filter is applied to reception of data (plots or tracks).
«idlEnum» **FILTER_BOTH**	The filter is applied to both transmission and reception of data (plots or tracks).

7.4.6 identity_type

Type: IDLEnum
Package: Common_Types
Identity according to STANAG 5516.

Table 7.3 - Attributes of IDLEnum identity_type

Attribute	Notes
PENDING	Value pending the completion of the initial identification process
UNKNOWN	Initial identification complete but identity is unknown.
ASSUMED_FRIEND	Assumed to be a friend
FRIEND	Known to be a friend
NEUTRAL	Known to be neutral
SUSPECT	Suspected to be hostile
HOSTILE	Known to be hostile

7.4.7 percentage_type

Type: IDLTypeDef
Package: Common_Types

7.4.8 quality_interval_type

Type: IDLStruct
Package: Common_Types
An abstraction for a range of track quality values.

Table 7.4 - Attributes of IDLStruct quality_interval_type

Attribute	Notes
minimum track_quality_type	The minimum inclusive value for the interval.
maximum track_quality_type	The maximum inclusive value for the interval.

7.4.9 strength_type

Type: Class
Package: Common_Types
Strength of the measurement (for a track or plot). The precise semantics of this type are sensor subsystem specific, but a typical interpretation is as a signal to noise ratio in dB.

7.4.10 subsystem_id_type

Type: IDLTypeDef
Package: Common_Types
This type provides a unique id for different subsystems. Subsystem ids shall be allocated by the platform integrator. Subsystem id equal to zero is reserved to imply applicability to all and any subsystem.
The lowest two bytes are used for designating subsystems within local control of a particular platform (e.g. a ship including any off-board sensors that it controls). The highest two bytes designate a platform within a co-operating task-force or group. (e.g. highest byte may designate country and the next highest, one of a country's platforms).

7.4.11 system_track_id_type

Type: Class
Package: Common_Types
System Track Identification

7.4.12 time_type

Type: Class
Package: Common_Types
based on start of Gregorian calendar (1582-10-15T 00:00UTC)
unit: 100 nano seconds
i.a.w CORBA Time Service Time T

7.4.13 track_priority_type

Type: Class
Package: Common_Types
The representation of the track's priority with respect to the allocation of the sensor's resources. The meaning of track_priority_type is to assign a priority among a set of tracks based on some criteria (i.e. subsystem's time dedicated to a track analysis). Higher values indicate higher priority and importance and hence that more resources should be extended.
Example of values:
1 Track While Scan (TWS)
2 Low Priority Target (LPT)
3 High Priority Target (HPT)
ElementTag: Range = 0 .. 100

7.4.14 track_quality_type

Type: Class
Package: Common_Types
The representation of the quality of a track for the purposes of comparison according to system specific criteria.
ElementTag: Range = 0.0 .. 1.0

7.4.15 frequency_band_type

Type: Class
Package: Common_Types
An index indicating a particular frequency channel or band. The actual frequency is typically not of concern to the command team. A band refers to a discrete frequency or a range of frequencies; such bands may overlap.

7.4.16 System_Track

Parent Package: Common_Types
A package to contain the system track class.

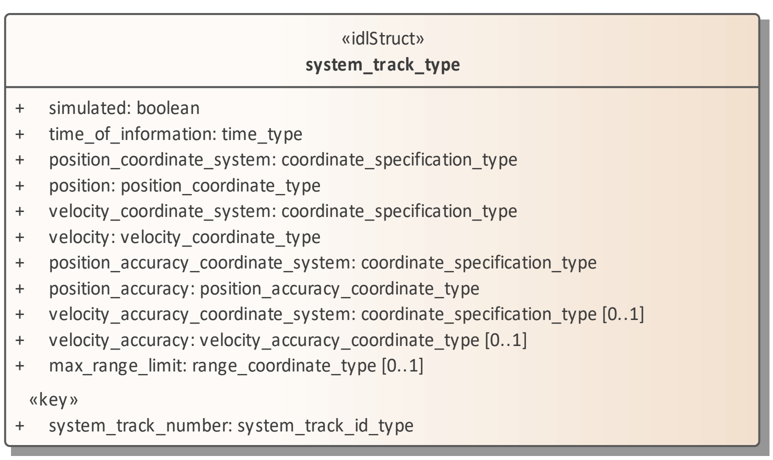


Figure 7.48 Domain Model (Class diagram)

7.4.16.1 system_track_type

Type: IDLStruct
Package: System_Track
System track information is limited to information required by a subsystem for missile guidance.

Table 7.5 - Attributes of IDLStruct system_track_type

Attribute	Notes
«key» **system_track_number** system_track_id_type	The identifier for the system track
simulated boolean	Whether the system track is part of a simulation
time_of_information time_type	The absolute time at which the information in the attributes of the system track is valid.
position_coordinate_system coordinate_specification_type	The coordinate system used for the system track's position.
position position_coordinate_type	The position of the system track.
velocity_coordinate_system coordinate_specification_type	The coordinate system used for the system track's velocity.
velocity velocity_coordinate_type	The velocity of the system track.
position_accuracy_coordinate_system coordinate_specification_type	The coordinate system used for the system track's position accuracy.
position_accuracy position_accuracy_coordinate_type	The position accuracy of the system track.
velocity_accuracy_coordinate_system coordinate_specification_type [0..1]	The coordinate system used for the system track's velocity accuracy.
velocity_accuracy velocity_accuracy_coordinate_type [0..1]	The velocity accuracy of the system track.
max_range_limit range_coordinate_type [0..1]	The estimated maximum range of the system track (for cases where the position coordinate does not specify range - i.e. bearing only).

7.4.17 Coordinates_and_Positions

Parent Package: Common_Types
Definitions of types to describe positions, in accordance with the ISO 19111 abstract model.

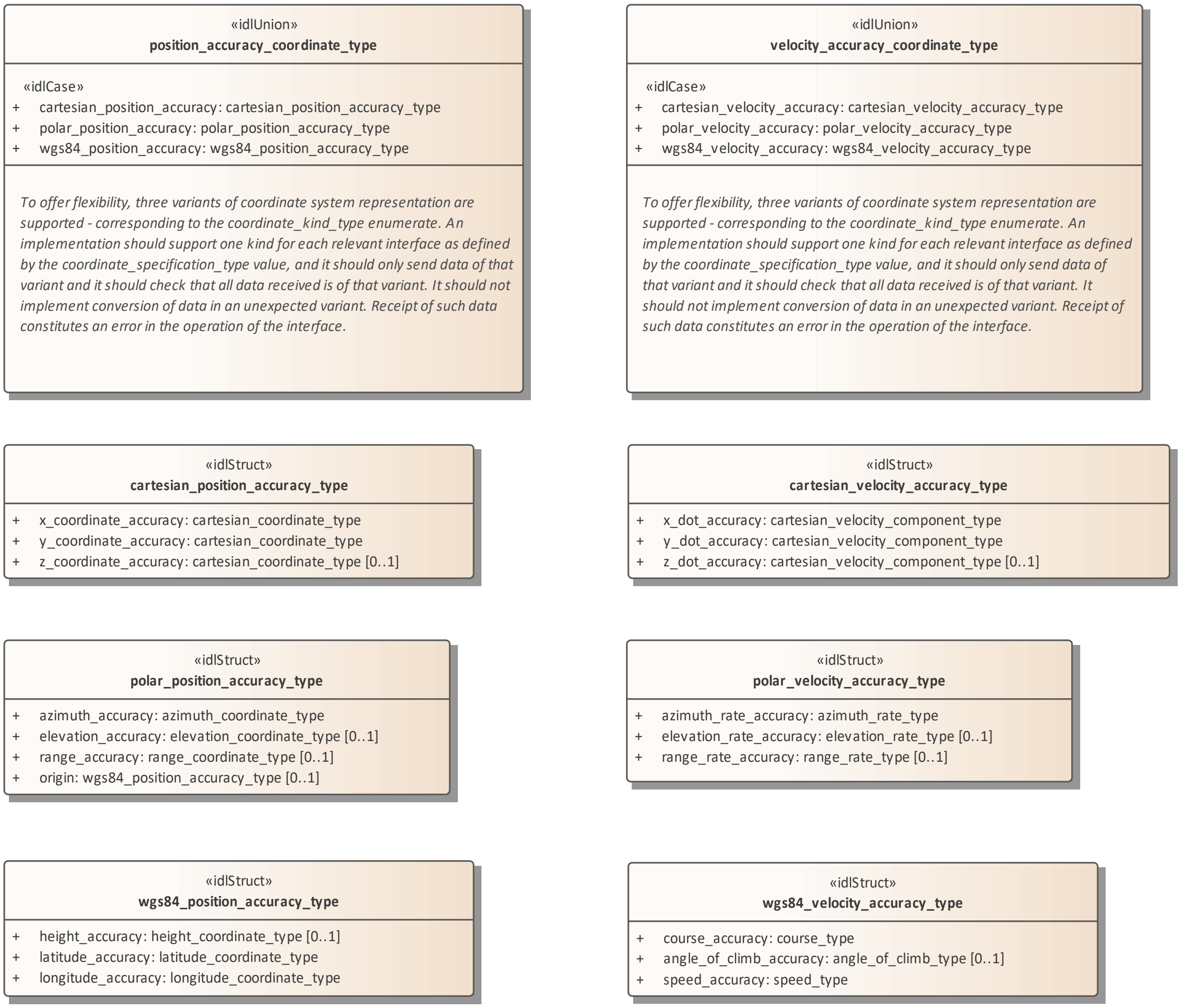


Figure 7.49 Accuracies (Class diagram)

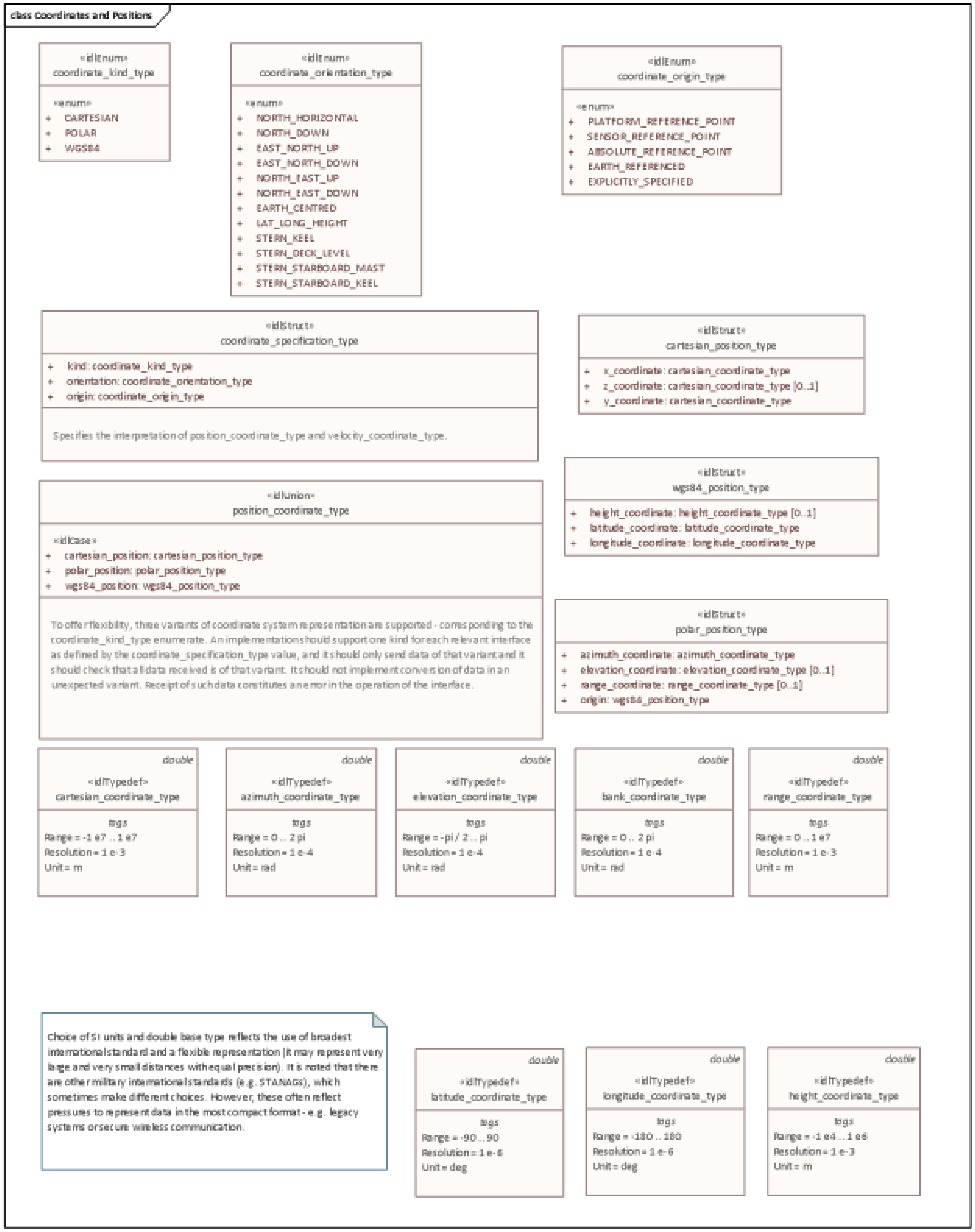


Figure 7.50 Coordinates and Positions (Class diagram)

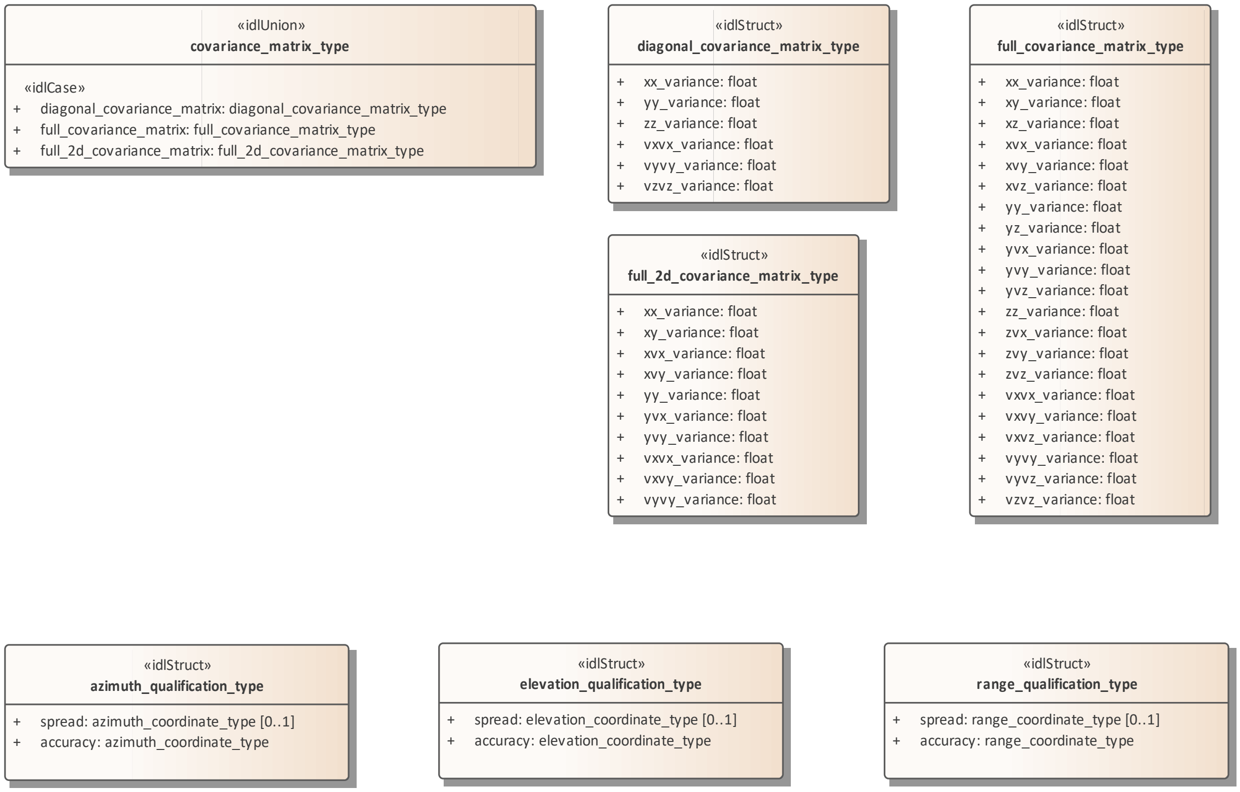


Figure 7.51 Covariance and Qualification (Class diagram)

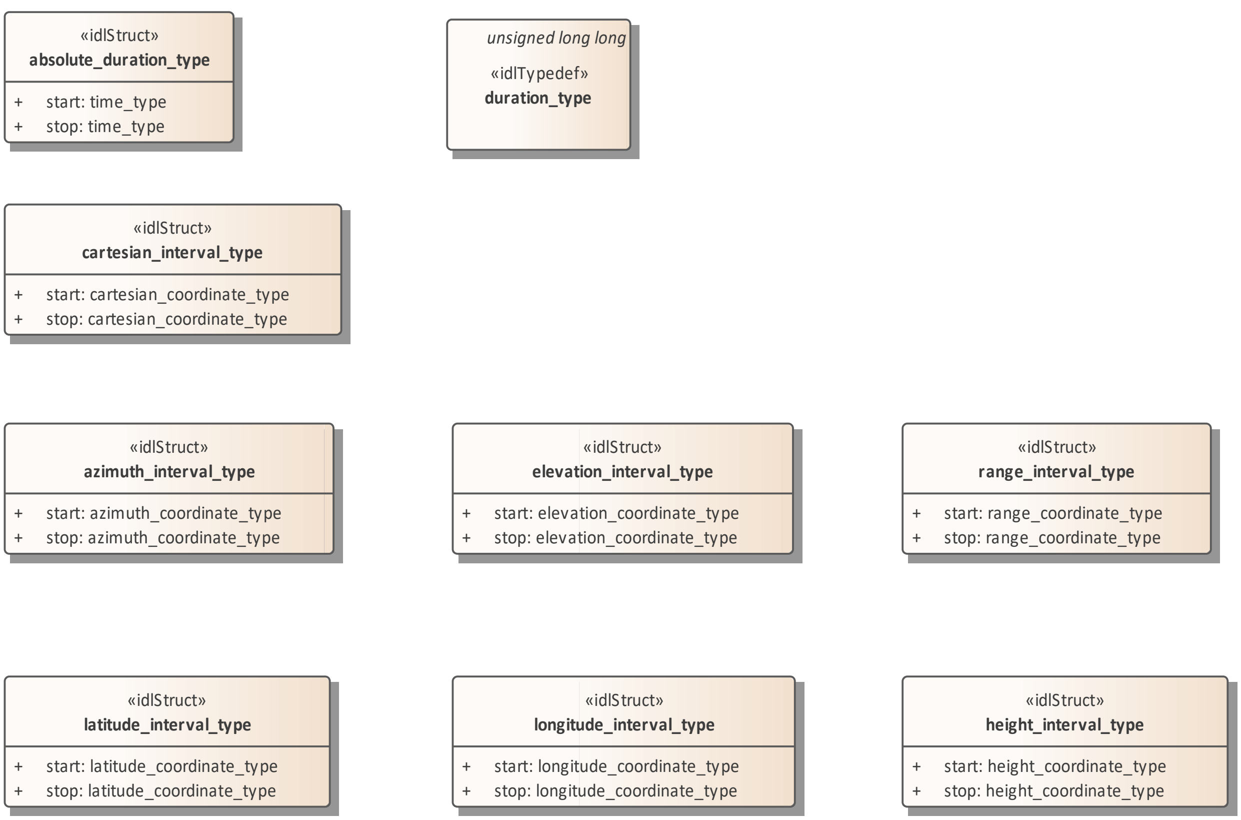


Figure 7.52 Intervals (Class diagram)

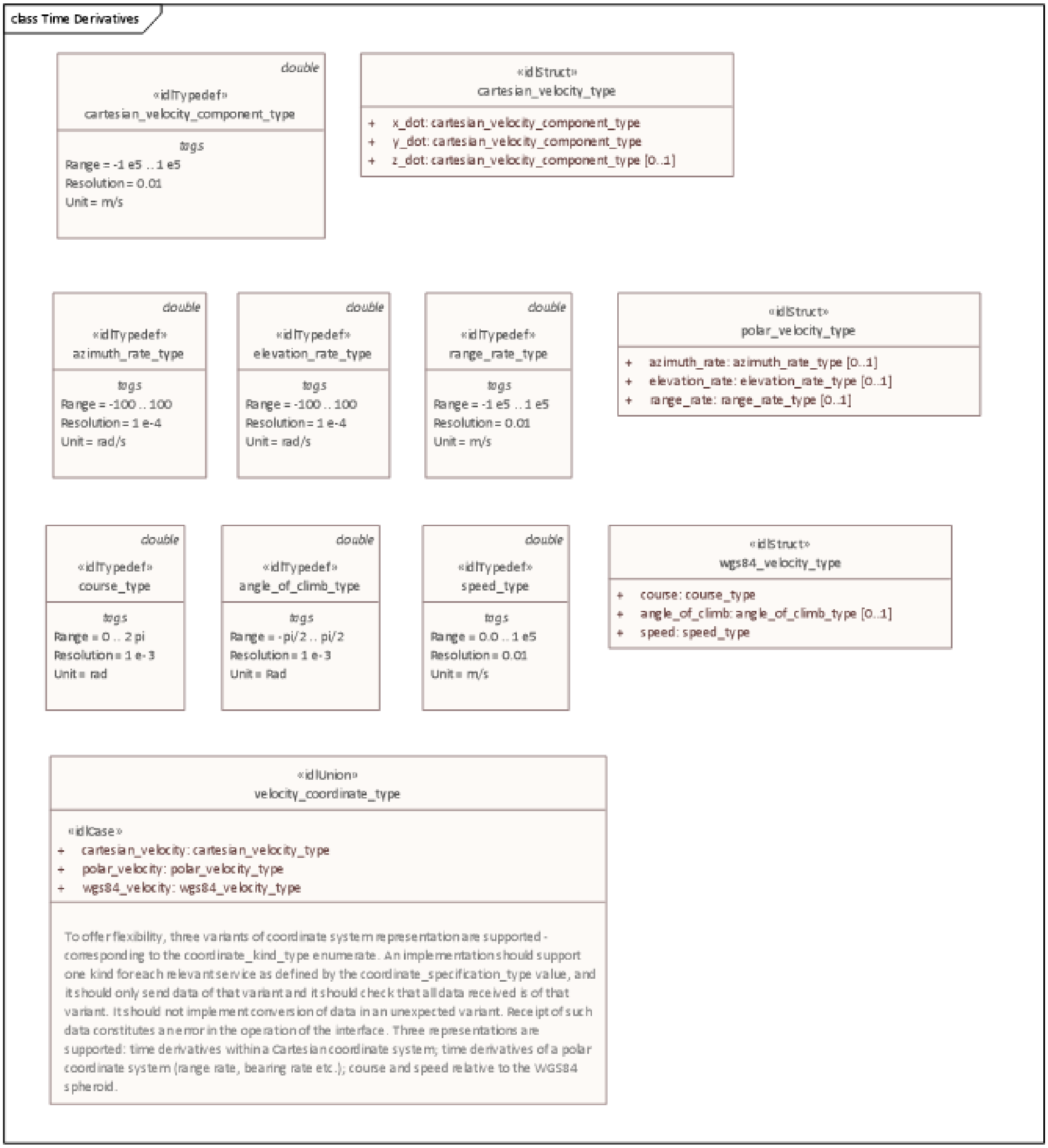


Figure 7.53 Time Derivatives (Class diagram)

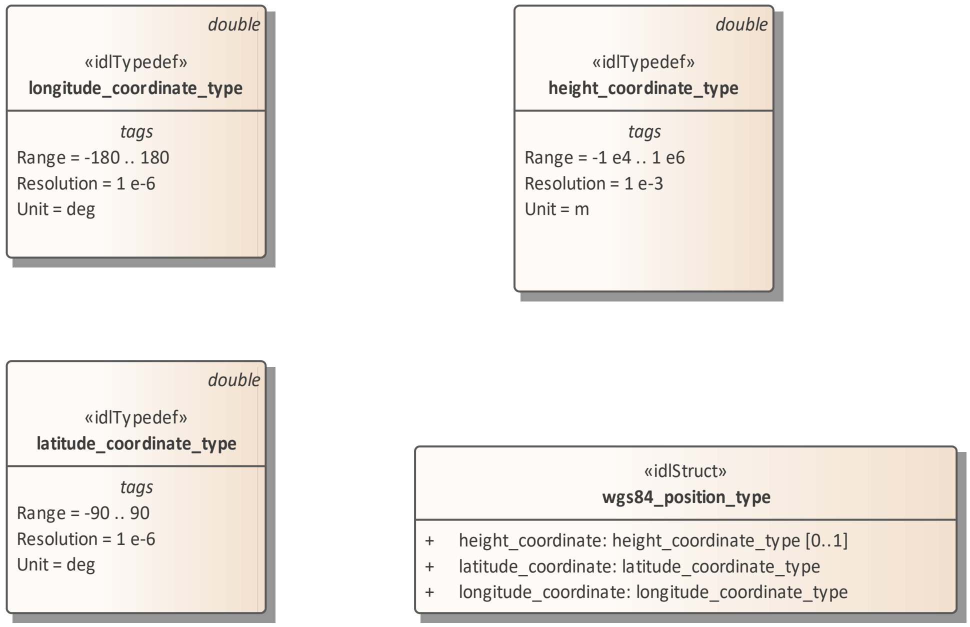


Figure 7.54 World Coordinates and Positions (Class diagram)

7.4.17.1 absolute_duration_type

Type: Class
Package: Coordinates_and_Positions
This class represents a duration fixed to an absolute point in time.

Table 7.6 - Attributes of Class absolute_duration_type

Attribute	Notes
start time_type	The earliest value at the beginning of the duration
stop time_type	The lateest value at the end of the duration

7.4.17.2 angle_of_climb_type

Type: Class
Package: Coordinates_and_Positions
The angle representing the direction of travel relative to the horizontal. Up is positive.
ElementTag: Range = -pi/2 .. pi/2
ElementTag: Resolution = 1 e-3
ElementTag: Unit = Rad

7.4.17.3 azimuth_coordinate_type

Type: Class
Package: Coordinates_and_Positions
Axis in the azimuth direction, i.e. horizontal angle from the associated coordinate system reference. Radians, positive clockwise from above.
See diagram note on choice of SI units
ElementTag: Range = 0 .. 2 pi
ElementTag: Resolution = 1 e-4
ElementTag: Unit = rad

7.4.17.4 azimuth_interval_type

Type: Class
Package: Coordinates_and_Positions
A set of azimuth values that are continuous on the circle. If start is greater than stop then the discontinuity due to angular wrap is included.

Table 7.7 - Attributes of Class azimuth_interval_type

Attribute	Notes
start azimuth_coordinate_type	The azimuth value at the beginning of the interval when progressing in a positive angular direction according to the orientation specified. This is not necessarily the lowest value as the discontinuity due to full-circle modularity may be included in the interval.
stop azimuth_coordinate_type	The azimuth value at the end of the interval when progressing in a positive angular direction according to the orientation specified. This is not necessarily the highest value as the discontinuity due to full-circle modularity may be included in the interval.

7.4.17.5 azimuth_qualification_type

Type: IDLStruct
Package: Coordinates_and_Positions
Qualifies a measurement of azimuth with attributes of accuracy and, if possible, variability.

Table 7.8 - Attributes of IDLStruct azimuth_qualification_type

Attribute	Notes
spread azimuth_coordinate_type [0..1]	The spread of the measurement. The combined measures of spread should encompass the full extent of the plot. This attribute is optional. Not all sensors are capable of measuring it.
accuracy azimuth_coordinate_type	The accuracy of the measurement; equal to one standard deviation of uncertainty.

7.4.17.6 azimuth_rate_type

Type: Class
Package: Coordinates_and_Positions
radians per second
ElementTag: Range = -100 .. 100
ElementTag: Resolution = 1 e-4
ElementTag: Unit = rad/s

7.4.17.7 bank_coordinate_type

Type: Class
Package: Coordinates_and_Positions
Axis in the bank direction, i.e. rotation angle from the associated coordinate system reference.
Radians, positive clockwise from behind.
ElementTag: Range = -pi .. pi
ElementTag: Resolution = 1 e-4
ElementTag: Unit = rad

7.4.17.8 cartesian_coordinate_type

Type: Class
Package: Coordinates_and_Positions
See diagram note on choice of SI units
ElementTag: Range = -1 e7 .. 1 e7
ElementTag: Resolution = 1 e-3
ElementTag: Unit = m

7.4.17.9 cartesian_interval_type

Type: IDLStruct
Package: Coordinates_and_Positions

Table 7.9 - Attributes of IDLStruct cartesian_interval_type

Attribute	Notes
start cartesian_coordinate_type	Lower valued, starting coordinate.
stop cartesian_coordinate_type	Higher valued, ending coordinate.

7.4.17.10 cartesian_position_type

Type: IDLStruct
Package: Coordinates_and_Positions
Coordinates in a Cartesian reference frame as described by a coordinate specification object

Table 7.10 - Attributes of IDLStruct cartesian_position_type

Attribute	Notes
x_coordinate cartesian_coordinate_type	The coordinate of the position on the x-axis
z_coordinate cartesian_coordinate_type [0..1]	The coordinate of the position on the z-axis. Optional as some sensors are 2D (horizontal plane or no elevation information)
y_coordinate cartesian_coordinate_type	The coordinate of the position on the y-axis

7.4.17.11 cartesian_velocity_component_type

Type: IDLTypeDef
Package: Coordinates_and_Positions

ElementTag: Range = -1 e5 .. 1 e5
ElementTag: Resolution = 0.01
ElementTag: Unit = m/s

7.4.17.12 cartesian_velocity_type

Type: IDLStruct
Package: Coordinates_and_Positions

Table 7.11 - Attributes of IDLStruct cartesian_velocity_type

Attribute	Notes
x_dot cartesian_velocity_component_type	The rate of change in the x-axis coordinate corresponding to the velocity

Attribute	Notes
y_dot cartesian_velocity_component_type	The rate of change in the y-axis coordinate corresponding to the velocity
z_dot cartesian_velocity_component_type [0..1]	The rate of change in the z-axis coordinate corresponding to the velocity

7.4.17.13 coordinate_kind_type

Type: IDLEnum
Package: Coordinates_and_Positions

Table 7.12 - Attributes of IDLEnum coordinate_kind_type

Attribute	Notes
«enum» **CARTESIAN**	Indicates a Cartesian Coordinate System
«enum» **POLAR**	Indicates a polar coordinate system
«enum» **WGS84**	Indicates a coordinate system based on the WGS-84 spheroid

7.4.17.14 coordinate_orientation_type

Type: IDLEnum
Package: Coordinates_and_Positions
This enumeration defines the set of coordinate systems, which compliant implementations may use. A compliant implementation may not fully support all of these coordinate systems.

Table 7.13 - Attributes of IDLEnum coordinate_orientation_type

Attribute	Notes
«enum» **NORTH_HORIZONTAL**	Valid for Polar Coordinate Kind Azimuth has origin (0.0) at North, positive clockwise, measured in the horizontal plane Elevation has origin (0.0) at the Horizontal, positive up, measured in the vertical plane.
«enum» **NORTH_DOWN**	Valid for Polar Coordinate Kind Azimuth has origin (0.0) at North, clockwise positive, measured in the horizontal plane Elevation has origin (0.0) when pointing directly down, and ~~180.0 degrees~~PI radians when pointing directly up, measured in the vertical plane.
«enum» **EAST_NORTH_UP**	Valid for Cartesian coordinate type x is positive to the East y is positive to the North z is positive up
«enum» **EAST_NORTH_DOWN**	Valid for Cartesian coordinate type x is positive to the East y is positive to the North z is positive down

Attribute	Notes
«enum» **NORTH_EAST_UP**	Valid for Cartesian coordinate type x is positive to the North y is positive to the East z is positive up
«enum» **NORTH_EAST_DOWN**	Valid for Cartesian coordinate type x is positive to the North y is positive to the East z is positive down
«enum» **EARTH_CENTRED**	Cartesian system with origin at centre of the Earth (absolute reference point) x positive through Greenwich meridian y positive through 90 degrees east (of Greenwich meridian) z positive through north pole x & y are in the equatorial plane
«enum» **LAT_LONG_HEIGHT**	WGS84 has unique well-defined orientation (NIMA Technical Report TR8350.2)
«enum» **STERN_KEEL**	Valid for Polar Coordinate Kind This is a platform orientation relative frame Azimuth has origin (0.0) in line with the ship's stern (heading), measured anti-clockwise Elevation has origin (0.0) when pointing directly down to the keel (perpendicular to the current inclination of the deck-level, not necessarily to the Earth's surface)
«enum» **STERN_DECK_LEVEL**	Valid for Polar Coordinate Kind This is a platform orientation relative frame Azimuth has origin (0.0) in line with the ship's stern (heading), measured anti-clockwise Elevation has origin (0.0) when pointing parallel to the deck-level (not necessarily parallel to the Earth's surface)
«enum» **STERN_STARBOARD_MAST**	Valid for Cartesian coordinate type This is a platform orientation relative frame x is positive towards the stern (negative to bow) y is positive to starboard (negative to port) z is positive towards the mast (negative to keel)
«enum» **STERN_STARBOARD_KEEL**	Valid for Cartesian coordinate type This is a platform orientation relative frame x is positive towards the stern (negative to bow) y is positive to starboard (negative to port) z is positive towards the keel (negative to mast)

7.4.17.15 coordinate_origin_type

Type: IDLEnum
Package: Coordinates_and_Positions

Table 7.14 - Attributes of IDLEnum coordinate_origin_type

Attribute	Notes
«enum» **PLATFORM_REFERENCE_POINT**	The origin of the coordinate system is 'well known' reference point for the platform (on which the CMS and subsystem reside)
«enum» **SENSOR_REFERENCE_POINT**	The origin for the coordinate system is the 'well known' reference/datum point for the sensor, which is interacting using the interface.
«enum» **ABSOLUTE_REFERENCE_POINT**	The origin for the coordinate system is a fixed point in Earth (WGS84) coordinates. This point is known to the CMS and Subsystems using the interface by means beyond the scope of the interface.
«enum» **EARTH_REFERENCED**	This value signifies that the origin for the coordinate system is well-defined with respect to the Earth by the coordinate system. E.g. centre of the Earth for Earth-Centred Earth-Fixed or the WGS84 spheroid for WGS84
«enum» **EXPLICITLY_SPECIFIED**	This value signifies that the origin is explicitly specified within the data model by the producer of the data.

7.4.17.16 coordinate_specification_type

Type: IDLStruct
Package: Coordinates_and_Positions
Specifies the interpretation of position_coordinate_type and velocity_coordinate_type.

Table 7.15 - Attributes of IDLStruct coordinate_specification_type

Attribute	Notes
kind coordinate_kind_type	The kind of coordinate system used.
orientation coordinate_orientation_type	The orientation convention used by the coordinates
origin coordinate_origin_type	The meaning of the coordinate origin.

7.4.17.17 course_type

Type: Class
Package: Coordinates_and_Positions
The angle representing the direction of travel relative to North in the horizontal plane. Clockwise (facing down) is positive.
ElementTag: Range = 0 .. 2 pi
ElementTag: Resolution = 1 e-3
ElementTag: Unit = rad

7.4.17.18 covariance_matrix_type

Type: Class
Package: Coordinates_and_Positions
This class represents a covariance matrix for coordinate estimates and their time derivatives through a choice of formats.

Table 7.16 - Attributes of Class covariance_matrix_type

Attribute	Notes
«idlCase» **diagonal_covariance_matrix** diagonal_covariance_matrix_type	The diagonal matrix option
«idlCase» **full_covariance_matrix** full_covariance_matrix_type	the full covariance option
«idlCase» **full_2d_covariance_matrix** full_2d_covariance_matrix_type	the full 2d covariance option

7.4.17.19 diagonal_covariance_matrix_type

Type: Class
Package: Coordinates_and_Positions
Covariance of just the diagonal elements (i.e. the variance of the coordinate estimates).

Table 7.17 - Attributes of Class diagonal_covariance_matrix_type

Attribute	Notes
xx_variance float	The variance of the x coordinate value
yy_variance float	The variance of the y coordinate value
zz_variance float	The variance of the z coordinate value
vxvx_variance float	The variance of the x component of velocity
vyvy_variance float	The variance of the y component of velocity
vzvz_variance float	The variance of the z component of velocity

7.4.17.20 duration_type

Type: Class
Package: Coordinates_and_Positions
The length of a time interval (not fixed to an absolute point in time).
unit: 100 nano seconds

7.4.17.21 elevation_coordinate_type

Type: Class
Package: Coordinates_and_Positions
Axis in the direction of elevation, i.e. vertical angle from the associated coordinate system datum, radians, positive up.
See diagram note on choice of SI un
ElementTag: Range = -pi / 2 .. pi~~/ 2~~
ElementTag: Resolution = 1 e-4
ElementTag: Unit = rad

7.4.17.22 elevation_interval_type

Type: Class
Package: Coordinates_and_Positions

Table 7.18 - Attributes of Class elevation_interval_type

Attribute	Notes
start elevation_coordinate_type	The lower starting elevation value.

Attribute	Notes
stop elevation_coordinate_type	The higher ending elevation value.

7.4.17.23 elevation_qualification_type

Type: IDLStruct
Package: Coordinates_and_Positions
Qualifies a measurement of elevation with attributes of accuracy and, if possible, variability.

Table 7.19 - Attributes of IDLStruct elevation_qualification_type

Attribute	Notes
spread elevation_coordinate_type [0..1]	The spread of the measurement. The combined measures of spread should encompass the full extent of the plot. This attribute is optional. Not all sensors are capable of measuring it.
accuracy elevation_coordinate_type	The accuracy of the measurement; equal to one standard deviation of uncertainty.

7.4.17.24 elevation_rate_type

Type: Class
Package: Coordinates_and_Positions
radians per second
ElementTag: Range = -100 .. 100
ElementTag: Resolution = 1 e-4
ElementTag: Unit = rad/s

7.4.17.25 full_2d_covariance_matrix_type

Type: IDLStruct
Package: Coordinates_and_Positions
The full covariance terms (in triangular form as necessarily a symmetric matrix) for reports in just the x and y dimensions.

Table 7.20 - Attributes of IDLStruct full_2d_covariance_matrix_type

Attribute	Notes
xx_variance float	The variance of the x coordinate value
xy_variance float	The covariance of the x coordinate with the y coordinate.
xvx_variance float	The covariance of the x coordinate with the x velocity coordinate.
xvy_variance float	The covariance of the x coordinate with the y velocity coordinate.
yy_variance float	The variance of the y coordinate value
yvx_variance float	The covariance of the y coordinate with the x velocity coordinate.
yvy_variance float	The covariance of the y coordinate with the y velocity coordinate.

Attribute	Notes
vxvx_variance float	The variance of the x component of velocity
vxvy_variance float	The covariance of the x velocity coordinate with the y velocity coordinate.
vyvy_variance float	The variance of the y component of velocity

7.4.17.26 full_covariance_matrix_type

Type: Class
Package: Coordinates_and_Positions
Triangular representation of a full covariance matrix (which is by definition symmetric).

Table 7.21 - Attributes of Class full_covariance_matrix_type

Attribute	Notes
xx_variance float	The variance of the x coordinate value
xy_variance float	The covariance of the x coordinate with the y coordinate.
xz_variance float	The covariance of the x coordinate with the z coordinate.
xvx_variance float	The covariance of the x coordinate with the x velocity coordinate.
xvy_variance float	The covariance of the x coordinate with the y velocity coordinate.
xvz_variance float	The covariance of the x coordinate with the z velocity coordinate.
yy_variance float	The variance of the y coordinate value
yz_variance float	The covariance of the y coordinate with the z coordinate.
yvx_variance float	The covariance of the y coordinate with the x velocity coordinate.
yvy_variance float	The covariance of the y coordinate with the y velocity coordinate.
yvz_variance float	The covariance of the y coordinate with the z velocity coordinate.
zz_variance float	The variance of the z coordinate value
zvx_variance float	The covariance of the z coordinate with the x velocity coordinate.
zvy_variance float	The covariance of the z coordinate with the y velocity coordinate.
zvz_variance float	The covariance of the z coordinate with the z velocity coordinate.

Attribute	Notes
vxvx_variance float	The variance of the x component of velocity
vxvy_variance float	The covariance of the x velocity coordinate with the y velocity coordinate.
vxvz_variance float	The covariance of the x velocity coordinate with the z velocity coordinate.
vyvy_variance float	The variance of the y component of velocity
vyvz_variance float	The covariance of the y velocity coordinate with the z velocity coordinate.
vzvz_variance float	The variance of the z component of velocity

7.4.17.27 height_coordinate_type

Type: Class
Package: Coordinates_and_Positions
For positive values, height above coordinate system ellipsoid, for negative values, depth below; measured in meters.
This quantity is height as a measured distance rather than an inference from (for instance) barometric pressure.
See diagram note on choice of SI units
ElementTag: Range = -1 e4 .. 1 e6
ElementTag: Resolution = 1 e-3
ElementTag: Unit = m

7.4.17.28 height_interval_type

Type: Class
Package: Coordinates_and_Positions

Table 7.22 - Attributes of Class height_interval_type

Attribute	Notes
start height_coordinate_type	The lower, starting height value
stop height_coordinate_type	The higher, ending height value

7.4.17.29 latitude_coordinate_type

Type: Class
Package: Coordinates_and_Positions
Degrees north (positive), south (negative) relative to coordinate system datum.
See diagram note on choice of SI units
ElementTag: Range = -90 .. 90
ElementTag: Resolution = 1 e-6
ElementTag: Unit = deg

7.4.17.30 latitude_interval_type

Type: Class
Package: Coordinates_and_Positions

Table 7.23 - Attributes of Class latitude_interval_type

Attribute	Notes
start latitude_coordinate_type	Lower valued starting latitude
stop latitude_coordinate_type	Higher valued ending latitude

7.4.17.31 longitude_coordinate_type

Type: Class
Package: Coordinates_and_Positions
Degrees east (positive), west (negative) relative to coordinate system datum.
See diagram note on choice of SI units
ElementTag: Range = -180 .. 180
ElementTag: Resolution = 1 e-6
ElementTag: Unit = deg

7.4.17.32 longitude_interval_type

Type: Class
Package: Coordinates_and_Positions
A range of longitude values

Table 7.24 - Attributes of Class longitude_interval_type

Attribute	Notes
start longitude_coordinate_type	The lowest longitude value at the beginning of the interval
stop longitude_coordinate_type	The highest longitude value at the end of the interval

7.4.17.33 polar_position_type

Type: IDLStruct
Package: Coordinates_and_Positions
Coordinates in a polar reference frame as a described by a coordinate specification object

Table 7.25 - Attributes of IDLStruct polar_position_type

Attribute	Notes
azimuth_coordinate azimuth_coordinate_type	The coordinate in the azimuth plane.
elevation_coordinate elevation_coordinate_type [0..1]	Optional as some sensors provide no elevation information
range_coordinate range_coordinate_type [0..1]	Optional as some sensors provide no range information (e.g. most passive sensors)
origin wgs84_position_type	Specifies the origin from which to interpret the polar position. This attribute is mandatory so that the originator of the data unambiguously specifies the origin (this is prioritizing accuracy and integration). AttributeTag: Issue =

7.4.17.34 polar_velocity_type

Type: IDLStruct
Package: Coordinates_and_Positions
Velocity defined in a polar reference frame as a described by a coordinate specification object

Table 7.26 - Attributes of IDLStruct polar_velocity_type

Attribute	Notes
azimuth_rate azimuth_rate_type [0..1]	The rate of change in azimuth corresponding to t velocity. Optional as some sensors do not report velocity
elevation_rate elevation_rate_type [0..1]	The rate of change in elevation corresponding to the velocity. Optional as some sensors provide no elevation information
range_rate range_rate_type [0..1]	The rate of change in range corresponding to the velocity. Optional as some sensors provide no range information (e.g. most passive sensors)

7.4.17.35 position_accuracy_coordinate_type

Type: Class
Package: Coordinates_and_Positions
To offer flexibility, three variants of coordinate system representation are supported - corresponding to the coordinate_kind_type enumerate. An implementation should support one kind for each relevant interface as defined by the coordinate_specification_type value, and it should only send data of that variant and it should check that all data received is of that variant. It should not implement conversion of data in an unexpected variant. Receipt of such data constitutes an error in the operation of the interface.

Table 7.27 - Attributes of Class position_accuracy_coordinate_type

Attribute	Notes
«idlCase» **cartesian_position_accuracy** cartesian_position_accuracy_type	The Cartesian accuracy option.
«idlCase» **polar_position_accuracy** polar_position_accuracy_type	The polar accuracy option.
«idlCase» **wgs84_position_accuracy** wgs84_position_accuracy_type	The accuracy option using the WGS-84 spheroid.

7.4.17.36 position_coordinate_type

Type: IDLUnion
Package: Coordinates_and_Positions
To offer flexibility, three variants of coordinate system representation are supported - corresponding to the coordinate_kind_type enumerate. An implementation should support one kind for each relevant interface as defined by the coordinate_specification_type value, and it should only send data of that variant and it should check that all data received is of that variant. It should not implement conversion of data in an unexpected variant. Receipt of such data constitutes an error in the operation of the interface.
ElementTag: case type = coordinate_kind_type

Table 7.28 - Attributes of IDLUnion position_coordinate_type

Attribute	Notes
«idlCase» **cartesian_position** cartesian_position_type	The Cartesian coordinate position option AttributeTag: case value = CARTESIAN
«idlCase» **polar_position** polar_position_type	The polar coordinates position option AttributeTag: case value = POLAR
«idlCase» **wgs84_position** wgs84_position_type	The position option using the WGS-84 spheroid. AttributeTag: case value = WGS84

7.4.17.37 range_coordinate_type

Type: Class
Package: Coordinates_and_Positions
Axis in range, i.e. linear distance from the coordinate system datum. Metres.
See diagram note on choice of SI units
ElementTag: Range = 0 .. 1 e7
ElementTag: Resolution = 1 e-3
ElementTag: Unit = m

7.4.17.38 range_interval_type

Type: Class
Package: Coordinates_and_Positions
A continuous set of range values.

Table 7.29 - Attributes of Class range_interval_type

Attribute	Notes
start range_coordinate_type	The nearest value at the beginning of the interval
stop range_coordinate_type	The furthest value at the end of the interval

7.4.17.39 range_qualification_type

Type: IDLStruct
Package: Coordinates_and_Positions
Qualifies a measurement of range with attributes of accuracy and, if possible, variability.

Table 7.30 - Attributes of IDLStruct range_qualification_type

Attribute	Notes
spread range_coordinate_type [0..1]	The spread of the measurement. The combined measures of spread should encompass the full extent of the plot. This attribute is optional. Not all sensors are capable of measuring it.
accuracy range_coordinate_type	The accuracy of the measurement; equal to one standard deviation of uncertainty.

7.4.17.40 range_rate_type

Type: Class
Package: Coordinates_and_Positions
metres per second
ElementTag: Range = ~~0.0~~-1 e5 .. 1 e5
ElementTag: Resolution = 0.01
ElementTag: Unit = m/s

7.4.17.41 speed_interval_type

Type: Class
Package: Coordinates_and_Positions
This class represents a range of speeds.

Table 7.31 - Attributes of Class speed_interval_type

Attribute	Notes
min speed_type	The minimum speed.

Attribute	Notes
max speed_type	The maximum speed.

7.4.17.42 speed_type

Type: Class
Package: Coordinates_and_Positions
metres per second
ElementTag: Range = 0.0 .. 1 e5
ElementTag: Resolution = 0.01
ElementTag: Unit = m/s

7.4.17.43 velocity_accuracy_coordinate_type

Type: Class
Package: Coordinates_and_Positions
To offer flexibility, three variants of coordinate system representation are supported - corresponding to the coordinate_kind_type enumerate. An implementation should support one kind for each relevant interface as defined by the coordinate_specification_type value, and it should only send data of that variant and it should check that all data received is of that variant. It should not implement conversion of data in an unexpected variant. Receipt of such data constitutes an error in the operation of the interface.

Table 7.32 - Attributes of Class velocity_accuracy_coordinate_type

Attribute	Notes
«idlCase» **cartesian_velocity_accuracy** cartesian_velocity_accuracy_type	The Cartesian velocity accuracy option.
«idlCase» **polar_velocity_accuracy** polar_velocity_accuracy_type	The polar velocity accuracy option.
«idlCase» **wgs84_velocity_accuracy** wgs84_velocity_accuracy_type	The velocity accuracy option using the WGS-84 spheroid.

7.4.17.44 velocity_coordinate_type

Type: IDLUnion
Package: Coordinates_and_Positions
To offer flexibility, three variants of coordinate system representation are supported - corresponding to the coordinate_kind_type enumerate. An implementation should support one kind for each relevant service as defined by the coordinate_specification_type value, and it should only send data of that variant and it should check that all data received is of that variant. It should not implement conversion of data in an unexpected variant. Receipt of such data constitutes an error in the operation of the interface. Three representations are supported: time derivatives within a Cartesian coordinate system; time derivatives of a polar coordinate system (range rate, bearing rate etc.); course and speed relative to the WGS84 spheroid.
ElementTag: case type = coordinate_kind_type

Table 7.33 - Attributes of IDLUnion velocity_coordinate_type

Attribute	Notes
«idlCase» **cartesian_velocity** cartesian_velocity_type	The Cartesian velocity option AttributeTag: case value = CARTESIAN
«idlCase» **polar_velocity** polar_velocity_type	The polar velocity option AttributeTag: case value = POLAR

Attribute	Notes
«idlCase» **wgs84_velocity** wgs84_velocity_type	The option of velocity specified with reference to the WGS-84 spheroid. AttributeTag: case value = WGS84

7.4.17.45 wgs84_position_type

Type: Class
Package: Coordinates_and_Positions
Coordinate in the WGS84 reference system.

Table 7.34 - Attributes of Class wgs84_position_type

Attribute	Notes
height_coordinate height_coordinate_type [0..1]	Optional as some sensors as 2D (work in horizontal plane) and some other functions do not supply or require this information either.
latitude_coordinate latitude_coordinate_type	The latitude of the position on the WGS-84 spheroid.
longitude_coordinate longitude_coordinate_type	The longitude of the position on the WGS-84 spheroid.

7.4.17.46 wgs84_velocity_type

Type: IDLStruct
Package: Coordinates_and_Positions
Velocity defined in the WGS84 grid system from the viewpoint of the object in terms of course and speed with optional angle of climb for changes in height.

Table 7.35 - Attributes of IDLStruct wgs84_velocity_type

Attribute	Notes
course course_type	Relative to North in the WGS84 spheroid.
angle_of_climb angle_of_climb_type [0..1]	Optional as some sensors as 2D (work in horizontal plane) and some other functions do not supply or require this information either.
speed speed_type	The total speed within the WGS84 spheroid (not speed over ground) in the direction of travel including angle of climb when present.

7.4.17.47 cartesian_position_accuracy_type

Type: Class
Package: Coordinates_and_Positions
The accuracy of the components of Cartesian position

Table 7.36 - Attributes of Class cartesian_position_accuracy_type

Attribute	Notes
x_coordinate_accuracy cartesian_coordinate_type	The accuracy of the x-axis coordinate
y_coordinate_accuracy cartesian_coordinate_type	The accuracy of the y-axis coordinate

Attribute	Notes
z_coordinate_accuracy cartesian_coordinate_type [0..1]	The accuracy of the z-axis coordinate. Optional as some sensors are 2D (horizontal plane or no elevation information)

7.4.17.48 cartesian_velocity_accuracy_type

Type: Class
Package: Coordinates_and_Positions
The accuracy of the components of Cartesian velocity

Table 7.37 - Attributes of Class cartesian_velocity_accuracy_type

Attribute	Notes
x_dot_accuracy cartesian_velocity_component_type	Accuracy of the x_dot velocity attribute
y_dot_accuracy cartesian_velocity_component_type	Accuracy of the y_dot velocity attribute
z_dot_accuracy cartesian_velocity_component_type [0..1]	Accuracy of the z_dot velocity attribute. Optional as some sensors are 2D (horizontal plane or no elevation information)

7.4.17.49 polar_position_accuracy_type

Type: Class
Package: Coordinates_and_Positions
The accuracy of the components of polar position

Table 7.38 - Attributes of Class polar_position_accuracy_type

Attribute	Notes
azimuth_accuracy azimuth_coordinate_type	The accuracy of the azimuth coordinate.
elevation_accuracy elevation_coordinate_type [0..1]	Optional as some sensors provide no elevation information
range_accuracy range_coordinate_type [0..1]	Optional as some sensors provide no range information (e.g. most passive sensors)
origin wgs84_position_accuracy_type [0..1]	Specifies the accuracy of the origin from which to interpret the polar position. This attribute is optional as the origin can be implicitly specified according to the value of the applicable coordinate specification enumeration.

7.4.17.50 polar_velocity_accuracy_type

Type: Class
Package: Coordinates_and_Positions
The accuracy of the components of polar velocity

Table 7.39 - Attributes of Class polar_velocity_accuracy_type

Attribute	Notes
azimuth_rate_accuracy azimuth_rate_type	The accuracy of the azimuth rate

Attribute	Notes
elevation_rate_accuracy elevation_rate_type [0..1]	The accuracy of the elevation rate. Optional as some sensors provide no elevation information
range_rate_accuracy range_rate_type [0..1]	The accuracy of the range rate. Optional as some sensors provide no range information (e.g. most passive sensors)

7.4.17.51 wgs84_position_accuracy_type

Type: Class
Package: Coordinates_and_Positions
The accuracy of the components of a WGS84 position

Table 7.40 - Attributes of Class wgs84_position_accuracy_type

Attribute	Notes
height_accuracy height_coordinate_type [0..1]	The accuracy of the height coordinate. Optional as some sensors as 2D (work in horizontal plane) and some other functions do not supply or require this information either.
latitude_accuracy latitude_coordinate_type	The accuracy of the latitude coordinate.
longitude_accuracy longitude_coordinate_type	The accuracy of the longitude coordinate.

7.4.17.52 wgs84_velocity_accuracy_type

Type: Class
Package: Coordinates_and_Positions
The accuracy of the components of a WGS84 velocity

Table 7.41 - Attributes of Class wgs84_velocity_accuracy_type

Attribute	Notes
course_accuracy course_type	The accuracy of the course attribute of the velocity
angle_of_climb_accuracy angle_of_climb_type [0..1]	The accuracy of the angle of climb attribute of the velocity. Optional as some sensors as 2D (work in horizontal plane) and some other functions do not supply or require this information either.
speed_accuracy speed_type	The accuracy of the speed attribute of the velocity

7.4.18 Shape_Model

Parent Package: Common_Types

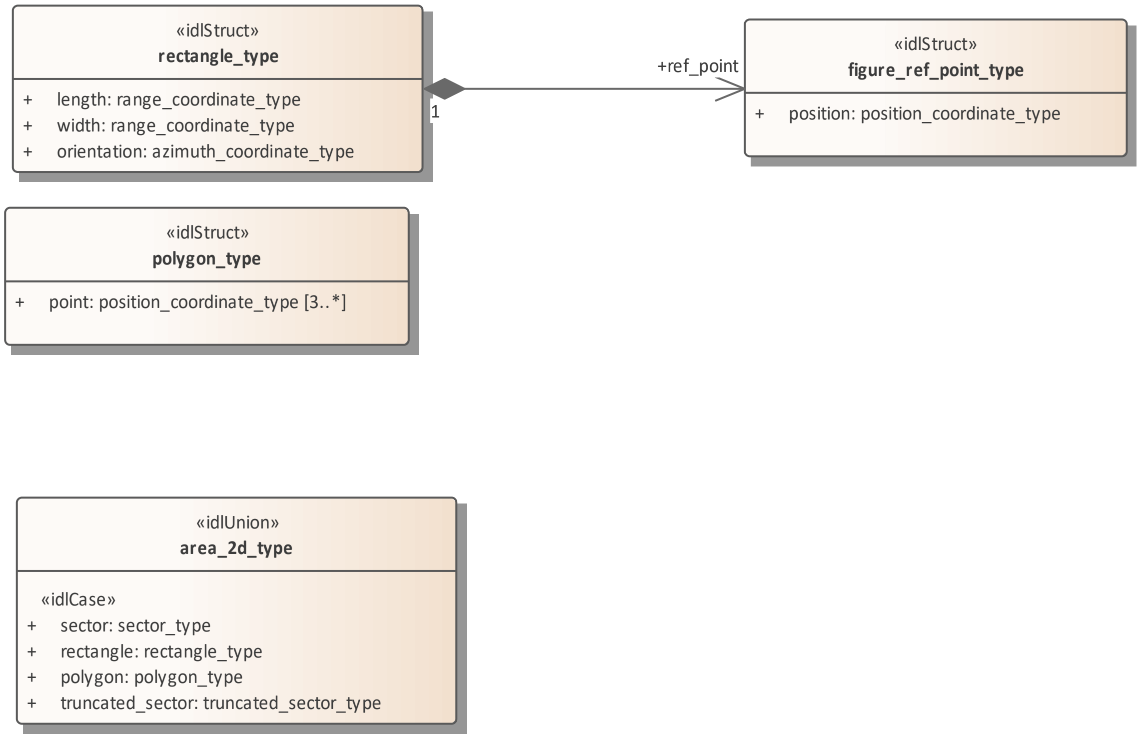


Figure 7.55 Domain Model - non polar (Class diagram)

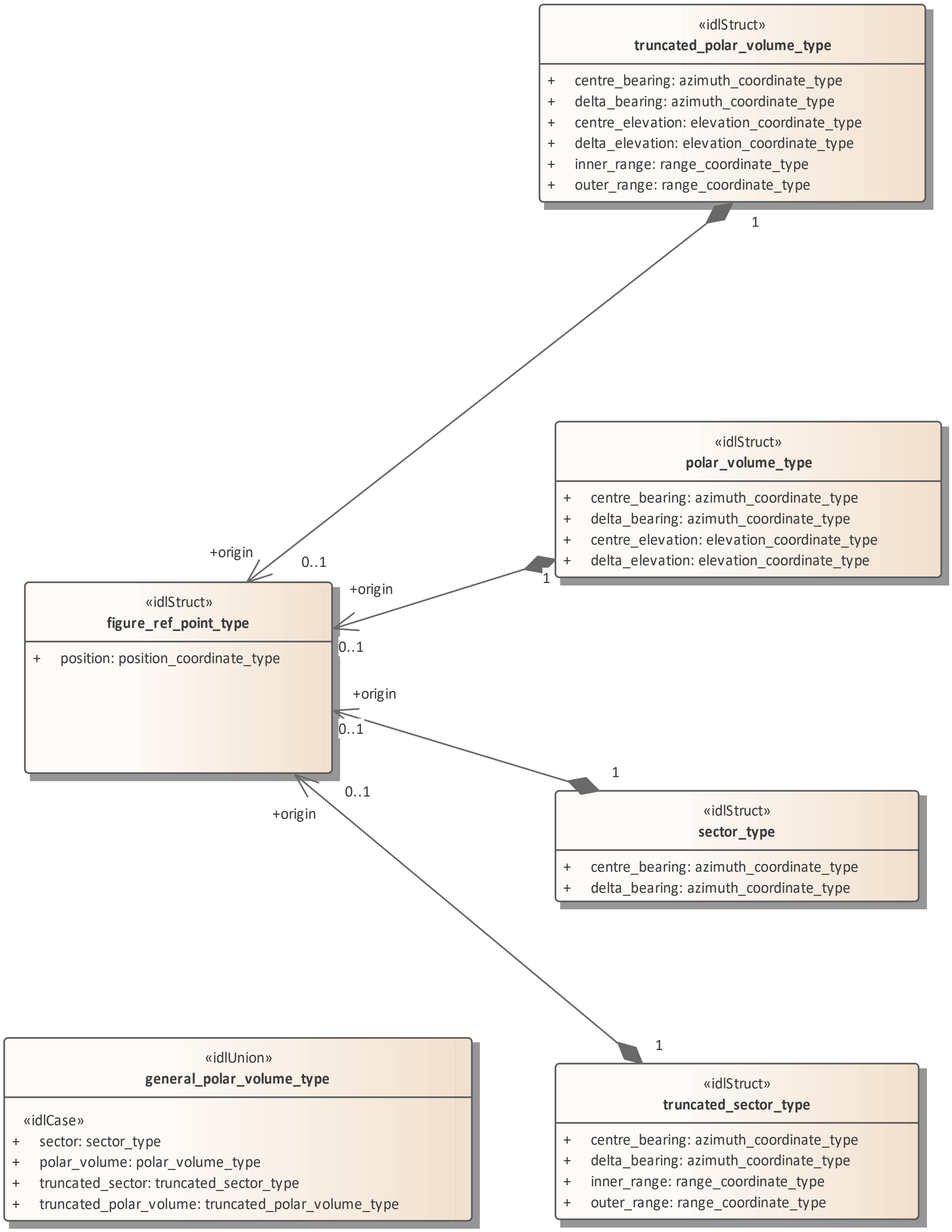


Figure 7.56 Domain Model - polar (Class diagram)

7.4.18.1 area_2d_type

Type: IDLUnion

Package: Shape_Model
An area for the sensor to keep under surveillance
ElementTag: switchType = long

Table 7.42 - Attributes of IDLUnion area_2d_type

Attribute	Notes
«idlCase» **sector** sector_type	The sector option for a 2d area
«idlCase» **rectangle** rectangle_type	The rectangle option for a 2d area
«idlCase» **polygon** polygon_type	The polygon option for a 2d area
«idlCase» **truncated_sector** truncated_sector_type	The truncated sector option for a 2d area

7.4.18.2 figure_ref_point_type

Type: Class
Package: Shape_Model
A figure_ref_point specifies a reference point for a figure.
This reference point is a mathematically meaningful point of the figure. For a circle it is the centre of the circle, for a polygon it is the centre of gravity of the polygon, etc.

When rotating the figure, the figure_ref_point acts as the rotation point.

When a figure is not slaved to a track its figure_ref_point shall be mapped on a (moving) geo point.
When the figure is slaved to an object (track, point) its figure_ref_point shall be mapped on an offset position which is relative to the master object.

Table 7.43 - Attributes of Class figure_ref_point_type

Attribute	Notes
position position_coordinate_type	The position of the reference point.

7.4.18.3 general_polar_volume_type

Type: IDLUnion
Package: Shape_Model
This class allow definition of a volume in space, bounded by standard polar coordinates (azimuth, elevation and range). The different options allow the dimension of either range, elevation or both to be omitted.

Table 7.44 - Attributes of IDLUnion general_polar_volume_type

Attribute	Notes
«idlCase» **sector** sector_type	The general polar volume is a sector
«idlCase» **polar_volume** polar_volume_type	The general polar volume is a polar volume
«idlCase» **truncated_sector** truncated_sector_type	The general polar volume is a truncated sector
«idlCase» **truncated_polar_volume** truncated_polar_volume_type	The general polar volume is a truncated polar volume.

7.4.18.4 polar_volume_type

Type: Class
Package: Shape_Model

A polar_volume specifies a 3D volume based on a horizontal plane by means of its origin, its centre bearing and centre elevation, its bearing delta and elevation delta.
The origin is the figure reference point of the Polar Volume.

Table 7.45 - Attributes of Class polar_volume_type

Attribute	Notes
centre_bearing azimuth_coordinate_type	This attribute specifies the horizontal angle measured clockwise between the Y-axis of the relevant coordinate system (true north, heading/course) and the centre bearing line of the volume.
delta_bearing azimuth_coordinate_type	This attribute specifies the bearing delta on each side of a specified centre bearing line.
centre_elevation elevation_coordinate_type	This attribute specifies the vertical angle measured counterclockwise between the horizontal plane and the centre elevation line of the volume.
delta_elevation elevation_coordinate_type	This attribute specifies the elevation delta on each side of a specified centre elevation line.

7.4.18.5 polygon_type

Type: IDLStruct
Package: Shape_Model
A geographically defined general area

Table 7.46 - Attributes of IDLStruct polygon_type

Attribute	Notes
point position_coordinate_type [3..*]	The set of points for the polygon; there must be at least 3. AttributeTag: Length = 12

7.4.18.6 rectangle_type

Type: IDLStruct
Package: Shape_Model
A geographically defined rectangle in the environment

Table 7.47 - Attributes of IDLStruct rectangle_type

Attribute	Notes
length range_coordinate_type	distance along angle of orientation from the ref point to the next corner
width range_coordinate_type	distance perpendicular to angle of orientation (clockwise) from ref point to the next corner
orientation azimuth_coordinate_type	angle of azimuth of the length sides of the rectangle

7.4.18.7 sector_type

Type: Class
Package: Shape_Model

A sector specifies a 2D area in a horizontal plane by means of its origin, its centre bearing with its bearing delta, that together define the sector.
The origin is the figure reference point of the sector.
In case the sector is north oriented, the centre bearing is specified with respect to true north; otherwise it is specified with respect to the object's (own ship/other track, point) heading/course.

Table 7.48 - Attributes of Class sector_type

Attribute	Notes
centre_bearing azimuth_coordinate_type	This attribute specifies the horizontal angle measured clockwise between the Y-axis of the relevant coordinate system (true north, heading/course) and the centre bearing line of the sector.
delta_bearing azimuth_coordinate_type	This attribute specifies the bearing delta on each side of a specified centre bearing line.

7.4.18.8 truncated_polar_volume_type

Type: Class
Package: Shape_Model
A truncated_polar_volume specifies a 3D volume based on a horizontal plane by means of its origin, its centre bearing and centre elevation, its bearing delta and elevation delta, its inner range and outer range

Table 7.49 - Attributes of Class truncated_polar_volume_type

Attribute	Notes
centre_bearing azimuth_coordinate_type	This attribute specifies the horizontal angle measured clockwise between the Y-axis of the relevant coordinate system (true north, heading/course) and the centre bearing line of the volume.
delta_bearing azimuth_coordinate_type	This attribute specifies the bearing delta on each side of a specified centre bearing line.
centre_elevation elevation_coordinate_type	This attribute specifies the vertical angle measured counterclockwise between the horizontal plane and the centre elevation line of the volume.
delta_elevation elevation_coordinate_type	This attribute specifies the elevation delta on each side of a specified centre elevation line.
inner_range range_coordinate_type	This attribute specifies the range that limits a volume; i.e. the minimum distance from the volume's origin.
outer_range range_coordinate_type	This attribute specifies the range that limits a volume; i.e. the maximum distance from the volume's origin.

7.4.18.9 truncated_sector_type

Type: Class
Package: Shape_Model
A truncated_sector specifies a 2D area in a horizontal plane by means of its origin, its centre bearing with its bearing delta, and its inner range and outer range, that together define the truncated sector.
The origin is the figure reference point of the truncated sector.
In case the truncated sector is north oriented, the centre bearing is specified with respect to true north; otherwise (object oriented) it is specified with respect to the object's (own ship/other track, point) heading/course.

Table 7.50 - Attributes of Class truncated_sector_type

Attribute	Notes
centre_bearing azimuth_coordinate_type	This attribute specifies the horizontal angle measured clockwise between the Y-axis of the relevant coordinate system (true north, heading/course) and the centre bearing line of the truncated sector.
delta_bearing azimuth_coordinate_type	This attribute specifies the bearing delta on each side of a centre bearing line.
inner_range range_coordinate_type	This attribute specifies the range that limits a truncated sector; i.e. the minimum distance from the truncated sector's origin.
outer_range range_coordinate_type	This attribute specifies the range that limits a truncated sector; i.e. the maximum distance from the truncated sector's origin.

7.4.19 Requests

Parent Package: Common_Types
This package contains common operations and associated parameters which are used by multiple interfaces. This includes the operation to acknowledge a CMS request as accepted or denied, as well as an operation to report errors while processing an accepted CMS request.

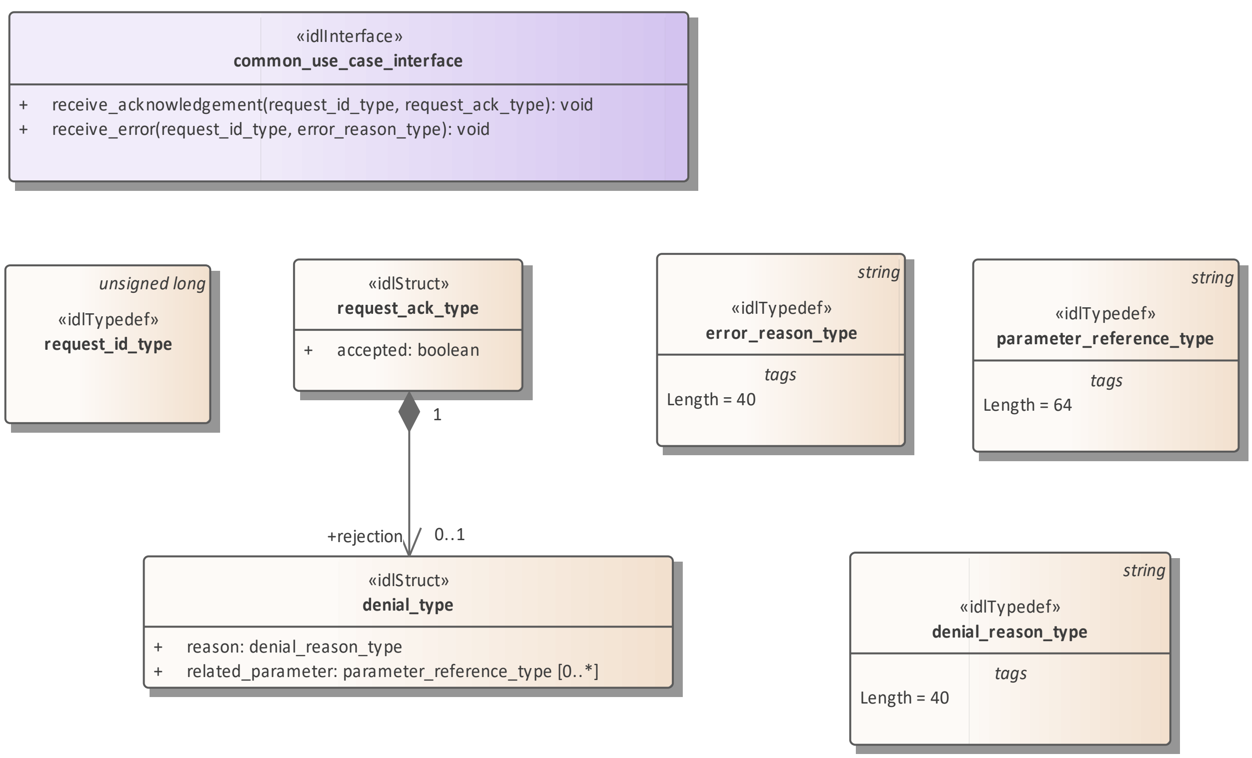


Figure 7.57 Domain Model (Class diagram)

7.4.19.1 denial_reason_type

Type: Class
Package: Requests
String which indicates rationale for rejection of the request. Is not valid when the request has been accepted.
ElementTag: Length = 40

7.4.19.2 denial_type

Type: Class
Package: Requests
Struct used within the receive_acknowledgement operation to provide information on (one of the reasons) why a request has been rejected.

Table 7.51 - Attributes of Class denial_type

Attribute	Notes
reason denial_reason_type	textual explanation of (one of) the reasons for rejection
related_parameter parameter_reference_type [0..*]	A reference to the parameter or parameters that relate to the reason for rejection. If no related_parameters are supplied the rejection relates to the whole request.

7.4.19.3 error_reason_type

Type: Class
Package: Requests
A string which gives an indication of the error associated with processing of the request.
ElementTag: Length = 40

7.4.19.4 parameter_reference_type

Type: IDLTypeDef
Package: Requests
A string which refers to a parameter in a request using an implementation specific notation.
ElementTag: Length = 64

7.4.19.5 request_ack_type

Type: Class
Package: Requests
Struct used within the receive_acknowledgement operation to indicate acceptance or rejection (which includes rationale).

Table 7.52 - Attributes of Class request_ack_type

Attribute	Notes
accepted boolean	Attribute to indicate whether a request has been accepted (1) or rejected (0).

7.4.19.6 request_id_type

Type: IDLTypeDef
Package: Requests
The purpose of the request_id is to uniquely relate responses of the subsystem (server) to requests of the CMS (client). The request_id is set by the client. It is the responsibility of the client to specify a system-wide unique request_id (e.g. based on a combination of client id and a sequence number / time of request).

7.4.19.7 common_use_case_interface

Type: Interface
Package: Requests
Interface which includes operations common to all CMS interfaces.

Table 7.53 - Methods of Interface common_use_case_interface

Method	Notes	Parameters
receive_acknowledgement()	This operation is used by the subsystem to indicate whether it has accepted or rejected a request from the CMS. MethodTag: ea_guid = {C15FF90A-E3EE-4c87-AFD6-2234A76786B2}	request_id_type **request_id** request_ack_type **request_ack**
receive_error()	This operation is used by the subsystem to indicate an error in processing a request.	request_id_type **request_id** error_reason_type **error_reason**

7.5 Subsystem_Domain

Parent Package: Domain_Model
This package contains the Domain Models for the Encyclopaedic Support, Extended Subsystem Control, Subsystem Control, Recording and Replay, and Simulation Support services.

7.5.1 Encyclopaedic_Support

Parent Package: Subsystem_Domain

Domain classes for Encyclopaedic Support

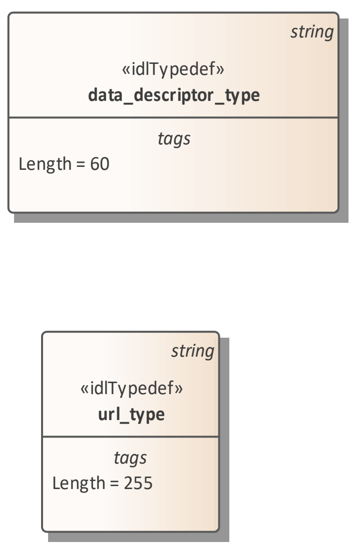


Figure 7.58 Domain Model (Class diagram)

7.5.1.1 data_descriptor_type

Type: Class
Package: Encyclopaedic_Support
Standard description of the encyclopaedic data set
ElementTag: Length = 60

7.5.1.2 url_type

Type: Class
Package: Encyclopaedic_Support
Representation of a Uniform Resource Locator see www.w3.org
ElementTag: Length = 255

7.5.2 Extended_Subsystem_Control

Parent Package: Subsystem_Domain
Contains Structs used within the Extended Subsystem Control service.

Figure 7.59 Domain Model (Class diagram)

7.5.2.1 configuration_url_type

Type: IDLTypeDef
Package: Extended_Subsystem_Control
String which provides a url location for configuration data.
ElementTag: Length = 255

7.5.2.2 network_name_type

Type: IDLTypeDef
Package: Extended_Subsystem_Control
The name identifying an external network.

7.5.2.3 offline_test_result_details_type

Type: IDLTypeDef
Package: Extended_Subsystem_Control
Subsystem specific detailed test results
ElementTag: Length = 4096

7.5.2.4 offline_test_result_type

Type: Class
Package: Extended_Subsystem_Control
Used to return the test results: failed, partial_pass or failed

Table 7.54 - Attributes of Class offline_test_result_type

Attribute	Notes
FAILED	A number of tests were not successful, such that the subsystem exceeded its failure threshold. Detailed information is available upon request.
PARTIAL_PASS	A number of tests were not successful, but the subsystem did not exceed its failure threshold. Detailed information is available upon request.
PASSED	All tests were successful.

7.5.2.5 offline_test_type

Type: IDLTypeDef
Package: Extended_Subsystem_Control
A subsystem specific string identifying the required test type.
ElementTag: Length = 255

7.5.3 Recording_and_Replay

Parent Package: Subsystem_Domain
Defines the domain model for the Recording and Replay interfaces.

This contains the classes associated with Recording and Replay

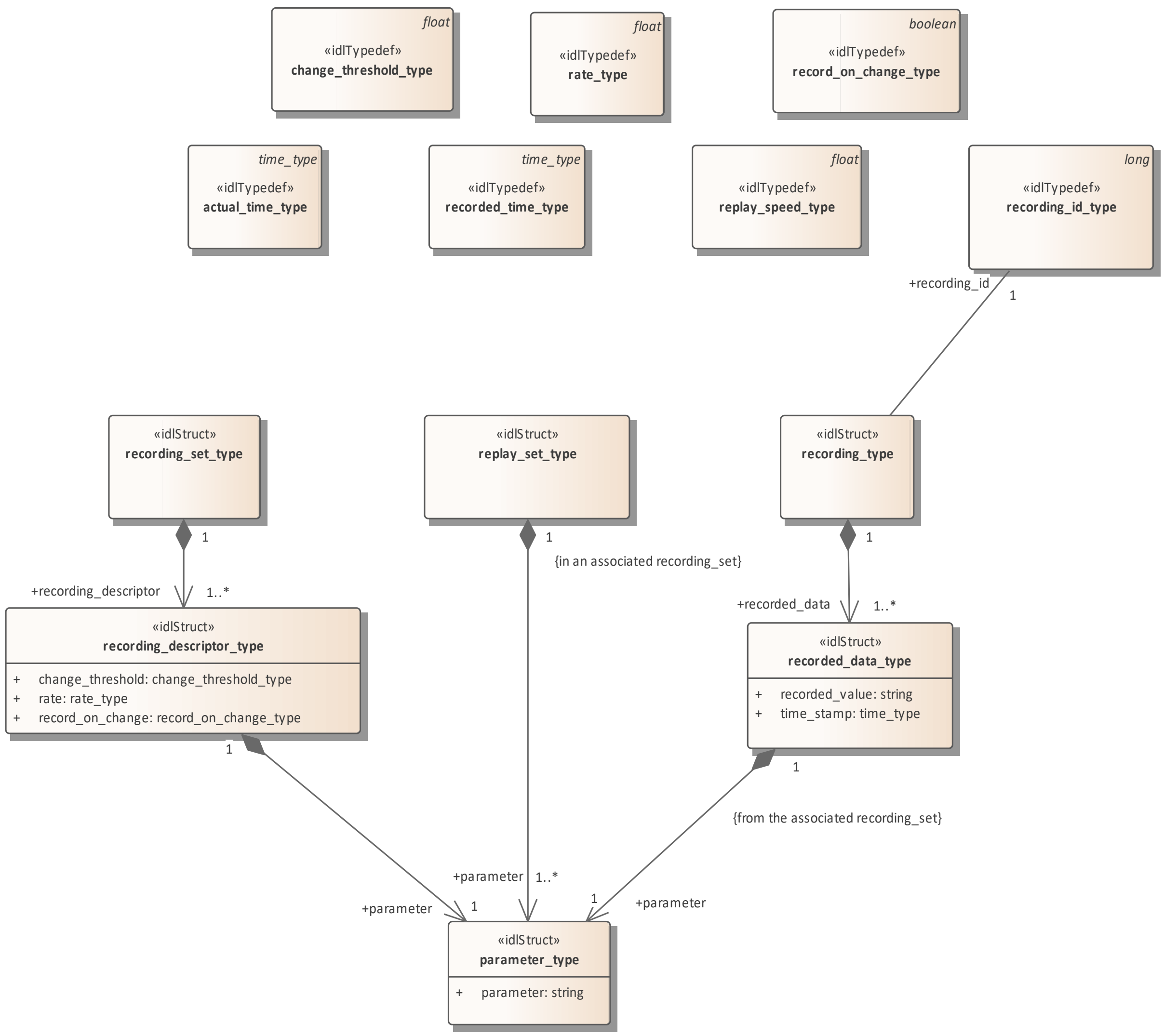


Figure 7.60 Domain Model (Class diagram)

7.5.3.1 actual_time_type

Type: Class
Package: Recording_and_Replay
The current time (time of day). Used to indicate when playback should start. This allows synchronisation of playback from different subsystems.

7.5.3.2 change_threshold_type

Type: IDLTypeDef
Package: Recording_and_Replay
The amount by which a parameter shall change in order to be recorded, when recording on change

7.5.3.3 parameter_type

Type: Class
Package: Recording_and_Replay
Identified the parameter to be recorded

Table 7.55 - Attributes of Class parameter_type

Attribute	Notes
parameter string	The parameter value. AttributeTag: StringLength = 32

7.5.3.4 rate_type

Type: IDLTypeDef
Package: Recording_and_Replay
Defined the rate at which the parameter is to be recorded for periodic recording

7.5.3.5 record_on_change_type

Type: IDLTypeDef
Package: Recording_and_Replay
Boolean specifying record on change (true) or periodic (false)

7.5.3.6 recorded_data_type

Type: Class
Package: Recording_and_Replay
Data recorded against the specified parameter

Table 7.56 - Attributes of Class recorded_data_type

Attribute	Notes
recorded_value string	This needs to reference allowable values defined by the possible recording parameters - see 'recording parameters'. AttributeTag: StringLength = 20
time_stamp time_type	The absolute time at which the value was recorded

Table 7.57 - Relations of Class recorded_data_type

Connector	Notes
Aggregation: **parameter** parameter_type [1]	

7.5.3.7 recorded_time_type

Type: Class
Package: Recording_and_Replay
The time in a recording. This is used to indicate the position in the recording at which playback should start.

7.5.3.8 recording_descriptor_type

Type: Class
Package: Recording_and_Replay
Specifies the recording characteristics required for each parameter

Table 7.58 - Attributes of Class recording_descriptor_type

Attribute	Notes
change_threshold change_threshold_type	When record_on_change is true, any change greater than the change_threshold from the last recorded value shall be recorded. This only applies for numeric quantities i.e. not enumerated types, and is ignored otherwise.
rate rate_type	Specifies recording rate when record_on_change is false. AttributeTag: Unit = Hz
record_on_change record_on_change_type	Indicates whether to record all changes greater than the change threshold or record at the specified rate.

Table 7.59 - Relations of Class recording_descriptor_type

Connector	Notes
Aggregation: **parameter** parameter_type [1]	

7.5.3.9 recording_id_type

Type: Class
Package: Recording_and_Replay
Used to identify a specific recording. The subsystem shall manage a number of recordings and associate recording ids with them in a subsystem dependent way. Once associated, it passes that reference through the parameter recording_id to the CMS so that the CMS may ask for a specific recording later on. Again, the CMS manages the relationship between the recording_id and the recording it requested to be made in a system dependent way.

There is no intention to model the method either the subsystem or the CMS uses to manage the relationship between recording_id and the recordings as this is transparent to the interface and would unnecessarily restrict the choices available to the designers.

7.5.3.10 recording_set_type

Type: Class
Package: Recording_and_Replay
A set of recording descriptors specifying what is to be recorded

Table 7.60 - Relations of Class recording_set_type

Connector	Notes
Aggregation: **recording_descriptor** recording_descriptor_type [1..*]	

7.5.3.11 recording_type

Type: Class
Package: Recording_and_Replay
A recording: a set of recorded data

Table 7.61 - Relations of Class recording_type

Connector	Notes
Aggregation: **recorded_data** recorded_data_type [1..*]	
Association: **recording_id** recording_id_type reference [1]	

7.5.3.12 replay_set_type

Type: Class
Package: Recording_and_Replay
A set of parameters required to be replayed. These must exist in the associated recording set to be of any use.

Table 7.62 - Relations of Class replay_set_type

Connector	Notes
Aggregation: **parameter** parameter_type [1..*]	

7.5.3.13 replay_speed_type

Type: Class
Package: Recording_and_Replay
Controls the replay speed. 1.0 represents real time.

7.5.4 Simulation_Support

Parent Package: Subsystem_Domain

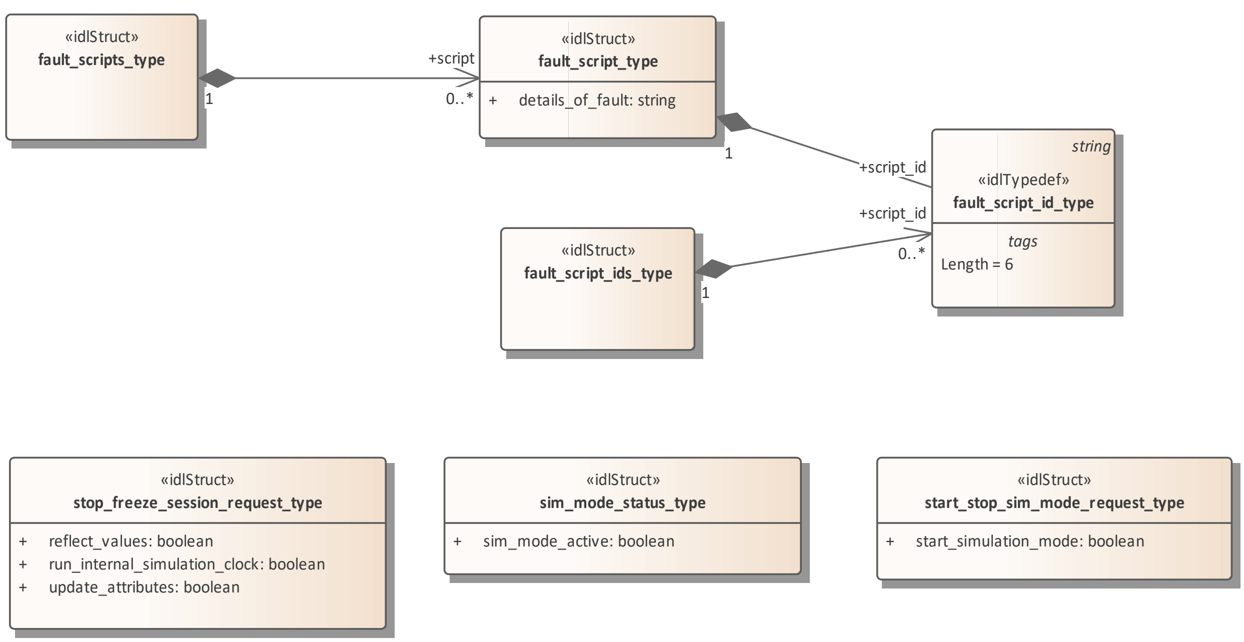


Figure 7.61 Domain Model (Class diagram)

7.5.4.1 fault_script_id_type

Type: Class
Package: Simulation_Support
Identifies a single fault script.
ElementTag: Length = 6

7.5.4.2 fault_script_ids_type

Type: Class
Package: Simulation_Support
This class represents a set of references to fault scripts

Table 7.63 - Relations of Class fault_script_ids_type

Connector	Notes
Aggregation: **script_id** fault_script_id_type [0..*]	

7.5.4.3 fault_script_type

Type: Class
Package: Simulation_Support
Definition of a fault script. The exact form of this is not yet defined, this class represents the essential attributes. It would probably be some form of string, perhaps an XML document.

Table 7.64 - Attributes of Class fault_script_type

Attribute	Notes
details_of_fault string	A description of the fault, such as is interpretable during the simulation AttributeTag: StringLength = 200

Table 7.65 - Relations of Class fault_script_type

Connector	Notes
Aggregation: **script_id** fault_script_id_type [1]	

7.5.4.4 fault_scripts_type

Type: Class
Package: Simulation_Support
This class represents a set of fault scripts

Table 7.66 - Relations of Class fault_scripts_type

Connector	Notes
Aggregation: **script** fault_script_type [0..*]	

7.5.4.5 sim_mode_status_type

Type: Class
Package: Simulation_Support
Whether simulated mode is in operation

Table 7.67 - Attributes of Class sim_mode_status_type

Attribute	Notes
sim_mode_active boolean	Flag to indicate if the simulation mode is active.

7.5.4.6 start_stop_sim_mode_request_type

Type: Class
Package: Simulation_Support
A request to change the simulation mode

Table 7.68 - Attributes of Class start_stop_sim_mode_request_type

Attribute	Notes
start_simulation_mode boolean	Flag to indicate if the simulation mode shall be started or stopped.

7.5.4.7 stop_freeze_session_request_type

Type: Class
Package: Simulation_Support
A Simulation Management (SIMAN) request, sent from a Simulation Manager to request that one or more entities either
a) pause their simulation session
or
b) stop their simulation session.

Table 7.69 - Attributes of Class stop_freeze_session_request_type

Attribute	Notes
reflect_values boolean	Whether the entity or entities being stopped/frozen should continue to reflect values when stopped/frozen.
run_internal_simulation_clock boolean	Whether the entity or entities being stopped/frozen should continue to run their internal simulation clock when stopped/frozen.
update_attributes boolean	Whether the entity or entities being stopped/frozen should continue to update attributes when stopped/frozen.

7.5.5 Subsystem_Control

Parent Package: Subsystem_Domain
Contains Structs used within the Subsystem Control service and a state diagram corresponding with the Manage Technical State interface.

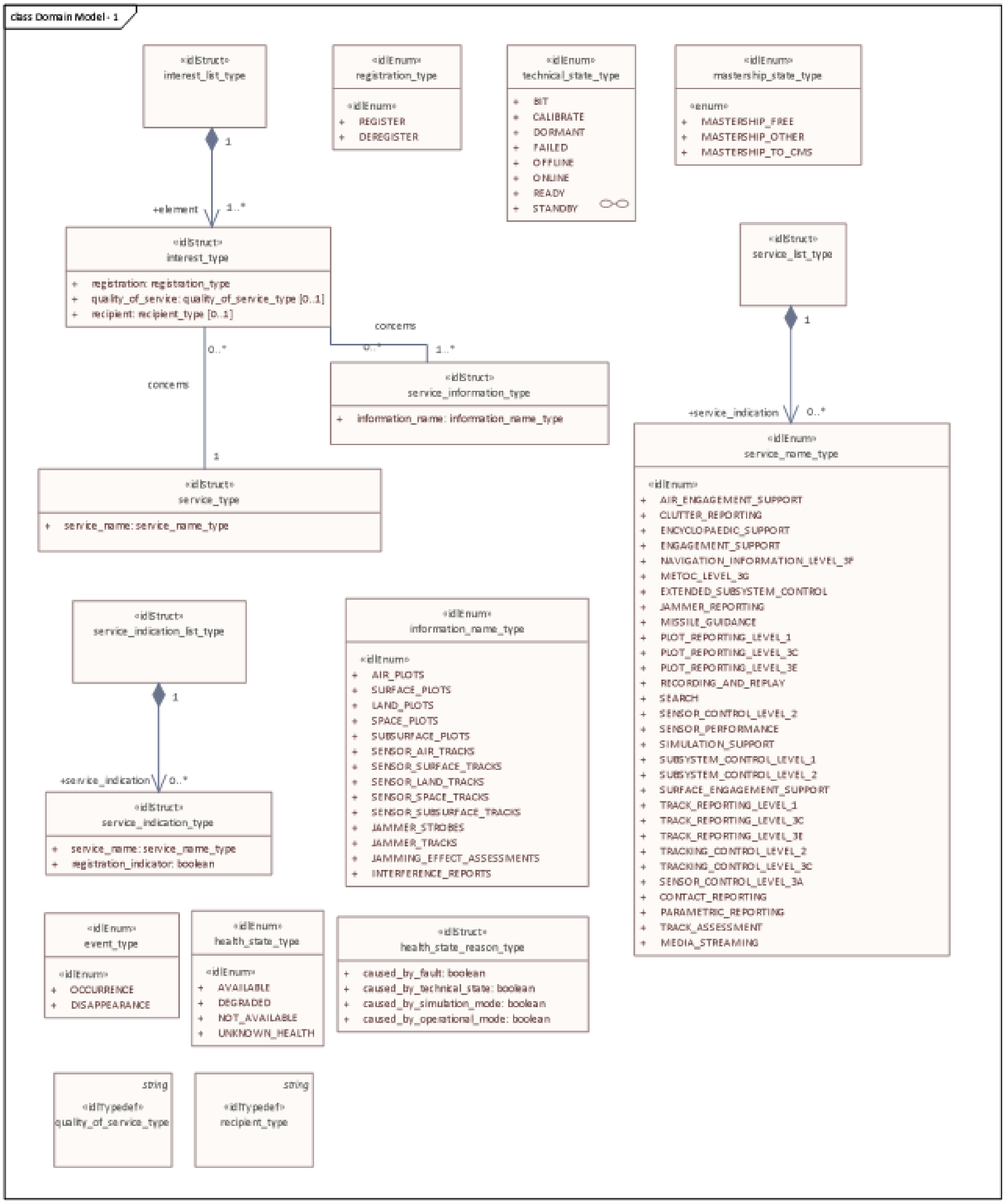
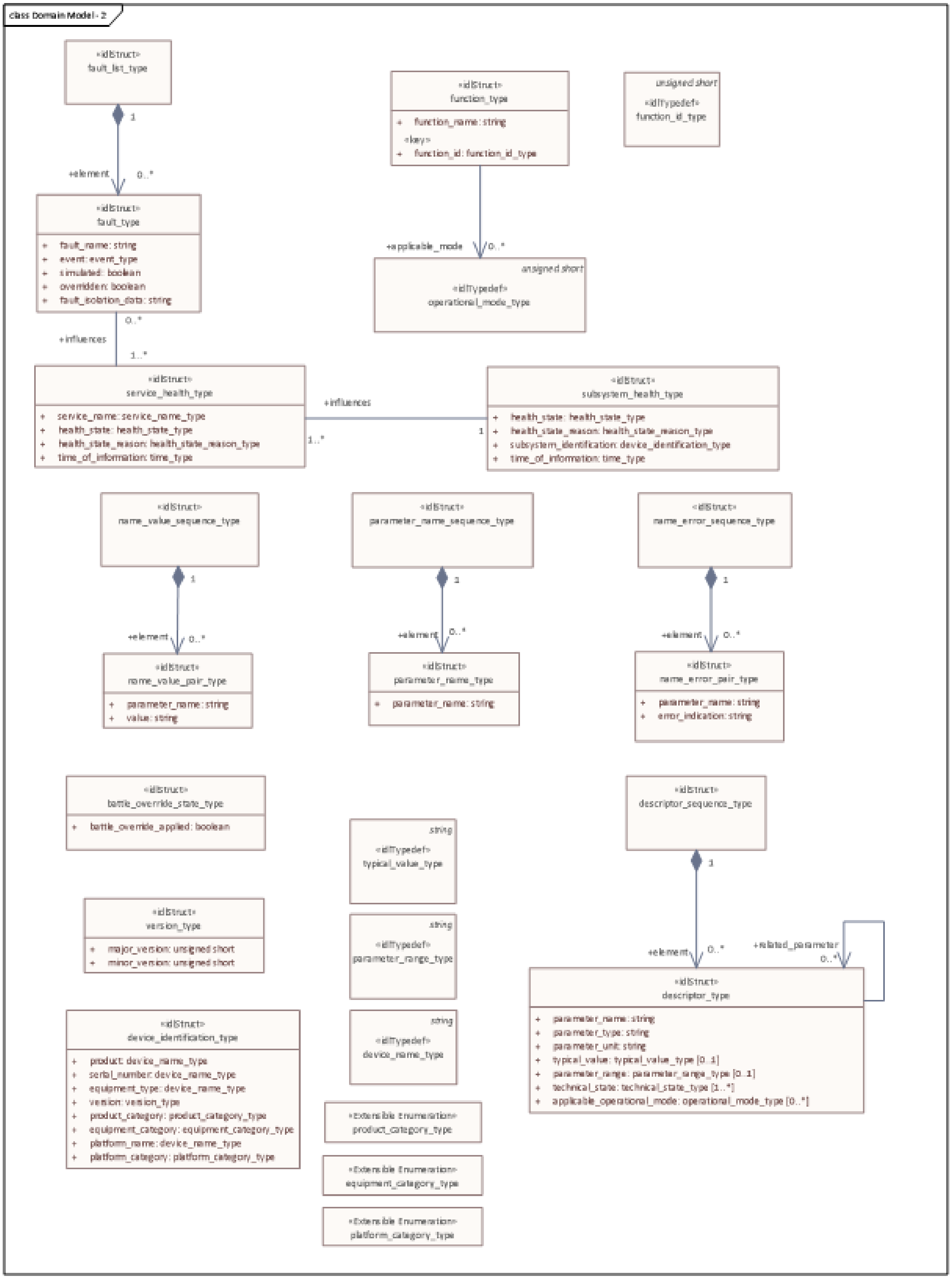


Figure 7.62 Domain Model - 1 (Class diagram)

Figure 7.63 Domain Model - 2 (Class diagram)


class Domain Model - 2
«idlStruct»
fault_list_type
1
+element
0..*
«idlStruct»
fault_type
+ fault_name: string
+ event: event_type
+ simulated: boolean
+ overridden: boolean
+ fault_isolation_data: string
0..*
+influences
1..*
«idlStruct»
function_type
+ function_name: string
«key»
+ function_id: function_id_type
unsigned short
«idlTypedef»
function_id_type
+applicable_mode
0..*
unsigned short
«idlTypedef»
operational_mode_type
«idlStruct»
service_health_type
+ service_name: service_name_type
+ health_state: health_state_type
+ health_state_reason: health_state_reason_type
+ time_of_information: time_type
+influences
1..*
1
«idlStruct»
subsystem_health_type
+ health_state: health_state_type
+ health_state_reason: health_state_reason_type
+ subsystem_identification: device_identification_type
+ time_of_information: time_type
«idlStruct»
name_value_sequence_type
1
+element
0..*
«idlStruct»
name_value_pair_type
+ parameter_name: string
+ value: string
«idlStruct»
parameter_name_sequence_type
1
+element
0..*
«idlStruct»
parameter_name_type
+ parameter_name: string
«idlStruct»
name_error_sequence_type
1
+element
0..*
«idlStruct»
name_error_pair_type
+ parameter_name: string
+ error_indication: string
«idlStruct»
battle_override_state_type
+ battle_override_applied: boolean
string
«idlTypedef»
typical_value_type
string
«idlTypedef»
parameter_range_type
string
«idlTypedef»
device_name_type
«idlStruct»
version_type
+ major_version: unsigned short
+ minor_version: unsigned short
«idlStruct»
device_identification_type
+ product: device_name_type
+ serial_number: device_name_type
+ equipment_type: device_name_type
+ version: version_type
+ product_category: product_category_type
+ equipment_category: equipment_category_type
+ platform_name: device_name_type
+ platform_category: platform_category_type
«Extensible Enumeration»
product_category_type
«Extensible Enumeration»
equipment_category_type
«Extensible Enumeration»
platform_category_type
«idlStruct»
descriptor_sequence_type
1
+element
0..*
+related_parameter
0..*
«idlStruct»
descriptor_type
+ parameter_name: string
+ parameter_type: string
+ parameter_unit: string
+ typical_value: typical_value_type [0..1]
+ parameter_range: parameter_range_type [0..1]
+ technical_state: technical_state_type [1..*]
+ applicable_operational_mode: operational_mode_type [0..*]

7.5.5.1 equipment_category_type

Type: Class
Package: Subsystem_Control
Categorization of equipment. Values correspond to items in an externally defined list.

7.5.5.2 function_id_type

Type: Class
Package: Subsystem_Control
Unique identifier for a function within a subsystem.

7.5.5.3 function_type

Type: Class
Package: Subsystem_Control
One of the functions of a subsystem

Table 7.70 - Attributes of Class function_type

Attribute	Notes
«key» **function_id** function_id_type	The functions unique idenitifier
function_name string	The name of function as understood by an operator AttributeTag: StringLength = 32

Table 7.71 - Relations of Class function_type

Connector	Notes
Association: **applicable_mode** operational_mode_type reference [0..*]	The operational modes in which the function is available

7.5.5.4 platform_category_type

Type: Class
Package: Subsystem_Control
Categorization of platforms (i.e. structures such as ships and planes) that host CMS, sensors and other subsystems. Values correspond to items in an externally defined list.

7.5.5.5 product_category_type

Type: Class
Package: Subsystem_Control
Categorization of a product. Values correspond to items in an externally defined list.

7.5.5.6 service_name_type

Type: IDLEnum
Package: Subsystem_Control
Enumeration of possible service names. Where a service may be offered at different compliance levels, multiple names are introduced with _LEVEL_x postfix to indicate different parts.

Table 7.72 - Attributes of IDLEnum service_name_type

Attribute	Notes
«idlEnum» **AIR_ENGAGEMENT_SUPPORT**	air engagement support service
«idlEnum» **CLUTTER_REPORTING**	clutter reporting service
«idlEnum» **ENCYCLOPAEDIC_SUPPORT**	encyclopaedic support service
«idlEnum» **ENGAGEMENT_SUPPORT**	engagement support service
«idlEnum» **NAVIGATION_INFORMATION_LEVEL_3F**	C2INav service support as per compliance level 3F
«idlEnum» **METOC_LEVEL_3G**	Meteorological and Oceanographic support service as per compliance level 3G
«idlEnum» **EXTENDED_SUBSYSTEM_CONTROL**	Extensions to the subsystem control service
«idlEnum» **JAMMER_REPORTING**	jammer reporting service
«idlEnum» **MISSILE_GUIDANCE**	missile guidance service
«idlEnum» **PLOT_REPORTING_LEVEL_1**	plot reporting service to compliance level 1
«idlEnum» **PLOT_REPORTING_LEVEL_3C**	plot reporting service to compliance level 3C
«idlEnum» **PLOT_REPORTING_LEVEL_3E**	plot reporting service to compliance level 3E
«idlEnum» **RECORDING_AND_REPLAY**	recording and replay service
«idlEnum» **SEARCH**	search service
«idlEnum» **SENSOR_CONTROL_LEVEL_2**	sensor control service to compliance level 2
«idlEnum» **SENSOR_PERFORMANCE**	sensor performance service
«idlEnum» **SIMULATION_SUPPORT**	simulation support service
«idlEnum» **SUBSYSTEM_CONTROL_LEVEL_1**	subsystem control service to compliance level 1
«idlEnum» **SUBSYSTEM_CONTROL_LEVEL_2**	subsystem control service to compliance level 2
«idlEnum» **SURFACE_ENGAGEMENT_SUPPORT**	surface engagement support service
«idlEnum» **TRACK_REPORTING_LEVEL_1**	track reporting service to compliance level 1
«idlEnum» **TRACK_REPORTING_LEVEL_3C**	track reporting service to compliance level 3C
«idlEnum» **TRACK_REPORTING_LEVEL_3E**	track reporting service to compliance level 3E
«idlEnum» **TRACKING_CONTROL_LEVEL_2**	tracking control service to compliance level 2
«idlEnum» **TRACKING_CONTROL_LEVEL_3C**	tracking control service to compliance level 3C
«idlEnum» **SENSOR_CONTROL_LEVEL_3A**	sensor control service to compliance level 3A

Attribute	Notes
«idlEnum» **CONTACT_REPORTING**	contact reporting service
«idlEnum» **PARAMETRIC_REPORTING**	parametric reporting
«idlEnum» **TRACK_ASSESSMENT**	track assessment service
«idlEnum» **MEDIA_STREAMING**	media streaming service

7.5.5.7 battle_override_state_type

Type: Class
Package: Subsystem_Control
If the boolean is true the battle override is applied.

Table 7.73 - Attributes of Class battle_override_state_type

Attribute	Notes
battle_override_applied boolean	Indicates if the battle override is applied or not.

7.5.5.8 descriptor_type

Type: Class
Package: Subsystem_Control
Type for parameter descriptors.

Table 7.74 - Attributes of Class descriptor_type

Attribute	Notes
parameter_name string	parameter_name values are unique within the scope of a subsystem. AttributeTag: StringLength = 128
parameter_type string	The type of the information parameter AttributeTag: StringLength = 32
parameter_unit string	The units in which the value of the parameter is expressed. AttributeTag: StringLength = 32
typical_value typical_value_type~~string~~ [0..1]	A typical value of the information parameter so as to assist in providing a suitable value. ~~AttributeTag: StringLength = 32~~
parameter_range ~~string~~ parameter_range_type [0..1]	The valid range of the information parameter. ~~AttributeTag: StringLength = 32~~
technical_state technical_state_type [1..*]	Technical state(s) in which this parameter may be modified.

Attribute	Notes
applicable_operational_mode operational_mode_type [0..*]	Operational modes to which the information value applies.

7.5.5.9 descriptor_sequence_type

Type: Class
Package: Subsystem_Control
Sequence of parameter descriptors, used in retrieving parameter descriptors.

Table 7.75 - Relations of Class descriptor_sequence_type

Connector	Notes
Aggregation: **element** descriptor_type [0..*]	

7.5.5.10 device_identification_type

Type: IDLStruct
Package: Subsystem_Control
Identification data of the equipment.

Table 7.76 - Attributes of IDLStruct device_identification_type

Attribute	Notes
product device_name_type	Name of the product. Example TRS3D
serial_number device_name_type	Serial number identifying the individual device. AttributeTag: Length = 64
equipment_type device_name_type	This describes the general type of the equipment. Example: Air Surveillance Radar
version version_type	Version of the device.
product_category product_category_type	Categorization of the product implementing the interface. This is the unique identification of the product given a particular external schema. AttributeTag: Issue =
equipment_category equipment_category_type	Categorization of the kind of equipment implementing the interface. This is the specific identification of the product's equipment category given a particular external schema. AttributeTag: Issue =
platform_name device_name_type	The name of the platform hosting the interface participant. AttributeTag: Issue =

Attribute	Notes
platform_category platform_category_type	Categorization of platform hosting the product implementing the interface. This is the unique identification of the platform given a particular external schema. AttributeTag: Issue =

7.5.5.11 device_name_type

Type: IDLTypeDef
Package: Subsystem_Control
Name of an entry in the device identification.
ElementTag: Length = 64

7.5.5.12 parameter_range_type

Type: IDLTypeDef
Package: Subsystem_Control
The valid range of the information parameter.
ElementTag: Length = 32

7.5.5.13 quality_of_service_type

Type: IDLTypeDef
Package: Subsystem_Control
The quality of service being requested of the information service.
ElementTag: Length = 32

7.5.5.14 recipient_type

Type: IDLTypeDef
Package: Subsystem_Control
Identification of the recipient of the information service.
ElementTag: Length = 32

7.5.5.15 typical_value_type

Type: IDLTypeDef
Package: Subsystem_Control
A typical value of the information parameter so as to assist in providing a suitable value.
ElementTag: Length = 32

7.5.5.16 event_type

Type: IDLEnum
Package: Subsystem_Control
Type of event

Table 7.77 - Attributes of IDLEnum event_type

Attribute	Notes
«idlEnum» **OCCURRENCE**	The event corresponds to the occurrence of some phenomena

Attribute	Notes
«idlEnum» **DISAPPEARANCE**	The event corresponds to the disappearance of some phenomena

7.5.5.17 fault_type

Type: IDLStruct
Package: Subsystem_Control
Class to represent a subsystem fault

Table 7.78 - Attributes of IDLStruct fault_type

Attribute	Notes
fault_name string	The name of the fault. Distinct instances of the same fault condition have the same name. AttributeTag: StringLength = 32
event event_type	The categorization of the fault as an event; whether it is an occurrence or the disappearance of some phenomenon
simulated boolean	Indicates whether this fault is real or simulated/inserted.
overridden boolean	Indicates whether this fault is overridden by Battle Override when determining the health state.
fault_isolation_data string	For instance cabinet id and rack id. AttributeTag: StringLength = 32

7.5.5.18 fault_list_type

Type: Class
Package: Subsystem_Control
A list of faults

Table 7.79 - Relations of Class fault_list_type

Connector	Notes
Aggregation: **element** fault_type [0..*]	

7.5.5.19 health_state_reason_type

Type: IDLStruct
Package: Subsystem_Control
Reason for the health state

Table 7.80 - Attributes of IDLStruct health_state_reason_type

Attribute	Notes
caused_by_fault boolean	The health state has been caused by a fault
caused_by_technical_state boolean	The health state is due to the subsystem being in a particular technical state

Attribute	Notes
caused_by_simulation_mode boolean	The health state is due to the subsystem being in a particular simulation mode
caused_by_operational_mode boolean	The health state is due to the subsystem being in a particular operational mode

7.5.5.20 health_state_type

Type: IDLEnum
Package: Subsystem_Control
Encapsulation of health state

Table 7.81 - Attributes of IDLEnum health_state_type

Attribute	Notes
«idlEnum» **AVAILABLE**	Service: Indicates that the service is available with specified performance. Subsystem: Indicates that all implemented services of the subsystem have health state AVAILABLE.
«idlEnum» **DEGRADED**	Service: Indicates that the service may perform its operational task, but possibly with less than specified performance. Subsystem: Indicates that at least one of the implemented services of the subsystem have health state other than AVAILABLE.
«idlEnum» **NOT_AVAILABLE**	Service: Indicates that the service is not available. Subsystem: Indicates that all implemented services of the subsystem have health state NOT_AVAILABLE.
«idlEnum» **UNKNOWN_HEALTH**	Indicates that the subsystem may not determine the health state of the service or subsystem (e.g. because BIT is not running).

7.5.5.21 information_name_type

Type: IDLEnum
Package: Subsystem_Control
Name of information

Table 7.82 - Attributes of IDLEnum information_name_type

Attribute	Notes
«idlEnum» **AIR_PLOTS**	Air plots information service
«idlEnum» **SURFACE_PLOTS**	Surface plots information service
«idlEnum» **LAND_PLOTS**	Land plots information service
«idlEnum» **SPACE_PLOTS**	Space plots information service

Attribute	Notes
«idlEnum» **SUBSURFACE_PLOTS**	Subsurface plots information service
«idlEnum» **SENSOR_AIR_TRACKS**	Air tracks information service
«idlEnum» **SENSOR_SURFACE_TRACKS**	Surface tracks information service
«idlEnum» **SENSOR_LAND_TRACKS**	Land tracks information service
«idlEnum» **SENSOR_SPACE_TRACKS**	Space tracks information service
«idlEnum» **SENSOR_SUBSURFACE_TRACKS**	Subsurface tracks information service
«idlEnum» **JAMMER_STROBES**	Jammer strobes information service
«idlEnum» **JAMMER_TRACKS**	jammer tracks information service
«idlEnum» **JAMMING_EFFECT_ASSESSMENTS**	jammer effect assessments information service
«idlEnum» **INTERFERENCE_REPORTS**	interference reports information service

7.5.5.22 interest_type

Type: IDLStruct
Package: Subsystem_Control
Encapsulation of interest in service

Table 7.83 - Attributes of IDLStruct interest_type

Attribute	Notes
registration registration_type	Whether adding or removing interest in an information service.
quality_of_service ~~string~~quality_of_service_type [0..1]	The quality of service being requested of the information service. ~~AttributeTag: StringLength = 32~~
recipient s~~tring~~recepient_type [0..1]	Identification of the recipient of the information service. ~~AttributeTag: StringLength = 32~~

7.5.5.23 interest_list_type

Type: Class
Package: Subsystem_Control
A list of interest

Table 7.84 - Relations of Class interest_list_type

Connector	Notes
Aggregation: **element** interest_type [1..*]	

7.5.5.24 mastership_state_type

Type: Class
Package: Subsystem_Control
This enumeration represents the state of the mastership.
The subsystem Mastership may be either "free", that is assigned to none and then available to anybody asks for it, or assigned to somebody: CMS or not.

Table 7.85 - Attributes of Class mastership_state_type

Attribute	Notes
«enum» **MASTERSHIP_FREE**	Mastership state is "free", the first received Mastership request shall be satisfied.
«enum» **MASTERSHIP_OTHER**	The Mastership is assigned to somebody other than CMS.
«enum» **MASTERSHIP_TO_CMS**	The Mastership is assigned to CMS.

7.5.5.25 parameter_name_type

Type: IDLStruct
Package: Subsystem_Control
Typedef for strings representing names of parameters.

Table 7.86 - Attributes of IDLStruct parameter_name_type

Attribute	Notes
parameter_name string	parameter_name values are unique within the scope of a subsystem. AttributeTag: StringLength = 128

7.5.5.26 name_error_pair_type

Type: Class
Package: Subsystem_Control
Combination of name of parameter (for which a request could not be processed) and an indication of the error.

Table 7.87 - Attributes of Class name_error_pair_type

Attribute	Notes
parameter_name string	parameter_name values are unique within the scope of a subsystem. AttributeTag: StringLength = 128
error_indication string	A description of or reference for the error condition. AttributeTag: StringLength = 32

7.5.5.27 name_error_sequence_type

Type: Class
Package: Subsystem_Control

sequence of error reports identifying the parameter names for which the request could not be processed, including an indication of the error (e.g. unknown parameter, illegal value).

Table 7.88 - Relations of Class name_error_sequence_type

Connector	Notes
Aggregation: **element** name_error_pair_type [0..*]	

7.5.5.28 parameter_name_sequence_type

Type: Class
Package: Subsystem_Control
A sequence of strings (names). Used in request for parameters and parameter descriptors. If the sequence is empty, the request is for all parameters.

Table 7.89 - Relations of Class parameter_name_sequence_type

Connector	Notes
Aggregation: **element** parameter_name_type [0..*]	

7.5.5.29 name_value_pair_type

Type: Class
Package: Subsystem_Control
A generic struct for (name, value) pairs. Used in multiple situations.

Table 7.90 - Attributes of Class name_value_pair_type

Attribute	Notes
parameter_name string	parameter_name values are unique within the scope of a subsystem. AttributeTag: StringLength = 128
value string	The value of the parameter AttributeTag: StringLength = 32

7.5.5.30 name_value_sequence_type

Type: Class
Package: Subsystem_Control
Sequence of (name, value) pairs used in retrieving and modifying parameters.

Table 7.91 - Relations of Class name_value_sequence_type

Connector	Notes
Aggregation: **element** name_value_pair_type [0..*]	

7.5.5.31 operational_mode_type

Type: IDLTypeDef
Package: Subsystem_Control
The value should be mapped to the corresponding operational mode. This mapping is retrieved through the service 'Manage Subsystem Parameters'.

7.5.5.32 parameter_value_response_type

Type: Class
Package: Subsystem_Control
Response type for retrieving and modifying sequences of parameters.

Table 7.92 - Attributes of Class parameter_value_response_type

Attribute	Notes
request_id long	The identifier for the request.

7.5.5.33 registration_type

Type: IDLEnum
Package: Subsystem_Control
Type of registration

Table 7.93 - Attributes of IDLEnum registration_type

Attribute	Notes
«idlEnum» **REGISTER**	Registering for a service
«idlEnum» **DEREGISTER**	Deregistering for a service

7.5.5.34 service_type

Type: IDLStruct
Package: Subsystem_Control
Type of service

Table 7.94 - Attributes of IDLStruct service_type

Attribute	Notes
service_name service_name_type	Only registrable services are allowed

7.5.5.35 service_health_type

Type: IDLStruct
Package: Subsystem_Control
Health of service

Table 7.95 - Attributes of IDLStruct service_health_type

Attribute	Notes
service_name service_name_type	The name of the service being reported on
health_state health_state_type	The state of health of the service
health_state_reason health_state_reason_type	The reason for the health state
time_of_information time_type	The absolute time at which the information was known to be valid

Table 7.96 - Relations of IDLStruct service_health_type

Connector	Notes
Association: **influences** fault_type reference [0..*]	

7.5.5.36 service_indication_list_type

Type: IDLStruct
Package: Subsystem_Control
A list of service indications as used by Provide_Subsystem_Services.

Table 7.97 - Relations of IDLStruct service_indication_list_type

Connector	Notes
Aggregation: **service_indication** service_indication_type [0..*]	

7.5.5.37 service_indication_type

Type: IDLStruct
Package: Subsystem_Control
Indication of a service provided by the subsystem.

Table 7.98 - Attributes of IDLStruct service_indication_type

Attribute	Notes
service_name service_name_type	Name of the service.
registration_indicator boolean	Indication whether the service is registered.

7.5.5.38 service_information_type

Type: IDLStruct
Package: Subsystem_Control
Information about a service

Table 7.99 - Attributes of IDLStruct service_information_type

Attribute	Notes
information_name information_name_type	The name of the information in the service.

7.5.5.39 service_list_type

Type: IDLStruct
Package: Subsystem_Control
A list of service names as used by Provide_Subsystem_Services.

Table 7.100 - Relations of IDLStruct service_list_type

Connector	Notes
Aggregation: **service_indication** service_name_type [0..*]	

7.5.5.40 subsystem_health_type

Type: IDLStruct
Package: Subsystem_Control
Type describing the health state of a subsystem

Table 7.101 - Attributes of IDLStruct subsystem_health_type

Attribute	Notes
health_state health_state_type	Current health state
health_state_reason health_state_reason_type	Reason for last change of health state
subsystem_identification device_identification_type	The subsystem being reported upon
time_of_information time_type	The absolute time at which the information provided in the report was known to be valid

Table 7.102 - Relations of IDLStruct subsystem_health_type

Connector	Notes
Association: **influences** service_health_type reference [1..*]	

7.5.5.41 technical_state_type

Type: IDLEnum~~Class~~
Package: Subsystem_Control
Type which is used to indicate a technical state.

Table 7.103 - Attributes of Class technical_state_type

Attribute	Notes
BIT	Subsystem is running Built-In-Test procedure. CMS may communicate with subsystem, but subsystem shall only respond affirmatively to a limited set of commands. From this state the subsystem may transition to *READY*, *FAILED*, *CALIBRATE*, STANDBY (transition may be ordered before completion of BIT if Battle Override is enabled), or *OFFLINE*.
CALIBRATE	Subsystem is running calibration procedure. Subsystem shall only respond to a limited set of commands from CMS. From this state the subsystem may transition to *READY*, *FAILED*, BIT, STANDBY (transition may be ordered before completion of calibration if Battle Override is enabled), or *OFFLINE*.
DORMANT	Interface between CMS and subsystem may or may not exist. Some power is applied to the subsystem and temperature control (e.g. cooling) is active. From this state, the sub-system may transition to FAILED, STANDBY, or OFFLINE.

Attribute	Notes
FAILED	Subsystem is non-operational due to a critical fault such as a primary power supply failure. CMS is able to communicate with subsystem to perform diagnostics. In the FAILED state, the health state of the sub-system and nearly all associated services is NOT AVAILABLE or UNKNOWN (provided via Health State). If the health state of the sub-system or some services is DEGRADED, the sub-system is not required to enter into this state. From this state the sub-system may transition to *BIT*, *STANDBY*, *READY*, CALIBRATE, DORMANT or *OFFLINE*.
OFFLINE	No connection between CMS and Subsystem is open. Main power is usually not applied to subsystem. From OFFLINE, subsystem transitions to FAILED, DORMANT, BIT, or STANDBY.
ONLINE	Subsystem is operational and may respond to all requests from CMS. Simulation and diagnostics may be allowed in this state. Radiation is allowed in this state but must be commanded on via Control Emissions. From this state the subsystem may transition to *BIT, CALIBRATE, READY, STANDBY, FAILED,* or *OF*FLINE.
READY	Subsystem is ready for CMS to command full operation. Simulation may be allowed in this state. Ready to transition to *ONLINE*, self-tests and calibration has been performed as necessary. Radiation is not allowed in the READY state. From this state the subsystem may transition to *STANDBY*, ONLINE, FAILED, BIT, CALIBRATE, or *OFFLINE*.
STANDBY	Interface between CMS and subsystem is established. Subsystem may not operate fully. Maintenance may be performed in this state. From this state the sub-system may transition to READY, CALIBRATE, BIT, FAILED, DORMANT, or OFFLINE.

7.5.5.42 version_type

Type: IDLStruct
Package: Subsystem_Control
Version of the equipment

Table 7.104 - Attributes of IDLStruct version_type

Attribute	Notes
major_version unsigned short	Major version number
minor_version unsigned short	Minor version number

7.6 Sensor_Domain

Parent Package: Domain_Model
This package contains the Domain Models for the Clutter Reporting, Plot Reporting, Sensor Control, Sensor Performance, Track Reporting, and Tracking Control services.

7.6.1 Clutter_Reporting

Parent Package: Sensor_Domain
Contains Structs used within the Clutter Reporting service.

Domain Model of the Clutter Reporting interfaces.

Figure 7.64 Domain Model (Class diagram)

7.6.1.1 clutter_assessment_request_type

Type: IDLStruct
Package: Clutter_Reporting
CMS generated request for a clutter assessment.

Table 7.105 - Attributes of IDLStruct clutter_assessment_request_type

Attribute	Notes
requested_region general_polar_volume_type	Region for which the CMS clutter request was generated.

7.6.1.2 clutter_indication_type

Type: Class
Package: Clutter_Reporting
Indicates if the clutter within the cell is of a specific type.

Table 7.106 - Attributes of Class clutter_indication_type

Attribute	Notes
LAND	clutter caused by land
SEA	clutter caused by sea surface
WEATHER	clutter caused by weather phenomena
NO_STATEMENT	clutter cause unknown or unstated

7.6.1.3 clutter_map_cell_type

Type: IDLStruct
Package: Clutter_Reporting
Indicates the intensity and type of clutter for a defined geometric type.

Table 7.107 - Attributes of IDLStruct clutter_map_cell_type

Attribute	Notes
cell_boundaries general_polar_volume_type	Indicates the boundaries of the cell for which clutter is being reported.
clutter_type clutter_indication_type	Indicates whether the clutter is LAND, SEA, WEATHER, or unspecified (NO_STATEMENT).
clutter_intensity double	Intensity of the clutter for the specified cell. Units indicated by the intensity type attribute.

7.6.1.4 clutter_report_type

Type: IDLStruct
Package: Clutter_Reporting
Clutter report generated by the subsystem.

Table 7.108 - Attributes of IDLStruct clutter_report_type

Attribute	Notes
intensity_type intensity_units_type	Indicates the units of the clutter intensity reported.
time_of_report time_type	Time of the clutter report.

Table 7.109 - Relations of IDLStruct clutter_report_type

Connector	Notes
Aggregation: **clutter_map_cell** clutter_map_cell_type [1..*]	

7.6.1.5 concentration_plot_cell_type

Type: Class
Package: Clutter_Reporting
Indicates the plot concentration of a defined geometric type.

Table 7.110 - Attributes of Class concentration_plot_cell_type

Attribute	Notes
cell_boundaries general_polar_volume_type	Specifies the dimension of the cell for which plot concentration is being reported.
plot_count unsigned long	The number of plots generated within the cell.

7.6.1.6 intensity_units_type

Type: Class
Package: Clutter_Reporting
Units of the clutter intensity

Table 7.111 - Attributes of Class intensity_units_type

Attribute	Notes
POWER_RECEIVED_LINEAR	Direct measurement of power in Watts on a linear scale
POWER_RECEIVED_LOG_LINEAR	Direct measurement of power on a logarithmic scale (e.g. dBW)
RCS_LINEAR	estimated radar cross section in square meters
RCS_LOG_LINEAR	estimated radar cross section on a logarithmic scale
SNR_LINEAR	Ratio of the signal and noise amplitudes
SNR_LOG_LINEAR	Ratio of the signal and noise amplitudes on a logarithmic scale

7.6.1.7 plot_concentration_report_type

Type: IDLStruct
Package: Clutter_Reporting
Plot concentration report as generated by the subsystem.

Table 7.112 - Attributes of IDLStruct plot_concentration_report_type

Attribute	Notes
time_of_report time_type	Time of the plot concentration report.

Table 7.113 - Relations of IDLStruct plot_concentration_report_type

Connector	Notes
Aggregation: **concentration_plot_cell** concentration_plot_cell_type [1..*]	

7.6.1.8 plot_concentration_request_data_type

Type: IDLStruct
Package: Clutter_Reporting
CMS request for plot concentration of a specified region.

Table 7.114 - Attributes of IDLStruct plot_concentration_request_data_type

Attribute	Notes
region_of_plot_concentration_request general_polar_volume_type	Region for which the plot concentration was requested.

7.6.2 Media_Streaming

Parent Package: Sensor_Domain
This package provides a data model for describing the metadata associated with a sensor's media streams.

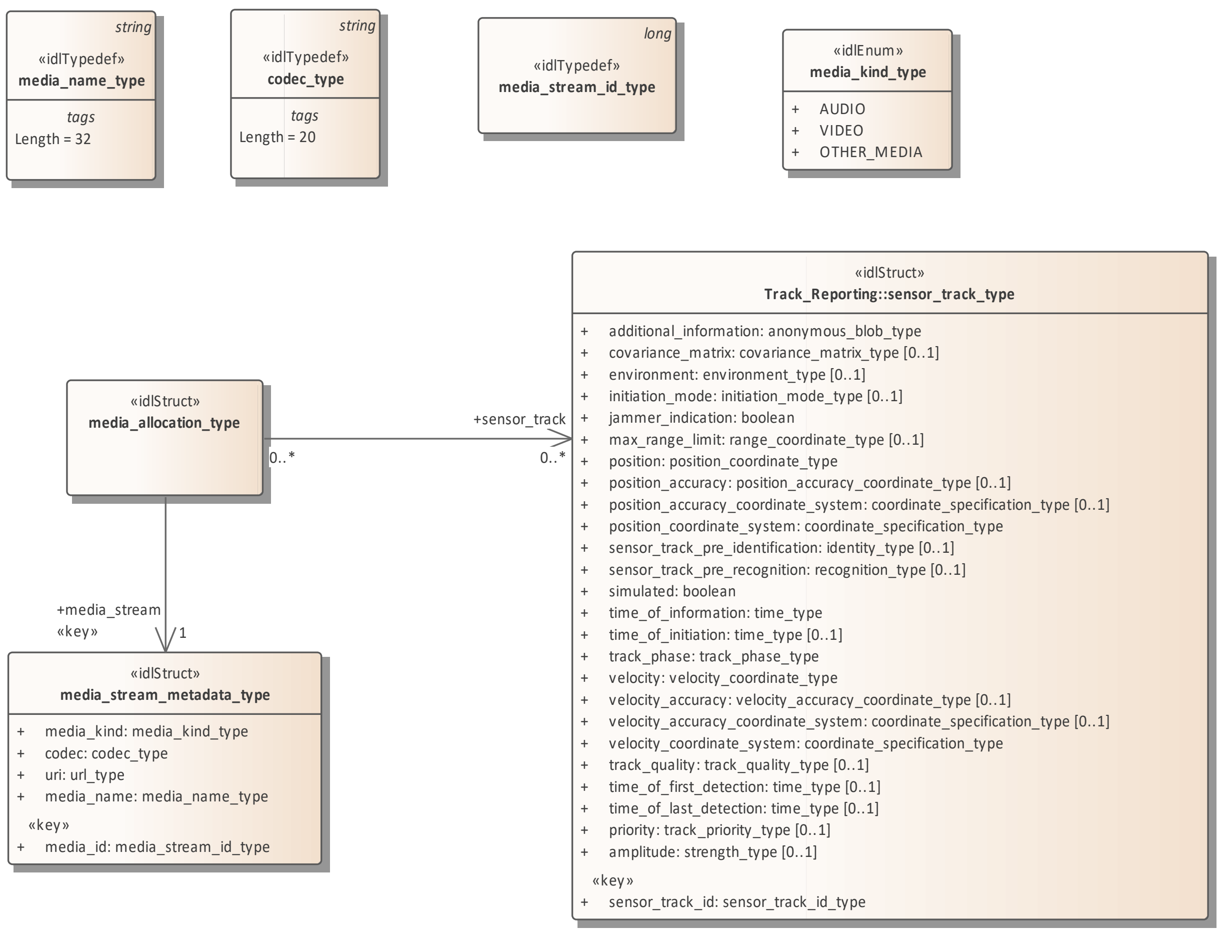


Figure 7.65 Media_Streaming (Class diagram)

7.6.2.1 codec_type

Type: Class

Package: Media_Streaming
The representation of the codec associated with the stream
ElementTag: Length = 20

7.6.2.2 media_allocation_type

Type: Class
Package: Media_Streaming
To represent the allocation of sensor tracks to media streams.

Table 7.115 - Relations of Class media_allocation_type

Connector	Notes
Association: **media_stream** media_stream_metadata_type reference [1]	The media stream relating to this allocation of tracks
Association: **sensor_track** sensor_track_type reference [0..*]	The sensor tracks that are present in the media stream

7.6.2.3 media_kind_type

Type: Class
Package: Media_Streaming
The high-level categorisation of types of media

Table 7.116 - Attributes of Class media_kind_type

Attribute	Notes
AUDIO	Audio media stream
VIDEO	Video media stream
OTHER_MEDIA	Another media stream

7.6.2.4 media_name_type

Type: Class
Package: Media_Streaming
The representation for the identifying name of a media stream
ElementTag: Length = 32

7.6.2.5 media_stream_id_type

Type: Class
Package: Media_Streaming
The representation for the unique identifier for the media stream

7.6.2.6 media_stream_metadata_type

Type: Class
Package: Media_Streaming
The representation of a media stream such as video or audio

Table 7.117 - Attributes of Class media_stream_metadata_type

Attribute	Notes
media_kind media_kind_type	The kind of media associated with the stream
codec codec_type	The code identifier for the string
uri url_type	The source (end point) of the stream.
media_name media_name_type	The identifying name for the media stream
«key» **media_id** media_stream_id_type	Identifier for the media stream.

7.6.3 Search

Parent Package: Sensor_Domain

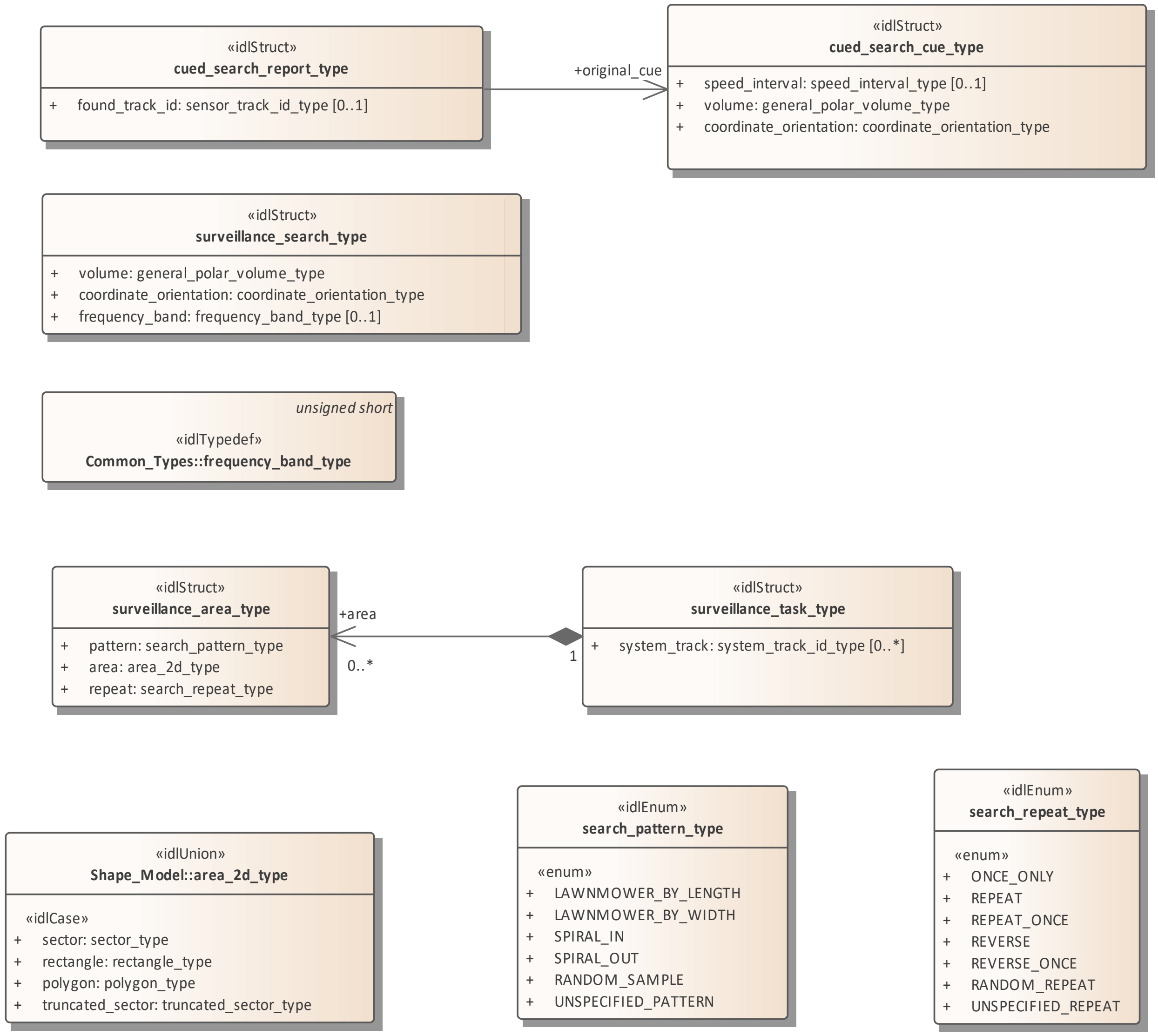


Figure 7.66 Domain Model (Class diagram)

7.6.3.1 cued_search_cue_type

Type: Class
Package: Search
Type used for specifying the constraints on a cued search.

Table 7.118 - Attributes of Class cued_search_cue_type

Attribute	Notes
speed_interval speed_interval_type [0..1]	The range of track-speed to search for from the cue.
volume general_polar_volume_type	The region in the environment, in which the cue to search for tracks is to be performed.

Attribute	Notes
coordinate_orientation coordinate_orientation_type	The orientation of the polar coordinates used in this class. Note that the origin is always the sensor reference point and that the coordinate system is always polar.

7.6.3.2 cued_search_report_type

Type: Class
Package: Search
Data returned to the CMS to indicate the results of a cued search.

Table 7.119 - Attributes of Class cued_search_report_type

Attribute	Notes
found_track_id sensor_track_id_type [0..1]	The identifier of the track formed as a result of the search request

Table 7.120 - Relations of Class cued_search_report_type

Connector	Notes
Association: **original_cue** cued_search_cue_type reference	

7.6.3.3 search_pattern_type

Type: IDLEnum
Package: Search
The types of search pattern that can be employed for search and surveillance tasks

Table 7.121 - Attributes of IDLEnum search_pattern_type

Attribute	Notes
«enum» **LAWNMOWER_BY_LENGTH**	Coverage by alternating traversal of the area length-wise. Valid for rectangular areas.
«enum» **LAWNMOWER_BY_WIDTH**	Coverage by alternating traversal of the area width-wise. Valid for rectangular areas.
«enum» **SPIRAL_IN**	Coverage by traversing the perimeter and then progressively smaller traversals of the interior towards the center..
«enum» **SPIRAL_OUT**	Coverage by starting at the center and traversing through the interior on a path that is (approximately) tangential to the center and parallel to the perimeter until the perimeter has been traversed.
«enum» **RANDOM_SAMPLE**	Search by sensing subsets of the area selected at random.
«enum» **UNSPECIFIED_PATTERN**	No search pattern is specified.

7.6.3.4 search_repeat_type

Type: IDLEnum

Package: Search
Defines the search behavior on repeat / subsequent searches

Table 7.122 - Attributes of IDLEnum search_repeat_type

Attribute	Notes
«enum» **ONCE_ONLY**	Complete a single search pattern.
«enum» **REPEAT**	Repeat the task indefinitely.
«enum» **REPEAT_ONCE**	Repeat the task once.
«enum» **REVERSE**	Repeat the task in reverse indefinitely.
«enum» **REVERSE_ONCE**	Repeat the task in reverse once.
«enum» **RANDOM_REPEAT**	Randomly repeat the elements of the task indefinitely.
«enum» **UNSPECIFIED_REPEAT**	No repeat specified.

7.6.3.5 surveillance_area_type

Type: IDLStruct
Package: Search
A 2D area that is included in a surveillance task

Table 7.123 - Attributes of IDLStruct surveillance_area_type

Attribute	Notes
pattern search_pattern_type	The pattern to apply to the area
area area_2d_type	The area to be kept under surveillance.
repeat search_repeat_type	The search behavior at the end of a search cycle.

7.6.3.6 surveillance_search_type

Type: IDLStruct
Package: Search
The parameters with which to task a sensor to concentrate its surveillance efforts within a spatial and / or frequency band.

Table 7.124 - Attributes of IDLStruct surveillance_search_type

Attribute	Notes
volume general_polar_volume_type	The region of surveillance in the environment to be searched for tracks.
coordinate_orientation coordinate_orientation_type	The orientation of the polar coordinates used in this class. Note that the origin is always the sensor reference point and that the coordinate system is always polar.

Attribute	Notes
frequency_band frequency_band_type [0..1]	The frequency band to be searched.

7.6.3.7 surveillance_task_type

Type: IDLStruct
Package: Search
The information for a CMS request to the subsystem (as appropriate to be a directional sensor that can be steered) to undertake a surveillance task.

Table 7.125 - Attributes of IDLStruct surveillance_task_type

Attribute	Notes
system_track system_track_id_type [0..*]	The system tracks to keep under surveillance. The information regarding the system tracks is published using another interface standard, such as the TACSIT Data Exchange specification (TEX), the choice of which may be system specific. AttributeTag: Length = 100

Table 7.126 - Relations of IDLStruct surveillance_task_type

Connector	Notes
Aggregation: **area** surveillance_area_type [0..*]	

7.6.4 Sensor_Assessment

Parent Package: Sensor_Domain
This package provides a data model to describe the identification and classification assessment that a sensor can make about its sensor tracks. The approach is to be agnostic to any specific assessment process or classification regime. The model assumes a general process whereby matches (with confidence values) are made between data relating to the sensor track and reference data. Matches can relate to previous matches building up a structured hierarchy of assumptions leading to progressively higher-level identification and classification assessments. The lowest level is to match measurement parameters with reference data; the next level is to match one or more of these with modes; then modes with equipment and finally equipment with platforms.
Assessment Objectives (Equipment, Function, Platform and Activity) are passed from Subsystems to CMS by value in the Sensor Assessment use cases. They contain an Objective Id key value which is used to refer to the assessment values in Track Reporting use cases.

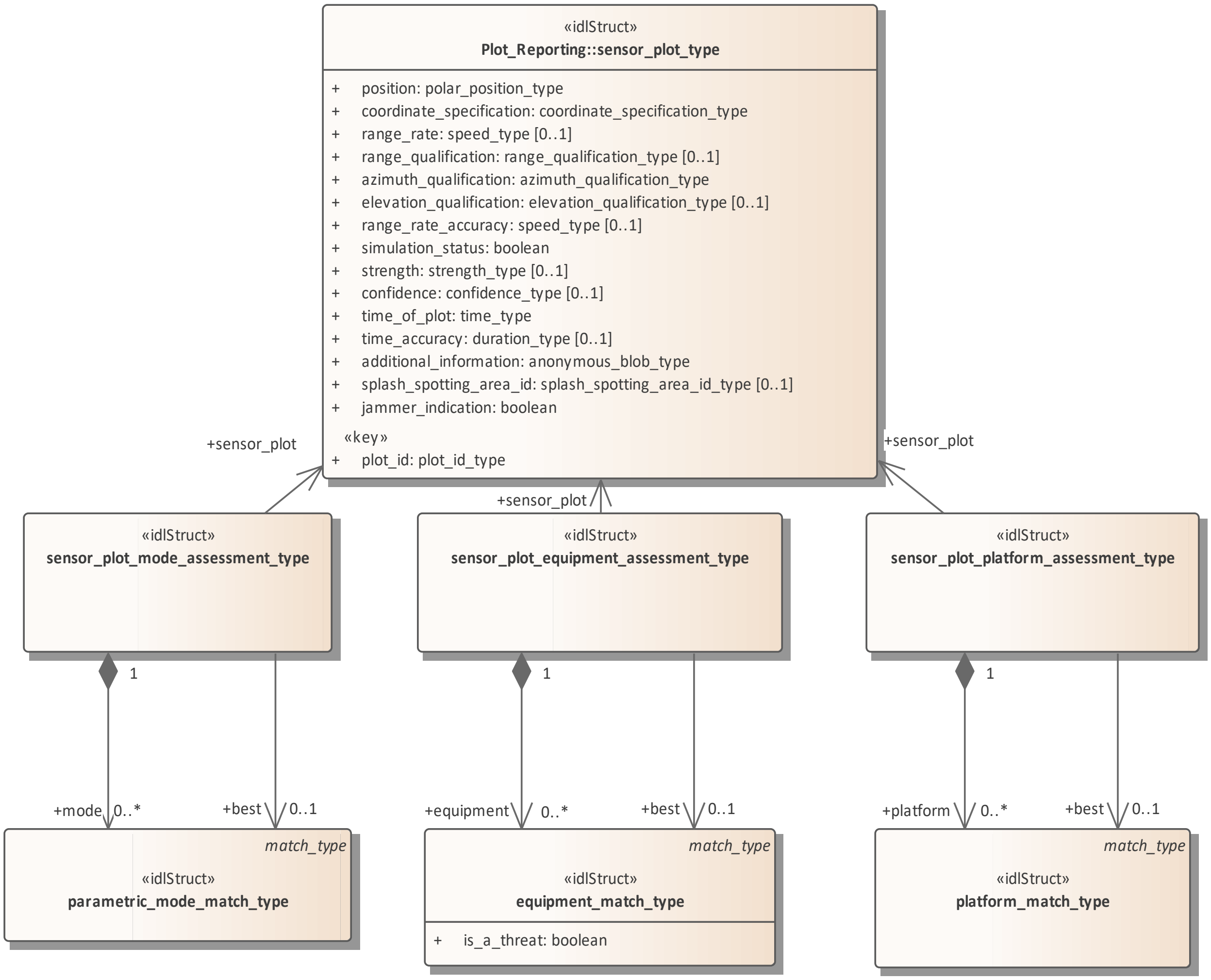


Figure 7.67 Sensor_Assessment - plots (Class diagram)

The classes to support assessment of a sensor track at the equipment and platform level.

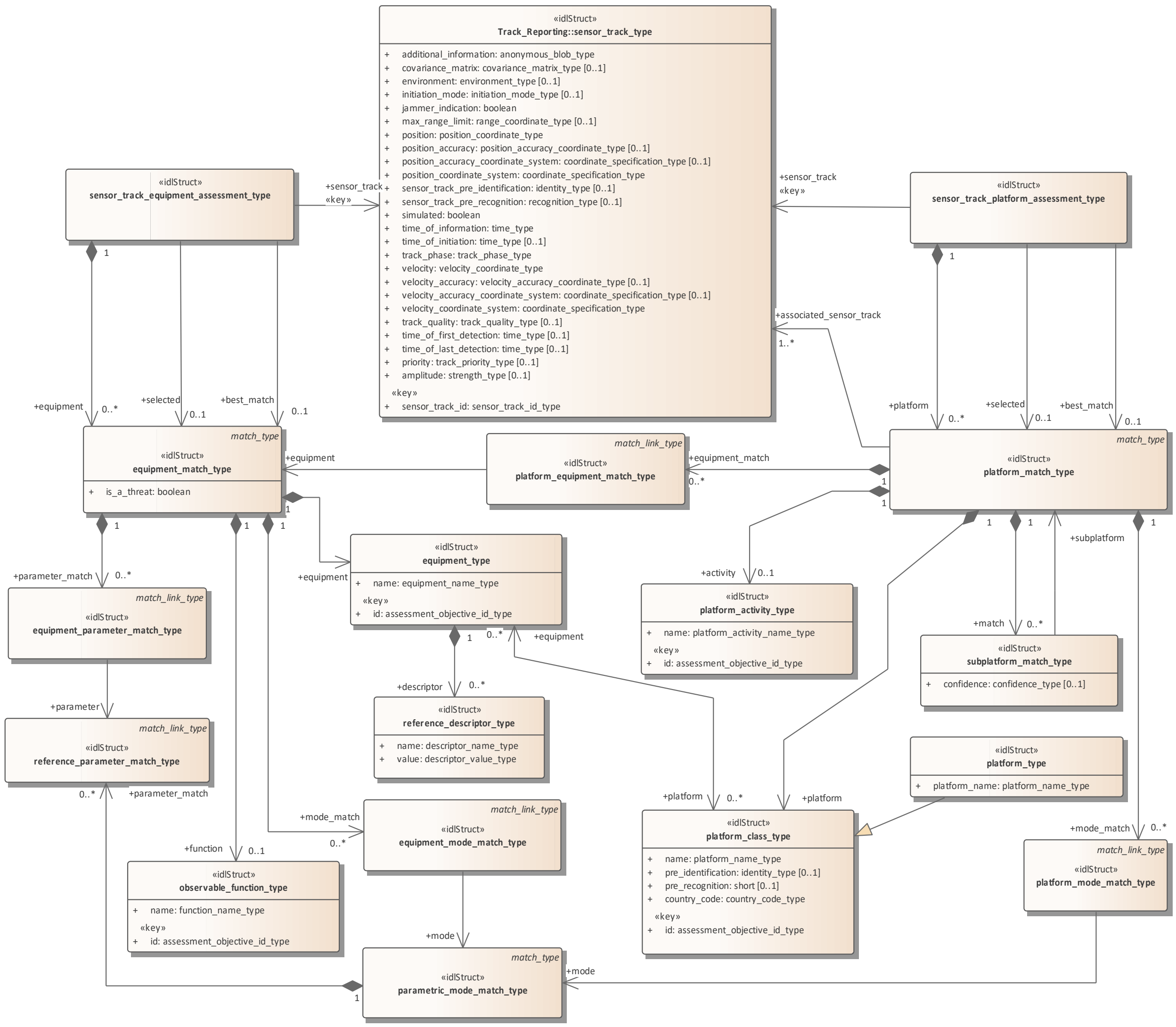


Figure 7.68 Sensor_Assessment - platform (Class diagram)

The classes to support the assessment of a sensor track at the mode level

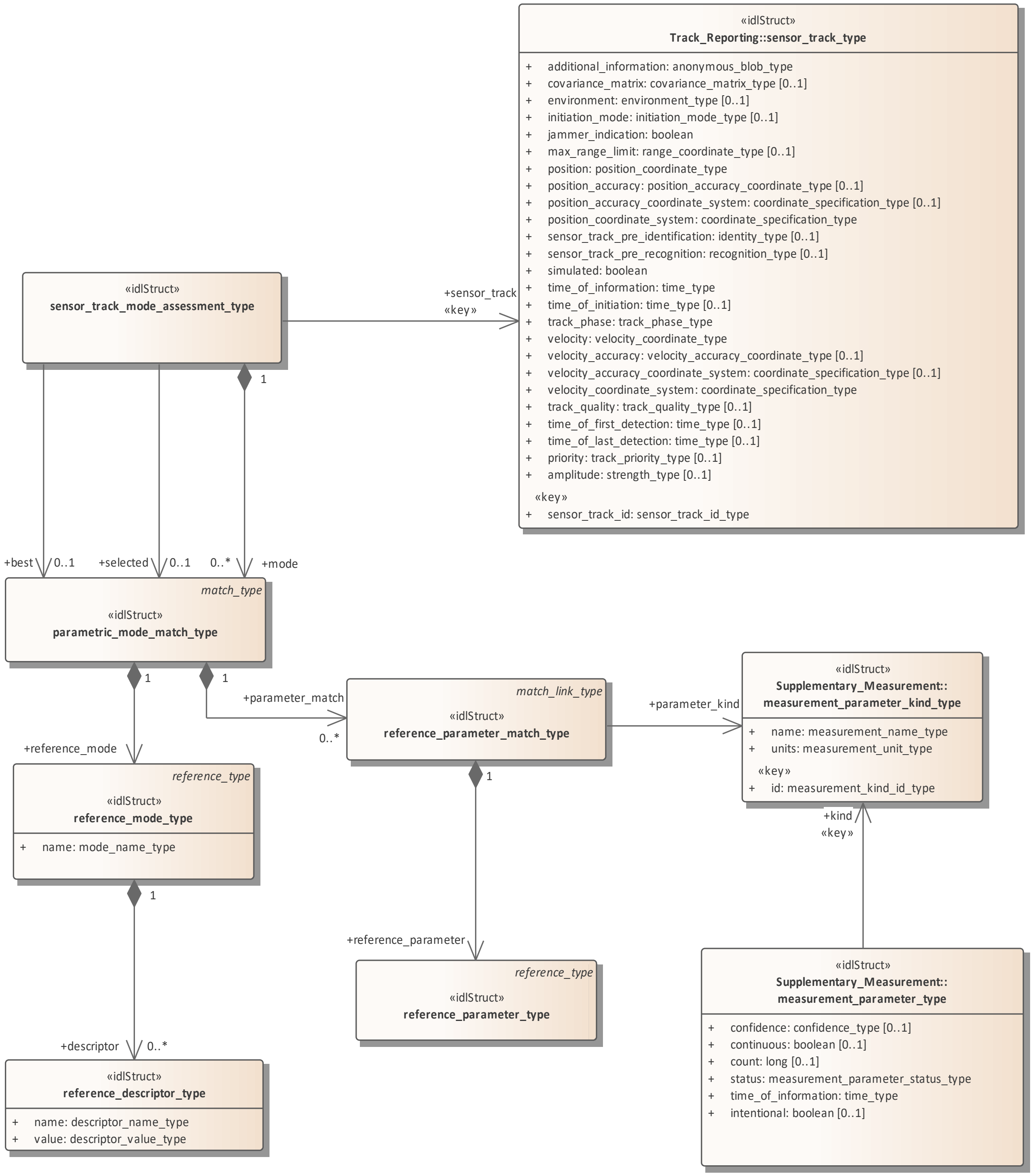


Figure 7.69 Sensor_Assessment - modes (Class diagram)

Basic types to support sensor assessment

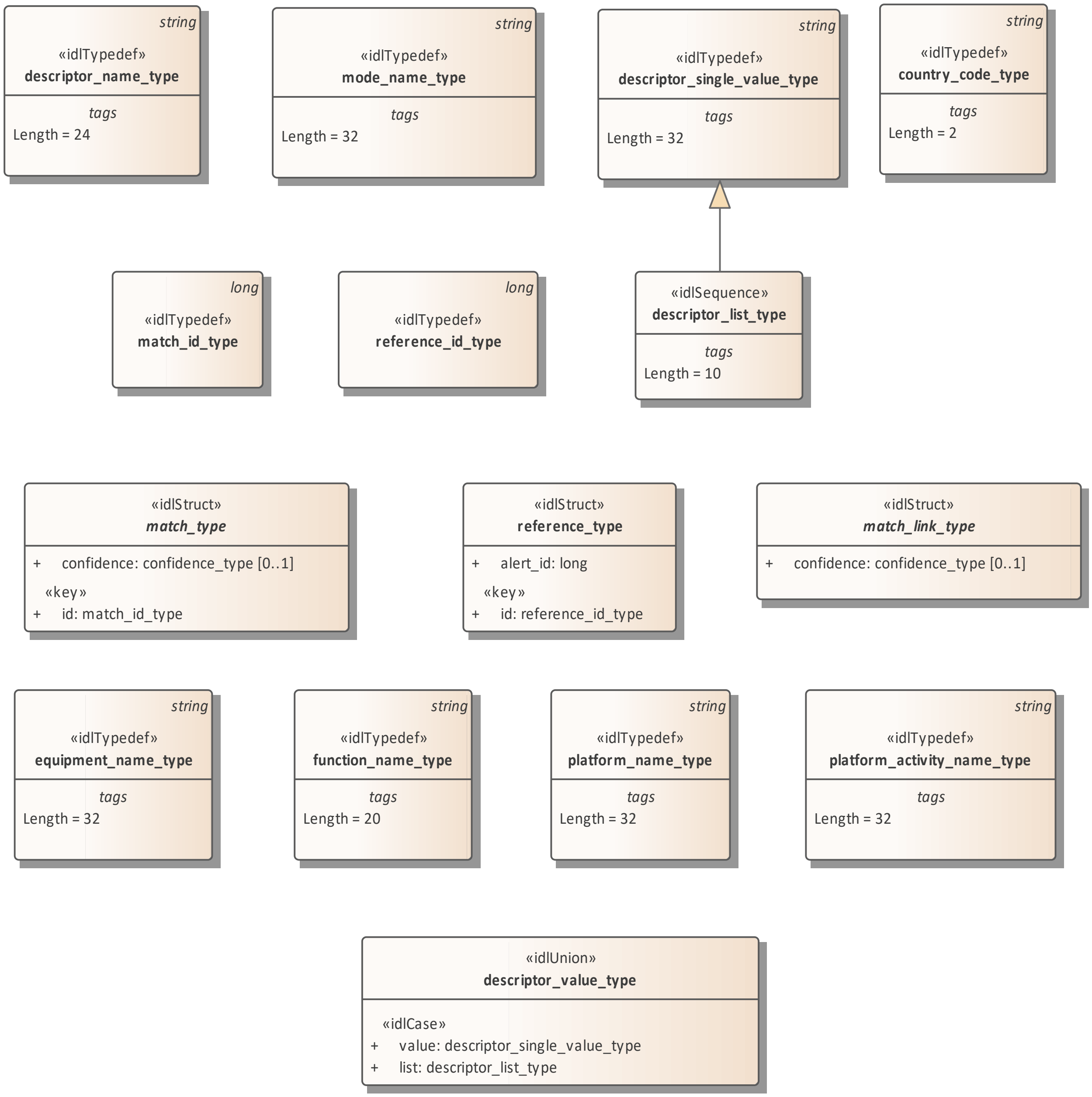


Figure 7.70 Sensor_Assessment - base types (Class diagram)

Classes to support the configuration of supported categories for sensor assessment

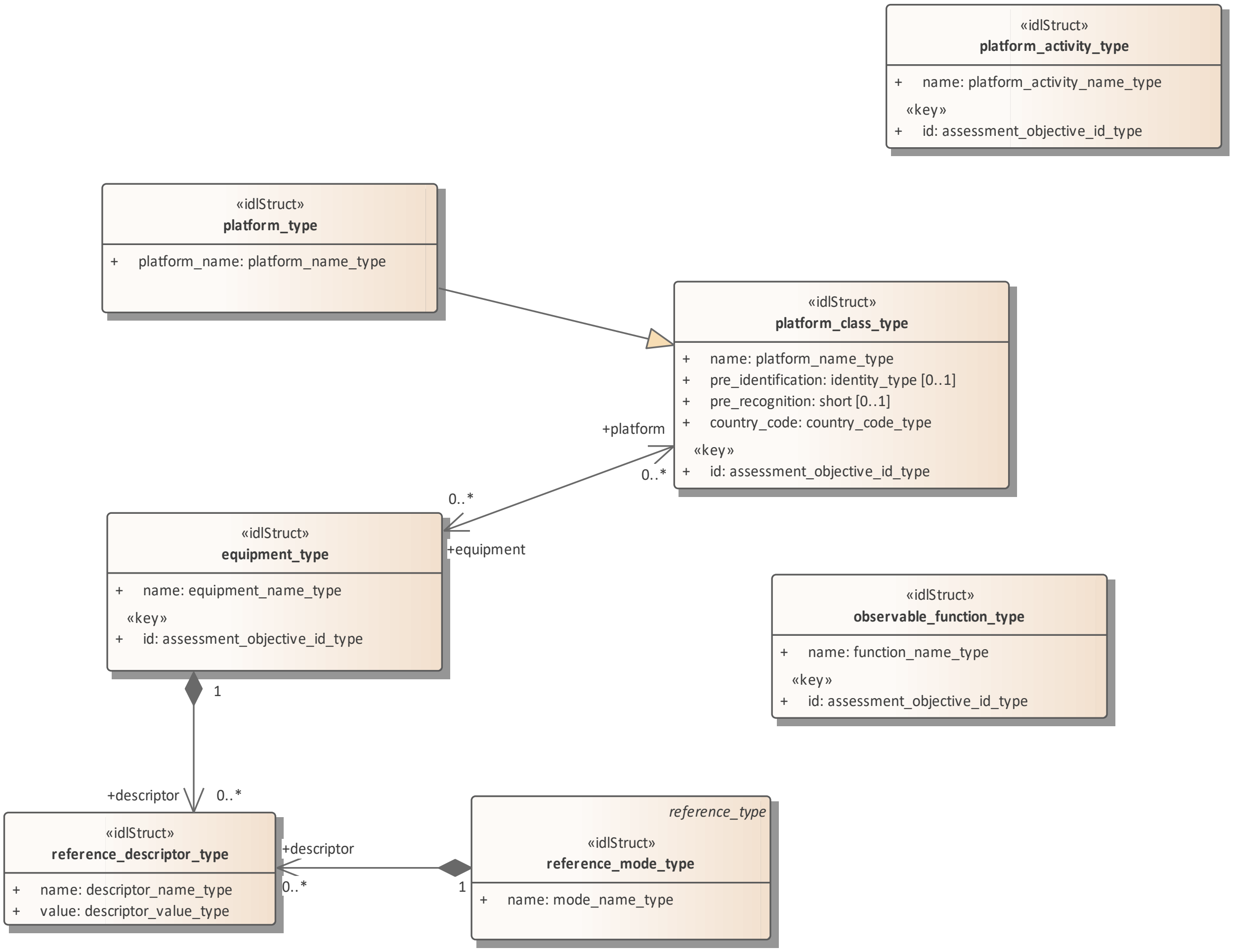


Figure 7.71 Sensor_Assessment - objectives (Class diagram)

Classes to support the assessment of multi-path effects.

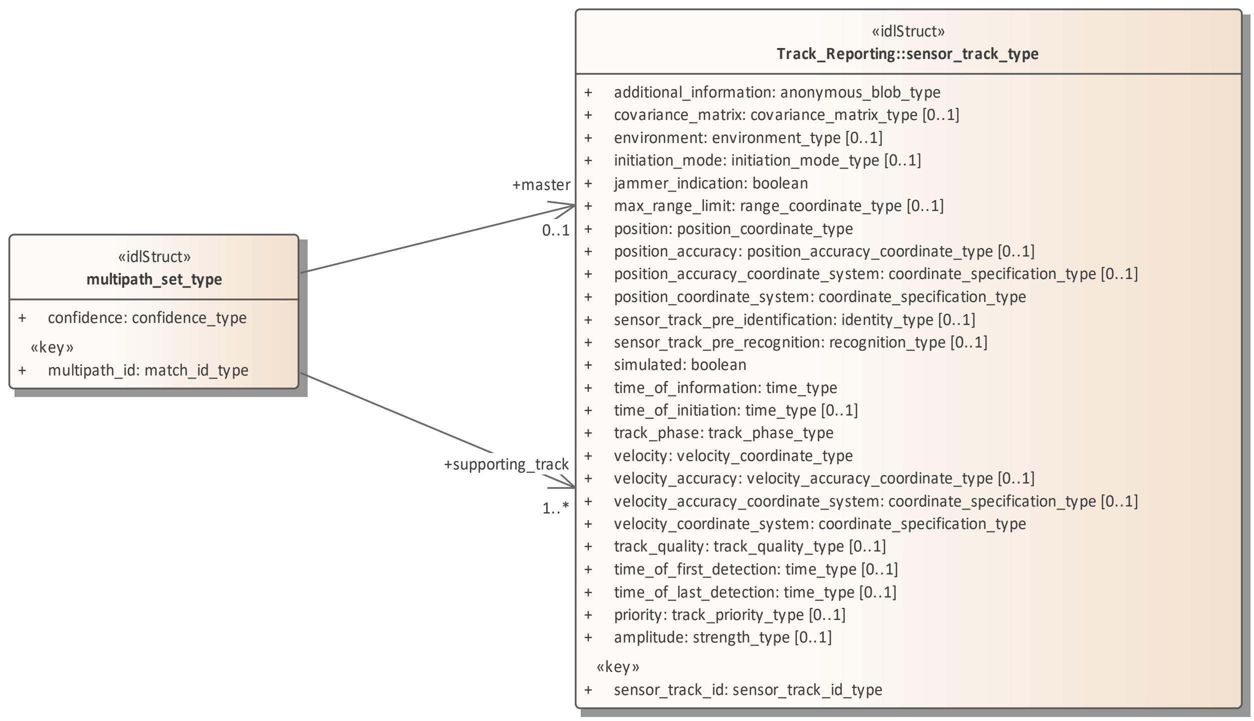


Figure 7.72 Sensor_Assessment - multipath (Class diagram)

7.6.4.1 country_code_type

Type: Class
Package: Sensor_Assessment
Two character (Alpha 2) country code as defined by ISO 3166-1.
An empty string represents undefined data.
ElementTag: Length = 2

7.6.4.2 descriptor_list_type

Type: Class
Package: Sensor_Assessment
list of descriptor values
ElementTag: Length = 10

7.6.4.3 descriptor_name_type

Type: Class
Package: Sensor_Assessment
Represents the name of a descriptor
ElementTag: Length = 24

7.6.4.4 descriptor_single_value_type

Type: Class
Package: Sensor_Assessment
Represents the value of a descriptor
ElementTag: Length = 32

7.6.4.5 descriptor_value_type

Type: IDLUnion
Package: Sensor_Assessment
The value of the descriptor - a single value or a list
ElementTag: switchType = long

Table 7.127 - Attributes of IDLUnion descriptor_value_type

Attribute	Notes
«idlCase» **value** descriptor_single_value_type	The option for a single value
«idlCase» **list** descriptor_list_type	The option for a list of values

7.6.4.6 equipment_match_type

Type: IDLStruct
Package: Sensor_Assessment
The representation of a match between a sensor track and an item of equipment.

Table 7.128 - Attributes of IDLStruct equipment_match_type

Attribute	Notes
is_a_threat boolean	Whether the equipment - function combination is considered to be threatening

Table 7.129 - Relations of IDLStruct equipment_match_type

Connector	Notes
Association: **function** observable_function_type reference [0..1]	The function the equipment has been matched as performing
Association: **equipment** equipment_type reference	
Aggregation: **mode_match** equipment_mode_match_type [0..*]	A match between an equipment an a mode identified in the sensor track's parametric measurements.
Aggregation: **parameter_match** equipment_parameter_match_type [0..*]	A match between an equipment an a measurement parameter of the sensor track.

7.6.4.7 equipment_mode_match_type

Type: IDLStruct
Package: Sensor_Assessment
A match between an equipment match and a mode match. For a possible match to an equipment this represents the linkage to a possible parametric mode that has been identified in the assessment of the sensor track.

Table 7.130 - Relations of IDLStruct equipment_mode_match_type

Connector	Notes
Association: **mode** parametric_mode_match_type reference	The mode from the sensor track's parametric measurements being matched.

7.6.4.8 equipment_name_type

Type: Class
Package: Sensor_Assessment
The name of or label for an item of equipment
ElementTag: Length = 32

7.6.4.9 equipment_parameter_match_type

Type: IDLStruct
Package: Sensor_Assessment
A match between an equipment match and a measurement parameter match. For a possible match to an equipment this represents the linkage to a possible reference parameter that has been identified in the assessment of the sensor track.

Table 7.131 - Relations of IDLStruct equipment_parameter_match_type

Connector	Notes
Association: **parameter** reference_parameter_match_type reference	The parameter from the sensor tracks measurement parameters being matched.

7.6.4.10 equipment_type

Type: IDLStruct
Package: Sensor_Assessment
The representation of an item of equipment that is relevant to assessment of parametric sensor measurements.

Table 7.132 - Attributes of IDLStruct equipment_type

Attribute	Notes
name equipment_name_type	The name of the equipment
«key» **id** assessment_objective_id_type	Unique identifier for the equipment (within the scope of the sensor).

Table 7.133 - Relations of IDLStruct equipment_type

Connector	Notes
Aggregation: **descriptor** reference_descriptor_type [0..*]	Descriptors for the equipment providing supporting, amplifying or qualifying information
Association: **platform** platform_class_type reference [0..*]	The platforms known to contain the equipment

7.6.4.11 function_name_type

Type: Class
Package: Sensor_Assessment
The name of some functional behavior exhibited by an equipment
ElementTag: Length = 20

7.6.4.12 match_id_type

Type: Class
Package: Sensor_Assessment
The unique identifier for a match instance (within the scope of a sensor).

7.6.4.13 match_link_type

Type: Class
Package: Sensor_Assessment
The representation of a link between an assessment match and an existing lower level match.

Table 7.134 - Attributes of Class match_link_type

Attribute	Notes
confidence confidence_type [0..1]	The confidence in the match between a one match and an existing lower level match for a sensor track. This is the result of a statistical hypothesis test.

7.6.4.14 match_type

Type: Class
Package: Sensor_Assessment
An abstract base class for matches between measurements and reference data in the assessment process

Table 7.135 - Attributes of Class match_type

Attribute	Notes
confidence confidence_type [0..1]	The confidence in the match between an equipment and a parametric measurement for a sensor track. This is result of a statistical hypothesis test.
«key» **id** match_id_type	The unique identifier (within the scope of a sensor) for the match instance. Match links for higher-level assessment objectives refer to lower-level matches using this identifier.

7.6.4.15 measurement_element_match_type

Type: Class
Package: Sensor_Assessment

Table 7.136 - Relations of Class measurement_element_match_type

Connector	Notes
Association: **discrete_parameter_element** measurement_element_type reference [1..*]	The discrete parameter measurement value contained in the element of the sequence. Each measurement_element_type instance referred to by an association instance belongs to a different discrete_set_measurement_type instance

7.6.4.16 mode_name_type

Type: Class
Package: Sensor_Assessment
The name or label for a kind of mode that gives rise to a set of measurements.
ElementTag: Length = 32

7.6.4.17 multipath_set_type

Type: Class
Package: Sensor_Assessment

Represents a set of tracks that correspond to the signal which has been measured through the detection of discrete signals that have arrived at the sensor by means of different paths through the environment.

Table 7.137 - Attributes of Class multipath_set_type

Attribute	Notes
«key» **multipath_id** match_id_type	The unique identifier for the multi-path set
confidence confidence_type	The probability that the set represents independently routed detections of the same real world object.

Table 7.138 - Relations of Class multipath_set_type

Connector	Notes
Association: **supporting_track** sensor_track_type reference [1..*]	One of the tracks in the multipath set
Association: **master** sensor_track_type reference [0..1]	The master track for the multipath set

7.6.4.18 observable_function_type

Type: IDLStruct
Package: Sensor_Assessment

The representation of a function observable by the sensor that can be exhibited by equipment detected by the sensor and matched to sensor parametric measurements

Table 7.139 - Attributes of IDLStruct observable_function_type

Attribute	Notes
name function_name_type	The name or label of the function
«key» **id** assessment_objective_id_type	Unique identifier for the function (within the scope of the sensor).

7.6.4.19 parametric_mode_match_type

Type: IDLStruct
Package: Sensor_Assessment

The identification of a mode within a sensor track's parametric data. A mode is a behavior of the real-world object being tracked by the sensor (or a component of that object).

Table 7.140 - Relations of IDLStruct parametric_mode_match_type

Connector	Notes
Association: **reference_mode** reference_mode_type reference	The reference mode being matched
Aggregation: **parameter_match** reference_parameter_match_type [0..*]	The reference parameters, which have been matched by the measurement parameters in determining the mode assessment.

7.6.4.20 platform_activity_name_type

Type: Class
Package: Sensor_Assessment
The name of or label for an activity that can be undertaken by a platform.
ElementTag: Length = 32

7.6.4.21 platform_activity_type

Type: IDLStruct
Package: Sensor_Assessment
An activity that can be undertaken by a platform

Table 7.141 - Attributes of IDLStruct platform_activity_type

Attribute	Notes
name platform_activity_name_type	The name of the platform's activity
«key» **id** assessment_objective_id_type	Unique identifier for the activity (within the scope of the sensor).

7.6.4.22 platform_class_type

Type: IDLStruct
Package: Sensor_Assessment
The class of an individual platform instance - i.e. a common design from which platform instances are manufactured. This contains attributes that apply to all the platform instances of a class

Table 7.142 - Attributes of IDLStruct platform_class_type

Attribute	Notes
name platform_name_type	The name of the platform (or class of platforms)
pre_identification identity_type [0..1]	The standard identification of the platform
pre_recognition short [0..1]	The discrete code representing the type of platform.
country_code country_code_type	The code representing the country of registration of the platform
«key» **id** assessment_objective_id_type	Unique identifier for the platform (within the scope of the sensor).

Table 7.143 - Relations of IDLStruct platform_class_type

Connector	Notes
Association: **equipment** equipment_type reference [0..*]	The equipment known to be associated with a platform

7.6.4.23 platform_equipment_match_type

Type: IDLStruct
Package: Sensor_Assessment
Represents the matching link between a platform and a constituent piece of equipment

Table 7.144 - Relations of IDLStruct platform_equipment_match_type

Connector	Notes
Association: **equipment** equipment_match_type reference	

7.6.4.24 platform_match_type

Type: IDLStruct
Package: Sensor_Assessment
The representation of a match between a sensor track and a platform.

Table 7.145 - Relations of IDLStruct platform_match_type

Connector	Notes
Association: **associated_sensor_track** sensor_track_type reference [1..*]	The set of sensor tracks that are all associated with the same platform instance and hence real world object under the hypothesis of this platform match.
Association: **activity** platform_activity_type reference [0..1]	The activity identified as being undertaken by the platform when matching the sensor track to it.
Association: **platform** platform_class_type reference	The platform being matched
Aggregation: **mode_match** platform_mode_match_type [0..*]	The observable equipment modes matched by the sensor in determining the platform match.
Aggregation: **match** subplatform_match_type [0..*]	A hierarchical match from a (super) platform to a separable (sub) platform that it is potentially hosting or carrying as one of its constituent parts.
Aggregation: **equipment_match** platform_equipment_match_type [0..*]	A match between a platform and a constituent equipment

7.6.4.25 platform_mode_match_type

Type: IDLStruct
Package: Sensor_Assessment
Represents the matching link between a platform and a mode of a constituent piece of equipment

Table 7.146 - Relations of IDLStruct platform_mode_match_type

Connector	Notes
Association: **mode** parametric_mode_match_type reference	

7.6.4.26 platform_name_type

Type: Class
Package: Sensor_Assessment
The name or label for a platform or class of platforms. A platform being a discrete independently acting object in the real-world environment.
ElementTag: Length = 32

7.6.4.27 platform_type

Type: IDLStruct
Package: Sensor_Assessment
The representation of a platform that an assessment of sensor track data can match against.

Table 7.147 - Attributes of IDLStruct platform_type

Attribute	Notes
platform_name platform_name_type	The name of the platform (or class of platforms)

7.6.4.28 reference_descriptor_type

Type: Class
Package: Sensor_Assessment
The representation of descriptor for a configuration reference data instance. Reference descriptor instances qualify the reference data instance (e.g. mode, equipment) and are a mechanism to specify aliases and other supporting information.

Table 7.148 - Attributes of Class reference_descriptor_type

Attribute	Notes
name descriptor_name_type	The name of the descriptor for the mode
value descriptor_value_type	The value of the descriptor for the mode

7.6.4.29 reference_id_type

Type: Class
Package: Sensor_Assessment
The unique identifier for a reference parameter, sequence or mode.

7.6.4.30 reference_mode_type

Type: Class
Package: Sensor_Assessment
This class represents a label for a reference mode for a sensor tracks measurement parameter. Such tactically significant labels and their underlying data sets may be made available as an encyclopedic library.

Table 7.149 - Attributes of Class reference_mode_type

Attribute	Notes
name mode_name_type	The name or label of the mode.

Table 7.150 - Relations of Class reference_mode_type

Connector	Notes
Aggregation: **descriptor** reference_descriptor_type [0..*]	The descriptors associated with the mode

7.6.4.31 reference_parameter_match_type

Type: Class
Package: Sensor_Assessment
The representation of a match to a reference parameter for a sensor track

Table 7.151 - Relations of Class reference_parameter_match_type

Connector	Notes
Association: **parameter_kind** measurement_parameter_kind_type reference	The kind of the measured parameters that support the mode identification.
Association: **reference_parameter** reference_parameter_type reference	The reference parameters matched by this mode identification

7.6.4.32 reference_parameter_type

Type: Class
Package: Sensor_Assessment
This class represents a label for a reference value, set or distribution for a parameter. Such tactically significant labels and their underlying data sets may be made available as an encyclopedic library.

7.6.4.33 reference_type

Type: IDLStruct
Package: Sensor_Assessment
A base class for reference data being compared with measurements in the assessment process.

Table 7.152 - Attributes of IDLStruct reference_type

Attribute	Notes
«key» **id** reference_id_type	The unique identifier for the reference data. This may facilitate the retrieval of additional data outside the scope of this specification.
alert_id long	The identifier for an alerting or warning process associated with the matching of this reference data.

7.6.4.34 sensor_plot_equipment_assessment_type

Type: IDLStruct
Package: Sensor_Assessment
The sensor subsystem's assessment of the equipment potentially matched by the sensor plot's measurement parameters.

Table 7.153 - Relations of IDLStruct sensor_plot_equipment_assessment_type

Connector	Notes
Association: **best** equipment_match_type reference [0..1]	The equipment assessed by the sensor as being the best match for the sensor plot
Association: **sensor_plot** sensor_plot_type reference	The sensor plot to which the assessment refers. The plot instance must contain a plot_id attribute.
Aggregation: **equipment** equipment_match_type [0..*]	The equipment assessed as potentially being represented by the sensor plot

7.6.4.35 sensor_plot_mode_assessment_type

Type: IDLStruct
Package: Sensor_Assessment
The sensor subsystem's assessment of the modes potentially matched by the sensor plot's measurement parameters.

Table 7.154 - Relations of IDLStruct sensor_plot_mode_assessment_type

Connector	Notes
Association: **sensor_plot** sensor_plot_type reference	The sensor plot to which the assessment refers. The plot instance must contain a plot_id attribute.
Association: **best** parametric_mode_match_type reference [0..1]	The mode assessed as most likely by the sensor.
Aggregation: **mode** parametric_mode_match_type [0..*]	The modes assessed as candidates for the sensor plot.

7.6.4.36 sensor_plot_platform_assessment_type

Type: IDLStruct
Package: Sensor_Assessment
The sensor subsystem's assessment of the platforms potentially matched by the sensor plot's measurement parameters.

Table 7.155 - Relations of IDLStruct sensor_plot_platform_assessment_type

Connector	Notes
Association: **sensor_plot** sensor_plot_type reference	The sensor plot to which the assessment refers. The plot instance must contain a plot_id attribute.
Association: **best** platform_match_type reference [0..1]	The platform assessed by the sensor as the best match for the sensor plot.
Aggregation: **platform** platform_match_type [0..*]	The platforms assessed as potentially being represented by the sensor plot

7.6.4.37 sensor_track_equipment_assessment_type

Type: IDLStruct
Package: Sensor_Assessment
A representation of an assessment of the equipment that potentially correspond to a sensor track.

Table 7.156 - Relations of IDLStruct sensor_track_equipment_assessment_type

Connector	Notes
Association: **sensor_track** sensor_track_type reference	
Association: **best_match** equipment_match_type reference [0..1]	The equipment assessed as most likely to correspond to the sensor track
Association: **selected** equipment_match_type reference [0..1]	The match selected as the authoritative assessment by command
Association: **equipment** equipment_match_type reference [0..*]	An item of equipment that has been assessed as a possible match for the sensor track

7.6.4.38 sensor_track_mode_assessment_type

Type: Class
Package: Sensor_Assessment
The representation of the state of the assessment of a sensor track's possible identified modes.

Table 7.157 - Relations of Class sensor_track_mode_assessment_type

Connector	Notes
Association: **sensor_track** sensor_track_type reference	The sensor track to which the mode assessment relates
Association: **selected** parametric_mode_match_type reference [0..1]	The match that has been authoritatively selected as the mode relating to the sensor track.
Association: **best** parametric_mode_match_type reference [0..1]	The sensor's best match
Aggregation: **mode** parametric_mode_match_type [0..*]	The parametric modes matched in the assessment of the sensor track's measurement parameters.

7.6.4.39 sensor_track_platform_assessment_type

Type: IDLStruct
Package: Sensor_Assessment

A representation of an assessment of the platforms that potentially correspond to a sensor track.

Table 7.158 - Relations of IDLStruct sensor_track_platform_assessment_type

Connector	Notes
Association: **sensor_track** sensor_track_type reference	The sensor track corresponding to the platform assessment
Association: **selected** platform_match_type reference [0..1]	The match selected as the authoritative assessment by command
Association: **best_match** platform_match_type reference [0..1]	The platform assessed as most likely to correspond to the sensor track
Association: **platform** platform_match_type reference [0..*]	A platform that has been assessed as a possible match for the sensor track

7.6.4.40 subplatform_match_type

Type: IDLStruct
Package: Sensor_Assessment

Table 7.159 - Attributes of IDLStruct subplatform_match_type

Attribute	Notes
confidence confidence_type [0..1]	The confidence in the match between a (super) platform and a potential constituent sub-platform. This is result of a statistical hypothesis test.

Table 7.160 - Relations of IDLStruct subplatform_match_type

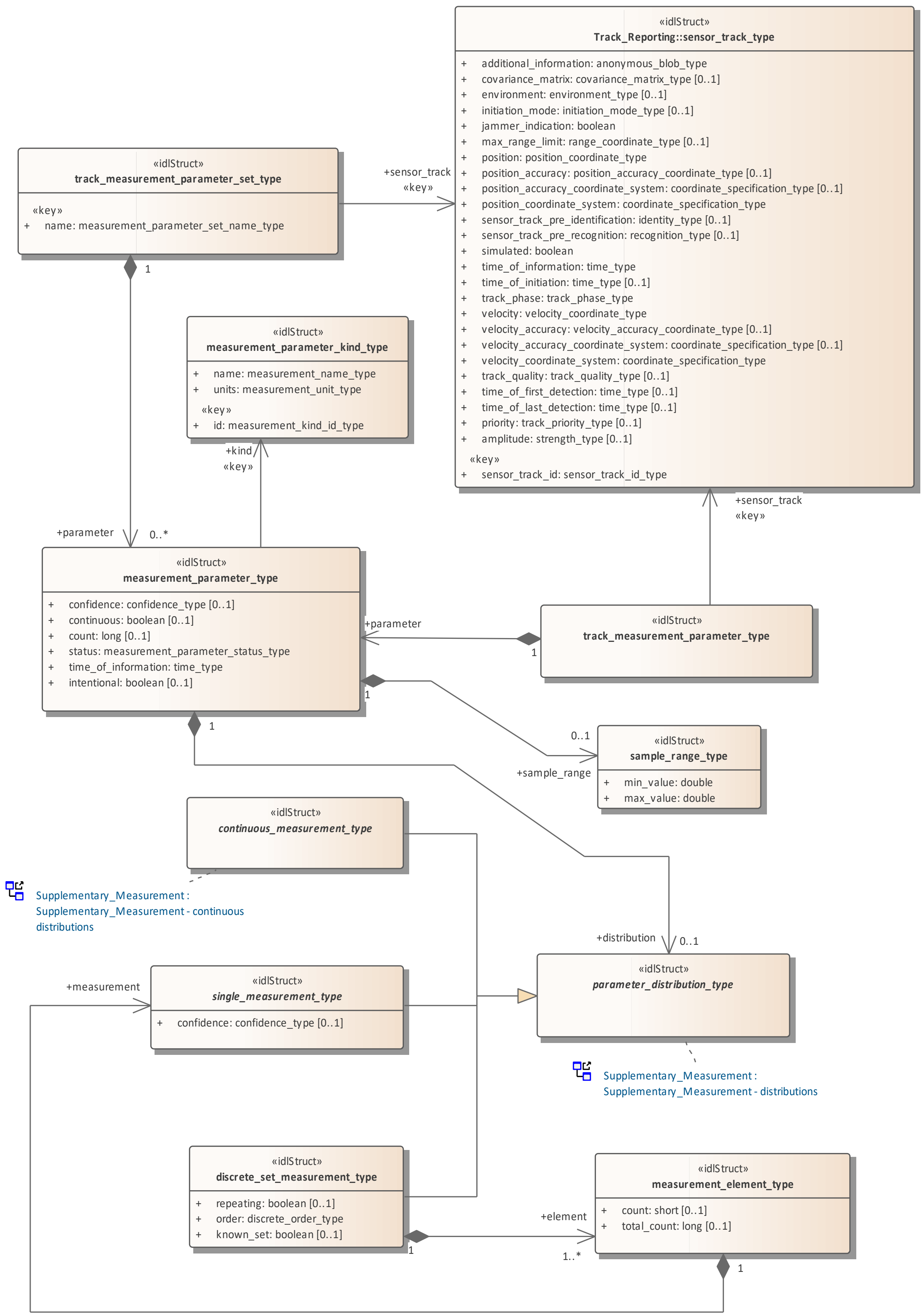
Connector	Notes
Association: **subplatform** platform_match_type reference	A (sub) platform (potentially independently operating real-world object) that is currently contained by the platform. Examples include a helicopter that is currently on a ship's landing deck.

7.6.5 Supplementary_Measurement

Parent Package: Sensor_Domain

This package provides a data model to describe supplementary parameters that a sensor can provide about a sensor track. The approach is to be agnostic to any specific type of measurement that a sensor may make. Rather, classes are provided that allow the sensor to describe the parameters that it supports and then to describe the measurements that it has made of those parameters. Measurement can be treated as a single instance, a continuous range or a discrete set; the quantity can be scalar, vector, discrete or qualitative; and confidence values can be supplied where appropriate as can accuracy estimates.

Supplementary measurements as relating to a sensor track.



«idlStruct»
Track_Reporting::sensor_track_type
+ additional_information: anonymous_blob_type
+ covariance_matrix: covariance_matrix_type [0..1]
+ environment: environment_type [0..1]
+ initiation_mode: initiation_mode_type [0..1]
+ jammer_indication: boolean
+ max_range_limit: range_coordinate_type [0..1]
+ position: position_coordinate_type
+ position_accuracy: position_accuracy_coordinate_type [0..1]
+ position_accuracy_coordinate_system: coordinate_specification_type [0..1]
+ position_coordinate_system: coordinate_specification_type
+ sensor_track_pre_identification: identity_type [0..1]
+ sensor_track_pre_recognition: recognition_type [0..1]
+ simulated: boolean
+ time_of_information: time_type
+ time_of_initiation: time_type [0..1]
+ track_phase: track_phase_type
+ velocity: velocity_coordinate_type
+ velocity_accuracy: velocity_accuracy_coordinate_type [0..1]
+ velocity_accuracy_coordinate_system: coordinate_specification_type [0..1]
+ velocity_coordinate_system: coordinate_specification_type
+ track_quality: track_quality_type [0..1]
+ time_of_first_detection: time_type [0..1]
+ time_of_last_detection: time_type [0..1]
+ priority: track_priority_type [0..1]
+ amplitude: strength_type [0..1]
«key»
+ sensor_track_id: sensor_track_id_type
«idlStruct»
track_measurement_parameter_set_type
«key»
+ name: measurement_parameter_set_name_type
+sensor_track
«key»
1
«idlStruct»
measurement_parameter_kind_type
+ name: measurement_name_type
+ units: measurement_unit_type
«key»
+ id: measurement_kind_id_type
+kind
«key»
+parameter
0..*
+sensor_track
«key»
«idlStruct»
measurement_parameter_type
+ confidence: confidence_type [0..1]
+ continuous: boolean [0..1]
+ count: long [0..1]
+ status: measurement_parameter_status_type
+ time_of_information: time_type
+ intentional: boolean [0..1]
+parameter
1
«idlStruct»
track_measurement_parameter_type
1
1
0..1
«idlStruct»
sample_range_type
+sample_range
+ min_value: double
+ max_value: double
«idlStruct»
continuous_measurement_type
Supplementary_Measurement :
Supplementary_Measurement - continuous
distributions
+distribution
0..1
«idlStruct»
parameter_distribution_type
+measurement
«idlStruct»
single_measurement_type
+ confidence: confidence_type [0..1]
Supplementary_Measurement :
Supplementary_Measurement - distributions
«idlStruct»
discrete_set_measurement_type
+ repeating: boolean [0..1]
+ order: discrete_order_type
+ known_set: boolean [0..1]
«idlStruct»
measurement_element_type
+ count: short [0..1]
+ total_count: long [0..1]
+element
1
1..*
1

Figure 7.73 Supplementary_Measurement - tracks (Class diagram)

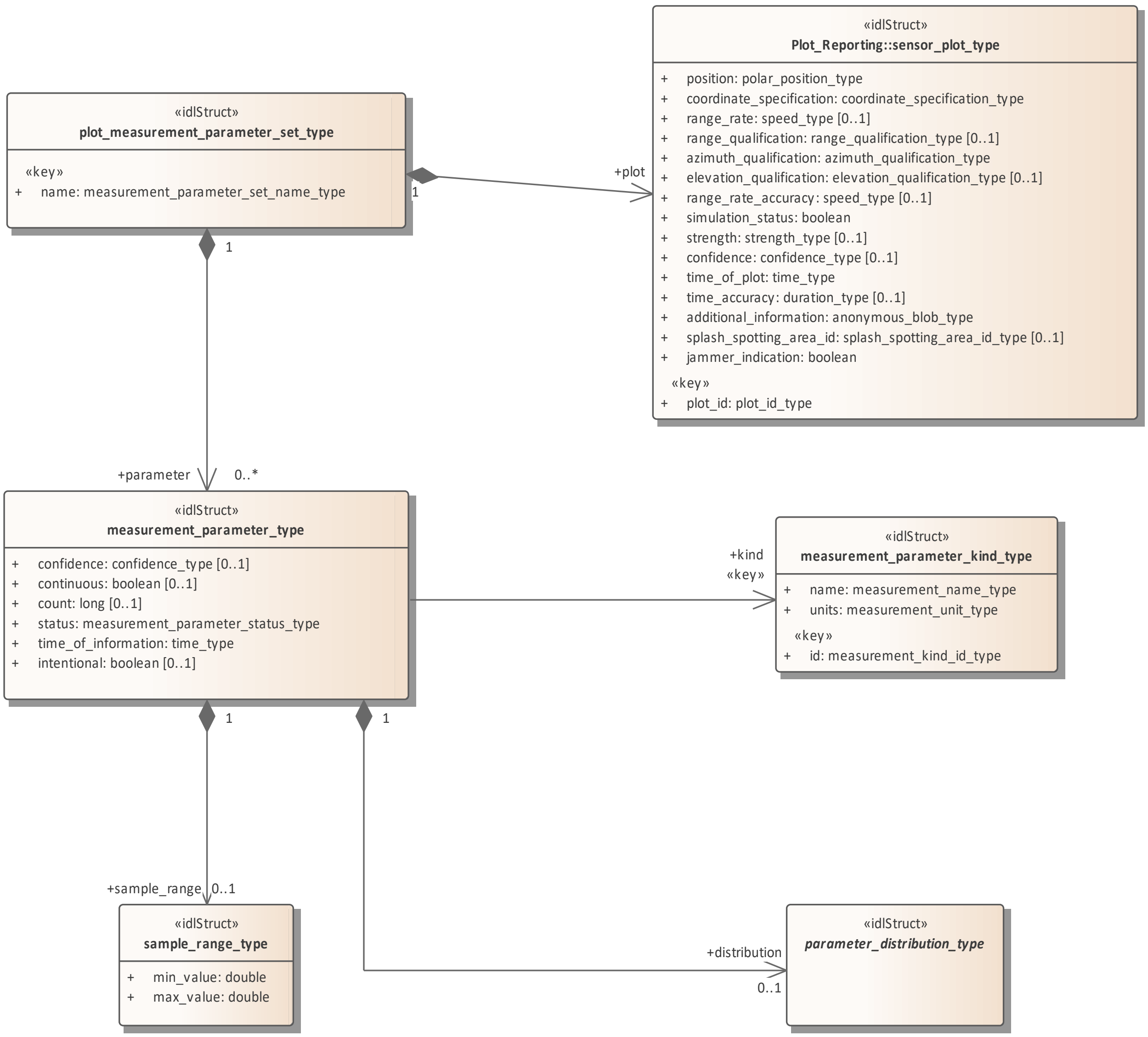


Figure 7.74 Supplementary_Measurement - plots (Class diagram)

Base types for supplementary measurements.

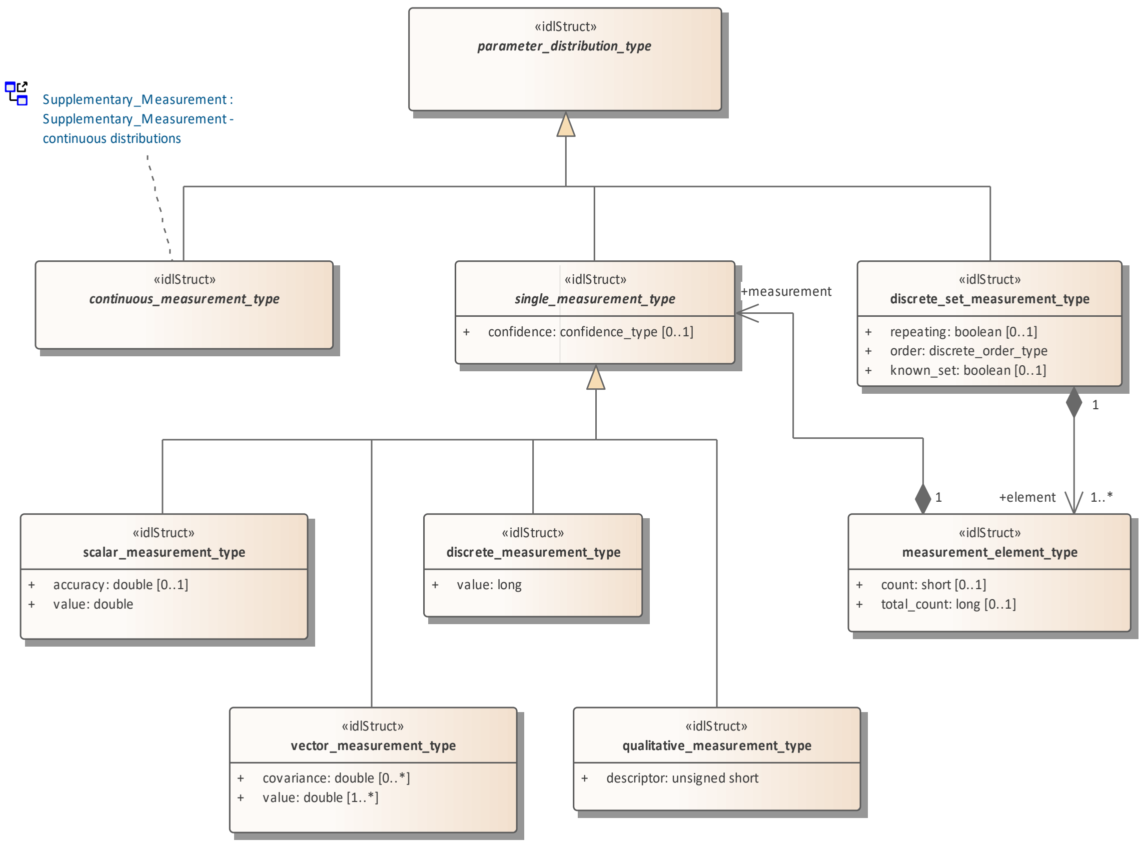


Figure 7.75 Supplementary_Measurement - distributions (Class diagram)

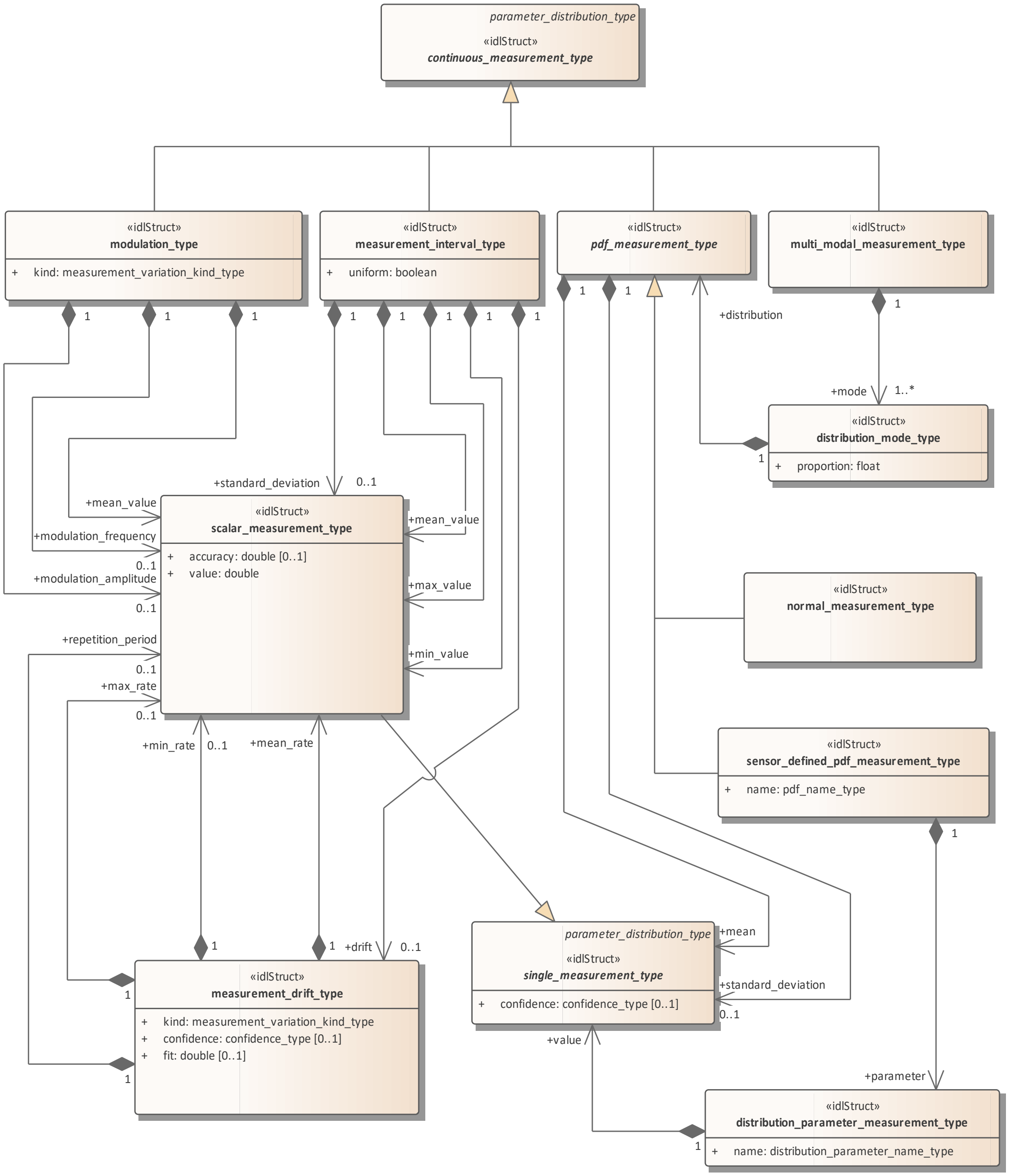


Figure 7.76 Supplementary_Measurement - continuous distributions (Class diagram)

Figure 7.77 Supplementary_Measurement - base types (Class diagram)

7.6.5.1 continuous_measurement_type

Type: IDLStruct
Package: Supplementary_Measurement
A continuous representation of a parameter measurement value.

7.6.5.2 discrete_measurement_type

Type: Class
Package: Supplementary_Measurement
This represents a parameter which takes discrete values.

Table 7.161 - Attributes of Class discrete_measurement_type

Attribute	Notes
value long	The discrete value of the measurement

7.6.5.3 discrete_order_type

Type: IDLEnum
Package: Supplementary_Measurement
The ordering semantics of a set of measurements of a parameter.

Table 7.162 - Attributes of IDLEnum discrete_order_type

Attribute	Notes
«enum» **ORDER_INCREASING_BY_VALUE**	The measurements are ordered by increasing value such that the smallest value is first and the largest value is last.
«enum» **ORDER_DECREASING_BY_VALUE**	The measurements are ordered by decreasing value such that the smallest value is last and the largest value is first.
«enum» **ORDER_OLDEST_FIRST**	The measurements are ordered by age such that the value received first is first and the latest value is last.
«enum» **ORDER_MOST_RECENT_FIRST**	The measurements are ordered by age such that the value received first is last and the latest value is first.
«enum» **ORDER_NOT_SPECIFIED**	The ordering is not specified as it is not semantically meaningful.

7.6.5.4 discrete_set_measurement_type

Type: IDLStruct
Package: Supplementary_Measurement
The values of the measurement parameter follow a discrete distribution

Table 7.163 - Attributes of IDLStruct discrete_set_measurement_type

Attribute	Notes
repeating boolean [0..1]	Whether the elements within the discrete distribution repeat (in the same order)
order discrete_order_type	The semantics of the ordering of the elements of the discrete distribution
known_set boolean [0..1]	Whether the elements within the discrete distribution correspond to a known set of measurement values

Table 7.164 - Relations of IDLStruct discrete_set_measurement_type

Connector	Notes
Aggregation: **element** measurement_element_type [1..*]	A discrete element within the discrete distribution

7.6.5.5 distribution_mode_type

Type: IDLStruct
Package: Supplementary_Measurement
The distribution of one mode independently contributing to a multi-modal distribution.

Table 7.165 - Attributes of IDLStruct distribution_mode_type

Attribute	Notes
proportion float	The proportion that this mode contributes to the overall distribution. The sum of all the modes equals 1.

Table 7.166 - Relations of IDLStruct distribution_mode_type

Connector	Notes
Aggregation: **distribution** pdf_measurement_type	The distribution for this mode.

7.6.5.6 distribution_parameter_measurement_type

Type: IDLStruct
Package: Supplementary_Measurement
The measurement estimation of a (sensor defined) parameter describing a measurement parameter's distribution

Table 7.167 - Attributes of IDLStruct distribution_parameter_measurement_type

Attribute	Notes
name distribution_parameter_name_type	The name of a parameter describing a distribution

Table 7.168 - Relations of IDLStruct distribution_parameter_measurement_type

Connector	Notes
Aggregation: **value** single_measurement_type	The measured value for a parameter describing a distribution.

7.6.5.7 distribution_parameter_name_type

Type: Class
Package: Supplementary_Measurement
The name of a sensor defined probability density function.
ElementTag: Length = 20

7.6.5.8 measurement_drift_type

Type: IDLStruct
Package: Supplementary_Measurement
Describes how a measurement varies with time (on a time-scale longer than that described by modulation measurements).

Table 7.169 - Attributes of IDLStruct measurement_drift_type

Attribute	Notes
kind measurement_variation_kind_type	The qualitative measure of the kind of drift detected.
confidence confidence_type [0..1]	The sensor's confidence in identifying the kind of drift.
fit double [0..1]	Sample size independent measure of the closeness by which the measurement sample fit the model of the identified kind of drift.

Table 7.170 - Relations of IDLStruct measurement_drift_type

Connector	Notes
Aggregation: **max_rate** scalar_measurement_type [0..1]	The maximum rate of change of the measurement detected (in the units of the measurement's kind per second).
Aggregation: **mean_rate** scalar_measurement_type	The mean rate of change of the measurement detected (in the units of the measurement's kind per second).
Aggregation: **repetition_period** scalar_measurement_type [0..1]	The time for the drift behavior to repeat itself.

Connector	Notes
Aggregation: **min_rate** scalar_measurement_type [0..1]	The minimum rate of change of the measurement detected (in the units of the measurement's kind per second).

7.6.5.9 measurement_element_type

Type: IDLStruct
Package: Supplementary_Measurement
The representation of an element within a discrete distribution

Table 7.171 - Attributes of IDLStruct measurement_element_type

Attribute	Notes
count short [0..1]	The number of times the parameter measured has corresponded to this element in a row. That is since another element was measured.
total_count long [0..1]	The total number of times this element has been measured with the discrete distribution for this parameter value for the sensor track.

Table 7.172 - Relations of IDLStruct measurement_element_type

Connector	Notes
Aggregation: **measurement** single_measurement_type	The measurement of the element within the discrete distribution.

7.6.5.10 measurement_interval_type

Type: IDLStruct
Package: Supplementary_Measurement
The representation of parameter measurement values that are distributed within a bounded interval

Table 7.173 - Attributes of IDLStruct measurement_interval_type

Attribute	Notes
uniform boolean	The measurement values are uniformly distributed within the bounded interval

Table 7.174 - Relations of IDLStruct measurement_interval_type

Connector	Notes
Aggregation: **drift** measurement_drift_type [0..1]	A qualitative description of how the parameter measurement changes over time
Aggregation: **min_value** scalar_measurement_type	The lower bound of the parameter measurements, which may be extrapolated from actual sample measurements received or processed.
Aggregation: **mean_value** scalar_measurement_type	The mean value of the measurement within the interval.
Aggregation: **max_value** scalar_measurement_type	The upper bound of the parameter measurements, which may be extrapolated from actual sample measurements received or processed.
Aggregation: **standard_deviation** scalar_measurement_type [0..1]	The standard deviation of the measurement within the interval.

7.6.5.11 measurement_kind_id_type

Type: Class
Package: Supplementary_Measurement
The unique identifier for describing kinds of measurements

7.6.5.12 measurement_name_type

Type: Class
Package: Supplementary_Measurement
The name or label for a kind of measurement
ElementTag: Length = 20

7.6.5.13 measurement_parameter_kind_type

Type: Class
Package: Supplementary_Measurement
Describes a kind of measurement parameters in terms of its meta-data (the information that applies to all measurement_parameter_type instances)

Table 7.175 - Attributes of Class measurement_parameter_kind_type

Attribute	Notes
name measurement_name_type	The name or label for the measurement.
«key» **id** measurement_kind_id_type	The unique identifier for this kind of parameter.
units measurement_unit_type	The units of the measurement.

7.6.5.14 measurement_parameter_set_name_type

Type: IDLTypeDef
Package: Supplementary_Measurement
Names of sets of parameters for sensor tracks
ElementTag: Length = 32

7.6.5.15 measurement_parameter_status_type

Type: IDLEnum
Package: Supplementary_Measurement
The measurement status of the parameter

Table 7.176 - Attributes of IDLEnum measurement_parameter_status_type

Attribute	Notes
«enum» **CURRENT**	The parameter is currently subject to measurement for this sensor track
«enum» **MISSING**	The parameter temporarily cannot be measured for this sensor track

Attribute	Notes
«enum» **NOT_PRESENT**	The parameter can no longer be measured for this sensor track
«enum» **NO_PARAMETER_STATUS**	No statement is available regarding the parameter's measurement status for this sensor track

7.6.5.16 measurement_parameter_type

Type: IDLStruct
Package: Supplementary_Measurement
A measurement of a parameter by a sensor for a sensor track.

Table 7.177 - Attributes of IDLStruct measurement_parameter_type

Attribute	Notes
confidence confidence_type [0..1]	The probability that the measurement corresponds to measure labelled for the sensor track.
continuous boolean [0..1]	Indicates that the phenomenon being measured is in an enduring steady state and hence that complementary/orthogonal measurements of more detailed time-varying characteristics/phenomena are not present for this track.
count long [0..1]	The number of coherent discrete measurements of this quantity.If the sensor detects a qualitative change then the count is reset.
status measurement_parameter_status_type	The measurement status of the parameter
time_of_information time_type	The time at which the parameter was measured
intentional boolean [0..1]	Whether or not the phenomenon being measured by this parameter is considered to be design feature of the equipment causing the phenomenon.

Table 7.178 - Relations of IDLStruct measurement_parameter_type

Connector	Notes
Association: **kind** measurement_parameter_kind_type reference	The description of the parameter and the unique identifier (within the scope of a sensor track) of a measurement parameter instance.
Aggregation: **distribution** parameter_distribution_type [0..1]	The representation of the statistical distribution of the measurement parameter.
Aggregation: **sample_range** sample_range_type [0..1]	The range in which samples contributing to the measurement have occurred.

7.6.5.17 measurement_unit_type

Type: IDLEnum
Package: Supplementary_Measurement
The units used to quantify the measurement values and accuracies

Table 7.179 - Attributes of IDLEnum measurement_unit_type

Attribute	Notes
«enum» **DIMENSIONLESS**	There are no units as the quantity is a dimensionless value
«enum» **DECIBEL**	units are in decibels to measure amplitudes
«enum» **DECIBELS_PER_SECOND**	units are in decibels per second to measure change in amplitude with time.
«enum» **HERTZ**	units are in Hertz to measure frequencies
«enum» **HERTZ_PER_SECOND**	Units are in Hertz per second to measure change in frequency with time
«enum» **METER**	Units are meters
«enum» **METERS_PER_SECOND**	Units are in meters per second to measure speeds
«enum» **OTHER_UNIT**	Another unit is used to quantify the measurements and accuracies
«enum» **RADIAN**	Units are in radians to measure angles
«enum» **RADIANS_PER_SECOND**	Units are in radians per seconds to measure the change in angles with time
«enum» **SECOND**	Units are in seconds to measure time or intervals.
«enum» **SECONDS_PER_SECOND**	Units are in seconds per second to measure the change in regular intervals over time.

7.6.5.18 measurement_variation_kind_type

Type: Class
Package: Supplementary_Measurement
A qualitative description of change in a parameter measurement value over time within a distribution. The characteristic quantitative values of the variation pattern can be represented by other related measurement parameters.

Table 7.180 - Attributes of Class measurement_variation_kind_type

Attribute	Notes
«enum» **DECREASING**	The value is decreasing monotonically; a minimum value is not yet determined
«enum» **DECREASING_ASYMPTOTICALLY**	The value is decreasing monotonically towards an asymptotic minimum value
«enum» **DECREASING_WITH_WRAP**	The value decreases monotonically until it reaches a minimum value at which point it wraps or resets to a maximum value.

Attribute	Notes
«enum» **DRIFT_NOT_SPECIFIED**	The drift behavior is not specified
«enum» **INCREASING**	The is increasing monotonically; a maximum value is not yet determined
«enum» **INCREASING_ASYMPTOTICALLY**	The is increasing monotonically towards a maximum asymptotic value.
«enum» **INCREASING_WITH_WRAP**	The value increases monotonically until it reaches a maximum value at which point it wraps or resets to a minimum value.
«enum» **LINEAR_ALTERNATING**	The value alternately increases monotonically until it reaches a maximum value and decreases monotonically until it reaches a minimum value.
«enum» **RANDOM_VARIATION**	The change in value over time is considered to be random
«enum» **SINE_WAVE_VARIATION**	The value of the measurement parameter value is sinusoidal over time.
«enum» **SQUARE_WAVE_VARIATION**	The value of the measurement parameter alternates discontinuously between minimum and maximum values over time.
«enum» **UNKNOWN_VARIATION**	The drift pattern is unknown to (not recognized by) the sensor.

7.6.5.19 modulation_type

Type: IDLStruct
Package: Supplementary_Measurement
The representation of parameter measurement values that are distributed within a bounded interval

Table 7.181 - Attributes of IDLStruct modulation_type

Attribute	Notes
kind measurement_variation_kind_type	The qualitative measure of the kind of modulation detected.

Table 7.182 - Relations of IDLStruct modulation_type

Connector	Notes
Aggregation: **modulation_amplitude** scalar_measurement_type [0..1]	The amplitude of the modulation of the parameter being measured.
Aggregation: **mean_value** scalar_measurement_type	The mean value of the measurement that is subject to modulation.
Aggregation: **modulation_frequency** scalar_measurement_type [0..1]	The frequency of the modulation of the parameter being measured.

7.6.5.20 multi_modal_measurement_type

Type: IDLStruct

Package: Supplementary_Measurement
The representation of parameter measurement values that have a multi-modal distribution

Table 7.183 - Relations of IDLStruct multi_modal_measurement_type

Connector	Notes
Aggregation: **mode** distribution_mode_type [1..*]	A mode within a multi-mode distribution for the parameter measurement

7.6.5.21 normal_measurement_type

Type: IDLStruct
Package: Supplementary_Measurement
The representation of a measurement parameter that is normally distributed

7.6.5.22 parameter_distribution_type

Type: IDLStruct
Package: Supplementary_Measurement
A representation of the statistical distribution of a parameter.

7.6.5.23 parameter_id_type

Type: Class
Package: Supplementary_Measurement
The unique identifier for a measurement parameter.

7.6.5.24 pdf_measurement_type

Type: IDLStruct
Package: Supplementary_Measurement
The values of the parameter measurement are distributed according to a probability density function.

Table 7.184 - Relations of IDLStruct pdf_measurement_type

Connector	Notes
Aggregation: **standard_deviation** single_measurement_type [0..1]	The standard deviation of values from the probability density function.
Aggregation: **mean** single_measurement_type	The mean (expected) value of the probability density function.

7.6.5.25 pdf_name_type

Type: Class
Package: Supplementary_Measurement
The name of a sensor defined probability density function.
ElementTag: Length = 20

7.6.5.26 plot_measurement_parameter_set_type

Type: IDLStruct
Package: Supplementary_Measurement

A set of the measurement parameters relating to a sensor track. Subsystems form measurement parameters into sets for efficient information transfer to the CMS. A subsystem may chose the number and composition of these sets. A subsystem may place all measurements into a single set per track, create multiple sets or create no sets and report measurement parameters individually instead.
For a particular sensor track, measurement parameter names shall be unique across all measurement parameter set instances - i.e. sets shall be non-overlapping.

Table 7.185 - Attributes of IDLStruct plot_measurement_parameter_set_type

Attribute	Notes
«key» **name** measurement_parameter_set_name_type	The name of the set of parameters

Table 7.186 - Relations of IDLStruct plot_measurement_parameter_set_type

Connector	Notes
Aggregation: **parameter** measurement_parameter_type [0..*]	The set of measurement parameters associated with the plot
Aggregation: **plot** sensor_plot_type	The plot associated with the set of parameter measurements

7.6.5.27 poisson_measurement_type

Type: IDLStruct
Package: Supplementary_Measurement
The parameter measurement follows a Poisson distribution

7.6.5.28 qualitative_measurement_type

Type: Class
Package: Supplementary_Measurement
This describes a qualitative measure

Table 7.187 - Attributes of Class qualitative_measurement_type

Attribute	Notes
descriptor unsigned short	The descriptor for the qualitative measurement

7.6.5.29 sample_range_type

Type: IDLStruct
Package: Supplementary_Measurement
The inclusive range of samples sensed that contribute to the measurement value

Table 7.188 - Attributes of IDLStruct sample_range_type

Attribute	Notes
min_value double	The minimum value of a sample for the measurement
max_value double	The maximum value of a sample for the measurement

7.6.5.30 scalar_measurement_type

Type: Class
Package: Supplementary_Measurement
This class represents individual scalar measurements of parameter values.

Table 7.189 - Attributes of Class scalar_measurement_type

Attribute	Notes
accuracy double [0..1]	The accuracy of the measurement value (one standard deviation)
value double	The value of the parameter measurement

7.6.5.31 sensor_defined_pdf_measurement_type

Type: IDLStruct
Package: Supplementary_Measurement
The representation of a measurement of generalised probability density function whose definition can be instantiated by a sensor for extensibility.

Table 7.190 - Attributes of IDLStruct sensor_defined_pdf_measurement_type

Attribute	Notes
name pdf_name_type	The name of the probability density function

Table 7.191 - Relations of IDLStruct sensor_defined_pdf_measurement_type

Connector	Notes
Aggregation: **parameter** distribution_parameter_measurement_type	The list of additional parameters required to describe the sensor defined probability density function

7.6.5.32 sequence_name_type

Type: Class
Package: Supplementary_Measurement
To name a sequence
ElementTag: Length = 32

7.6.5.33 single_measurement_type

Type: IDLStruct
Package: Supplementary_Measurement
A single discrete representation of a parameter measurement value.

Table 7.192 - Attributes of IDLStruct single_measurement_type

Attribute	Notes
confidence confidence_type [0..1]	The confidence in the parameter measurement value; this is the probability that the value and accuracy represent the true distribution of the physical effect they are labelled as measuring in the real world.

7.6.5.34 track_measurement_parameter_set_type

Type: IDLStruct
Package: Supplementary_Measurement

A set of the measurement parameters relating to a sensor track. Subsystems form measurement parameters into sets for efficient information transfer to the CMS. A subsystem may chose the number and composition of these sets. A subsystem may place all measurements into a single set per track, create multiple sets or create no sets and report measurement parameters individually instead.

For a particular sensor track, measurement parameter names shall be unique across all measurement parameter set instances - i.e. sets shall be non-overlapping.

Table 7.193 - Attributes of IDLStruct track_measurement_parameter_set_type

Attribute	Notes
«key» **name** measurement_parameter_set_name_type	The name of the set of parameters

Table 7.194 - Relations of IDLStruct track_measurement_parameter_set_type

Connector	Notes
Association: **sensor_track** sensor_track_type reference	
Aggregation: **parameter** measurement_parameter_type [0..*]	The parameter measurement for the element

7.6.5.35 track_measurement_parameter_type

Type: IDLStruct
Package: Supplementary_Measurement

To represent parameter measurements for a sensor track reported individually

Table 7.195 - Relations of IDLStruct track_measurement_parameter_type

Connector	Notes
Association: **sensor_track** sensor_track_type reference	The sensor track to which the measurement parameter relates.
Aggregation: **parameter** measurement_parameter_type	The individual parameter

7.6.5.36 vector_measurement_type

Type: Class
Package: Supplementary_Measurement

This class represents individual vector measurements of parameter values.

Table 7.196 - Attributes of Class vector_measurement_type

Attribute	Notes
covariance double [0..*]	The covariance between the elements of the vector value in a 1-dimensional representation of the triangular matrix. The i,j element (i >= j) of a covariance matrix for a vector of size N is at position sum(k=0..j-1, N - k) + (i - j). The covariance is zero length if not specified. AttributeTag: Length = 21
value double [1..*]	The vector values AttributeTag: Length = 6

7.6.6 Plot_Reporting

Parent Package: Sensor_Domain

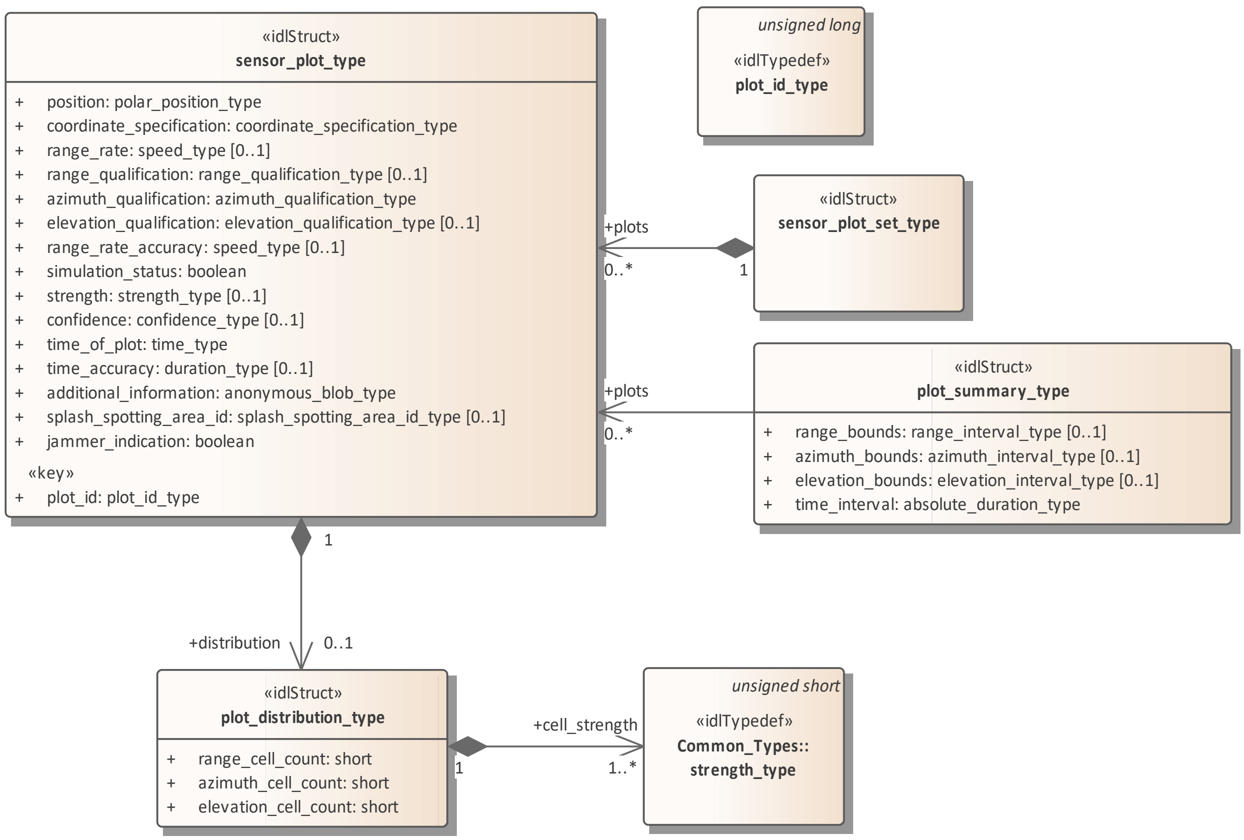


Figure 7.78 Domain Model - Sensor Plot (Class diagram)

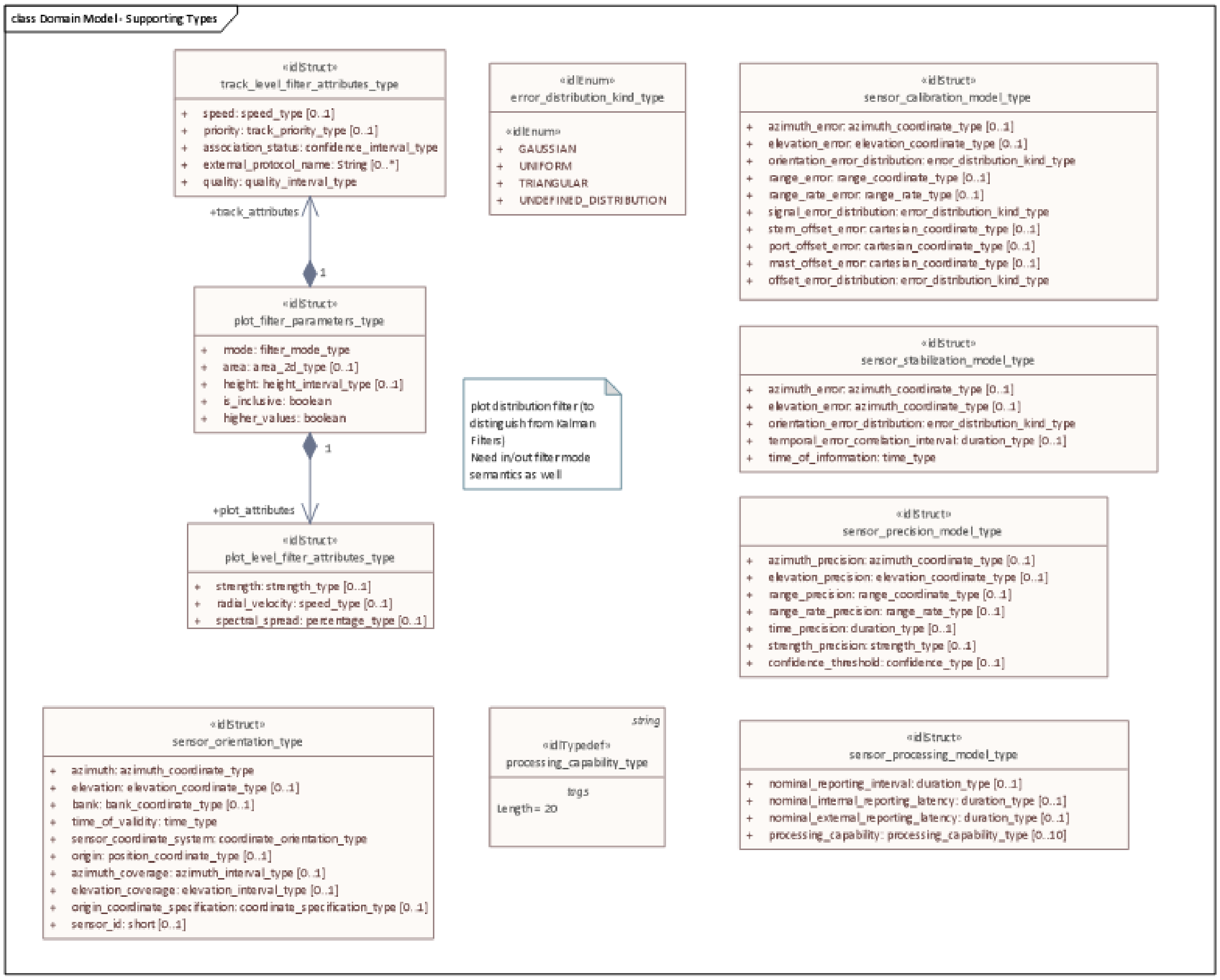


Figure 7.79 Domain Model - Supporting Types (Class diagram)

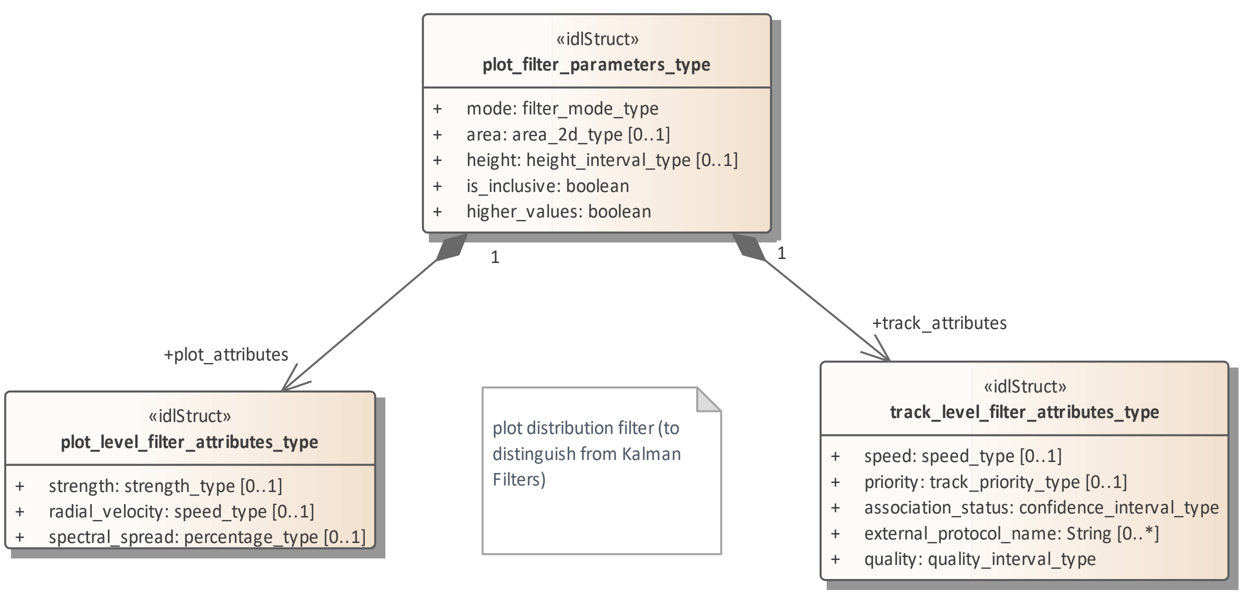


Figure 7.80 Domain Model - Filtering (Class diagram)

7.6.6.1 error_distribution_kind_type

Type: IDLEnum
Package: Plot_Reporting
The class models the kinds of error distribution supported for sensors accuracy.

Table 7.197 - Attributes of IDLEnum error_distribution_kind_type

Attribute	Notes
«idlEnum» **GAUSSIAN**	The error has a Gaussian distribution with zero mean and stated standard deviation.
«idlEnum» **UNIFORM**	The error has a uniform distribution with zero mean and stated standard deviation.
«idlEnum» **TRIANGULAR**	The error has a symmetric triangular distribution with zero mean and stated standard deviation.
«idlEnum» **UNDEFINED_DISTRIBUTION**	The distribution of the error is not defined

7.6.6.2 plot_distribution_type

Type: IDLStruct
Package: Plot_Reporting
This class encapsulates the strength of the plot over a grid of higher-resolution range cells. The spatial extent of the distribution in range, azimuth and elevation is defined by the spread attribute of the relevant qualification attribute. The sequence of strength values represents a 3D array over range (inner iteration), azimuth and elevation (outer iteration). The cell at logical index i,j,k for range, azimuth and elevation respectively is at index:
i + j * range_cell_count + k * azimuth_cell_count * range_cell_count
The cell indexed zero represents the lowest values of range, azimuth or elevation and cells are equally spaced such that the cell at logical index i,j,k for range, azimuth and elevation respectively corresponds to:
min_range + i * range_spread / range_cell_count < range < min_range + (i + 1) * range_spread / range_cell_count; min_range = range - range_spread / 2
min_azimuth + i * azimuth_spread /azimuth_cell_count < azimuth < min_azimuth + (i + 1) * azimuth_spread / azimuth_cell_count; min_azimuth = azimuth - azimuth_spread / 2
min_elevation + i * elevation_spread / elevation_cell_count < elevation < min_elevation + (i + 1) * elevation_spread / elevation_cell_count; min_elevation = elevation - elevation_spread / 2
where range_spread = range_qualification.spread for the plot and range, azimuth and elevation are the mean coordinates reported for the plot.
If no spread is defined for a plot in the range, azimuth or elevation qualification then it is only valid for there to be a count of one defined for that dimension.

Table 7.198 - Attributes of IDLStruct plot_distribution_type

Attribute	Notes
range_cell_count short	The number of cells in the range dimension. Only one cell is valid if no range spread is defined.
azimuth_cell_count short	The number of cells in the azimuth dimension. Only one cell is valid if no azimuth spread is defined.

Attribute	Notes
elevation_cell_count short	The number of cells in the elevation dimension. Only one cell is valid if no elevation spread is defined.

Table 7.199 - Relations of IDLStruct plot_distribution_type

Connector	Notes
Aggregation: **cell_strength** strength_type [1..*]	The strength of plot signal in a higher resolution spatial cell

7.6.6.3 plot_filter_parameters_type

Type: IDLStruct
Package: Plot_Reporting
The criteria that must all be met for a plot to pass the filter. The filter attributes are applied with and-wise logic. For or-wise logic define multiple filter objects.

Table 7.200 - Attributes of IDLStruct plot_filter_parameters_type

Attribute	Notes
mode filter_mode_type	The mode in which the plots are filtered.
area area_2d_type [0..1]	An area which is optionally part of the filter.
height height_interval_type [0..1]	The height values that are optionally part of the filter.
is_inclusive boolean	If true, tracks that pass the filter are included in transmission and/or reception dependent upon the mode attribute. Otherwise, they are excluded.
higher_values boolean	When true, for real-valued criteria, plots meet the criteria of the filter if the plot's value is equal to or higher than the corresponding filter criteria value. Otherwise, the criteria is met if equal to or lower.

Table 7.201 - Relations of IDLStruct plot_filter_parameters_type

Connector	Notes
Aggregation: **plot_attributes** plot_level_filter_attributes_type	The filter criteria for plots that relate to characteristics of the plots themsleves.
Aggregation: **track_attributes** track_level_filter_attributes_type	The filter criteria for plots that relate to the tracks to which the plots contribute.

7.6.6.4 plot_id_type

Type: Class
Package: Plot_Reporting
Identifier for a plot, unique within a given sensor. Such plot ids should not be reused between sensor subsystem restarts.

7.6.6.5 plot_level_filter_attributes_type

Type: IDLStruct
Package: Plot_Reporting
The plot-level criteria which the plot attributes must pass in order to pass the filter.

Table 7.202 - Attributes of IDLStruct plot_level_filter_attributes_type

Attribute	Notes
strength strength_type [0..1]	The plot strength criterion for the filter.
radial_velocity speed_type [0..1]	The plot radial velocity criterion for the filter.
spectral_spread percentage_type [0..1]	The plot spectral spread criterion for the filter.

7.6.6.6 plot_summary_type

Type: IDLStruct
Package: Plot_Reporting
The class provides a summary of plots found by the sensor in a region of the environment. Objects expected to be in the region for which there is no corresponding plot have not been detected by sensor, therefore missed measurements can be identified from this information.

Table 7.203 - Attributes of IDLStruct plot_summary_type

Attribute	Notes
range_bounds range_interval_type [0..1]	The bounds of the region being summarized in range. If omitted the region in unbounded in range.
azimuth_bounds azimuth_interval_type [0..1]	The bounds of the region being summarized in azimuth. If omitted the region in unbounded in azimuth.
elevation_bounds elevation_interval_type [0..1]	The bounds of the region being summarized in elevation. If omitted the region in unbounded in elevation.
time_interval absolute_duration_type	The period of time during which the sensor sensed the region

Table 7.204 - Relations of IDLStruct plot_summary_type

Connector	Notes
Association: **plots** sensor_plot_type reference [0..*]	The set of plots found in the region

7.6.6.7 processing_capability_type

Type: IDLTypeDef
Package: Plot_Reporting
Encapsulates a category of sensor processing capability. The set of known categories of sensor processing is defined on an implementation specific basis.
ElementTag: Length = 20

7.6.6.8 sensor_calibration_model_type

Type: IDLStruct
Package: Plot_Reporting
This class models the residual global sensor error estimate after calibration

Table 7.205 - Attributes of IDLStruct sensor_calibration_model_type

Attribute	Notes
azimuth_error azimuth_coordinate_type [0..1]	Residual error in azimuth to one standard deviation
elevation_error elevation_coordinate_type [0..1]	Residual error in elevation to one standard deviation
orientation_error_distribution error_distribution_kind_type	The statistical distribution of the azimuth and elevation errors.
range_error range_coordinate_type [0..1]	Residual error in range to one standard deviation
range_rate_error range_rate_type [0..1]	Residual error in range rate to one standard deviation
signal_error_distribution error_distribution_kind_type	The statistical distribution of the range and range rate errors.
stern_offset_error cartesian_coordinate_type [0..1]	Residual error in offset of the sensor bore-sight origin on the stern-bow axis to one standard deviation
port_offset_error cartesian_coordinate_type [0..1]	Residual error in offset of the sensor bore-sight origin on the port-starboard axis to one standard deviation
mast_offset_error cartesian_coordinate_type [0..1]	Residual error in offset of the sensor bore-sight origin on the mast-keel axis to one standard deviation
offset_error_distribution error_distribution_kind_type	The statistical distribution of the sensor origin offset errors.

7.6.6.9 sensor_plot_set_type

Type: Class
Package: Plot_Reporting
Set of one or more sensor plots.

Table 7.206 - Relations of Class sensor_plot_set_type

Connector	Notes
Aggregation: **plots** sensor_plot_type [0..*]	The plots in the sensor plot set

7.6.6.10 sensor_plot_type

Type: Class
Package: Plot_Reporting
One plot from a sensor, a plot being a measurement estimate of an object's state in terms of location, motion and optionally size at a particular moment in time.

Table 7.207 - Attributes of Class sensor_plot_type

Attribute	Notes
«key» **plot_id** plot_id_type	A unique identifier for the plot within the scope of the sensor. This attribute is mandatory so that a sensor's plot summary can refer to published plots. AttributeTag: Issue =

Attribute	Notes
position polar_position_type	The position of the plot in polar coordinates (measurements assumed to be relative to a particular sensor position). This is the mean, central position. Note the qualification attributes, which give information on accuracy and spread estimates. AttributeTag: Issue =
coordinate_specification coordinate_specification_type	This attribute defines the characteristics of the coordinate system used
range_rate speed_type [0..1]	The speed of the object detected along the line-of-sight of the sensor; positive values for an object receding from the sensor. Doppler processing can derive this value.
range_qualification range_qualification_type [0..1]	A measure of the spread and accuracy of the plot in range. This is optional as not all sensors measure range.
azimuth_qualification azimuth_qualification_type	A measure of the spread and accuracy of the plot in azimuth.
elevation_qualification elevation_qualification_type [0..1]	A measure of the spread and accuracy of the plot in elevation. This is optional as not all sensors measure elevation.
range_rate_accuracy speed_type [0..1]	A measure of the accuracy of the plot in range rate equal to one standard deviation of uncertainty. This is optional as not all sensors measure range rate. Note that for rigid objects a continuous spread in the measurement of range rate is not expected. AttributeTag: Issue =
simulation_status boolean	If true, the plot is simulated. See also simulation support services within this standard.
strength strength_type [0..1]	The signal strength of the plot. This attribute is optional as not all sensors measure a quantity which has equivalence to strength.
confidence confidence_type [0..1]	The probability that the plot represents a true object of interest as opposed to clutter, noise or other false objects. AttributeTag: Issue =
time_of_plot time_type	The time at which the plot was measured.
time_accuracy duration_type [0..1]	A measure of the accuracy of the time-stamping of the plot's time_of_plot attribute. This is equal to one standard deviation of uncertainty. This is optional as not all sensors estimate time accuracy and for some applications the uncertainty is negligible. AttributeTag: Issue =

Attribute	Notes
additional_information anonymous_blob_type	Potentially classified information about the plot, which may be used in a system specific way to distribute information about a plot to other subsystems. Further information about this attribute, including layout semantics is outside of the scope of this interface standard.
splash_spotting_area_id splash_spotting_area_id_type [0..1]	Indicates which splash spotting area the plot refers to - if any - hence it is optional.
jammer_indication boolean	Indication whether or not a plot is from a source of jamming.

Table 7.208 - Relations of Class sensor_plot_type

Connector	Notes
Aggregation: **distribution** plot_distribution_type [0..1]	The optional spatial distribution of plot strength in higher resolution. It is only valid to specify a distribution when a spread had been specified in one of the plot's qualification attributes.

7.6.6.11 sensor_precision_model_type

Type: IDLStruct
Package: Plot_Reporting
This class models the precision of the sensor - i.e. the smallest changes in measurement quantities that it is capable of distinguishing.

Table 7.209 - Attributes of IDLStruct sensor_precision_model_type

Attribute	Notes
azimuth_precision azimuth_coordinate_type [0..1]	The precision with which the sensor is capable of measuring azimuth.
elevation_precision elevation_coordinate_type [0..1]	The precision with which the sensor is capable of measuring elevation.
range_precision range_coordinate_type [0..1]	The precision with which the sensor is capable of measuring range.
range_rate_precision range_rate_type [0..1]	The precision with which the sensor is capable of measuring range rate.
time_precision duration_type [0..1]	The precision with which the sensor is capable of measuring time.
strength_precision strength_type [0..1]	The precision with which the sensor is capable of measuring signal strength.
confidence_threshold confidence_type [0..1]	The threshold probability for signal strength to identify a plot AttributeTag: Issue =

7.6.6.12 sensor_processing_model_type

Type: IDLStruct
Package: Plot_Reporting
This class encapsulates sensor processing parameters to promote the accurate statistical processing of its measurements

Table 7.210 - Attributes of IDLStruct sensor_processing_model_type

Attribute	Notes
nominal_reporting_interval duration_type [0..1]	The nominal period between successive measurements on the same object for the sensor. AttributeTag: Issue =
nominal_internal_reporting_latency duration_type [0..1]	The nominal period between the sensor's measurement of an object and its reporting of the object to systems on the same platform. AttributeTag: Issue =
nominal_external_reporting_latency duration_type [0..1]	The nominal period between the sensor's measurement of an object and its reporting of the object to a system on any other connected platform. AttributeTag: Issue =
processing_capability processing_capability_type [0..10]	The set of processing capabilities of which the sensor is capable. These capabilities have quality implications for the sensors plot measurement information.

7.6.6.13 sensor_stabilization_model_type

Type: IDLStruct
Package: Plot_Reporting
This class models the sensor error estimate due to sensor stabilization. These are errors that are in addition to any calibration errors,

Table 7.211 - Attributes of IDLStruct sensor_stabilization_model_type

Attribute	Notes
azimuth_error azimuth_coordinate_type [0..1]	Current error in azimuth due to stabilization to one standard deviation
elevation_error azimuth_coordinate_type [0..1]	Current error in elevation due to stabilization to one standard deviation
orientation_error_distribution error_distribution_kind_type	The statistical distribution of the azimuth and elevation stabilization errors.
temporal_error_correlation_interval duration_type [0..1]	The time period, centered on the time of information, such that the coefficient of correlation between stabilization errors at either end is expected to be 0.5. Measurements made within this interval are expected to have stabilization errors that are strongly correlated with each other.

Attribute	Notes
time_of_information time_type	The time for which the stabilization error estimates are valid.

7.6.6.14 track_level_filter_attributes_type

Type: IDLStruct
Package: Plot_Reporting
The track-level criteria that must be met for the plot to pass the filter. These are criteria applied with respect to any track to which the plot is contributing.

Table 7.212 - Attributes of IDLStruct track_level_filter_attributes_type

Attribute	Notes
speed speed_type [0..1]	A speed criterion. A track to which the plot contributes must have an absolute speed greater than this.
priority track_priority_type [0..1]	A priority criterion. A track to which the plot contributes must have a priority greater than or equal to this.
association_status confidence_interval_type	A criterion relating to whether the plot is contributing to a track. The cumulative probability of being associated to a track must be within the interval defined to pass the filter. If no filtering is required then an interval including all confidence values is defined.
external_protocol_name String [0..*]	Filter on the basis of the external protocols on which the track is known. AttributeTag: Issue =
quality quality_interval_type	A criterion relating to the quality of a track. A track to which the plot contributes must have a track quality within the interval defined to pass the filter. If no filtering is required then an interval including all quality values is defined.

7.6.6.15 sensor_orientation_type

Type: Class
Package: Plot_Reporting
This class describes the orientation of the sensor at a particular moment in time. This is useful for plot processing functionality such as track extraction as it allows instantaneous coverage of the sensor to be estimated.

Table 7.213 - Attributes of Class sensor_orientation_type

Attribute	Notes
azimuth azimuth_coordinate_type	The (azimuth) direction of the head of the sensor (e.g. antenna, lens or hydro-phone)

Attribute	Notes
elevation elevation_coordinate_type [0..1]	The (elevation) direction of the head of the sensor (e.g. antenna, lens or hydro-phone). If not supplied either horizontal is assumed or a constant angle is defined through the Manage_Subsystem_Parameters use case.
bank bank_coordinate_type [0..1]	The (bank) direction of the head of the sensor (e.g. antenna, lens or hydro-phone). If not supplied either no bank is assumed or a constant angle is defined through the Manage_Subsystem_Parameters use case.
time_of_validity time_type	The time for which is sensor orientation is valid
sensor_coordinate_system coordinate_orientation_type	This attribute defines the interpretation of azimuth and elevation. Valid enumerates are: NORTH_HORIZONTAL, NORTH_DOWN, STERN_KEEL, STERN_DECK_LEVEL
origin position_coordinate_type [0..1]	The position of the origin of the head of the sensor.
azimuth_coverage azimuth_interval_type [0..1]	The instantaneous extent of the coverage of the sensor in azimuth with respect the origin of its head.
elevation_coverage elevation_interval_type [0..1]	The instantaneous extent of the coverage of the sensor in elevation with respect the origin of its head. It is only valid to set this when the elevation attribute is also specified.
origin_coordinate_specification coordinate_specification_type [0..1]	This attribute defines the characteristics of the coordinate system used to define the origin. It is only valid to set this when origin attribute is also specified.
sensor_id short [0..1]	For a multi sensor radar system, this attribute defines the sensor_id for which this orientation applies.

7.6.7 Sensor_Control

Parent Package: Sensor_Domain
This package contains structs and type defs for managing frequency usage, transmission sectors, emission control, and test target scenarios.

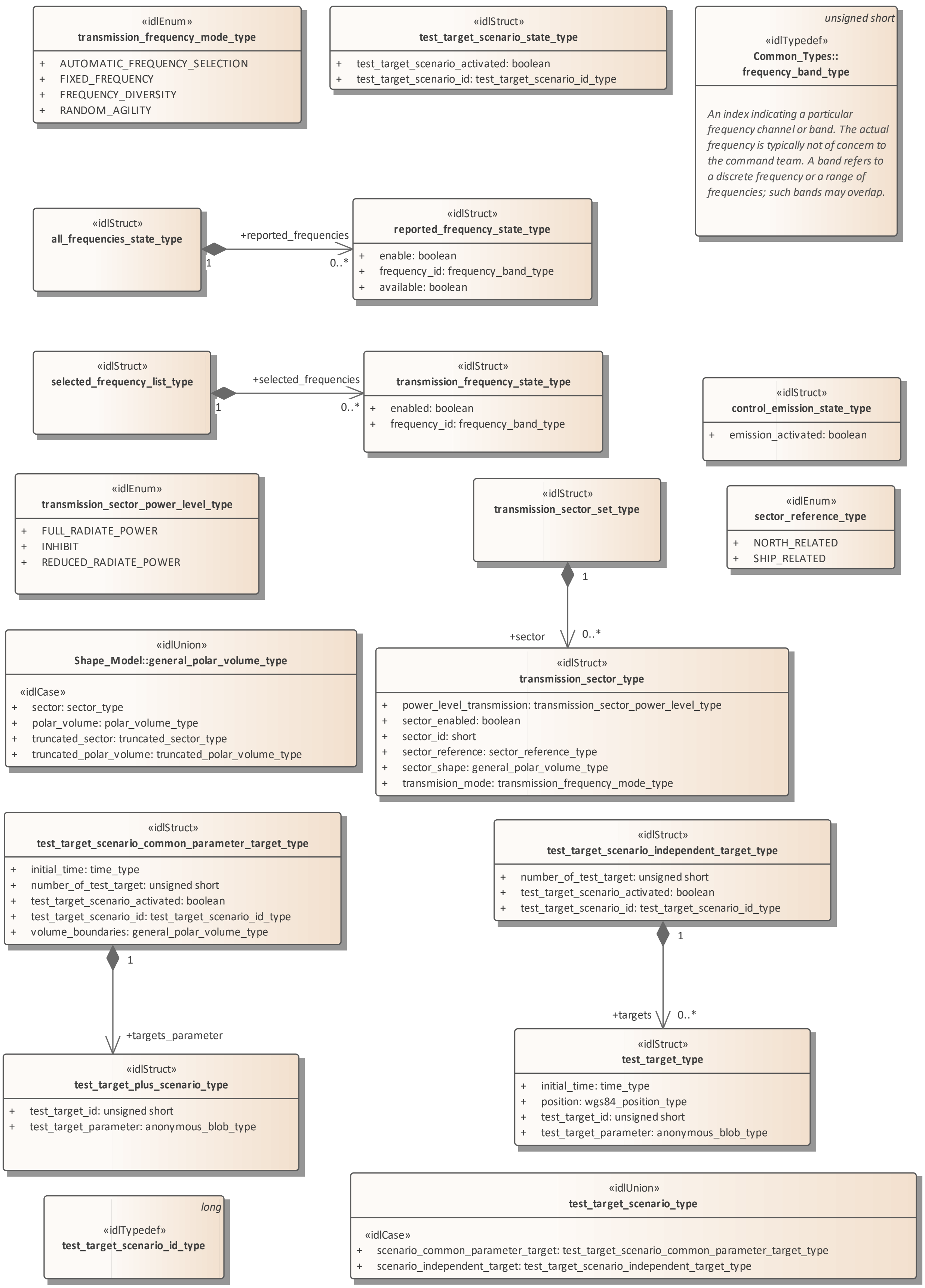

«idlEnum»
transmission_frequency_mode_type
+ AUTOMATIC_FREQUENCY_SELECTION
+ FIXED_FREQUENCY
+ FREQUENCY_DIVERSITY
+ RANDOM_AGILITY
«idlStruct»
test_target_scenario_state_type
+ test_target_scenario_activated: boolean
+ test_target_scenario_id: test_target_scenario_id_type
unsigned short
«idlTypedef»
Common_Types::
frequency_band_type
An index indicating a particular frequency channel or band. The actual frequency is typically not of concern to the command team. A band refers to a discrete frequency or a range of frequencies; such bands may overlap.
«idlStruct»
all_frequencies_state_type
+reported_frequencies
1
0..*
«idlStruct»
reported_frequency_state_type
+ enable: boolean
+ frequency_id: frequency_band_type
+ available: boolean
«idlStruct»
selected_frequency_list_type
+selected_frequencies
1
0..*
«idlStruct»
transmission_frequency_state_type
+ enabled: boolean
+ frequency_id: frequency_band_type
«idlStruct»
control_emission_state_type
+ emission_activated: boolean
«idlEnum»
transmission_sector_power_level_type
+ FULL_RADIATE_POWER
+ INHIBIT
+ REDUCED_RADIATE_POWER
«idlStruct»
transmission_sector_set_type
«idlEnum»
sector_reference_type
+ NORTH_RELATED
+ SHIP_RELATED
1
+sector
0..*
«idlUnion»
Shape_Model::general_polar_volume_type
«idlCase»
+ sector: sector_type
+ polar_volume: polar_volume_type
+ truncated_sector: truncated_sector_type
+ truncated_polar_volume: truncated_polar_volume_type
«idlStruct»
transmission_sector_type
+ power_level_transmission: transmission_sector_power_level_type
+ sector_enabled: boolean
+ sector_id: short
+ sector_reference: sector_reference_type
+ sector_shape: general_polar_volume_type
+ transmision_mode: transmission_frequency_mode_type
«idlStruct»
test_target_scenario_common_parameter_target_type
+ initial_time: time_type
+ number_of_test_target: unsigned short
+ test_target_scenario_activated: boolean
+ test_target_scenario_id: test_target_scenario_id_type
+ volume_boundaries: general_polar_volume_type
«idlStruct»
test_target_scenario_independent_target_type
+ number_of_test_target: unsigned short
+ test_target_scenario_activated: boolean
+ test_target_scenario_id: test_target_scenario_id_type
1
+targets_parameter
1
+targets
0..*
«idlStruct»
test_target_plus_scenario_type
+ test_target_id: unsigned short
+ test_target_parameter: anonymous_blob_type
«idlStruct»
test_target_type
+ initial_time: time_type
+ position: wgs84_position_type
+ test_target_id: unsigned short
+ test_target_parameter: anonymous_blob_type
long
«idlTypedef»
test_target_scenario_id_type
«idlUnion»
test_target_scenario_type
«idlCase»
+ scenario_common_parameter_target: test_target_scenario_common_parameter_target_type
+ scenario_independent_target: test_target_scenario_independent_target_type

Figure 7.81 Domain Model (Class diagram)

7.6.7.1 selected_frequency_list_type

Type: IDLStruct
Package: Sensor_Control
This struct contains zero to many frequencies which may be enabled/disabled by the CMS

Table 7.214 - Relations of IDLStruct selected_frequency_list_type

Connector	Notes
Aggregation: **selected_frequencies** transmission_frequency_state_type [0..*]	

7.6.7.2 transmission_frequency_state_type

Type: Class
Package: Sensor_Control
State of frequency transmission

Table 7.215 - Attributes of Class transmission_frequency_state_type

Attribute	Notes
enabled boolean	Indicates whether the CMS is enabling or disabling a transmission frequency.
frequency_id frequency_band_type	A unique identifier for the transmission frequency.

7.6.7.3 all_frequencies_state_type

Type: Class
Package: Sensor_Control
This struct contains zero to many "available" or "not available" frequencies which may be enabled/disabled by the CMS

Table 7.216 - Relations of Class all_frequencies_state_type

Connector	Notes
Aggregation: **reported_frequencies** reported_frequency_state_type [0..*]	

7.6.7.4 reported_frequency_state_type

Type: IDLStruct
Package: Sensor_Control
reported frequency state

Table 7.217 - Attributes of IDLStruct reported_frequency_state_type

Attribute	Notes
enable boolean	Indicates whether the CMS is enabling or disabling a transmission frequency.
frequency_id frequency_band_type	A unique identifier for the transmission frequency.

Attribute	Notes
available boolean	Indicates whether a transmission frequency is available or not available.

7.6.7.5 transmission_frequency_mode_type

Type: Class
Package: Sensor_Control
The mode

Table 7.218 - Attributes of Class transmission_frequency_mode_type

Attribute	Notes
AUTOMATIC_FREQUENCY_SELECTION	The sensor always uses the same pre-selected frequency
FIXED_FREQUENCY	At each transmission sensor selects the frequency to be used inside a pre-selected subset of frequencies
FREQUENCY_DIVERSITY	At each transmission sensor selects the frequency to be used among the least jammed frequencies
RANDOM_AGILITY	At each transmission sensor random selects the frequency to be used.

7.6.7.6 transmission_sector_set_type

Type: IDLStruct
Package: Sensor_Control
This struct contains zero to many transmission sectors which must be set/reset by the CMS.

Table 7.219 - Relations of IDLStruct transmission_sector_set_type

Connector	Notes
Aggregation: **sector** transmission_sector_type [0..*]	

7.6.7.7 transmission_sector_type

Type: Class
Package: Sensor_Control
Sector for transmission

Table 7.220 - Attributes of Class transmission_sector_type

Attribute	Notes
power_level_transmission transmission_sector_power_level_type	Indicates the transmission power level of the sector.
sector_enabled boolean	Indicates whether the CMS is enabling or disabling a transmission sector.
sector_id short	A unique identifier for the transmission sector.
sector_reference sector_reference_type	This indicates the reference system of the transmission sector.

Attribute	Notes
sector_shape general_polar_volume_type	Note that the azimuth dimension of the sector shape (polar volume) applies to the horizon plane (i.e. elevation=0)
transmision_mode transmission_frequency_mode_type	Indicates the transmission mode used within the sector

7.6.7.8 transmission_sector_power_level_type

Type: Class
Package: Sensor_Control
This enumeration allows specification of a CMS commanded power level for a sector.

Table 7.221 - Attributes of Class transmission_sector_power_level_type

Attribute	Notes
FULL_RADIATE_POWER	radiate with full power
INHIBIT	inhibit transmission
REDUCED_RADIATE_POWER	radiate with reduced power

7.6.7.9 sector_reference_type

Type: IDLEnum
Package: Sensor_Control
This enumeration specifies the sectors reference systems.

Table 7.222 - Attributes of IDLEnum sector_reference_type

Attribute	Notes
NORTH_RELATED	Indicates values referenced with respect to true North
SHIP_RELATED	Indicates values referenced with respect to ship's heading

7.6.7.10 control_emission_state_type

Type: Class
Package: Sensor_Control
Emission state

Table 7.223 - Attributes of Class control_emission_state_type

Attribute	Notes
emission_activated boolean	Indicates whether the CMS is enabling or disabling the sensor emission state.

7.6.7.11 test_target_scenario_type

Type: IDLUnion

Package: Sensor_Control
Scenario for test targets

Table 7.224 - Attributes of IDLUnion test_target_scenario_type

Attribute	Notes
«idlCase» **scenario_common_parameter_target** test_target_scenario_common_parameter_target_type	This case is used when a test target scenario is constituted by a number of targets distributed in a defined area/volume and having the same common parameters.
«idlCase» **scenario_independent_target** test_target_scenario_independent_target_type	This case is used when a test target scenario is constituted by a number of independent targets.

7.6.7.12 test_target_scenario_independent_target_type

Type: IDLStruct
Package: Sensor_Control
The scenario is defined by a number of independent targets, with each target having own characteristic parameters.

Table 7.225 - Attributes of IDLStruct test_target_scenario_independent_target_type

Attribute	Notes
number_of_test_target unsigned short	This is the number of the test targets composing the scenario.
test_target_scenario_activated boolean	Indicates whether the CMS is enabling or disabling the generation of a test target scenario.
test_target_scenario_id test_target_scenario_id_type	A unique identifier for the test target scenario.

Table 7.226 - Relations of IDLStruct test_target_scenario_independent_target_type

Connector	Notes
Aggregation: **targets** test_target_type [0..*]	

7.6.7.13 test_target_scenario_common_parameter_target_type

Type: IDLStruct
Package: Sensor_Control
The scenario is defined by a number of targets distributed in a defined area/volume and having the same common parameters.

Table 7.227 - Attributes of IDLStruct test_target_scenario_common_parameter_target_type

Attribute	Notes
initial_time time_type	This indicates the common initial time of the targets.
number_of_test_target unsigned short	This is the number of the test targets composing the scenario.
test_target_scenario_activated boolean	Indicates whether the CMS is enabling or disabling the generation of a test target scenario.

Attribute	Notes
test_target_scenario_id test_target_scenario_id_type	A unique identifier for the test target scenario.
volume_boundaries general_polar_volume_type	This indicates the area/volume boundaries where the test targets are distributed.

Table 7.228 - Relations of IDLStruct test_target_scenario_common_parameter_target_type

Connector	Notes
Aggregation: **targets_parameter** test_target_plus_scenario_type	

7.6.7.14 test_target_type

Type: IDLStruct
Package: Sensor_Control
Encapsulation of a test target (simulated target to enable technical testing of a sensor)

Table 7.229 - Attributes of IDLStruct test_target_type

Attribute	Notes
initial_time time_type	This attribute defines the relevant initial time.
position wgs84_position_type	This attribute defines the initial target position.
test_target_id unsigned short	A identifier for the test targets.
test_target_parameter anonymous_blob_type	This attribute defines: - the target motion type, with the relevant motion parameters - the target generation parameters, such as injection type (internal / external), attenuation law (constant / variable-with-range), doppler type (0 / PRF/2).

7.6.7.15 test_target_plus_scenario_type

Type: Class
Package: Sensor_Control
Test target with its scenario

Table 7.230 - Attributes of Class test_target_plus_scenario_type

Attribute	Notes
test_target_id unsigned short	A identifier for the test targets.
test_target_parameter anonymous_blob_type	This attribute defines: - the target motion type, with the relevant motion parameters - the target generation parameters, such as injection type (internal / external), attenuation law (constant / variable-with-range), doppler type (0 / PRF/2).

7.6.7.16 test_target_scenario_id_type

Type: IDLTypeDef
Package: Sensor_Control
This typedef is used to identify a specific test target scenario.

7.6.7.17 test_target_scenario_state_type

Type: Class
Package: Sensor_Control
scenario state

Table 7.231 - Attributes of Class test_target_scenario_state_type

Attribute	Notes
test_target_scenario_activated boolean	Indicates whether the CMS is enabling or disabling the execution of the test target scenario.
test_target_scenario_id test_target_scenario_id_type	A unique identifier for the test target scenario.

7.6.8 Sensor_Performance

Parent Package: Sensor_Domain

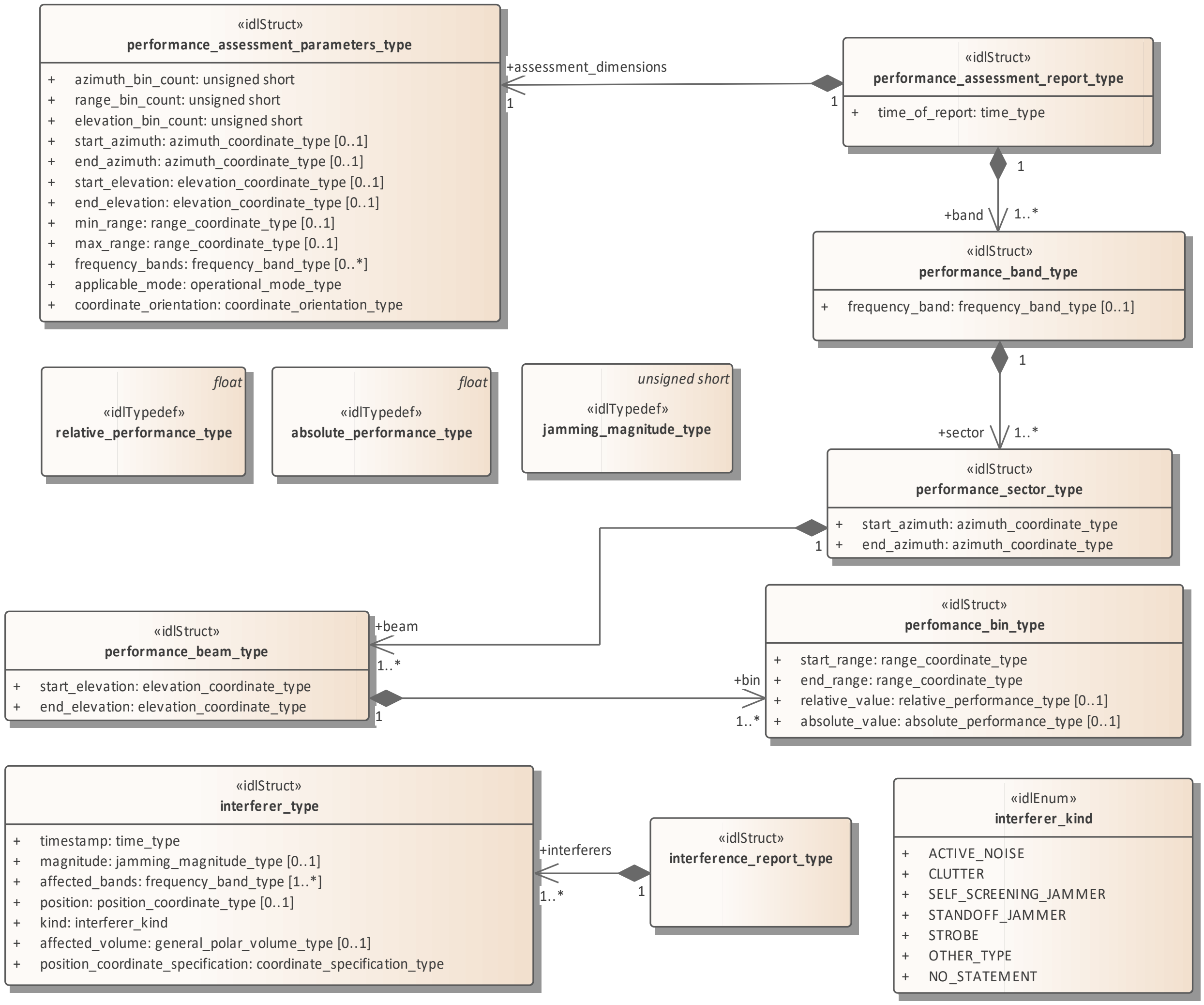


Figure 7.82 Domain Model (Class diagram)

7.6.8.1 absolute_performance_type

Type: IDLTypeDef
Package: Sensor_Performance
Defined as a signal excess in dB above noise floor for a nominal ideal target with 1m^2 tangential cross-section.

7.6.8.2 interference_report_type

Type: Class
Package: Sensor_Performance
Set of interferer objects in a report.

Table 7.232 - Relations of Class interference_report_type

Connector	Notes
Aggregation: **interferers** interferer_type [1..*]	The interference sources, which are described by the report.

7.6.8.3 interferer_kind

Type: Class
Package: Sensor_Performance
Enumeration of the types of interferers that are known about.

Table 7.233 - Attributes of Class interferer_kind

Attribute	Notes
ACTIVE_NOISE	Interference from active noise.
CLUTTER	Interference from clutter.
SELF_SCREENING_JAMMER	Interference from a jammer, which is self screening.
STANDOFF_JAMMER	Interference from a stand-off jammer
STROBE	Interference from a strobe jammer.
OTHER_TYPE	The interference source is of a different type to the other declared interference kinds
NO_STATEMENT	The interference source could not be classified by the sensor subsystem.

7.6.8.4 interferer_type

Type: Class
Package: Sensor_Performance
A single source of interference.

Table 7.234 - Attributes of Class interferer_type

Attribute	Notes
timestamp time_type	Time to which the performance report applies.
magnitude jamming_magnitude_type [0..1]	The Effective Radiated Power (ERP) of the source of interference. This is an optional attribute, which may not all sensors may be able to calculate.
affected_bands frequency_band_type [1..*]	A list of frequency bands which are effected by the source of interference.
position position_coordinate_type [0..1]	The source position of the interference. This is an optional attribute that not all sensors may be able to calculate.
kind interferer_kind	A classification of the interference source.
affected_volume general_polar_volume_type [0..1]	The volume in space, which the interference source is affecting. This is an optional attribute, which may not all sensors may be able to calculate.

Attribute	Notes
position_coordinate_specification coordinate_specification_type	Specifies the coordinate system used to define the interferer.

7.6.8.5 jamming_magnitude_type

Type: Class
Package: Sensor_Performance
Target strength (Effective Radiated Power - ERP) of a jammer. The precise semantics of this type are sensor subsystem specific, but a typical interpretation is as a signal to noise ratio in dB.

7.6.8.6 perfomance_bin_type

Type: IDLStruct
Package: Sensor_Performance
Value of performance in a volume of space. This is given as a signal excess in dB above noise floor for a nominal 0dB target strength. For a current performance report, this noise floor shall include clutter and jamming. These are not included in a nominal performance report.

Table 7.235 - Attributes of IDLStruct perfomance_bin_type

Attribute	Notes
start_range range_coordinate_type	The start of the bin in range.
end_range range_coordinate_type	The end of the bin in range.
relative_value relative_performance_type [0..1]	The assessed relative level of performance (comparable with other instances of the sensor or the same sensor in a different context). If no value present, there is no performance data available for this bin.
absolute_value absolute_performance_type [0..1]	The assessed absolute level of performance (comparable with other sensors). If no value present, there is no performance data available for this bin.

7.6.8.7 performance_assessment_parameters_type

Type: Class
Package: Sensor_Performance
A performance assessment request consists of an overall volume of interest and a specification of a number of 'bins' into which that volume is to be sub-divided. In response the sensor assess performance for each 'bin'.
The coordinate origin for the request is the SENSOR_REFERENCE_POINT as defined in coordinate_origin_type.

Table 7.236 - Attributes of Class performance_assessment_parameters_type

Attribute	Notes
azimuth_bin_count unsigned short	Number of azimuth bins that the CMS would like in the performance report. The subsystem should try to honour this request but does not have to.

Attribute	Notes
range_bin_count unsigned short	Number of range bins that the CMS would like in the report. The subsystem should try to honour this request but does not have to.
elevation_bin_count unsigned short	The number of elevation bins that the CMS would like in the report. The subsystem should try to honour this request but does not have to.
start_azimuth azimuth_coordinate_type [0..1]	Defines the start of the arc of azimuth (positive orientation) of the volume in which the sensor's performance is to be assessed.
end_azimuth azimuth_coordinate_type [0..1]	Defines the end of the arc of azimuth (positive orientation) of the volume in which the sensor's performance is to be assessed.
start_elevation elevation_coordinate_type [0..1]	Defines the start of the arc of elevation (positive orientation) of the volume in which the sensor's performance is to be assessed.
end_elevation elevation_coordinate_type [0..1]	Defines the end of the arc of elevation (positive orientation) of the volume in which the sensor's performance is to be assessed.
min_range range_coordinate_type [0..1]	Defines the minimum range of the volume in which the sensor's performance is to be assessed.
max_range range_coordinate_type [0..1]	Defines the maximum range of the volume in which the sensor's performance is to be assessed.
frequency_bands frequency_band_type [0..*]	The set of frequency bands to assess the performance for. Where no bands are specified the performance is assessed for the sensor in general in the specified operational mode.
applicable_mode operational_mode_type	The performance assessment is to be in the context of this operational mode of the sensor subsystem.
coordinate_orientation coordinate_orientation_type	The orientation of the polar coordinates used in this class. Note that the origin is always the sensor reference point and that the coordinate system is always polar.

7.6.8.8 performance_assessment_report_type

Type: Class
Package: Sensor_Performance
Contains the results of a performance assessment.

Table 7.237 - Attributes of Class performance_assessment_report_type

Attribute	Notes
time_of_report time_type	The time of validity of the performance assessment.

Table 7.238 - Relations of Class performance_assessment_report_type

Connector	Notes
Aggregation: **band** performance_band_type [1..*]	The performance assessment for the band (or the sensor in general)
Aggregation: **assessment_dimensions** performance_assessment_parameters_type [1]	The actual dimensions of the assessment that is performed are reported with the result.

7.6.8.9 performance_band_type

Type: IDLStruct
Package: Sensor_Performance
The performance reported in a particular band (or in general)

Table 7.239 - Attributes of IDLStruct performance_band_type

Attribute	Notes
frequency_band frequency_band_type [0..1]	The specific band to which the contained performance assessments refers

Table 7.240 - Relations of IDLStruct performance_band_type

Connector	Notes
Aggregation: **sector** performance_sector_type [1..*]	The list of sectors in the performance assessment

7.6.8.10 performance_beam_type

Type: IDLStruct
Package: Sensor_Performance
Set of performance values for a line of points in space. Each value applies to a volume whose boundaries may be inferred from the numbers of bins and the min and max values in the report.

Table 7.241 - Attributes of IDLStruct performance_beam_type

Attribute	Notes
start_elevation elevation_coordinate_type	The start of the beam in elevation (positive orientation).
end_elevation elevation_coordinate_type	The end of the beam in elevation (positive orientation).

Table 7.242 - Relations of IDLStruct performance_beam_type

Connector	Notes
Aggregation: **bin** perfomance_bin_type [1..*]	The list of 'bins' in a beam of the performance assessment

7.6.8.11 performance_sector_type

Type: Class
Package: Sensor_Performance
A set of performance values for a sector of azimuth [start_azimuth..end_azimuth].

Table 7.243 - Attributes of Class performance_sector_type

Attribute	Notes
start_azimuth azimuth_coordinate_type	The start of the sector of azimuth (positive orientation).
end_azimuth azimuth_coordinate_type	The end of the sector of azimuth (positive orientation).

Table 7.244 - Relations of Class performance_sector_type

Connector	Notes
Aggregation: **beam** performance_beam_type [1..*]	The list of beams in the sector of the performance report

7.6.8.12 relative_performance_type

Type: Class
Package: Sensor_Performance
Defined as a signal excess in dB above noise floor for a nominal 0dB target strength, when assessing nominal performance or for the jammer when providing jammer assessment.

7.6.9 Track_Reporting

Parent Package: Sensor_Domain
This service provides facilities to report different types of sensor tracks.

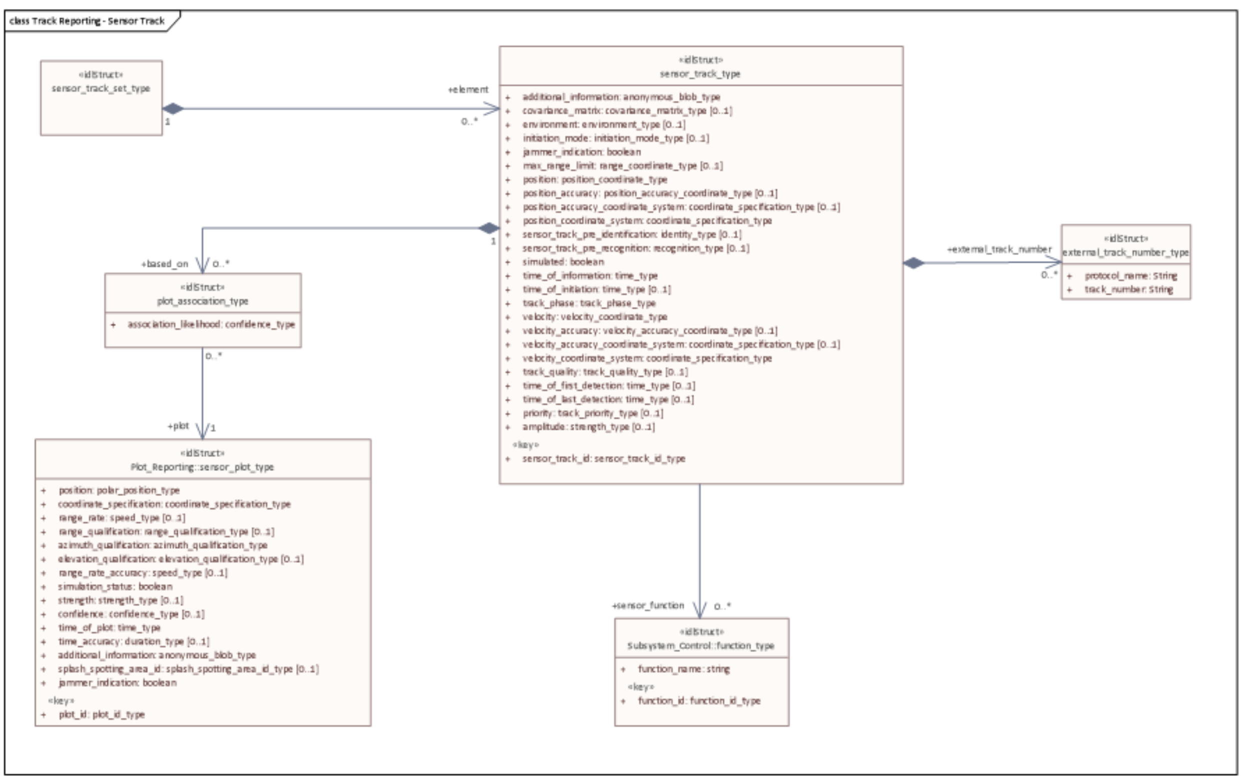


Figure 7.83 Track Reporting - Sensor Track (Class diagram)

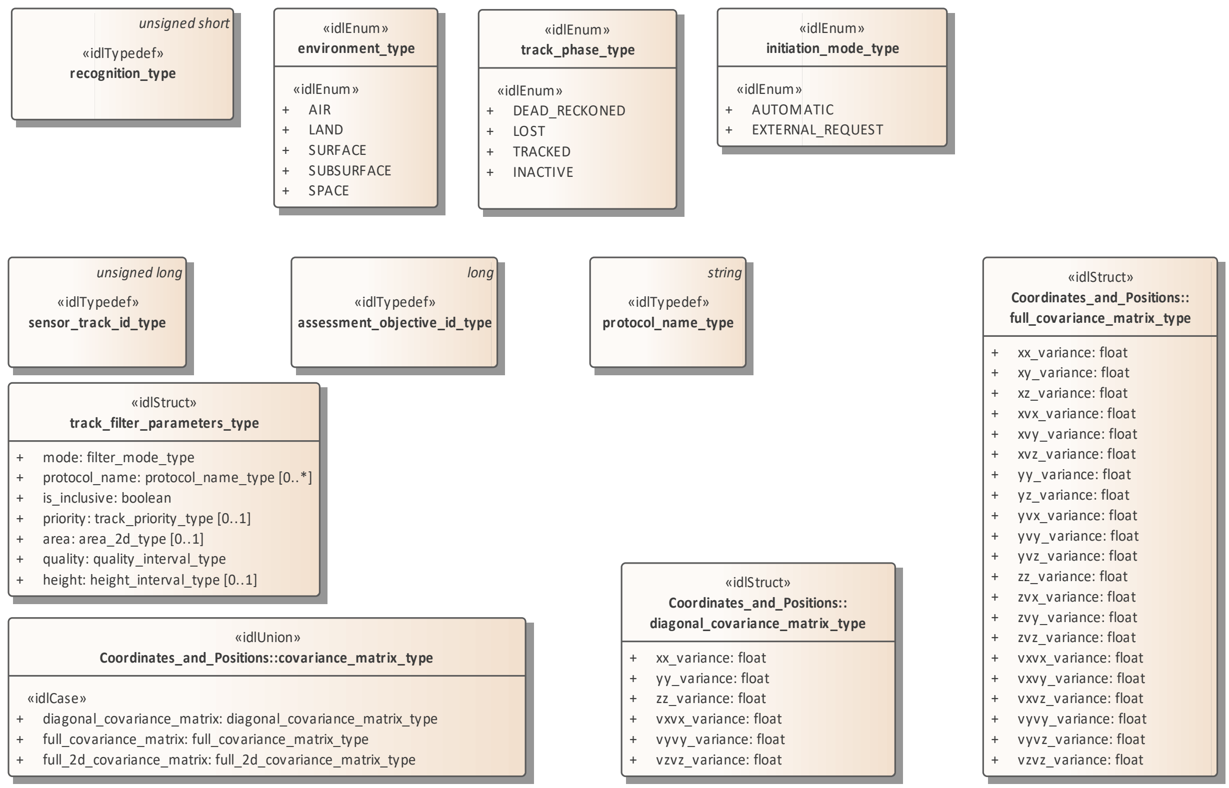


Figure 7.84 Track Reporting - Type Definitions (Class diagram)

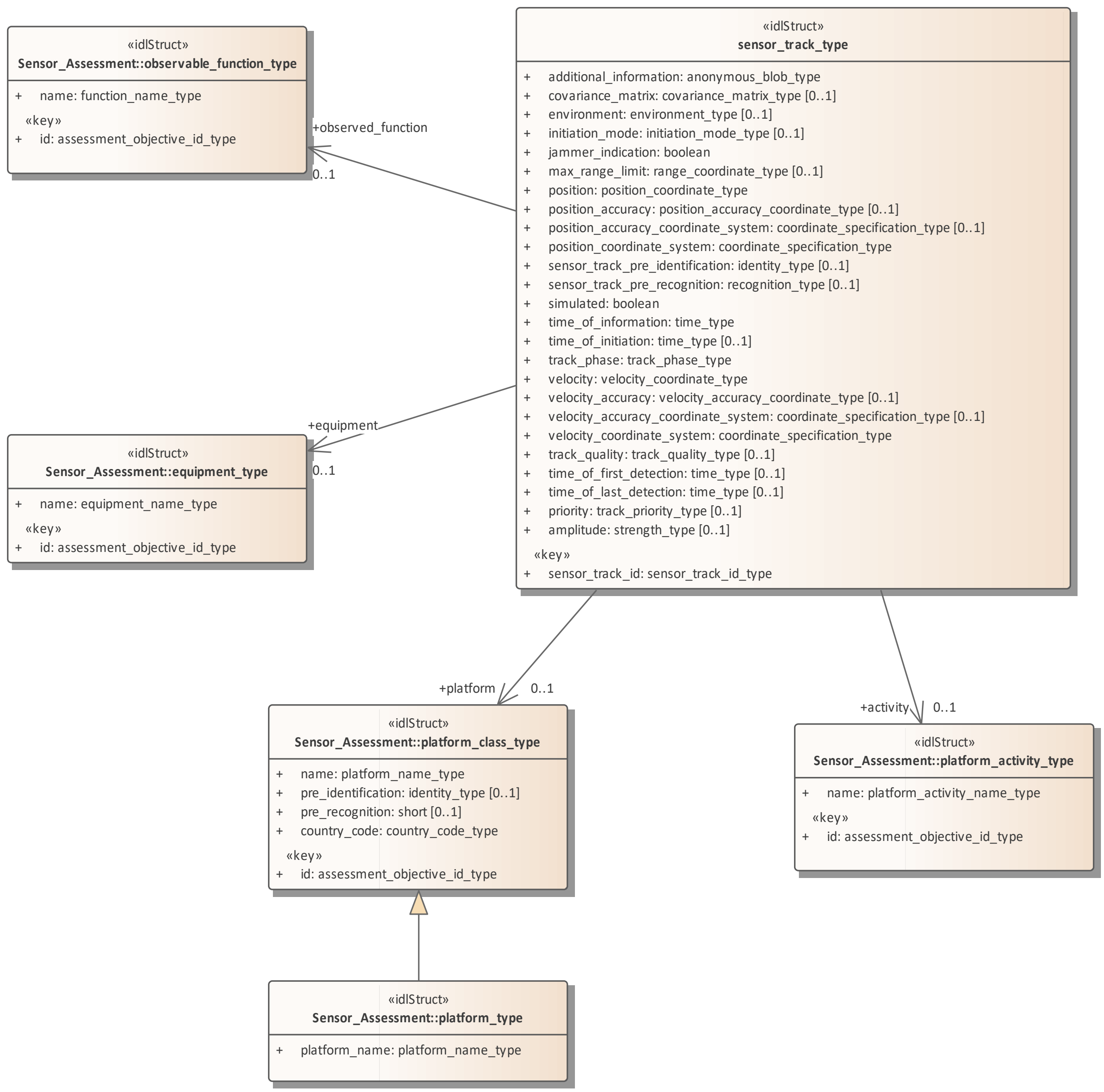


Figure 7.85 Track_Reporting - Assessment (Class diagram)

7.6.9.1 assessment_objective_id_type

Type: Class
Package: Track_Reporting

Unique identifier for the objects to which the sensor assessment is attempting to match the measurement parameters.

7.6.9.2 external_track_number_type

Type: IDLStruct
Package: Track_Reporting

A track number from an external protocol (to OARIS) such as a data-link.

Table 7.245 - Attributes of IDLStruct external_track_number_type

Attribute	Notes
protocol_name String	The name of the external protocol, system of network
track_number String	The string representation (human readable) of the external track number value.

7.6.9.3 plot_association_type

Type: IDLStruct
Package: Track_Reporting
This class represents an association between a sensor track and a sensor plot, supporting a multi-hypothesis many-to-many mapping between plots and tracks.

Table 7.246 - Attributes of IDLStruct plot_association_type

Attribute	Notes
association_likelihood confidence_type	The likelihood of this sensor plot given the prior sensor track. This is independent of the likelihood of other track associations. In general, the association likelihood values for a plot do not sum to one. AttributeTag: Issue =

Table 7.247 - Relations of IDLStruct plot_association_type

Connector	Notes
Association: **plot** sensor_plot_type reference [1]	The sensor plot that the sensor track is based on

7.6.9.4 protocol_name_type

Type: IDLTypeDef
Package: Track_Reporting
The name of an external protocol on which objects (e.g. tracks) could also be reported. Values are system implementation specific.

7.6.9.5 sensor_track_id_type

Type: IDLTypeDef
Package: Track_Reporting
Sensor Track Identification

7.6.9.6 track_filter_parameters_type

Type: IDLStruct
Package: Track_Reporting
The criteria that must all be met for a track to pass the filter. The filter attributes are applied with and-wise logic. For or-wise logic define multiple filter objects.

Table 7.248 - Attributes of IDLStruct track_filter_parameters_type

Attribute	Notes
mode filter_mode_type	The mode in which the tracks are filtered.
protocol_name protocol_name_type [0..*]	Filter tracks that are also being reported on these protocols.
is_inclusive boolean	If true, tracks that pass the filter are included in transmission and/or reception dependent upon the mode attribute. Otherwise, they are excluded.
priority track_priority_type [0..1]	A priority criterion. A track must have a priority greater than or equal to this.
area area_2d_type [0..1]	An area which is optionally part of the filter.
quality quality_interval_type	A track quality criterion. A track must have a track quality within the interval defined to pass the filter. If no filtering is required then an interval including all quality values is defined.
height height_interval_type [0..1]	The height values that are optionally part of the filter.

7.6.9.7 environment_type

Type: Class
Package: Track_Reporting
The sensor tracking environment

Table 7.249 - Attributes of Class environment_type

Attribute	Notes
«idlEnum» **AIR**	In the air
«idlEnum» **LAND**	On land
«idlEnum» **SURFACE**	On the sea surface
«idlEnum» **SUBSURFACE**	Below the sea surface
«idlEnum» **SPACE**	Outside the Earth's atmosphere

7.6.9.8 initiation_mode_type

Type: Class
Package: Track_Reporting
Type of track initiation

Table 7.250 - Attributes of Class initiation_mode_type

Attribute	Notes
«idlEnum» **AUTOMATIC**	Automatic track initiation mode

Attribute	Notes
«idlEnum» **EXTERNAL_REQUEST**	Track initiation on external request (e.g. from CMS)

7.6.9.9 recognition_type

Type: IDLTypeDef
Package: Track_Reporting
The recognition type indicates the type of the real-world physical object being tracked.
The numeric value is used to map to a system or implementation specific taxonomy of real-world physical objects that are of tactical interest.

7.6.9.10 sensor_track_type

Type: IDLStruct
Package: Track_Reporting
Encapsulation of a sensor track

Table 7.251 - Attributes of IDLStruct sensor_track_type

Attribute	Notes
additional_information anonymous_blob_type	Additional, vendor-specific information
covariance_matrix covariance_matrix_type [0..1]	The number of elements in the covariance matrix is dependent on the sensor. When present, the position_accuracy and velocity_accuracy attributes should not be present.
environment environment_type [0..1]	Environment of the track (air, surface etc)
initiation_mode initiation_mode_type [0..1]	Initiation mode of track (automatic or externally initiated)
jammer_indication boolean	Indication whether or not a track is jamming.
max_range_limit range_coordinate_type [0..1]	Maximal range for a bearing track
position position_coordinate_type	The location of the track as calculated in the sensor's chosen coordinate system at the stated time.
position_accuracy position_accuracy_coordinate_type [0..1]	The sensor's stated accuracy for its calculated position. When present, the covariance_matrix attribute should not be present.
position_accuracy_coordinate_system coordinate_specification_type [0..1]	The coordinate system chosen by the sensor for reporting accuracy.
position_coordinate_system coordinate_specification_type	The coordinate system chosen by the sensor.
«key» **sensor_track_id** sensor_track_id_type	The sensor's unique identifying reference for the track. Sensors may reuse identifiers after they have deleted the corresponding track. The scheme used for identifier reallocation is system dependent.

Attribute	Notes
sensor_track_pre_identification identity_type [0..1]	Identification information for the sensor track (if available)
sensor_track_pre_recognition recognition_type [0..1]	Recognition information for the sensor track (if available)
simulated boolean	Whether the CMS should process the track as having been synthetically generated as opposed to corresponding to an actual detection in the real world.
time_of_information time_type	The time at which the information in this object is valid, in particular its position.
time_of_initiation time_type [0..1]	The time at which the sensor first determined the existence of this track.
track_phase track_phase_type	Track phase (e.g. TRACKED, DELETED, LOST)
velocity velocity_coordinate_type	The velocity of the track as calculated in the sensor's chosen coordinate system at the stated time.
velocity_accuracy velocity_accuracy_coordinate_type [0..1]	The sensor's stated accuracy for its calculated velocity. When present, the covariance_matrix attribute should not be present.
velocity_accuracy_coordinate_system coordinate_specification_type [0..1]	The coordinate system chosen by the sensor for reporting accuracy.
velocity_coordinate_system coordinate_specification_type	The coordinate system chosen by the sensor.
track_quality track_quality_type [0..1]	The sensor specific quality of this track in comparison to its typical tracks.
time_of_first_detection time_type [0..1]	The time at which the sensor first made measurements leading to the detection of the existence of this track (as opposed to the time of initiation when there was sufficient confidence in one or more detection to initiate a track).
time_of_last_detection time_type [0..1]	The time at which the sensor last detected the existence of this track.
priority track_priority_type [0..1]	The relative priority of a track with regard to the sensor's resources
amplitude strength_type [0..1]	The amplitude or strength of the measurement(s) being tracked by the sensor

Table 7.252 - Relations of IDLStruct sensor_track_type

Connector	Notes
Association: **platform** platform_class_type reference [0..1]	The sensor's assessment of the name of the platform or class of platform of the real world object.
Association: **activity** platform_activity_type reference [0..1]	The sensor's assessment of the activity being undertaken by the real world object represented by the sensor track as observed by the sensor.

Connector	Notes
Association: **observed_function** observable_function_type reference [0..1]	The sensor's assessment of the function being undertaken by the equipment of real world object represented by the sensor track that has been observed by the sensor.
Association: **sensor_function** function_type reference [0..*]	The sensor functions whose processing has contributed to the track's data.
Association: **equipment** equipment_type reference [0..1]	The sensor's assessment of the equipment on the real world object represented by the sensor track that has been detected by the sensor.
Aggregation: **external_track_number** external_track_number_type [0..*]	A track number for this sensor track from another protocol
Aggregation~~ssociation~~: **based_on** plot_association_type reference [0..*]	The set of plots on which the creation or update of sensor track is based

7.6.9.11 sensor_track_set_type

Type: Class
Package: Track_Reporting
A set of sensor tracks (to enable batch reporting)

Table 7.253 - Relations of Class sensor_track_set_type

Connector	Notes
Aggregation: **element** sensor_track_type [0..*]	

7.6.9.12 track_phase_type

Type: Class
Package: Track_Reporting
The detection lifecycle phase of the track

Table 7.254 - Attributes of Class track_phase_type

Attribute	Notes
«idlEnum» **DEAD_RECKONED**	Track provided based on extrapolated position (dead-reckoned)
«idlEnum» **LOST**	Track has been lost
«idlEnum» **TRACKED**	Regular update of new and existing track
«idlEnum» **INACTIVE**	No new measurements were available to contribute to this track at the last opportunity to do so. It is expected that should such measurements be made at the next opportunity, these will successfully update the track.

7.6.10 Tracking_Control

Parent Package: Sensor_Domain
This package contains structs and type defs for managing tracking zones and sensor track information.

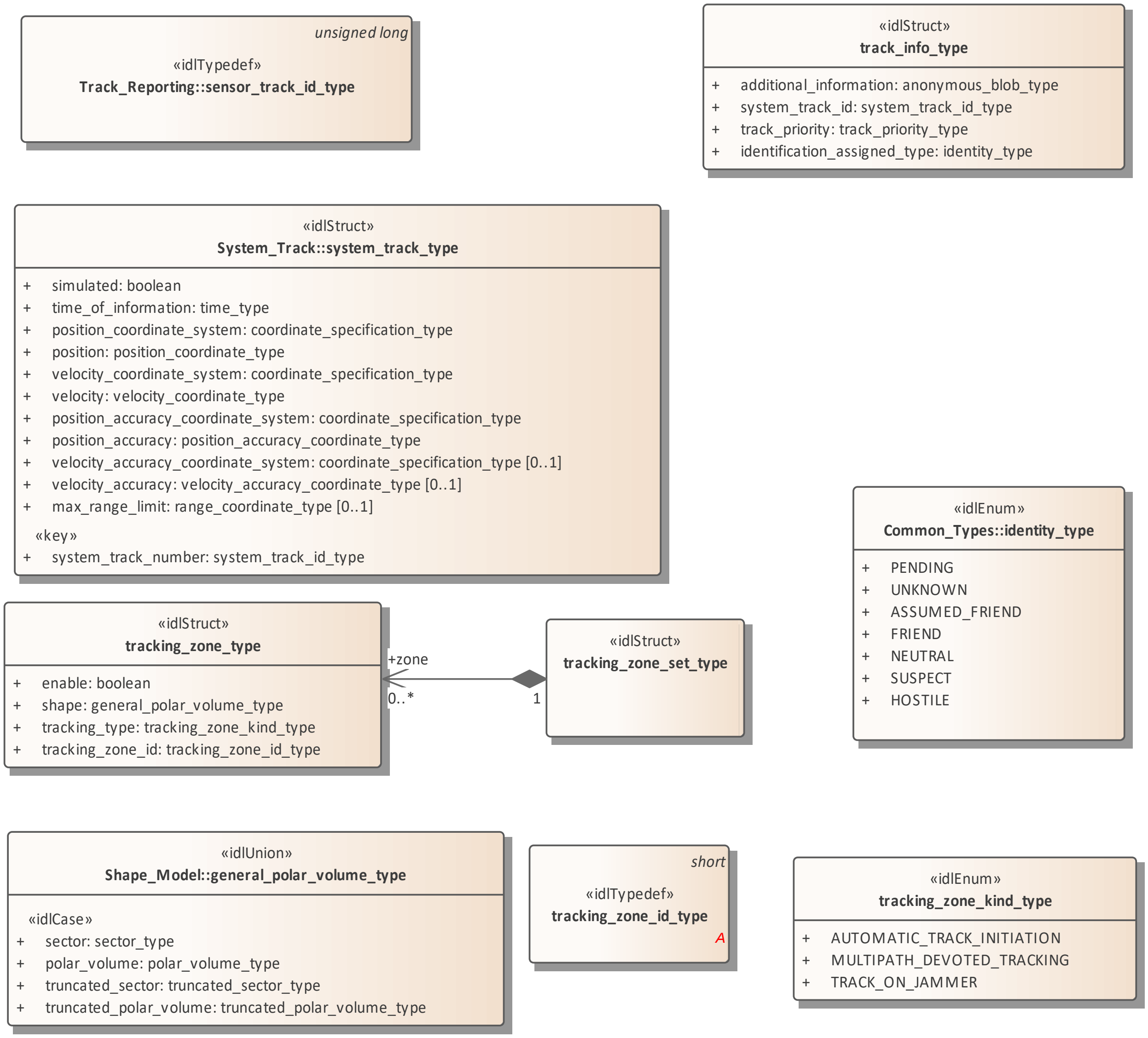


Figure 7.86 Domain Model (Class diagram)

7.6.10.1 track_info_type

Type: Class
Package: Tracking_Control
This struct identifies track information.

Table 7.255 - Attributes of Class track_info_type

Attribute	Notes
additional_information anonymous_blob_type	This is additional information that is not specified as part of the interface. Candidate information includes: - Track type~~,~~ ~~- Track priority,~~ ~~- Track Identification Category Assigned (Pending, Friend, Assumed Friend, Neutral, Unknown, Suspect, Hostile)~~.
system_track_id system_track_id_type	Identifier of the system track being described
track_priority track_priority_type	The priority assigned to the system track AttributeTag: Issue =
identification_assigned_type identity_type	The standard identity of the system track.

7.6.10.2 tracking_zone_set_type

Type: IDLStruct
Package: Tracking_Control
This struct contains zero to many tracking zones which must be set/reset by the CMS.

Table 7.256 - Relations of IDLStruct tracking_zone_set_type

Connector	Notes
Aggregation: **zone** tracking_zone_type [0..*]	

7.6.10.3 tracking_zone_type

Type: Class
Package: Tracking_Control
This struct identifies a tracking zone.

Table 7.257 - Attributes of Class tracking_zone_type

Attribute	Notes
enable boolean	Indicates whether the CMS is enabling or disabling a tracking zone.
shape general_polar_volume_type	This is the polar volume of the zone.
tracking_type tracking_zone_kind_type	This indicates the tracking zone type.
tracking_zone_id tracking_zone_id_type	A unique identifier for the tracking zone.

7.6.10.4 tracking_zone_kind_type

Type: Class
Package: Tracking_Control
Identifies the kind of a tracking zone.

Table 7.258 - Attributes of Class tracking_zone_kind_type

Attribute	Notes
AUTOMATIC_TRACK_INITIATION	Zones where the sensor is allowed to auto initiate new tracks. Depending on the sensor type and its capabilities, such a type of zones may be delimited in azimuth only, or both in azimuth and elevation, or may have further range bounds, and in some cases also additional constraints (such as target type, velocity bounds, etc.).
MULTIPATH_DEVOTED_TRACKING	Sectors where the sensor is required to use, for tracking activities, devoted waveforms to reduce the multipath effects. This capability is usually provided by multifunctional radars. Such a type of sectors is usually limited in azimuth only, below a defined elevation.
TRACK_ON_JAMMER	Sectors where the sensor is allowed to manage Track-On-Jammer. Depending on the sensor type and its capabilities, such a type of sectors may be delimited either in azimuth only or both in azimuth and elevation.

7.6.10.5 tracking_zone_id_type

Type: Class
Package: Tracking_Control
This typedef is used to identify a specific tracking zone.

7.7 Radar_Domain

Parent Package: Domain_Model
This package contains the Domain Models for the Air Engagement Support, Engagement Support, Missile Guidance, Search, and Surface Engagement Support services.

7.7.1 Air_Engagement_Support

Parent Package: Radar_Domain

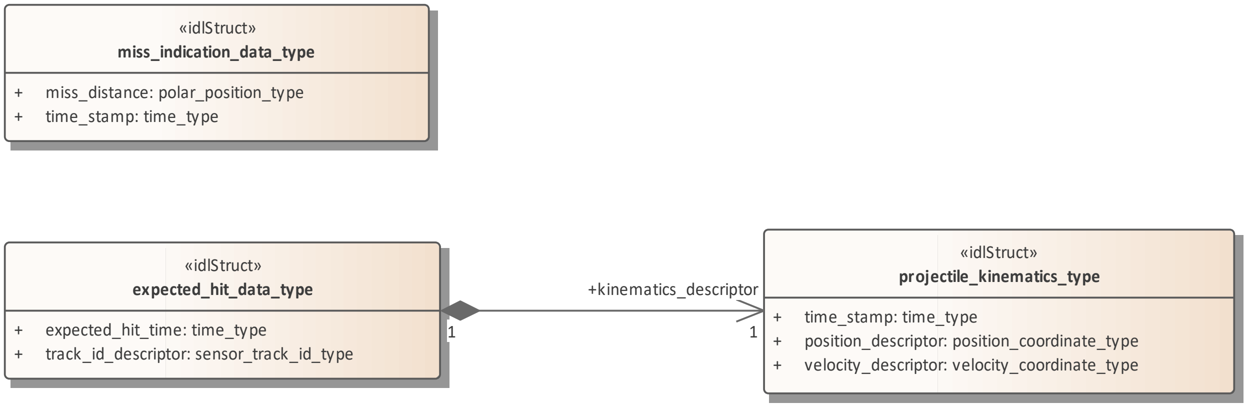


Figure 7.87 Domain Model (Class diagram)

7.7.1.1 expected_hit_data_type

Type: IDLStruct
Package: Air_Engagement_Support
Expected hit identifies the target and the time a hit is expected. This data is used to initiate the evaluation of a miss indication within the radar.

Table 7.259 - Attributes of IDLStruct expected_hit_data_type

Attribute	Notes
expected_hit_time time_type	Time when projectile is expected to hit the target.
track_id_descriptor sensor_track_id_type	The target track id.

Table 7.260 - Relations of IDLStruct expected_hit_data_type

Connector	Notes
Aggregation: **kinematics_descriptor** projectile_kinematics_type [1]	

7.7.1.2 miss_indication_data_type

Type: IDLStruct
Package: Air_Engagement_Support
Is sent once a hit or miss is noted.

Table 7.261 - Attributes of IDLStruct miss_indication_data_type

Attribute	Notes
miss_distance polar_position_type	Closest distance of the projectile to the target expressed in polar coordinates.
time_stamp time_type	Closest time of approach of the projectile to the target.

7.7.1.3 projectile_kinematics_type

Type: IDLStruct
Package: Air_Engagement_Support

Identifies the kinematics of the projectile that is expected to hit the target.

Table 7.262 - Attributes of IDLStruct projectile_kinematics_type

Attribute	Notes
time_stamp time_type	The timestamp when the kinematics was valid/measured.
position_descriptor position_coordinate_type	The projectile's position.
velocity_descriptor velocity_coordinate_type	The projectile's velocity.

7.7.2 Engagement_Support

Parent Package: Radar_Domain

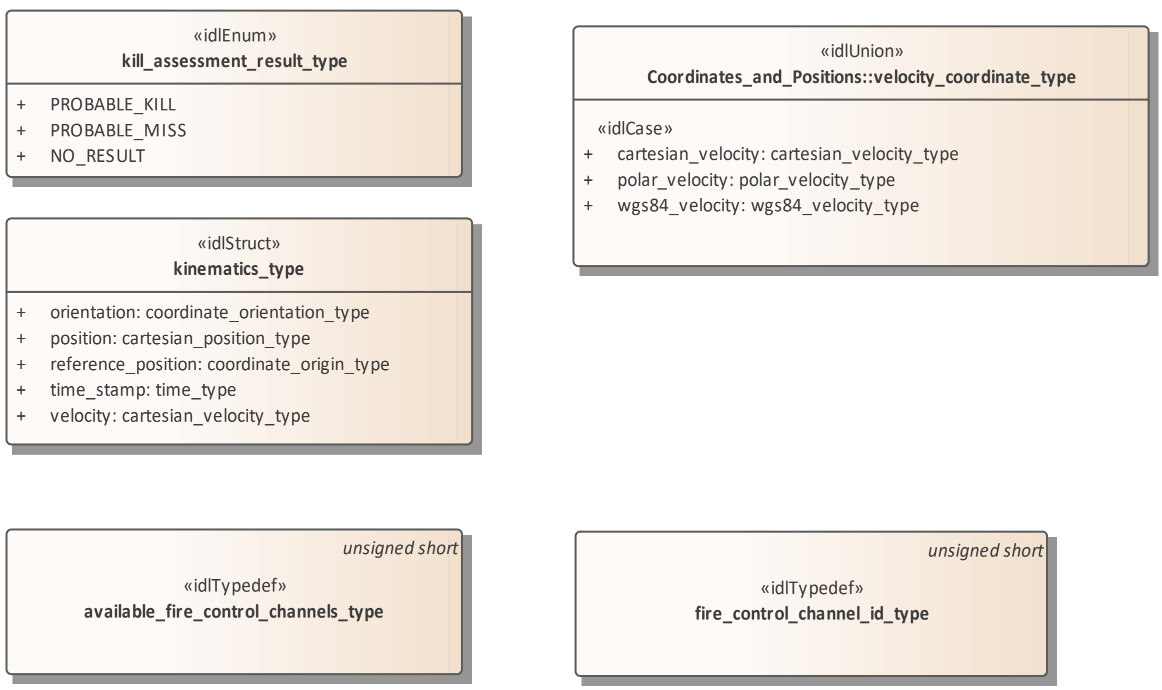


Figure 7.88 Domain Model (Class diagram)

7.7.2.1 available_fire_control_channels_type

Type: Class
Package: Engagement_Support
The number/amount of available fire control channels.

7.7.2.2 fire_control_channel_id_type

Type: Class

Package: Engagement_Support
The fire control channel ID as assigned by the subsystem.

7.7.2.3 kill_assessment_result_type

Type: Class
Package: Engagement_Support
The possible outcomes of a kill assessment.

Table 7.263 - Attributes of Class kill_assessment_result_type

Attribute	Notes
PROBABLE_KILL	Kill Probability > 50%
PROBABLE_MISS	Kill Probability < 50%
NO_RESULT	Assessment indeterminate

7.7.2.4 kinematics_type

Type: IDLStruct
Package: Engagement_Support
Target position/kinematics for which a fire control channel is requested to designate.

Table 7.264 - Attributes of IDLStruct kinematics_type

Attribute	Notes
orientation coordinate_orientation_type	The orientation of the kinematic coordinates
position cartesian_position_type	The positional element of the kinematics in Cartesian coordinates
reference_position coordinate_origin_type	The origin of the Cartesian coordinate frame
time_stamp time_type	The absolute time at which the kinematic information is valid.
velocity cartesian_velocity_type	The velocity element of the kinematics in Cartesian coordinates.

7.7.3 Missile_Guidance

Parent Package: Radar_Domain

Figure 7.89 Missile Guidance - Track (Class diagram)

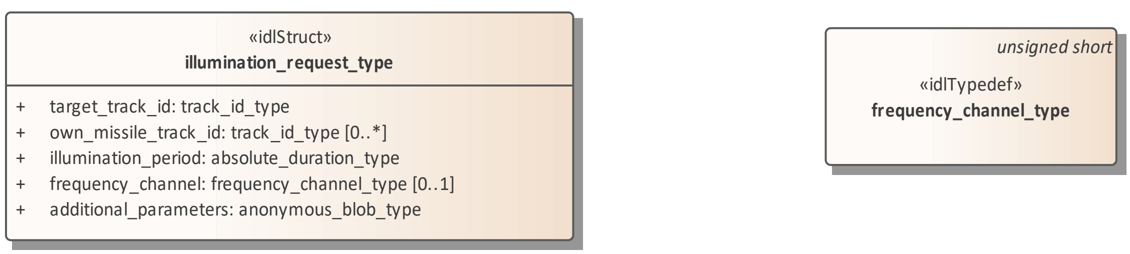


Figure 7.90 Illumination (Class diagram)

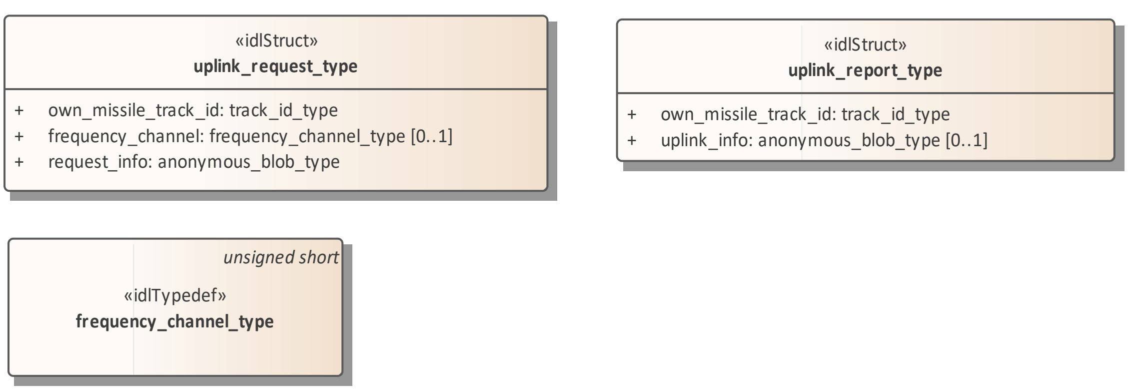


Figure 7.91 Missile Uplink (Class diagram)

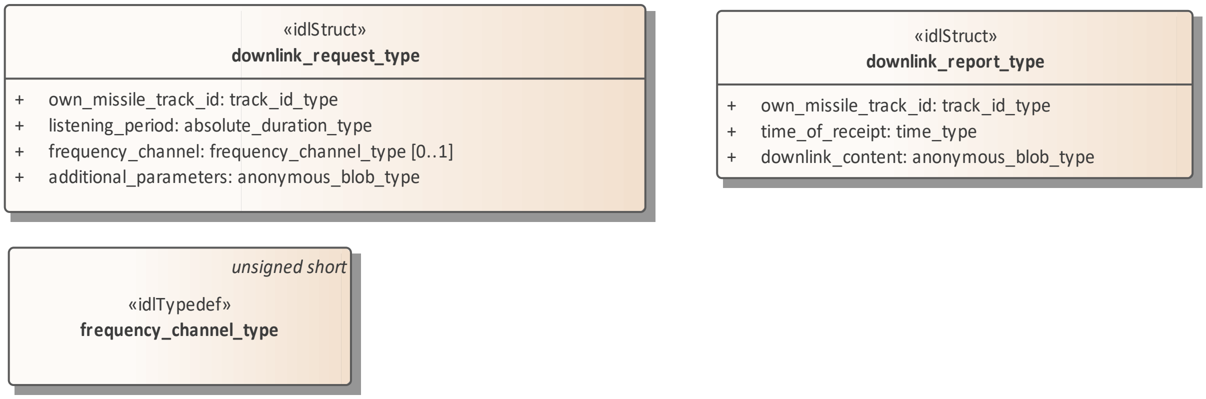


Figure 7.92 Missile Downlink (Class diagram)

7.7.3.1 downlink_report_type

Type: Class
Package: Missile_Guidance
Information downlinked by the missile to the radar.

Table 7.265 - Attributes of Class downlink_report_type

Attribute	Notes
own_missile_track_id track_id_type	The identifier for the track representing the missile providing the downlink.
time_of_receipt time_type	The absolute time at which the downlink was received
downlink_content anonymous_blob_type	The system specific content of the downlink from the missile.

7.7.3.2 downlink_request_type

Type: Class
Package: Missile_Guidance
request to downlink

Table 7.266 - Attributes of Class downlink_request_type

Attribute	Notes
own_missile_track_id track_id_type	The identifier for track that is representing the system's own missile in the engagement.
listening_period absolute_duration_type	The absolute period of time during which downlinks shall be received
frequency_channel frequency_channel_type [0..1]	Optionally the frequency channel to use for the downlink.

Attribute	Notes
additional_parameters anonymous_blob_type	System specific information to support the downlink

7.7.3.3 frequency_channel_type

Type: Class
Package: Missile_Guidance
A frequency channel identifies a specific radar frequency.

7.7.3.4 illumination_request_type

Type: Class
Package: Missile_Guidance
semantics of selects association is implementation specific.

Table 7.267 - Attributes of Class illumination_request_type

Attribute	Notes
target_track_id track_id_type	The identifier for the target track
own_missile_track_id track_id_type [0..*]	The identifier for track that is representing the system's own missile in the engagement.
illumination_period absolute_duration_type	The length of time to provide illumination of the target
frequency_channel frequency_channel_type [0..1]	The frequency channel to use for target illumination
additional_parameters anonymous_blob_type	System specific information to support the illumination

7.7.3.5 track_id_type

Type: Class
Package: Missile_Guidance
The track referred to by a missile guidance command may either be a system track (provided by the CMS) or a sensor track (if the object is already tracked by the sensor). Therefore, the track_id(s) in the missile guidance command may be a sensor_track_id or a missile_track_id.

Table 7.268 - Attributes of Class track_id_type

Attribute	Notes
«idlCase» **sensor_track_id** sensor_track_id_type	sensor track id option
«idlCase» **system_track_id** system_track_id_type	system track id option

7.7.3.6 uplink_report_type

Type: IDLStruct
Package: Missile_Guidance
a report from uplink

Table 7.269 - Attributes of IDLStruct uplink_report_type

Attribute	Notes
own_missile_track_id track_id_type	The identifier for track that is representing the system's own missile in the engagement.
uplink_info anonymous_blob_type [0..1]	System specific information to support the uplink

7.7.3.7 uplink_request_type

Type: IDLStruct
Package: Missile_Guidance
a request to downlink

Table 7.270 - Attributes of IDLStruct uplink_request_type

Attribute	Notes
own_missile_track_id track_id_type	The identifier for track that is representing the system's own missile in the engagement.
frequency_channel frequency_channel_type [0..1]	Optionally, the frequency channel to use for the uplink.
request_info anonymous_blob_type	System specific information regarding the uplink.

7.7.4 Surface_Engagement_Support

Parent Package: Radar_Domain

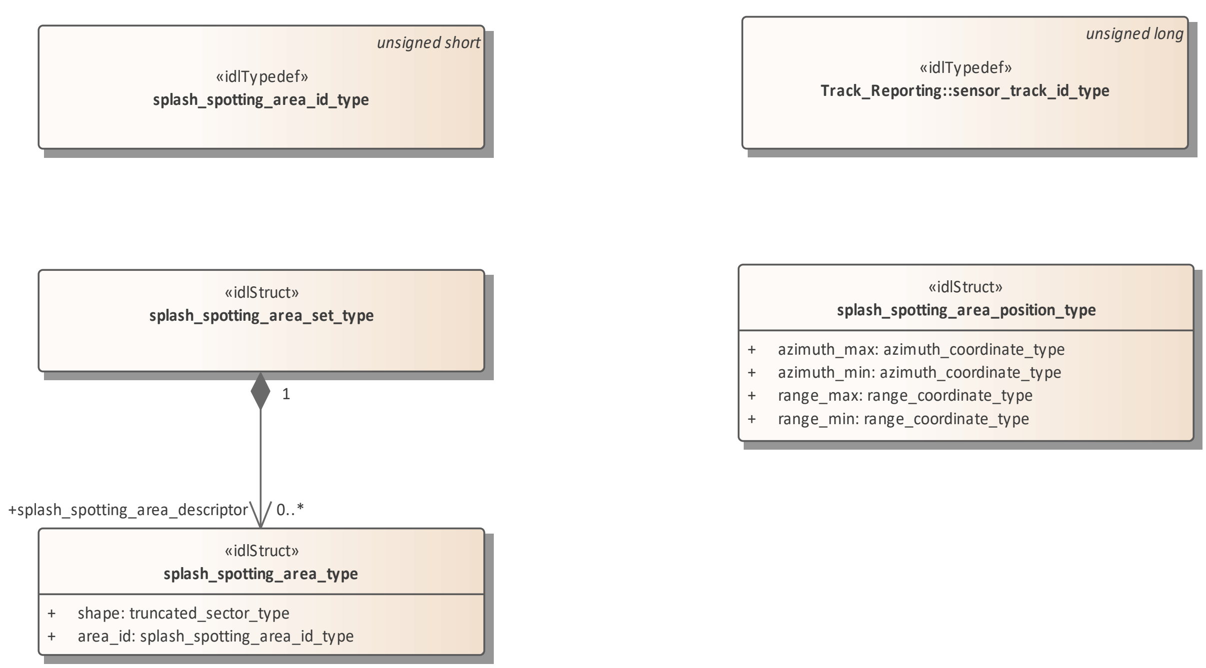


Figure 7.93 Domain Model (Class diagram)

7.7.4.1 splash_spotting_area_id_type

Type: Class
Package: Surface_Engagement_Support
the area ID assigned by the sensor.

7.7.4.2 splash_spotting_area_position_type

Type: Class
Package: Surface_Engagement_Support
The area definition from the User (CMS) when Splash Spotting is defined using the service "activate splash spotting area by position". The minimum and maximum available sizes are defined in "Manage Subsystem Parameters".

Table 7.271 - Attributes of Class splash_spotting_area_position_type

Attribute	Notes
azimuth_max azimuth_coordinate_type	when max is less than min, areas covers the north azimuth
azimuth_min azimuth_coordinate_type	when min is less than max, areas covers the north azimuth
range_max range_coordinate_type	limited to less than or equal to instrumented range
range_min range_coordinate_type	limited to greater than or equal to minimum visible range

7.7.4.3 splash_spotting_area_set_type

Type: Class
Package: Surface_Engagement_Support
A set consisting of splash spotting areas.

Table 7.272 - Relations of Class splash_spotting_area_set_type

Connector	Notes
Aggregation: **splash_spotting_area_descriptor** splash_spotting_area_type [0..*]	

7.7.4.4 splash_spotting_area_type

Type: Class
Package: Surface_Engagement_Support
Definition of a single splash spotting area.

Table 7.273 - Attributes of Class splash_spotting_area_type

Attribute	Notes
shape truncated_sector_type	Shape and size of the splash spotting area
area_id splash_spotting_area_id_type	Area ID of the splash spotting area.

7.8 Subsystem_Services

Parent Package: Service_Interfaces
Contains services associated with the Subsystem Domain.

7.8.1 Encyclopaedic_Support

Parent Package: Subsystem_Services

7.8.1.1 Receive_Encyclopaedic_Data

Parent Package: Encyclopaedic_Support

Receive_Encyclopaedic_Data_CMS
Type: Interface
Package: Receive_Encyclopaedic_Data
This interface describes the process whereby the subsystem receives encyclopedic data.Such data is used by the subsystem to perform self-adaptation to the prevailing environmental conditions.
This interface is modelled as a control interaction between the CMS and the subsystem rather than a data flow interaction. The CMS controls the loading of subsystem encyclopaedic data by sending the location of the data, rather than sending the data itself. Of course an implementation may move the encyclopaedic data around a file system beforehand, but that is outside the scope of this standard.
The subsystem is aware of its real-time geographic position and orientation.
It is expected that the transfer of this data would be initiated at the start of the 'mission of the day'. Updates would only be envisaged when the current data set became inapplicable to the current mission.
Specific encyclopedic data might be requested by the subsystem. Alternatively, a default set of summary data is sent. Such data, which is an example of 'reference' data, would generally be non-sensor in origin and static i.e. not changing in real-time. In the simplest case this data might simply define clutter areas and known jammer locations to assist the subsystem in effecting suitable mitigation for these effects. For a subsystem such as a more complex multi-function radar this might include relevant extracts from a commercial shipping database (Lloyd's etc.), giving shipping lanes or ship movements or civil airline flight plan data (Civil Aviation Authority etc), locations of wind-farms, major highways, significant structures and potential sources of interference, such as other radars, including consorts, cellular phone masts etc. This data would be used by the subsystem to contribute to the tactical picture. Alternatively, it could be used within the automatic tracking function to enable the identification/elimination from the track picture of non-hostile tracks. Such data could also include, for example, the reference data types communicated via Link 16 such as hazard areas and other fixed point type data. Navigational charts might also be a part of such data. The subsystem VOI (volume of interest) or other filter mechanisms might be supplied in a request from the actor.
Pre-condition: Technical State: The subsystem is in technical state STANDBY, READY or ONLINE
Pre-condition: Mastership Required: The CMS has mastership
Pre-condition: Subsystem Services: Provide Subsystem Services has completed successfully, in particular this service is available.
Post-condition: Success: The subsystem has received updated Encyclopedic Data.
Post-condition: No Success: The subsystem has not received updated Encyclopedic Data

Table 7.274 - Methods of Interface Receive_Encyclopaedic_Data_CMS

Method	Notes	Parameters
encyclopaedic_data_loaded()	The subsystem responds to the CMS that the encyclopaedic data previously requested has been loaded.	request_id_type **request_id** The unique id for this request - corresponds to the parameter in the load_encyclopaedic_data request

Receive_Encyclopaedic_Data_Sub
Type: Interface
Package: Receive_Encyclopaedic_Data

Table 7.275 - Methods of Interface Receive_Encyclopaedic_Data_Sub

Method	Notes	Parameters
load_encyclopaedic_data()	The CMS requests the subsystem to load encyclopaedic data of a particular type from a particular location.	request_id_type **request_id** The unique identifier for this request url_type **url** The location of the file containing the encyclopaedic data data_descriptor_type **data_descriptor** A description of the type of encyclopaedic data (e.g. name of the data set). It is expected that implementations will specify a list of descriptors known to particular subsystems. Such a list may be accessible at run-time through the Manage Subsystem Parameters interface.

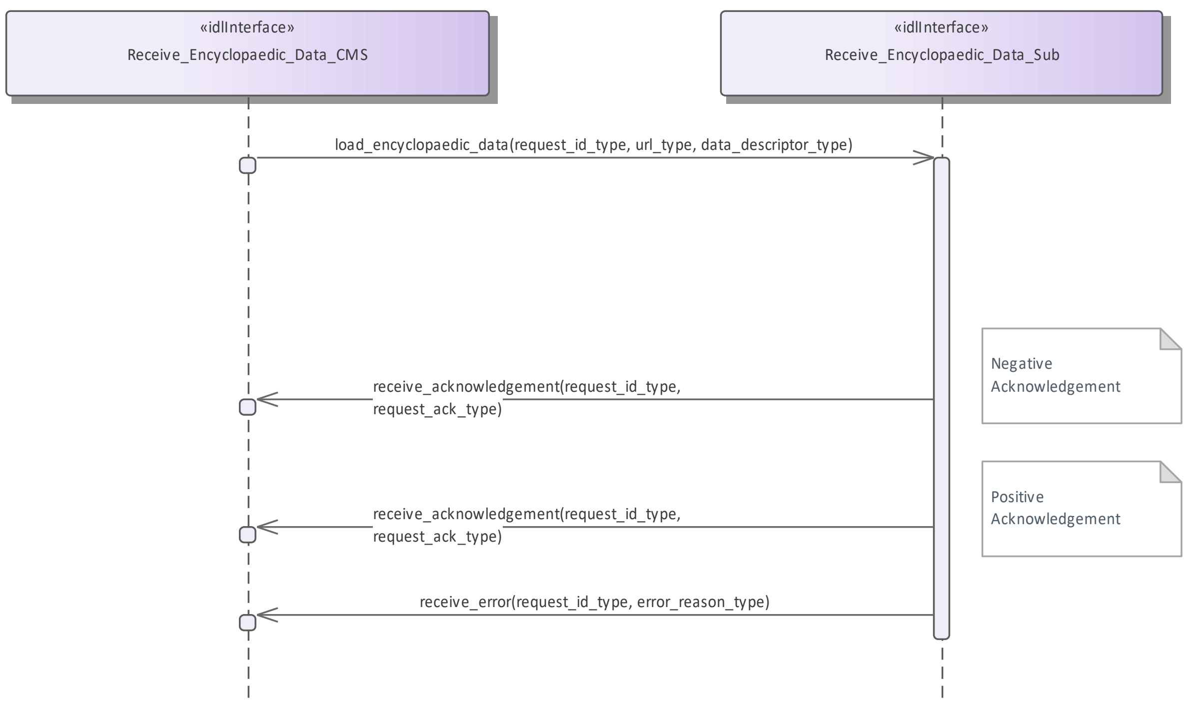


Figure 7.94 Alternate Flow - Receive Encyclopaedic Data (Interaction diagram)

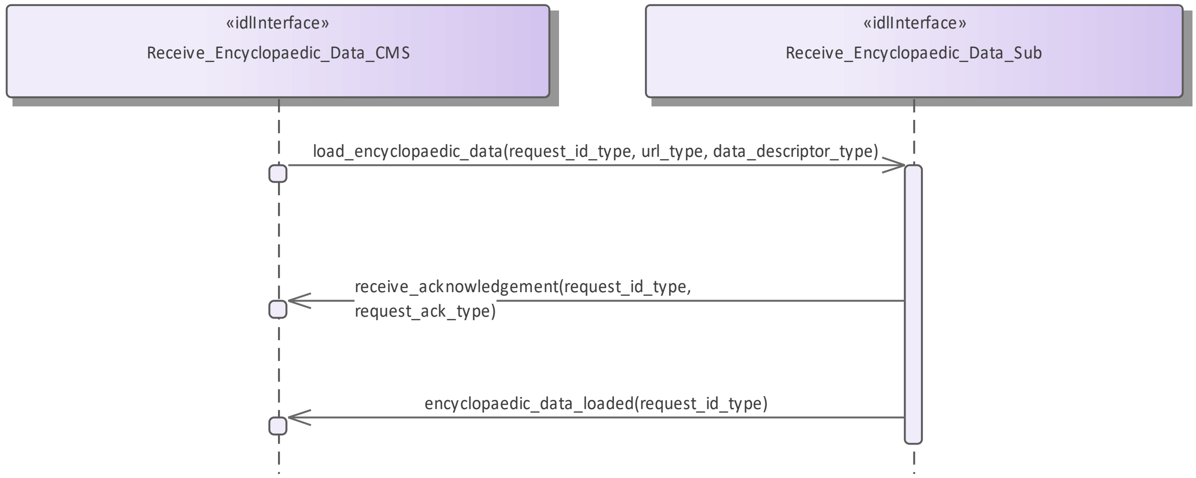


Figure 7.95 Basic Flow - Receive Encyclopaedic Data (Interaction diagram)

7.8.2 Extended_Subsystem_Control

Parent Package: Subsystem_Services
Contains interfaces for the Extended Subsystem Control service.

7.8.2.1 Manage_Physical_Configuration

Parent Package: Extended_Subsystem_Control
Contains operations and sequence diagrams for the Manage Physical Configuration interface.
Manage_Physical_Configuration_CMS
Type: Interface
Package: Manage_Physical_Configuration
The purpose of this interface is to provide a mechanism to exchange a physical configuration data file between a subsystem and the CMS (potentially xml format). The exact format of the file is subsystem specific. The purpose of the file is to support the maintainer with facilities to configure the internal parts of the subsystem; also to be used as integration support.

Additional Information:

There are at least two cases where the CMS would provide a sub-system's physical configuration. Case 1 is when the sub-system was able to detect a configuration change and the data must be manually entered in sub-system configuration data (e.g. a servo type and serial number). Case 2 is when the sub-system is being developed and changes to the configuration which cause changes in system behavior are being tested.
Pre-condition: Subsystem must be in a STANDBY state in order for the CMS to request changes to Physical Configuration Data. This precondition does not apply if the CMS is only requesting current Physical Configuration Data to be provided by the subsystem. :
Pre-condition: CMS must have mastership in order for the CMS to request changes to Physical Configuration Data. This precondition does not apply if the CMS is only requesting current Physical Configuration Data to be provided by the subsystem. :
Post-condition: For a change in Physical Configuration Data Request, configuration data is properly updated.:

Table 7.276 - Methods of Interface Manage_Physical_Configuration_CMS

Method	Notes	Parameters
receive_physical_configuration()	Interface used by CMS to receive a url to access physical configuration data from the subsystem.	configuration_url_type **configuration_url** request_id_type **request_id**

receive_physical_configuration_success()	Interface used by CMS to receive an indication from the subsystem that it has successfully changed its physical configuration data.	request_id_type **request_id**

Manage_Physical_Configuration_Sub
Type: Interface
Package: Manage_Physical_Configuration

Table 7.277 - Methods of Interface Manage_Physical_Configuration_Sub

Method	Notes	Parameters
change_physical_configuration()	Interface used by the subsystem to receive requests from the CMS to change its physical configuration data to align with data located at the url specified in the request.	request_id_type **request_id** configuration_url_type **configuraiton_url**
provide_physical_configuration()	Interface used by the subsystem to receive requests from the CMS to provide its current physical configuration data.	request_id_type **request_id**

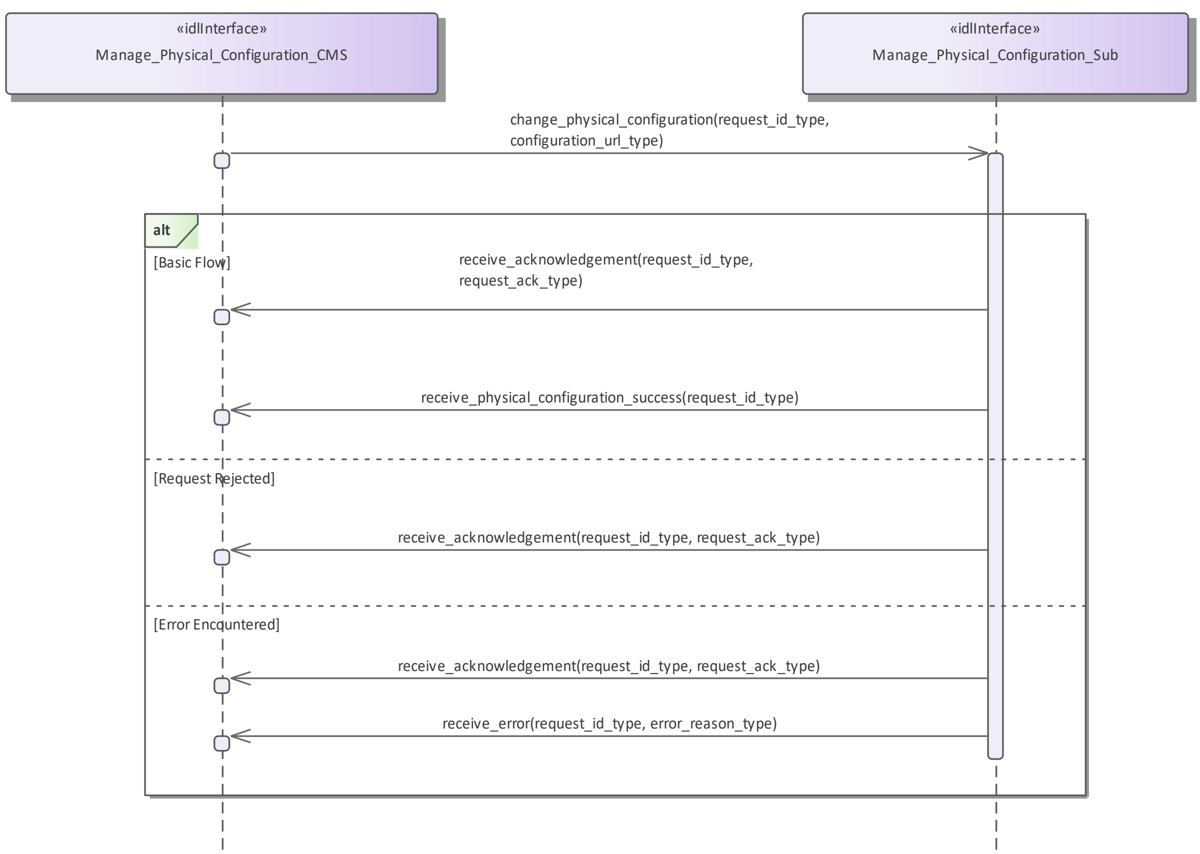


Figure 7.96 Manage Physical Configuration - Change (Interaction diagram)

Flow of events which depicts the CMS requesting that the subsystem changing its physical configuration data (also depicts alternate rejection and error paths).

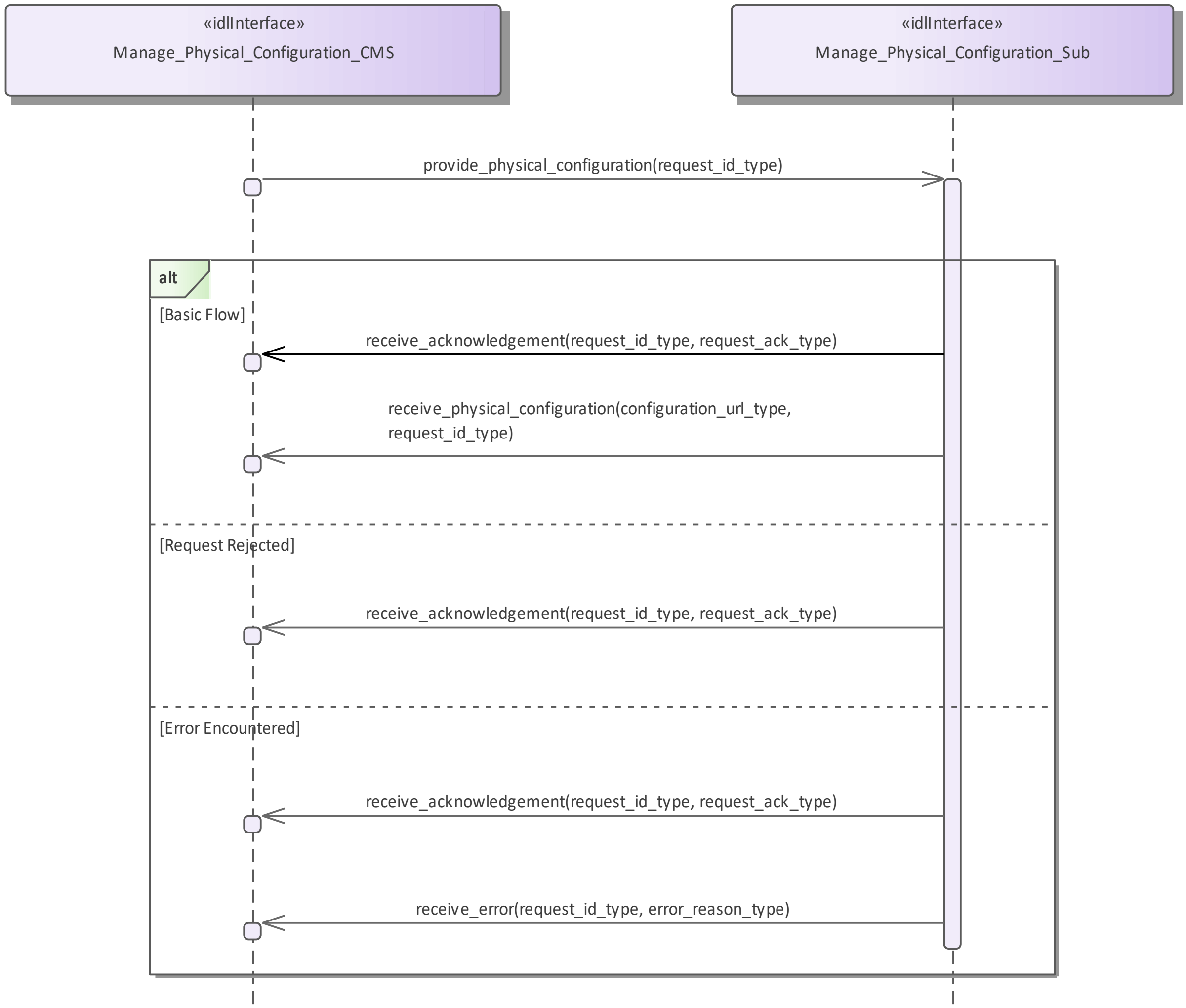


Figure 7.97 Manage Physical Configuration - Request (Interaction diagram)

Flow of events which depicts the CMS requesting that the subsystem report on its current physical configuration data (also depicts alternate rejection and error paths).

7.8.2.2 Perform_Offline_Test

Parent Package: Extended_Subsystem_Control
Contains the interface for offline testing.
Perform_Offline_Test_CMS
Type: Interface
Package: Perform_Offline_Test
This is used to instruct the subsystem to perform offline test and return the results to the CMS. The nature of the offline tests is subsystem specific
Pre-condition: Provide Subsystem Services must have executed successfully.:
Pre-condition: The CMS must have Mastership:
Pre-condition: The subsystem may be in any Technical State except for ONLINE:
Post-condition: For the response FAILED, the subsystem transitions to Technical State FAILED, but otherwise remains in the previous Technical State.:

Table 7.278 - Methods of Interface Perform_Offline_Test_CMS

Method	Notes	Parameters
receive_detailed_test_results()	Provides the CMS with subsystem specific information concerning offline test failures	request_id_type **request_id** offline_test_result_details_type **offline_test_detailed_results**
receive_test_results()	Informs the CMS whether the offline tests passed, passed partially, or failed.	request_id_type **request_id** offline_test_result_type **test_results**

Perform_Offline_Test_Sub
Type: Interface
Package: Perform_Offline_Test

Table 7.279 - Methods of Interface Perform_Offline_Test_Sub

Method	Notes	Parameters
perform_tests()	Instructs the subsystem to perform the offline tests.	request_id_type **request_id** offline_test_type **test_name** Allows a particular test to be selected. If null, all tests are performed.
request_detailed_test_results()	Asks the subsystem to provide detailed information on the failures.	request_id_type **request_id**

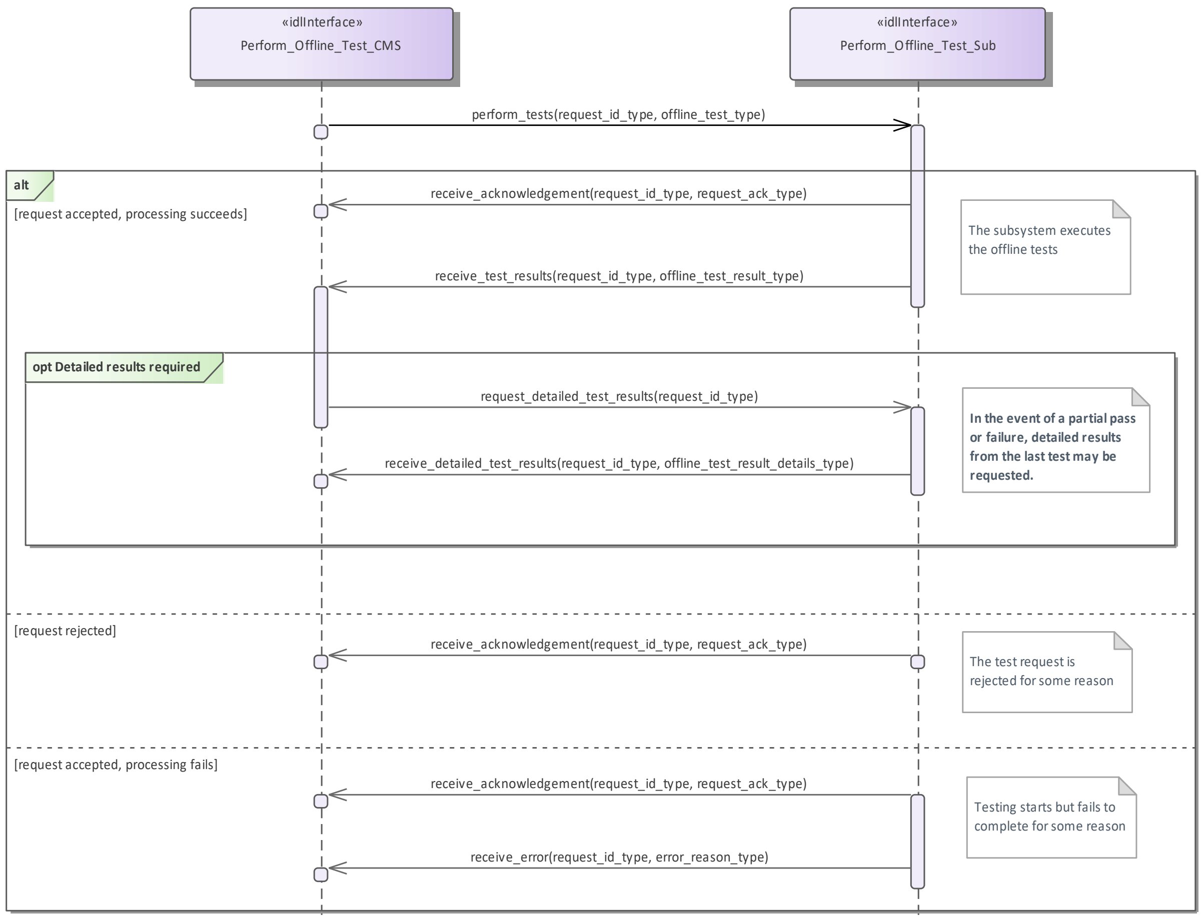


Figure 7.98 Perform Offline Test (Interaction diagram)

This shows the required sequential behaviour for Perform_Offline_Test, See diagram embedded notes for further explanation

7.8.2.3 Manage_Network_Participation

Parent Package: Extended_Subsystem_Control

Manage_Network_Participation_CMS
Type: Interface
Package: Manage_Network_Participation
The purpose of this interface is to provide a mechanism for a CMS to manage the connectivity of the OARIS data exchange through some external network gateway represented by the Subsystem interface in this use case. When connectivity is established, information can be exchanged between the local CMS and local Subsystems with other CMS and Subsystems connected by this network as if they were locally connected from a functional viewpoint.
Additional Information:
The management of such network connectivity may be integral for the sharing of plot data between distributed platforms.

Table 7.280 - Methods of Interface Manage_Network_Participation_CMS

Method	Notes	Parameters
report_network_status()	Send a report to the CMS when the	String **network_name** The name identifying the network subsystem_id_type **joined_local_subsystems** The

		subsystems joined to the network at the node that the CMS is controlling subsystem_id_type **joined_remote_subsystems** The subsystems joined to the network at other nodes
network_status_response()		String **network_name** The identifyng name for the network boolean **is_joined** True if the subsystem is joined to the network; false otherwise subsystem_id_type **statused_subsystem_id** The identifier for the subsystem that is the subject of the request request_id_type **request_id** The identifier of the originating request

Manage_Network_Participation_Sub
Type: Interface
Package: Manage_Network_Participation

Table 7.281 - Methods of Interface Manage_Network_Participation_Sub

Method	Notes	Parameters
request_network_status()		String **network_name** The name identifying the network request_id_type **request_id** The unique identifier for this specific request
join_network()		String **network_name** The name identifying the network to be joined request_id_type **request_id** The unique identifier for this specific request subsystem_id_type **joining_subsystem_id** The unique identifier of the subsystem to join the network
leave_network()		String **network_name** The name identifying the network to be left request_id_type **request_id** The unique identifier for this specific request subsystem_id_type **leaving_subsystem_id** The unique identifier for the subsystem to leave the network

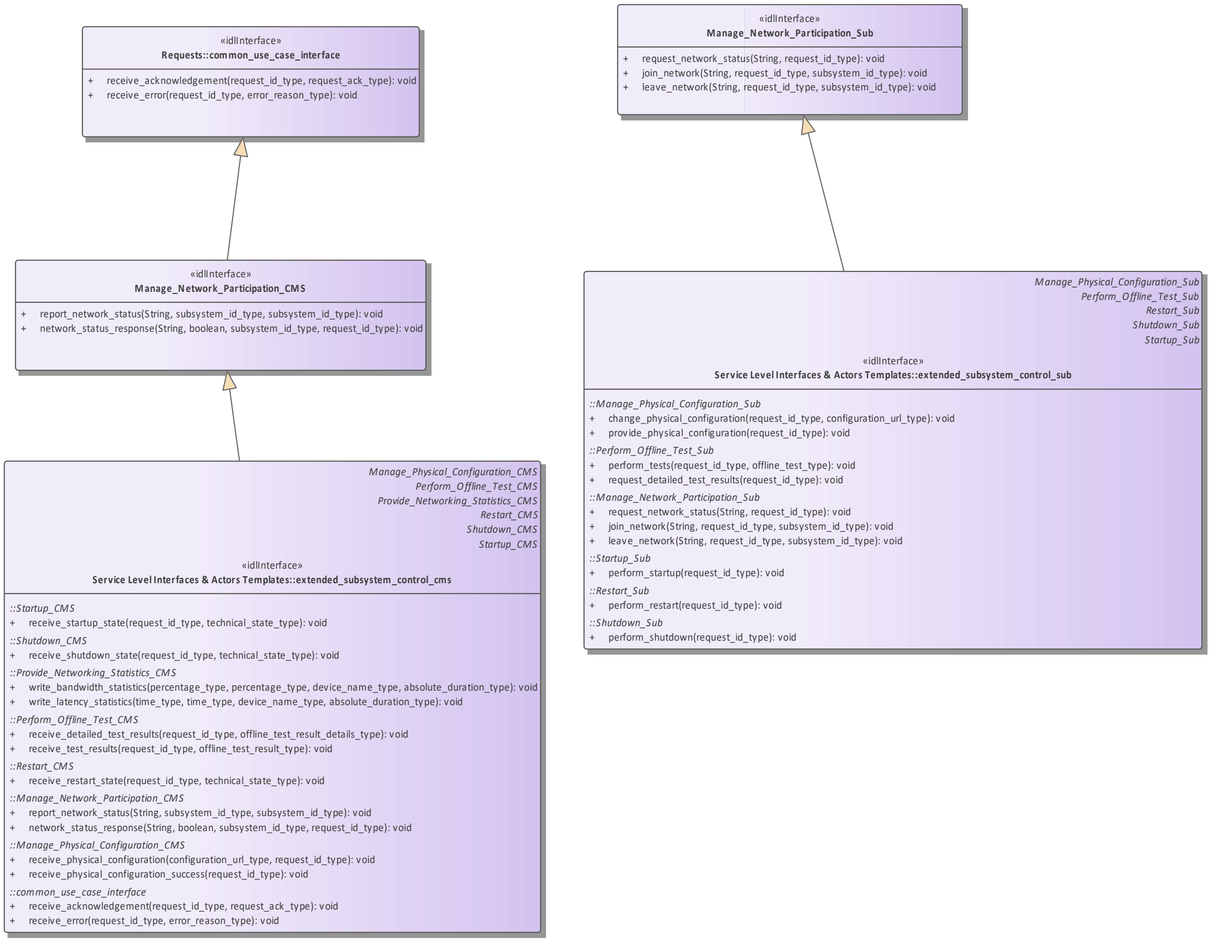


Figure 7.99 Manage_Network_Participation (Class diagram)

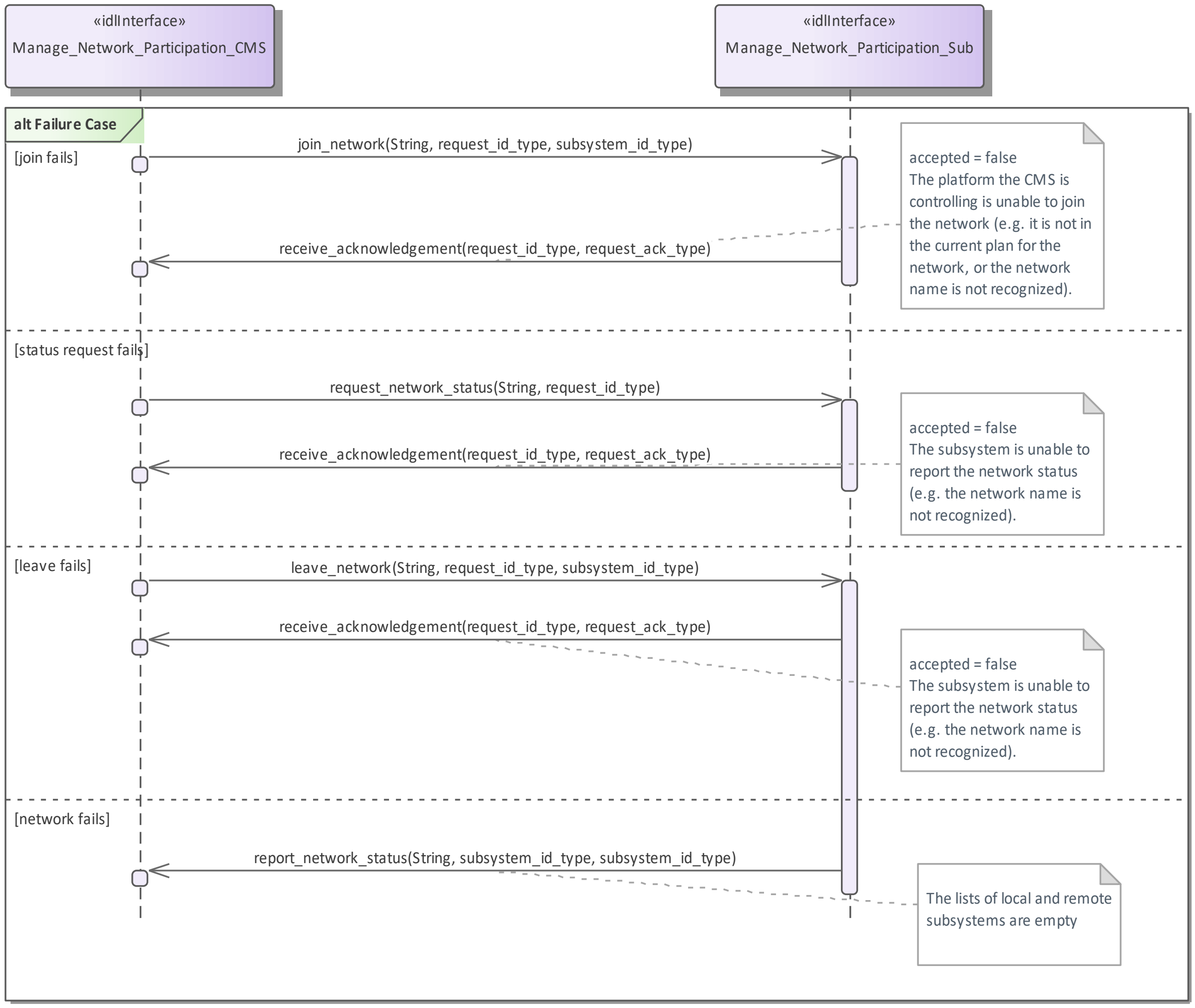


Figure 7.100 Manage_Network_Participation - alternate flow - unable to join (Interaction diagram)

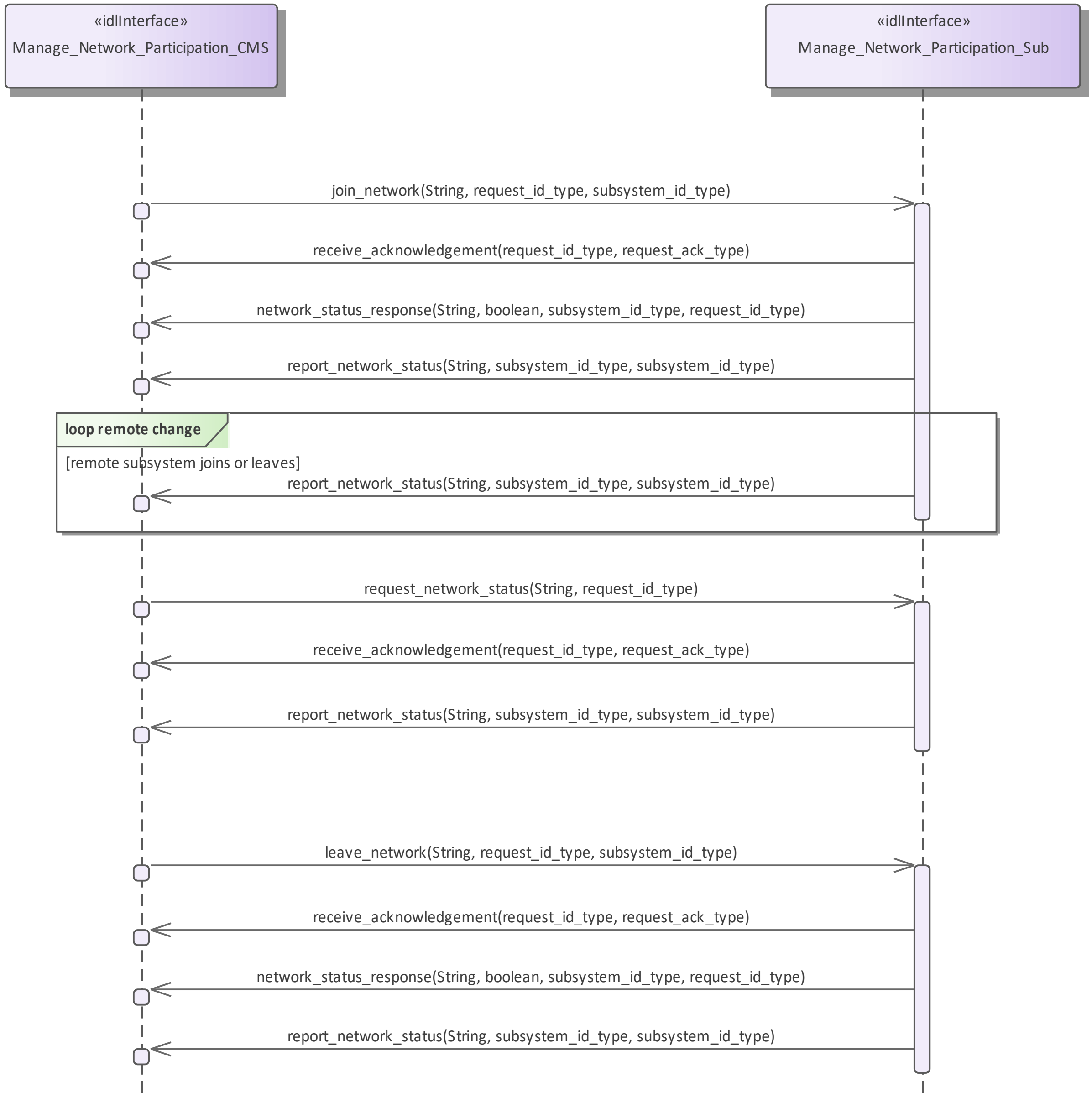


Figure 7.101 Manage_Network_Participation - basic flow (Interaction diagram)

7.8.2.4 Startup

Parent Package: Extended_Subsystem_Control
Contains operations and sequence diagrams for the Startup interface.
Startup_CMS
Type: Interface
Package: Startup
The purpose of this interface is to cause a normal transition from the STANDBY state to the READY state using the transitions defined in the Manage Technical State service.

Pre-condition: Subsystem is in STANDBY State.:

Pre-condition: CMS has mastership of subsystem.:

Post-condition: Subsystem is in READY state if successful. If not execute successful, current state shall be reported by subsystem.:

Table 7.282 - Methods of Interface Startup_CMS

Method	Notes	Parameters
receive_startup_state()	Interface used by CMS to receive an indication from the subsystem that it has successfully performed startup.	request_id_type **request_id** technical_state_type **technical_state**

Startup_Sub
Type: Interface
Package: Startup

Table 7.283 - Methods of Interface Startup_Sub

Method	Notes	Parameters
perform_startup()	Interface used by the subsystem to receive a request from the CMS to execute startup.	request_id_type **request_id**

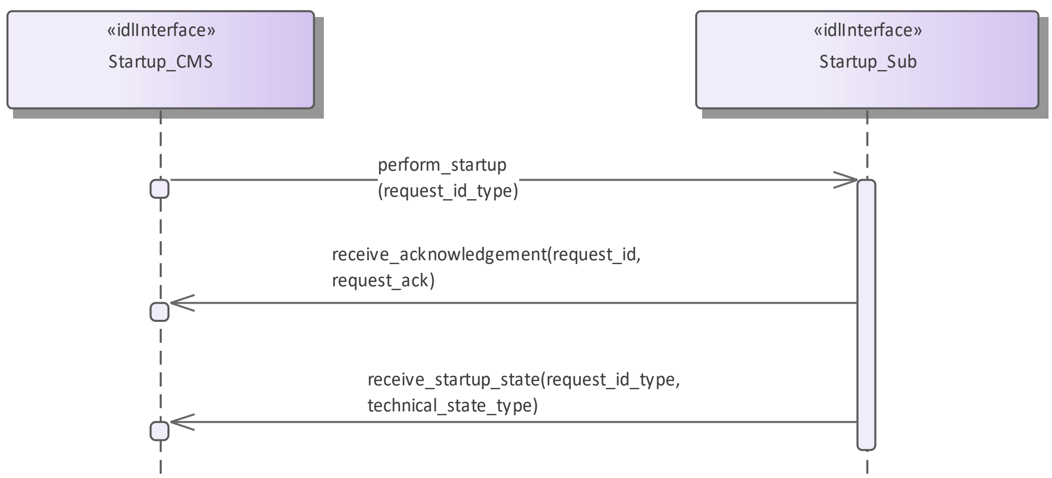


Figure 7.102 Basic Flow -Startup (Interaction diagram)

Basic flow for CMS requesting the subsystem to transition from STANDBY to READY.

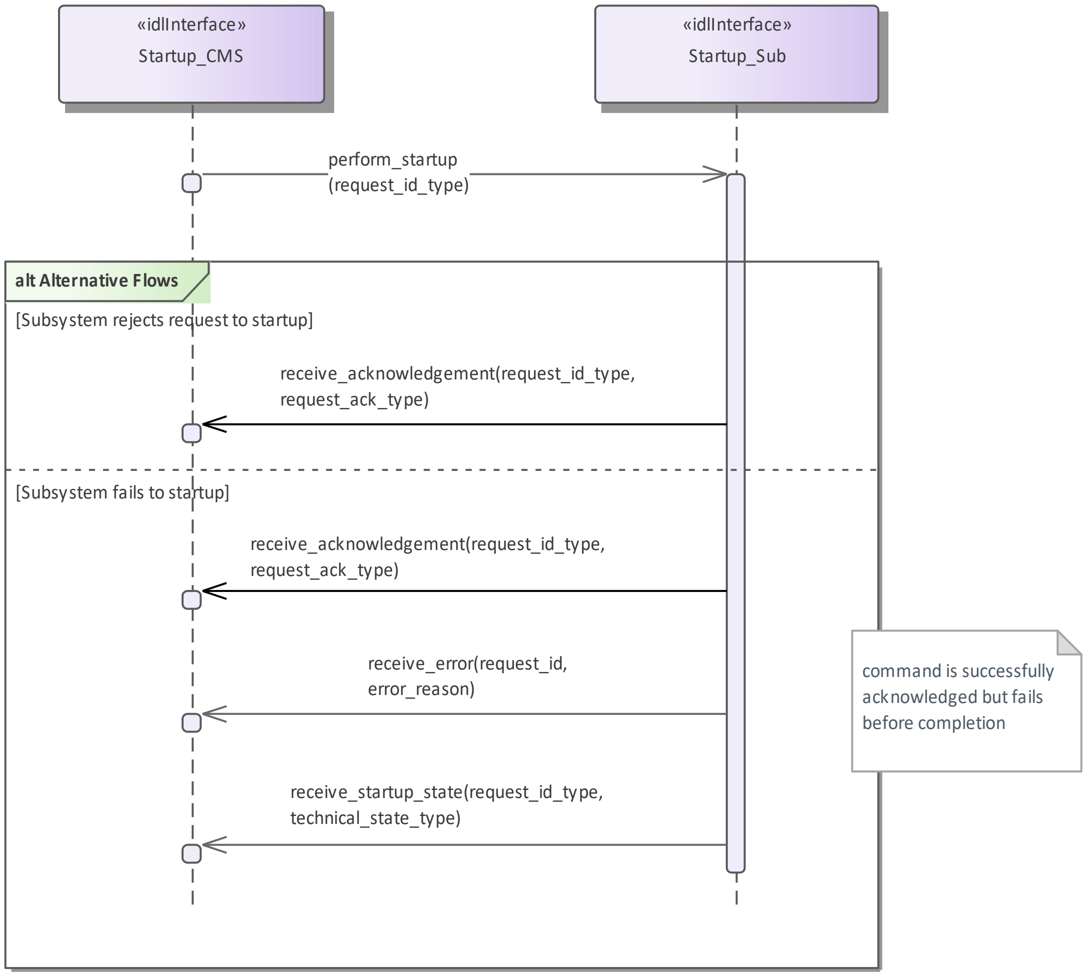


Figure 7.103 Alternative Flow - Startup (Interaction diagram)

Alternate flow for CMS requesting the subsystem to transition from STANDBY to READY (depicts rejection and error paths).

7.8.2.5 Provide_Networking_Statistics

Parent Package: Extended_Subsystem_Control

Contains operations and sequence diagrams for the Provide Bandwidth Statistics interface.

Provide_Networking_Statistics_CMS

Type: Interface

Package: Provide_Networking_Statistics

This is used to inform the CMS of the bandwidth being used by and quality of service achieved by the Subsystem (e.g. an off-platform communications and/or networking device).

Table 7.284 - Methods of Interface Provide_Networking_Statistics_CMS

Method	Notes	Parameters
write_bandwidth_statistics()	Informs the CMS of the most recent bandwidth utilization	percentage_type **peak_utilization** The greatest utlization of bandwidth since the last update percentage_type **mean_utilization** The average utilization of bandwidth since the last update device_name_type **connection** The

		device specific to the connection to which the statistic pertain absolute_duration_type **period_of_validity** The period of time for which the statistic apply
write_latency_statistics()	Informs the CMS of the most recent latency associated with data transfer	time_type **peak_latency** The greatest latency experienced across the network scope since the last update time_type **mean_latency** The average latency experienced across the network scope since the last update device_name_type **connection** The device specific to the connection to which the statistic pertain absolute_duration_type **period_of_validity** The period of time for which the statistics apply

Figure 7.104 Basic Flow - Provide_Networking_Statistics (Interaction diagram)

7.8.2.6 Shutdown

Parent Package: Extended_Subsystem_Control

Contains operations and sequence diagrams for the Shutdown interface.
Shutdown_CMS
Type: Interface
Package: Shutdown
The purpose of this interface is to transition the sub-system to the STANDBY state from any other state as defined by Manage Technical State. Note: this shall cause the Subsystem to cease radiating if it is in an ONLINE state with emissions enabled.
Pre-condition: Subsystem is in ONLINE, READY, FAILED, BIT, or CALIBRATION:
Pre-condition: CMS has mastership of subsystem.:
Post-condition: Sub-system is in STANDBY state if successful, otherwise the current state is reported by the subsystem.:

Table 7.285 - Methods of Interface Shutdown_CMS

Method	Notes	Parameters
receive_shutdown_state()	Interface used by CMS to receive an indication from the subsystem that it has successfully performed shutdown.	request_id_type **request_id** technical_state_type **technical_state**

Shutdown_Sub
Type: Interface
Package: Shutdown

Table 7.286 - Methods of Interface Shutdown_Sub

Method	Notes	Parameters
perform_shutdown()	Interface used by the subsystem to receive a request from the CMS to execute a shutdown.	request_id_type **request_id**

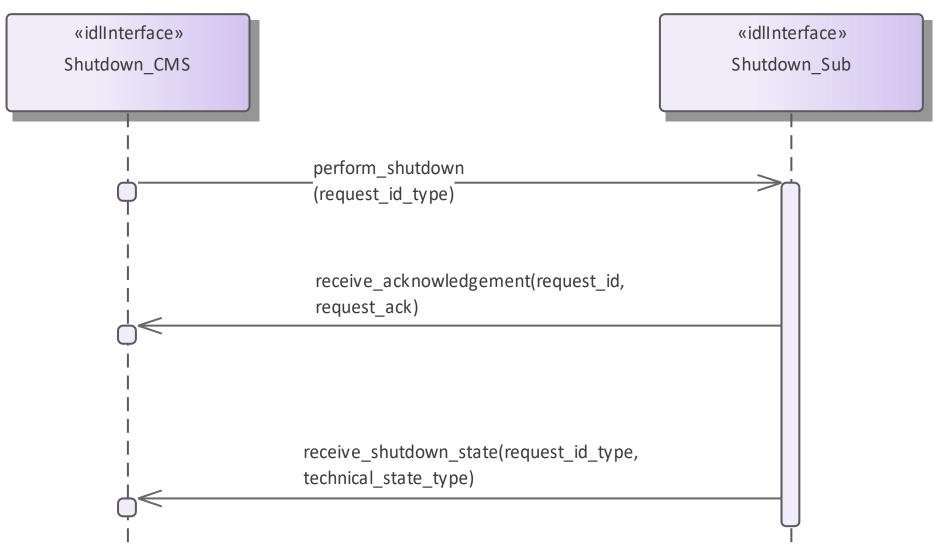


Figure 7.105 Basic Flow - Shutdown (Interaction diagram)

Basic flow for CMS requesting the subsystem to transition to STANDBY.

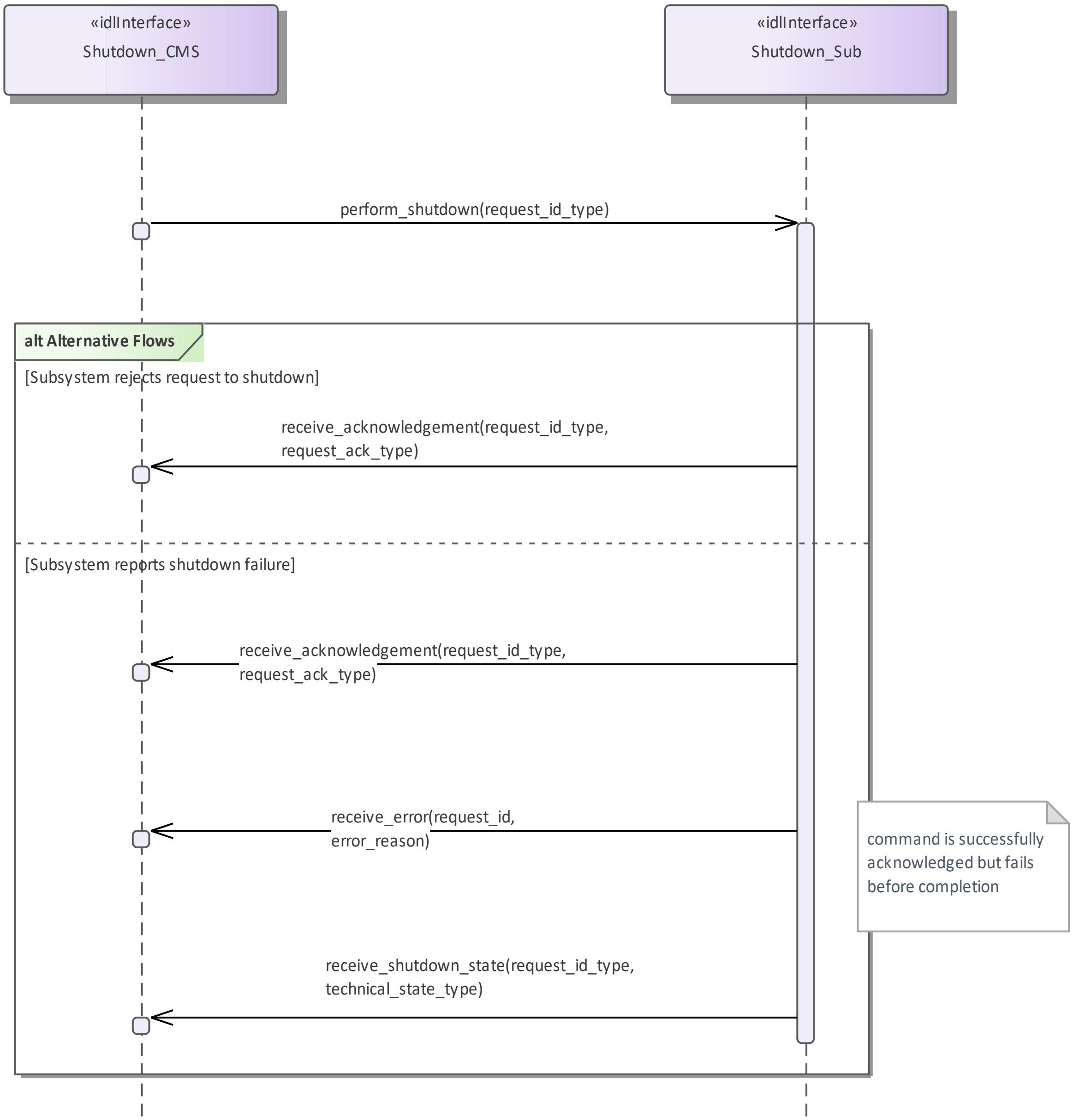


Figure 7.106 Alternative Flow - Shutdown (Interaction diagram)

Alternate flow for CMS requesting the subsystem to transition to STANDBY (depicts rejection and error paths).

7.8.2.7 Restart

Parent Package: Extended_Subsystem_Control

Contains operations and sequence diagrams for the Restart interface.

Restart_CMS

Type: Interface

Package: Restart

The purpose of this interface is to cause a normal transition to STANDBY and then to READY states as defined by Manage Technical State.

Pre-condition: Sub-system is in ONLINE, READY, FAILED, BIT, or CALIBRATION:

Pre-condition: CMS has mastership of sub-system:

Post-condition: Sub-system is in READY state if successful, otherwise current state is reported by subsystem.:

Table 7.287 - Methods of Interface Restart_CMS

Method	Notes	Parameters
receive_restart_state()	Interface used by CMS to receive an indication from the subsystem that it has successfully performed restart.	request_id_type **request_id** technical_state_type **technical_state**

Restart_Sub
Type: Interface
Package: Restart

Table 7.288 - Methods of Interface Restart_Sub

Method	Notes	Parameters
perform_restart()	Interface used by the subsystem to receive a request from the CMS to execute a restart.	request_id_type **request_id**

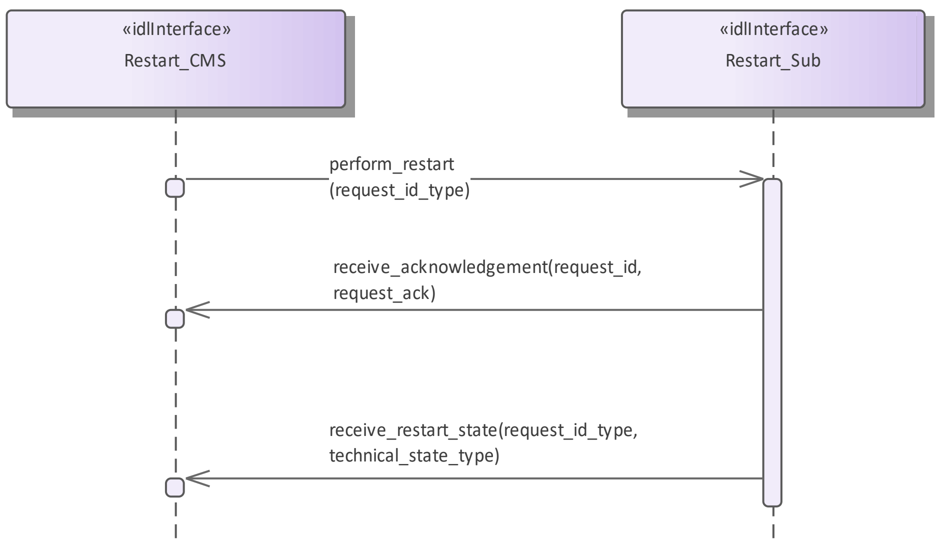


Figure 7.107 Basic Flow - Restart (Interaction diagram)

Basic flow for CMS requesting the subsystem to transition to STANDBY followed by a transition to READY.

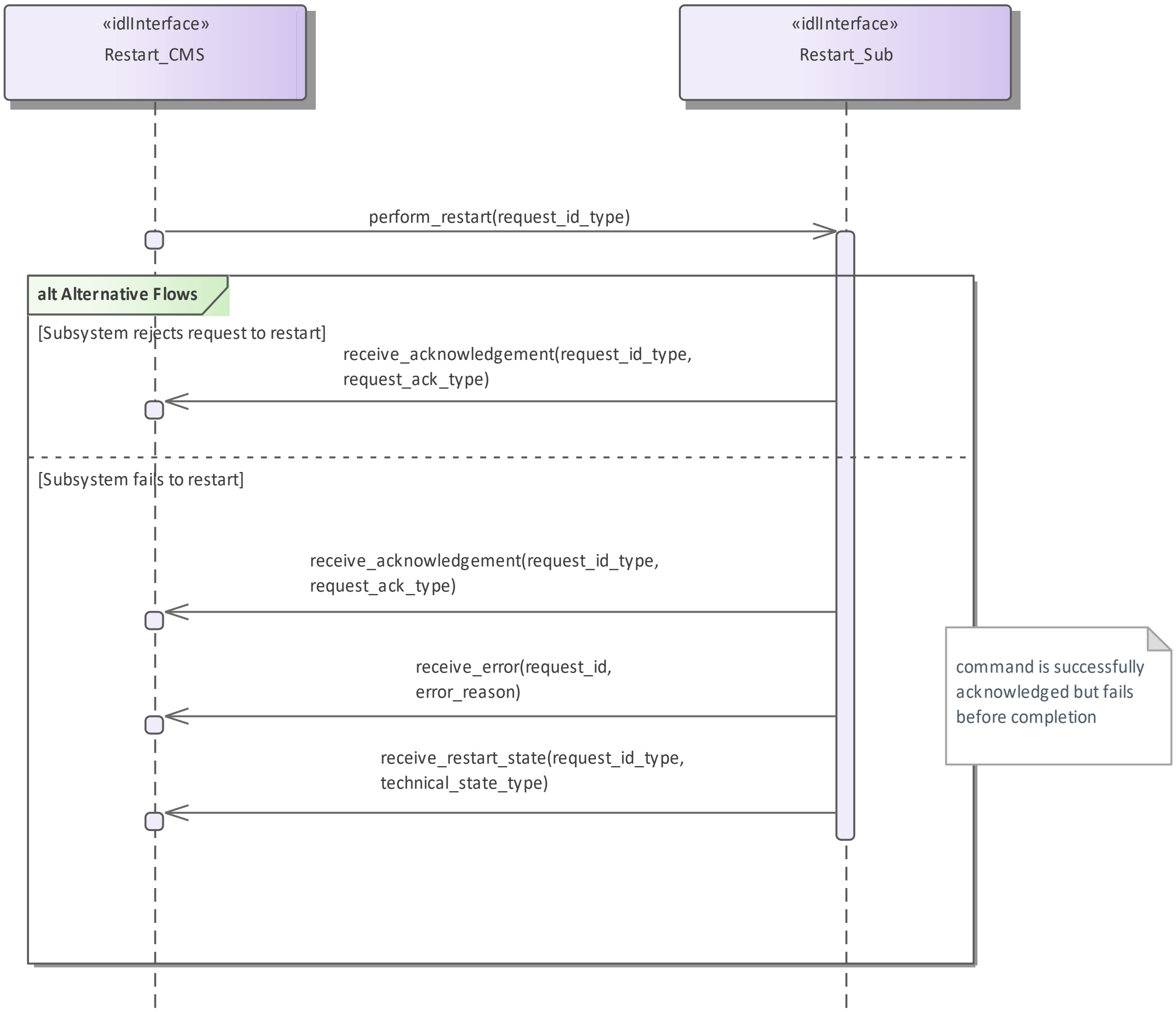


Figure 7.108 Alternative Flow - Restart (Interaction diagram)

Alternate flow for CMS requesting the subsystem to transition to STANDBY followed by a transition to READY (depicts rejection and error paths).

7.8.3 Subsystem_Control

Parent Package: Subsystem_Services
Contains interfaces for the Subsystem Control service.

7.8.3.1 Manage_Technical_State

Parent Package: Subsystem_Control
Contains operations and sequence diagrams for the Manage Technical State interface.
Manage_Technical_State_CMS
Type: Interface
Package: Manage_Technical_State
Manage Technical State causes the subsystem to provide or change its technical state.

Special Requirements:

Initialization: Upon initialization, reset or power-on, the sub-system shall transition to a pre-defined state and report the current state to the CMS.

Additional Information:

If a critical component of the subsystem becomes NOT AVAILABLE, the technical state shall transition to FAILED.

All states may transition to OFFLINE, but the subsystem shall only do so in emergency situations or catastrophic damage, to indicate an uncontrolled shutdown

Startup, *Shutdown*, and *Restart* explain the sequence of actions for nominal progression through the technical states.

Pre-condition: If the CMS requests a Technical State to change, mastership of the subsystem is required.:

Pre-condition: CMS is aware of the current subsystem state.:

Pre-condition: CMS is aware of the possible technical states supported by the subsystem.:

Post-condition: None.:

Table 7.289 - Methods of Interface Manage_Technical_State_CMS

Method	Notes	Parameters
receive_periodic_technical_state()	Interface used by CMS to receive periodic technical state reports from the subsystem.	technical_state_type **technical_state**
receive_technical_state()	Interface used by CMS to receive technical state reports from the subsystem which were the result of a transition request from the CMS.	request_id_type **request_id** technical_state_type **technical_state**

Manage_Technical_State_Sub

Type: Interface

Package: Manage_Technical_State

Table 7.290 - Methods of Interface Manage_Technical_State_Sub

Method	Notes	Parameters
change_technical_state()	Interface used by the subsystem to receive requests from the CMS to change its technical state.	request_id_type **request_id** technical_state_type **technical_state**
provide_technical_state()	Interface used by the subsystem to receive requests from the CMS to provide its current technical state.	request_id_type **request_id**

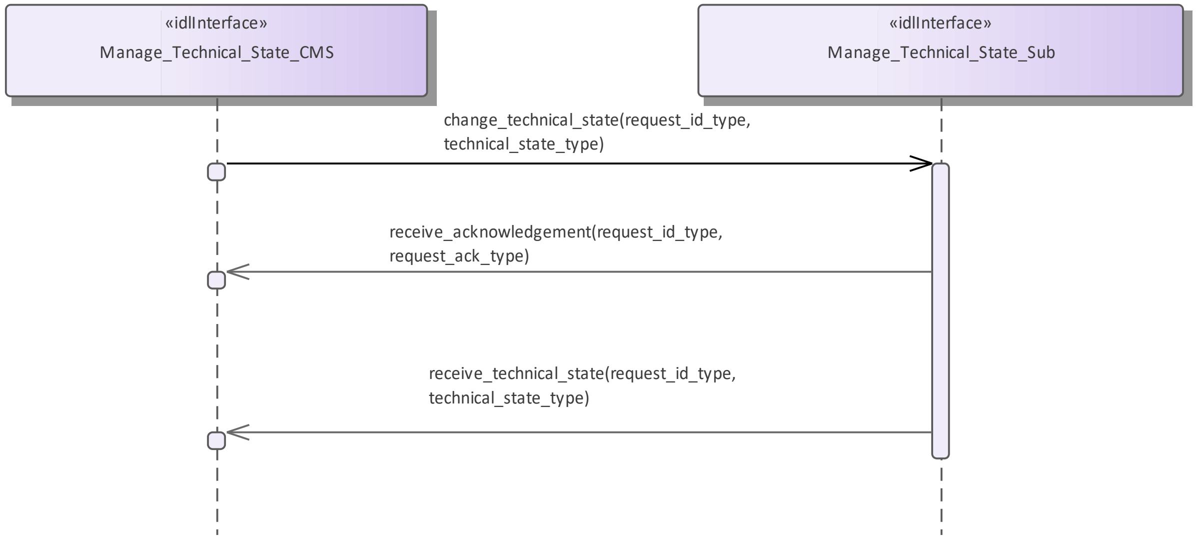


Figure 7.109 Basic Flow - Manage Technical State - Change (Interaction diagram)

Flow of events which depicts the CMS requesting that the subsystem changing its current technical state.

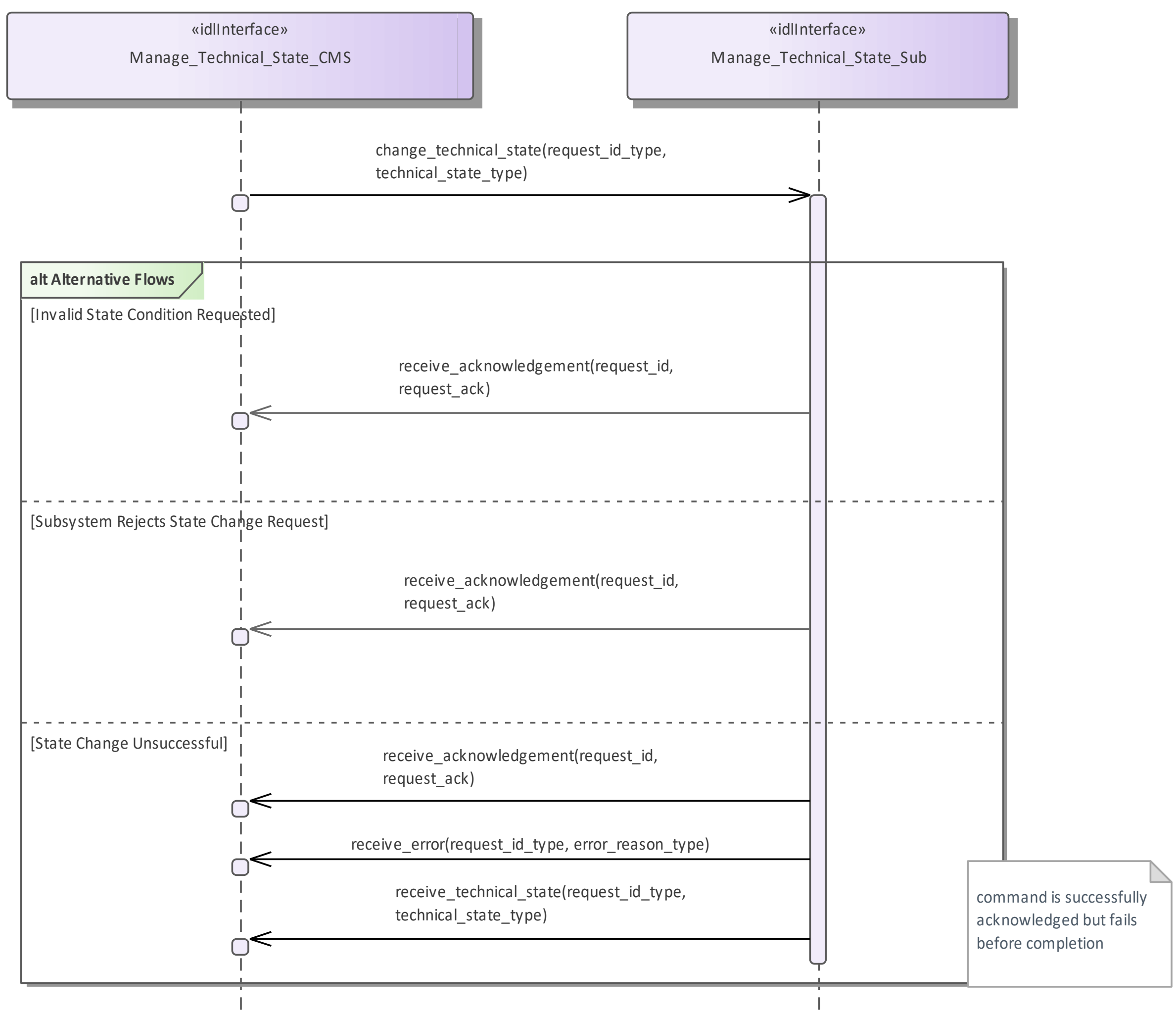


Figure 7.110 Alternative Flow - Manage Technical State - Change (Interaction diagram)

Alternate flow depicting rejection and error cases for a CMS requesting the subsystem to change its Technical State.

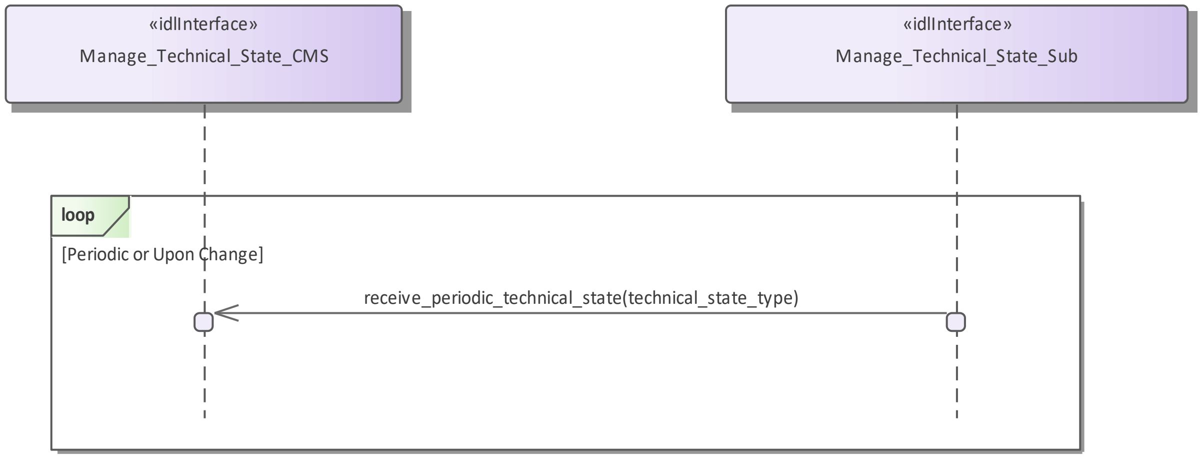


Figure 7.111 Basic Flow - Manage Technical State - Periodic Reporting (Interaction diagram)

Flow of events which depicts a subsystem that periodically reports its technical state (without the need for a CMS request).

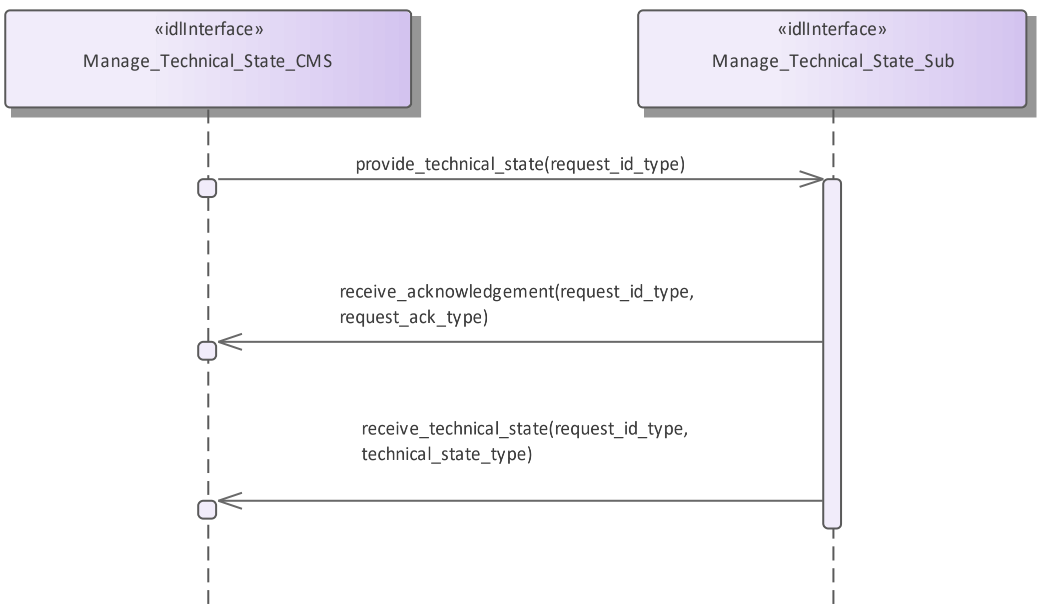


Figure 7.112 Basic Flow - Manage Technical State - Request (Interaction diagram)

Flow of events which depicts the CMS requesting that the subsystem report on its current technical state.

7.8.3.2 Heartbeat_Signal

Parent Package: Subsystem_Control

Heartbeat_Signal_CMS

Type: Interface

Package: Heartbeat_Signal

The service describes how the availability of an established communication between CMS and the subsystem as well as the subsystem itself shall be monitored. The heartbeat signal is triggered by Control Interface Connection. The basic flow is asynchronous.

The actor is the Combat Management System.

Pre-condition: Connection established: Provide Subsystem Services has successfully established communication between CMS and the subsystem.

Post-condition: Interface is alive: The heartbeat has been received successful.

Post-condition: Interface is not alive: The heartbeat has not been received.

Table 7.291 - Methods of Interface Heartbeat_Signal_CMS

Method	Notes	Parameters
receive_subsystem_heartbeat_signal( )	Receive the periodic heartbeat signal to verify, that the connection is still alive.	unsigned long **count** This parameter is used with implementation specific semantics for monitoring interface participant liveliness.

Heartbeat_Signal_Sub

Type: Interface

Package: Heartbeat_Signal

Table 7.292 - Methods of Interface Heartbeat_Signal_Sub

Method	Notes	Parameters
receive_cms_heartbeat_signal()	Receive the periodic heartbeat signal to verify, that the connection is still alive.	unsigned long **count** This parameter is used with implementation specific semantics for monitoring interface participant liveliness.

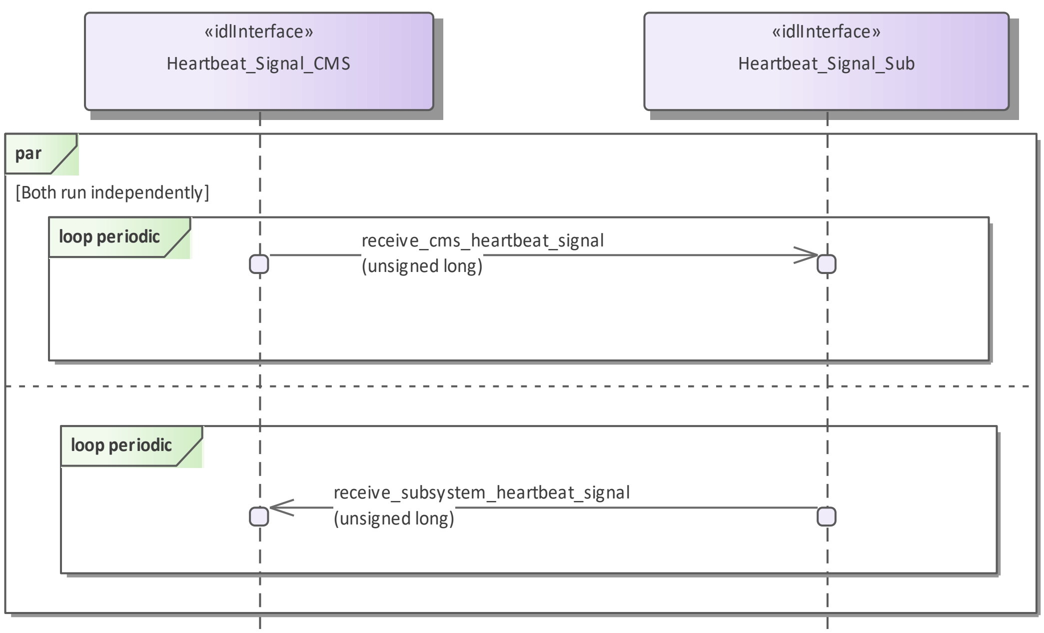


Figure 7.113 Basic Flow - Heartbeat Signal (Interaction diagram)

7.8.3.3 Provide_Subsystem_Identification

Parent Package: Subsystem_Control

Provide_Subsystem_Identification_CMS
Type: Interface
Package: Provide_Subsystem_Identification
In order to enable two interface partners to connect to each other and to open mutual communication, one partner shall initiate and the other to answer. The intention is to let the subsystem initiate the communication. Consequently, the subsystem introduces itself to the CMS identifying e.g. the type of subsystem, the product and its version. That allows the CMS to decide whether it may work with that subsystem.

The actor is the Combat Management System.

The possibility that CMS and subsystem are connected without being capable to work with each other is a consequence of a plug-&-play concept.
Although the interface is standardized the CMS may need a setup process to prepare it for a subsystem. This process shall introduce the information necessary to configure functions of that particular CMS with respect to the subsystem.
This may also be necessary on side of the subsystem.
The preparation for a subsystem may be done by means of system configuration data which are implemented on installation of the combat system. It does not address security information.
Pre-condition: CMS and Subsystem can communicate with each other.:
Post-condition: CMS and subsystem may work together.: CMS and subsystem have verified that they may work with each other.
They shall do some organization regarding the communication (out of scope).
Post-condition: CMS and subsystem may not work together.: The interface between CMS and subsystem is closed.

Table 7.293 - Methods of Interface Provide_Subsystem_Identification_CMS

Method	Notes	Parameters
receive_sub_identification_data()	Receive the identification data from the subsystem.	device_identification_type **identification** request_id_type **the_request_id**

Provide_Subsystem_Identification_Sub
Type: Interface
Package: Provide_Subsystem_Identification

Table 7.294 - Methods of Interface Provide_Subsystem_Identification_Sub

Method	Notes	Parameters
receive_cms_identification_data()	Receive the identification data from the CMS.	device_identification_type **identification** request_id_type **the_request_id**

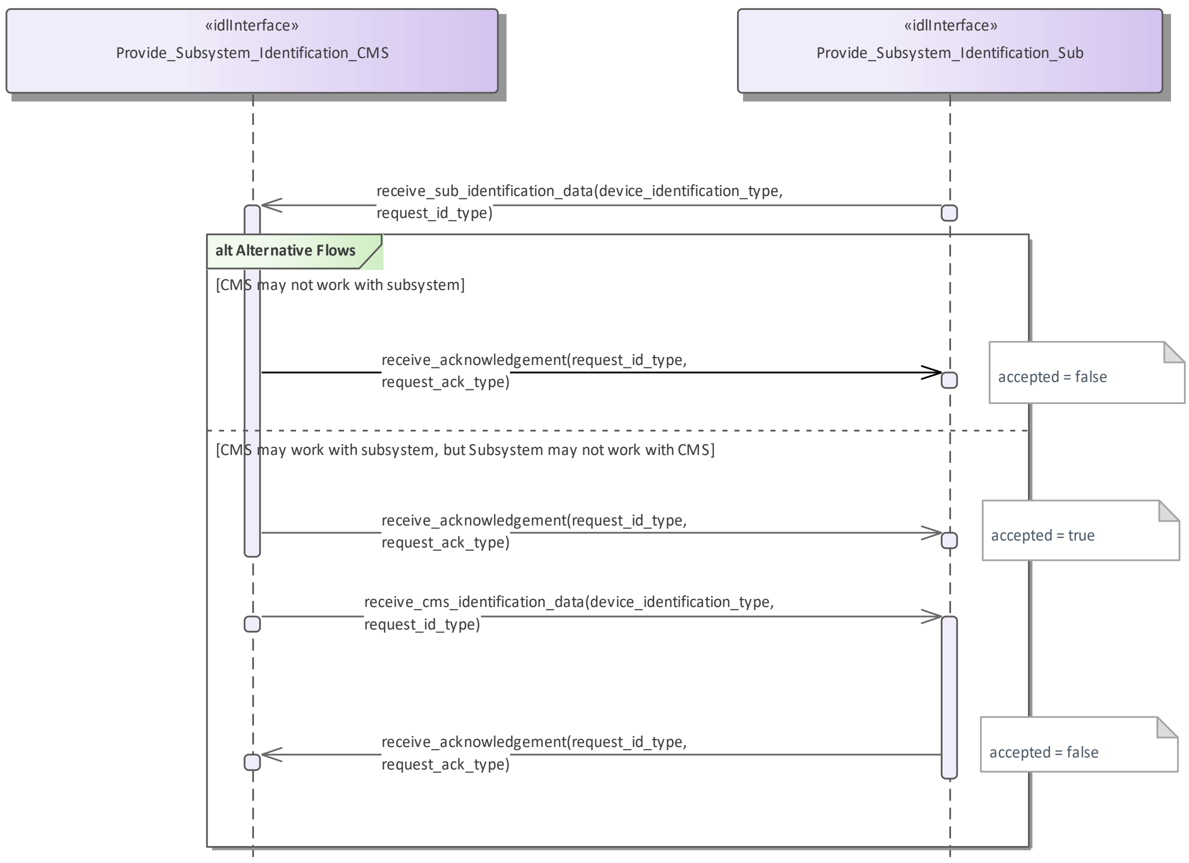


Figure 7.114 Alternative Flow - Introduction of subsystems (Interaction diagram)

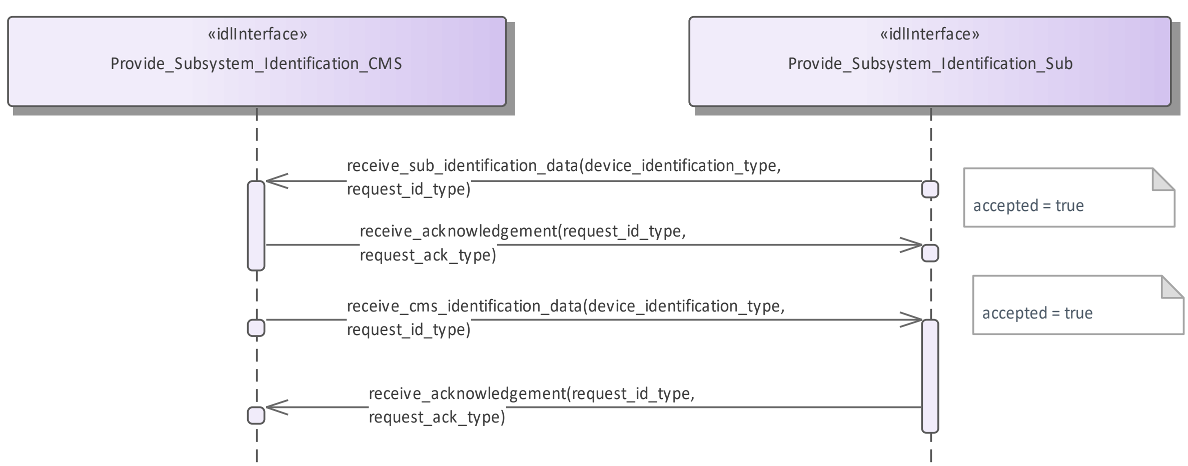


Figure 7.115 Basic Flow - Introduction of the subsystem (Interaction diagram)

7.8.3.4 Provide_Health_State

Parent Package: Subsystem_Control

Provide_Health_State_CMS
Type: Interface
Package: Provide_Health_State
The service allows the CMS to monitor and evaluate the health state of the subsystem. The health state information describes functional availability of the subsystem and the services it provides.

The service may be triggered by several possible situations:

- Periodic event, for example by internal clock,
- Actor (CMS) request,
- Health state change,
- Initialization (start-up),
- Recovery of the subsystem after a failure.

In addition to the health state being provided, additional information may be provided to the CMS. In case of a service, the information may include a list of detected faults. In case of a subsystem, the information may include the list of services together with their health state, and for every service which has health state other than AVAILABLE, a list of detected faults. This two dimensional structure is called the service availability matrix.

The state NOT AVAILABLE may also describe the situation in which the service is not implemented. In this case the list of faults shall be empty. In the state UNKNOWN, the subsystem may provide the reason for not being able to evaluate health state (e.g. BIT process not running).

The service ends with success when the health state (possibly accompanied by additional information) is provided to the actor.

Relationship to technical state.
The reported health state of the services is dependent on the technical state.
In the technical state ONLINE, the health state of the services is determined based on the detected faults (if any).
In all technical states other than ONLINE (except OFFLINE), the health state of all services, except the service Subsystem_Control, is NOT AVAILABLE.
The health state of the service Subsystem_Control shall then be DEGRADED, since some functions (e.g. Control Battle Override) are not available in those technical states, and some functions are (e.g. Manage Technical State).
In the technical state OFFLINE no communication at all is possible with the CMS so the health state is not reported.

Relationship to battle override.
When Battle Override is set (see service Control Battle Override), certain faults are not taken into account when determining the health state. These overridable faults generally refer to circumstances that may cause damage to own equipments, but do not prohibit executing the requested task.

Relationship to simulation mode.
If the subsystem is in Simulation mode (technical state is ONLINE), only the faults for parts needed for the simulated execution of the service are taken into account when determining the health state of a service.
For instance, if the transmitter is defective, the service Track_Reporting is reported AVAILABLE when in Simulation mode, but is reported NOT AVAILABLE when not in Simulation mode.
Faults may also be simulated for training purposes (see service Define Fault Script). Therefore, irrespective of the Simulation mode, all faults (real and simulated) are included in the reported list of detected faults, each with an indication whether the fault is real or simulated.
If a real system part is simulated, faults of the simulated part should have a different identification.
For instance (see previous example) in Simulation mode, a simulated transmitter could be used, for which the trainer has inserted a simulated fault.
Any faults in the real transmitter would be reported (real fault) as well as the injected fault in the simulated transmitter (simulated fault). However, the health state of the service Track_Reporting would be based only on the status of the simulated transmitter.

Reason for health state

Each reported health state other than AVAILABLE is accompanied by the reason(s) for that health. In this way the CMS may for instance derive that although the technical state of the subsystem is STANDBY (and NOT AVAILABLE for that reason), there are also faults that would prevent the service to become AVAILABLE when the technical state would be switched to ONLINE.

Pre-condition: Subsystem technical state: The subsystem is in technical state ONLINE, STANDBY or READY.

Post-condition: CMS awareness: CMS is aware of the health state of the subsystem and/or its services.

Table 7.295 - Methods of Interface Provide_Health_State_CMS

Method	Notes	Parameters
report_fault()	Report a fault to CMS	fault_type **the_fault**
report_service_health()	Report health of service	request_id_type **request_id** service_health_type **health** fault_list_type **the_fault_list**
report_subsystem_health()	Report health of subsystem	request_id_type **request_id** subsystem_health_type **health**
provide_service_health()	Report health of service (on subsystem initiative)	service_health_type **health** fault_list_type **the_fault_list**
provide_subsystem_health()	Report health of subsystem (on subsystem initiative)	subsystem_health_type **health**

Provide_Health_State_Sub

Type: Interface

Package: Provide_Health_State

Table 7.296 - Methods of Interface Provide_Health_State_Sub

Method	Notes	Parameters
request_service_health()	Request service health	request_id_type **request_id** service_name_type **service_name**
request_subsystem_health()	Request subsystem health	request_id_type **request_id**

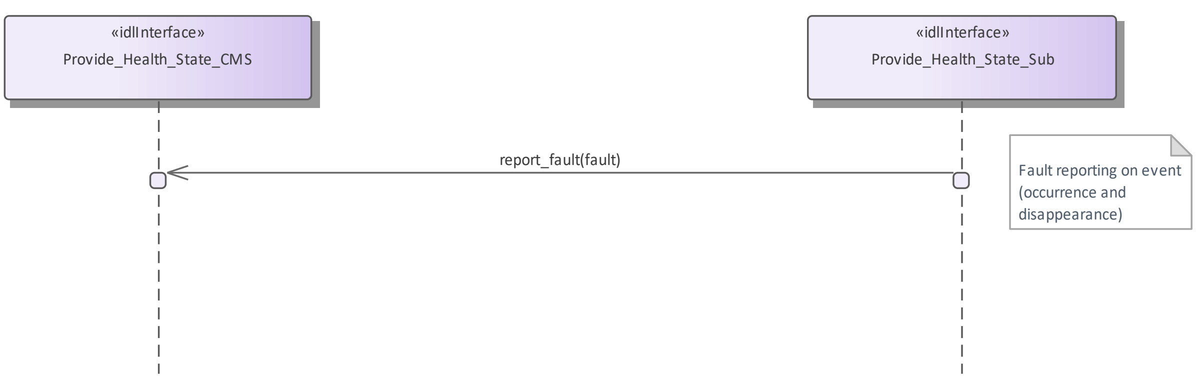


Figure 7.116 Basic Flow - Fault Reporting (Interaction diagram)

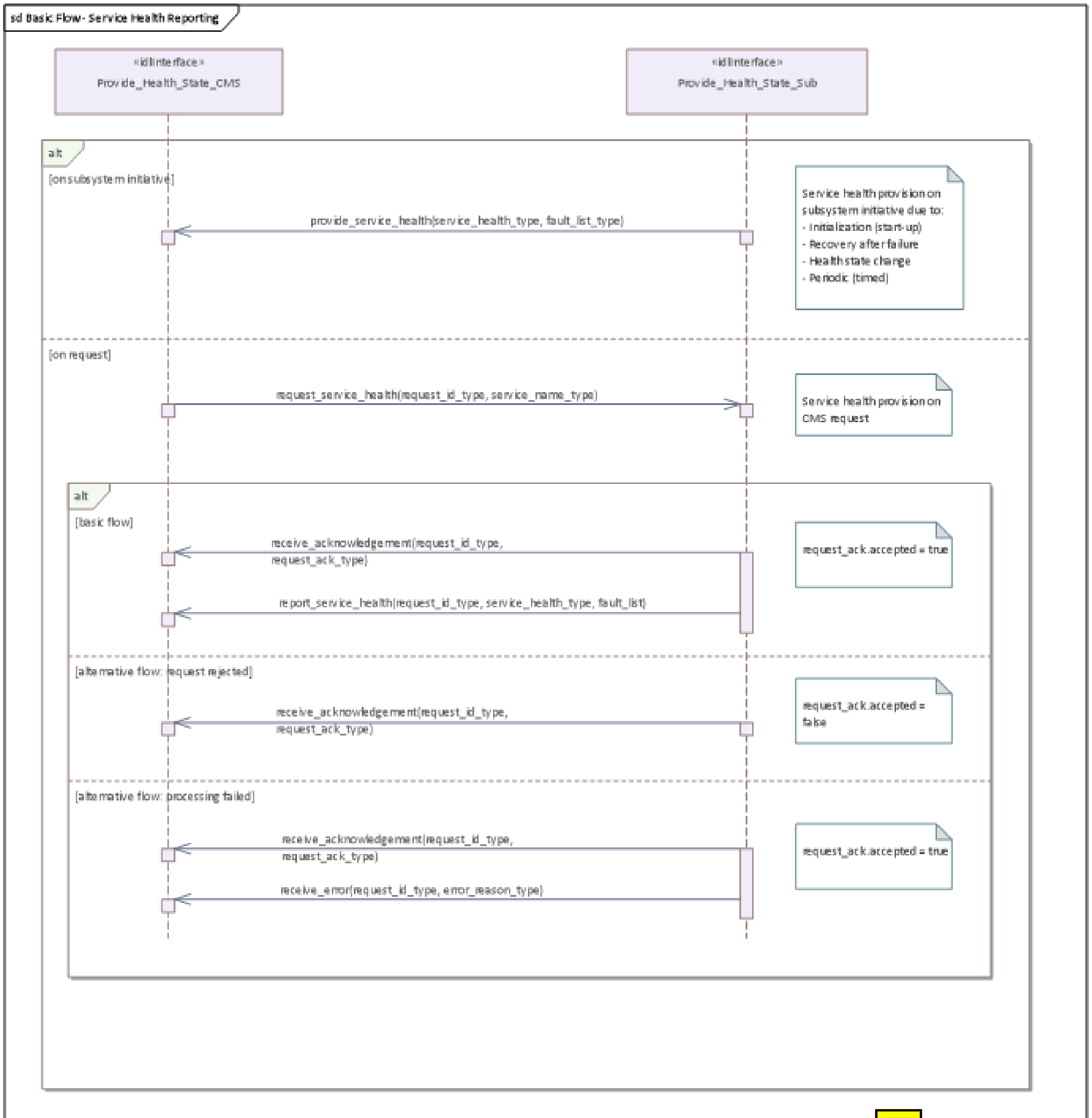


Figure 7.117 Basic Flow - Service Health Reporting (Interaction diagram)

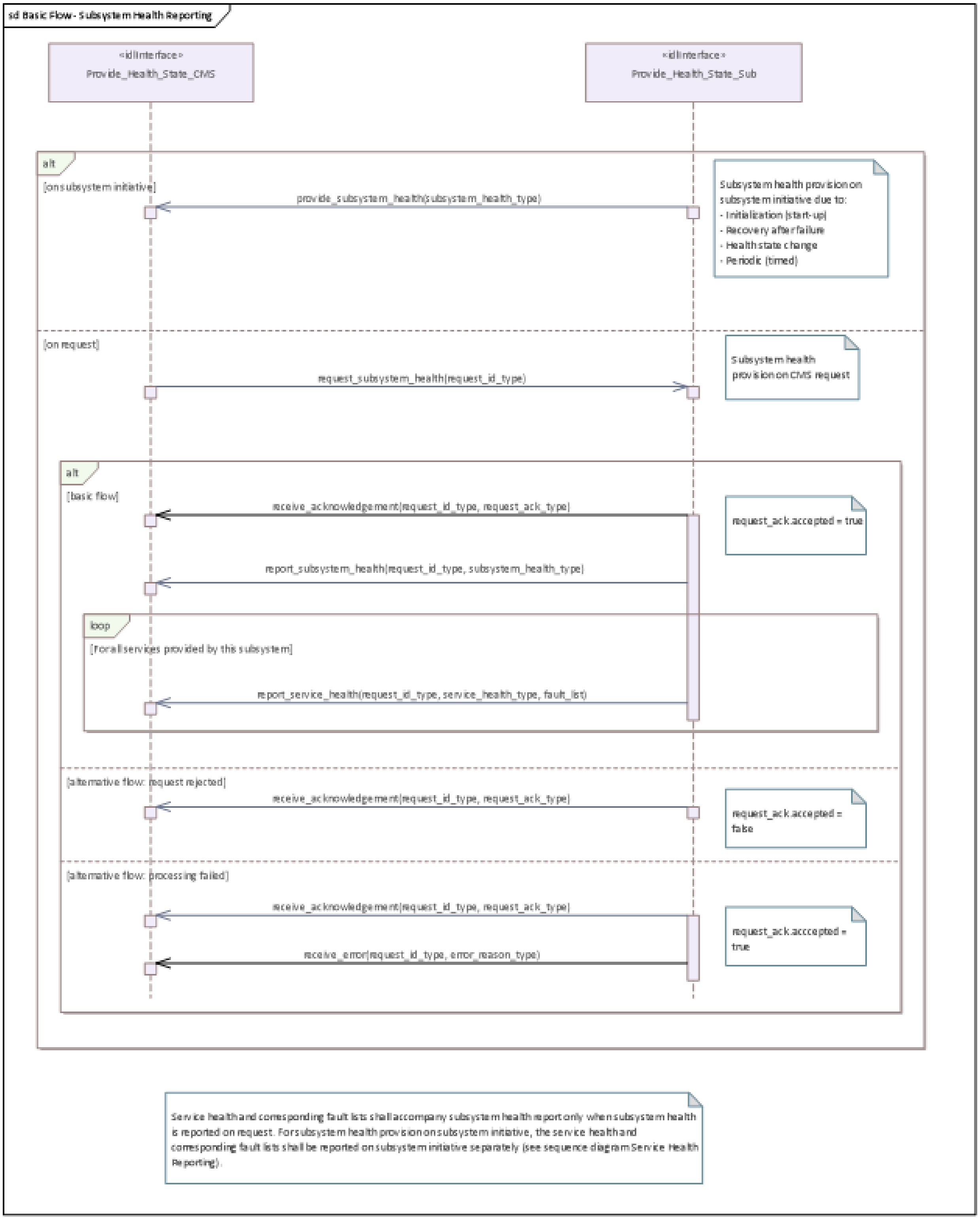


Figure 7.118 Basic Flow - Subsystem Health Reporting (Interaction diagram)

7.8.3.5 Manage_Operational_Mode

Parent Package: Subsystem_Control

Manage_Operational_Mode_CMS
Type: Interface
Package: Manage_Operational_Mode
Subsystems provide several operational modes like long-range-detection, missile-detection, surface surveillance etc. in case of surveillance radar, normal tracking, slaved, joystick controlled in case of fire control radar etc.

Operational modes summarise a set of subsystem parameters optimising the subsystem with respect to an operational purpose.

The names of modes of a specific type of subsystem (e.g. or a radar) differ from supplier to supplier. Consequently, they shall be handled as configuration parameters. They shall be offered to the operator to enable him for a selection and shall be transferred to the subsystem to achieve the intended reaction.

The definition of names of operational modes is not within the scope of this standard.

It is the CMS's responsibility to initiate the determination of initial state by making a request for information to the subsystem.

In the case where the CMS does not have mastership of the subsystem, a change of the operational mode shall be indicated by informing the CMS about the new operational mode (see service "Provide health state").

Configuration data like the set of available operational modes may be received at runtime but may also be inserted by means of an automatic or manual setup process. Although automatic runtime transfer of such information may be achieved through ‘Manage Subsystem Parameters’ it is not a mandatory requirement of this standard for that mechanism to be used.
Pre-condition: Technical state READY or ONLINE.:
Pre-condition: "Manage Subsystem Parameters" executed successfully:
Pre-condition: CMS must have Mastership:
Post-condition: Service ends with success: - the subsystem is in the commanded operational state, the CMS is informed that this is the case
Post-condition: Service ends with fail: - the subsystem is still in the original operational state, the CMS has the correct information regarding that state.

Table 7.297 - Methods of Interface Manage_Operational_Mode_CMS

Method	Notes	Parameters
report_operational_mode()	The current operational mode is reported via this interface method.	request_id_type **request_id** operational_mode_type **current_mode**
configure_operational_functions()		function_type **function** A function operated by the subsystem

Manage_Operational_Mode_Sub
Type: Interface
Package: Manage_Operational_Mode

Table 7.298 - Methods of Interface Manage_Operational_Mode_Sub

Method	Notes	Parameters
request_get_operational_mode()	The subsystem is requested to report the current operational mode.	request_id_type **request_id**

request_set_operational_mode()	The subsystem is requested to change the operational mode to the given new operational mode.	request_id_type **request_id** operational_mode_type **new_operational_mode**

Manage_Operational_Mode_CMS
Type: ActivityPartition
Package: Manage_Operational_Mode

Manage_Operational_Mode_Sub
Type: ActivityPartition
Package: Manage_Operational_Mode

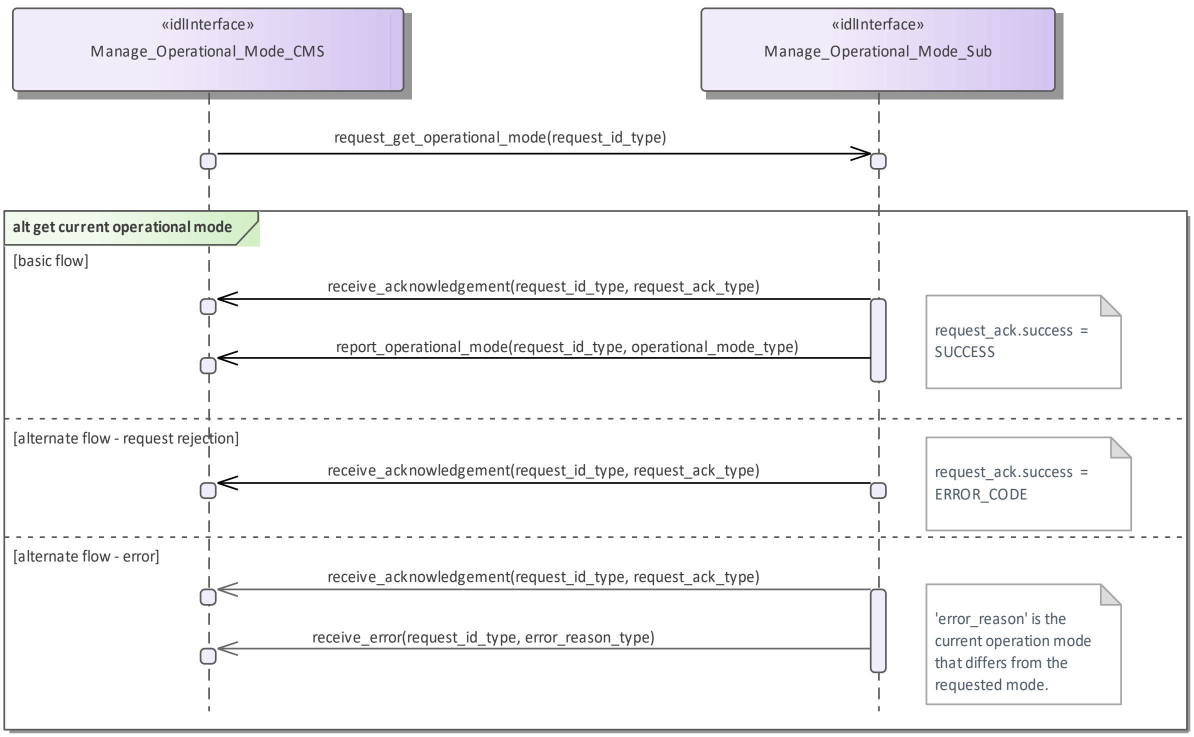


Figure 7.119 Manage Operational Mode - get current operational mode (Interaction diagram)

This sequence diagram shows how the CMS and the subsystem operate with each other during the operation "get current operational mode" of the service "Manage Operational Mode".

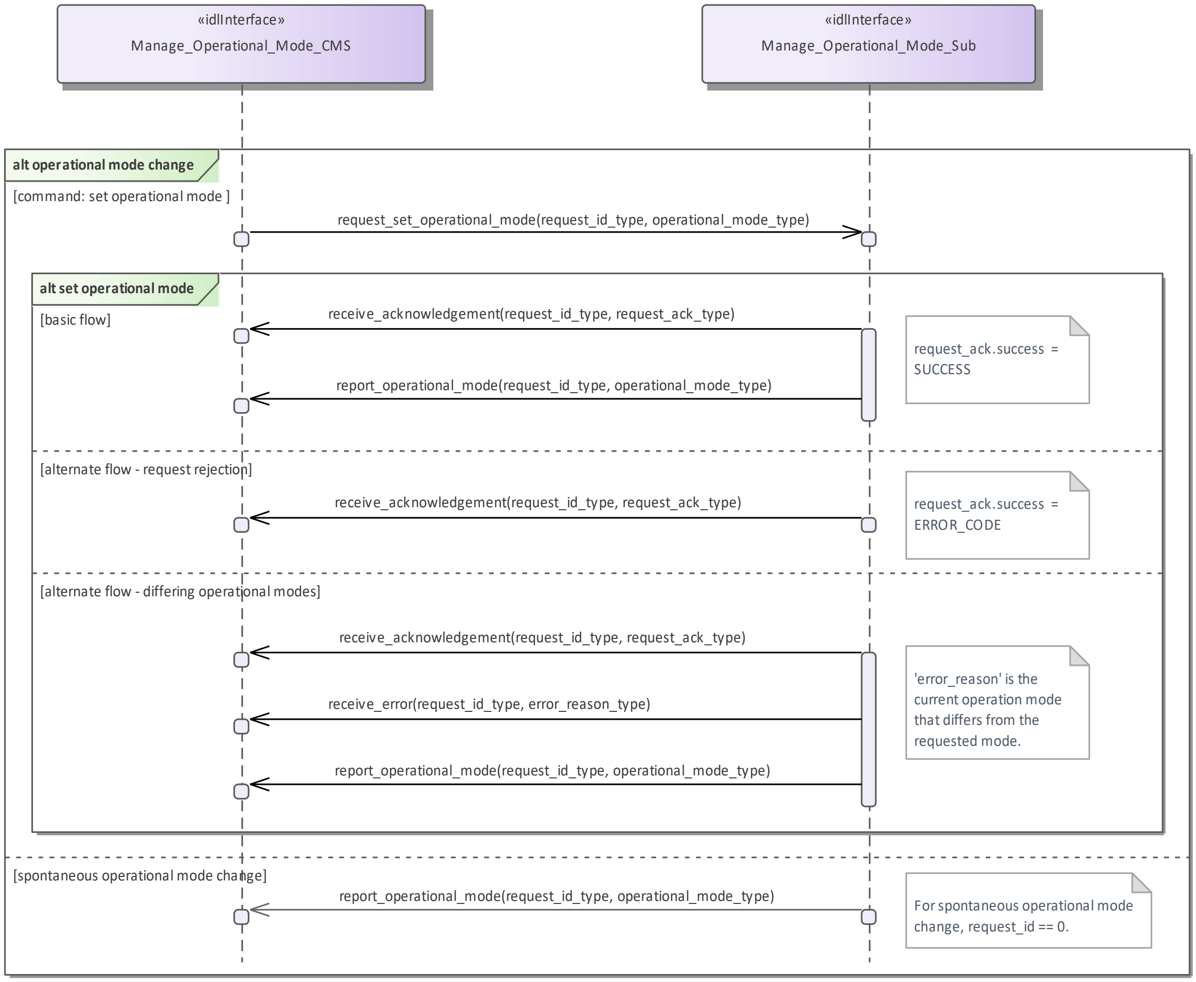


Figure 7.120 Manage Operational Mode - set operational mode (Interaction diagram)

This sequence diagram shows how the CMS and the subsystem operate with each other during the operation "set operational mode" of the service "Manage Operational Mode".

7.8.3.6 Control_Battle_Override

Parent Package: Subsystem_Control

This package contains interfaces for the Control Battle Override service.

Control_Battle_Override_CMS

Type: Interface

Package: Control_Battle_Override

The subsystem is requested to set/reset the Battle Override. When Battle Override is set the subsystem disregards warnings on circumstances which may cause damage to own equipment, typically the overtemperature protections.

It is the CMS's responsibility to initiate the determination of initial state by making a request for information to the subsystem.

Provision of the Battle Override state

Subsystem shall keep CMS informed about the current Battle Override state and its changes (if any).

Lack of mastership

In the case where CMS does not have mastership of the subsystem, CMS shall be informed about the current Battle Override state and its changes (if any).

Relationship to the subsystem health state
As long as the Battle Override is set, the subsystem internal overtemperature indications shall not result in any heath state set to "NOT AVAILABLE" (see *Provide health state*).

Pre-condition: Mastership Required: CMS has mastership of the subsystem

Pre-condition: Subsystem Services: *Provide subsystem services* has been completed successfully.

Post-condition: Success: The subsystem Battle Override is set/reset as requested and CMS is informed that this is the case.

Post-condition: No Success: The subsystem Battle Override is still equal to the original one and CMS has the correct information regarding that state.

Table 7.299 - Methods of Interface Control_Battle_Override_CMS

Method	Notes	Parameters
battle_override_setting()	This metod is used by the subsystem to return the current Battle Override state.	request_id_type **request_id** battle_override_state_type **battle_override_state**

Control_Battle_Override_Sub
Type: Interface
Package: Control_Battle_Override

Table 7.300 - Methods of Interface Control_Battle_Override_Sub

Method	Notes	Parameters
set_battle_override()	This method is used by the CMS to send a Battle Override set/reset request to the subsystem,	request_id_type **request_id** battle_override_state_type **battle_override_state**

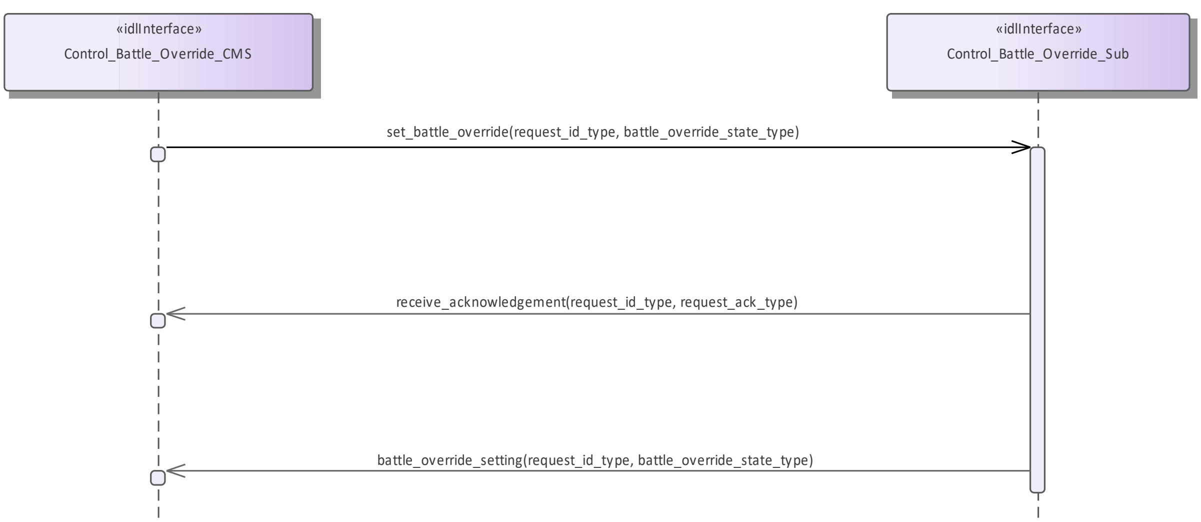


Figure 7.121 Basic Flow - Control Battle Override - Set/Reset (Interaction diagram)

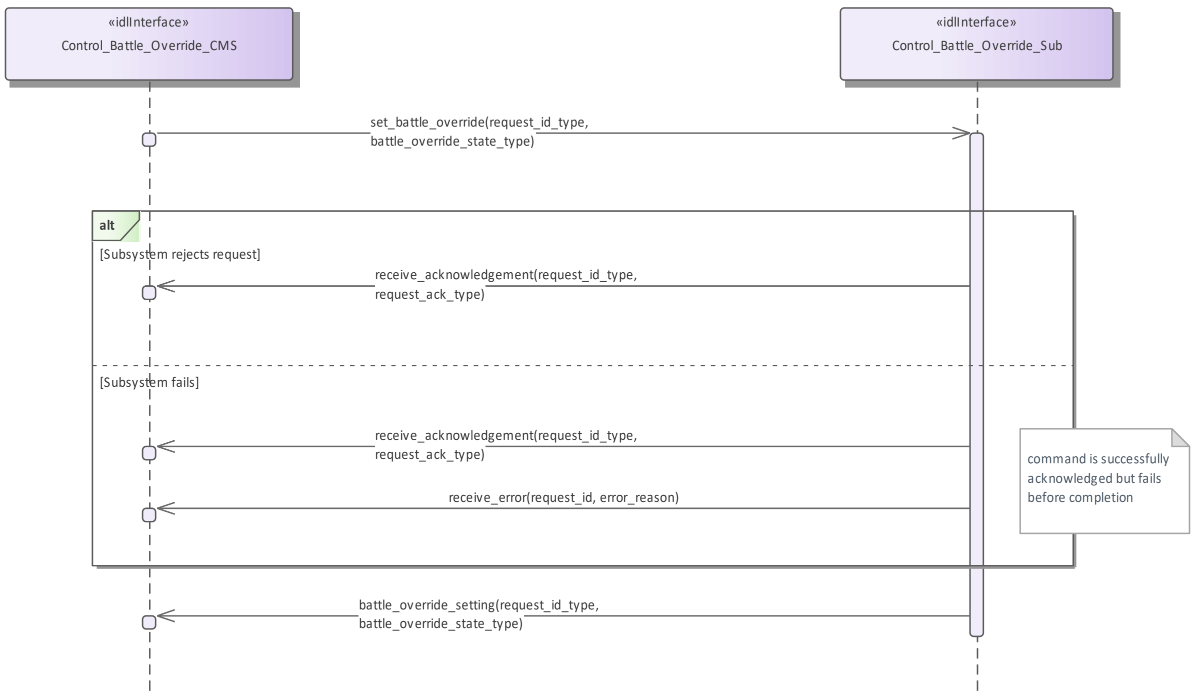


Figure 7.122 Alternative Flow - Control Battle Override - Set/Reset - loss of mastership (Interaction diagram)

7.8.3.7 Manage_Subsystem_Parameters

Parent Package: Subsystem_Control

Manage_Subsystem_Parameters_CMS
Type: Interface
Package: Manage_Subsystem_Parameters
The service allows the actor to obtain and modify the values of parameters of the subsystem. It also provides the facilities to retrieve the descriptions of parameters available in a certain subsystem.

The actor of the service is the Combat Management System.

The service starts when the CMS requests one of the following:

- Parameter value retrieval
- Parameter value modification
- Retrieval of parameter descriptor,

with a list of parameter names (and values in case of modification).
A parameter value may be structured (e.g. a vector or a table).
The service ends when the subsystem has provided the requested information or modified the parameter value.

It is the CMS's responsibility to initiate the determination of initial state by making a request for information to the subsystem.

Parameter names used by a subsystem are to be unique within the scope of that subsystem. Requests for parameter descriptions and to get and set current values are consequently well-defined. Parameter names may be structured using a namespace scheme to promote uniqueness.

Unknown parameter

On receipt of a request for parameter value retrieval, parameter value modification or parameter descriptor retrieval for an unknown parameter name, the subsystem responds with an indication “unknown parameter”. Other (correctly identified) parameters in the same request are processed as requested.

Illegal parameter value
On receipt of a request for parameter value modification with a parameter value that is outside the allowable range of the specified parameter, the subsystem responds with an indication “illegal parameter value” and does not change the parameter value.
This includes inconsistencies of parameter type (e.g. real where integer is expected) and structure (e.g. vector of 2 elements, where a vector of 3 is expected).
Other parameters with legal values in the same request are modified as requested.
In case of an illegal value for an element of a structured parameter, the entire parameter remains unchanged.

Modification of parameter value
A parameter value may only be modified in the technical state(s) as specified in the descriptor of that parameter.

Security
Access to the service may be restricted to certain parts of the CMS because of security restrictions.

Pre-condition: Subsystem technical state: The subsystem is in a technical state other than OFFLINE.

Pre-condition: Mastership: The CMS has mastership of the subsystem in case of parameter value modification.

Table 7.301 - Methods of Interface Manage_Subsystem_Parameters_CMS

Method	Notes	Parameters
report_parameter_values()		request_id_type **request_id** name_value_sequence_type **the_name_value_set** name_error_sequence_type **the_name_error_set**
report_parameter_descriptors()		request_id_type **request_id** descriptor_sequence_type **the_descriptor_sequence** name_error_sequence_type **the_name_error_set**

Manage_Subsystem_Parameters_Sub
Type: Interface
Package: Manage_Subsystem_Parameters

Table 7.302 - Methods of Interface Manage_Subsystem_Parameters_Sub

Method	Notes	Parameters
retrieve_parameter_values()		request_id_type **request_id** parameter_name_sequence_type **the_name_set**
modify_parameter_values()		request_id_type **request_id** name_value_sequence_type **the_name_value_set**
retrieve_parameter_descriptors()		request_id_type **request_id** parameter_name_sequence_type **the_name_set**

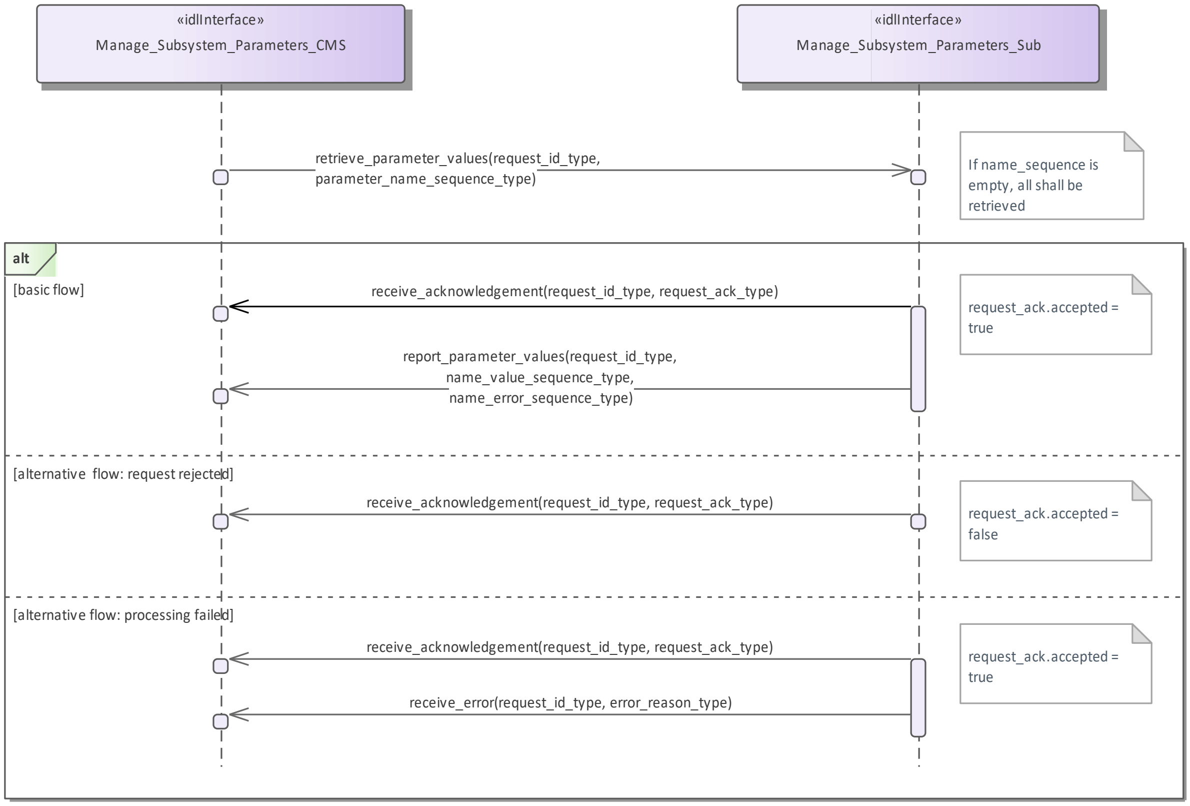


Figure 7.123 Basic Flow - Parameter Retrieval (Interaction diagram)

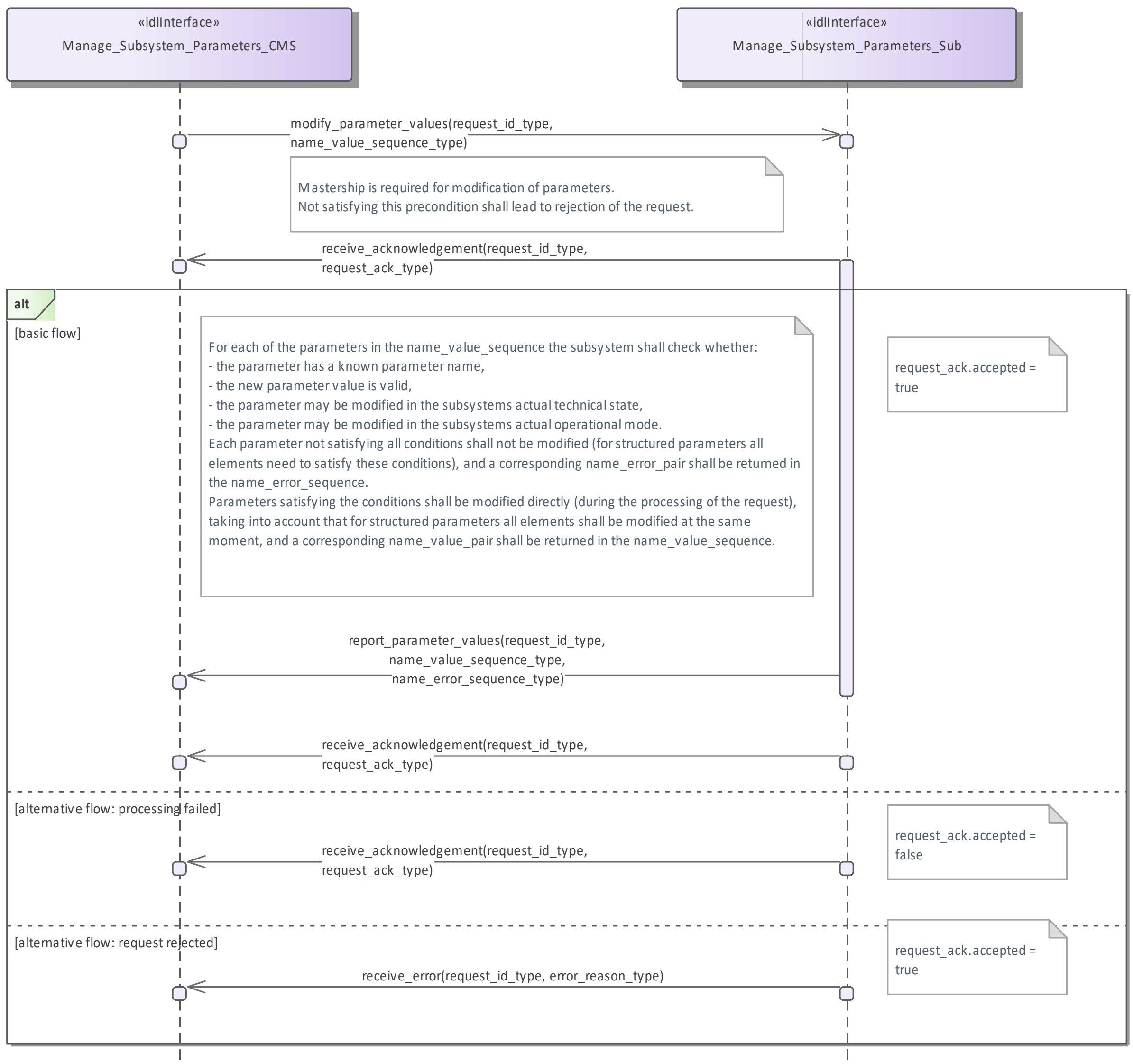


Figure 7.124 Basic Flow - Parameter Value Modification (Interaction diagram)

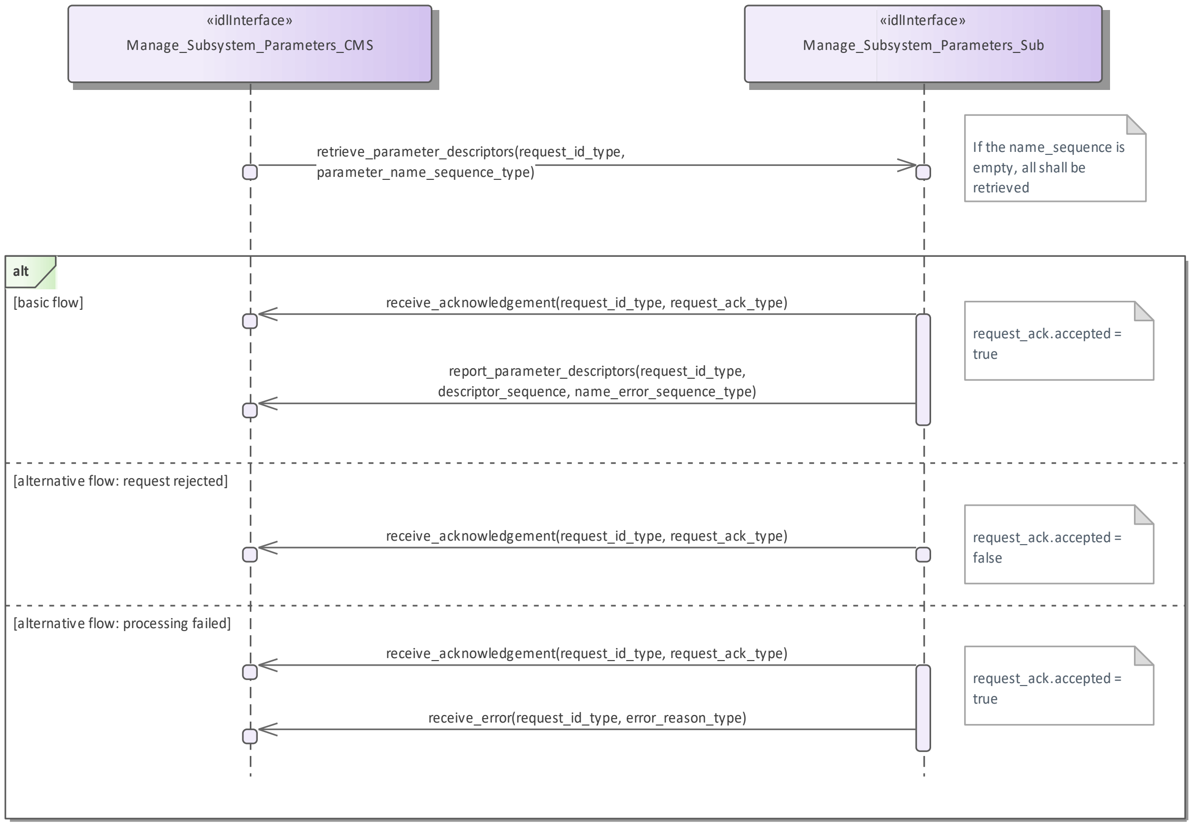


Figure 7.125 Basic Flow - Parameter Descriptor Retrieval (Interaction diagram)

7.8.3.8 Provide_Subsystem_Services

Parent Package: Subsystem_Control

PackageTag: No_PSM = DDS
Provide_Subsystem_Services_CMS
Type: Interface
Package: Provide_Subsystem_Services
Subsystems offer a number of services to a CMS. Some of the services are mandatory for the type of subsystem, others are optional. New services may be known to the CMS or may not be known.
Consequently, the CMS needs to know which services are provided by a subsystem and the subsystem needs to know which services the CMS is able to interact with.
The services considered here are the final versions of those that are specified and defined by the rest of this standard. Some of them are not necessarily implemented by each product of the type of subsystem but also not necessarily supported by each CMS.
The service-related information provided by the subsystem to the CMS deals with both, the interfaces offered by the subsystem and the interfaces expected on CMS side which are necessary to use the service.

Lack of mastership
Mastership of the subsystem must not have an impact upon this interface.

Plug-&-Play aspect

Both sides, subsystem and CMS, shall follow a technical evolution process which is not necessarily coordinated. Therefore, the latest subsystem version may provide a service which is not yet supported by the CMS or the CMS may be prepared to use a service which is not provided by the subsystem.
This may also cause inconsistencies regarding the interfaces to be made available on both sides. As the subsystem may not have an own operator display, it is intended to use the health state of the subsystem if an indication at CMS is to be achieved saying that the interface to the CMS is not implemented properly.

Configuration data of services
The information to be provided to the CMS as information about the implemented services may include related configuration data and may include the information which parts of the service interfaces are supported.

System integration test
After installation of a subsystem on-board, connecting the hardware interfaces with the related CMS hardware interfaces and performing a setup process if applicable it is expected that an interface verification procedure shall be performed. This procedure shall apply all negotiated interfaces so that an improper implementation shall turn-up at that occasion, already. Insofar, the alternative flows should be considered as an integration aid, only.

Spontaneous reporting
Interfaces for which registration/de-registration is considered as an optional facility are written, accordingly.
Registration/de-registration of recipients is done using standard registration mechanism (register interest)

Pre-condition: Subsystem identification.: Provide subsystem identification has been passed successfully.

Post-condition: The CMS is aware of the services and related interfaces supported by the subsystem.:

Post-condition: The subsystem is aware of the service-related interfaces the CMS may interact with.:

Post-condition: The Services do not match.: Each of the alternative flows indicates a fatal error which means that the interface is not implemented properly. The CMS does not take any further action but alerts the operator, accordingly.

Table 7.303 - Methods of Interface Provide_Subsystem_Services_CMS

Method	Notes	Parameters
receive_implemented_services()	Receive services which are implemented by a subsystem	request_id_type **the_request_id** service_indication_list_type **service_indication_list**

Provide_Subsystem_Services_Sub
Type: Interface
Package: Provide_Subsystem_Services

Table 7.304 - Methods of Interface Provide_Subsystem_Services_Sub

Method	Notes	Parameters
receive_supported_services()	Receive services which are supported by the CMS	request_id_type **the_request_id** service_list_type **supported_service_list**

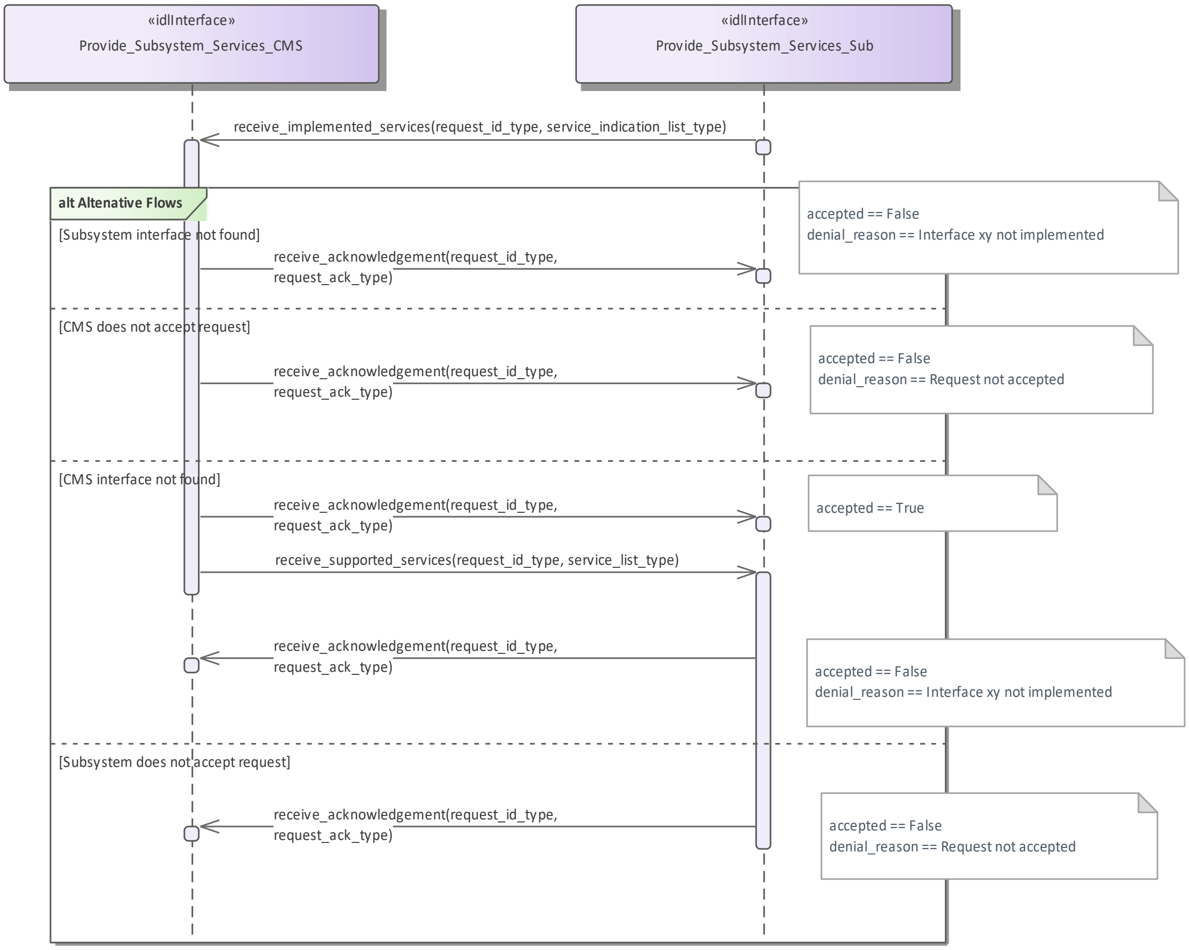


Figure 7.126 Alternative Flow - Service negotiation (Interaction diagram)

Figure 7.127 Basic Flow - Service negotiation (Interaction diagram)

7.8.3.9 Manage_Mastership

Parent Package: Subsystem_Control
This package contains interfaces for the Manage Mastership service.
Manage_Mastership_CMS
Type: Interface
Package: Manage_Mastership
Besides the CMS, the subsystem may be controlled via other control points, e.g. the subsystem local control unit. This interface describes how the CMS, as any other actor, shall handle the exclusive control of the subsystem (mastership). In fact, every subsystem may be controlled by only one actor at the same time. Only the actor who has the mastership of a subsystem may have exclusive control of the subsystem. Exclusive control means that the subsystem may accept only commands sent by the actor who has its mastership.
The subsystem Mastership may be acquired in two ways:

1. PERIODIC MASTERSHIP REQUEST: The actor who wants to acquire the mastership of a subsystem send to it a periodic Mastership request; the subsystem may accept or deny. Once acquired, the subsystem Mastership is released giving up the periodic Mastership requests sending. This happens both in case of intentional decision and critical event as CMS unavailability or connection loss. As long as CMS wants to maintain the Mastership of the subsystem, it shall continue the periodic Mastership requests sending. The CMS is informed about the Mastership control state by receiving a periodic message sent by the subsystem.

1. ASYNCHRONOUS MASTERSHIP REQUEST: The actor who wants to acquire the mastership of a subsystem send to it an asynchronous request. the subsystem may accept or deny. Once acquired, the mastership is until the mastership owner decides to intentionally release it or until a critical event, which is mastership owner unavailability or connection failure, occurs. In case of intentional mastership release, the CMS shall send an asynchronous mastership release request. In case of critical event, the mastership of the subsystem is automatically released. This happens when the subsystem does no longer receive the CMS heartbeat. The CMS is informed about the Mastership control state by receiving an asynchronous message sent on change by the subsystem.

Mastership management rules
The subsystem Mastership assignment is controlled by the subsystem itself according to the following rules:

- no more than one Master at any time, so the subsystem may not be commanded by more than one control point
- the actor which wants to acquire the subsystem Mastership shall ask the subsystem for it, so no request no assignment
- subsystem assigns the Mastership to any actor asking for it without any priority policy, no actor is "more important" than any other.
- On each request, the mastership may be assigned only if it’s free, that is not already assigned (unless a Mastership override request is received)

The Mastership management protocol is managed as follows:

- actor which wants to acquire the subsystem Mastership shall ask for it sending to the subsystem the Mastership requests which could be asynchronous or periodic
- in case of periodic request for Mastership assignment, as long as the actual Master wants to maintain the Mastership, it shall continue the periodic Mastership requests sending
- if the actual Master wants to release the Mastership in case of periodic request for Mastership management, it shall give up the periodic Mastership requests sending, otherwise, in case of asynchronous request, it shall send an asynchronous request for mastership release
- subsystem keeps informed about the actual Mastership state and its changes (if any).

At any time the subsystem Mastership may be either “free”, that is assigned to none and then available to anybody asks for it, or assigned to somebody, where this somebody may be CMS or not. At the subsystem power-on the Mastership is “free”, then:

- as long as the Mastership state is "free", the first received Mastership request shall be satisfied (whether the requestor is CMS or not)
- as long as the Mastership is assigned (to CMS or to somebody other than CMS), the current Master shall maintain the Mastership possession until the Mastership owner is no longer available or decides to release it
- as long as the Mastership is assigned (to CMS or to somebody other than CMS), Mastership requests received from other than the current Master shall be no satisfied, unless a Mastership Override is received, which shall force a Mastership switch to another Master

Note that the Mastership possession is required to control the subsystem (e.g. execute write commands to it), but it is not required to communicate with subsystem and receive information from it.

Mastership Override
The Mastership management protocol could include a Mastership Override to force a Mastership switch from a Master to another one.

Pre-condition: Subsystem Services: *Provide subsystem services* is successfully passed

Post-condition: Success: The subsystem Mastership state is assigned to CMS or not assigned to CMS, according to the CMS requests, and CMS is informed about.

Post-condition: No Success: The subsystem Mastership state is not according to the CMS requests and CMS has the correct information regarding that state (except in the case of connection loss).

Table 7.305 - Methods of Interface Manage_Mastership_CMS

Method	Notes	Parameters
report_mastership_setting()	This method is used by the subsystem to return the mastership state.	mastership_state_type **control_state**

Manage_Mastership_Sub
Type: Interface
Package: Manage_Mastership

Table 7.306 - Methods of Interface Manage_Mastership_Sub

Method	Notes	Parameters
acquire_mastership()	This method is used by the CMS to acquire the mastership.	~~unsigned long **count** This parameter is used with implementation specific semantics to manage subsystem mastership.~~subsystem_id_type **target_subsystem_id** The subsystem to which the request to acquire mastership is being sent. request_id_type **request_id** unsigned long **count** This parameter is used with implementation specific semantics to manage subsystem mastership.
release_mastership()	This method is used by the CMS to release the mastership.	~~unsigned long **count** This parameter is used with implementation specific semantics to manage subsystem mastership.~~subsystem_id_type **target_subsystem_id** The subsystem to which the request to acquire

		mastership is being sent. request_id_type **request_id** unsigned long **count** This parameter is used with implementation specific semantics to manage subsystem mastership.

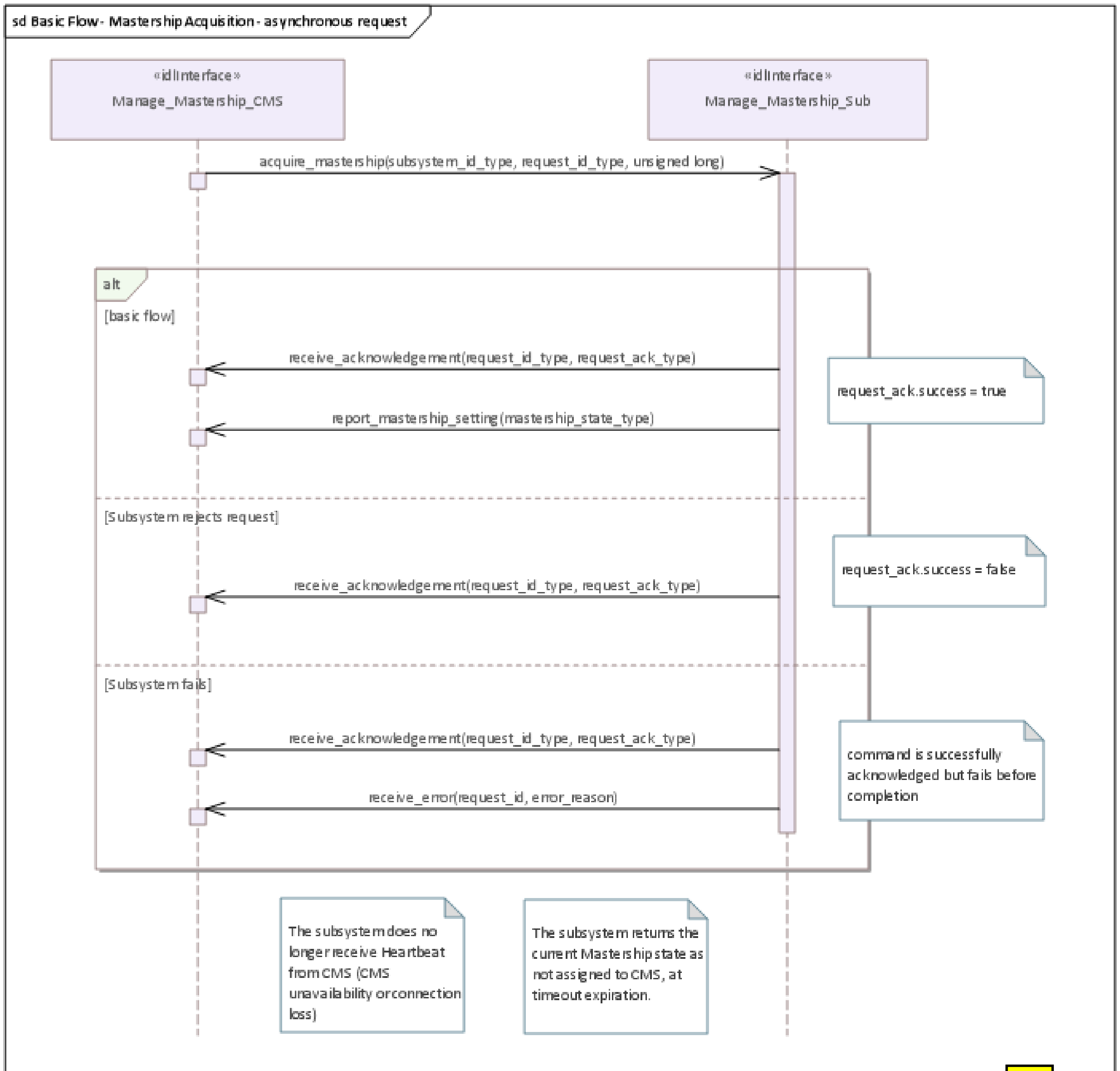


Figure 7.128 Basic Flow - Mastership Acquisition - asynchronous request (Interaction diagram)

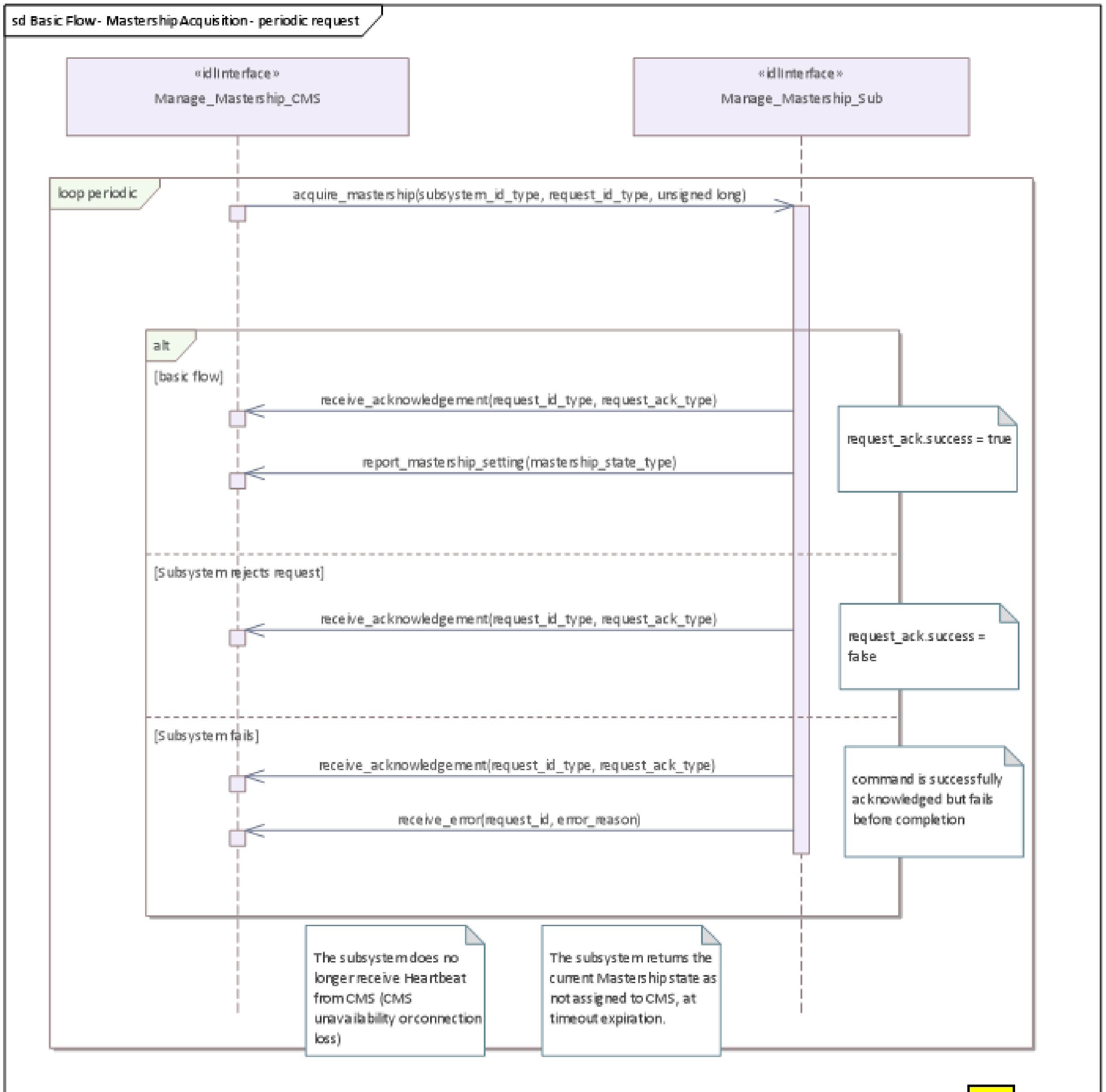


Figure 7.129 Basic Flow - Mastership Acquisition - periodic request (Interaction diagram)

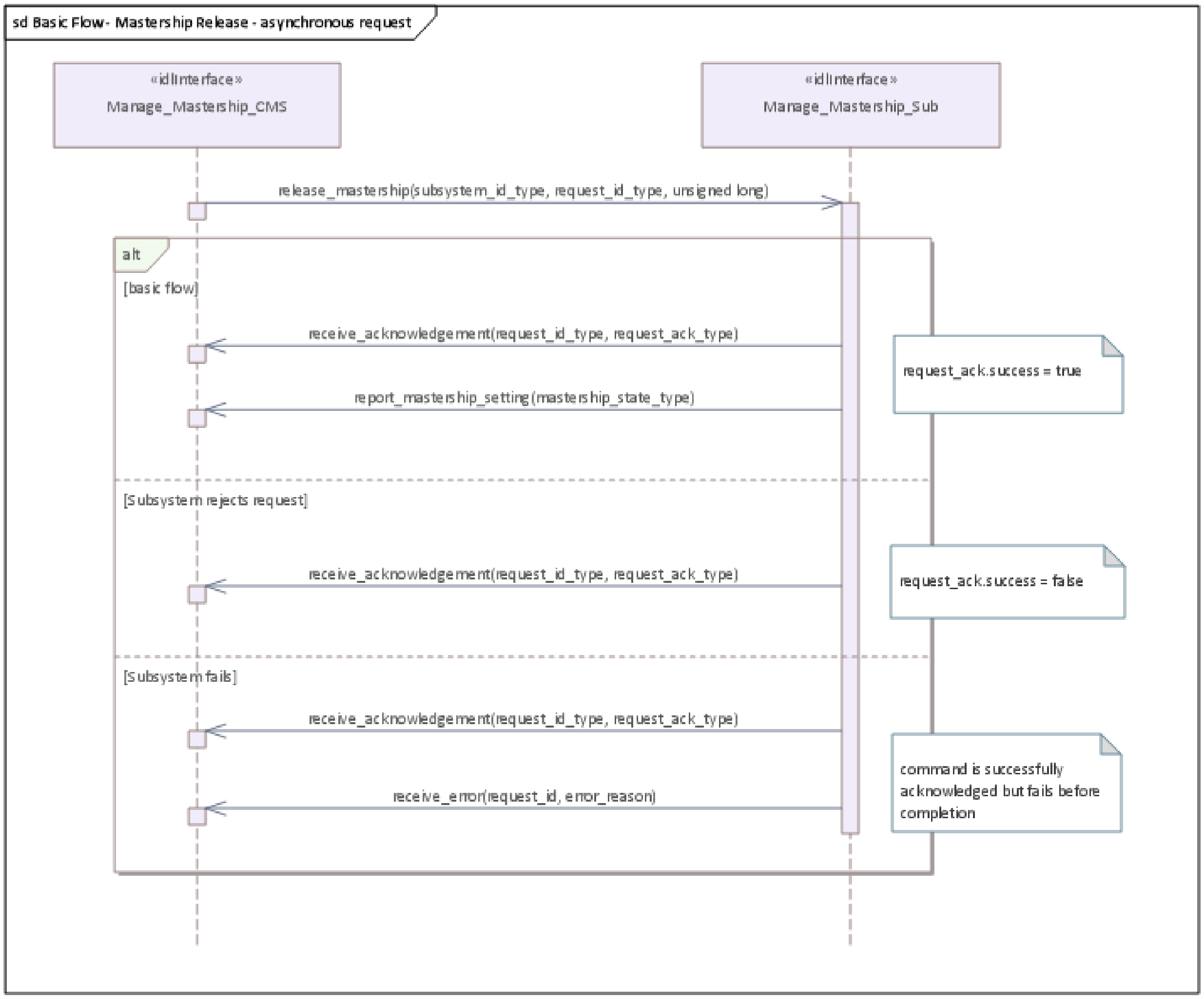


Figure 7.130 Basic Flow - Mastership Release - asynchronous request (Interaction diagram)

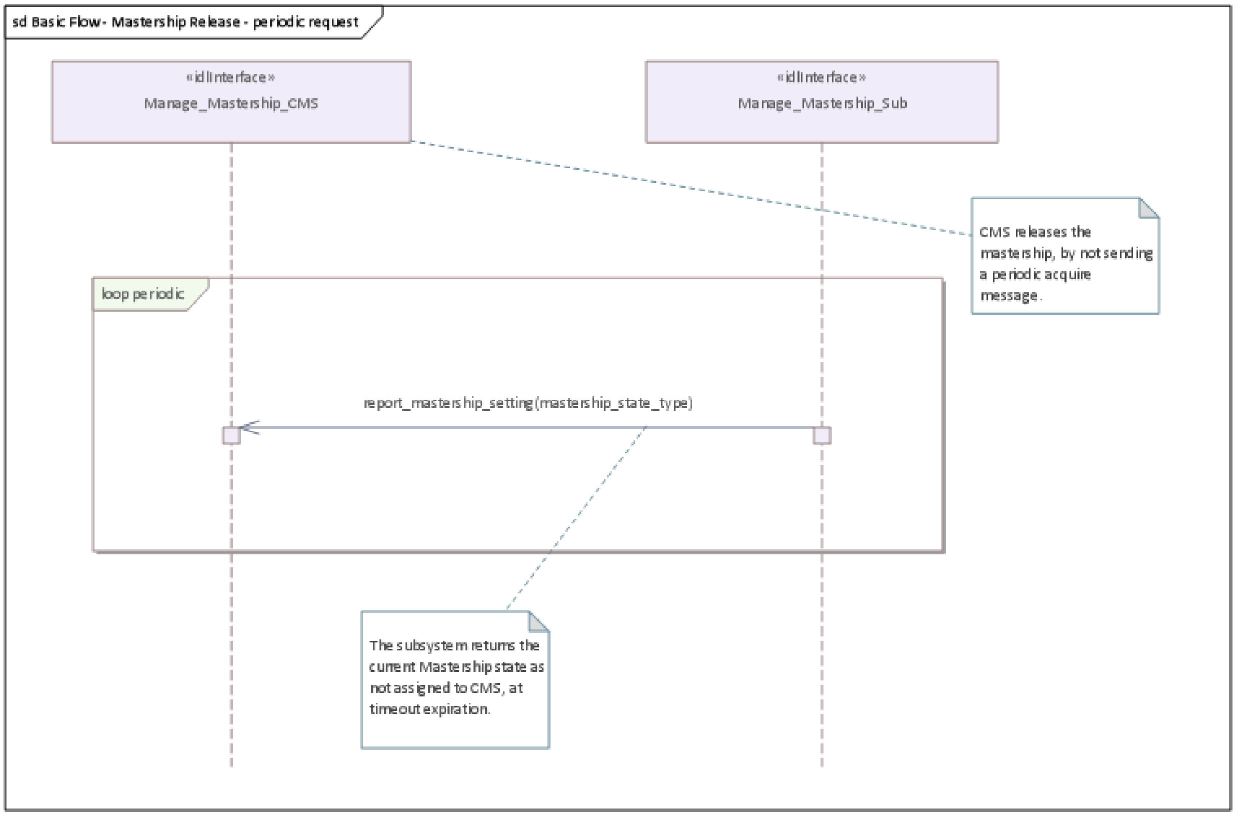


Figure 7.131 Basic Flow - Mastership Release - periodic request (Interaction diagram)

7.8.3.10 Register_Interest

Parent Package: Subsystem_Control

PackageTag: No_PSM = DDS
Register_Interest_CMS
Type: Interface
Package: Register_Interest
This service allows the CMS to register (and deregister) interest in other services. It is explicitly meant to address the possibility of CMS "subscribing" to information supplied by the subsystem, with the understanding that the information shall be provided by the subsystem, without the need for further request. Such mode of operation may be applicable for those services, which have been reported as such in Provide subsystem services. This includes typically track and plot reporting services, but may involve other services as well.

The service starts when the actor registers interest in information provided by a service. The registration shall include information on:

- The service for which the actor wants to register / deregister his interest
- The information within the service for which the actor wants to register / deregister his interest
- The intended (direct or indirect) recipient(s) of the information provided by the subsystem.
- Any parameters of the provision needed such as Quality of Service parameters.

The service ends when the subsystem confirms registration / deregistration of interest.
Pre-condition: Sensor health state: The sensor and the service need to be in the health state AVAILABLE or DEGRADED.

Table 7.307 - Methods of Interface Register_Interest_CMS

Method	Notes	Parameters
confirm_registration()	Confirm registration of interest	request_id_type **request_id**

Register_Interest_Sub
Type: Interface
Package: Register_Interest

Table 7.308 - Methods of Interface Register_Interest_Sub

Method	Notes	Parameters
register_interest()	Register interest in the service	request_id_type **request_id** interest_list_type **the_interest_list**

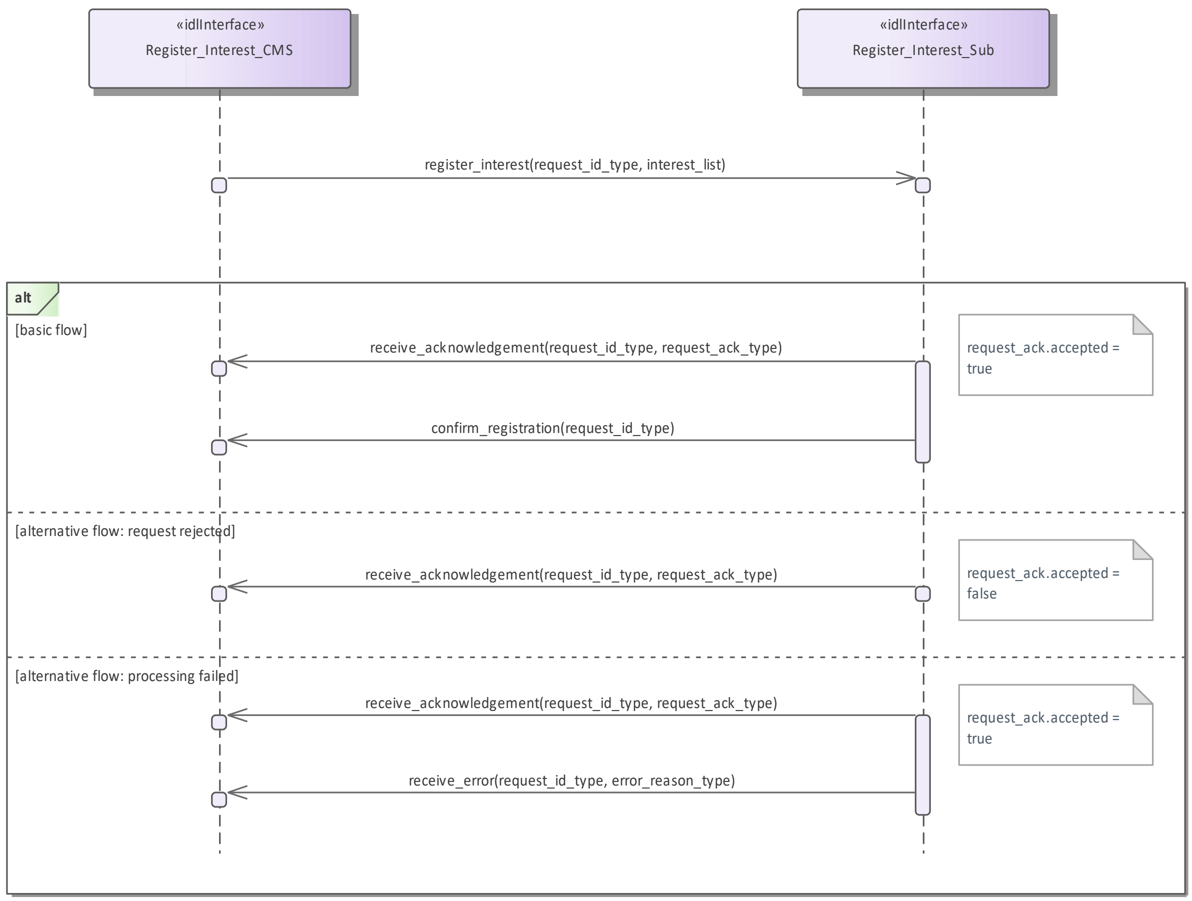


Figure 7.132 Basic Flow - Interest Registration (Interaction diagram)

7.8.4 Recording_and_Replay

Parent Package: Subsystem_Services
Contains the interfaces controlling recording and replay.

7.8.4.1 Control_Recording

Parent Package: Recording_and_Replay

Contains the interface controlling the recording of information.
Control_Recording_CMS
Type: Interface
Package: Control_Recording
The interface describes how the CMS controls the recording of information. Such information may be used to support:

- Setting-to Work/Commissioning
- Equipment monitoring
- Performance monitoring and evaluation
- 'Black Box' recording
- Safety of Life at Sea (SOLAS) recording
- De-briefing
- Training
- Post exercise analysis

For the purposes of this interface, 'recording' is defined as the synchronous capture of real-time information at a defined rate. Provision of additional 'live' real-time data for instrumentation purposes, i.e. for display rather than recording, is outside the scope.

Each record within the recording must be identified and time-stamped.
The operation of the recording function must not affect normal operation of the subsystem.
For simplicity, concurrent recording and replay is not supported.
Pre-condition: Provide Subsystem Services must have executed successfully.:
Pre-condition: The subsystem must be in Technical State READY or ONLINE:
Pre-condition: The CMS must have Mastership.:
Post-condition: After successful termination, the recording is available for replay via Control_Replay, using the identifier specified.:
Post-condition: In the case of abnormal termination, there is a possible fault in the recording subsystem.:
Control_Recording_Sub
Type: Interface
Package: Control_Recording

Table 7.309 - Methods of Interface Control_Recording_Sub

Method	Notes	Parameters
define_recording_set()	Specifies what is to be recorded	request_id_type **request_id** recording_set_type **recording_parameters_list**
start_recording()	Starts the recording as specified. Note that only one recording may be running at a time.	request_id_type **request_id** recording_id_type **id**
stop_recording()	Stops the recording	request_id_type **request_id**

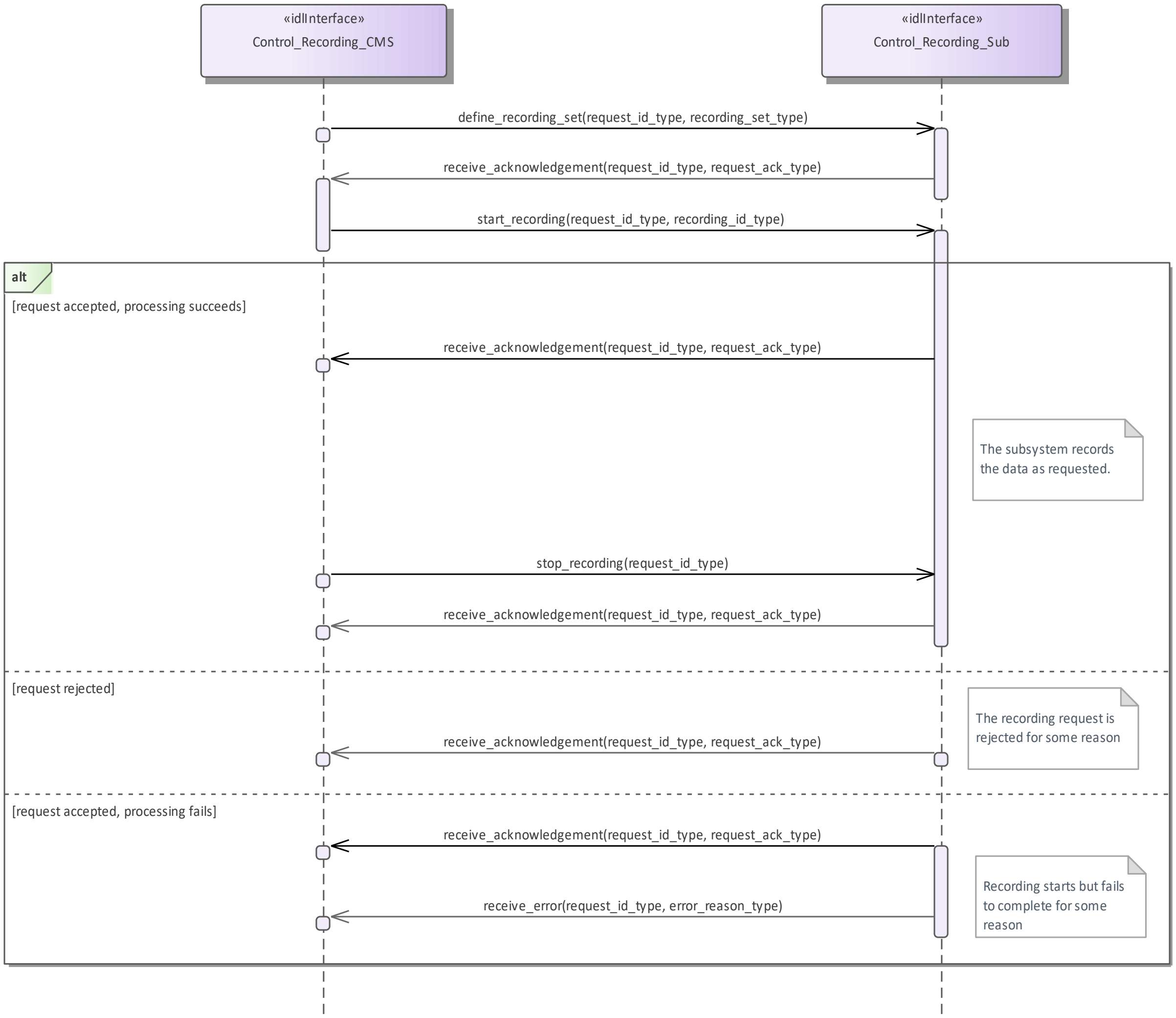


Figure 7.133 Control Recording (Interaction diagram)

This shows the required sequential behaviour for Control_Recording, See diagram embedded notes for further explanation.

7.8.4.2 Control_Replay

Parent Package: Recording_and_Replay
Contains the interfaces controlling the replay of information; either using the original interfaces or as a data dump for offline processing.
Control_Replay_CMS
Type: Interface
Package: Control_Replay
This interface defines how the CMS controls the replay of information previously recorded using Control_Recording
Replay is supported in two modes: REAL-TIME and RAW. REAL-TIME mode is used to replay in real time, or at a multiple of real-time, data that was visible on other OARIS interfaces via the interfaces used during recording. RAW mode is used to replay data that was visible on other OARIS interfaces and/or internal subsystem data that was not available on other OARIS interfaces. In this case the data is merely transferred to the CMS as a set of time-tagged values with no attempt made to reconstruct real-time behaviour.
One or more recordings must have been made using Control_Recording.
For simplicity, concurrent recording and replay is not supported.

Pre-condition: Provide Subsystem Services must have executed successfully.:
Pre-condition: The subsystem must be in Technical State READY or ONLINE:
Pre-condition: The CMS must have Mastership..:
Pre-condition: In the case of abnormal termination, there is a possible fault in the replay subsystem. :

Table 7.310 - Methods of Interface Control_Replay_CMS

Method	Notes	Parameters
end_of_recording()	The subsystem has reached the end of the recording before a stop command was received.	request_id_type **request_id**
receive_recording()	Used to transfer a raw recording to the CMS	request_id_type **request_id** recording_type **requested_recording** The raw recording data.

Control_Replay_Sub
Type: Interface
Package: Control_Replay

Table 7.311 - Methods of Interface Control_Replay_Sub

Method	Notes	Parameters
resume_replay()	Resumes replay following a stop command	request_id_type **request_id** actual_time_type **actual_time** The current time (time of day) at which playback should start. This allows synchronisation of playback from different subsystems. replay_speed_type **replay_speed** Controls the replay speed. 1.0 represents real time.
start_replay()	Starts replay as specified	request_id_type **request_id** replay_set_type **replay_parameters_list** recording_id_type **id** actual_time_type **actual_time** The current time (time of day) at which playback should start. This allows synchronisation of playback from different subsystems. recorded_time_type **recorded_time** The time in the recording at which playback should start. replay_speed_type **replay_speed** Controls the replay speed. 1.0 represents real time.
stop_replay()	Stops replay	request_id_type **request_id**
upload_recording()	Requests transfer of a raw recording	request_id_type **request_id** recording_id_type **id**

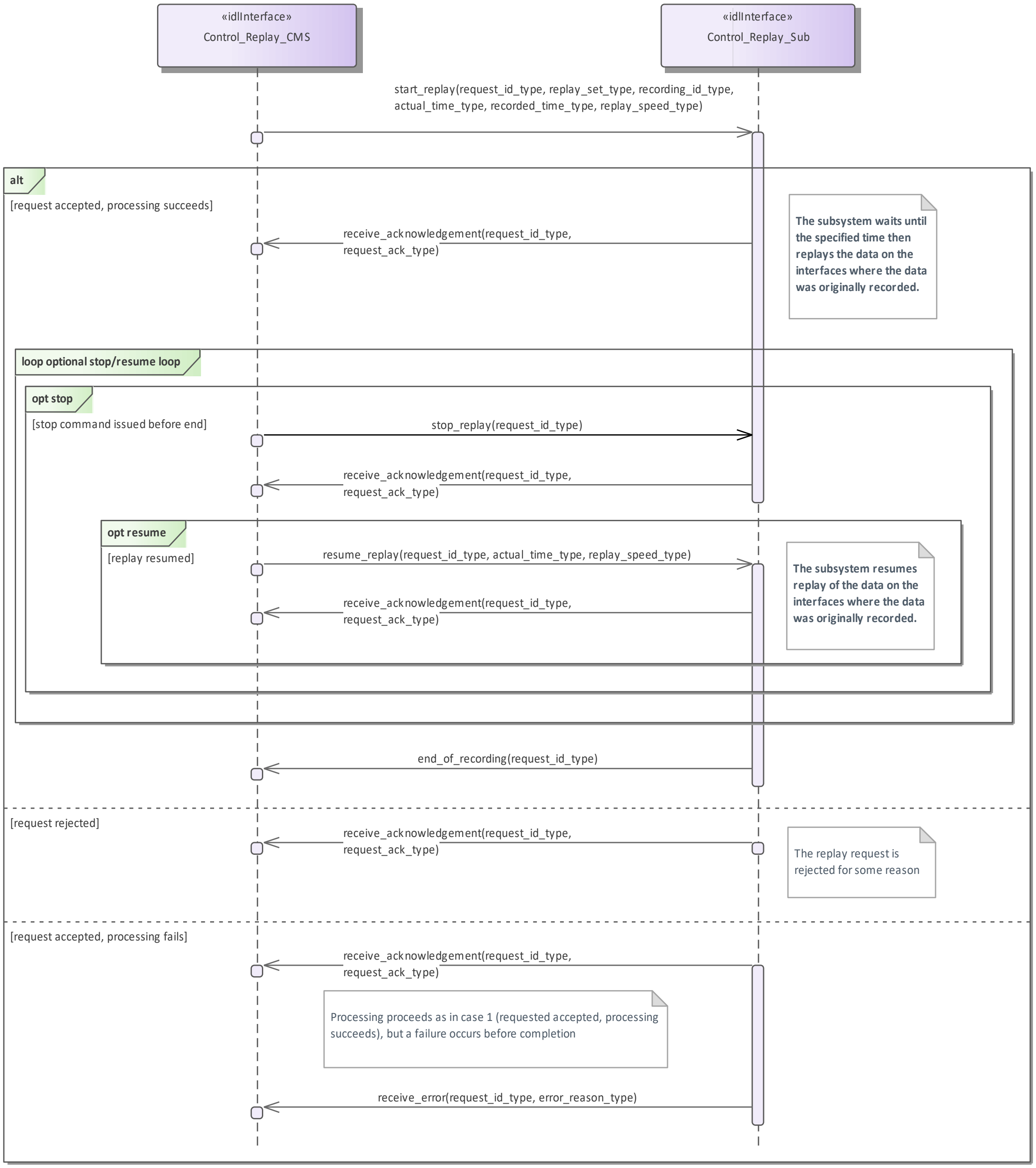


Figure 7.134 Control Replay (Interaction diagram)

This shows the required sequential behaviour for Control_Replay using real_time mode, See diagram embedded notes for further explanation.

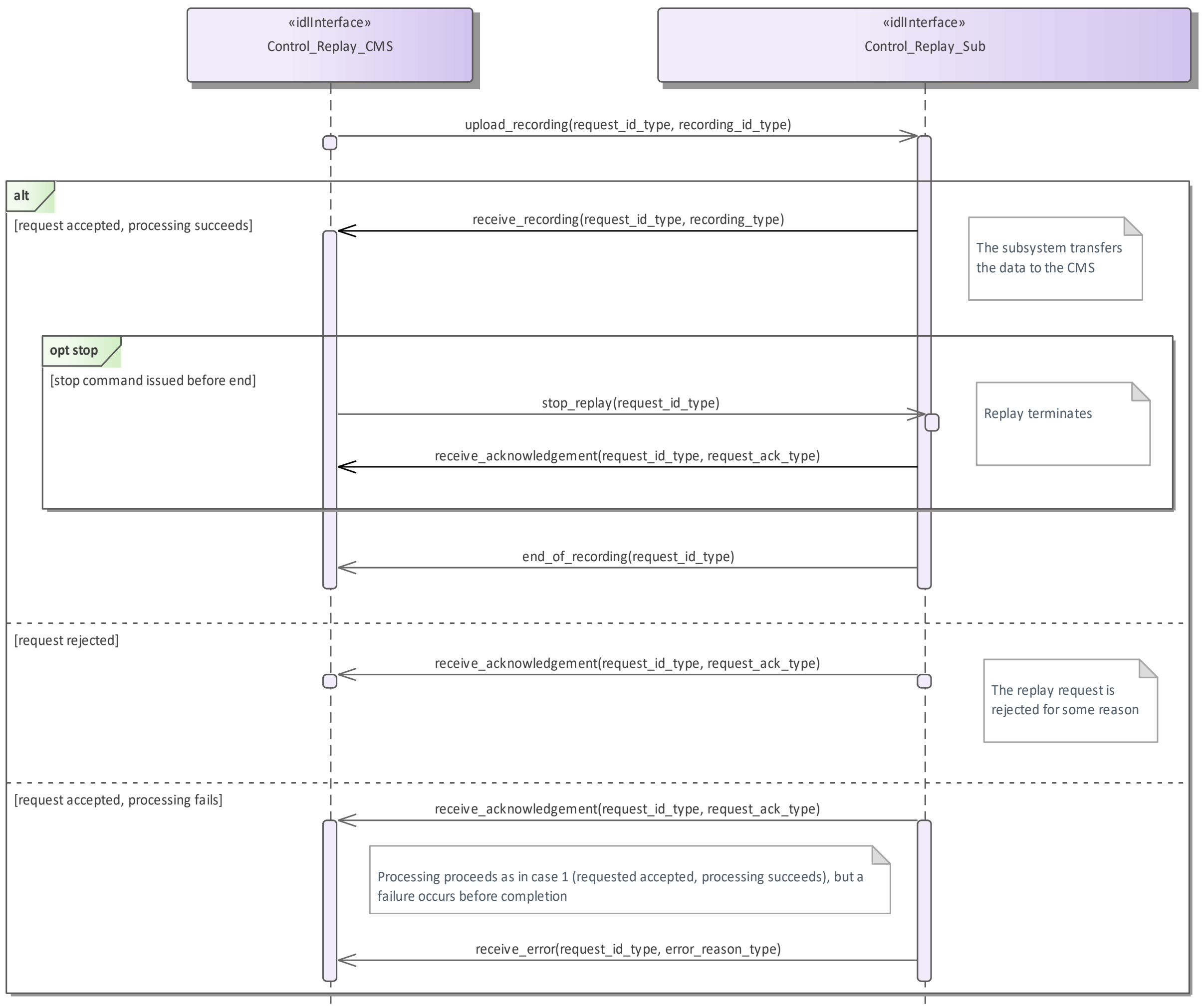


Figure 7.135 Control Replay (RAW) (Interaction diagram)

This shows the required sequential behaviour for Control_Replay using raw mode, See diagram embedded notes for further explanation.

7.8.5 Simulation_Support

Parent Package: Subsystem_Services

7.8.5.1 Define_Simulation_Scenario

Parent Package: Simulation_Support

Define_Simulation_Scenario_CMS
Type: Interface
Package: Define_Simulation_Scenario
This describes how the contents of a simulation scenario are communicated between the CMS and the subsystem.
The CMS provides the subsystem with a simulated environment which consists of simulated objects of different kinds.
A subsystem with built-in simulation capability may participate in this simulation not only by being a consumer of the simulated environment but by contributing actively to it.

Radar type subsystems shall typically build simulated plots or tracks from the simulated environment, while contributing simulated electromagnetic emissions to it. These simulated emissions may in turn be used and detected by other (ESM type) simulations.
Weapon type subsystems when in simulation mode shall typically contribute simulated objects to the simulation that represent the launch/firing and movement of own missiles, bullets or torpedoes and their effect on other simulated objects.
Thus CMS, and subsystem both contribute to the simulated environment. Together they form a simulation federation.

The actor is the Combat Management System.

Relationship to 'control simulation'
The definition of simulation mode and flow of commands to start/stop/freeze/resume a simulation scenario are defined in 'control simulation'.

Relationship to provision of tracks
A radar type subsystem shall provide tracks based on information from the simulated environment, as described above. The interfaces that deal with the provision of tracks indicate whether tracks are simulated or not under amplifying information. This indication should be set for all tracks that are reported in the context of this interface.

Relationship to Receive geographic information
Geographic information is received by using 'Receive geographic information'.

Pre-condition: Subsystem health state.: The subsystem and the relevant subsystem services need to be in the health state AVAILABLE or DEGRADED.

Pre-condition: CMS has mastership.:

Pre-condition: Subsystem simulation mode.: The subsystem must be in subsystem simulation mode ON to participate in the simulation federation.

Pre-condition: Simulation scenario started.: The actor must have started or resumed a simulation scenario.

Pre-condition: Geographic information.: The subsystem may need geographic information about its simulated surroundings available locally or by means of other interfaces in order to calculate the detectability or reachability of simulated objects due to obstacles in the surroundings.

Table 7.312 - Methods of Interface Define_Simulation_Scenario_CMS

Method	**Notes**	**Parameters**
write_emitter_system_data_CMS()	Write emitter system data	anonymous_blob_type **emitter_system_data**
write_radar_beam_data()	Write radar beam data	anonymous_blob_type **radar_beam_data**

Define_Simulation_Scenario_Sub
Type: Interface
Package: Define_Simulation_Scenario

Table 7.313 - Methods of Interface Define_Simulation_Scenario_Sub

Method	**Notes**	**Parameters**
write_emitter_system_data_Sub()	Write emitter system data	anonymous_blob_type **emitter_system_data**
write_environment_data()	Write environment data	anonymous_blob_type **environmental_entity_data**
write_jammer_beam_data()	Write jammer beam data	anonymous_blob_type

		jammer_beam_data
write_platform_data()	Write platform data	anonymous_blob_type **platform_data**

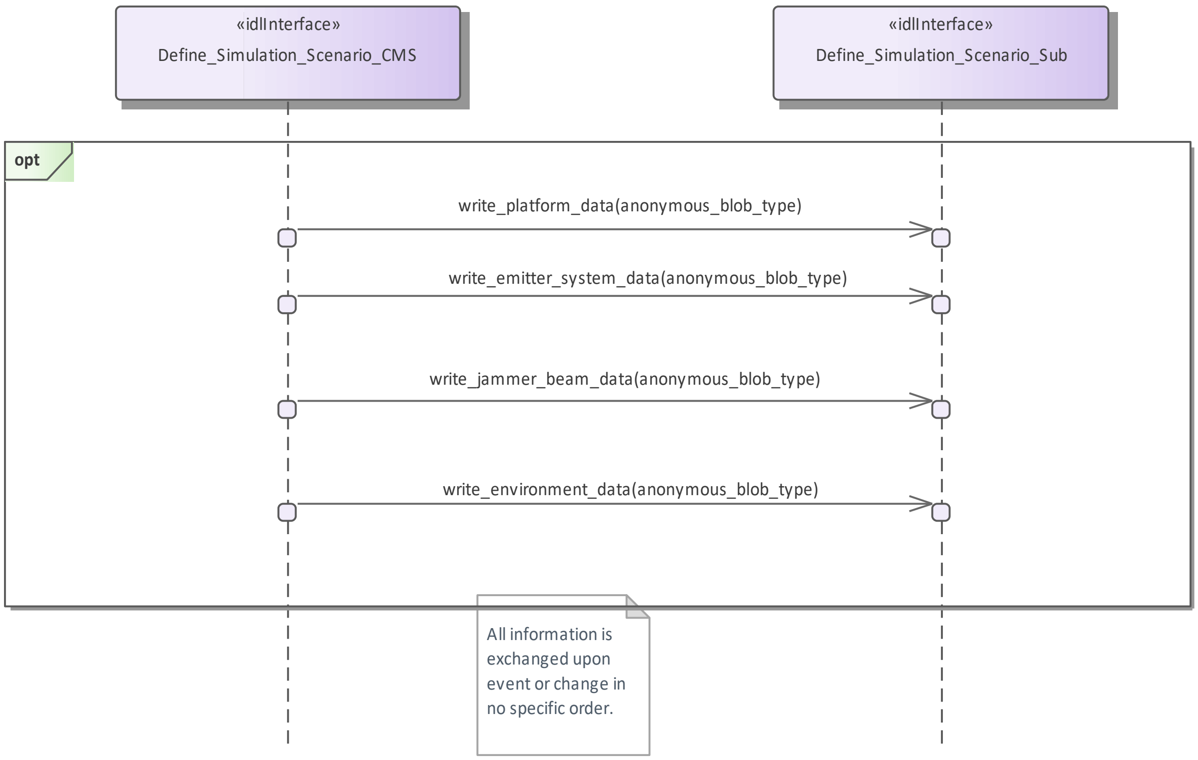


Figure 7.136 Basic Flow - Define Simulation Scenario Data (Interaction diagram)

Figure 7.137 Basic Flow - Define Subsystem Scenario Data (Interaction diagram)

7.8.5.2 Control_Simulation

Parent Package: Simulation_Support

Control_Simulation_CMS
Type: Interface
Package: Control_Simulation
This service controls the simulation mode of a subsystem. This simulation mode is independent of the operational mode of the subsystem. Simulation mode is either ON or OFF. "ON" has different meanings for different kinds of subsystems. Effector type subsystems shall not engage real targets but shall simulate the engagement instead. Sensor type subsystems may be fed with simulated targets which shall be reported as plots or tracks. In each case while in simulation mode "ON" the subsystem shall strictly avoid any impact on the environment that could be the result if simulation mode was "OFF".

The actor is the Combat Management System.

Basic Flow – Control simulation mode

Start event – command of simulation-mode

The service is triggered by the actor. The actor commands the simulation mode which may be one of the following:

- ON: This indicates that the subsystem shall operate in simulation mode
- OFF: This indicates that the subsystem shall stop operating in simulation mode and that any current simulation shall be terminated

On occurrence of the trigger provision of subsystem-simulation-mode is executed.

Provision of subsystem-simulation-mode
After receipt of the simulation mode from the actor the subsystem responds with its subsystem simulation mode. The subsystem simulation mode may be one of the two:

- ON: This indicates that the subsystem is operating in simulation mode
- OFF: This indicates that the subsystem is not operating in simulation mode

Basic Flow – Control Simulation (Start/Resume, Stop/Freeze)

START/RESUME simulation scenario
Only when in simulation mode ON:
Upon provision of the START/RESUME command by the actor the simulation scenario starts or is resumed after a previously issued FREEZE.

STOP/FREEZE simulation scenario
Only when in simulation mode ON:
Upon provision of the STOP/FREEZE command by the actor the simulation scenario stops or stays frozen.
The service ends.

Provision on initialization
The simulation mode shall be provided by the actor after initialization of the CMS.

The flow of information relevant to subsystem simulation are the subject of another service: Define simulation scenario.
If simulation is stopped or frozen simulation time of the subsystem and the actor shall be also stopped.
The synchronization of simulation time may be performed using START/RESUME command.
Pre-condition: CMS has mastership.:

Table 7.314 - Methods of Interface Control_Simulation_CMS

Method	Notes	Parameters
report_sim_mode_status()	Receive the status and mode of simulation.	request_id_type **request_id** sim_mode_status_type **the_status**

Control_Simulation_Sub
Type: Interface
Package: Control_Simulation

Table 7.315 - Methods of Interface Control_Simulation_Sub

Method	Notes	Parameters
start_resume_session()	This request shall be initiated on demand of the CMS. If the subsystem is in simulation mode it shall start/resume its simulation session and acknowledges the request.	request_id_type **request_id**
start_stop_sim_mode()	This request shall be initiated on demand of the CMS to activate/deactivate the simulation mode of the subsystem. The subsystem needs to acknowledge the request.	request_id_type **request_id** start_stop_sim_mode_request_type **the_request**
stop_freeze_session()	This request shall be initiated on demand of the CMS. If the subsystem is in simulation mode and the session state is running the subsystem needs to stop/freeze its session and acknowledges the request.	request_id_type **request_id** stop_freeze_session_request_type **the_request**

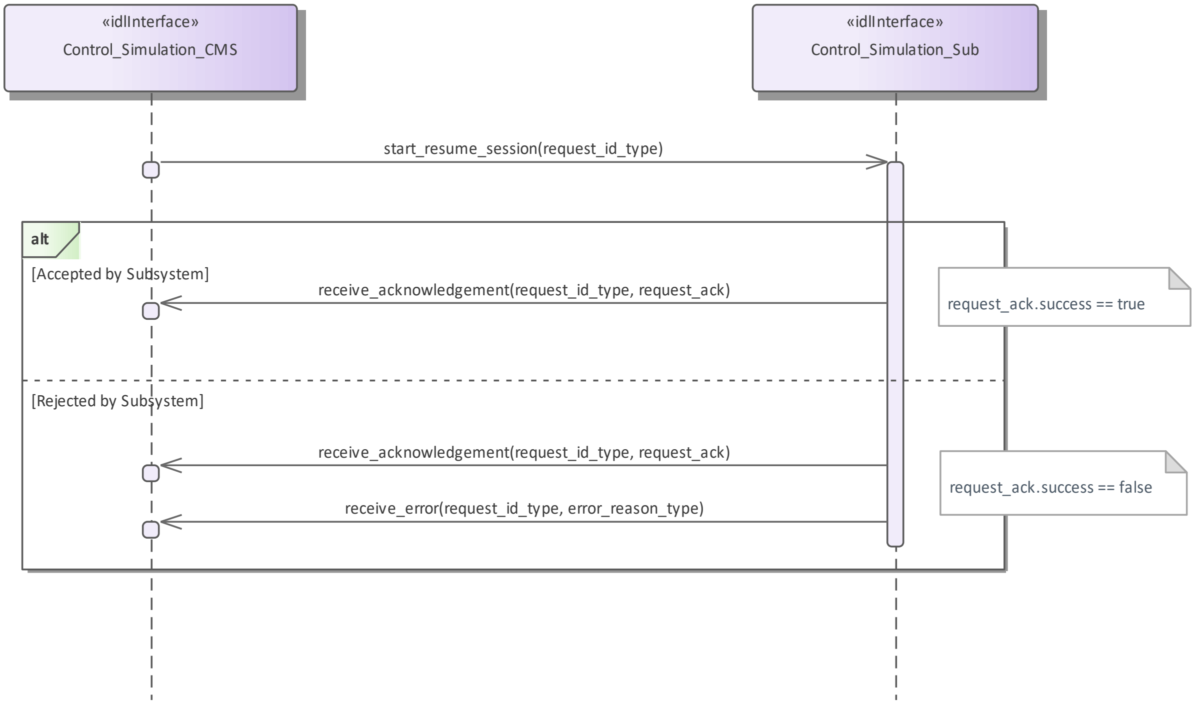


Figure 7.138 Basic Flow - Control Simulation Start/Resume (Interaction diagram)

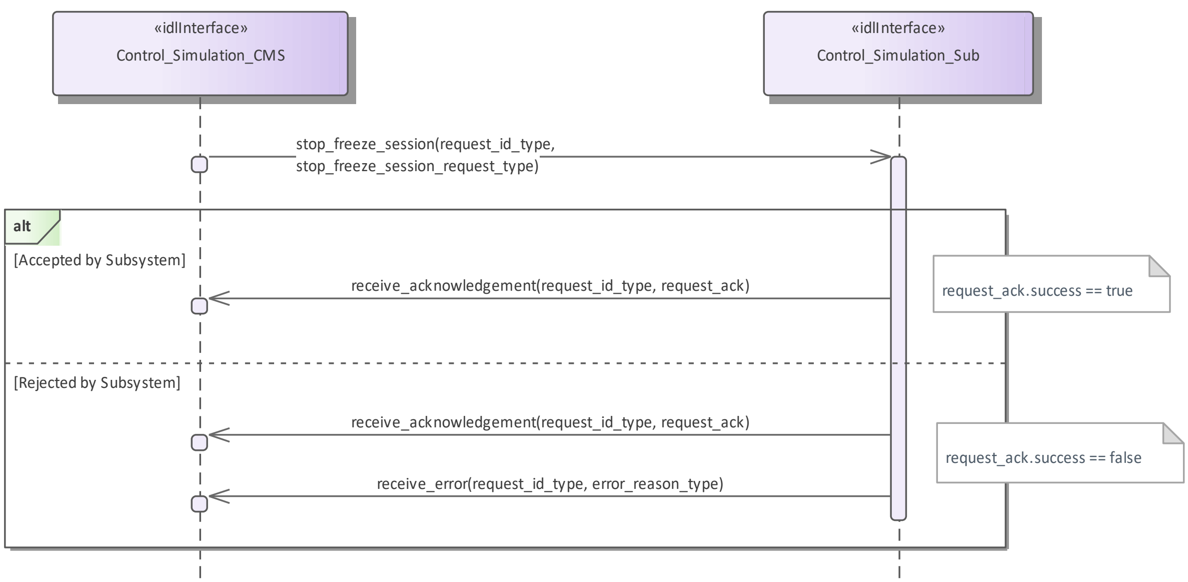


Figure 7.139 Basic Flow - Control Simulation Stop/Freeze (Interaction diagram)

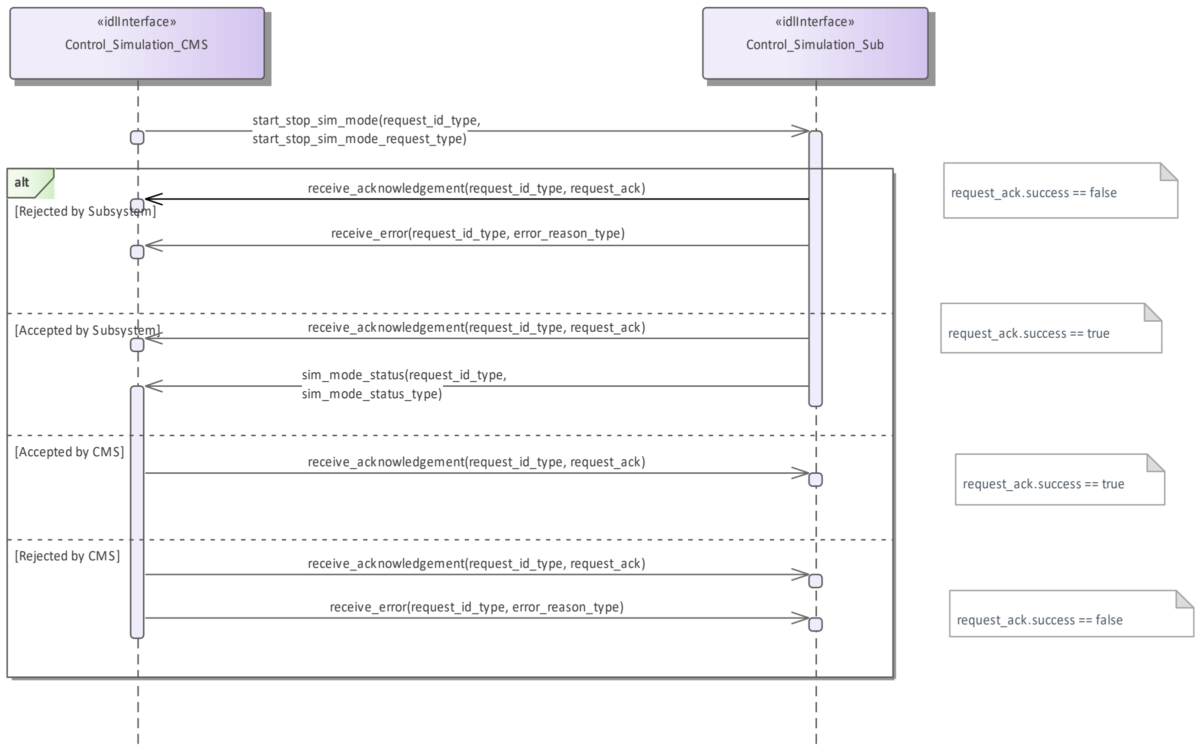


Figure 7.140 Basic Flow - Control Simulation Mode (Interaction diagram)

7.8.5.3 Define_Fault_Scripts

Parent Package: Simulation_Support

Define_Fault_Scripts_CMS
Type: Interface
Package: Define_Fault_Scripts
This enables a maintainer trainer to script a set of subsystem faults, the effects of which would be simulated for training purposes. The faults may be scripted in relation to a specific simulation scenario. Each fault script shall include a unique identifier.

Pre-condition: Subsystem Services: Provide subsystem services has been completed successfully, in particular this service is available.

Table 7.316 - Methods of Interface Define_Fault_Scripts_CMS

Method	Notes	Parameters
fault_script_summary()	This provides a list of all fault scripts for a subsystem to the CMS for confirmation.	request_id_type **request_id** fault_scripts_type **faults** The list of fault scripts

Define_Fault_Scripts_Sub
Type: Interface
Package: Define_Fault_Scripts

Table 7.317 - Methods of Interface Define_Fault_Scripts_Sub

Method	Notes	Parameters
add_fault_scripts()	Adds the given fault scripts to the	request_id_type **request_id**

	subsystem's simulation.	fault_scripts_type **scripts** The fault scripts to be added
remove_fault_scripts()	Removes the given fault scripts from the subsystem's simulation.	request_id_type **request_id** fault_script_ids_type **fault_scripts** The ids of the fault scripts to be removed

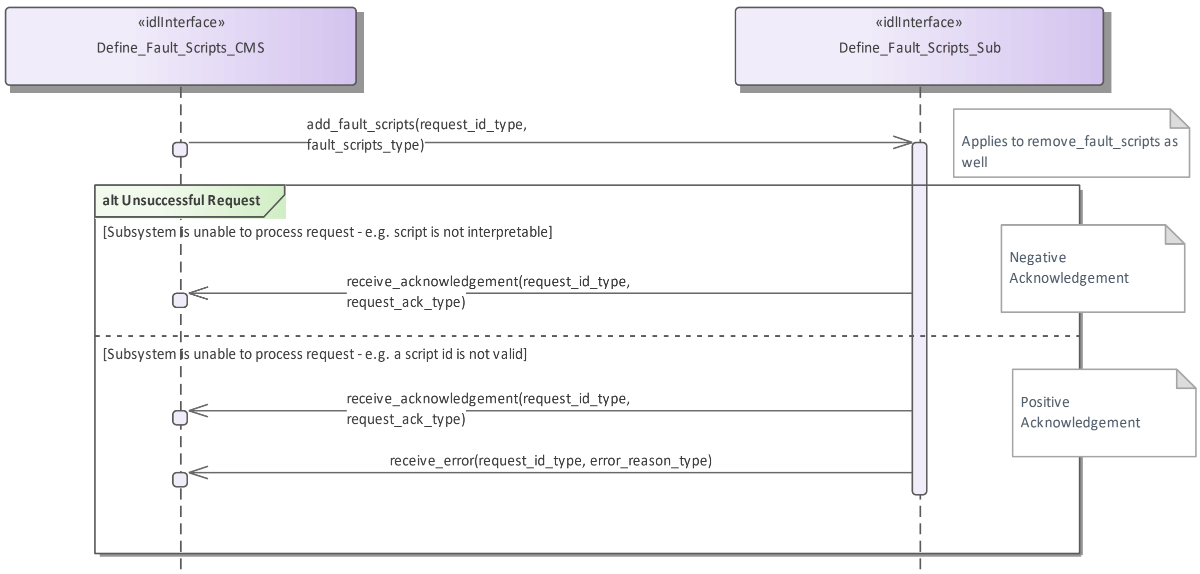


Figure 7.141 Alternative Flow - Define Fault Scripts (Interaction diagram)

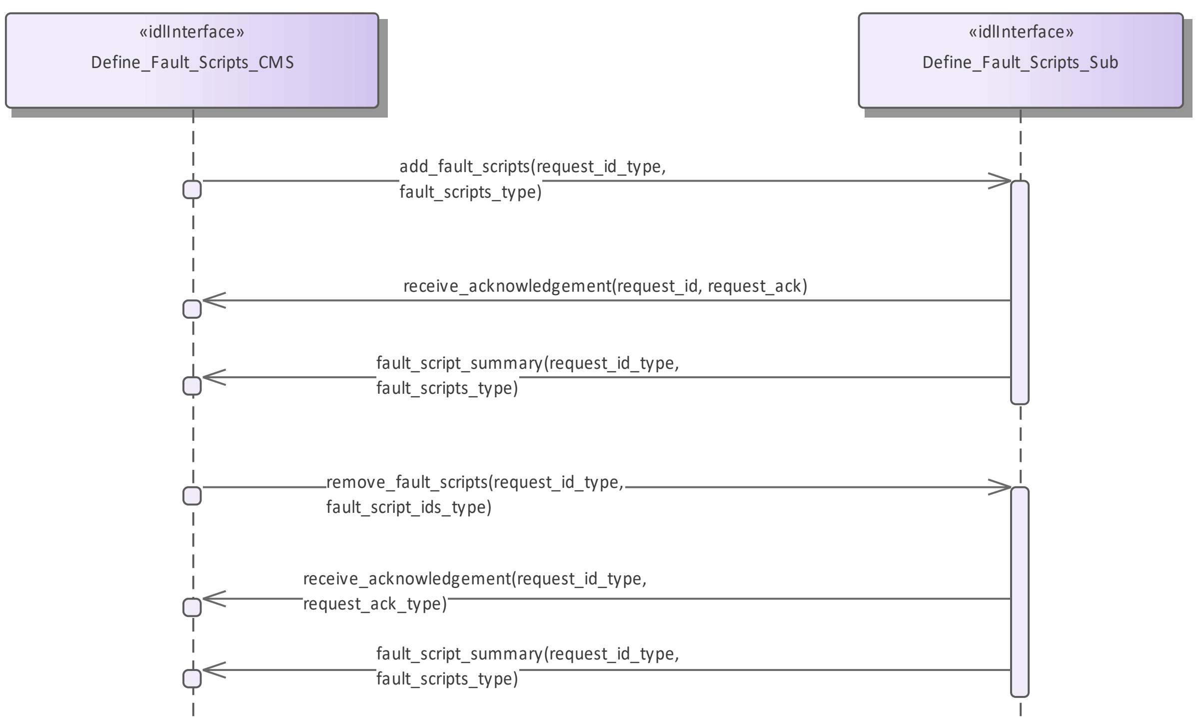


Figure 7.142 Basic Flow - Define Fault Scripts (Interaction diagram)

7.8.5.4 Control_Fault_Scripts

Parent Package: Simulation_Support

Control_Fault_Scripts_CMS
Type: Interface
Package: Control_Fault_Scripts
This enables a trainee, at a CMS Console to cause the generation of predefined fault messages for training purposes (see also Define Fault Scripts). The subsystem shall output Fault Reports to the CMS which a trainee may respond to via the CMS Console. Fault clearance messages shall also be sent to the CMS in response to the trainee taking the appropriate action.

Pre-condition: Technical State: Subsystem is in technical state READY or ONLINE

Pre-condition: Fault Script: Subsystem has a fault script which has been defined previously

Pre-condition: Mastership Required: The CMS has Mastership

Pre-condition: Subsystem Services: Provide Subsystem Services has successfully completed; in particular this service is available

Pre-condition: Simulation Mode: Simulation Mode is ON

Post-condition: Success: Subsystem has provided simulated fault and response to clearance action

Post-condition: Failure: Subsystem has not provided simulated fault and response to clearance action

Control_Fault_Scripts_Sub
Type: Interface
Package: Control_Fault_Scripts

Table 7.318 - Methods of Interface Control_Fault_Scripts_Sub

Method	**Notes**	**Parameters**
enable_fault_script()	Causes the subsystem to indicate the faults specified by the given fault scripts when appropriately stimulated. The faults remain in place until they are cleared either by a call to clear_fault or by an action on another interface that would clear the equivalent non-simulated fault.	request_id_type **request_id** fault_script_ids_type **scripts** The script ids to be enabled
clear_faults()	Clears the faults defined by the given fault scripts.	request_id_type **request_id** fault_script_ids_type **fault_scripts** The script ids to be cleared

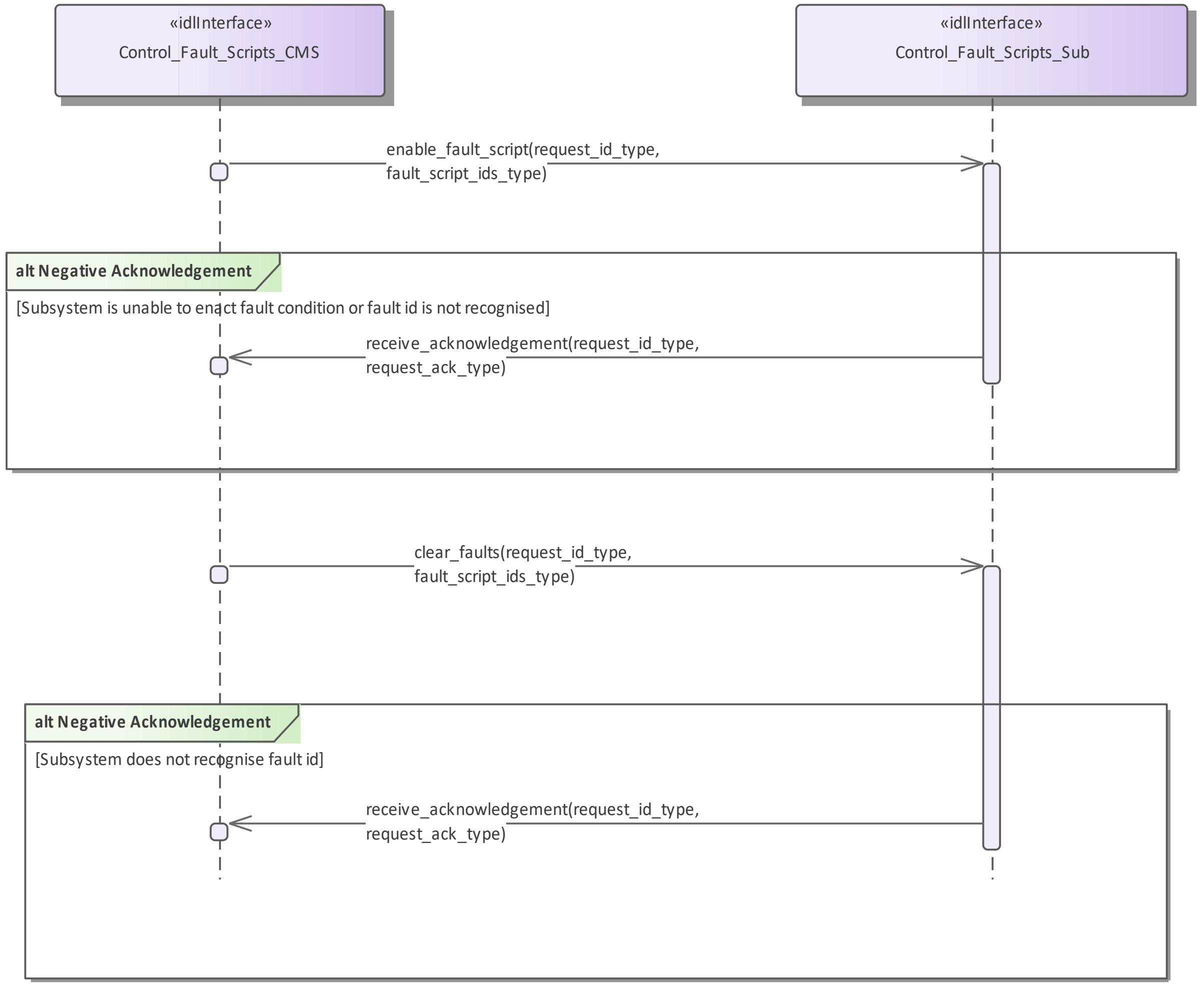


Figure 7.143 Alternative Flow - Control Fault Scripts (Interaction diagram)

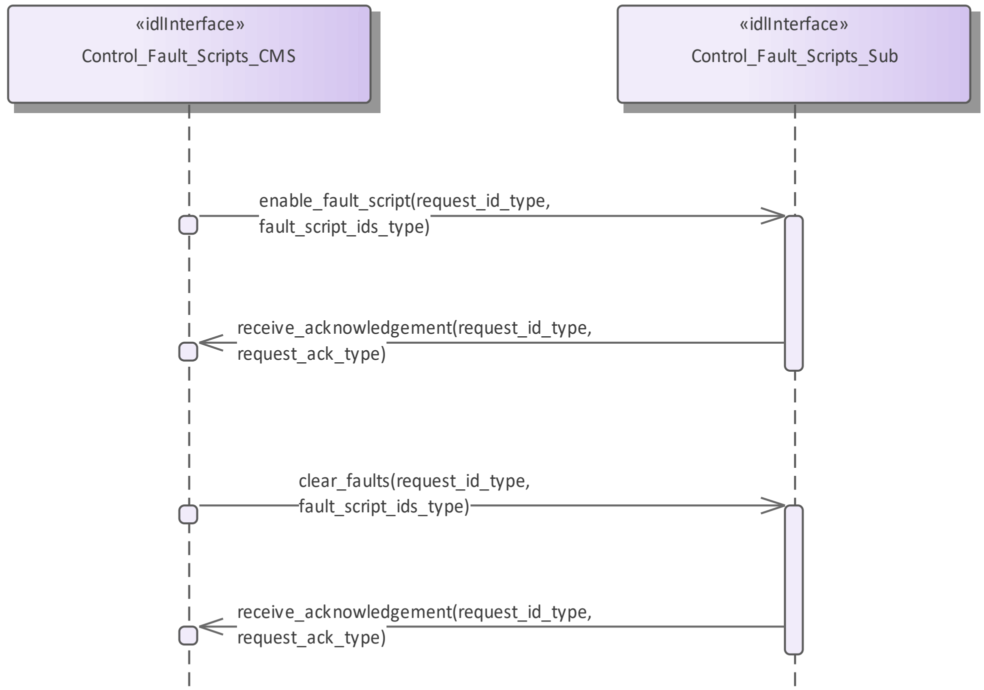


Figure 7.144 Basic Flow - Control Fault Scripts (Interaction diagram)

7.9 Sensor_Services

Parent Package: Service_Interfaces
Contains services associated with the Sensor Domain.

7.9.1 Clutter_Reporting

Parent Package: Sensor_Services
This package contains interfaces for the Clutter Reporting service.

7.9.1.1 Provide_Area_with_Plot_Concentration

Parent Package: Clutter_Reporting
Contains operations and sequence diagrams for the Provide Area with Plot Concentration interface.
Provide_Plot_Concentration_CMS
Type: Interface
Package: Provide_Area_with_Plot_Concentration
The Radar provides the combat management system with the number of plots in a specific sector. The sector information consists of range, azimuth, and elevation. The number of plots observed in the region may provide an indication of high clutter.

Additional Information:

The information may be developed when requested or based on scan histories. The choice of methods depends upon radar design. The timestamp should indicate the oldest data used to create the report to allow the CMS or an operator to determine the validity of the report (i.e. day old data mixed with recent is still only as good as day old data).

Sector Information must consist of a measurement time stamp, range extents, azimuth extents, and elevation extents in platform coordinates.

For radars which report plot concentration without a CMS request, the CMS shall begin to receive reports upon registration of the Provide Plot Concentration interface.

Pre-condition: Radar in ONLINE State:

Post-condition: None:

Table 7.319 - Methods of Interface Provide_Plot_Concentration_CMS

Method	Notes	Parameters
receive_periodic_plot_concentration( )	Interface used by CMS to receive periodic plot concentration reports from the subsystem.	plot_concentration_report_type **plot_concentration_report**
receive_plot_concentration()	Interface used by the CMS to receive a requested plot concentration report from the subsystem.	request_id_type **request_id** plot_concentration_report_type **plot_concentratrion**

Provide_Plot_Concentration_Sub

Type: Interface

Package: Provide_Area_with_Plot_Concentration

Table 7.320 - Methods of Interface Provide_Plot_Concentration_Sub

Method	Notes	Parameters
provide_plot_concentration()	Interface used by the subsystem to receive a plot concentration request from the CMS.	request_id_type **request_id** plot_concentration_request_data_type **plot_request**

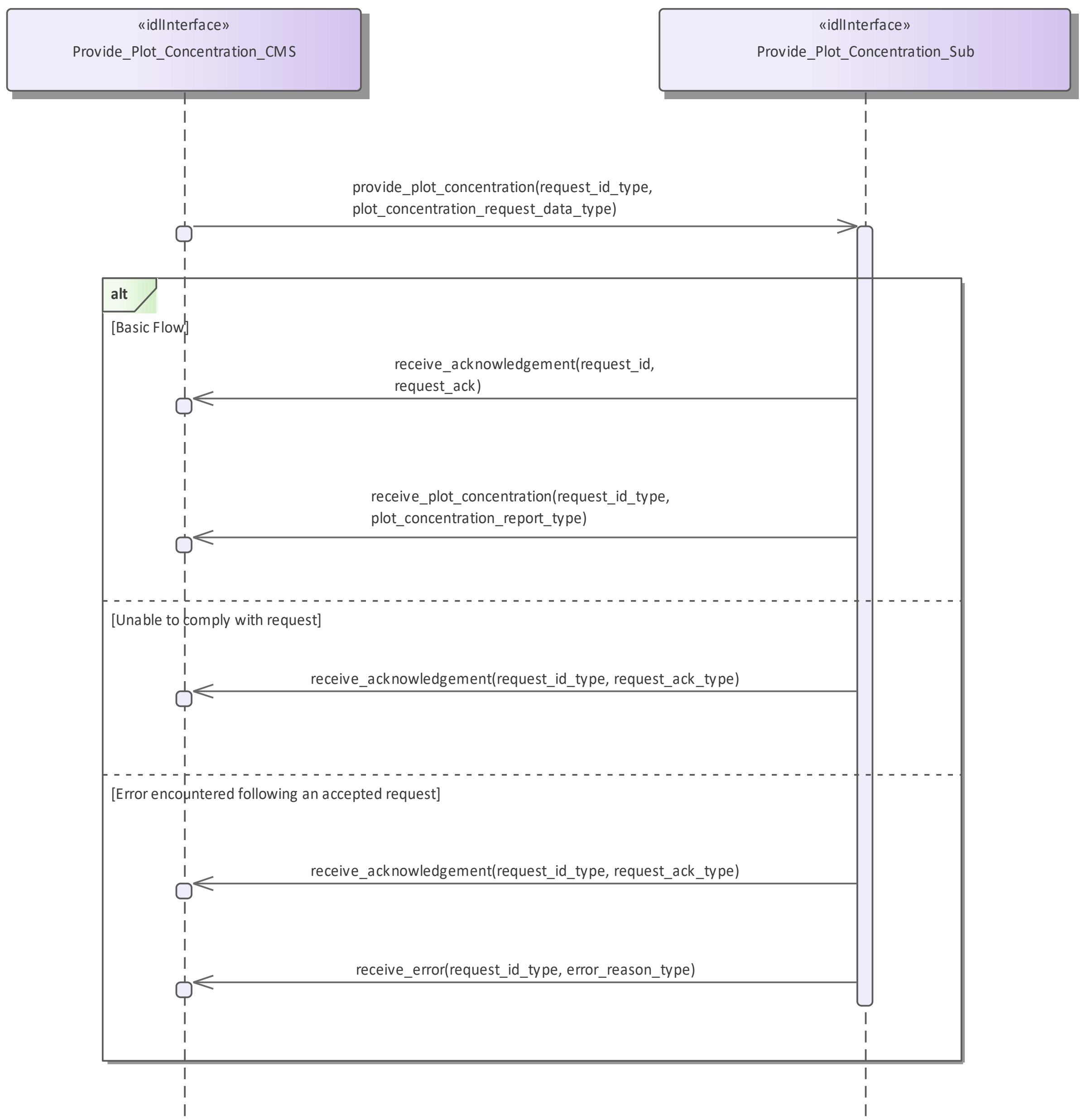


Figure 7.145 Provide Plot Concentration - Report Requested by CMS (Interaction diagram)

Flow of events which depicts a subsystem that reports plot concentration following an explicit request from the CMS (also depicts alternate rejection and error paths).

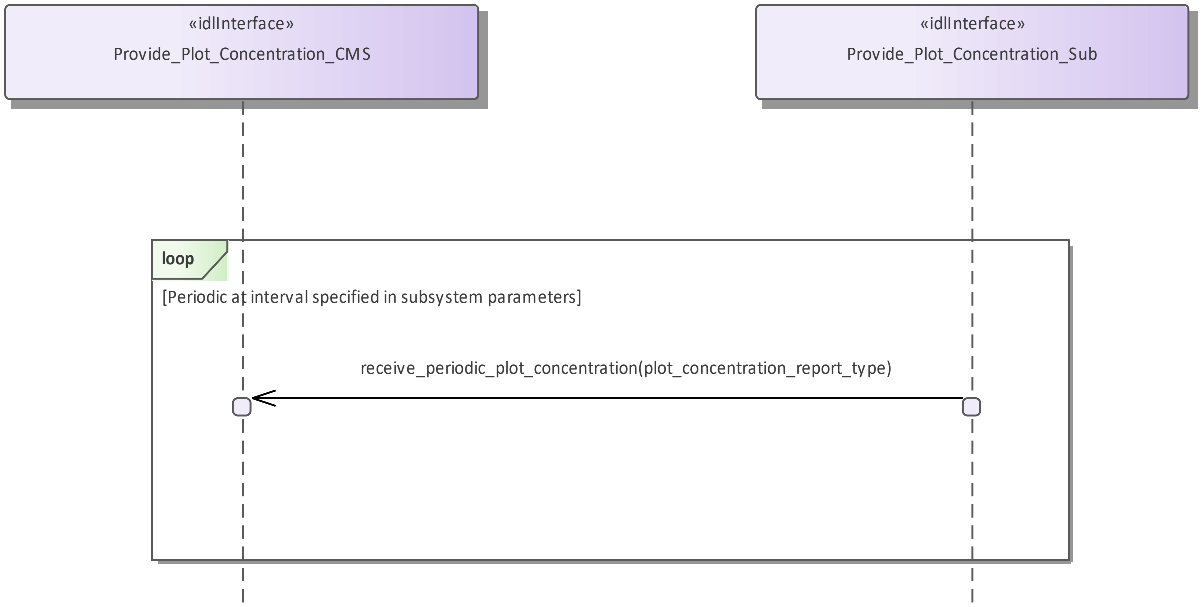


Figure 7.146 Provide Plot Concentration - Periodic (Interaction diagram)

Flow of events which depicts a subsystem that periodically reports plot concentration reports (without the need for a CMS request).

7.9.1.2 Provide_Clutter_Assessment

Parent Package: Clutter_Reporting

Contains operations and sequence diagrams for the Provide Clutter Assessment interface.

Provide_Clutter_Assessment_CMS

Type: Interface

Package: Provide_Clutter_Assessment

The radar reports visible clutter to the combat management system. The report shall include a map (collection of cells) with information on range, azimuth, elevation and intensity in platform relative coordinates. Clutter may be classified by type, Land, Sea, Weather (optional), etc.. Intensity may be indicated by linear signal-to-noise ratio (SNR), log-linear SNR, linear power received, log-linear power received (e.g. dBm, dBW), linear Radar Cross Section (square meters), or log-linear RCS (dbsm).

For radars which report clutter assessment without a CMS request, the CMS shall begin to receive reports upon registration of the Provide Clutter Assessment interface.

Pre-condition: Radar is in ONLINE State:

Pre-condition: The Radar is capable of distinguishing clutter from targets.:

Post-condition: None:

Table 7.321 - Methods of Interface Provide_Clutter_Assessment_CMS

Method	Notes	Parameters
receive_clutter_assessment()	Interface used by the CMS to receive a requested clutter assessment report from the subsystem.	request_id_type **request_id** clutter_report_type **clutter_report**
receive_periodic_clutter_assessment( )	Interface used by CMS to receive periodic clutter assessment reports from the subystem.	clutter_report_type **clutter_report**

Provide_Clutter_Assessment_Sub

Type: Interface
Package: Provide_Clutter_Assessment

Table 7.322 - Methods of Interface Provide_Clutter_Assessment_Sub

Method	Notes	Parameters
provide_clutter_assessment()	Interface used by the subsystem to receive a clutter assessment request from the CMS.	request_id_type **request_id** clutter_assessment_request_type **clutter_request**

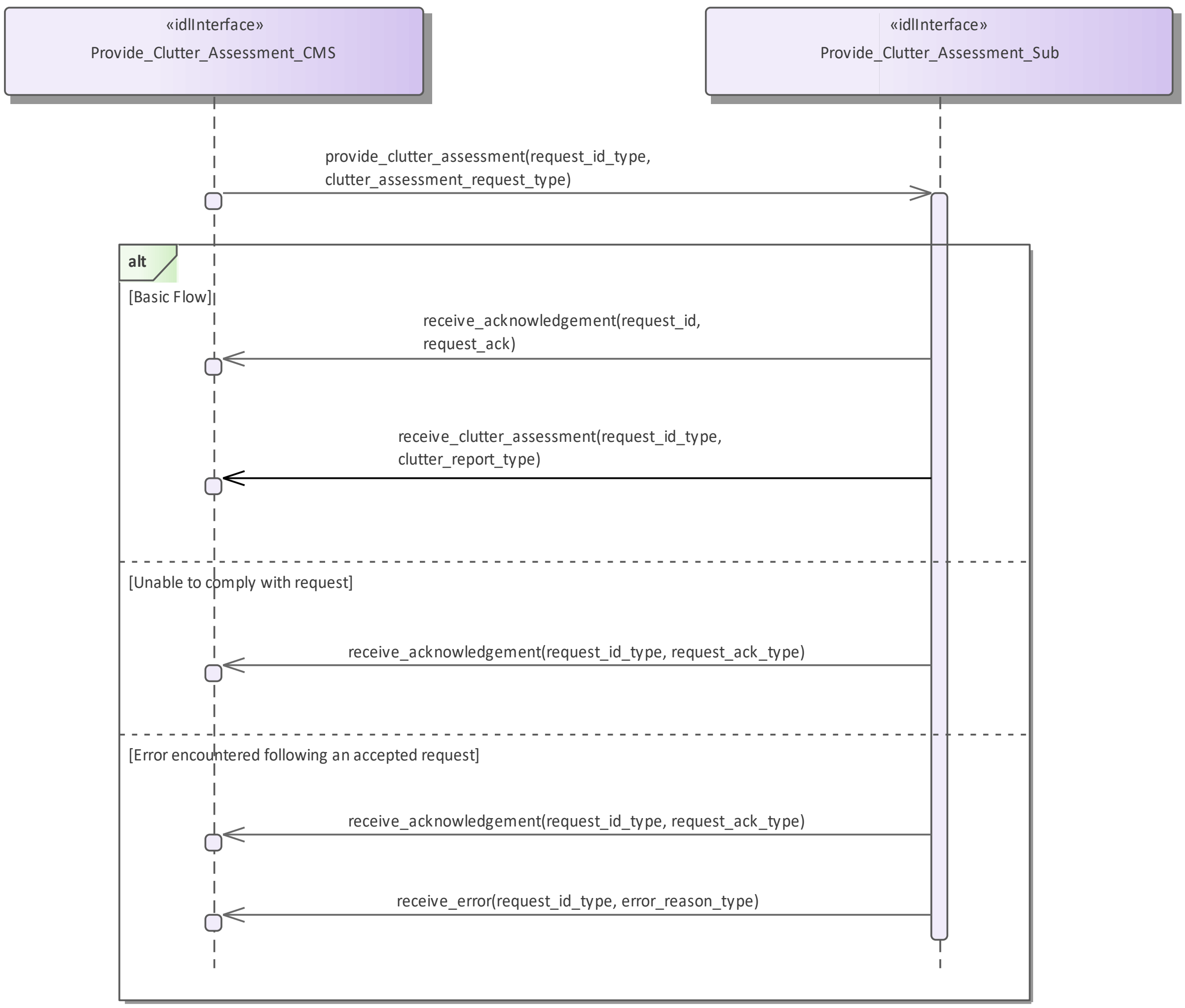


Figure 7.147 Provide Clutter Assessment (Interaction diagram)

Flow of events which depicts a subsystem that reports a clutter assessment following an explicit request from the CMS (also depicts alternate rejection and error paths).

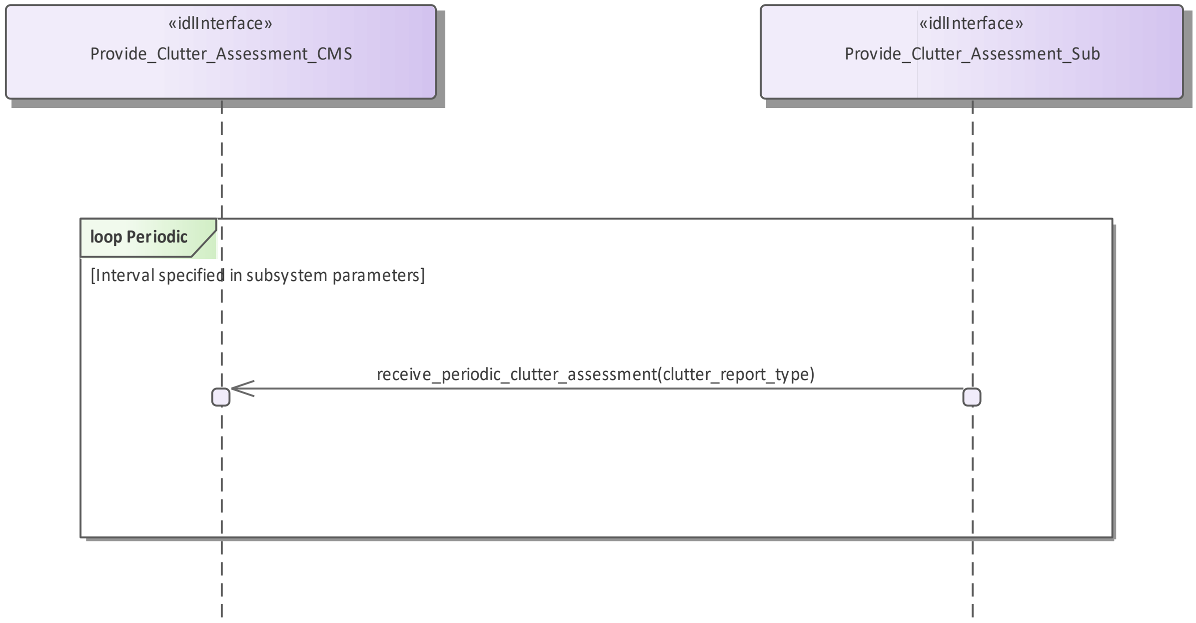


Figure 7.148 Periodic Clutter Reporting (Interaction diagram)

Flow of events which depicts a subsystem that periodically reports a clutter assessment (without the need for a CMS request).

7.9.2 Media_Streaming

Parent Package: Sensor_Services
This package contains interfaces for the Media Streaming service.

7.9.2.1 Allocate_Tracks_To_Stream

Parent Package: Media_Streaming

Allocate_Tracks_to_Stream_CMS
Type: Interface
Package: Allocate_Tracks_To_Stream
This service allows the CMS to receive the allocation of sensor tracks to media streams

Table 7.323 - Methods of Interface Allocate_Tracks_to_Stream_CMS

Method	Notes	Parameters
report_media_stream_allocation()	For a sensor to report on the allocation of tracks to one of its media streams	media_allocation_type **allocation** The allocation of sensor tracks to a media stream

Allocate_Tracks_To_Stream_Sub
Type: Interface
Package: Allocate_Tracks_To_Stream

Table 7.324 - Methods of Interface Allocate_Tracks_To_Stream_Sub

Method	Notes	Parameters
Add_Track_To_Stream()	A request to add the sensor's track to the specified stream.	media_stream_id_type **Stream_Id** sensor_track_id_type **Track_Id** request_id_type **Request_Id**

Remove_Track_From_Stream()	A request to remove the sensor's track from the specified stream.	media_stream_id_type **Stream_Id** sensor_track_id_type **Track_Id** request_id_type **Request_Id**
Add_All_Tracks_To_Stream()	A request to add all the sensor's tracks to the specified stream.	media_stream_id_type **Stream_Id** request_id_type **Request_Id**
Remove_All_Tracks_From_Stream()	A request to remove all the sensor's tracks from the specified stream.	media_stream_id_type **Stream_Id** request_id_type **Request_Id**

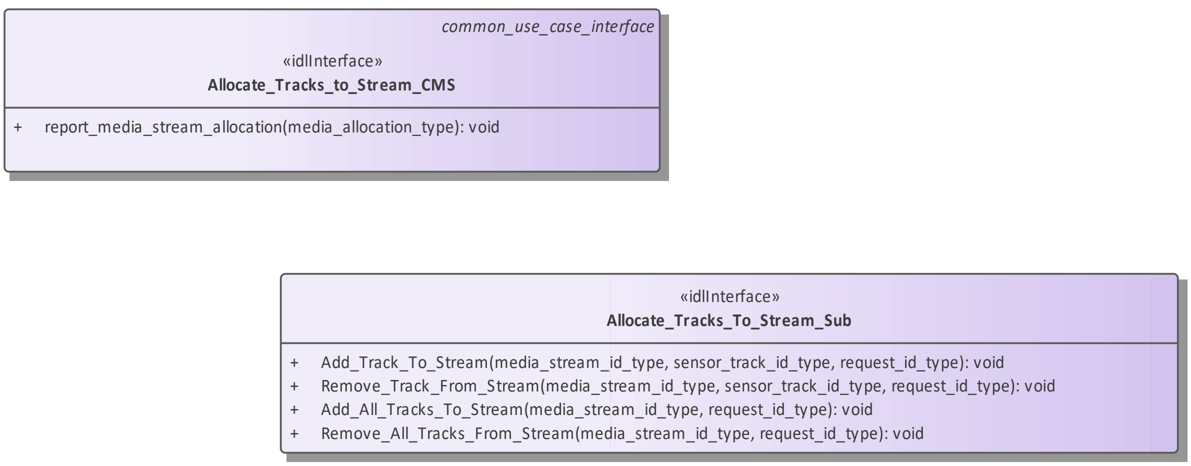


Figure 7.149 Allocate_Tracks_To_Stream (Class diagram)

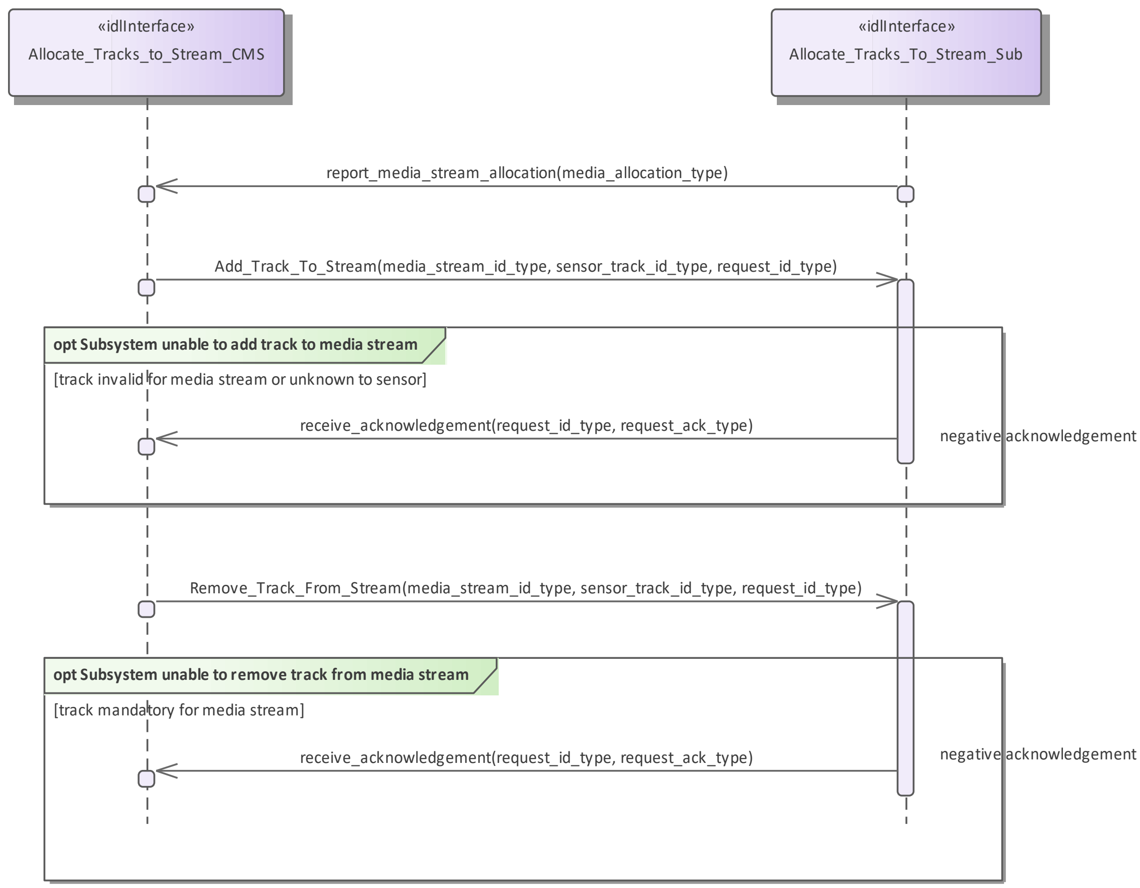


Figure 7.150 Allocate_Tracks_To_Stream - Alternate Flow (Interaction diagram)

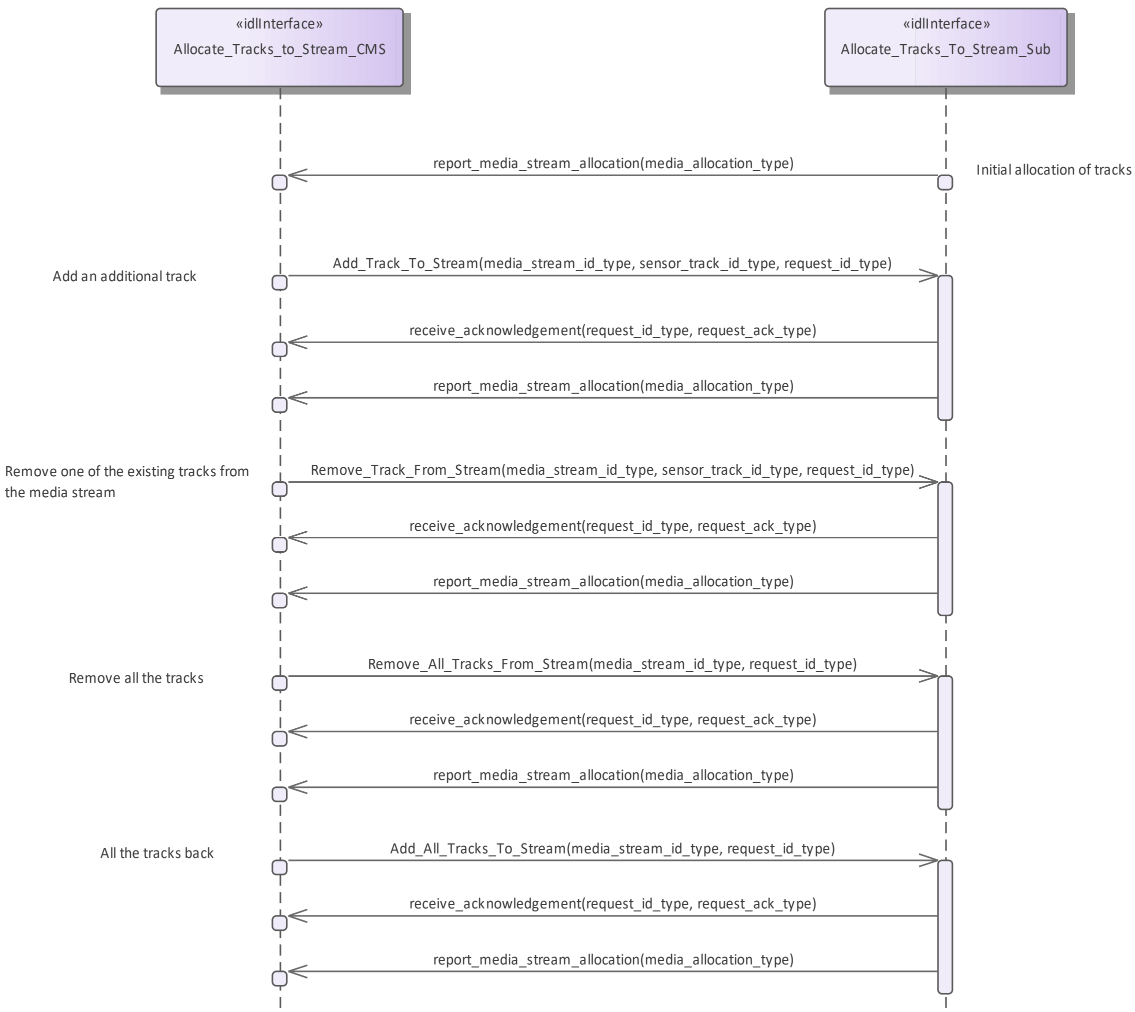


Figure 7.151 Allocate_Tracks_To_Stream - Basic Flow (Interaction diagram)

7.9.2.2 Configure_Media_Streams

Parent Package: Media_Streaming

Configure_Media_Streams_CMS
Type: Interface
Package: Configure_Media_Streams
This service allows the CMS to be informed about the configuration of media streams provided by the sensor.

Table 7.325 - Methods of Interface Configure_Media_Streams_CMS

Method	Notes	Parameters
configure_media_stream()		media_stream_metadata_type **metadata** The metadata for the stream

Configure_Media_Streams_Sub
Type: Interface
Package: Configure_Media_Streams

Figure 7.152 Configure_Media_Streams (Interaction diagram)

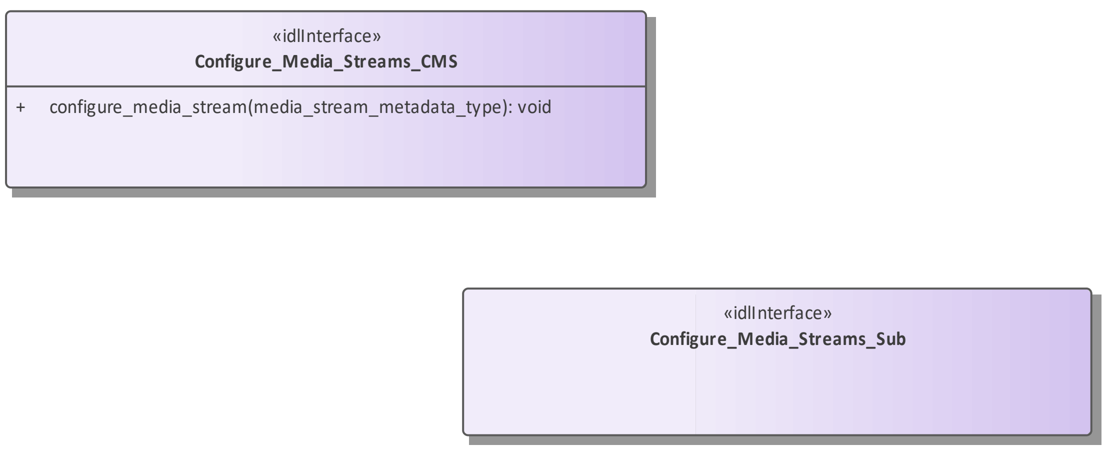


Figure 7.153 Configure_Media_Streams (Class diagram)

7.9.3 Search

Parent Package: Sensor_Services
This package contains interfaces for the Search service.

7.9.3.1 Perform_Cued_Search

Parent Package: Search

Perform_Cued_Search_CMS
Type: Interface
Package: Perform_Cued_Search
The CMS Search Interface.
The subsystem is requested to undertake a cued search in the requested cue volume or to the requested track. The cue may be 1D (azimuth only), 2D (has an additional elevation constraint), 3D (has a further range constraint) or 4D (has a further target velocity constraint). The response of the subsystem is either to reject the cued search request if it is invalid within the current mode/configuration or to provide a cue request reply containing data relating to any resulting tracks.
Depending upon the individual radar it may be possible to predefine a cued search waveform
The cued search request may contain azimuth, elevation and range data along with time of the positional data.
Pre-condition: Technical State: The Subsystem is in Technical State ONLINE.

Pre-condition:	Mastership: The CMS has Mastership
Pre-condition:	Subsystem Services: The Provide Subsystem Services Service has been executed successfully.
Post-condition:	Success: The CMS has received a 'Cued Search Report'
Post-condition:	Failure: The CMS has not received a 'Cued Search Report'

Table 7.326 - Methods of Interface Perform_Cued_Search_CMS

Method	Notes	Parameters
report_cued_search_result()	Send a report to the CMS containing the results of a previously cued search.	cued_search_report_type **result_report** The result of the search. request_id_type **request_id** The unique id relating to this cued search request as supplied by the CMS.

Perform_Cued_Search_Sub
Type: Interface
Package: Perform_Cued_Search
The Subsystem Search Interface.

Table 7.327 - Methods of Interface Perform_Cued_Search_Sub

Method	Notes	Parameters
perform_cued_search()	Request to subsystem to perform a cued search in accordance with the given set of constraints.	cued_search_cue_type **constraint** The details of the constraints on where the radar is to look for tracks. request_id_type **request_id** The unique id for this request. The radar includes this in all replies relating to this request.
perform_cue_to_track()	Request to subsystem to perform a cue to the position of a track produced by a different subsystem.	sensor_track_id_type **sensor_track_id** The identifier of the track to cue to. string **subsystem_name** The name of the subsystem that produced the track to cue to. request_id_type **request_id** The unique id for this request. The radar includes this in all replies relating to this request.
perform_surveillance()		surveillance_task_type **surveillance_task** The surveillance task to be performed request_id_type **request_id** The unique id for this request. The sensor includes this in all replies relating to this request.
stop_surveillance()		request_id_type **request_id** The unique id for this request. The sensor includes this in all replies relating to this request.
stop_cued_search()	Request to subsystem to stop a cued	request_id_type **request_id** The

	search.	unique id for this request. The sensor includes this [illegible]ll replies relating to this request.

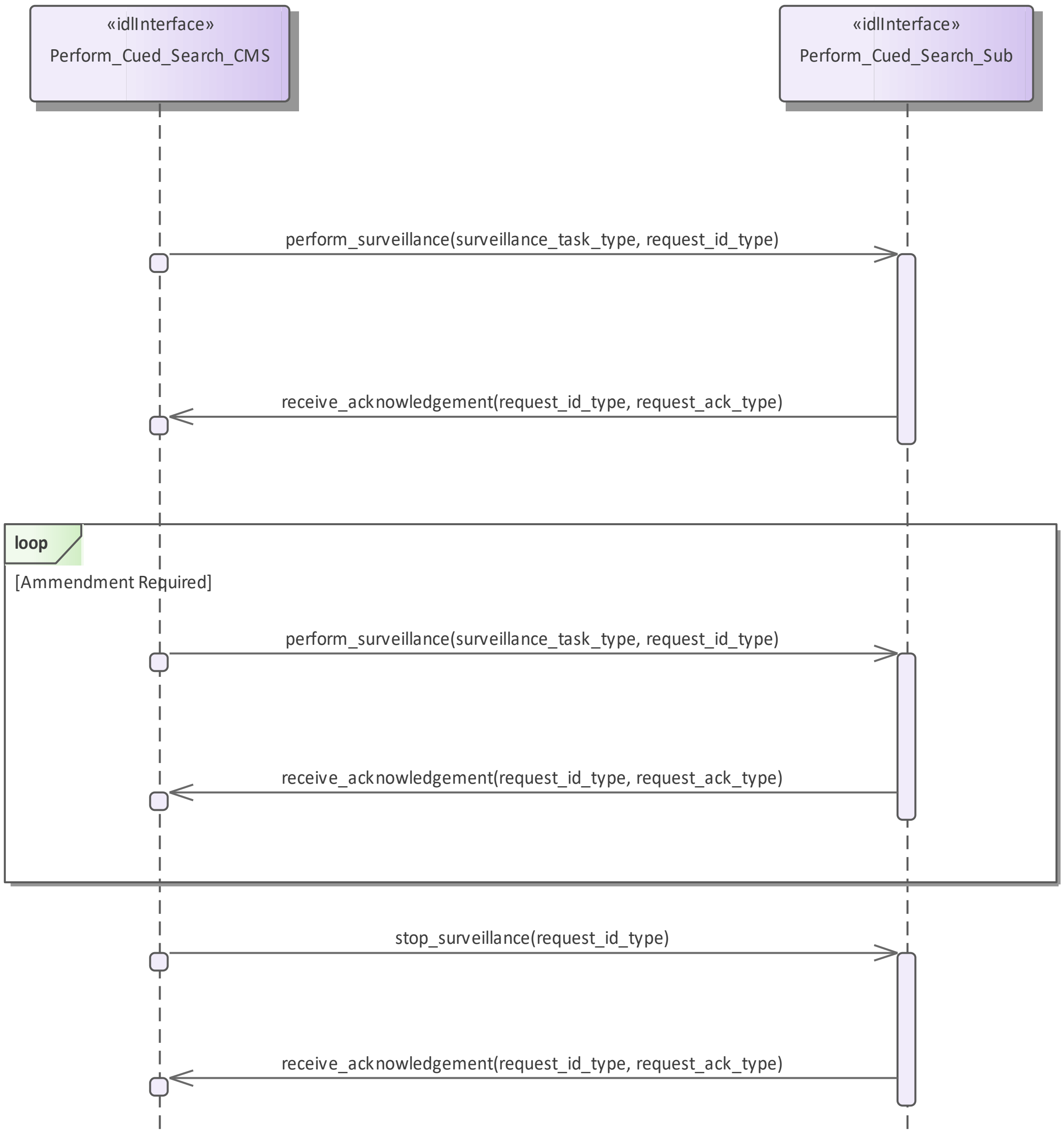


Figure 7.154 Basic Flow - Perform Surveillance (Interaction diagram)

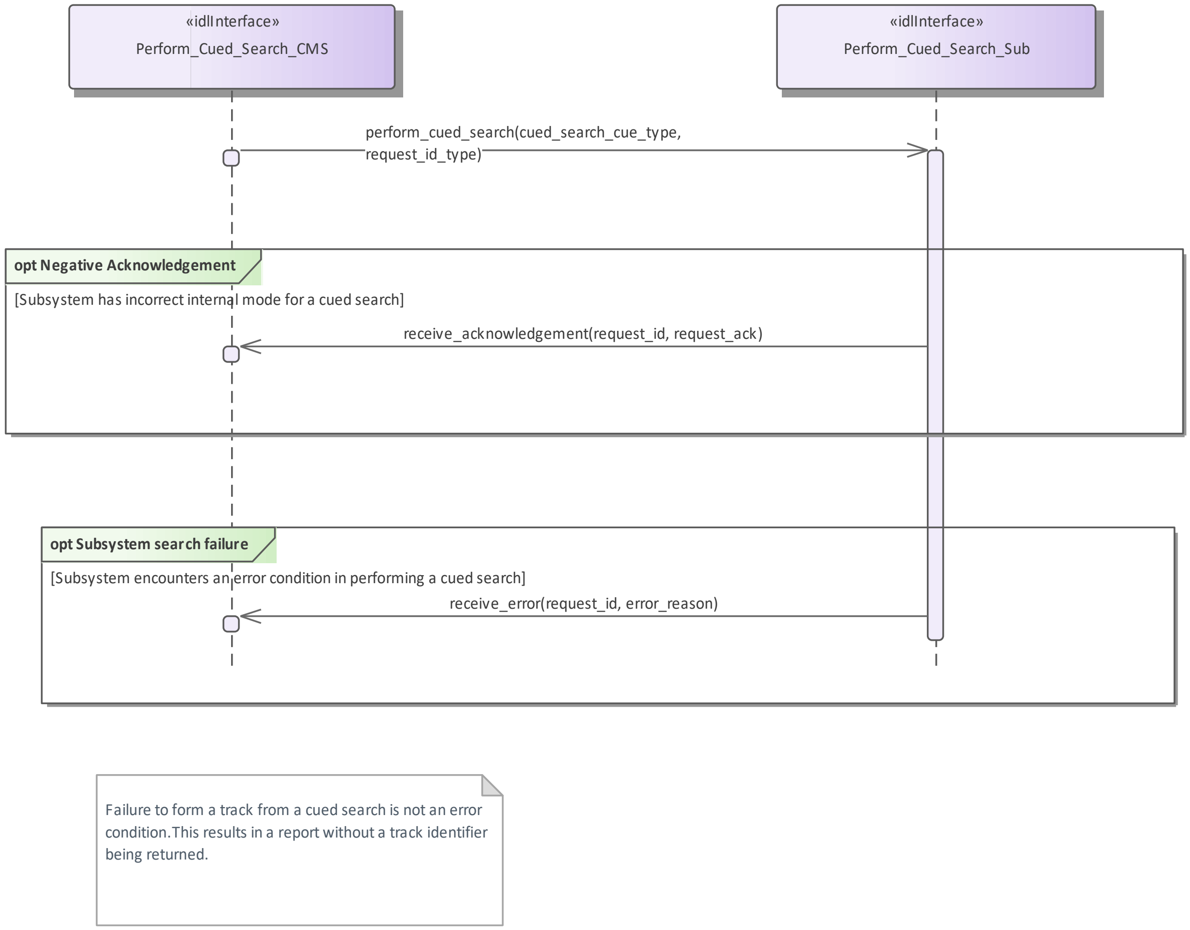


Figure 7.155 Alternative Flow - Sensor does not Perform Cued Search (Interaction diagram)

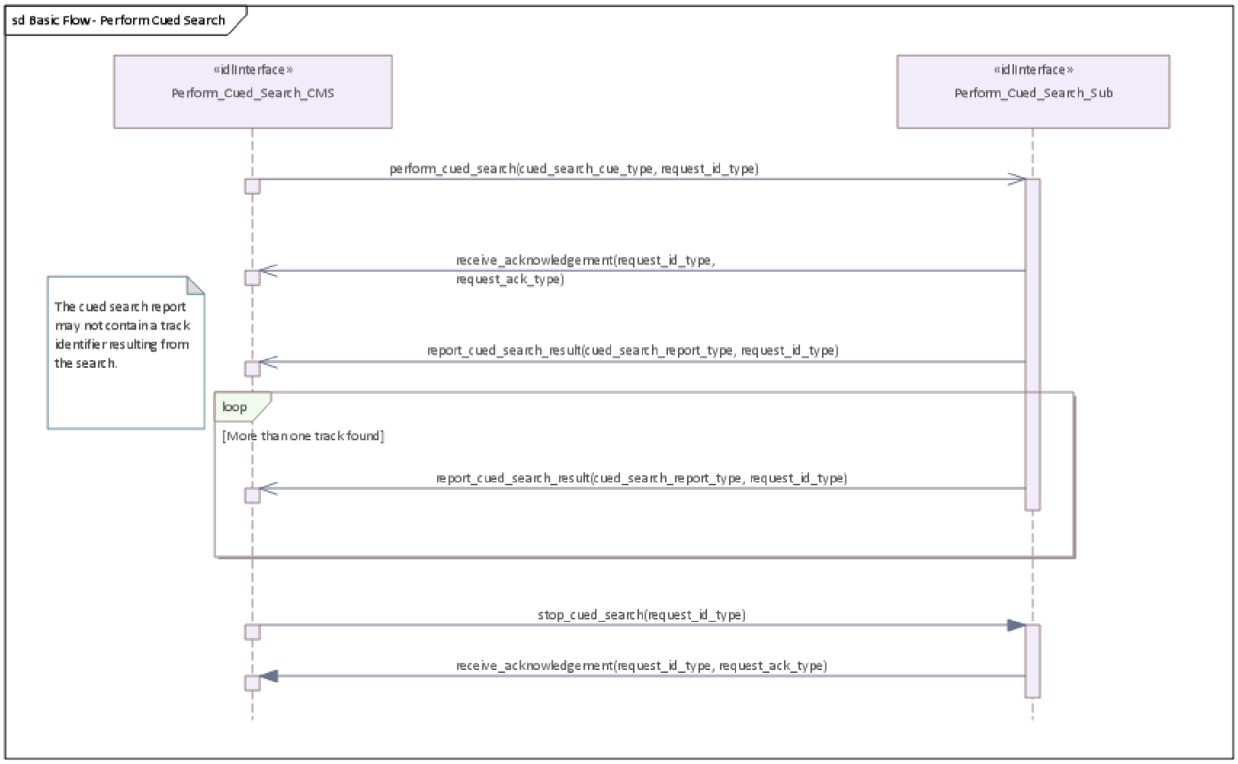


Figure 7.156 Basic Flow - Perform Cued Search (Interaction diagram)

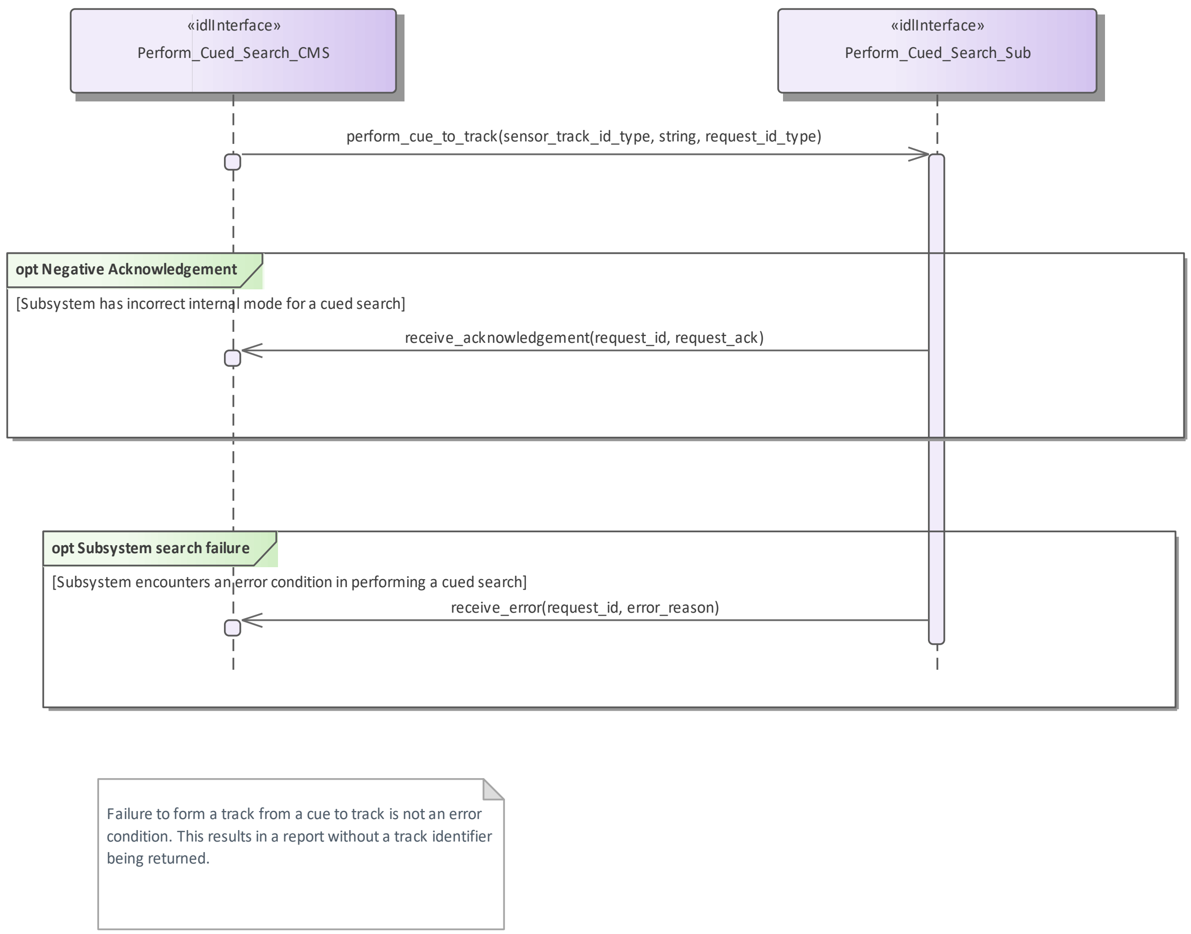


Figure 7.157 Alternative Flow - Sensor does not Perform Cued To Track (Interaction diagram)

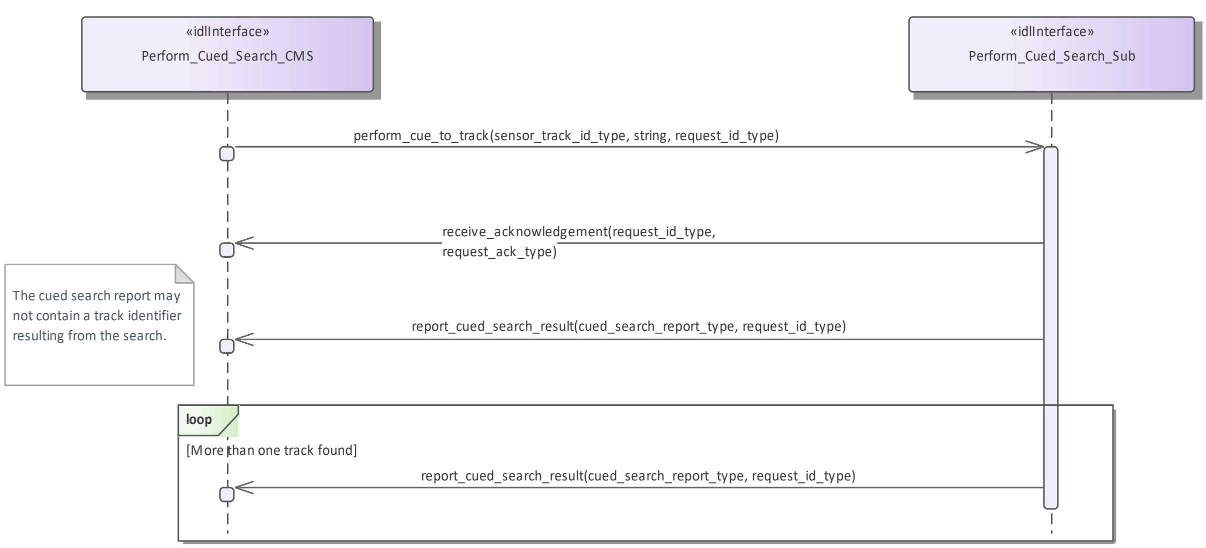


Figure 7.158 Basic Flow - Perform Cued To Track (Interaction diagram)

7.9.4 Sensor_Assessment

Parent Package: Sensor_Services
This package contains interfaces for the Sensor Assessment service.

Figure 7.159 Sensor_Assessment (Class diagram)

7.9.4.1 Assess_Sensor_Plot

Parent Package: Sensor_Assessment

Provide_Sensor_Plot_Assessment_CMS
Type: Interface
Package: Assess_Sensor_Plot
The interface for a sensor to provide assessments (identification and classification) of sensor plots to the CMS. It is expected that the assessment relates to matching the plot's measurement parameters to reference data. The sensor provides a set of mode, equipment and/or platform matches relating to a particular plot (referenced by the plot id). Therefore the sensor plot must have a plot_id attribute defined.

Table 7.328 - Methods of Interface Provide_Sensor_Plot_Assessment_CMS

Method	Notes	Parameters
write_equipment_assessment()	To report on the overall equipment assessment for a sensor plot.	sensor_plot_equipment_assessment_type **equipment_assessment** The assessment of the equipment to which the sensor track's data may correspond.
write_platform_assessment()	To report on the overall platform assessment for a sensor plot.	sensor_plot_platform_assessment_type **platform_assessment** The assessment of the platform to which the sensor track's data may correspond.
write_mode_assessment()	To report on the overall mode assessment for a sensor plot.	sensor_plot_mode_assessment_type **assessment** The overall assessment of mode (of the detected equipment) for a sensor track

Provide_Sensor_Plot_Assessment_Sub
Type: Interface
Package: Assess_Sensor_Plot
The interface by which a CMS can control the sensor's assessment of the plot data. The sensor matches parametric measurements to reference data and then reports each of these sets as an assessment for each plot for the categories of equipment modes, equipment marks (build standards / versions) and platform instances (or platform classes). The sensor also reports what it has assessed to be the best match.

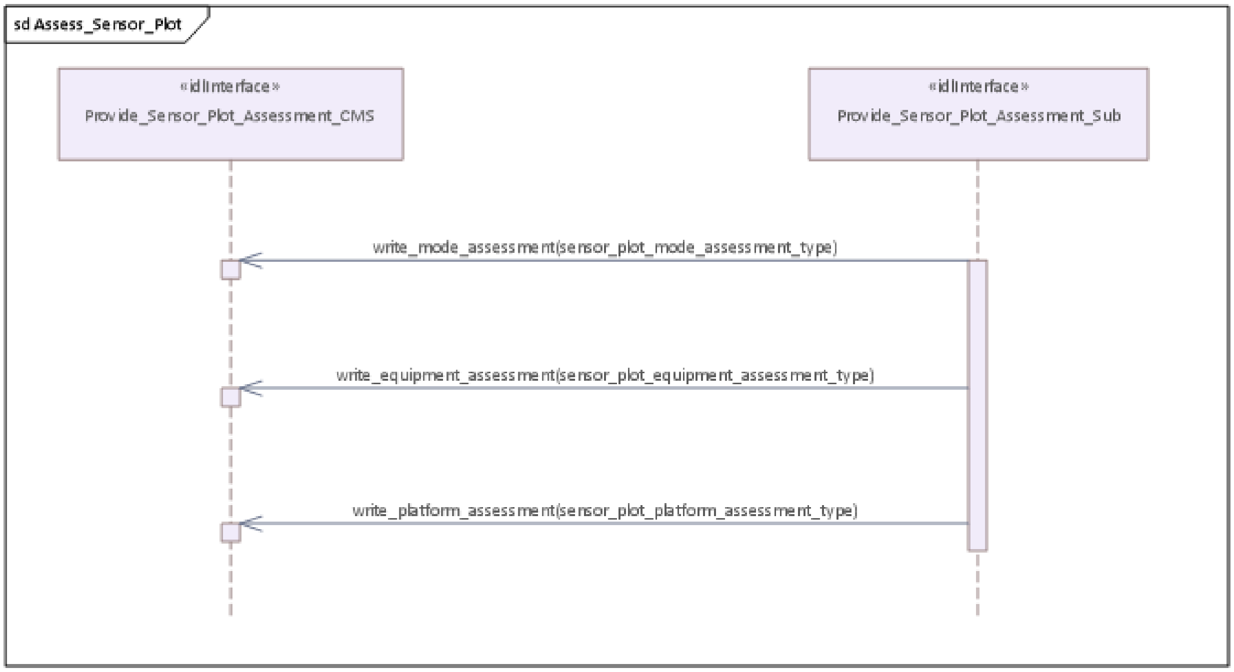


Figure 7.160 Assess_Sensor_Plot (Interaction diagram)

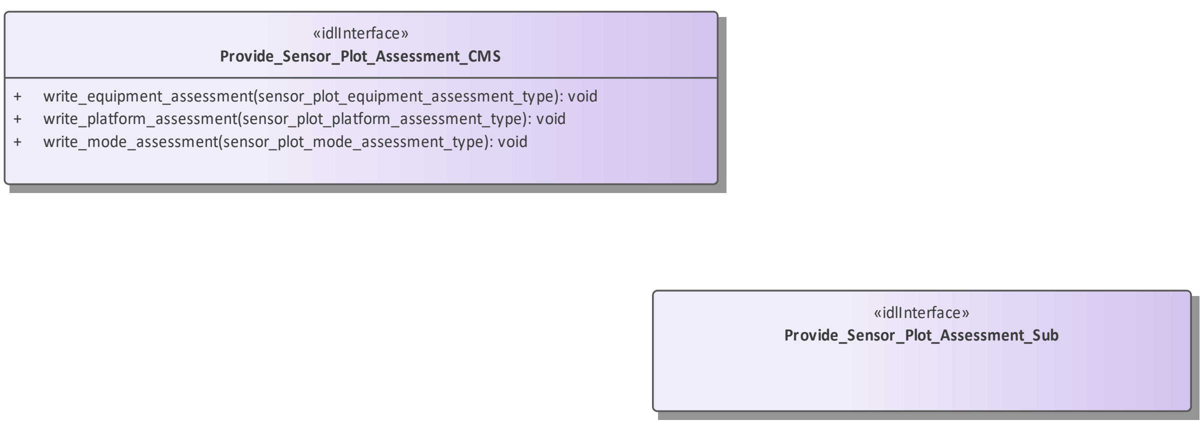


Figure 7.161 Assess_Sensor_Plot (Class diagram)

7.9.4.2 Assess_Sensor_Track

Parent Package: Sensor_Assessment

Provide_Sensor_Track_Assessment_CMS
Type: Interface
Package: Assess_Sensor_Track
The interface for a sensor to provide assessments (identification and classification) of sensor tracks to the CMS. The sensor matches parametric measurements to reference data and then reports each of these sets as an assessment for each track for the categories of equipment modes, equipment marks (build standards / versions) and platform instances (or platform classes). The sensor also reports what it has assessed to be the best match and the match

currently selected. The currently selected match influences the attributes reported for the sensor track (including its recognition and identification).
The CMS uses the select and deselect methods to set or override the match that is selected for an assessment of a sensor track.
Sensors report a track to the CMS using the Track Reporting use case before providing an assessment. Assessments are only reported for tracks whilst the sensor track is in the TRACKED track state.

Table 7.329 - Methods of Interface Provide_Sensor_Track_Assessment_CMS

Method	Notes	Parameters
write_equipment_assessment()	To report on the overall equipment assessment for a sensor track.	sensor_track_equipment_assessment_type **equipment_assessment** The assessment of the equipment to which the sensor track's data may correspond.
write_platform_assessment()	To report on the overall platform assessment for a sensor track.	sensor_track_platform_assessment_type **platform_assessment** The assessment of the platform to which the sensor track's data may correspond.
write_multipath_set()	To report on the assessment of a set of sensor track representing the same real world object through multiple paths.	multipath_set_type **set**
write_mode_assessment()	To report on the overall mode assessment for a sensor track.	sensor_track_mode_assessment_type **assessment** The overall assessment of mode (of the detected equipment) for a sensor track

Provide_Sensor_Track_Assessment_Sub
Type: Interface
Package: Assess_Sensor_Track
The interface by which a CMS can control the sensor's assessment of the track data.

Table 7.330 - Methods of Interface Provide_Sensor_Track_Assessment_Sub

Method	Notes	Parameters
select_equipment_assessment()	The CMS selects a particular equipment match as being the authoritative assessment for the sensor track with regard to the equipment it is a detection of. The Subsystem, thereafter reports the sensor track in accordance with this assessment.	request_id_type **request_id** The unique identifier of the request to select the match long **match_id** The identifier of the match to be selected sensor_track_id_type **sensor_track_id** The sensor track to which the assessment applies
select_platform_assessment()	The CMS selects a particular platform match as being the authoritative assessment for the sensor track with regard to the platform it is a detection of. The Subsystem, thereafter reports the sensor track in accordance with this assessment.	request_id_type **request_id** The unique identifier of the request to select the match long **match_id** The identifier of the match to be selected sensor_track_id_type **sensor_track_id** The sensor track to which the assessment applies

select_mode_assessment()	The CMS selects a particular mode match as being the authoritative assessment for the sensor track with regard to the equipment mode it is a detection of. The Subsystem, thereafter reports the sensor track in accordance with this assessment.	request_id_type **request_id** The unique identifier of the request to select the match long **match_id** The identifier of the match to be selected sensor_track_id_type **sensor_track_id** The sensor track to which the assessment applies
deselect_equipment_assessment()	The CMS deselects equipment match as being the authoritative assessment for the sensor track with regard to the equipment it is a detection of. The Subsystem, stops reporting the sensor track in accordance with the previously selected assessment.	request_id_type **request_id** The unique identifier of the request to deselect matches sensor_track_id_type **sensor_track_id** The sensor track to which the assessment applies
deselect_platform_assessment()	The CMS deselects platform match as being the authoritative assessment for the sensor track with regard to the platform it is a detection of. The Subsystem, stops reporting the sensor track in accordance with the previously selected assessment.	request_id_type **request_id** The unique identifier of the request to deselect matches sensor_track_id_type **sensor_track_id** The sensor track to which the assessment applies
deselect_mode_assessment()	The CMS deselects mode match as being the authoritative assessment for the sensor track with regard to the equipment it is a detection of. The Subsystem, stops reporting the sensor track in accordance with the previously selected assessment.	request_id_type **request_id** The unique identifier of the request to deselect matches sensor_track_id_type **sensor_track_id** The sensor track to which the assessment applies

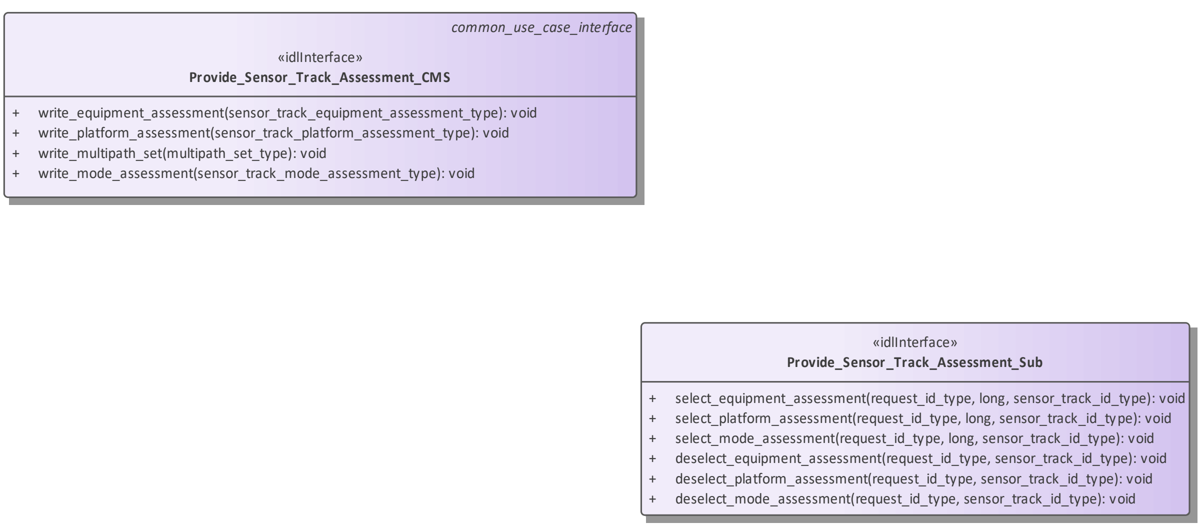


Figure 7.162 Assess_Sensor_Track (Class diagram)

Figure 7.163 Assess_Sensor_Track - alternate flows (Interaction diagram)

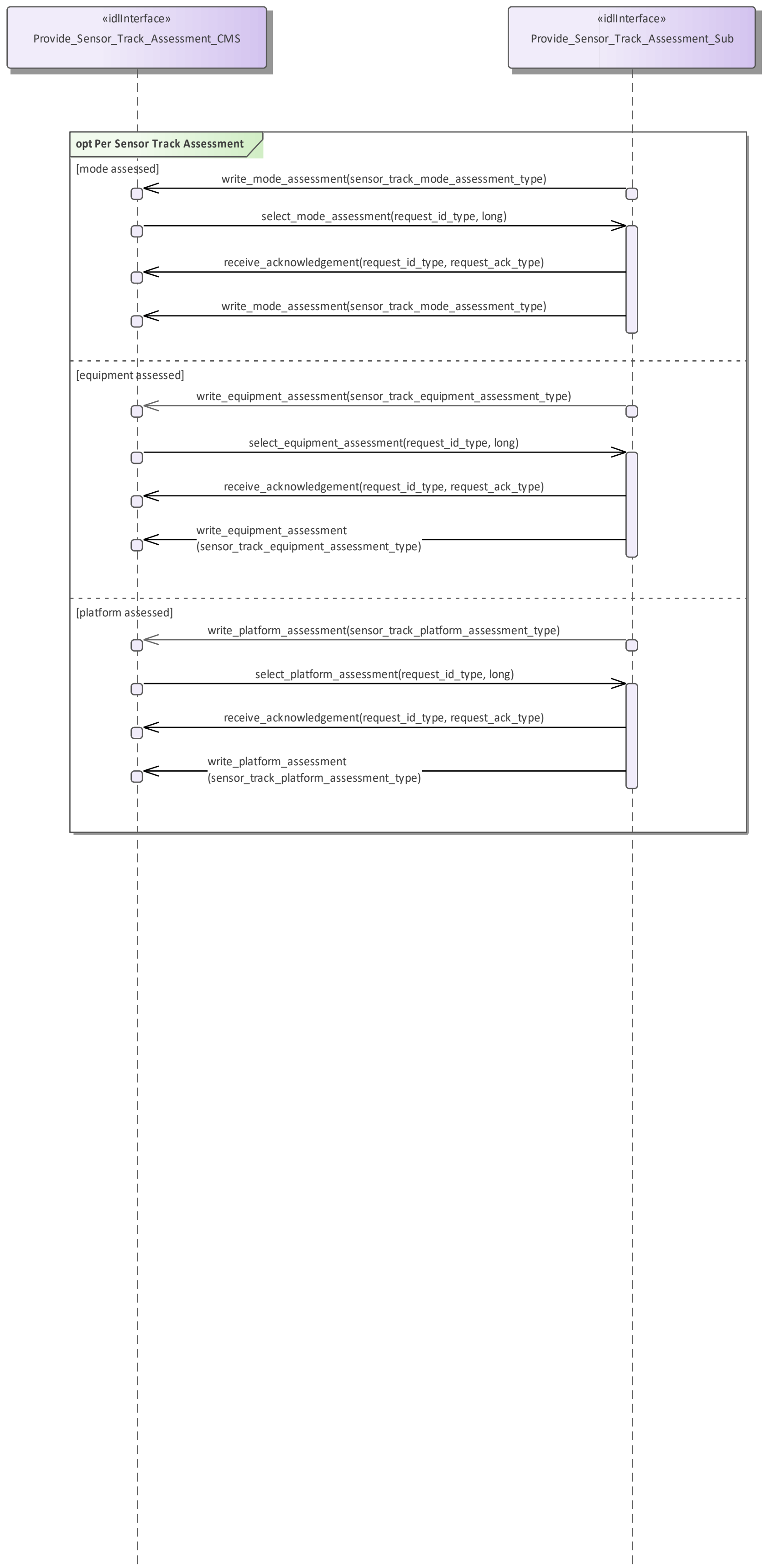

«idlInterface»
Provide_Sensor_Track_Assessment_CMS
«idlInterface»
Provide_Sensor_Track_Assessment_Sub
opt Per Sensor Track Assessment
[mode assessed]
write_mode_assessment(sensor_track_mode_assessment_type)
select_mode_assessment(request_id_type, long)
receive_acknowledgement(request_id_type, request_ack_type)
write_mode_assessment(sensor_track_mode_assessment_type)
[equipment assessed]
write_equipment_assessment(sensor_track_equipment_assessment_type)
select_equipment_assessment(request_id_type, long)
receive_acknowledgement(request_id_type, request_ack_type)
write_equipment_assessment
(sensor_track_equipment_assessment_type)
[platform assessed]
write_platform_assessment(sensor_track_platform_assessment_type)
select_platform_assessment(request_id_type, long)
receive_acknowledgement(request_id_type, request_ack_type)
write_platform_assessment
(sensor_track_platform_assessment_type)

Figure 7.164 Assess_Sensor_Track - assessment and selection (Interaction diagram)

Figure 7.165 Assess_Sensor_Track - equipment deselection (Interaction diagram)

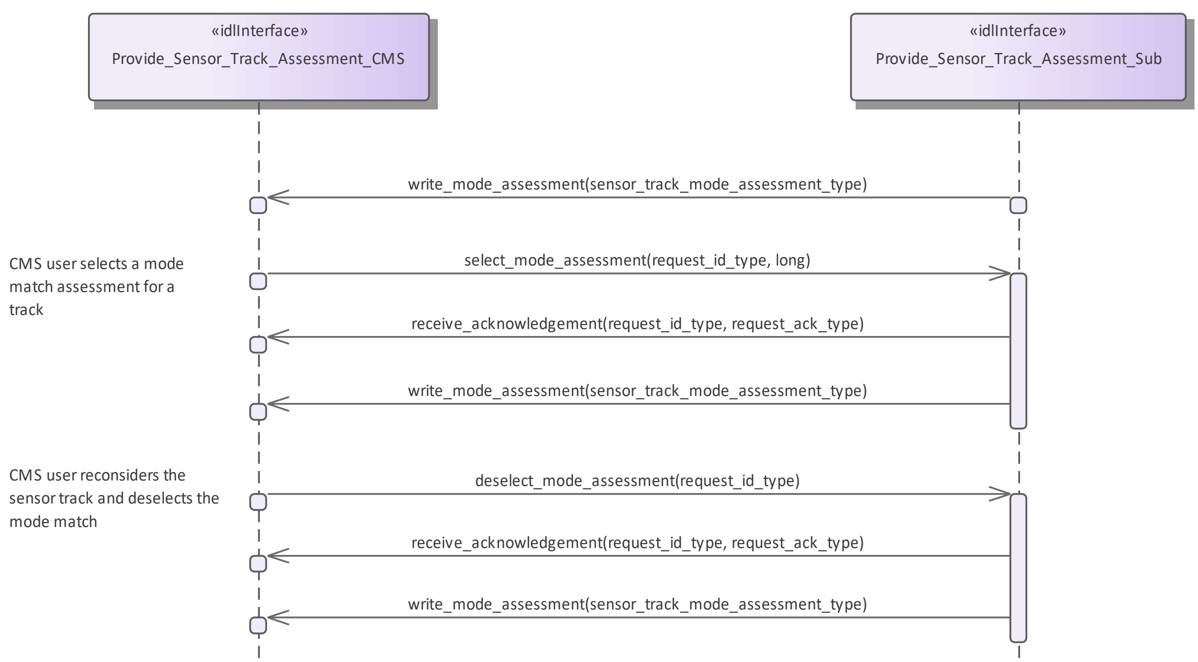


Figure 7.166 Assess_Sensor_Track - mode deselection (Interaction diagram)

Figure 7.167 Assess_Sensor_Track - multipath (Interaction diagram)

Figure 7.168 Assess_Sensor_Track - platform deselection (Interaction diagram)

7.9.5 Supplementary_Measurement

Parent Package: Sensor_Services
This package contains interfaces for the Supplementary Measurement service.

7.9.5.1 Configure_Measurement_Parameters

Parent Package: Supplementary_Measurement

Configure_Measurement_Parameters_CMS
Type: Interface
Package: Configure_Measurement_Parameters
The configuration of measurement parameters allows integrated systems to specify the set of measurement types for which the installed equipment has a measurement capability and semantics associated with these measurement types.

Measurement types have the potential to be classified and also the set of measurement types can be expected to grow as technology advances. Therefore the meaning of the associated identifiers are systems specific and determined from configuration data.

Table 7.331 - Methods of Interface Configure_Measurement_Parameters_CMS

Method	Notes	Parameters
configure_parametric_measurement( )		measurement_parameter_kind_type **parameter** A kind of parameter supported by the sensor

Configure_Measurement_Parameters_Sub
Type: Interface
Package: Configure_Measurement_Parameters

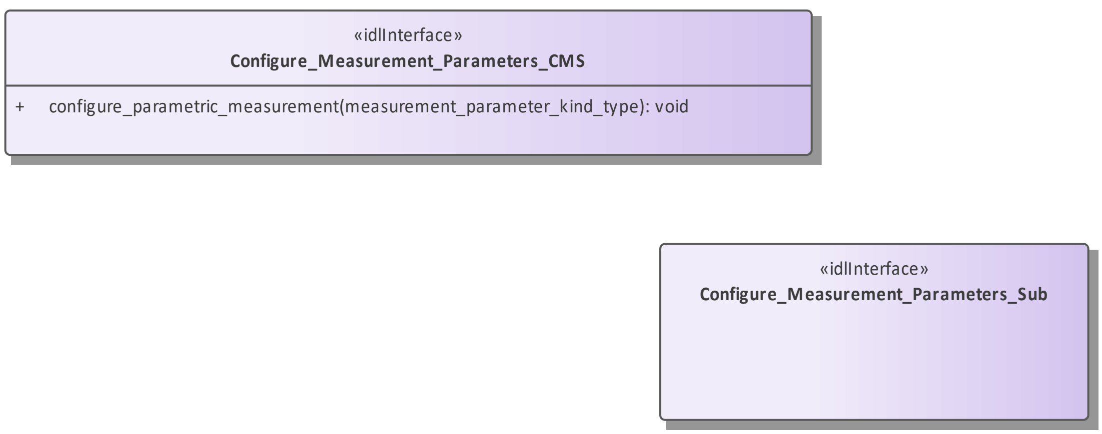


Figure 7.169 Configure_Measurement_Parameters (Class diagram)

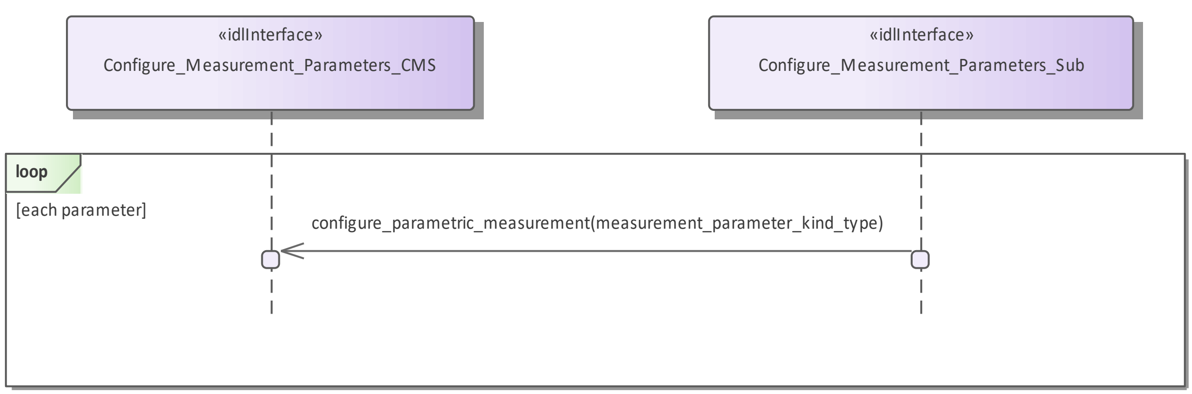


Figure 7.170 Configure_Measurement_Parameters - basic flow (Interaction diagram)

7.9.5.2 Provide_Sensor_Plot_Parameters

Parent Package: Supplementary_Measurement

Provide_Sensor_Plot_Parameters_CMS
Type: Interface

Package: Provide_Sensor_Plot_Parameters
Interface for a sensor to provide its supplementary parametric data with respect to plots to the CMS. A sensor can pass a set of measurements with plot data when reporting to the CMS.

Table 7.332 - Methods of Interface Provide_Sensor_Plot_Parameters_CMS

Method	Notes	Parameters
write_parameter_measurement_set()	For the sensor to report on the measurement of a set of parameters with a plot defined by configuration data.	plot_measurement_parameter_set_type **parameter_set** The sensor's measurement of a set of parameters

Provide_Sensor_Plot_Parameters_Sub
Type: Interface
Package: Provide_Sensor_Plot_Parameters

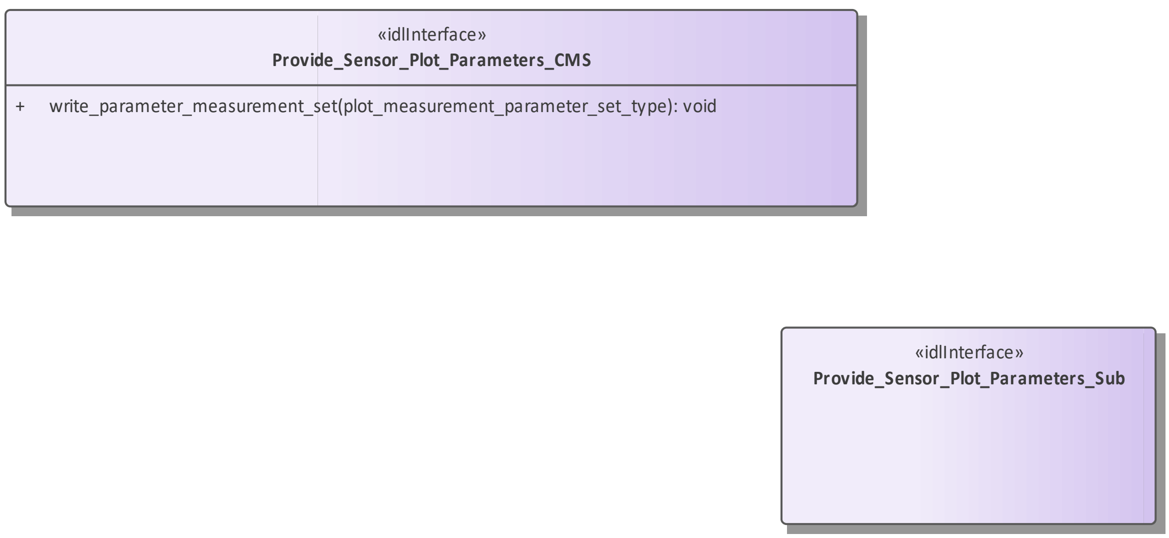


Figure 7.171 Provide_Sensor_Plot_Parameters (Class diagram)

7.9.5.3 Provide_Sensor_Track_Parameters

Parent Package: Supplementary_Measurement

Provide_Sensor_Track_Parameters_CMS
Type: Interface
Package: Provide_Sensor_Track_Parameters
Interface for a sensor to provide its supplementary parametric data with respect to tracks to the CMS. A sensor can pass measurements to the CMS individually or as a set (relating to the same track).
Sensors report a track to the CMS using the Track Reporting use case before reporting any of its supplementary measurements. Supplementary measurements are only reported for tracks whilst the sensor track is in the TRACKED track state.

Table 7.333 - Methods of Interface Provide_Sensor_Track_Parameters_CMS

Method	Notes	Parameters
write_parameter_measurement()	For the sensor to report on the measurement of an individual parameter defined by configuration	track_measurement_parameter_type **parameter** The sensor's measurement of a parameter

	data.	
write_parameter_measurement_set()	For the sensor to report on the measurement of a set of parameters for a track defined by configuration data.	track_measurement_parameter_set_type **parameter_set** The sensor's measurement of a set of parameters

Provide_Sensor_Track_Parameters_Sub
Type: Interface
Package: Provide_Sensor_Track_Parameters

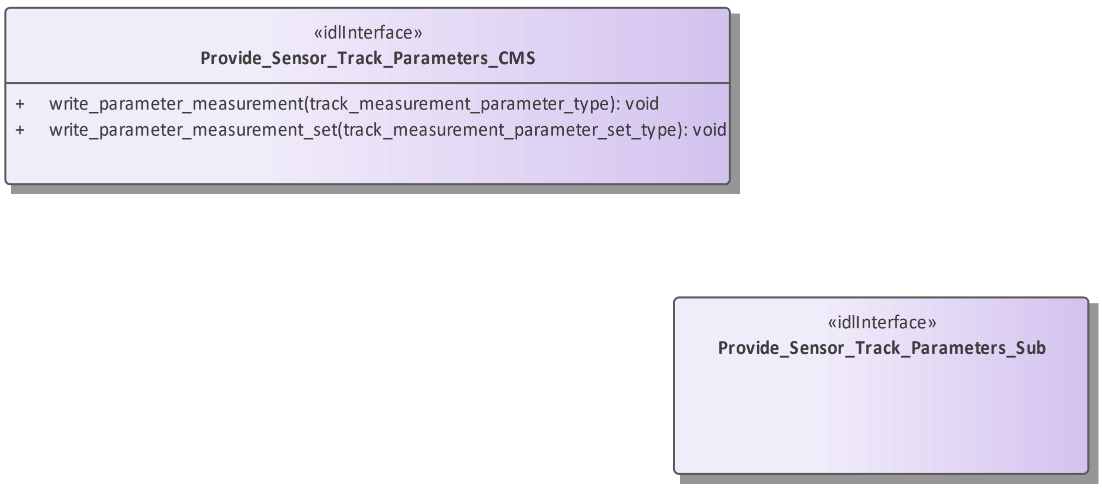


Figure 7.172 Provide_Sensor_Track_Parameters (Class diagram)

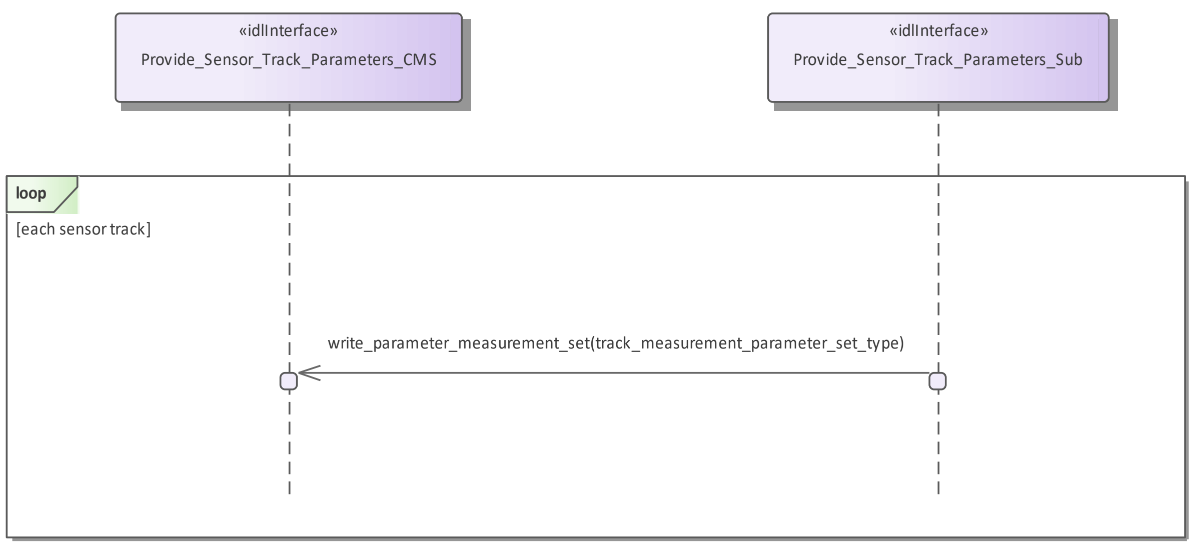


Figure 7.173 Provide_Sensor_Track_Parameters - parameter sets (Interaction diagram)

Figure 7.174 Provide_Sensor_Track_Parameters - single parameters (Interaction diagram)

7.9.6 Plot_Reporting

Parent Package: Sensor_Services
This package contains interfaces for the Plot Reporting service.

7.9.6.1 Filter_Plots

Parent Package: Plot_Reporting
This package contains interfaces for the Filter Plots service.
Filter_Plots_CMS
Type: Interface
Package: Filter_Plots
The interface to the CMS for receiving information relating to the filters used to control which plots are made available to other network segments.
The plot sharing architecture recognizes that connectivity between different platforms hosting sensors may not support the bandwidth required to share all plot and track updates. It is possible for a sensor also to provide the networking functionality in which case it is providing an additional role in the interface.

Table 7.334 - Methods of Interface Filter_Plots_CMS

Method	Notes	Parameters
report_plot_filter()	Reports the parameters of one of the filters that are active for plots in the communication and networking subsystem. Plots are transmitted or received, according to their mode, if they pass the conditions of at least one of the active filters.	filter_id_type **filter_id** The identifier for the filter plot_filter_parameters_type **filter_value** The criteria for the filter
plot_filter_removed()	Reports that a particular plot filter has been removed.	filter_id_type **filter_id** The identifier of the filter removed

Filter_Plots_Sub
Type: Interface
Package: Filter_Plots
The interface to the subsystem for receiving updates to the filters used to control which plots are made available to other network segments.

The plot sharing architecture recognizes that connectivity between different platforms hosting sensors may not support the bandwidth required to share all plot and track updates.
In this use case the subsystem is the network component providing connectivity to other platforms, as distinct from the local sensors providing the plots. It is possible for a sensor to also provide the networking functionality, in which case it is providing an additional role in the interface.

Table 7.335 - Methods of Interface Filter_Plots_Sub

Method	Notes	Parameters
add_plot_filter()	Adds an active filter for plots to the communication and networking subsystem.	request_id_type **request_id** unique reference for the request plot_filter_parameters_type **filter** the values to be used to filter plots by
remove_plot_filter()	Removes a filter for plots from the communication and networking subsystem.	request_id_type **request_id** The unique reference for the request filter_id_type **filter_id** The identifier for the filter to be removed

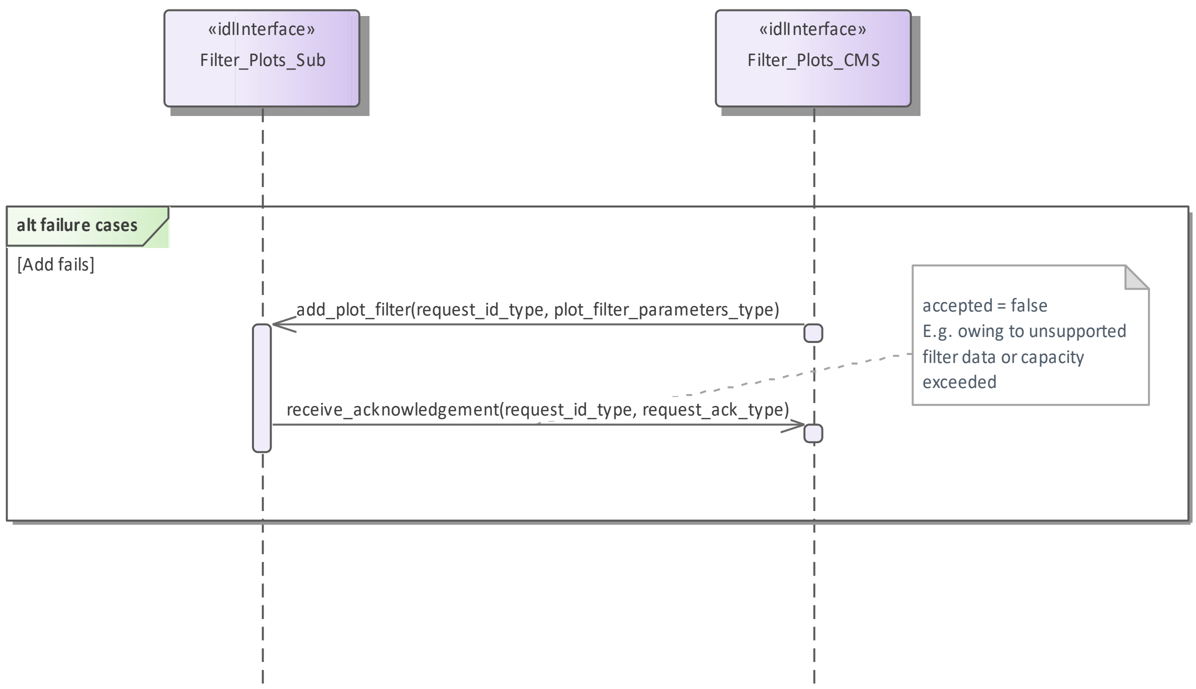


Figure 7.175 Filter_Plots - alternative flows (Interaction diagram)

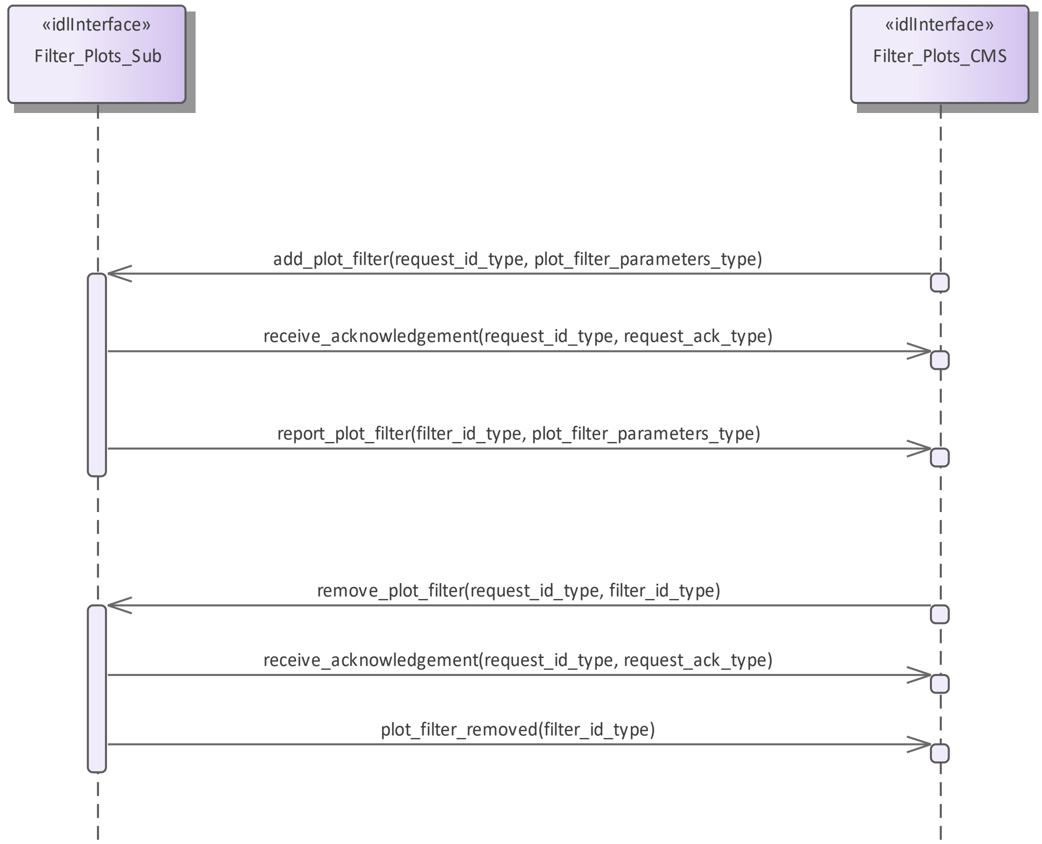


Figure 7.176 Filter_Plots - Basic Flow (Interaction diagram)

7.9.6.2 Provide_Sensor_Characteristics

Parent Package: Plot_Reporting
This package contains interfaces for the Provide Sensor Characteristic service.
Provide_Sensor_Characteristics_CMS
Type: Interface
Package: Provide_Sensor_Characteristics
The interface to the CMS for providing information about the characteristics of a sensor. This enables sensor agnostic processing of sensor data particularly plot data.
Sensor characteristics are sent by the subsystem when it receives a request from the CMS.

Table 7.336 - Methods of Interface Provide_Sensor_Characteristics_CMS

Method	Notes	Parameters
report_sensor_calibration_model()	Method for a sensor subsystem to inform CMS of its calibration model.	request_id_type **request_id** The unique identifier of the request for the sensor's calibration model sensor_calibration_model_type **model** The sensor's calibration model
report_sensor_precision_model()	Method for a sensor subsystem to inform CMS of its precision model.	request_id_type **request_id** The unique identifier of the request for the sensor's precision model sensor_precision_model_type **model** The sensor's model of its precision

report_sensor_stabilization_model()	Method for a sensor subsystem to inform CMS of its stabilization model.	request_id_type **request_id** The unique identifier of the request for the sensor's stabilization model sensor_stabilization_model_type **model** The sensor's model of stabilization characteristics
report_sensor_processing_model()	Method for a sensor subsystem to inform CMS of its processing model.	request_id_type **request_id** The unique identifier of the request for the sensor's processing model sensor_processing_model_type **model** The sensor's model of its own processing algorithms

Provide_Sensor_Characteristics_Sub
Type: Interface
Package: Provide_Sensor_Characteristics
The interface to the Subsystem for requesting sensor characteristics.

Table 7.337 - Methods of Interface Provide_Sensor_Characteristics_Sub

Method	Notes	Parameters
request_characteristics()	This requests the subsystem to send its characteristics to the CMS.	request_id_type **request_id** The unique identifier for the request

Figure 7.177 Basic Flow - Provide_Sensor_Characteristics on request (Interaction diagram)

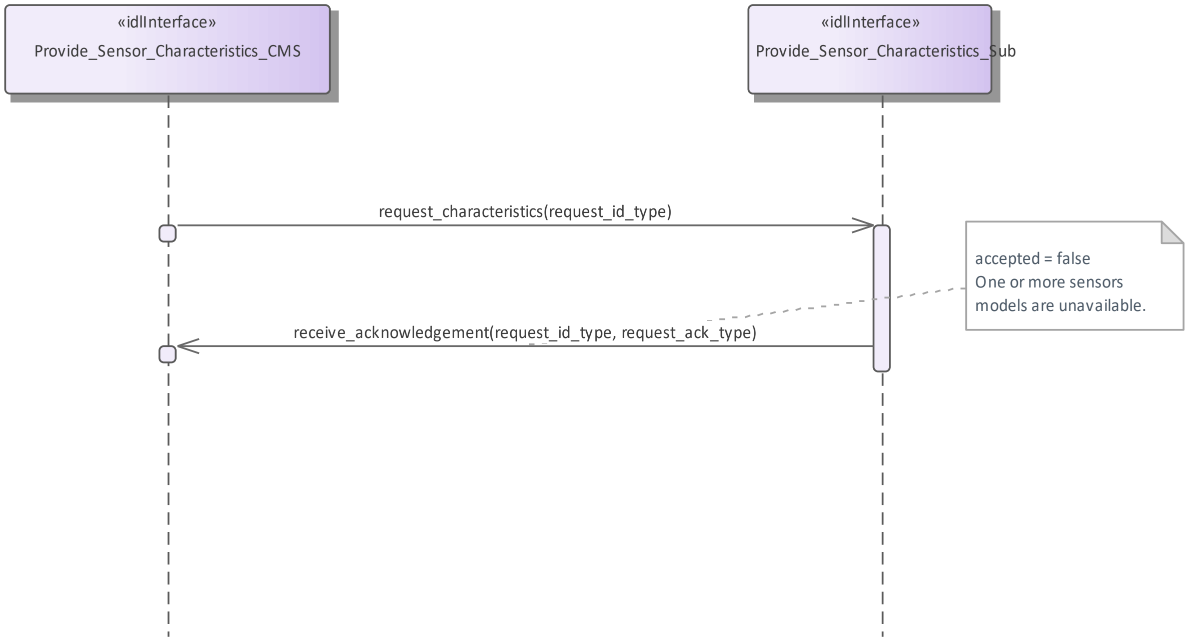


Figure 7.178 Provide_Sensor_Characteristics - Alternate Flow - models unavailable (Interaction diagram)

7.9.6.3 Provide_Plots

Parent Package: Plot_Reporting

Provide_Plots_CMS
Type: Interface
Package: Provide_Plots
Interface to the CMS for receiving plot updates.
This interface provides sensor plots to the CMS (filterable to air, surface, land and space environments). The transfer of data is expected to take place asynchronously, although for certain classes of sensor it may appear periodic

Pre-condition: Subsystem Services: Provide Subsystem Services has successfully executed
Pre-condition: Register Interest: The CMS has successfully registered interest in this service
Post-condition: Success: CMS has received plot datastream

Table 7.338 - Methods of Interface Provide_Plots_CMS

Method	Notes	Parameters
write_sensor_plot()	This method receives a individual plot update from the sensor. It is expected to be called periodically from the sensor.	sensor_plot_type **plots** The set of plots
write_sensor_plot_set()	This method receives a set of one or more plot updates from the sensor. It is expected to be called periodically from the sensor.	sensor_plot_set_type **plots** The set of plots
write_sensor_plot_summary()	This method receives a summary of plots found by the sensor in a region of the environment. It is expected to be called periodically from the	plot_summary_type plot_summary The summary of the plots

	sensor.	

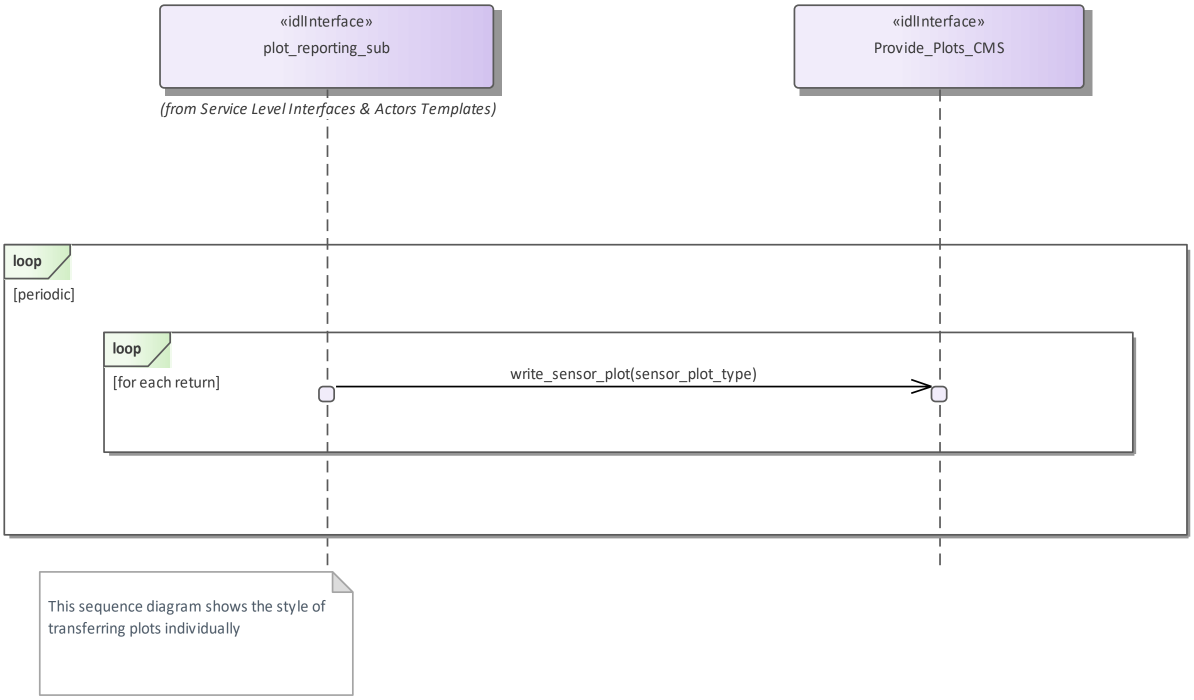


Figure 7.179 Basic Flow - Provide Plots (Individual) (Interaction diagram)

Figure 7.180 Basic Flow - Provide Plots (Sets) (Interaction diagram)

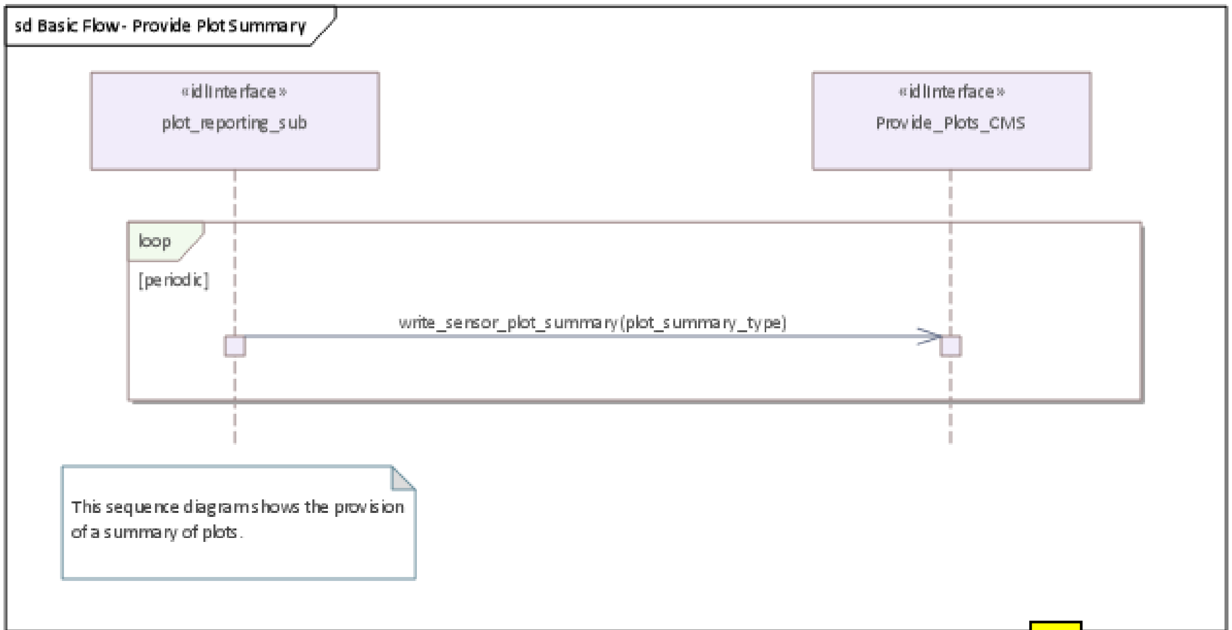


Figure 7.181 Basic Flow - Provide Plot Summery (Interaction diagram)

7.9.6.4 Provide_Sensor_Orientation

Parent Package: Plot_Reporting

Provide_Sensor_Orientation_CMS
Type: Interface
Package: Provide_Sensor_Orientation
The interface to the CMS for receiving sensor orientation updates.
The sensor provides its orientation in the case that it has movement that is independent of that for the overall platform. It is provided periodically with a frequency defined using the manage subsystem parameters use case.

Pre-condition: Subsystem Services: Provide Subsystem Services has successfully executed

Pre-condition: Register Interest: The CMS has successfully registered interest in this service

Post-condition: Success: CMS has received sensor orientation datastream

Table 7.339 - Methods of Interface Provide_Sensor_Orientation_CMS

Method	Notes	Parameters
write_sensor_orientation()	Informs the CMS of the orientation of the sensor	sensor_orientation_type **orientation** The orientation of the sensor

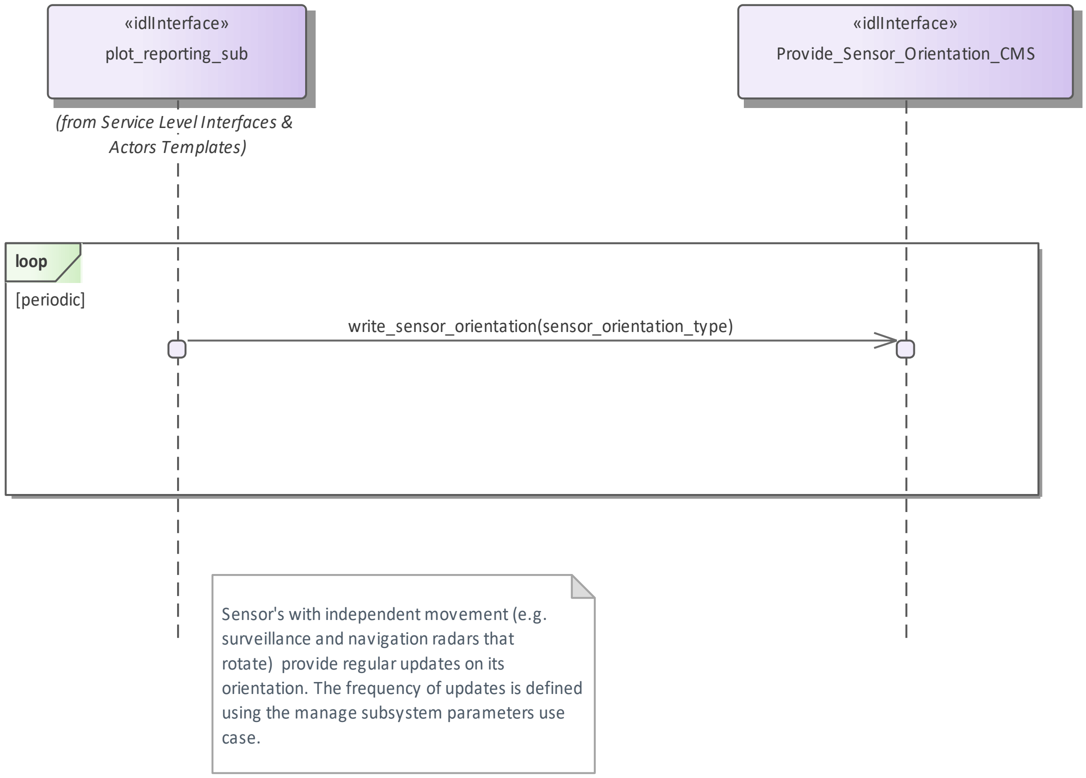


Figure 7.182 Basic Flow - Provide Sensor Orientation (Interaction diagram)

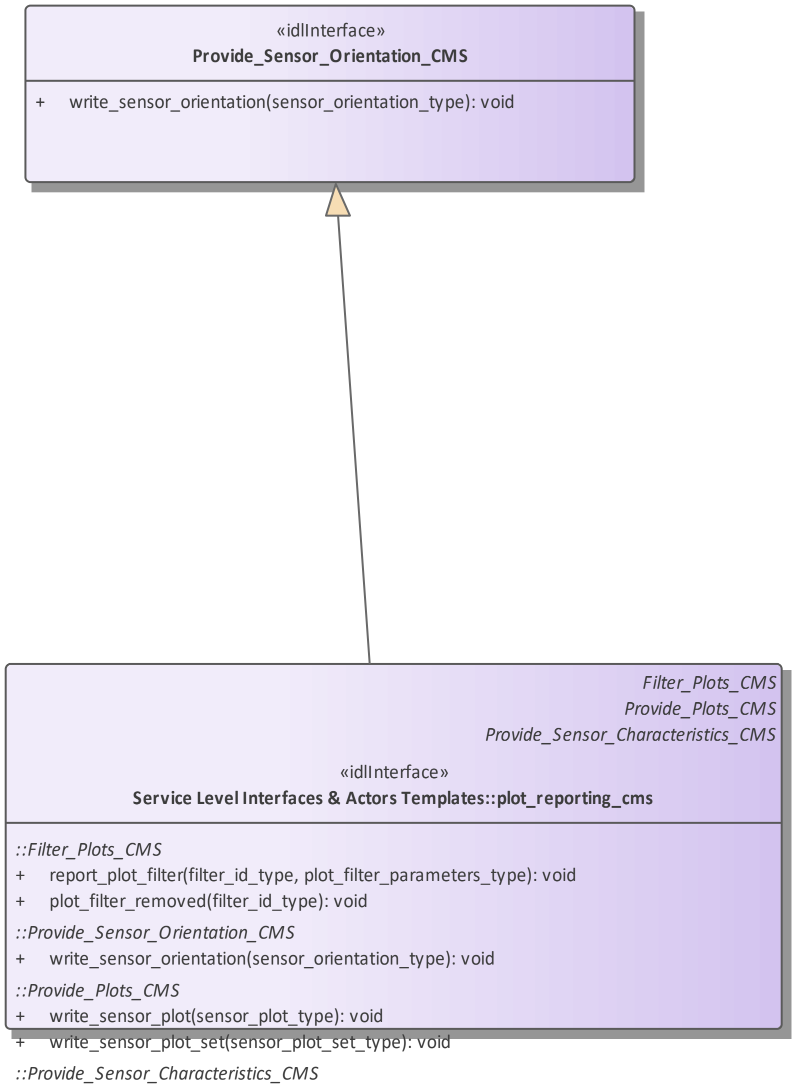


Figure 7.183 Provide_Sensor_Orientation (Class diagram)

7.9.7 Sensor_Control

Parent Package: Sensor_Services
This package contains interfaces for the Sensor Control service.

7.9.7.1 Manage_Frequency_Usage

Parent Package: Sensor_Control
This package contains interfaces for the Manage Frequency Usage service.
Manage_Frequency_Usage_CMS
Type: Interface
Package: Manage_Frequency_Usage
This controls the sensor behaviour with respect to the transmission frequency management. Basing on a discrete set of transmission frequencies offered by the sensor, CMS may disable/enable the use of a subset of them. As well CMS may select the sensor transmission mode, i.e. how the sensor shall select the transmission frequencies, among the set of transmission modes supported by the sensor.

The transmission mode defines how the sensor selects the transmission frequencies, which may be:

- Fixed Frequency: sensor always uses the same pre-selected frequency
- Frequency Diversity: at each transmission sensor selects the frequency to be used inside a pre-selected subset of frequencies
- Automatic Frequency Selection: at each transmission sensor selects the frequency to be used among the least jammed frequencies
- Random Agility: at each transmission sensor random selects the frequency to be used.

The availability of each of the above listed transmission modes depends on the sensor type and its capabilities (not all the sensor types support all them). Besides a transmission mode supported by the sensor may be “selectable” or “not selectable” according to the specific sensor rules and the state of transmission frequencies.

Both the set of transmission frequencies offered by the sensor and the supported transmission modes (names and characteristics) differ from sensor to sensor, so they shall be handled as configuration parameters. The sensor reports all supported frequencies whether or not currently available or enabled.
Sensors cannot enable/disable the setting of the frequency usage at its own initiative, but at any time a transmission frequency could become not available because of a fault (e.g. fault of the relevant oscillator), and this could affect the effective availability of one or more sensor supported transmission modes.

Provision of the frequency usage state
Sensor shall keep CMS informed about the current availability of the frequency usage and its changes (if any).

Provision of the transmission mode
Sensor shall keep CMS informed about the currently selected transmission mode, with the relevant parameters, and its changes (if any).

It is the CMS's responsibility to initiate the determination of initial state by making a request for information to the subsystem.

Lack of mastership
In the case where CMS does not have mastership of the sensor, CMS shall be informed about both the actual setting of the frequency usage and the actual transmission mode, with its changes (if any).

State of transmission frequencies
With respect to its operational use each sensor transmission frequency may be “enabled” or “disabled”, according to the relevant setting. On the other hand, with respect to its health status, each transmission frequency may be “available” or “not available” according to the presence of faults.
Note that a transmission frequency may be effectively selectable for the sensor transmission if it is both “enabled” and not in fault.

Relationship to *Manage Transmission Sectors*
As well as the overall transmission mode, here specified, CMS may define sectors where a devoted transmission mode is to be applied (see *Manage Transmission Sectors*).

Pre-condition: Mastership Required: CMS has mastership of the sensor.

Pre-condition: Subsystem Services: *Provide subsystem services* is successfully passed.

Pre-condition: Transmission Frequencies: CMS knows the transmission frequencies offered by the sensor and their actual availability.

Pre-condition: Selectable Transmission modes and frequencies: CMS is aware of the currently selectable transmission modes and transmission frequencies.

Post-condition: Success: Both the setting of the frequency usage and the sensor transmission mode are according to the request and CMS is informed that this is the case.

Post-condition: No Success: Both the setting of the frequency usage and the sensor transmission mode are unchanged with respect to the original one and CMS is informed that this is the case.

Table 7.340 - Methods of Interface Manage_Frequency_Usage_CMS

Method	Notes	Parameters
report_frequencies_state()	Method used by the sensor to return the current availability of the frequency usage and its changes (if any).	all_frequencies_state_type **frequencies_state**
report_transmission_mode_state()	Method used by the sensor to return the selected transmission mode, with the relevant parameters, and its changes (if any).	request_id_type **request_id** transmission_frequency_mode_type **transmissionModeSetting**
transmission_frequency_state_response()	Method used by the sensor to return the actual setting of the frequency usage modified according to the request.	request_id_type **request_id** selected_frequency_list_type **setting_message**

Manage_Frequency_Usage_Sub
Type: Interface
Package: Manage_Frequency_Usage
This is the Subsystem interface for managing frequency usage.

Table 7.341 - Methods of Interface Manage_Frequency_Usage_Sub

Method	Notes	Parameters
set_frequencies()	Method used by the CMS to enable or disable frequency bands or discrete frequencies.	request_id_type **request_id** selected_frequency_list_type **request**
set_transmission_mode()	Method used by the CMS to select the available sensor transmission mode.	request_id_type **request_id** transmission_frequency_mode_type **trasmissionmode**

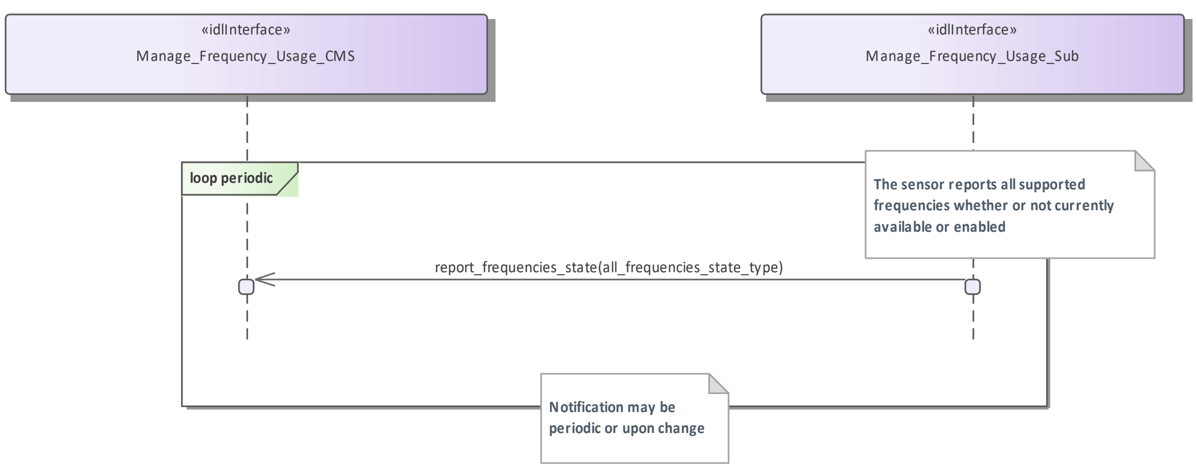


Figure 7.184 Basic Flow - Frequency Availability Change Notification (Interaction diagram)

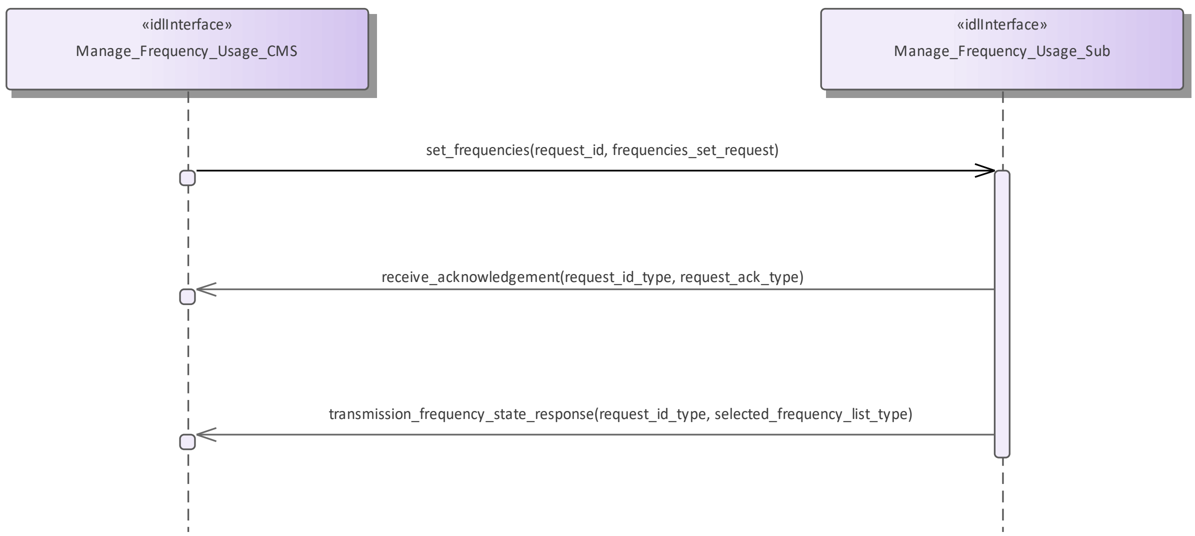


Figure 7.185 Basic Flow - Enable/Disable Frequency Usage (Interaction diagram)

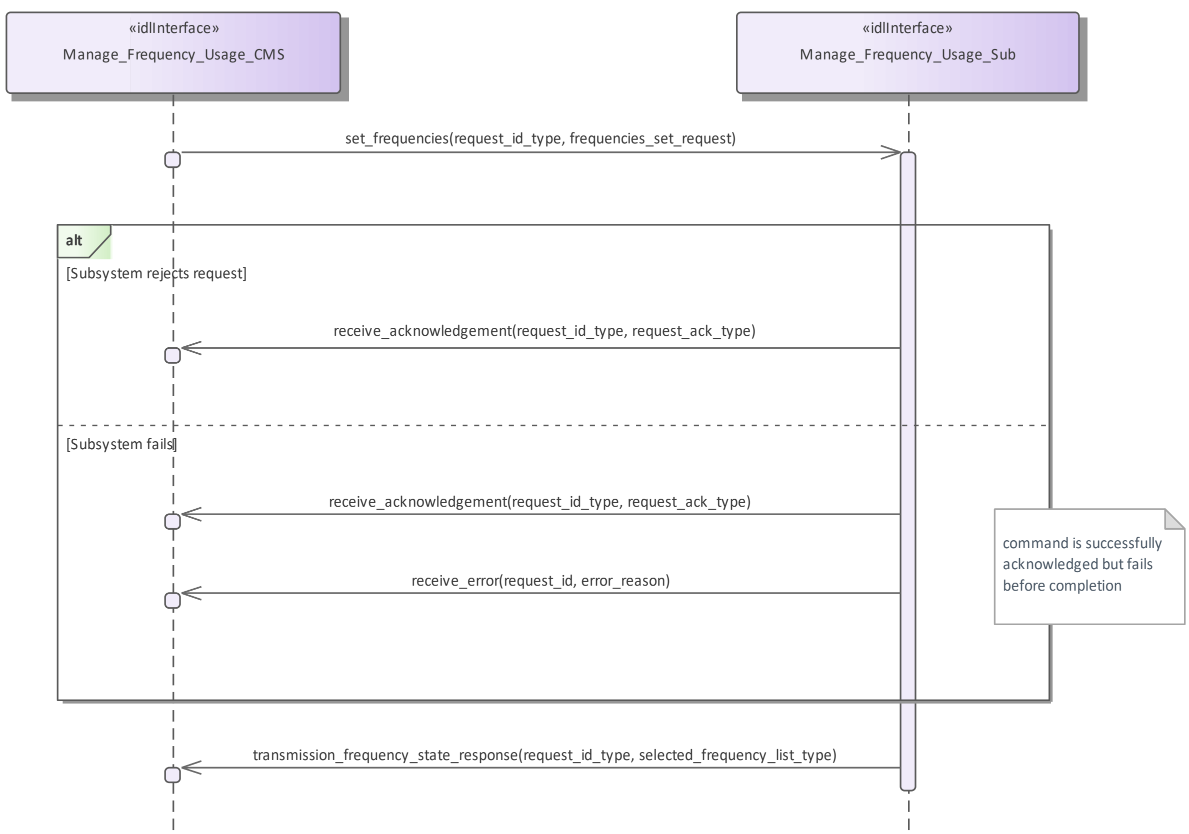


Figure 7.186 Alternative Flow - Enable/Disable Frequency Usage - loss of mastership (Interaction diagram)

Figure 7.187 Basic Flow - Transmission Mode Selection (Interaction diagram)

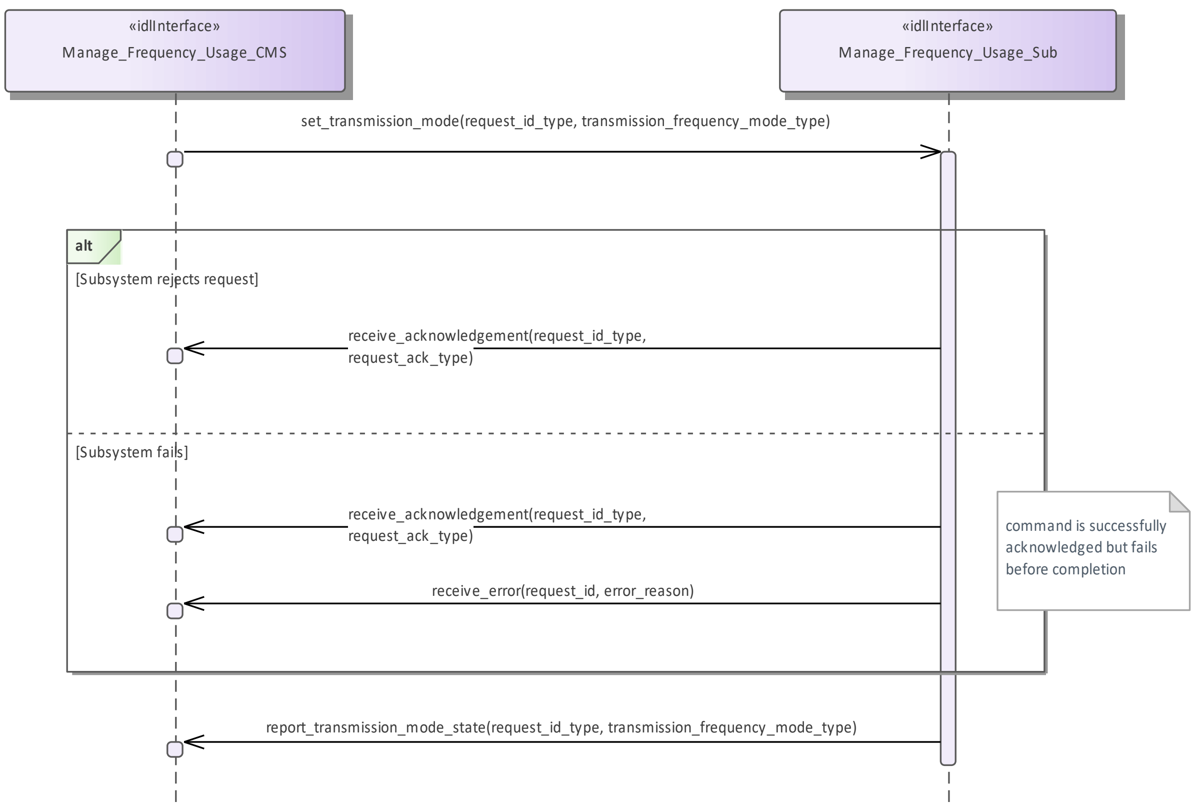


Figure 7.188 Alternative Flow - Transmission Mode Selection - loss of mastership (Interaction diagram)

7.9.7.2 Manage_Transmission_Sectors

Parent Package: Sensor_Control
This package contains interfaces for the Manage Transmission Sectors service.
Manage_Transmission_Sectors_CMS
Type: Interface
Package: Manage_Transmission_Sectors

This determines the sectors where the sensor is allowed to radiate together with the relevant transmission modes and parameters. Sectors may be delimited in azimuth only, or both in azimuth and elevation; for each sector the sensor may be requested either to no transmit at all or to apply a proper transmission mode. Typical transmission sectors types are:

- Transmit Inhibit Sectors

sectors where the sensor is not allowed to radiate. Depending on the sensor type and its capabilities, such a type of sectors may be delimited in azimuth only, or both in azimuth and elevation.

- Reduced Radiate Power Sectors

sectors where the sensor shall radiate at reduced power. Depending on the sensor type and its capabilities, such a type of sectors may be delimited either in azimuth only or both in azimuth and elevation.

- Transmission Mode Sectors

sectors where the sensor is required to apply a devoted transmission mode (see *Manage Frequency Usage*). Depending on the sensor type and its capabilities, such a type of sectors may be delimited either in azimuth only or both in azimuth and elevation, but they may not overlap each other.

- Blind Arc Sectors

sectors where the sensor is not allowed to radiate. Such a type of sectors may be delimited in azimuth only, or both in azimuth and elevation, depending on the sensor type and its capabilities. (Note: the same as "Transmit Inhibit Sectors", with the difference that sectors are defined in Ship's Reference System.)

Provision of the sensor transmission sectors setting
Sensor shall keep CMS informed about the actual setting of the transmission sectors and its changes (if any).

It is the CMS's responsibility to initiate the determination of initial state by making a request for information to the subsystem.

Lack of mastership
In the case where CMS does not have mastership of the sensor, CMS shall be informed about the actual setting of the transmission sectors and its changes (if any).
Pre-condition: Mastership Required: CMS has mastership of the sensor
Pre-condition: Subsystem Services: *Provide subsystem services* is successfully passed
Pre-condition: Transmission Sectors: CMS is aware of which types of transmission sectors the sensor may manage and of their current setting.
Post-condition: Success: The setting of the transmission sectors has been modified according to the request and CMS is informed that this is the case.
Post-condition: No Success: The setting of the transmission sectors is unchanged with respect to the original one and CMS is informed that this is the case.

Table 7.342 - Methods of Interface Manage_Transmission_Sectors_CMS

Method	Notes	Parameters
transmission_sector_setting()	Method used by the sensor to return the actual setting of the transmission sectors and its changes (if any).	request_id_type **request_id** transmission_sector_set_type **setting_message**

Manage_Transmission_Sectors_Sub
Type: Interface
Package: Manage_Transmission_Sectors
This is the Subsystem interface for managing transmission sectors.

Table 7.343 - Methods of Interface Manage_Transmission_Sectors_Sub

Method	Notes	Parameters
set_transmission_sector()	Method used by the CMS to send a set/reset transmission sector request to the sensor.	request_id_type **request_id** transmission_sector_set_type **sector**

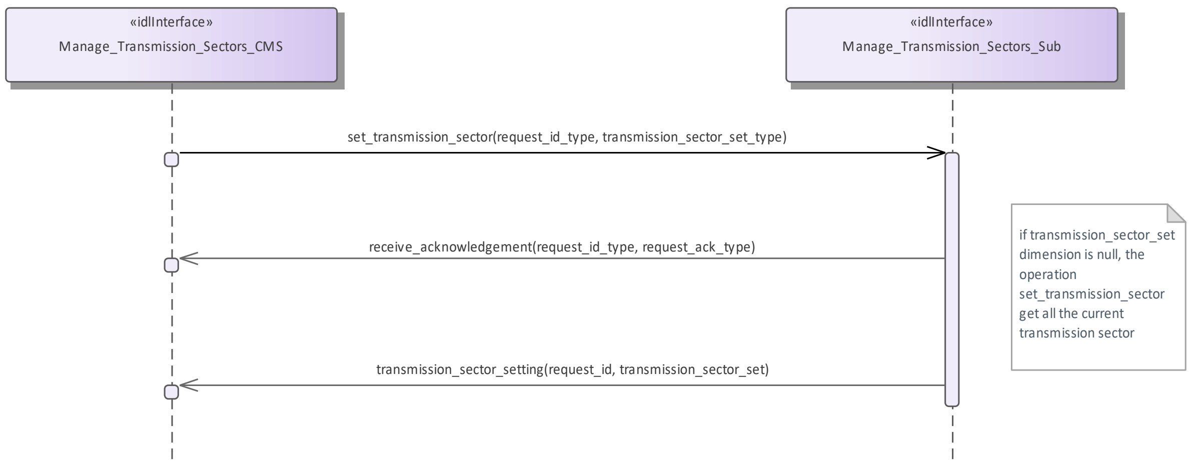


Figure 7.189 Basic Flow - Manage Transmission Sectors - Enable/Disable (Interaction diagram)

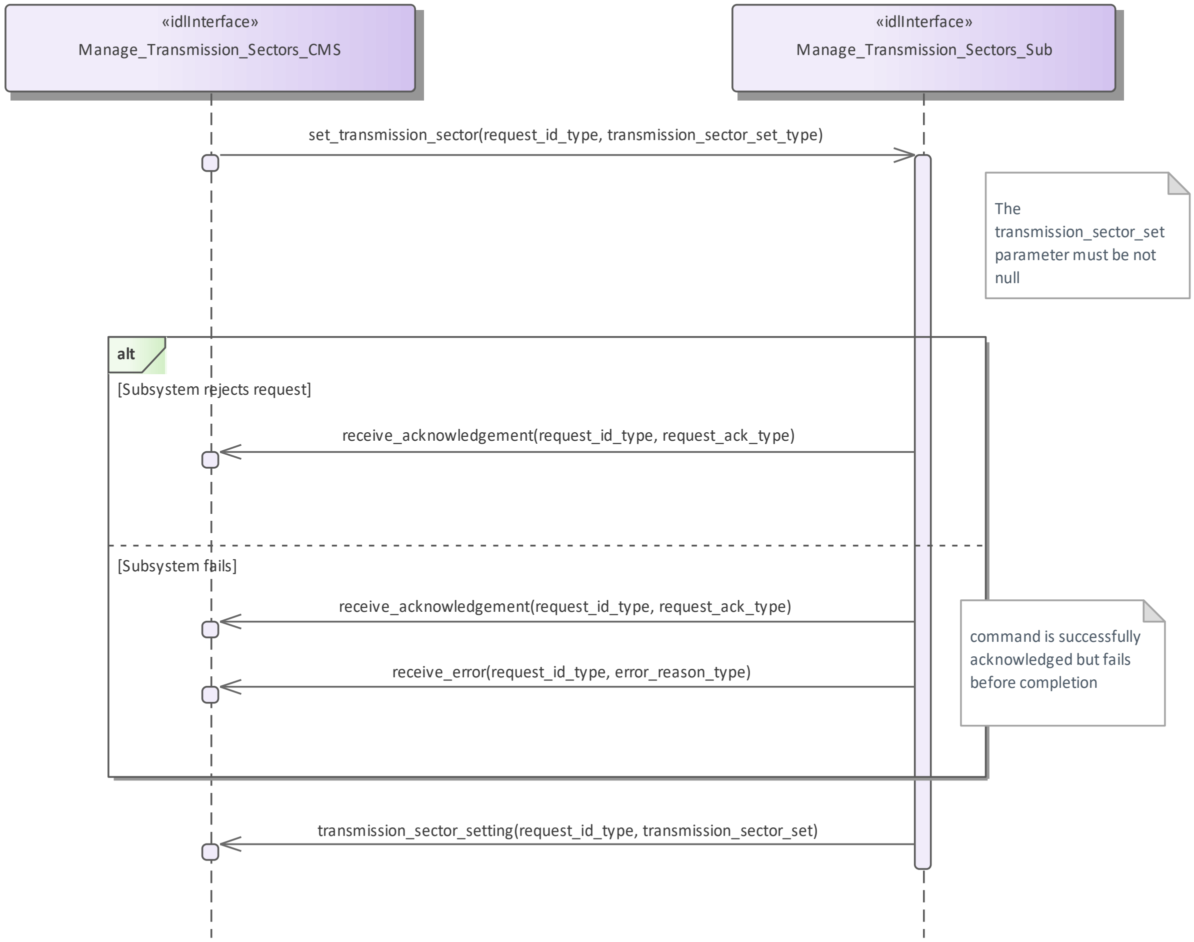


Figure 7.190 Alternative Flow - Manage Transmission Sectors - Enable/Disable - loss of mastership (Interaction diagram)

7.9.7.3 Control_Emissions

Parent Package: Sensor_Control
This package contains interfaces for the Control Emissions service.
Control_Emissions_CMS
Type: Interface
Package: Control_Emissions
The sensor is requested to inhibit/enable own emissions. In the case where the sensor is a radar, this shall result in the Radiation on/off command.
Note that this interface just covers the software managed control of the emission state. For safety reasons many sensors are supplied with an additional hardware control of own emission state, such as a pushbutton directly connected to the transmitter.

Provision of the Emission state
Sensor shall keep CMS informed about the current state of emissions and its changes (if any).

It is the CMS's responsibility to initiate the determination of initial state by making a request for information to the subsystem.

Lack of mastership

In the case where CMS does not have mastership of the sensor, CMS shall be informed about the current emissions state and its changes (if any).

Relationship to the Transmission Sectors management
As long as emissions are on, the sensor shall transmit in the sectors where transmission is allowed and according to the relevant transmission modes and parameters, as determined through *Manage Transmission Sectors*.

Pre-condition: Mastership Required: CMS has mastership of the sensor

Pre-condition: Subsystem Services: *Provide subsystem services* is successfully passed

Pre-condition: Emissions State: CMS is aware that actually the sensor may switch its emissions state, e.g. both the technical state and the health state allow the sensor to switch to Radiation on, no engagement in execution to switch to Radiation off, and so on.

Post-condition: Success: The sensor emissions state is on/off as requested and CMS is informed that this is the case.

Post-condition: No Success: The sensor emissions state is still equal to the original one and CMS has the correct information regarding that state

Table 7.344 - Methods of Interface Control_Emissions_CMS

Method	Notes	Parameters
control_emission_setting()	Method used by the sensor to return the current state of emissions and its changes (if any).	request_id_type **request_id** control_emission_state_type **emission_state**

Control_Emissions_Sub
Type: Interface
Package: Control_Emissions
This is the Subsystem interface for controlling emissions.

Table 7.345 - Methods of Interface Control_Emissions_Sub

Method	Notes	Parameters
set_control_emission()	Method used by the CMS to send an Emissions on/off request to the sensor.	request_id_type **request_id** control_emission_state_type **control_emission_state**

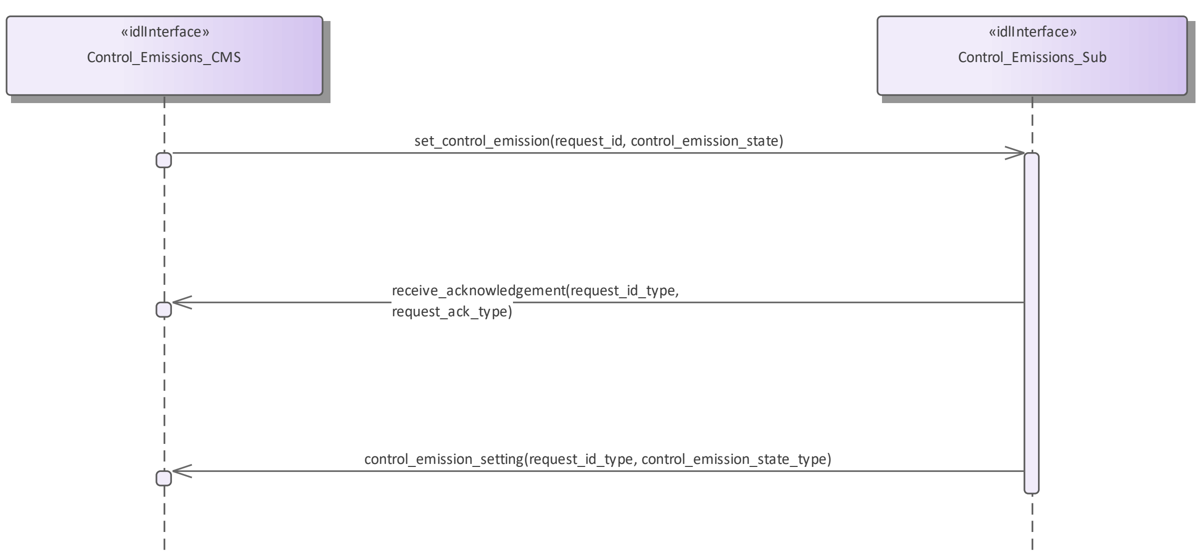


Figure 7.191 Basic Flow - Control Emissions - On/Off (Interaction diagram)

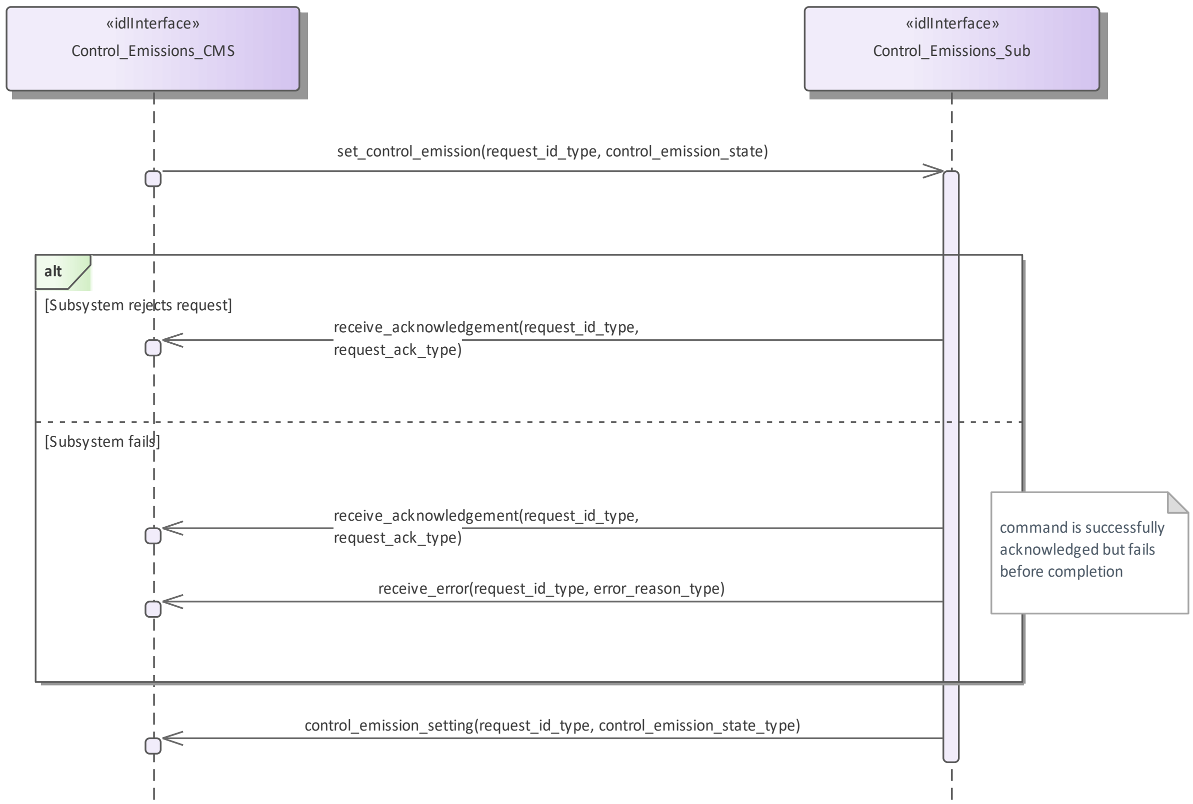


Figure 7.192 Alternative Flow - Control Emissions - On/Off - loss of masterhip (Interaction diagram)

7.9.7.4 Define_Test_Target_Scenario

Parent Package: Sensor_Control
This package contains interfaces for the Define Test Target Scenario service.
Define_Test_Target_Scenario_CMS
Type: Interface
Package: Define_Test_Target_Scenario
This specifies the interactions for defining and modifying a test target scenario. A Test Target scenario consists of a number of Test Targets to be generated according to their characteristics (positions, motion law, generation parameters) with the purpose of producing stimuli devoted to the execution of an internal functional test of the sensor.
A number of Test Target scenarios may be maintained in a sensor internal Test Targets scenarios database, where each scenario is identified by a unique identification number. Write accesses to this database shall rejected if the sensor Mastership is not actually assigned to CMS, but the possession of the sensor Mastership is not required for executing read accesses.
The generation of the so defined Test Target scenarios may be activated as specified in *Control Test Target Facility*. For the generation mechanism see the interface *Control Test Target Facility*

One or more Test Target scenarios may be maintained in a sensor internal Test Targets scenarios database, where each scenario is identified by an unique identification number. The number of available Test Target scenarios is accessed by *Manage subsystem parameters*.

Depending on the sensor type and its capabilities, a Test Target scenario may be constituted by:

a) a number of independent targets, with each target having own characteristic parameters; so the scenario is defined by:

- number of targets

and for each target

- the initial target position with the relevant initial time
- target parameters

b) a number of targets distributed in a defined area/volume and having the same common parameters, so the scenario is defined by:

- number of targets
- area/volume boundaries
- common initial time
- common targets parameters

Target parameters define:
a. the target motion type, with the relevant motion parameters
b. the target generation parameters, such as injection type (internal / external), attenuation law (constant / variable-with-range), doppler type (0 / PRF/2).

Pre-condition: Mastership Required: CMS has mastership of the sensor

Pre-condition: Subsystem Services: *Provide subsystem services* is successfully passed

Pre-condition: Test Target Facility: Test Target facility is supported by the sensor and CMS is aware of which types of Test Target the sensor may manage

Post-condition: Success: Write access:
The specified Test Target scenario is modified according to the request and CMS is informed that this is the case.

Read access:
The requested Test Target scenario is reported to CMS.

Post-condition: No Success: Write access:
The specified Test Target scenario is unchanged and CMS is informed about the denial reason.

Read access:
The requested Test Target scenario is not reported to CMS and CMS is informed about the denial reason.

Table 7.346 - Methods of Interface Define_Test_Target_Scenario_CMS

Method	Notes	Parameters
test_target_scenario_setting()	Method used by the sensor to return the identification number of the modified or created test target scenario.	request_id_type **request_id** test_target_scenario_id_type **test_target_scenario_id**
test_target_scenario_setting_all_feature()	Method used by the sensor to return the required test target scenario with its parameters.	request_id_type **request_id** test_target_scenario_type **test_target_features**

Define_Test_Target_Scenario_Sub
Type: Interface
Package: Define_Test_Target_Scenario
This is the Subsystem interface for defining test target scenarios.

Table 7.347 - Methods of Interface Define_Test_Target_Scenario_Sub

Method	Notes	Parameters
read_test_target_scenario()	Method used by the CMS to send to the sensor a read request of a specified Test Target scenario.	request_id_type **request_id** test_target_scenario_id_type **test_target_scenario_id**
write_test_target_scenario()	Method used by the CMS to send to the sensor a write request of a specified Test Target scenario.	request_id_type **request_id** test_target_scenario_type **test_target_scenario**

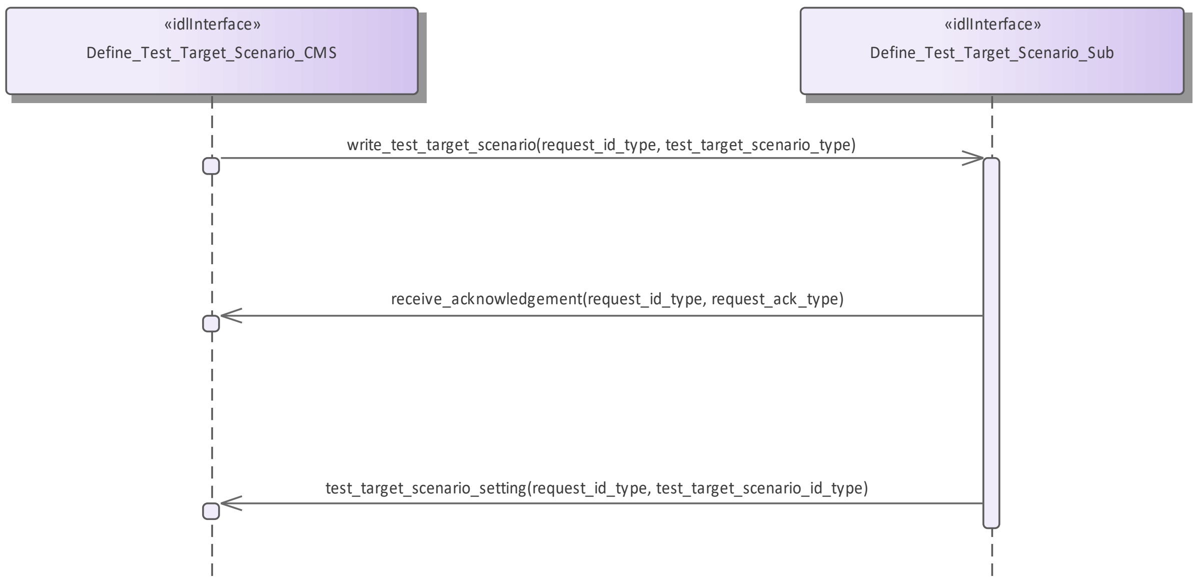


Figure 7.193 Basic Flow - Write a Target Test Target Scenario (Interaction diagram)

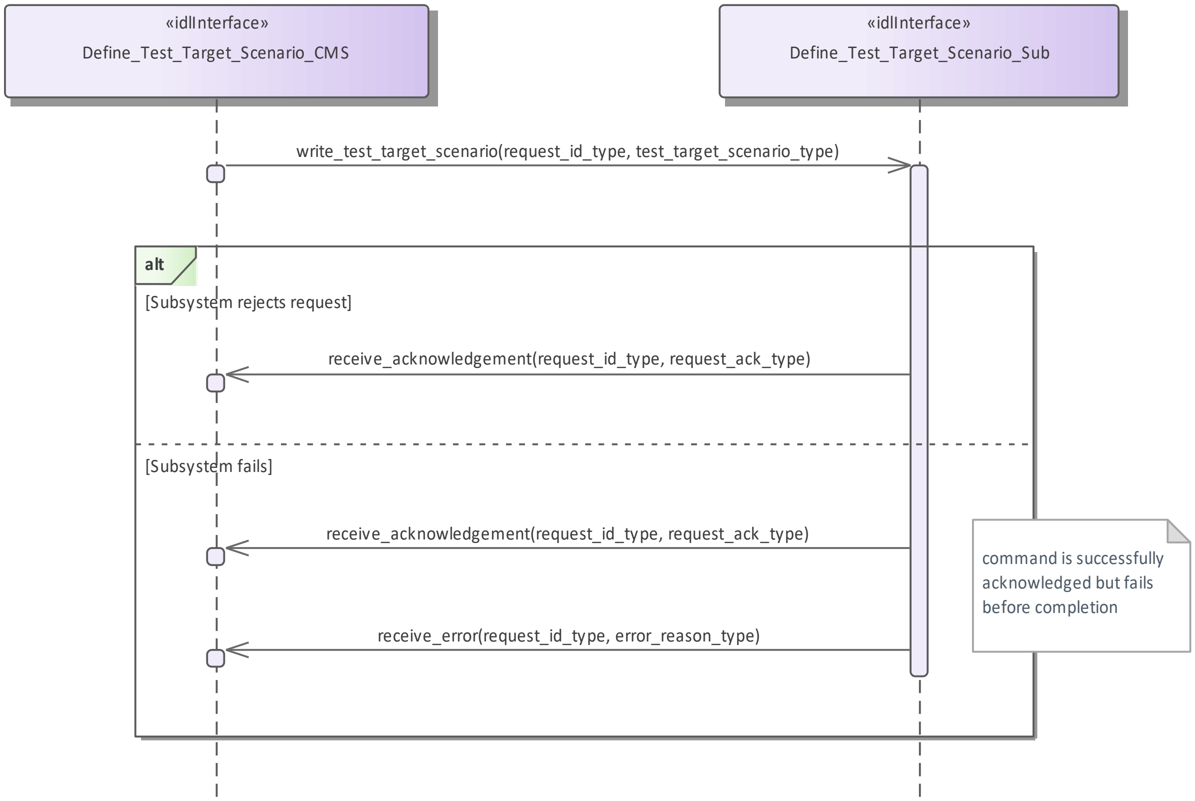


Figure 7.194 Alternative Flow - Write a Target Test Target Scenario - loss of mastership (Interaction diagram)

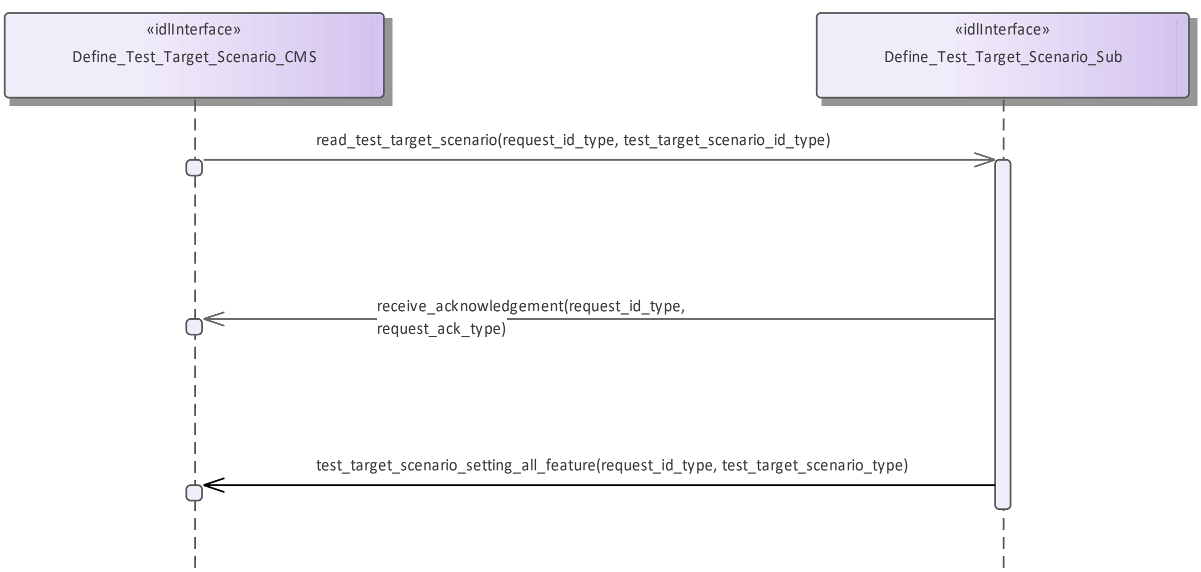


Figure 7.195 Basic Flow - Inspect a Test Target Scenario (Interaction diagram)

7.9.7.5 Test_Target_Facility

Parent Package: Sensor_Control

This package contains interfaces for the Test Target Facility service.
Test_Target_Facility_CMS
Type: Interface
Package: Test_Target_Facility
The sensor is requested to activate/deactivate the execution of its internal functional test and stimulation realized by means of test targets generation. A number of Test Target scenarios may be defined and modified as specified in *Define Test Target Scenario*, each scenario is identified by a proper identification. At any time no more than one Test Target scenario may be active.

Test Target generation mechanism (applicable to some sensors)
The Test Target generation consists of the injection of proper signals at different points of the receiver chain in order to produce the relevant detections in input to the RMC (Radar Management Computer); these Test Target detections are processed by the RMC as the real ones, so they shall generate one o more plots ("Test Target" plots) and tracks ("Test Target" tracks).
Such a generation mechanism is controlled by the RMC driving a devoted hardware, its purpose is to execute an on-line BITE of the complete receiver chain.
Test Target generation is executed while the radar is working in operational mode, so Test Target detections and real detections live together, forming "Test Target" plots and tracks at the same time as real plots and tracks. This implies that CMS shall receive "Test Target" plots and tracks together with real plots and tracks.

Lack of mastership
In the case where CMS does not have mastership of the sensor, CMS shall be informed about the actual state of the Test Target generation and its changes (if any).

Provision of the Test Target generation state
Sensor shall keep CMS informed about the actual state of the Test Target generation and its changes (if any).

Relationship to the subsystem health state
As long as a Test Target scenario is in generation sensor checks the relevant returns at different points of the receiver chain, up to form plots in the same positions where Test Targets have been generated. The relevant results contribute to the sensor health state.
Pre-condition: Mastership Required: CMS has mastership of the sensor
Pre-condition: Subsystem Services: *Provide subsystem services* is successfully passed
Pre-condition: Test Target facility: Test Target facility is supported by the sensor and CMS is aware of the current availability of the Test Target generation.
Post-condition: Success: The state of the Test Target generation is modified according to the request and CMS is informed that this is the case.
Post-condition: No Success: The state of the Test Target generation is unchanged with respect the original one and CMS is informed about the denial reason.

Table 7.348 - Methods of Interface Test_Target_Facility_CMS

Method	Notes	Parameters
notify_test_target()	Method used by the sensor to return the actual state of the Test Target generation consistent with the request.	request_id_type **request_id** test_target_scenario_state_type **test_target_scenario_state**

Test_Target_Facility_Sub
Type: Interface
Package: Test_Target_Facility
This is the Subsystem interface for testing target facilities.

Table 7.349 - Methods of Interface Test_Target_Facility_Sub

Method	Notes	Parameters

set_test_target_facility_state()	Method used by the CMS to send an activation request of a specified Test Target scenario.	request_id_type **request_id** test_target_scenario_state_type **scenario_state**

Figure 7.196 Basic Flow - Activate/Deactivate Test Target Facility (Interaction diagram)

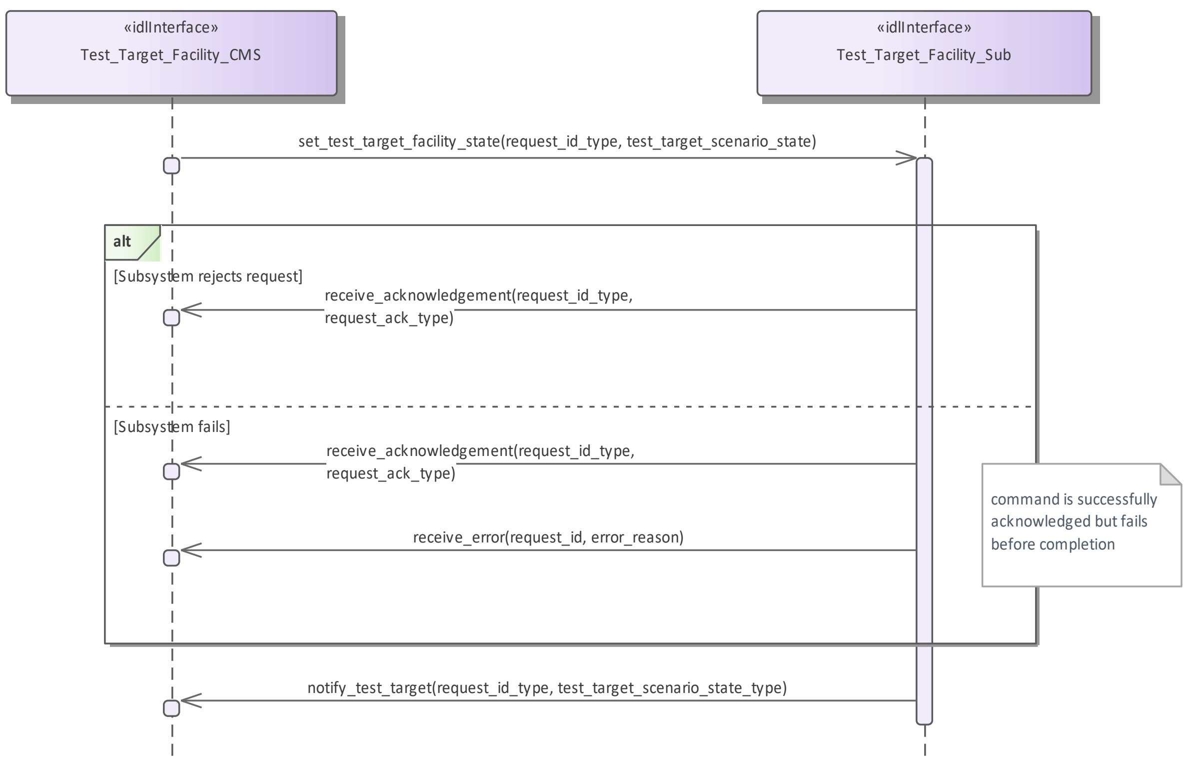


Figure 7.197 Alternative Flow - Activate/Deactivate Test Target Facility - loss of mastership (Interaction diagram)

7.9.8 Sensor_Performance

Parent Package: Sensor_Services

This package contains interfaces for the Sensor Performance service.

7.9.8.1 Provide_Interference_Reports

Parent Package: Sensor_Performance

Provide_Interference_Reports_CMS

Type: Interface

Package: Provide_Interference_Reports

This describes the process whereby the subsystem provides a set of reports on sources of interference, including jammers. The data shall, therefore, in general, be non-real-time but should, where appropriate, be time-tagged and shall be updated when any observed data changes.

The sensor need not be radiating but shall at least be receiving. The subsystem VOI (volume of interest) or other filter mechanisms might be supplied in a request to the subsystem

For a nominal effect assessment, the request might contain data on number, strength/Effective Radiated Power (ERP), type and deployment of jammers and other interferers affecting radar operations. For example, for each interferer

- Sensor time-tag
- Interference type - active noise, self-screening jammer, standoff jammer etc
- Strength/Effective Radiated Power
- Locations - strobes etc.
- Affected sectors
- Frequency bands affected

Pre-condition: Technical State: The subsystem is in technical state ONLINE.

Pre-condition: Subsystem Services: The Provide Subsystem Services Service has been completed successfully

Pre-condition: Register Interest: The Register Interest Service has been executed successfully to register interest in Interference Reports.

Post-condition: Success: The CMS has received Interference Reports

Post-condition: Failure: The CMS receives no Interference Reports

Table 7.350 - Methods of Interface Provide_Interference_Reports_CMS

Method	Notes	Parameters
interference_report_response()	Provides an updated set of interference reports to the CMS.	request_id_type **request_id** interference_report_type **interference_report** The report on interference
interference_report_periodic()	Provides an updated set of interference reports to the CMS.	interference_report_type **interference_report** The report on interference

Provide_Interference_Reports_Sub

Type: Interface

Package: Provide_Interference_Reports

Table 7.351 - Methods of Interface Provide_Interference_Reports_Sub

Method	Notes	Parameters
volume_for_interference_reports()	This allows definition of the volume in space which is of interest with regard to the provision of	request_id_type **request_id** The unique identifier for this request. This is referenced in acknowledgement

	interference reports.	and any error reporting regarding this definition of the volume of interest. polar_volume_type **volume** The volume in space coordinate_orientation_type **coordinate_orientation** specifies the orientation of the polar volume

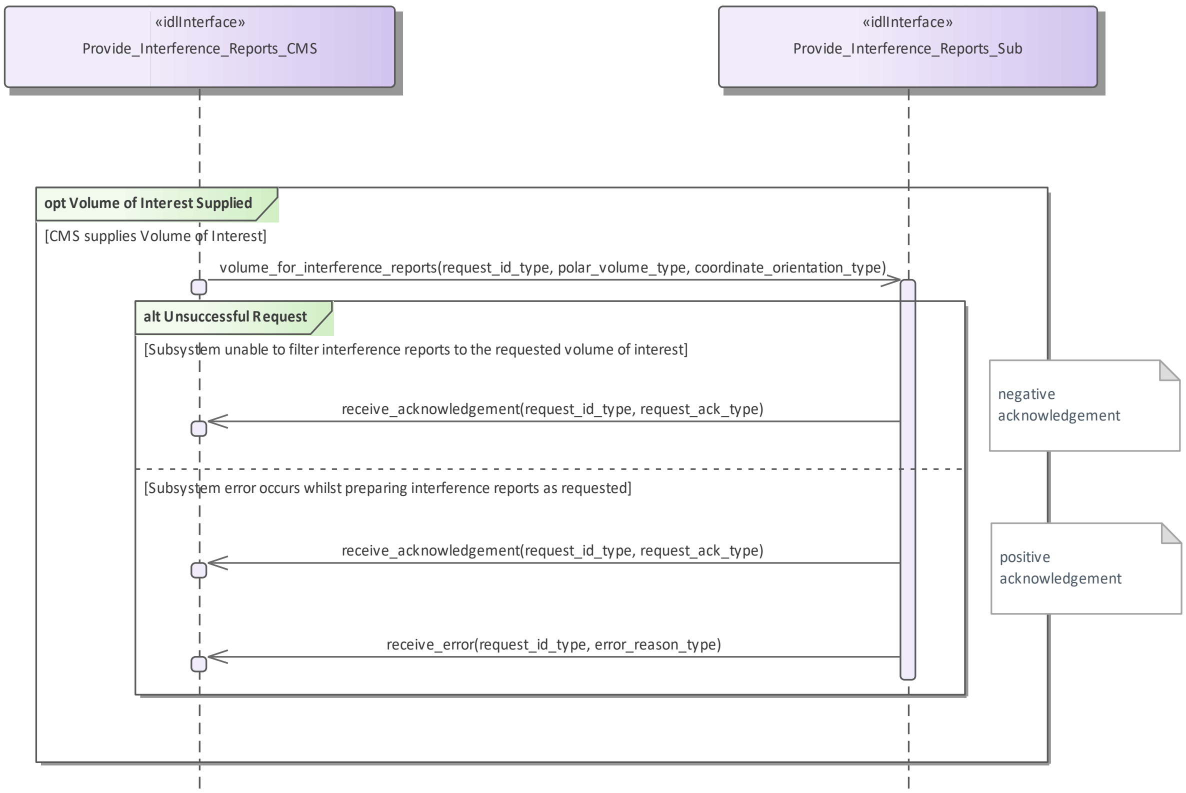


Figure 7.198 Alternative Flow - Provide Interference Reports (Interaction diagram)

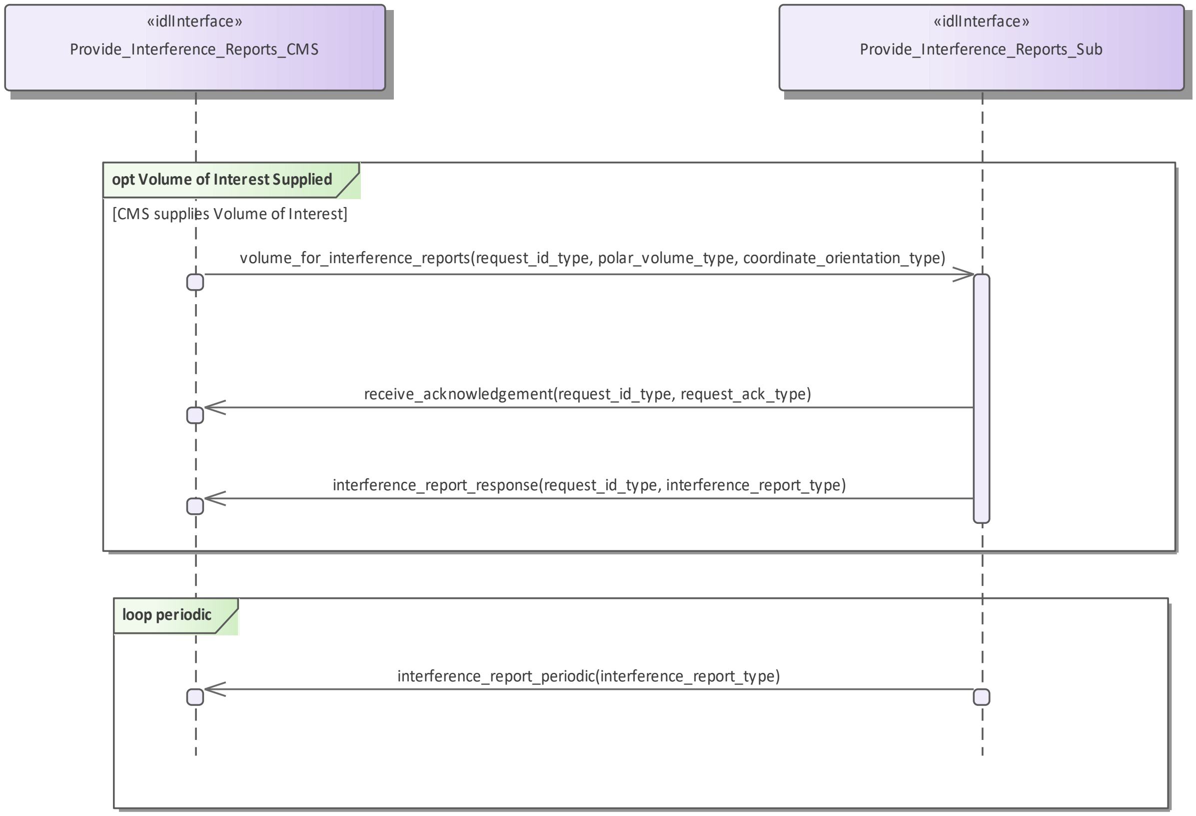


Figure 7.199 Basic Flow - Provide Interference Reports (Interaction diagram)

7.9.8.2 Provide_Nominal_Performance

Parent Package: Sensor_Performance

Provide_Nominal_Performance_CMS
Type: Interface
Package: Provide_Nominal_Performance
This is incremental to *Register Interest*, which deals with the subscription to subsystem functions. It provides an indication of the expected performance of the available subsystem services such as those presented in Provide Subsystem Services, based upon the current environmental conditions (See Receive Meteorological Data - METOC).
The subsystem need not be radiating to provide this assessment. This interface is more targeted towards a subsystem such as the complex MFR than the 2D surveillance radar. The most basic example of performance would be reporting of the nominal coverage, in elevation, azimuth and range, given an assumed operating regime with no jamming and with default clutter conditions. Other examples might be that the actor requests the probability of detection for a specified target type or perhaps the probability of correct automatic classification of such a target within a specified sector of coverage under current environmental conditions.

Pre-condition: Technical State: The Subsystem is in the Technical State ONLINE.

Pre-condition: Subsystem Services: The Provide Subsystem Services Service has been executed successfully.

Post-condition: Success: The CMS is aware of the Nominal Performance of the Subsystem

Post-condition: Failure: The CMS is not aware of the Nominal Performance of the Subsystem

Table 7.352 - Methods of Interface Provide_Nominal_Performance_CMS

Method	Notes	Parameters

nominal_performance_response()	The subsystem responds to the previous nominal performance request with its determination of the requested aspect of nominal performance.	request_id_type **request_id** The unique id from the request performance_assessment_report_type **report** The report on nominal performance

Provide_Nominal_Performance_Sub
Type: Interface
Package: Provide_Nominal_Performance
Subsystem interface for provision of nominal performance assessment.

Table 7.353 - Methods of Interface Provide_Nominal_Performance_Sub

Method	Notes	Parameters
nominal_performance_request()	The CMS requests nominal performance of the subsystem in the current environmental conditions. The aspect of performance requested is a parameter of the request.	request_id_type **request_id** The unique id which identifies this request. It is used to mark replies from the sensor relating to this request. performance_assessment_parameters_type **request** The details of the performance request

Figure 7.200 Alternative Flow - Provide Nominal Performance (Interaction diagram)

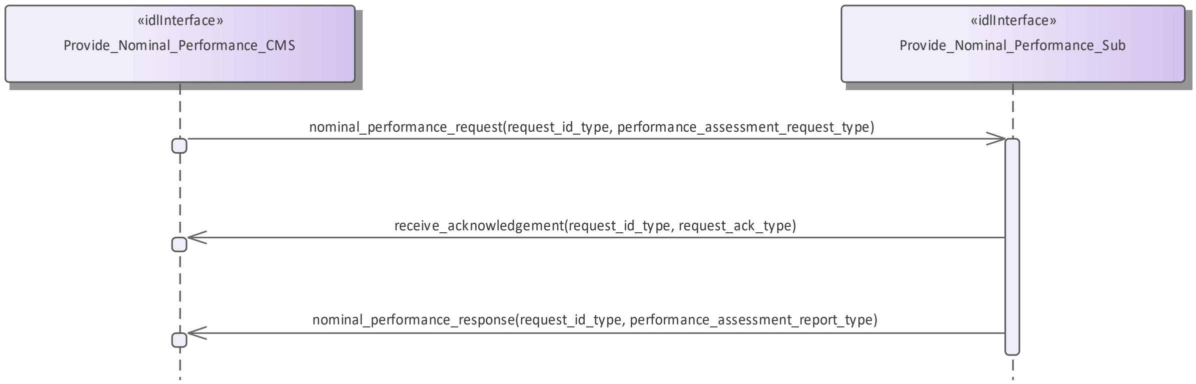


Figure 7.201 Basic Flow - Provide Nominal Performance (Interaction diagram)

7.9.8.3 Provide_Performance_Assessment

Parent Package: Sensor_Performance

Provide_Performance_Assessment_CMS
Type: Interface
Package: Provide_Performance_Assessment

This is incremental to *Register Interest*, which deals with the subscription to subsystem functions and *Provide Nominal Performance* which provides the subsystem nominal performance. This interface reports the real-time performance of the available subsystem functions against the goals of the mission. The reported performance is that currently being attained by the subsystem subject to the current operating regime and environmental conditions, including any clutter and jamming and taking account of any mitigation/cancellation of such effects by the subsystem.

This interface is aimed at a subsystem such as an MFR radar. Information is provided to the Command function allowing decisions to be made on the achieved performance, which is often considerably different to the anticipated performance level as reported through the Provide Nominal Performance Service.

The most basic example of performance would be reporting of the radar coverage, in elevation, azimuth and range, for the current operating regime and environmental conditions. This would take account of any clutter and jamming present. Other examples might be that the actor requests the probability of detection for a specified target type or perhaps the probability of correct automatic classification of such a target within a specified range under current environmental conditions N.B. if the radar is operating in an appropriate mode then real-time clutter and/or jamming data might be available to the radar subsystem. Otherwise the actor would have to supply any known data to the subsystem for performance assessment (see Receive Encyclopaedic Data and Receive Geographic Information). If no environmental data is specified then the design performance would be reported.

Pre-condition: Technical State: The Subsystem is in the technical state ONLINE.
Pre-condition: Subsystem Services: The Provide Subsystem Services Service has completed successfully.
Post-condition: Success: The CMS is aware of the assessed performance of the subsystem
Post-condition: Failure: The CMS is not aware of the assessed performance of the subsystem

Table 7.354 - Methods of Interface Provide_Performance_Assessment_CMS

Method	**Notes**	**Parameters**
performance_assessment_response()	The subsystem responds to the previous performance assessment request with its assessment of the requested aspect of actual performance.	request_id_type **request_id** The unique identifier for this assessment. This identifier is supplied by the CMS when the assessment is requested. performance_assessment_report_type **performance_assessment** The

		details of the assessment

Provide_Performance_Assessment_Sub
Type: Interface
Package: Provide_Performance_Assessment
Subsystem interface for provision of current performance assessment.
Note that the coordinates are always polar for this service and that the origin is always the sensor reference point as per the coordinates and positions package.

Table 7.355 - Methods of Interface Provide_Performance_Assessment_Sub

Method	Notes	Parameters
performance_assessment_request()	The CMS requests assessment of actual performance of the subsystem. The aspect of performance requested is a parameter of the request.	request_id_type **request_id** The unique identifier for this assessment. This identifier is contained in all related replies from the sensor. performance_assessment_parameters_type **request** Details of the assessment

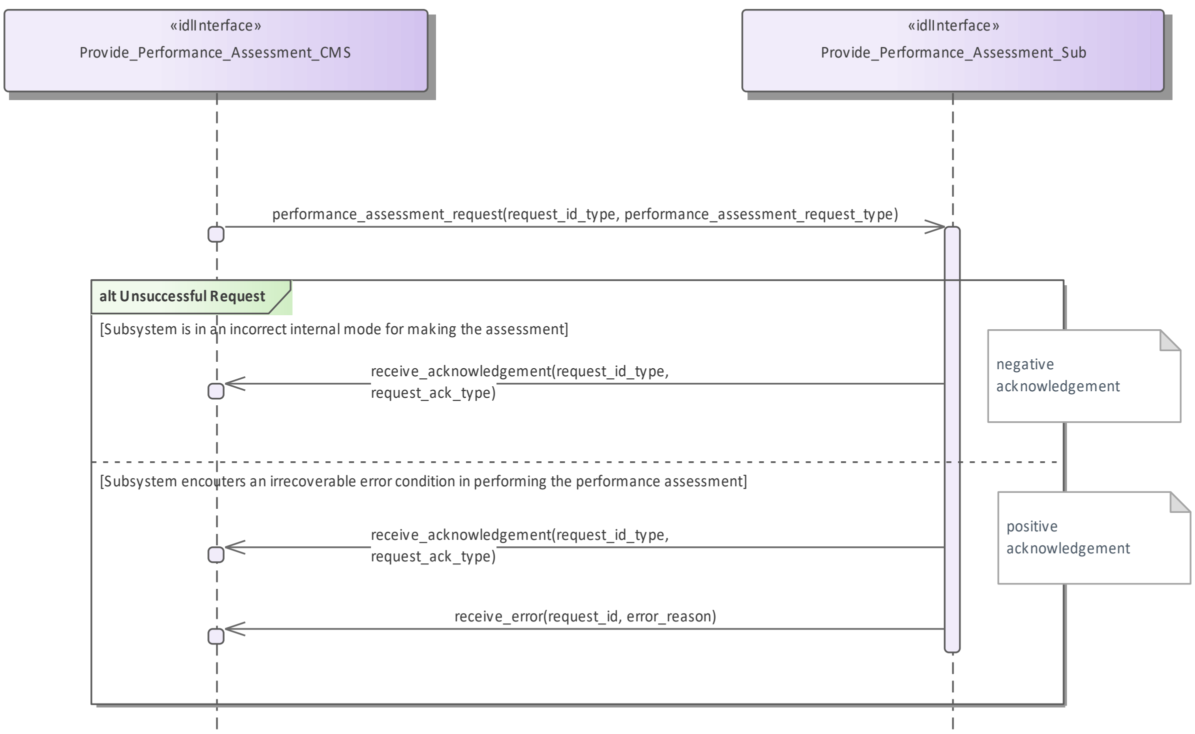


Figure 7.202 Alternate Flow - Provide_Performance_Assessment (Interaction diagram)

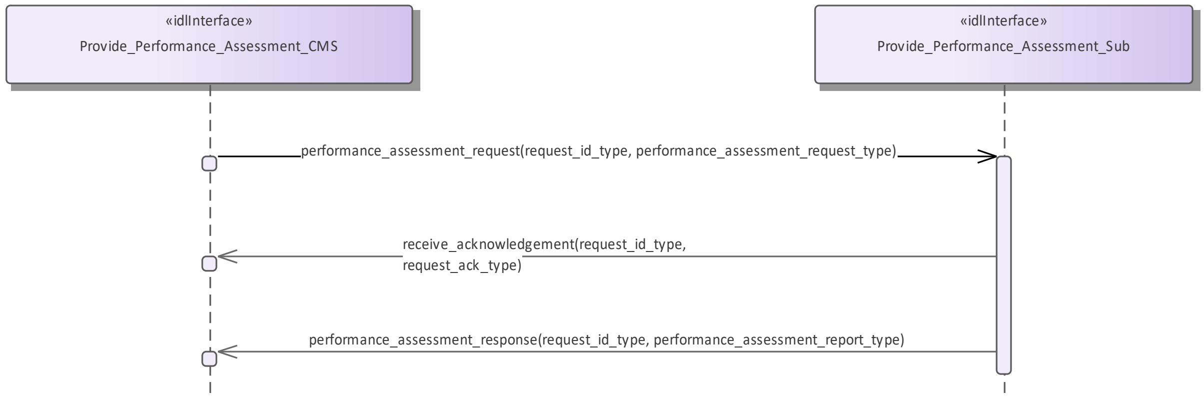


Figure 7.203 Basic Flow - Provide Performance Assessment (Interaction diagram)

7.9.8.4 Provide_Jammer_Assessment

Parent Package: Sensor_Performance

Provide_Jammer_Assessment_CMS
Type: Interface
Package: Provide_Jammer_Assessment
This interface describes the process whereby the subsystem provides a periodic assessment of the effects of actual jamming on the detection and tracking performance of the subsystem. The actual subsystem performance vs the nominal (see Provide Nominal Performance) shall be reported so that this data is current and real-time. This should include the effects on (spatial) coverage caused by any jamming. The impact on frequencies used e.g. operating band limitations is dealt with in Provide Interference Reports
Mastership is not required.
The radar need not be radiating in the ONLINE state but shall at least be receiving. The subsystem VOI (volume of interest) or other filter mechanisms might be supplied in a request to the subsystem.
The kind of information which could be provided in the returned assessment, depending on any jamming mitigation strategy (frequency agility, moving target indication, low side-lobe levels, main beam or side-lobe cancellation, side-lobe blanking etc.) might then include:

- Noise floor pre-/post-jammer cancellation, as applicable
- Degradation in detectability (compared with the nominal)

Pre-condition: Technical State: The subsystem is in the technical state ONLINE
Pre-condition: Subsystem Services: The Provide Subsystem Services Service has been successfully executed
Pre-condition: Register Interest: The Register Interest Service has completed successfully.
Post-condition: Success: CMS has received Jamming Effect Assessments
Post-condition: No Success: The CMS has not received Jamming Effect Assessments.

Table 7.356 - Methods of Interface Provide_Jammer_Assessment_CMS

Method	Notes	Parameters
jammer_assessment_response()		request_id_type **request_id** performance_assessment_report_type **report**

Provide_Jammer_Assessment_Sub
Type: Interface
Package: Provide_Jammer_Assessment

Table 7.357 - Methods of Interface Provide_Jammer_Assessment_Sub

Method	Notes	Parameters
jammer_assessment_request()		request_id_type **request_id** performance_assessment_parameters_type **jammer_assessment_request**

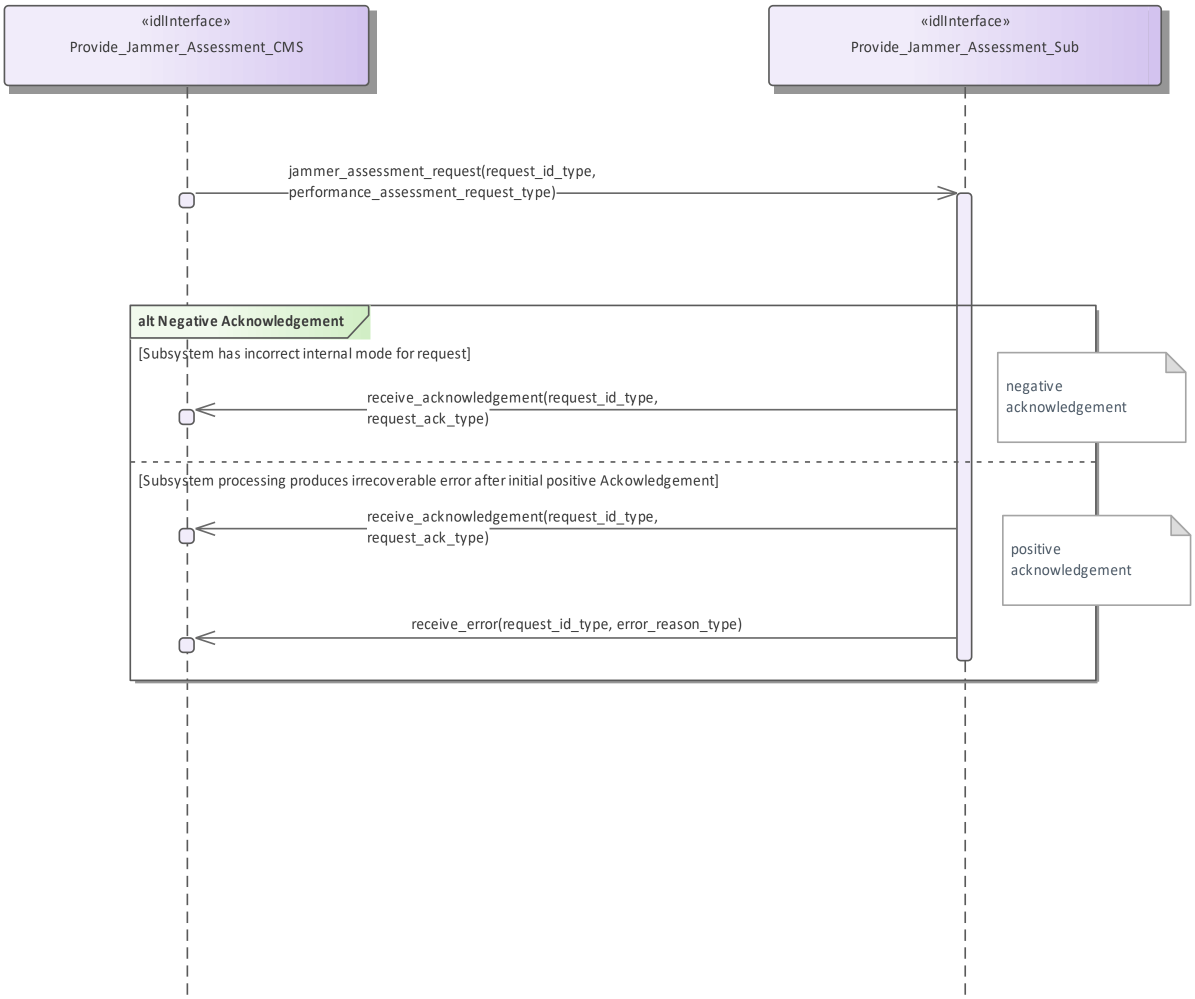


Figure 7.204 Alternate Flow - Provide Jammer Assessment (Interaction diagram)

Figure 7.205 Basic Flow - Provide Jammer Assessment (Interaction diagram)

7.9.9 Track_Reporting

Parent Package: Sensor_Services
This package contains interfaces for the Track Reporting service.

7.9.9.1 Filter_Tracks

Parent Package: Track_Reporting

Filter_Tracks_CMS
Type: Interface
Package: Filter_Tracks
The interface to the CMS for receiving information relating to the filters used to control which tracks are made available to other network segments.
The plot (and track) sharing architecture recognizes that connectivity between different platforms hosting sensors may not support the bandwidth required to share all plot and track updates. It is possible for a sensor also to provide the networking functionality in which case it is providing an additional role in the interface.

Table 7.358 - Methods of Interface Filter_Tracks_CMS

Method	Notes	Parameters
report_track_filter()	Reports the parameters of one of the filters that are active for tracks in the communication and networking subsystem. Tracks are transmitted or received, according to their mode, if they pass the conditions of at least one of the active filters.	filter_id_type **filter_id** The identifier for the filter track_filter_parameters_type **filter** The criteria for the filter
track_filter_removed()	Reports that a particular track filter has been removed.	filter_id_type **filter_id** The identifier of the filter removed

Filter_Tracks_Sub
Type: Interface
Package: Filter_Tracks
The interface to the subsystem for receiving updates to the filters used to control which tracks are made available to other network segments.

The plot (and track) sharing architecture recognizes that connectivity between different platforms hosting sensors may not support the bandwidth required to share all plot and track updates.
In this use case the subsystem is the network component providing connectivity to other platforms, as distinct from the local sensors providing the plots. It is possible for a sensor to also provide the networking functionality, in which case it is providing an additional role in the interface.

Table 7.359 - Methods of Interface Filter_Tracks_Sub

Method	Notes	Parameters
add_track_filter()	Adds an active filter for tracks to the communication and networking subsystem.	request_id_type **request_id** unique reference for the request track_filter_parameters_type **filter** the values to be used to filter tracks by
remove_track_filter()	Removes a filter for tracks from the communication and networking subsystem.	request_id_type **request_id** The unique reference for the request filter_id_type **filter_id** The identifier for the filter to be removed

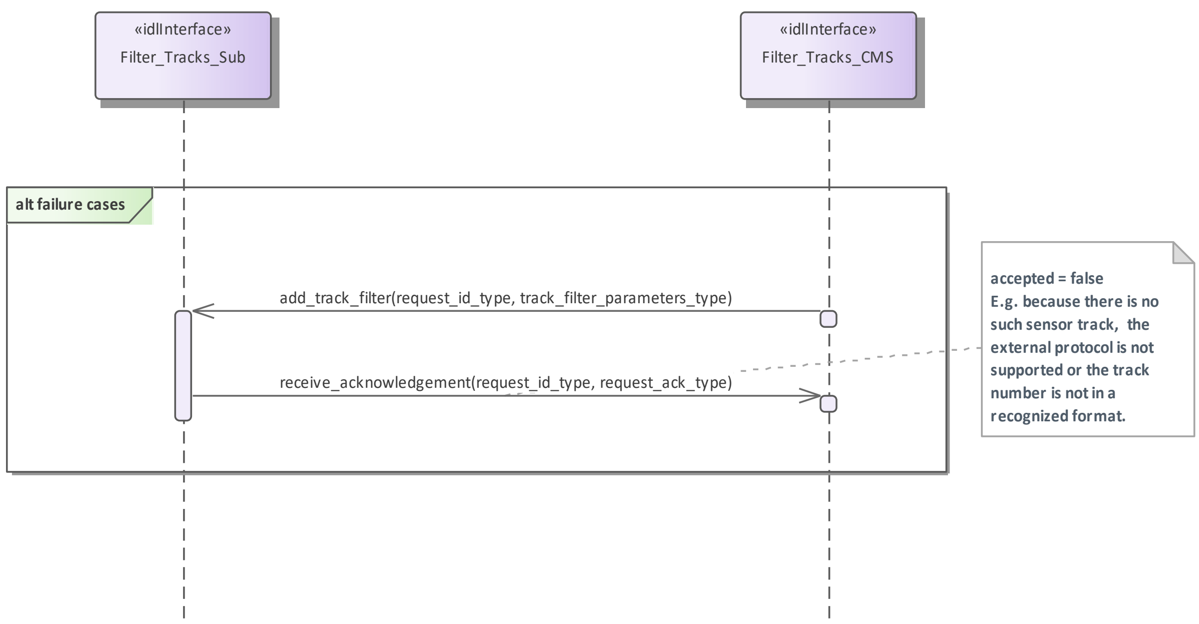


Figure 7.206 Alternative Flow Filter_Tracks (Interaction diagram)

Figure 7.207 Basic Flow Filter_Tracks (Interaction diagram)

7.9.9.2 Label_Tracks

Parent Package: Track_Reporting

Label_Tracks_CMS
Type: Interface
Package: Label_Tracks
The interface to the CMS for track number labeling. The subsystem uses the mappings received in transmissions to other platforms. This enables distributed local pictures to be formed that are coherent with other protocols being used for sharing data.

Table 7.360 - Methods of Interface Label_Tracks_CMS

Method	Notes	Parameters
external_track_label_response()	MethodTag: ea_guid = {63D77188-01A4-484c-AC10-F2C575C4AF6C}	request_id_type **request_id** The unique identifier for the request sensor_track_id_type **sensor_track_id** The identifier for the sensor track external_track_number_type **external_track_number** The external track numbers labelling the sensor track

Label_Tracks_Sub
Type: Interface
Package: Label_Tracks

The interface to the subsystem for track number labeling. The CMS instructs the subsystem responsible for transmission of sensor tracks to other platforms to label such sensor tracks with the track numbers of any external protocols on which the sensor track data is being transmitted or otherwise corresponds.
The subsystem acknowledges the request and responds with the sensor track's mapping to external track numbers used for transmission to other platforms.

Table 7.361 - Methods of Interface Label_Tracks_Sub

Method	Notes	Parameters
label_track_for_external_protocol()	This operation is used to instruct the subsystem to label a sensor track with an external track number when transmitting it off-platform. MethodTag: ea_guid = {0751D342-4ED1-4513-8FA6-5B97A5FFC475}	request_id_type **request_id** The unique identifier for the request sensor_track_id_type **sensor_track_id** The identifier for the sensor track external_track_number_type **external_track_number** The track number on an external protocol to label the track with
unlabel_track_for_external_protocol( )	This operation is used to instruct the subsystem to no longer label a sensor track with a track number for a particular external protocol when transmitting it off-platform. MethodTag: ea_guid = {7C320CF0-ABD8-4a76-998D-D2077C43497F}	request_id_type **request_id** The unique identifier for the request sensor_track_id_type **sensor_track_id** The identifier for the sensor track String **external_protocol** The name of the external protocol

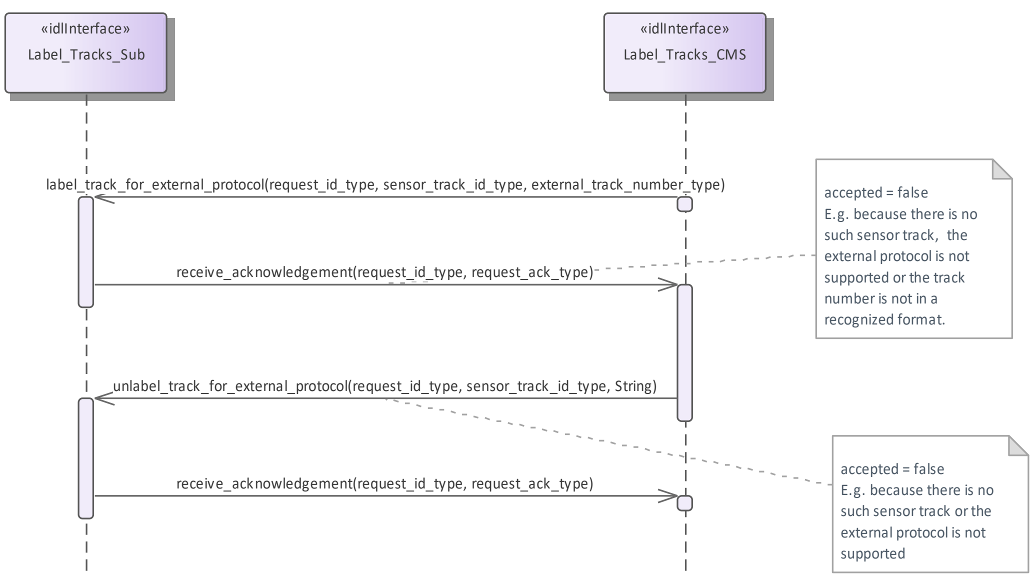


Figure 7.208 Alternate Flow Label_Tracks (Interaction diagram)

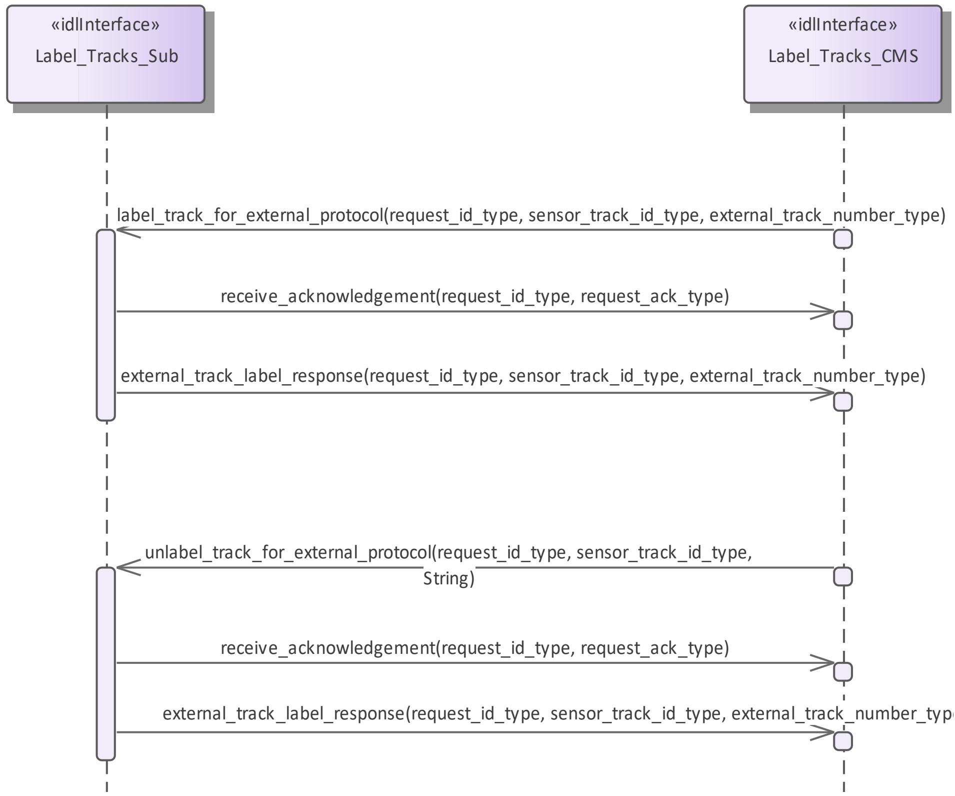


Figure 7.209 Basic Flow Label_Tracks (Interaction diagram)

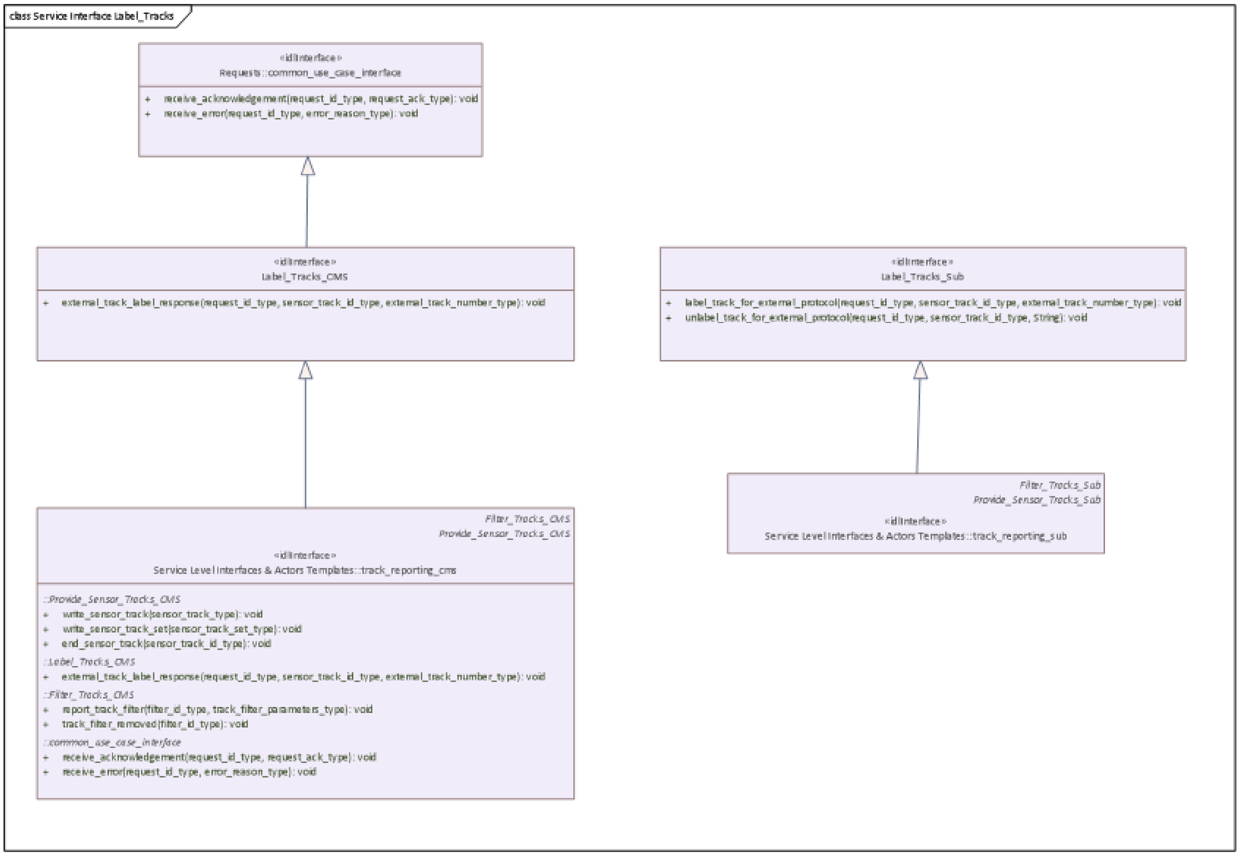


Figure 7.210 Service Interface Label_Tracks (Class diagram)

7.9.9.3 Provide_Sensor_Tracks

Parent Package: Track_Reporting

Provide_Sensor_Tracks_CMS
Type: Interface
Package: Provide_Sensor_Tracks
This service allows the CMS to obtain an overview of (real and/or simulated) air / land / space / surface objects observed or simulated. Information may cover all aspects of a track such as kinematic and amplifying information.
The service does not cover:

- additional track information provision dedicated for engagement support,
- special search functions such as cued search, volume search and horizon search (however, if such a search function is initiated by means of another service, the tracks shall be provided by this service),

Although the service focuses on radar as an example of a sensor, the service also applies to other sensors, like IR/EO sensors and ECM/ESM sensors.

The actor is the Combat Management System.

The service starts when:

- if the service does provide registration capabilities: the service "Register interest" has completed successfully, or
- if the service does not provide registration capabilities: the service "Provide subsystem services" has completed successfully for this service.

The sensor provides, periodically or on event, a set of sensor tracks observed by the sensor. These may be sensor point or bearing tracks. The set of sensor tracks includes:

- Track updates of existing and new sensor tracks. These are provided when there are sufficient measurements (e.g. plots) in the last observation cycle, which may be associated with the sensor track.
- Dead-reckoned tracks. These are sensor track updates for which in the last observation cycle there are no measurements that may be associated with the sensor track. For dead-reckoned tracks, the sensor track information (e.g. kinematics) is extrapolated. The dead-reckoned tracks may become"normal" tracks again if, in the next scan, there are measurement(s) that may be associated with the track. Alternatively, dead-reckoned tracks (after n unsuccessful scans) may become lost tracks.
- Lost tracks. These are sensor track updates that are reported once, if in the last n scans, there are no measurements that may be associated with the sensor track. The value of n is typically a sensor parameter that is managed by the service "Manage subsystem parameters".

Some sensors are not capable of reporting lost and/or dead-reckoned tracks.
The sensor may also provide single sensor tracks periodically or on event.

The service ends with success when:

- if the service does provide registration capabilities: the service "Register interest" has completed successfully for a deregistration request, or
- if the service does not provide registration capabilities: the sensor is shutdown using service "Shut down".

Pre-condition: Sensor health state: The sensor and the service need to be in the health state AVAILABLE or DEGRADED

Pre-condition: Sensor parameters: The relevant sensor parameters (e.g. allowed frequencies, transmission sectors) need to be set[1].

----

[1] The manner in which this is done is described in other services of the OARIS ("Manage frequency usage", "Manage transmission sectors", "Control emissions" and "Manage subsystem parameters").

Table 7.362 - Methods of Interface Provide_Sensor_Tracks_CMS

Method	Notes	Parameters
write_sensor_track()	The method represents a write of a single sensor track (air, land, space or surface) to the CMS. The write may be periodic or not.	sensor_track_type **the_sensor_track**
write_sensor_track_set()	The method represents a single write of a set of sensor tracks to the CMS. The write may be: - periodic or not - include all tracks observed during a sensor scan - be an update of just one track (a set of 1) if this is how the sensor works	sensor_track_set_type **the_track_set**
end~~delete~~_sensor_track()	The method represents a deletion of a single sensor track (air, land, space or surface) to the CMS. After a deletion, no further writes for that sensor track instance are made. Subsystems may subsequently use the same sensor_track_id to denote another sensor track instance possibly after a 'cooling off period'. Such behavior is implementation	sensor_track_id_type **the_sensor_track_id**

	specific.	

Provide_Sensor_Tracks_Sub
Type: Interface
Package: Provide_Sensor_Tracks

Table 7.363 - Methods of Interface Provide_Sensor_Tracks_Sub

Method	Notes	Parameters
prioritize_track()	CMS requests the subsystem to report the referenced sensor track with the stated priority. MethodTag: Issue =	request_id_type **request_id** sensor_track_id_type **sensor_track_id** track_priority_type **priority**
remove_track_priority()	The CMS requests the subsystem to report the track with the default priority for that subsystem.	request_id_type **request_id** sensor_track_id_type **sensor_track_id**

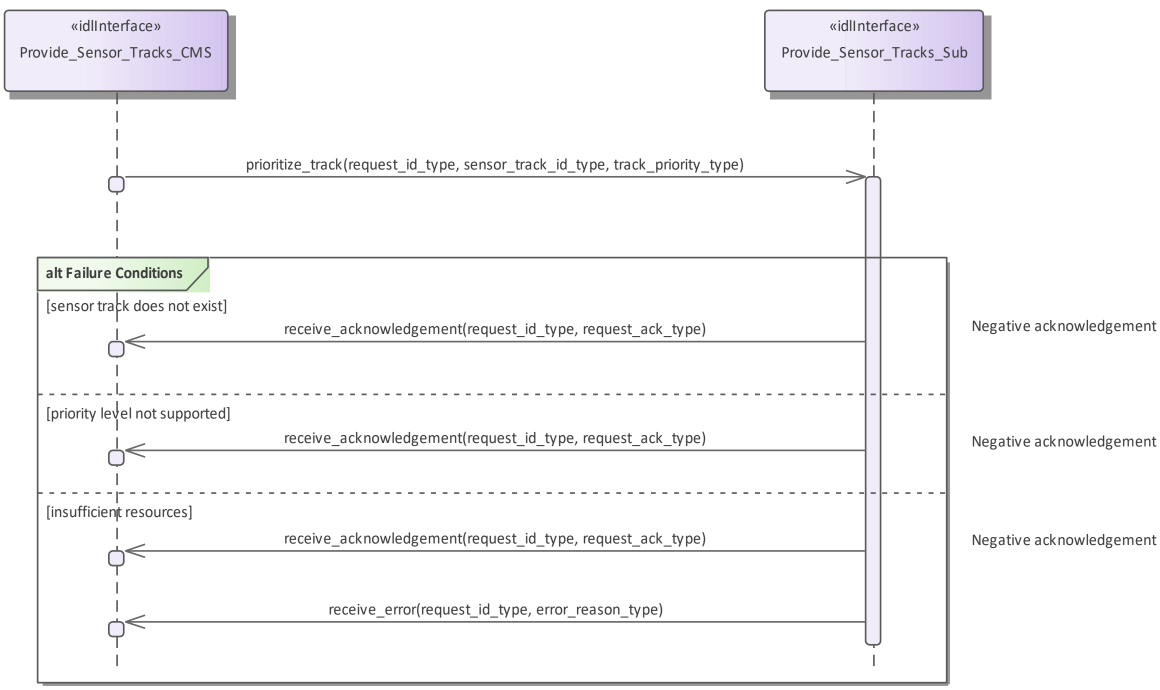


Figure 7.211 Alternative Flow - Track Prioritization (Interaction diagram)

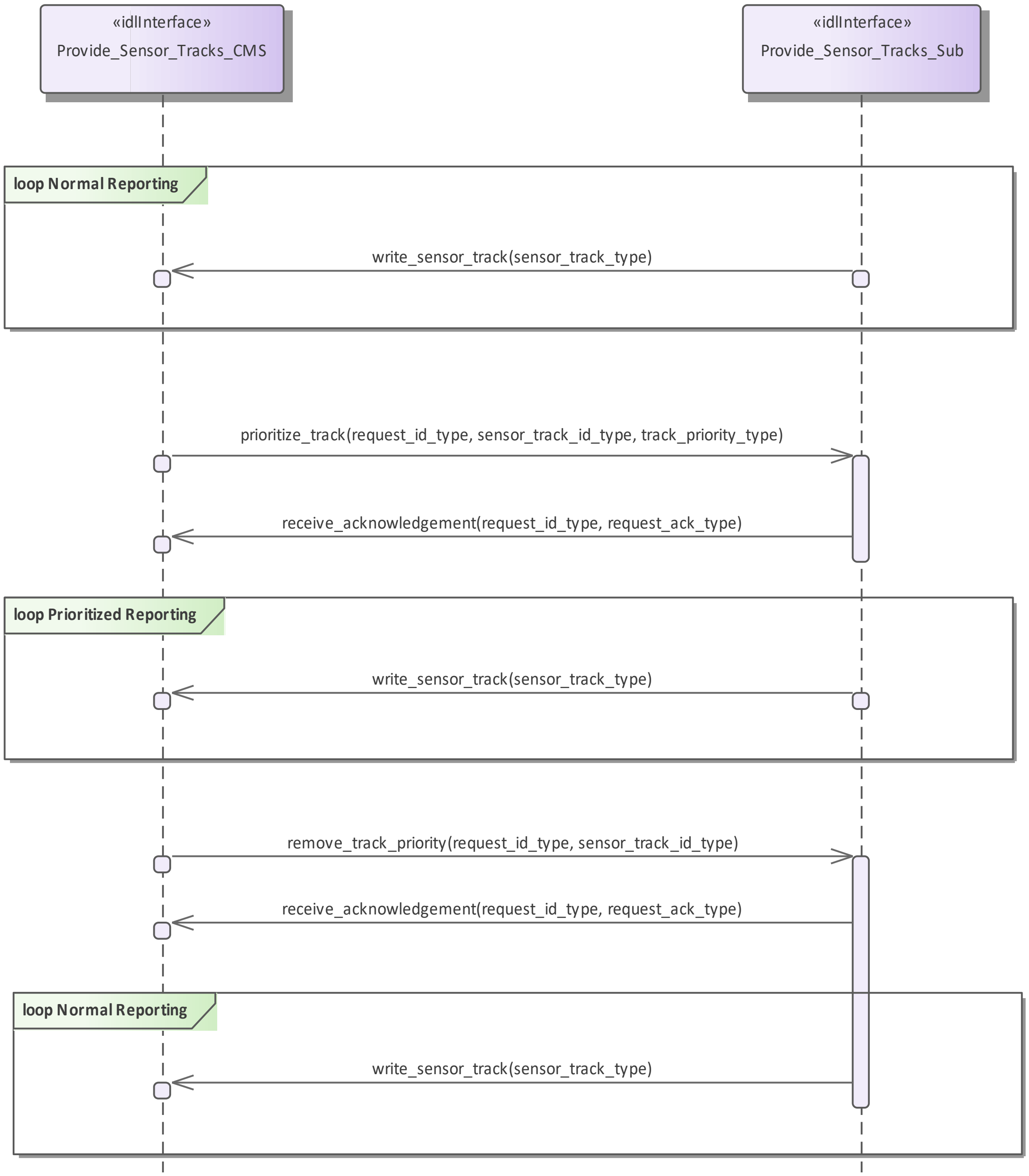


Figure 7.212 Basic Flow - Change Priority (Interaction diagram)

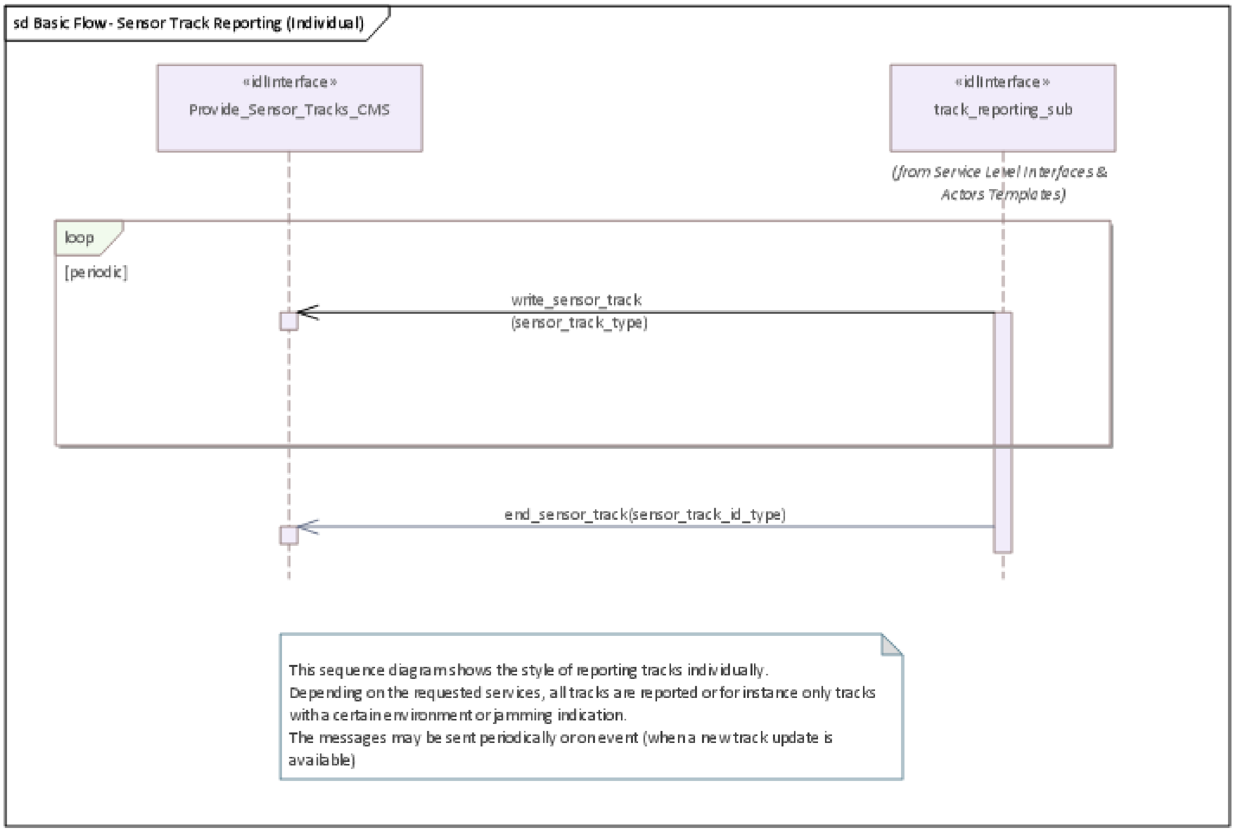


Figure 7.213 Basic Flow - Sensor Track Reporting (Individual) (Interaction diagram)

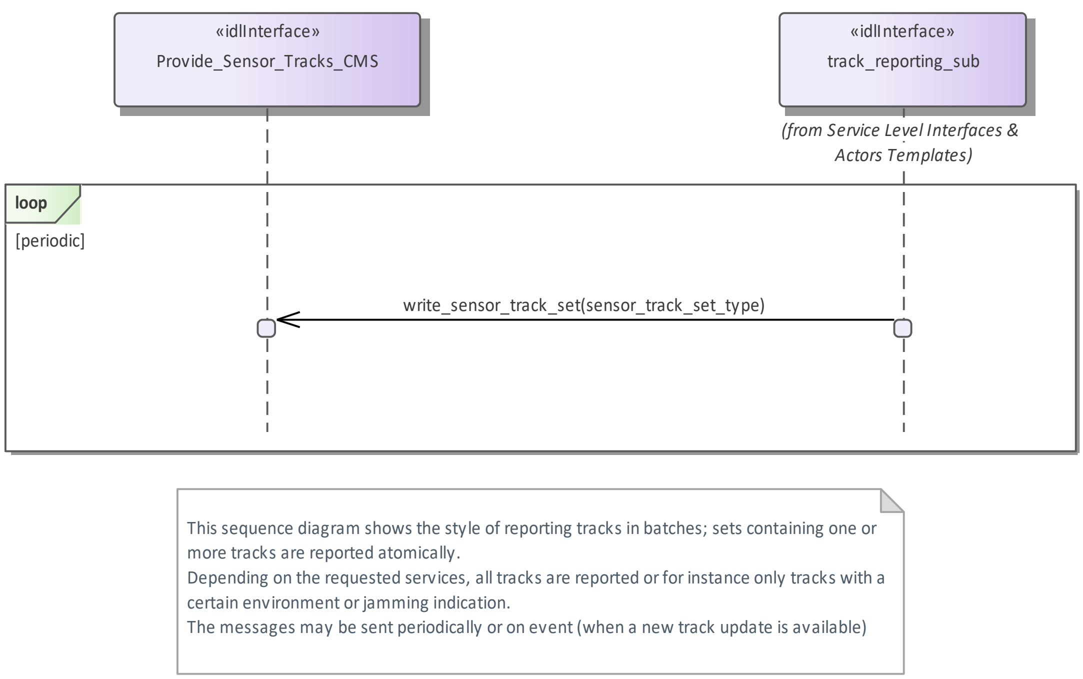


Figure 7.214 Basic Flow - Sensor Track Reporting (Sets) (Interaction diagram)

7.9.10 Tracking_Control

Parent Package: Sensor_Services
This package contains interfaces for the Tracking Control service.

7.9.10.1 Delete_Sensor_Track

Parent Package: Tracking_Control
This package contains interfaces for the Delete Sensor Track service.
Delete_Sensor_Track_CMS
Type: Interface
Package: Delete_Sensor_Track
The sensor is requested to remove a specified track from its internal Track Data Base; obviously the deleted track may come back (with another track identification number) within a few seconds if it was a living track.

Pre-condition: Mastership Required: CMS has mastership of the sensor

Pre-condition: Subsystem Services: *Provide subsystem services* is successfully passed

Pre-condition: Tracking capability: Tracking capability is supported by the sensor, and CMS is aware that actually the sensor may delete that track

Post-condition: Success: CMS is informed of the successful deletion of the required track, and the next track reporting shall no contain the deleted track. Obviously, the deleted track may come back within a few seconds if it was a living target, but with another identification number.

Post-condition: No Success: CMS is informed of the request rejection and of the denial reason. No impact on the sensor track management evolution.

Delete_Sensor_Track_Sub
Type: Interface
Package: Delete_Sensor_Track
This is the Subsystem interface for deleting sensor tracks.

Table 7.364 - Methods of Interface Delete_Sensor_Track_Sub

Method	Notes	Parameters
delete_track()	Method used by the CMS to send a track deletion request, specifying the identification number of the track to be deleted.	sensor_track_id_type **trackId** request_id_type **request_id**

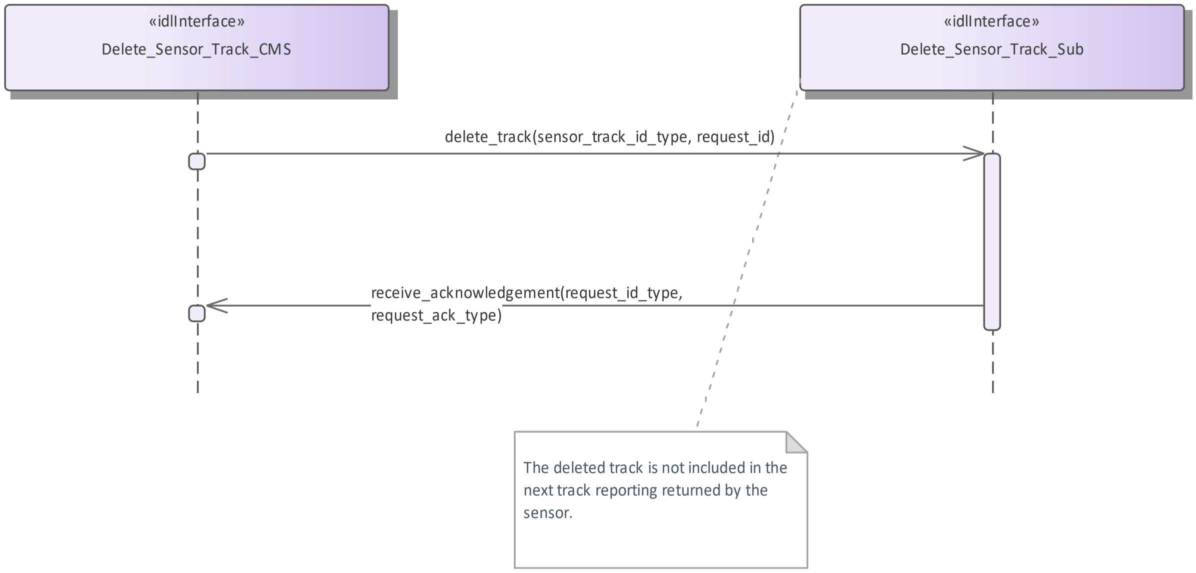


Figure 7.215 Basic Flow - Delete Sensor Track (Interaction diagram)

Figure 7.216 Alternative Flow - Delete Sensor Track (Interaction diagram)

7.9.10.2 Receive_Track_Information

Parent Package: Tracking_Control

This package contains interfaces for the Receive Track Information service.

Receive_Track_Information_CMS

Type: Interface

Package: Receive_Track_Information
CMS may provide information belonging to a sensor track in order to enable for a coordinated presentation of the sensor track both on CMS consoles and a dedicated radar console. The track information which may be supplied are:

1. External track identification number
2. Additional Information – this is not specified as part of the interface, candidate information includes:
- Track type
- Track priority
- Track Identification Category Assigned (Pending, Friend, Assumed Friend, Neutral, Unknown, Suspect, Hostile)

Track identities management
Each sensor track shall have an "Internal Track Identification Number" and may one or more additional "External Track Identification Numbers". The former shall be assigned by the sensor when the track is formed and, as long as the track is alive, it cannot be changed for any reason. The latter shall be set to "none" when the track is formed and then overwritten, during the track life, to report the track identity/ies externally assigned to the track.
All track identification numbers shall be reported together with the track data, but the track identification shall be made through the "Internal Track Identification Number".
Pre-condition: Mastership Required: CMS has mastership of the sensor
Pre-condition: Subsystem Services: *Provide subsystem services* is successfully passed
Pre-condition: Tracking capability: Tracking capability is supported by the sensor, and CMS is aware that actually the sensor may manage that track
Pre-condition: Technical State: Sensor is working in Operational
Post-condition: Success: CMS is informed of the successful execution of the request, and the next track reporting shall contain the identified track with the provided information.
Post-condition: No Success: CMS is informed of the request rejection and of the denial reason. No impact on the sensor track management evolution.
Receive_Track_Information_Sub
Type: Interface
Package: Receive_Track_Information
This is the Subsystem interface for receiving track information.

Table 7.365 - Methods of Interface Receive_Track_Information_Sub

Method	Notes	Parameters
insert_info_track()	Method used by the CMS to send a receive track information request, specifying the track identification number and related track information.	request_id_type **request_id** sensor_track_id_type **trackId** track_info_type **trackInfo**

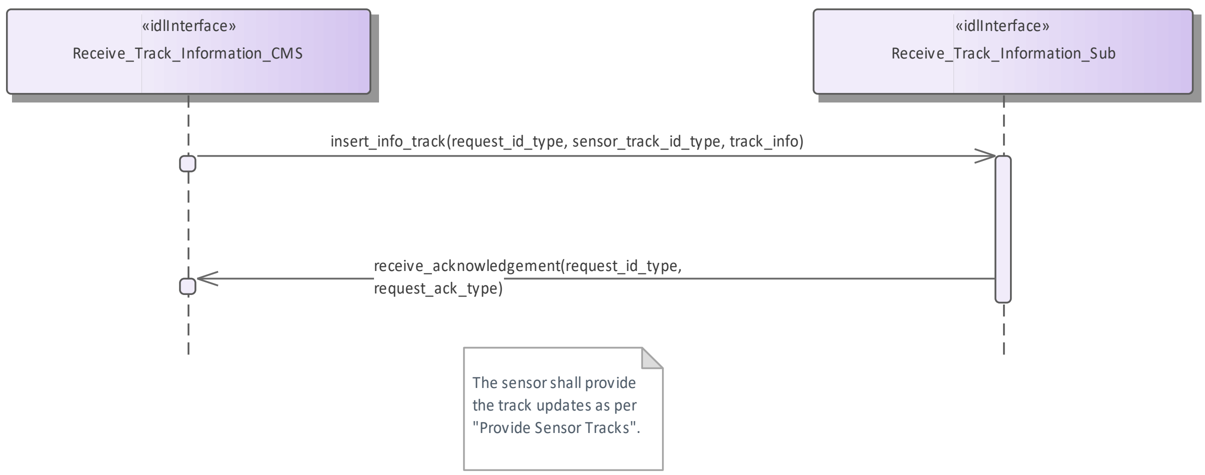


Figure 7.217 Basic Flow - Receive Track Information (Interaction diagram)

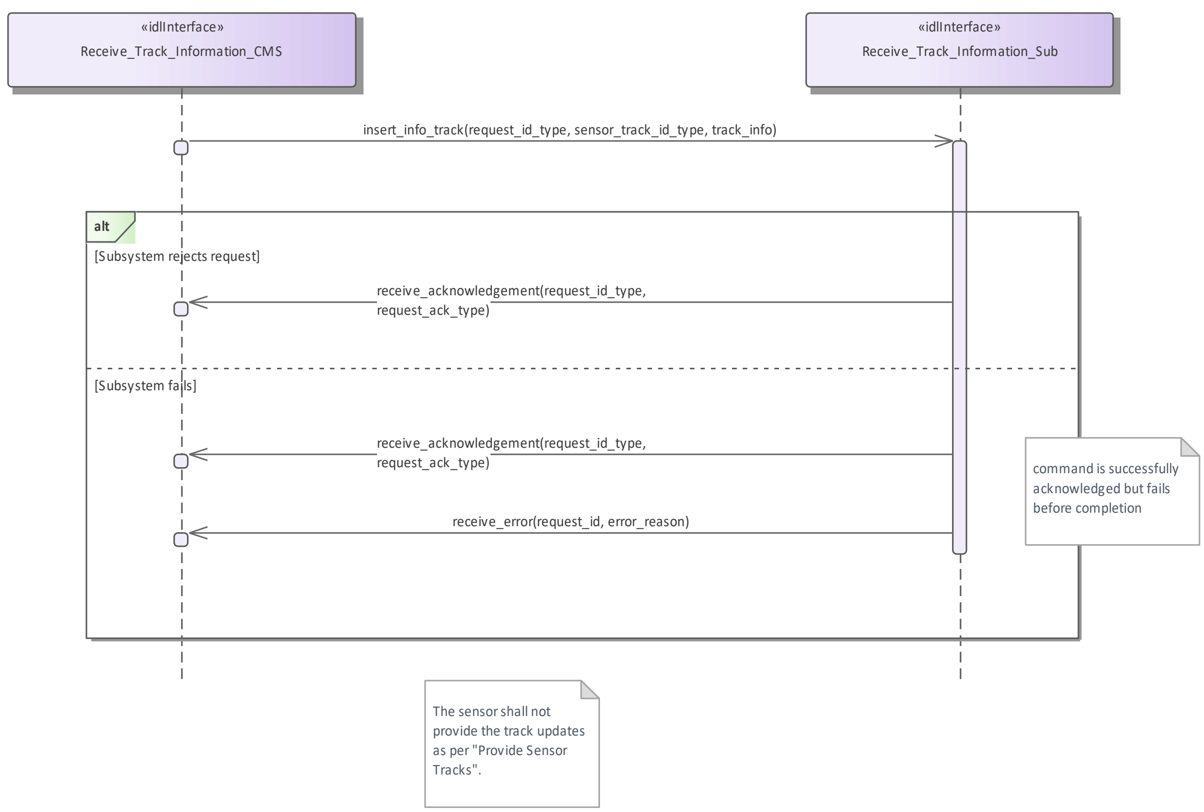


Figure 7.218 Alternative Flow - Receive Track Information (Interaction diagram)

7.9.10.3 Initiate_Track

Parent Package: Tracking_Control

This package contains interfaces for the Initiate Track service.

Initiate_Track_CMS

Type: Interface

Package: Initiate_Track

The sensor is requested to start tracking on a new target based on given information, such as positional data and additionally also kinematic data. Sensor replies indicating the request acceptance or rejection. If accepted, the initiation of a new track shall be attempted as required, and the relevant result shall be reported later through an “externally designated track initiation report” containing the identification number of the resulting track (if any).

Additional Information

Data reported in the “externally designated track initiation request”

The provided information depends on the sensor type and its capabilities, typically they are:

- Identification number of the designation (mandatory)
- Position and time (mandatory)
- Accuracy of the provided positional data (optional)
- Velocity and relevant accuracy (optional)
- Track characteristics (optional)

Data reported in the “externally designated track initiation report”

The purpose is this report is to inform CMS about the final result of the track initiation request, i.e. it reports to CMS if the track has been successfully initiated or not, and (in case of success) the identification number of the new formed track.

The provided information depends on the sensor type and its capabilities, typically they are:

- Identification number of the designation (mandatory)
- Initiation result (mandatory)
- Identification number of the initiated track, if any (mandatory)
- other info (optional).

Pre-condition: Mastership Required: CMS has mastership of the sensor

Pre-condition: Subsystem Services: *Provide subsystem services* is successfully passed

Post-condition: Success: The setting of the tracking zones has been modified according to the request and CMS is informed that this is the case.

Post-condition: No Success: The setting of the tracking zones is unchanged with respect to the original one and CMS is informed that this is the case.

Table 7.366 - Methods of Interface Initiate_Track_CMS

Method	Notes	Parameters
report_track()	Method used by the sensor to issue an "externally designated track initiation report” containing data of the successfully initiated track.	request_id_type **request_id** sensor_track_id_type **id_report**

Initiate_Track_Sub

Type: Interface

Package: Initiate_Track

This is the Subsystem interface for initiating tracks.

Table 7.367 - Methods of Interface Initiate_Track_Sub

Method	Notes	Parameters
initiate_track()	Method used by the CMS to send an "externally designated track initiation request”, specifying a timed position and kinematic.	request_id_type **request_id** system_track_type **track_info**

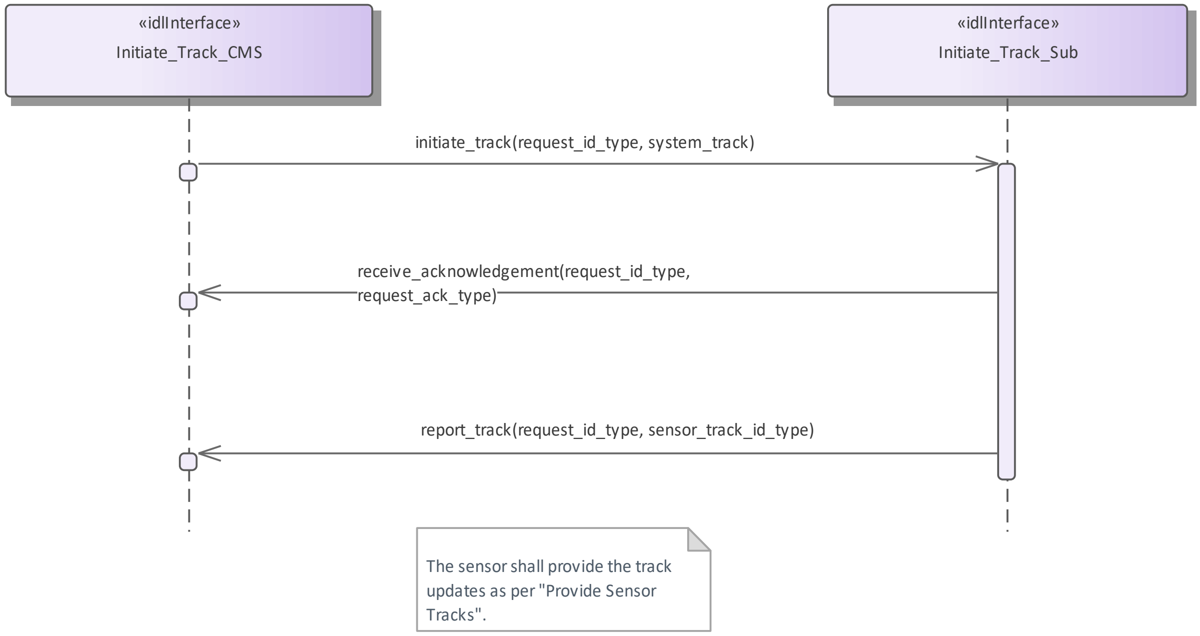


Figure 7.219 Basic Flow Initiate Track (Interaction diagram)

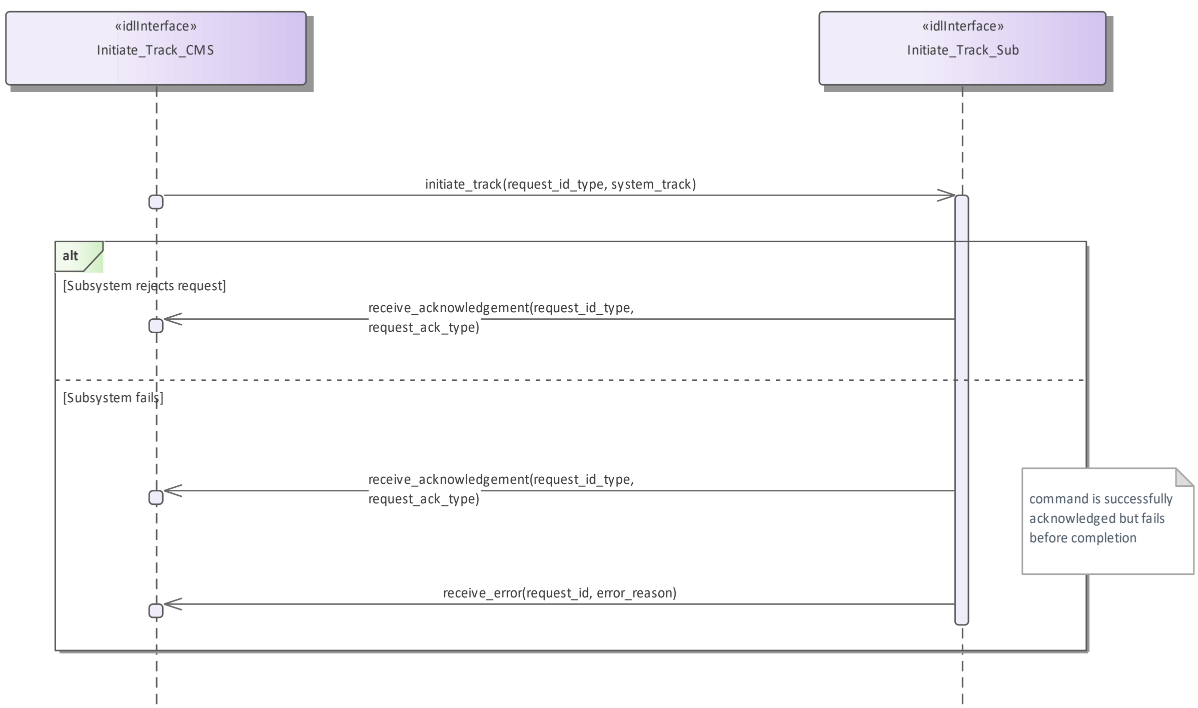


Figure 7.220 Alternative Flow - Initiate Track - loss of mastership (Interaction diagram)

7.9.10.4 Manage_Tracking_Zones

Parent Package: Tracking_Control

This package contains interfaces for the Manage Tracking Zones service.

Manage_Tracking_Zones_CMS
Type: Interface
Package: Manage_Tracking_Zones
This controls the sensor tracking behaviour in selected zones, which may be 1D (delimited in azimuth only), 2D (have additional elevation bounds) or 3D (have further range bounds). Depending on the zone type the sensor may be requested to modify its normal tracking behaviour, such as enable/disable the capability to auto initiate new tracks, or the capability of managing Track-On-Jammer. A list of typical tracking zones is

- Automatic Track Initiation Zones

zones where the sensor is allowed to auto initiate new tracks. Depending on the sensor type and its capabilities, such a type of zones may be delimited in azimuth only, or both in azimuth and elevation, or may have further range bounds, and in some cases also additional constraints (such as target type, velocity bounds, etc.).

- Track-On-Jammer Sectors

sectors where the sensor is allowed to manage Track-On-Jammer. Depending on the sensor type and its capabilities, such a type of sectors may be delimited either in azimuth only or both in azimuth and elevation.

- Multipath Devoted Tracking Sectors

sectors where the sensor is required to use, for tracking activities, devoted waveforms to reduce the multipath effects. This capability is usually provided by multifunctional radars. Such a type of sectors is usually limited in azimuth only, below a defined elevation.

The supported tracking zone types (names and characteristics) differ from sensor to sensor, so they shall be handled as configuration parameters. They shall be offered to the operator to enable him for a selection and then transferred to the sensor to achieve the intended response.

Special Requirements
Provision of the sensor tracking zones setting
Sensor shall keep CMS informed about the actual setting of the tracking zones and its changes (if any).

It is the CMS's responsibility to initiate the determination of initial state by making a request for information to the subsystem.

Additional Information
Lack of mastership
In the case where CMS does not have mastership of the sensor, CMS shall be informed about the actual setting of the tracking zones and its changes (if any).

Pre-condition: Mastership Required: CMS has mastership of the sensor

Pre-condition: Subsystem Services: *Provide subsystem services* is successfully passed

Pre-condition: Tracking zones setting: CMS is aware of which types of tracking zones the sensor may manage and of their current setting.

Post-condition: Success: The setting of the tracking zones has been modified according to the request and CMS is informed that this is the case.

Post-condition: No Success: The setting of the tracking zones is unchanged with respect to the original one and CMS is informed that this is the case.

Table 7.368 - Methods of Interface Manage_Tracking_Zones_CMS

Method	Notes	Parameters
tracking_zone_setting()	Method used by the CMS to send an enable/disable tracking zone request to the sensor.	request_id_type **request_id** tracking_zone_set_type **setting_message**

Manage_Tracking_Zones_Sub
Type: Interface
Package: Manage_Tracking_Zones
This is the Subsystem interface for managing tracking zones.

Table 7.369 - Methods of Interface Manage_Tracking_Zones_Sub

Method	Notes	Parameters
set_tracking_zone()	Method used by the sensor to return the actual setting of the tracking zones modified according to the request.	request_id_type **request_id** tracking_zone_set_type **zone**

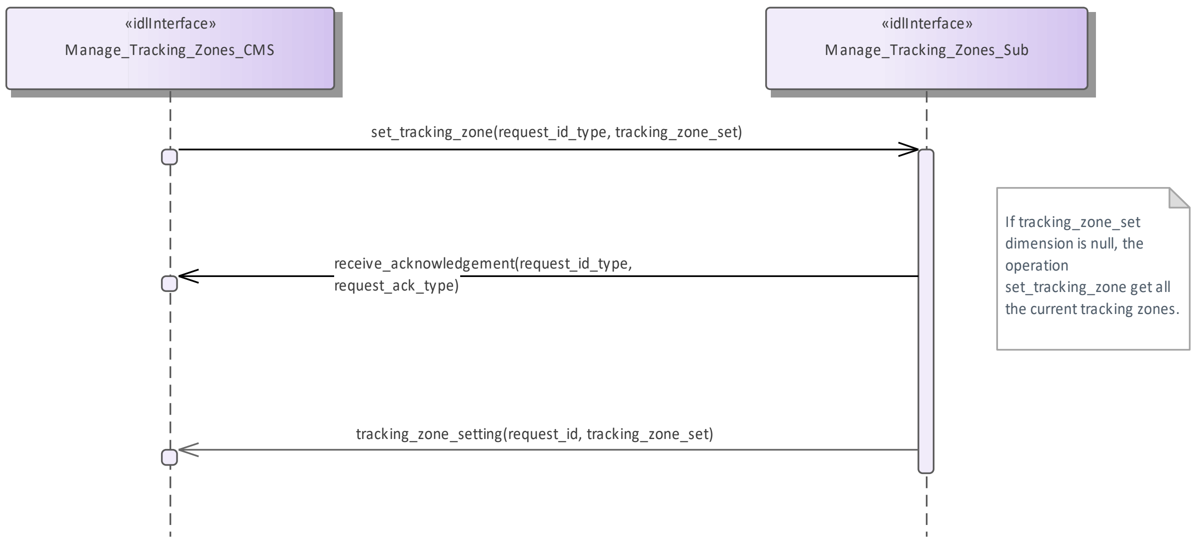


Figure 7.221 Basic Flow - Manage Tracking Zone - Enable/Disable (Interaction diagram)

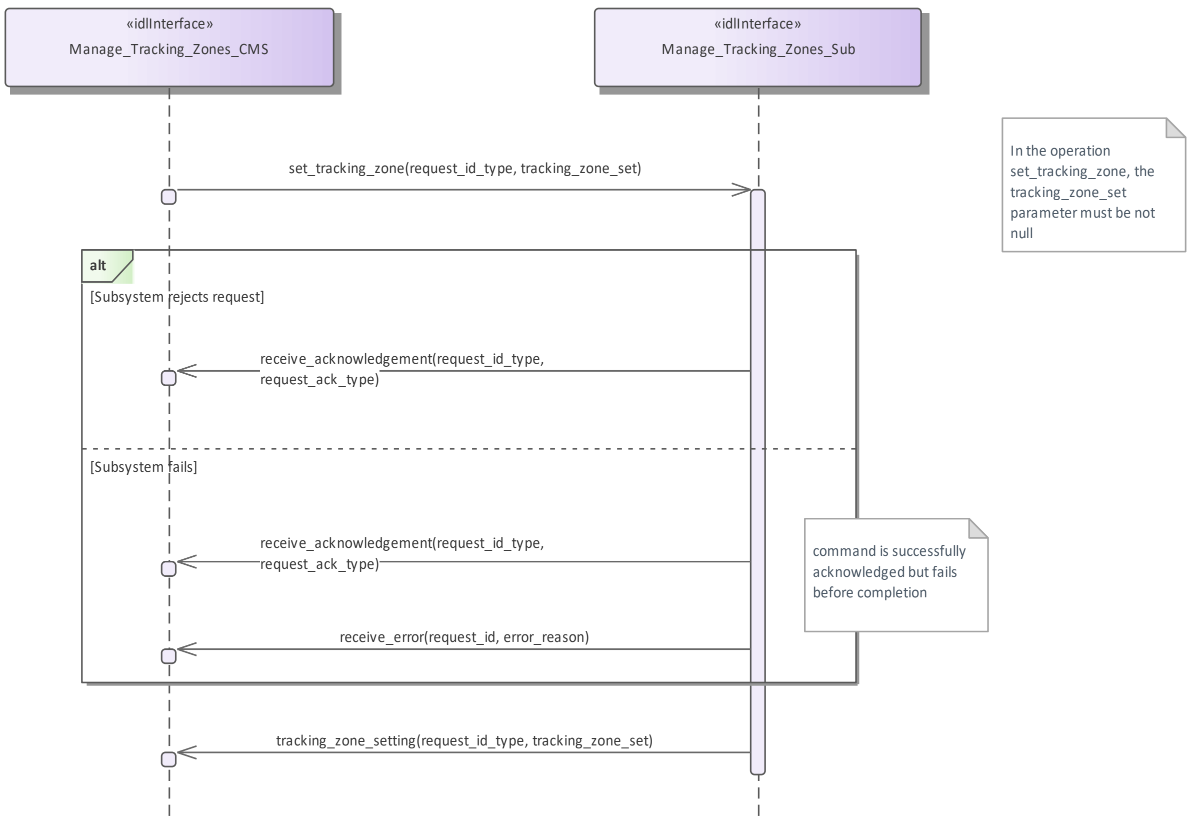


Figure 7.222 Alternative Flow - Manage Tracking Zone - Enable/Disable - loss of Mastership (Interaction diagram)

7.10 Radar_Services

Parent Package: Service_Interfaces
Contains services associated with the Radar Domain.

7.10.1 Air_Engagement_Support

Parent Package: Radar_Services

7.10.1.1 Provide_Projectile_Positional_Information

Parent Package: Air_Engagement_Support

Provide_Projectile_Positional_Information_CMS
Type: Interface
Package: Provide_Projectile_Positional_Information
Fire control radars suitable for Close-In-Weapon-Systems need the capability to observe the projectiles in flight, to measure at which distance they pass the target so that related shot corrections for the gun may be calculated, automatically. The measured distance in azimuth and elevation is called miss indication in the following.

This capability may be available in a non-close-in-weapon-system environment, too. It may also be available for phased-array radars.

Mastership of the subsystem must not have any impact upon the miss indication capability.

See also service 'Process Target Designation'.

Pre-condition: "Process Target Designation" was successfully carried out and a target is being tracked.:

Pre-condition: CMS must have mastership.:

Table 7.370 - Methods of Interface Provide_Projectile_Positional_Information_CMS

Method	Notes	Parameters
report_miss_indication()	Via this message, the subsystem reports to the CMS the miss indication.	miss_indication_data_type **MissIndicationData** request_id_type **RequestID**

Provide_Projectile_Positional_Information_Sub

Type: Interface

Package: Provide_Projectile_Positional_Information

Table 7.371 - Methods of Interface Provide_Projectile_Positional_Information_Sub

Method	Notes	Parameters
request_miss_indication()	Request the subsystem to report a miss indication.	request_id_type **RequestID** expected_hit_data_type **ExpectedHitData**

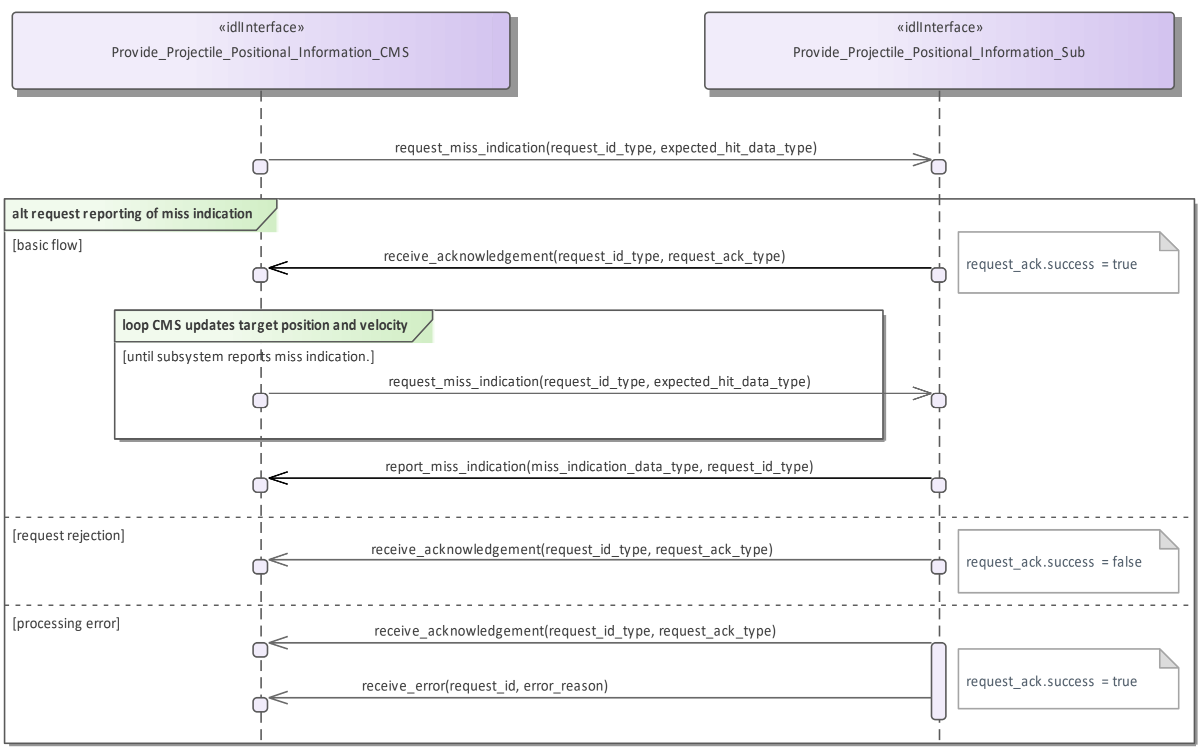


Figure 7.223 Provide projectile positional information - Request reporting of miss indications (Interaction diagram)

This sequence diagram shows how the CMS and the subsystem operate with each other during the operation "request reporting of miss indications" of the service 'Provide projectile position information'.

7.10.2 Engagement_Support

Parent Package: Radar_Services

7.10.2.1 Process_Target_Designation

Parent Package: Engagement_Support

Process_Target_Designation_CMS
Type: Interface
Package: Process_Target_Designation
Fire control radars are designed to perform one target engagement at a time with respect to an air, surface or land target and provide the necessary information for a fire control solution regarding that target.

The CMS selects a track and requests the fire control radar to acquire and track the target behind that track. If the acquisition is successful the radar starts tracking the target and reporting fire control information.

Some fire control radars provide information about one or more other targets appearing in its field of view and may even provide associated sensor tracks. This is, however, not within the scope of this service interface but covered by "Provide sensor tracks".

The fire control information may be plots and/or tracks, depending on the product.

On receiving the de-designation request the fire control radar stops following the target and stops providing fire control information.

Phased array radars may include fire control capabilities as well. If they do, they provide a number of ‘virtual fire control radars’. To the extent that these virtual fire control radars are comparable in function and performance, there may be no need for the CMS to select a specific fire control channel to be used for a particular engagement.

In the case where the CMS looses or releases mastership of the subsystem, the subsystems ceases all fire control activities.

A target designation to a weapon with its own fire control capabilities may be done in an analogous way. In that sense, the service (interface) may also be employed by weapon systems.
Pre-condition: CMS must have Mastership.:
Pre-condition: Technical state READY or ONLINE.:

Table 7.372 - Methods of Interface Process_Target_Designation_CMS

Method	Notes	Parameters
receive_fire_control_channel_released()	Via this message, the subsystem confirms the release of a target acquisition.	request_id_type **RequestID** fire_control_channel_id_type **FireControlChannelID**
receive_target_acquired()	Via this message, the subsystem confirms the target acquisition.	request_id_type **RequestID** sensor_track_id_type **TrackID** fire_control_channel_id_type **FireControlChannelID**
receive_target_dedesignation()	Via this message, the subsystem reports the de-designation of a target.	request_id_type **RequestID** sensor_track_id_type **TrackID**

Process_Target_Designation_Sub
Type: Interface
Package: Process_Target_Designation

Table 7.373 - Methods of Interface Process_Target_Designation_Sub

Method	Notes	Parameters
dedesignate_target()	The subsystem is requested to de-designate a fire control channel.	request_id_type **RequestID** fire_control_channel_id_type **FireControlChannelID**
designate_target_by_position()	The subsystem is requested to designate a fire control channel based on a position/kinematics.	request_id_type **RequestID** kinematics_type **PositionVelocity**
designate_target_by_track()	The subsystem is requested to designate a fire control channel based on a track.	request_id_type **RequestID** sensor_track_id_type **TrackID**

Sensor Track Reporting
Type: InteractionOccurrence
Package: Process_Target_Designation
The sensor track reporting itself is not covered in this service interface. See the corresponding service interface 'Sensor Track Reporting'.

Sensor Track Reporting
Type: InteractionOccurrence
Package: Process_Target_Designation
The sensor track reporting itself is not covered in this service interface. See the corresponding service interface 'Sensor Track Reporting'.

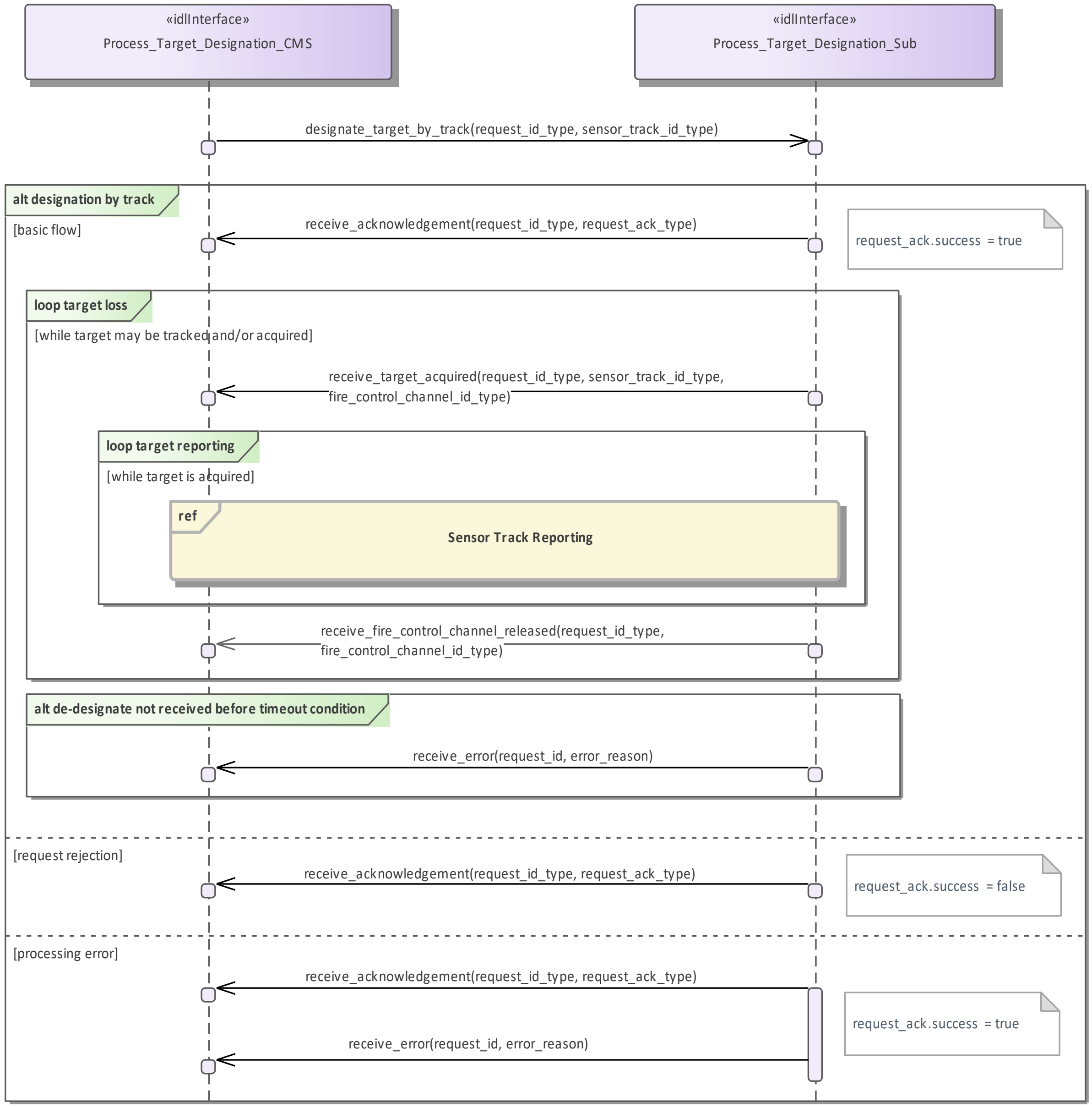


Figure 7.224 Process Target Designation - Designation by track (Interaction diagram)

This sequence diagram shows how the CMS and the subsystem operate with each other during the operation "designate (target) by track" of the service "Process Target Designation".

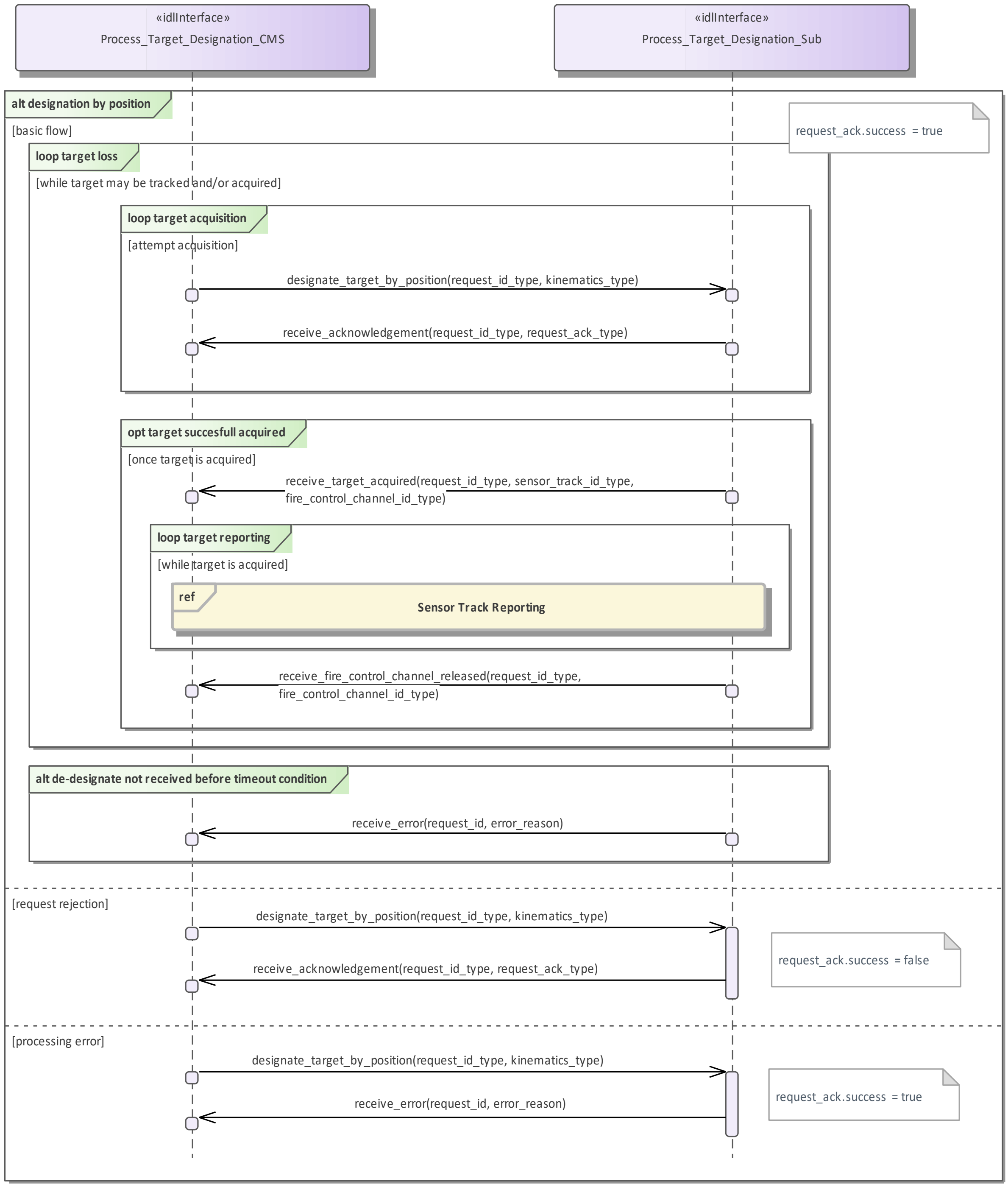


Figure 7.225 Process Target Designation - Designation by position (Interaction diagram)

This sequence diagram shows how the CMS and the subsystem operate with each other during the operation "designate (target) by position" of the service "Process Target Designation".

Figure 7.226 Process Target Designation - De-designation (Interaction diagram)

This sequence diagram shows how the CMS and the subsystem operate with each other during the operation "de-designate (target)" of the service "Process Target Designation". It applies to a fire control channel that has been designated by position or by track.

7.10.2.2 Support_Kill_Assessment

Parent Package: Engagement_Support

Support_Kill_Assessment_CMS
Type: Interface
Package: Support_Kill_Assessment
With this service the subsystem provides of kill assessment information to the CMS. The information relates to an above water engagement primarily against an air target.

The kill assessment report of the subsystem may be one of the three:

- PROBABLE-KILL. This indicates that the subsystem assumes the target to be killed.
- PROBABLE-MISS. This indicates that the subsystem assumes the target to be missed by the used weapon system.
- NO-RESULT. This indicates that the subsystem was not able to determine a valid result for this request.

See also service (interface) "Process Target Designation".
Pre-condition: Service "Process Target Designation" successfully carried out.:
Pre-condition: CMS must have Mastership.:

Table 7.374 - Methods of Interface Support_Kill_Assessment_CMS

Method	Notes	Parameters
report_kill_assessment_result()	Via this message, the subsystem reports the kill assessment to the CMS.	request_id_type **RequestID** kill_assessment_result_type **KillAssessmentReport**

Support_Kill_Assessment_Sub
Type: Interface
Package: Support_Kill_Assessment

Table 7.375 - Methods of Interface Support_Kill_Assessment_Sub

Method	Notes	Parameters
request_kill_assessment()	The subsystem is requested to evaluate and report a kill assessment.	request_id_type **RequestID** expected_hit_data_type **KillAssessmentData**

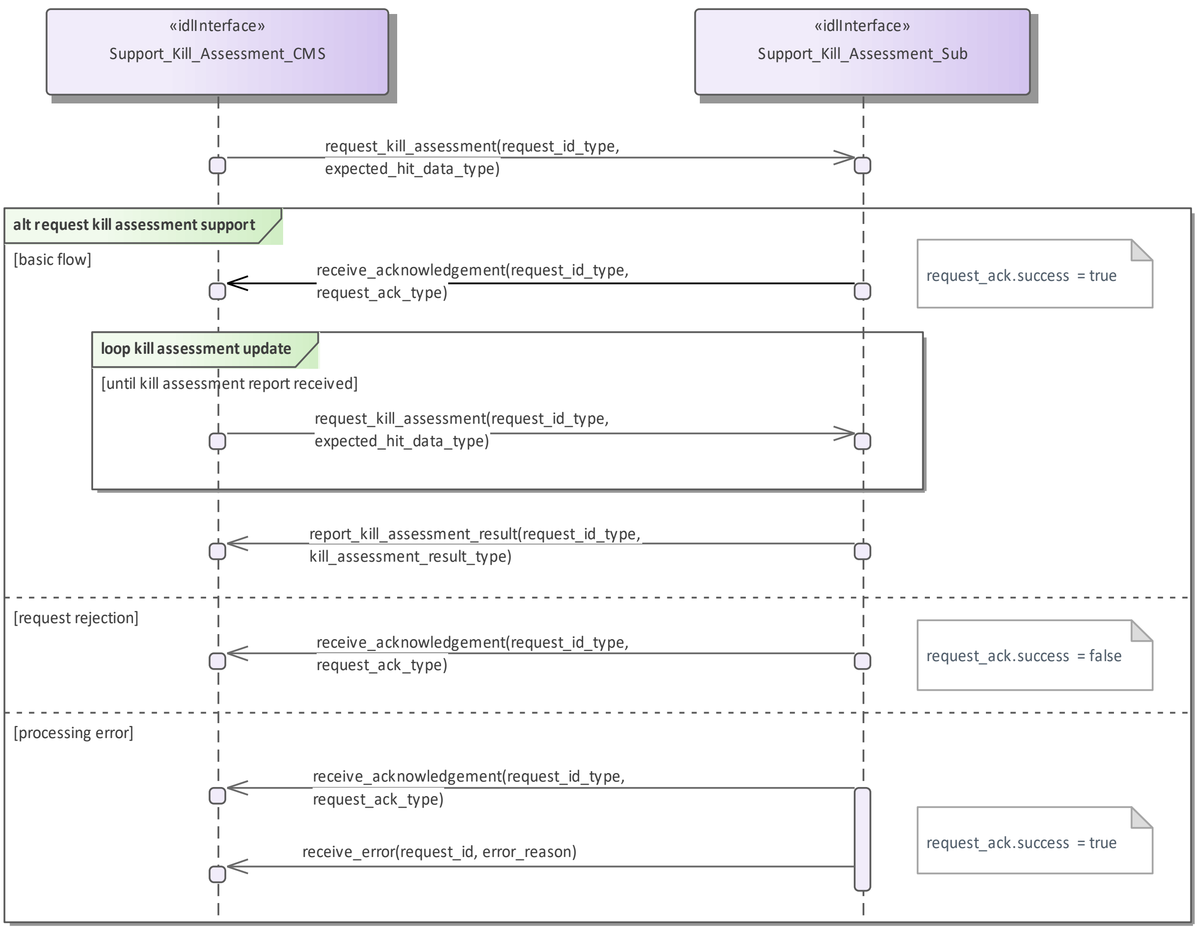


Figure 7.227 Basic Flow - Support Kill Assessment - Request Kill Assessment Support (Interaction diagram)

This sequence diagram shows how the CMS and the subsystem operate with each other during the operation "request kill assessment support " of the service "Support Kill Assessment".

7.10.2.3 Support_Surface_Target_Engagement

Parent Package: Engagement_Support

Support_Surface_Target_Engagement_CMS
Type: Interface
Package: Support_Surface_Target_Engagement
This service is intended for fire control radars, as well as surveillance radar systems that have facilities to perform surface target engagements by means of dedicated fire control channels. These fire control channels may need a differently parameterized or more elaborate track algorithm, and they may be combined with related splash spotting video.

The CMS requests the surface track to be engaged. The maximum number of tracks that may be engaged simultaneously is determined by the radar.
The functionality may also be available for land targets, provided they may be tracked by the radar.

In the case where the CMS looses or releases mastership of the subsystem, a change of the availability of fire control channels shall be indicated to the CMS. Fire control radars shall cease all fire control activities.

The set of operational modes that make fire control channels available, as well as the number of available channels shall be provided by means of service "Manage Subsystem Parameters".
Pre-condition: Technical state ONLINE.:
Pre-condition: CMS must have Mastership.:
Post-condition: Service ends with success - check availability: - the CMS is informed about the availability of fire control channels.
Post-condition: Service ends with success - target designation: - the radar provides a fire control track for the selected sensor track.
Post-condition: Service ends with success - reporting: - the CMS receives regular updates of the fire control track.
Post-condition: Service ends with success - de-designation: - the fire control channel is de-assigned and has become available.
Post-condition: Service ends with fail - target designation: - the fire control channel is not assigned; no fire control track.
Post-condition: Service ends with fail - surface track is lost: - the fire control channel is not assigned; the fire control track is terminated. The CMS is informed about the availability of fire control channel.
Post-condition: Service ends with Fail - de-designation: - the fire control channel is not assigned.

Table 7.376 - Methods of Interface Support_Surface_Target_Engagement_CMS

Method	Notes	Parameters
report_availability_state_of_fire_control_channels()	Via this interface method, the number of available fire control channels are returned from the subsystem to the CMS. If no channel is available, the value '0' is returned.	request_id_type **RequestID** available_fire_control_channels_type **AvailableFireControlChannels**
report_available_fire_control_channel()	Via this interface method, the number of available fire control channels are returned from the subsystem to the CMS.	request_id_type **RequestID** fire_control_channel_id_type **FireControlChannelID**
report_selected_fire_control_channel()	Via this interface method, the selected fire control channel is returned from the subsystem to the CMS.	request_id_type **RequestID** fire_control_channel_id_type **FireControlChannelID** sensor_track_id_type **SensorTrackId**

Support_Surface_Target_Engagement_Sub

Type: Interface
Package: Support_Surface_Target_Engagement

Table 7.377 - Methods of Interface Support_Surface_Target_Engagement_Sub

Method	Notes	Parameters
dedesignate_fire_control_channel()	Request to the subsystem to de-designate a fire control channel.	request_id_type **RequestID** fire_control_channel_id_type **FireControlChannelID**
designate_fire_control_channel()	Request to the subsystem to designate a fire control channel.	request_id_type **request_id** sensor_track_id_type **track_id**
request_availability_of_fire_control_channels()	Request to the subsystem to report the available fire control channels.	request_id_type **RequestID**

sensor track reporting
Type: InteractionOccurrence
Package: Support_Surface_Target_Engagement

Support_Surface_Target_Engagement_CMS
Type: ActivityPartition
Package: Support_Surface_Target_Engagement

Support_Surface_Target_Engagement_Sub
Type: ActivityPartition
Package: Support_Surface_Target_Engagement

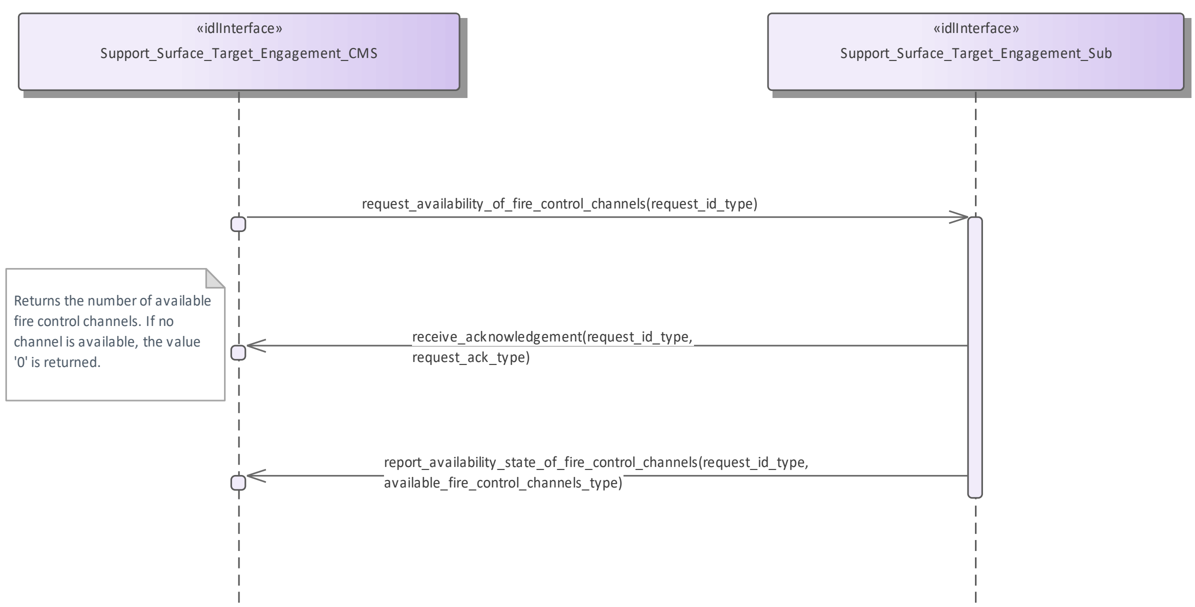


Figure 7.228 Support surface target engagement - Check availability (Interaction diagram)

This sequence diagram shows how the CMS and the subsystem operate with each other during the operation "check availability" of the service "Support surface target engagement".

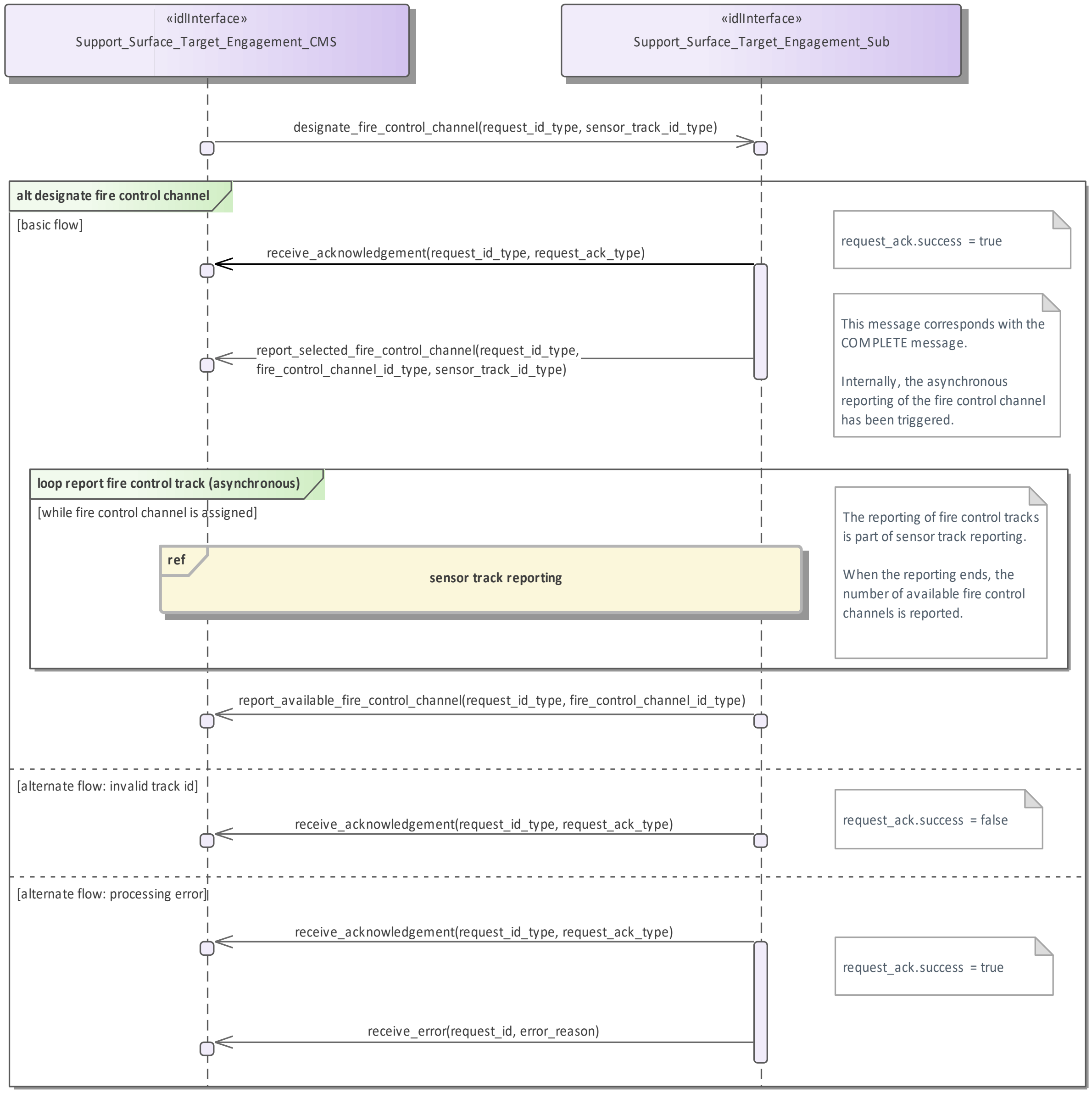


Figure 7.229 Support surface target engagement - Designate fire control channel (Interaction diagram)

This sequence diagram shows how the CMS and the subsystem operate with each other during the operation "designate fire control channel" of the service "Support surface target engagement".

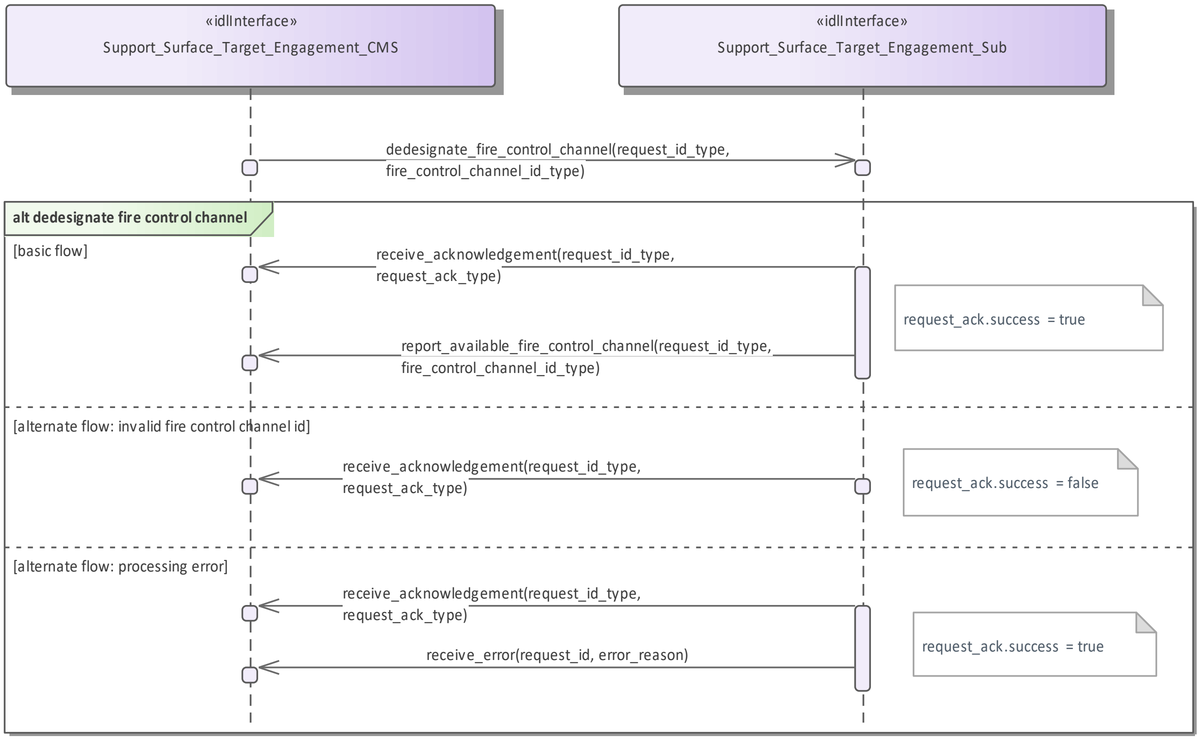


Figure 7.230 Support surface target engagement - Dedesignate fire control channel (Interaction diagram)

This sequence diagram shows how the CMS and the subsystem operate with each other during the operation "De-designate fire control channel" of the service "Support surface target engagement".

7.10.3 Missile_Guidance

Parent Package: Radar_Services

7.10.3.1 Perform_Illumination

Parent Package: Missile_Guidance

Perform_Illumination_CMS
Type: Interface
Package: Perform_Illumination
This service covers the control of target illumination to support a semi-active homing missile engagement.

The actor is the Combat Management System.

The service is triggered by the illumination request of the actor. Typically, illumination takes place during a specific period within the engagement sequence.
The actor sends an illumination request to the radar.
On the requested start time, the radar starts illuminating the target with specified parameters.
During the illumination, the actor may provide updates of illumination parameters, e.g. to change the stop time.
The service ends at stop time of the illumination.

If the radar may not fulfil the illumination request, this is reported to the actor and the service stops.

If during the illumination a radar fault takes place that prevents execution of illumination (e.g. illumination frequency not more available), the health state of the Missile Guidance service (of which this service is part) becomes DEGRADED (if the Missile Guidance service is still capable of performing uplinks and/or downlinks) or NOT AVAILABLE, and the service stops.

If the target track becomes lost during the illumination, the service stops.

Pre-condition: Sensor health state: The sensor and the Missile Guidance service are in the health state AVAILABLE or DEGRADED.

Pre-condition: Sensor parameters: The relevant sensor parameters (e.g. allowed frequencies, transmission sectors) are set[1].

----

[1] The manner in which this is done is described in other services of the OARIS ("Manage frequency usage", "Manage transmission sectors", "Control emissions" and "Manage subsystem parameters").

Table 7.378 - Methods of Interface Perform_Illumination_CMS

Method	Notes	Parameters
report_illumination_completed()		request_id_type **request_id**

Perform_Illumination_Sub

Type: Interface

Package: Perform_Illumination

Table 7.379 - Methods of Interface Perform_Illumination_Sub

Method	Notes	Parameters
request_illumination()		request_id_type **request_id** illumination_request_type **request**
provide_track()		system_track_type **track**

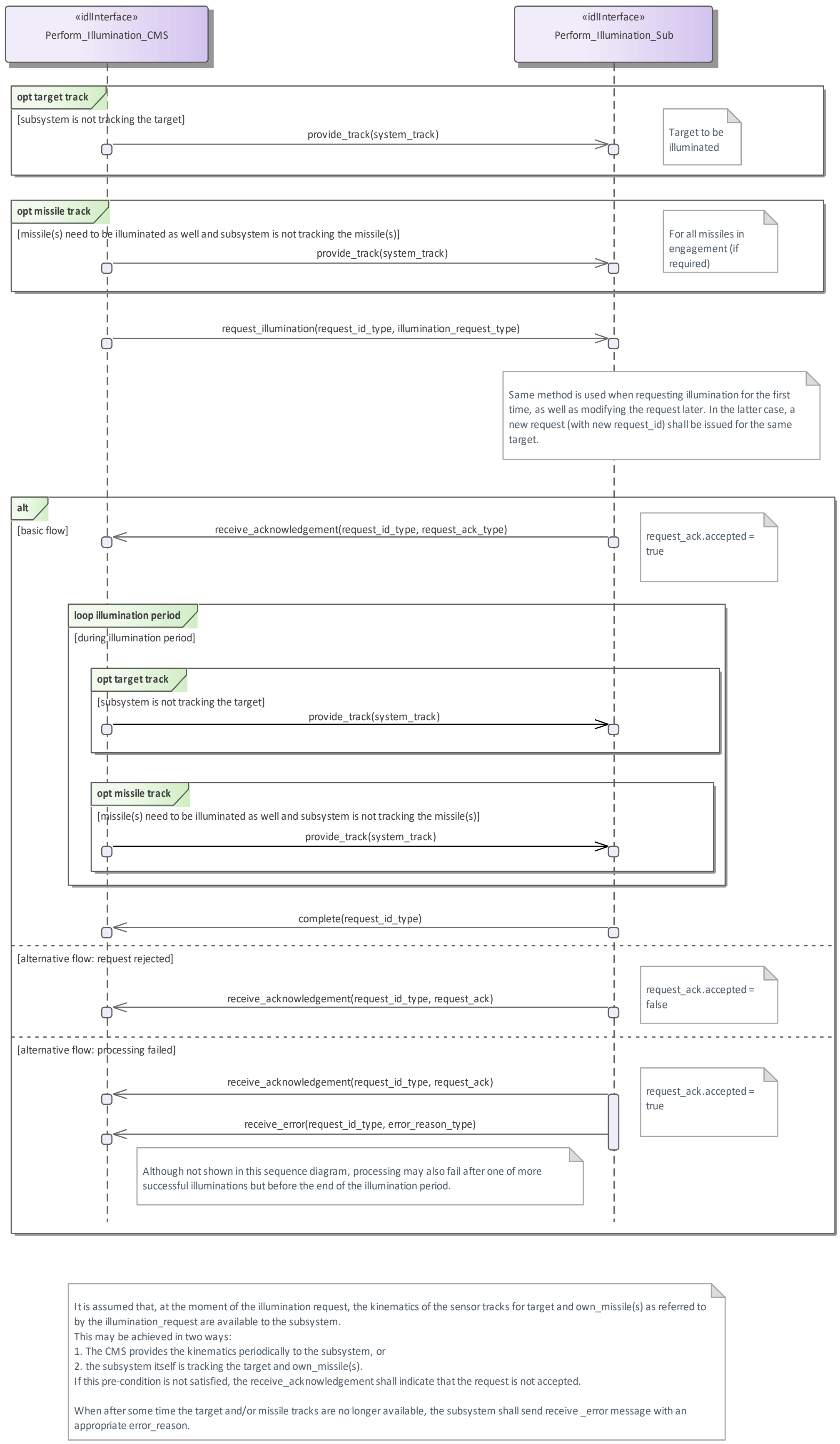

«idlInterface»
Perform_Illumination_CMS
«idlInterface»
Perform_Illumination_Sub
opt target track
[subsystem is not tracking the target]
provide_track(system_track)
Target to be illuminated
opt missile track
[missile(s) need to be illuminated as well and subsystem is not tracking the missile(s)]
provide_track(system_track)
For all missiles in engagement (if required)
request_illumination(request_id_type, illumination_request_type)
Same method is used when requesting illumination for the first time, as well as modifying the request later. In the latter case, a new request (with new request_id) shall be issued for the same target.
alt
[basic flow]
receive_acknowledgement(request_id_type, request_ack_type)
request_ack.accepted = true
loop illumination period
[during illumination period]
opt target track
[subsystem is not tracking the target]
provide_track(system_track)
opt missile track
[missile(s) need to be illuminated as well and subsystem is not tracking the missile(s)]
provide_track(system_track)
complete(request_id_type)
[alternative flow: request rejected]
receive_acknowledgement(request_id_type, request_ack)
request_ack.accepted = false
[alternative flow: processing failed]
receive_acknowledgement(request_id_type, request_ack)
request_ack.accepted = true
receive_error(request_id_type, error_reason_type)
Although not shown in this sequence diagram, processing may also fail after one of more successful illuminations but before the end of the illumination period.
It is assumed that, at the moment of the illumination request, the kinematics of the sensor tracks for target and own_missile(s) as referred to by the illumination_request are available to the subsystem.
This may be achieved in two ways:
1. The CMS provides the kinematics periodically to the subsystem, or
2. the subsystem itself is tracking the target and own_missile(s).
If this pre-condition is not satisfied, the receive_acknowledgement shall indicate that the request is not accepted.
When after some time the target and/or missile tracks are no longer available, the subsystem shall send receive _error message with an appropriate error_reason.

Figure 7.231 Basic Flow - Illumination (Interaction diagram)

7.10.3.2 Perform_Missile_Downlink

Parent Package: Missile_Guidance

Perform_Missile_Downlink_CMS
Type: Interface
Package: Perform_Missile_Downlink
The service describes the reception and provision of missile downlink information to the CMS.
Downlink consists of transmission of energy by the missile. The radar subsystem may track a missile based on these downlink transmissions (beacon track). Provision of the beacon track of the missile to the CMS is covered by service Provide sensor tracks.
This service handles the situation where the downlink also has content.
Generally, a sequence of downlinks is transmitted by the missile, on periodic basis or triggered by an uplink. However, the CMS (or a dedicated missile subsystem) is responsible for evaluating the downlinks in this sequence. The radar subsystem only receives downlinks and provides them to the CMS, and does not keep track of the sequence. In the special case where the downlink contains own missile kinematics, this data may also be used internally by the radar subsystem.

The actor is the Combat Management System.

Although the downlink may be evaluated by a missile subsystem (which is not part of the CMS), the downlink is assumed to be passed to that missile subsystem via the CMS.

The service is triggered by the downlink request of the actor.
The actor sends a downlink request to the radar.
During the request listening period, the radar listens to transmissions that are in accordance with the provided downlink parameters.
The radar reports to the actor the occurrence of the downlink, including the (decoded) content of the downlink.
The information provided by the missile may vary depending on the applied missile fire control principle, and lies outside the scope of the OARIS standard.
The information within the downlink may be used internally by the radar.
The service ends at the end of the listening period.

If the downlink transmission is interrupted, this is reported to the actor, and the service stops.

If during the downlink a radar fault takes place that prevents execution of the downlink, the health state of the Missile Guidance service (of which this service is part) becomes DEGRADED (if the Missile Guidance service is still capable of performing uplinks and/or illumination) or NOT AVAILABLE, and the service stops.

Relationship to missile uplink
For some missile types a downlink may be transmitted as a response to a received uplink (e.g. an acknowledge of receipt). This relationship (including the inherent timing relationship) depends heavily on the missile type and lies outside the scope of the OARIS standard.

Relationship to provide sensor tracks
If the downlink contains kinematic information about the missile, the radar subsystem may use this information internally to improve the own missile track (provided service Provide sensor tracks or service Process target designation).
It is also possible that the missile is tracked based on the fact that it transmits energy and not based on the contents of the downlink. This so-called beacon tracking is covered by service Provide sensor tracks.
Pre-condition: Sensor health state: The sensor and the Missile Guidance service are in the health state AVAILABLE or DEGRADED.

Pre-condition: Sensor parameters: The relevant sensor parameters (e.g. allowed frequencies, transmission sectors) are set. (The manner in which this is done is described in other services of the specification: see “Manage frequency usage”, “Manage transmission sectors”, “Control emissions” and “Manage subsystem parameters”).

Pre-condition: Engagement phase: An engagement must be taking place.

Pre-condition: Missile downlink parameters: The parameters of the missile downlink transmission must be known to the radar. Note that this does not concern the content of the transmission, but rather the transmission characteristics (e.g. frequency).

Table 7.380 - Methods of Interface Perform_Missile_Downlink_CMS

Method	Notes	Parameters
report_downlink()		request_id_type **request_id** downlink_report_type **the_downlink_info**
report_downlink_completed()		request_id_type **request_id**

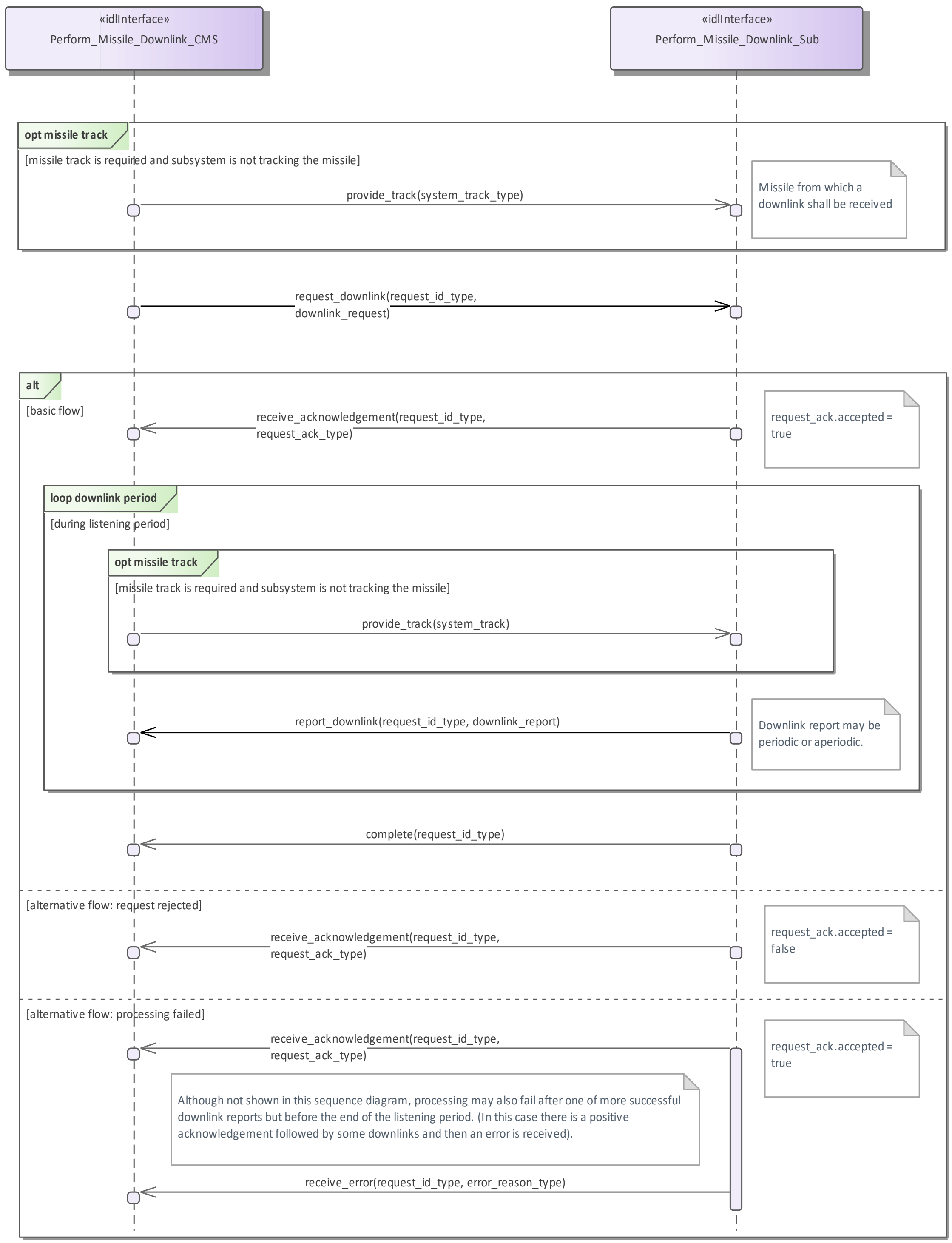
Perform_Missile_Downlink_Sub

Type: Interface

Package: Perform_Missile_Downlink

Table 7.381 - Methods of Interface Perform_Missile_Downlink_Sub

Method	Notes	Parameters
request_downlink()		request_id_type **request_id** downlink_request_type **request**
provide_track()		system_track_type **track**



The request_downlink operation has not been identified in the service Description.
The reasons for introducing it here are:
1. There are no provisions (e.g. services) to satisfy the missile downlink parameters precondition.
2. The CMS is only interested in downlink information from own missiles in flight belonging to an active engagement.
3. Generally, the missile downlink parameters (e.g. frequency) are engagement dependent.

Figure 7.232 Basic Flow - Downlink (Interaction diagram)

7.10.3.3 Perform_Missile_Uplink

Parent Package: Missile_Guidance

Perform_Missile_Uplink_CMS
Type: Interface
Package: Perform_Missile_Uplink
The service describes the execution of uplink of relevant information from the radar to the missile in flight during an engagement.
Generally, a sequence of uplinks (of various types) must be transmitted to a missile during an engagement. However, the CMS (or a dedicated missile subsystem) is responsible for planning and requesting the correct sequence of uplinks. The radar subsystem only transmits an uplink on request of the CMS. Therefore, this service starts with the request of a single uplink and ends when the radar subsystem has transmitted the uplink.

The actor is the Combat Management System. Although the uplink may be initiated by a missile subsystem (which is not part of the CMS), the uplink is assumed to be passed through the CMS to the radar subsystem.

The service is triggered by the uplink request of the actor.
The actor sends an uplink request to the radar.
At the requested time, the radar sends the uplink to the missile in accordance with the provided uplink parameters.
The information provided to the missile may vary depending on the applied missile fire control principle, and lies outside the scope of the OARIS standard.
The service ends when the radar has confirmed the transmission of the uplink.

If the radar may not fulfil the uplink request, this is reported to the actor and the service stops.

If during the uplink a radar fault takes place that prevents execution of the uplink (e.g. uplink frequency not more available), the health state of the Missile Guidance service (of which this service is part) becomes DEGRADED (if the Missile Guidance service is still capable of performing illumination and/or downlinks) or NOT AVAILABLE, and the service stops.

If the missile track becomes lost during the uplink, the service stops.

Network Centric engagements
In Network-Centric or Network-Enabled systems, guidance of the missile may be transferred during the flight of the missile to another surface platform. As the related technologies are still being developed, it shall be too early to include specific NEC requirements here. However, care should be taken in the design of OARIS that such capabilities could be included at a later date. This means that there should be no built-in restrictions in the standard, which would prevent addition of such facilities in the future.

Relationship to missile downlink
For some missile types an uplink transmission may trigger the transmission of a downlink by the missile (e.g. an acknowledge of receipt). This relation depends heavily on the missile type and lies outside the scope of the OARIS standard.
Pre-condition: Sensor health state: The sensor and the Missile Guidance service are in the health state AVAILABLE or DEGRADED.
Pre-condition: Sensor parameters: The relevant sensor parameters (e.g. allowed frequencies, transmission sectors) are set[1].

----

[1] The manner in which this is done is described in other services of the OARIS (“Manage frequency usage”, “Manage transmission sectors”, “Control emissions” and “Manage subsystem parameters”).

Pre-condition: Engagement phase: An engagement must be taking place.

Pre-condition: Known position of missile: The position of the missile must be known, i.e. own missile track must exist. The missile track may be provided by the CMS or by the radar subsystem itself.

Table 7.382 - Methods of Interface Perform_Missile_Uplink_CMS

Method	Notes	Parameters
report_uplink_completed()		request_id_type **request_id** uplink_report_type **report**

Perform_Missile_Uplink_Sub
Type: Interface
Package: Perform_Missile_Uplink

Table 7.383 - Methods of Interface Perform_Missile_Uplink_Sub

Method	Notes	Parameters
request_uplink()		request_id_type **request_id** uplink_request_type **request**
provide_track()		system_track_type **track**

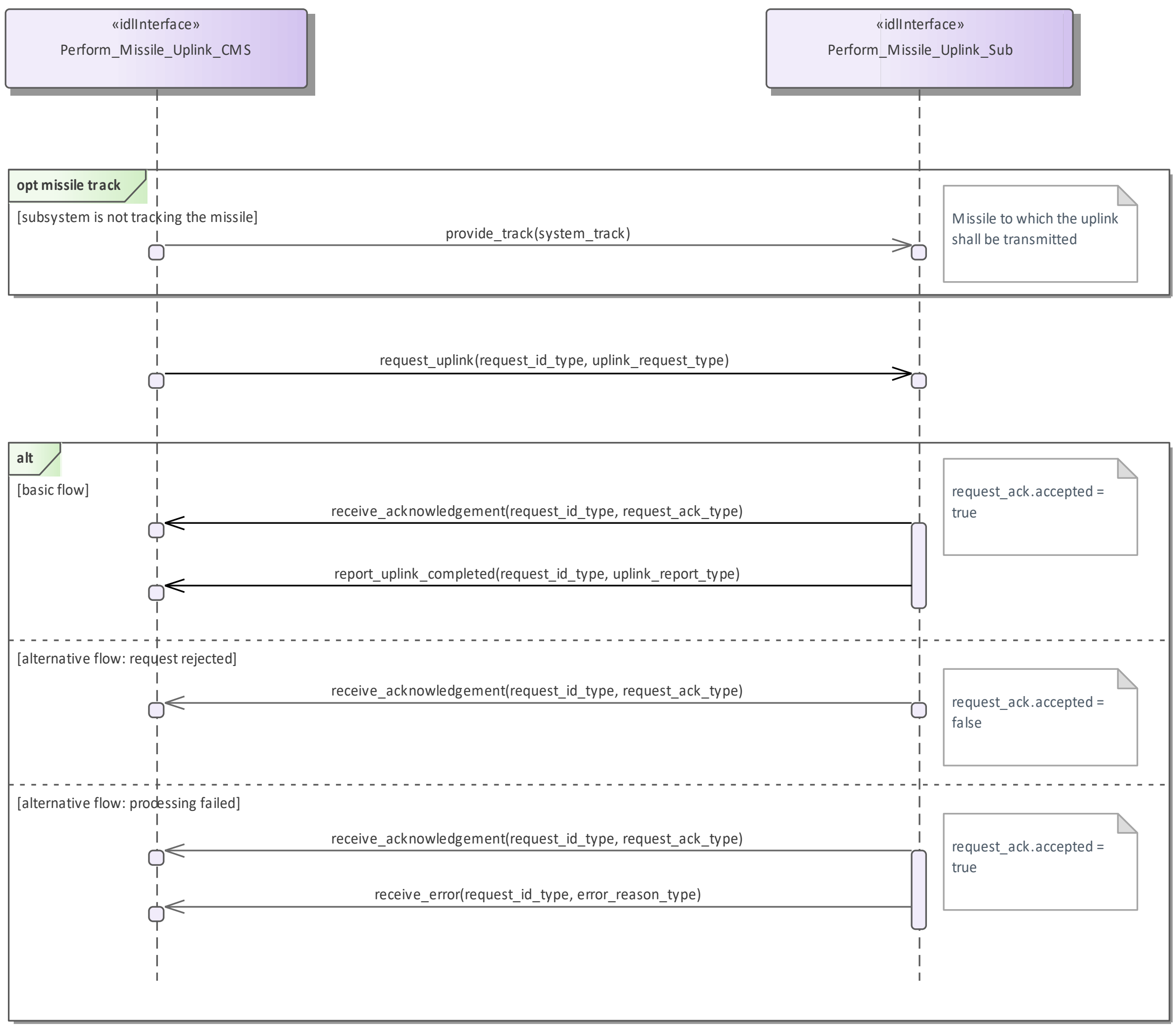


Figure 7.233 Basic Flow - Uplink (Interaction diagram)

7.10.4 Surface_Engagement_Support

Parent Package: Radar_Services

7.10.4.1 Perform_Splash_Spotting

Parent Package: Surface_Engagement_Support

Perform_Splash_Spotting_CMS
Type: Interface
Package: Perform_Splash_Spotting
Surveillance radar systems may support engagements against surface targets by means of a splash spotting video or measured splash positions. In the vicinity of the target a signal processing is applied which is optimized to observe splashes of the shells hitting the sea surface.

The splash spotting information may be used to achieve shot corrections for a running engagement. The engagement may use a fire control channel of the radar but also of another device like fire control radar. The CMS requests the radar to localize a splash spotting area at a defined position derived from the target kinematics.

The use of splash spotting areas may be limited to fire control channels of the radar. Then, only the localization of a splash spotting area may be done in accordance with this service. Normally, it shall be localized at the predicted hitting point.

These splash spotting areas shall not differ in terms of function and performance so that the selection of the area to be applied to an engagement may be done by the radar, automatically. The CMS just indicates where to localize it.

If mastership is lost during execution in any of the flows the services are terminated.
Pre-condition: Technical state ONLINE.:
Pre-condition: Assigned fire control channel.: - a fire control channel has been assigned using "Support Surface Target Engagement"
Pre-condition: CMS must have Mastership:
Post-condition: Success: The subsystem provides splash spotting videos as long as the splash spotting areas are active.:
Post-condition: No success: The subsystem does not perform as requested.:

Table 7.384 - Methods of Interface Perform_Splash_Spotting_CMS

Method	Notes	Parameters
confirm_reposition_splash_~~splotting~~ spotting_area()	Via this method, the request for the repositioning of a splash spotting area is confirmed by the subsystem.	request_id_type **RequestID** splash_spotting_area_id_type **SplashSpottingAreaID**
confirm_splash_spotting_area_deactivation()	Via this method, the request for the deactivation of a splash spotting area is confirmed by the subsystem.	request_id_type **RequestID** splash_spotting_area_id_type **SplashSpottingAreaId**
receive_splash_~~splotting~~spotting_area_position()	Via this method, the request for a new splash spotting area based on a position is confirmed by the subsystem.	request_id_type **RequestID** splash_spotting_area_id_type **SplashSpottingAreaID**
receive_splash_~~splotting~~spotting_are	Via this method, the request for a	request_id_type **RequestID**

a_track()	new splash spotting area based on a track is confirmed by the subsystem.	splash_spotting_area_id_type **SplashSpottingAreaID**
report_splash_spotting_area_activation_state()	Via this interface, the splash spotting areas are reported to the CMS.	request_id_type **RequestID** splash_spotting_area_set_type **SplashSpottingAreaSet**

Perform_Splash_Spotting_Sub
Type: Interface
Package: Perform_Splash_Spotting

Table 7.385 - Methods of Interface Perform_Splash_Spotting_Sub

Method	Notes	Parameters
activate_splash_spotting_area_by_position()	Requests the subsystem to activate a new splash spotting area based on a area/position.	request_id_type **RequestID** splash_spotting_area_position_type **SplashSpottingAreaPosition**
activate_splash_spotting_area_by_track()	Requests the subsystem to activate a new splash spotting area based on a sensor track.	request_id_type **RequestID** sensor_track_id_type **TrackID**
deactivate_splash_spotting_area()	Requests the subsystem to de-activate a splash spotting area.	request_id_type **RequestID** splash_spotting_area_id_type **SplashSpottingAreaID**
report_splash_spotting_information()	Requests the subsystem to report splash spotting information/splash positions for an existing splash spotting area.	request_id_type **RequestID** splash_spotting_area_id_type **SplashSpottingAreaID**
reposition_splash_spotting_area()	Requests the subsystem to reposition a existing splash spotting area.	request_id_type **RequestID** splash_spotting_area_id_type **SplashSpottingAreaID** splash_spotting_area_position_type **SplashSpottingAreaPosition**
request_splash_spotting_areas()	Request the subsystem to report the splash spotting areas to the CMS.	request_id_type **RequestID**

Perform_Splash_Spotting_CMS
Type: ActivityPartition
Package: Perform_Splash_Spotting

Perform_Splash_Spotting_Sub
Type: ActivityPartition
Package: Perform_Splash_Spotting

Report measured splash positions
Type: InteractionOccurrence
Package: Perform_Splash_Spotting

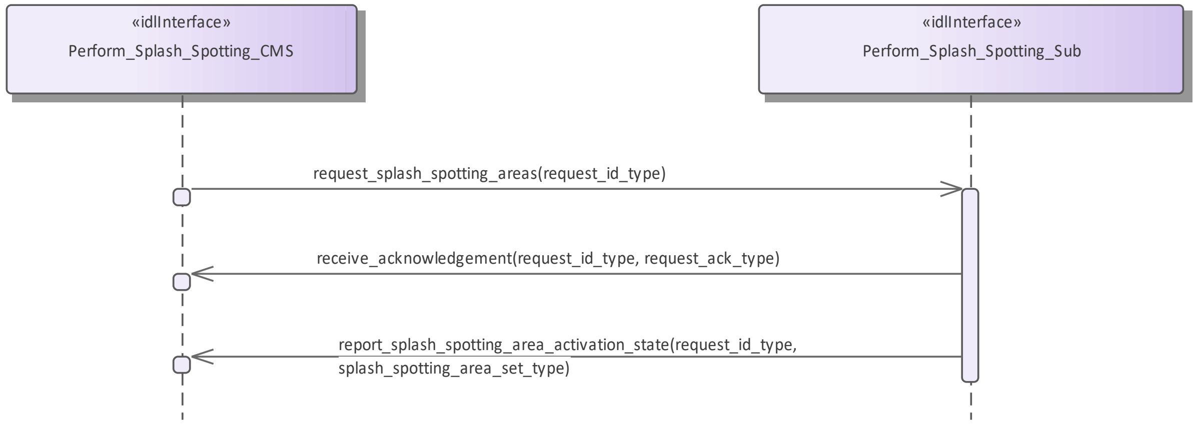


Figure 7.234 Perform Splash Spotting - Check Activation (Interaction diagram)

This sequence diagram shows how the CMS and the subsystem operate with each other during the operation "check activation" of the service "Perform splash spotting".

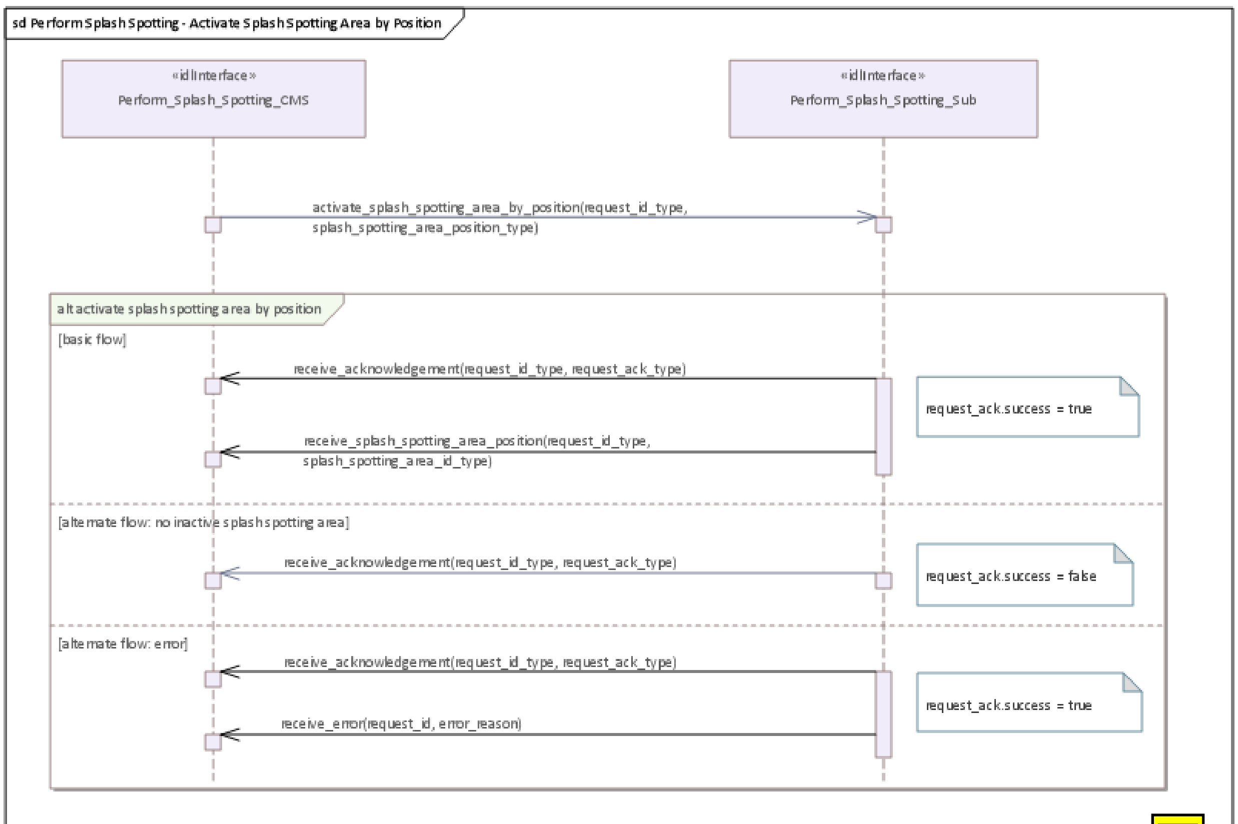


Figure 7.235 Perform Splash Spotting - Activate Splash Spotting Area by Position (Interaction diagram)

This sequence diagram shows how the CMS and the subsystem operate with each other during the operation "activate splash spotting area by position" of the service "Perform Splash Spotting".

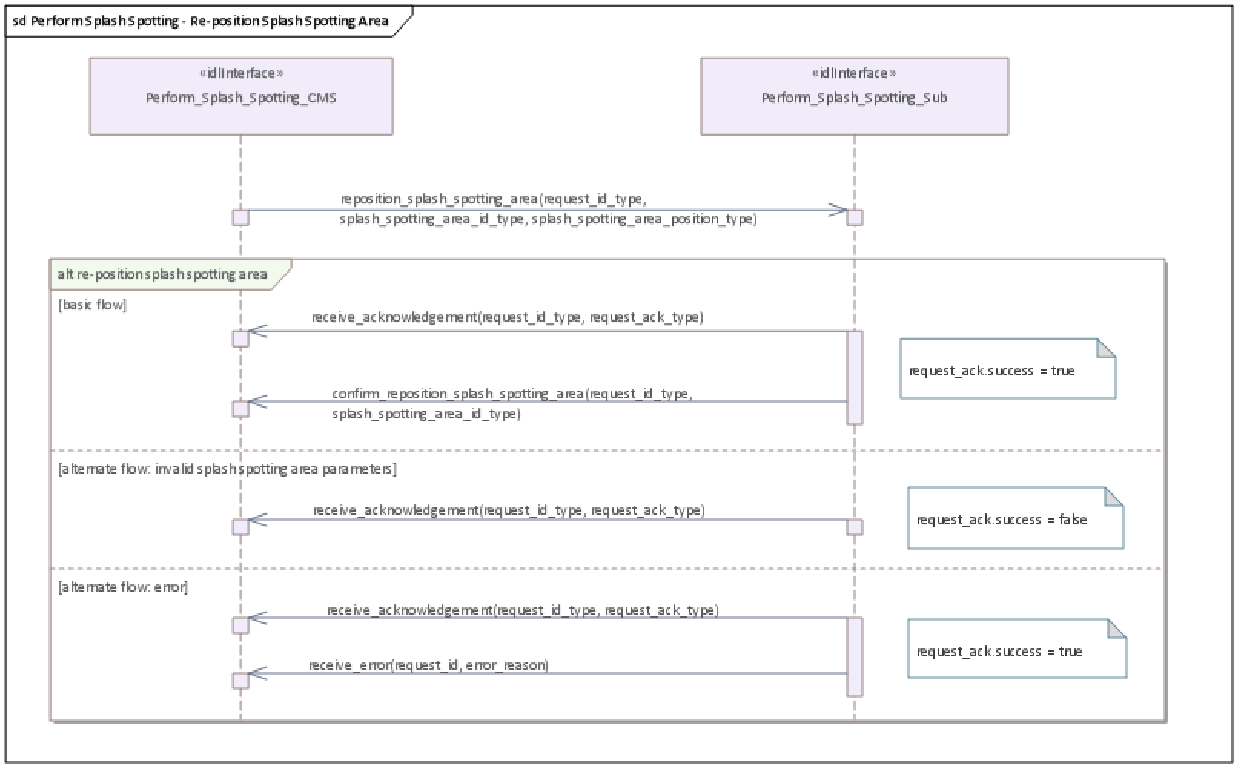


Figure 7.236 Perform Splash Spotting - Re-position Splash Spotting Area (Interaction diagram)

This sequence diagram shows how the CMS and the subsystem operate with each other during the operation "reposition splash spotting area" of the service "Perform splash spotting".

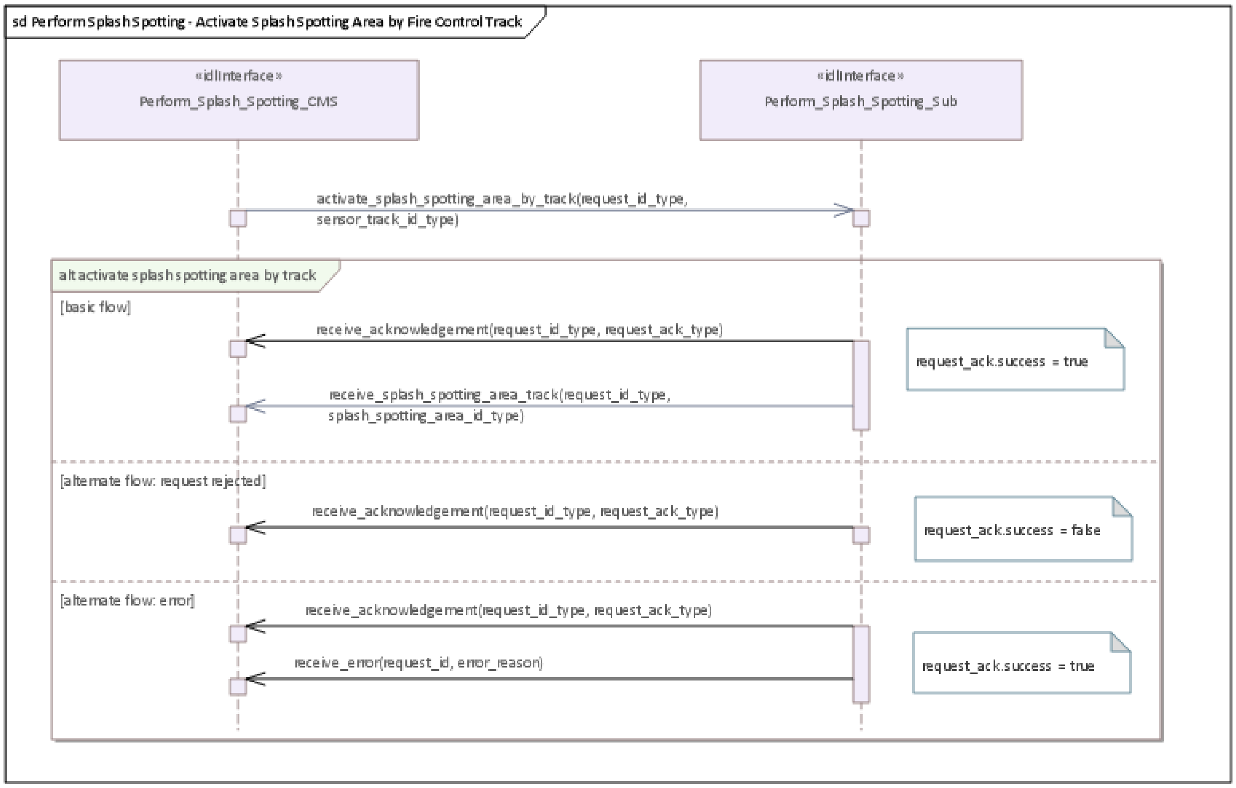


Figure 7.237 Perform Splash Spotting - Activate Splash Spotting Area by Fire Control Track (Interaction diagram)

This sequence diagram shows how the CMS and the subsystem operate with each other during the operation "activate splash spotting area by fire control track" of the service "Perform splash spotting".

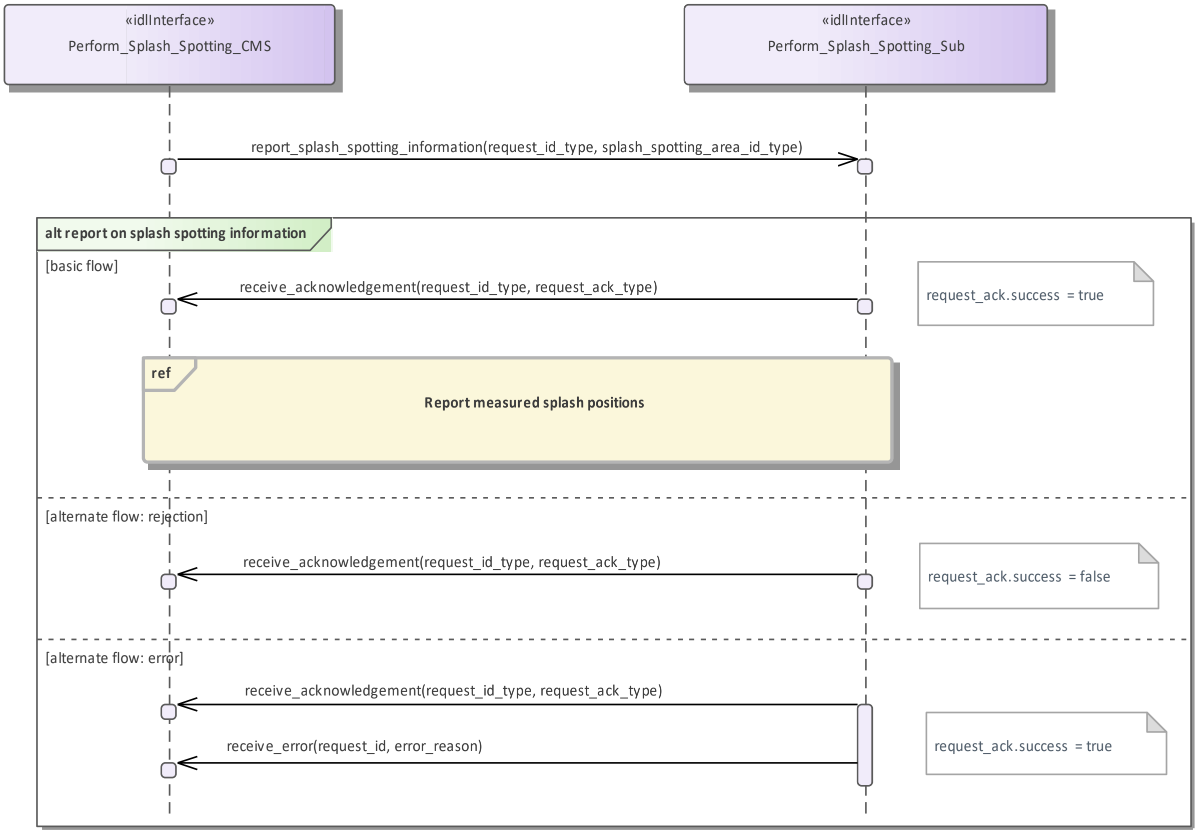


Figure 7.238 Perform Splash Spotting - Report On Splash ~~Splotting~~Spotting Information (Interaction diagram)

This sequence diagram shows how the CMS and the subsystem operate with each other during the operation "report on splash spotting information" of the service "Perform splash spotting".

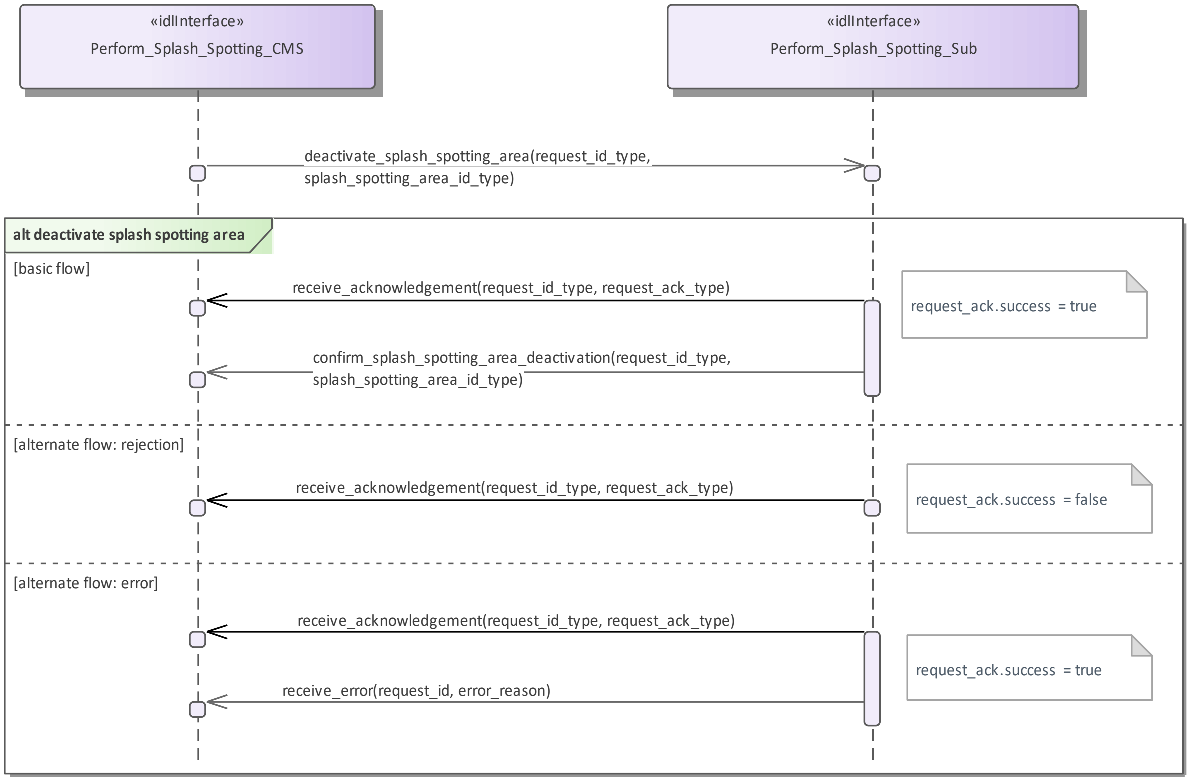


Figure 7.239 Perform Splash Spotting - Deactivate Splash Spotting Area (Interaction diagram)

This sequence diagram shows how the CMS and the subsystem operate with each other during the operation "deactivate splash spotting area" of the service "Perform splash spotting".

8 Platform-Specific Models

8.1 DDS Data Model PSM

The DDS Data Model PSM defines a set of IDL files for the Data Model packages defined by the PIM. Comments are added to the IDL files to reflect the mapping rules below.

The detailed rules for the MDA code generation from the Data Model PIM to the DDS PSM IDL are as follows:

- PIM attributes and compositions are mapped to IDL attributes;
  - Optional PIM attributes are mapped to a union type with a single member present when the exists case attribute is true;
- Collections in the PIM are mapped to IDL sequences; a Length tag determines the sequence bounds;
  - Specialization / Generalization PIM relationships are mapped to IDL unions. Additional data classes are introduced for non-abstract generalization classes that have attributes

8.2 DDS Services PSM

The DDS Services PSM defines IDL files for each package defined in the Services PIM. For each method on each interface class an IDL struct for a DDS topic named for the method is generated; each parameter is mapped to an attribute of the IDL struct. Note that the PIM only defines parameters with an 'in' mode, there are no 'return' parameters defined and all methods have at least one parameter. Comments are generated to match the PIM notes and to include the version number of this standard in each file.

Additionally the struct contains a subsystem_id key attribute of type subsystem_id_type. This indicates which subsystem published the data or is intended to read it as a subscriber.

Operations that require a response contain a request_id in the PIM that logically links request and response instances. In the DDS PSM, each request_id operation parameter is mapped to a keyed attribute of the DDS topic so that distinct request and response pairs can be retrieved from the DDS data space.

To robustly and efficiently ensure that the data exchanged between a particular subsystem and a CMS is recognized correctly a system integrator should mandate a strategy to ensure that subsystems and topic instances can not clash. Two such strategies are:

- separating subsystems by topic (for example using "write_sensor_track__RADAR1" as a topic name for all sensor track samples being provided by the RADAR1 subsystem).
- separating subsystems by partition (for example using "write_sensor_track" on partition "RADAR1" for all sensor track samples being provided by the RADAR1 subsystem).

Also, the CMS uses the receive_cms_identification_data topic to allocate a subsystem_id to a subsystem; the subsystem sets the subsystem_id to zero for the receive_subsystem_identification_data topic, for which the CMS subscribes on the wildcard partition "*". Subsequently, for data intended for all subsystems, the CMS publishes samples on partition "*" with a subsystem id of zero.

However, the Register Interest use case is mapped to the DDS DCPS Reader Listener interface and the Provide Subsystem Services use case is mapped to the DDS DCPS Data Reader and Data Writer interfaces, so there are no IDL files for these use cases.

8.3 GraphQL Data Model and Services PSM

The GraphQL PSM defines a set of schema definition language files, one for each Service interface defined by the PIM; each of these files represents a self-contained service and contains definition for the types represented. Comments are added to these files to reflect the mapping rules below.

The detailed rules for the MDA code generation from the Data Model PIM to the GraphQL are as follows:

- Enumerations are mapped to GraphQL enums;
- PIM Classes with an 'idlStruct' stereotype are mapped to both a GraphQL object and input type;
    - Scalar idlTypedef stereotyped classes are inlined to primitive GraphQL types in the types that use them;
- PIM attributes and compositions are mapped to GraphQL object and input attributes;
- Non-optional PIM attributes are mandatory GraphQL attributes;
- Collections in the PIM are mapped to GraphQL lists (which are unbounded);
    - Specialization / Generalization PIM relationships are mapped to GraphQL union and interface object types and an input type with optional attributes (and the same semantics). Additional data classes are introduced for non-abstract generalization classes that have attributes

The GraphQL services derived from CMS interfaces allow the CMS to query and subscribe to operations invoked by a Subsystem, whilst Subsystem can invoke the interface by making mutations. Services derived from Subsystem interfaces allow the Subsystem to query and subscribe and the CMS to mutate. Each GraphQL service contains:

- A schema object declaring query, mutation and subscription attributes;
    - Query (also used for subscription) with an argument list allowing filtering by subsystem and whether simulated
- Mutation object types each returning lists of operations;
- A union type with choices for each operation on the interface;
- A options input type with optional attributes for each operation on the interface for mutations;
    - An object type for each operation including a argument list containing each key in the operation types (as well as request_id if present) and an additional list of subsystem names returning a list of operations;
- An input type for each operation including an additional list of subsystem names;
    - Sensor Assessment, Supplementary Measurement and Track Reporting operations also support additional arguments to filter by environment and area

1 PIM Classes with an 'idlStruct' stereotype are mapped to both a GraphQL object and input type;

2 Scalar idlTypedef stereotyped classes are inlined to primitive GraphQL types in the types that use them;

3 PIM attributes and compositions are mapped to GraphQL object and input attributes;

4 Non-optional PIM attributes are mandatory GraphQL attributes;

5 Collections in the PIM are mapped to GraphQL lists (which are unbounded);

6 Specialization / Generalization PIM relationships are mapped to GraphQL union and interface object types and an input type with optional attributes (and the same semantics). Additional data classes are introduced for non-abstract generalization classes that have attributes

The GraphQL services derived from CMS interfaces allow the CMS to query and subscribe to operations invoked by a Subsystem, whilst Subsystem can invoke the interface by making mutations. Services derived from Subsystem interfaces allow the Subsystem to query and subscribe and the CMS to mutate. Each GraphQL service contains:

7 A schema object declaring query, mutation and subscription attributes;

8 Query (also used for subscription) with an argument list allowing filtering by subsystem and whether simulated

9 Mutation object types each returning lists of operations;

10 A union type with choices for each operation on the interface;

11 A options input type with optional attributes for each operation on the interface for mutations;

12 An object type for each operation including a argument list containing each key in the operation types (as well as request_id if present) and an additional list of subsystem names returning a list of operations;

13 An input type for each operation including an additional list of subsystem names;

14 Sensor Assessment, Supplementary Measurement and Track Reporting operations also support additional arguments to filter by environment and area