Date: Mar 2019

Open Architecture Radar Interface Standard

RTF - 1.1

OMG Document Number: dtc/19-03-04

Standard document URL: http://www.omg.org/spec/OARIS/1.1

Machine Consumable File(s):

http://www.omg.org/spec/OARIS/20190201/dtc/19-03-05.xml

http://www.omg.org/spec/OARIS/20190201/dtc/19-02-05.zip

http://www.omg.org/spec/OARIS/20190201/dtc/19-03-06.eap

USE OF SPECIFICATION - TERMS, CONDITIONS & NOTICES

The material in this document details an Object Management Group specification in accordance with the terms, conditions and notices set forth below. This document does not represent a commitment to implement any portion of this specification in any company's products. The information contained in this document is subject to change without notice.

LICENSES

The company listed above have granted to the Object Management Group, Inc. (OMG) a nonexclusive, royalty-free, paid up, worldwide license to copy and distribute this document and to modify this document and distribute copies of the modified version. Each of the copyright holders listed above has agreed that no person shall be deemed to have infringed the copyright in the included material of any such copyright holder by reason of having used the specification set forth herein or having conformed any computer software to the specification.
Subject to all of the terms and conditions below, the owners of the copyright in this specification hereby grant you a fully-paid up, non-exclusive, nontransferable, perpetual, worldwide license (without the right to sublicense), to use this specification to create and distribute software and special purpose specifications that are based upon this specification, and to use, copy, and distribute this specification as provided under the Copyright Act; provided that: (1) both the copyright notice identified above and this permission notice appear on any copies of this specification; (2) the use of the specifications is for informational purposes and will not be copied or posted on any network computer or broadcast in any media and will not be otherwise resold or transferred for commercial purposes; and (3) no modifications are made to this specification. This limited permission automatically terminates without notice if you breach any of these terms or conditions. Upon termination, you will destroy immediately any copies of the specifications in your possession or control.

PATENTS

The attention of adopters is directed to the possibility that compliance with or adoption of OMG specifications may require use of an invention covered by patent rights. OMG shall not be responsible for identifying patents for which a license may be required by any OMG specification, or for conducting legal inquiries into the legal validity or scope of those patents that are brought to its attention. OMG specifications are prospective and advisory only. Prospective users are responsible for protecting themselves against liability for infringement of patents.

GENERAL USE RESTRICTIONS

Any unauthorized use of this specification may violate copyright laws, trademark laws, and communications regulations and statutes. This document contains information, which is protected by copyright. All Rights Reserved. No part of this work covered by copyright herein may be reproduced or used in any form or by any means--graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems--without permission of the copyright owner.

DISCLAIMER OF WARRANTY

WHILE THIS PUBLICATION IS BELIEVED TO BE ACCURATE, IT IS PROVIDED "AS IS" AND MAY CONTAIN ERRORS OR MISPRINTS. THE OBJECT MANAGEMENT GROUP AND THE COMPANIES LISTED ABOVE MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS PUBLICATION, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OF TITLE OR OWNERSHIP, IMPLIED WARRANTY OF MERCHANTABILITY OR WARRANTY OF FITNESS FOR A PARTICULAR PURPOSE OR USE. IN NO EVENT SHALL THE OBJECT MANAGEMENT GROUP OR ANY OF THE COMPANIES LISTED ABOVE BE LIABLE FOR ERRORS CONTAINED HEREIN OR FOR DIRECT, INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL, RELIANCE OR COVER DAMAGES, INCLUDING LOSS OF PROFITS, REVENUE, DATA OR USE, INCURRED BY ANY USER OR ANY THIRD PARTY IN CONNECTION WITH THE FURNISHING, PERFORMANCE, OR USE OF THIS MATERIAL, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

The entire risk as to the quality and performance of software developed using this specification is borne by you. This disclaimer of warranty constitutes an essential part of the license granted to you to use this specification.

RESTRICTED RIGHTS LEGEND

Use, duplication or disclosure by the U.S. Government is subject to the restrictions set forth in subparagraph (c) (1) (ii) of The Rights in Technical Data and Computer Software Clause at DFARS 252.227-7013 or in subparagraph (c)(1) and (2) of the Commercial Computer Software - Restricted Rights clauses at 48 C.F.R. 52.227-19 or as specified in 48 C.F.R. 227-7202-2 of the DoD F.A.R. Supplement and its successors, or as specified in 48 C.F.R. 12.212 of the Federal Acquisition Regulations and its successors, as applicable. The specification copyright owners are as indicated above and may be contacted through the Object Management Group, 109 Highland Avenue, Needham, MA 02494, U.S.A.

TRADEMARKS

IMM®, MDA®, Model Driven Architecture®, UML®, UML Cube logo®, OMG Logo®, CORBA® and XMI® are registered trademarks of the Object Management Group, Inc., and Object Management Group™, OMG™, Unified Modeling Language™, Model Driven Architecture Logo™, Model Driven Architecture Diagram™, CORBA logos™, XMI Logo™, CWM™, CWM Logo™, IIOP™, MOF™, OMG Interface Definition Language (IDL)™, and OMG SysML™ are trademarks of the Object Management Group. All other products or company names mentioned are used for identification purposes only, and may be trademarks of their respective owners.

COMPLIANCE

The copyright holders listed above acknowledge that the Object Management Group (acting itself or through its designees) is and shall at all times be the sole entity that may authorize developers, suppliers and sellers of computer software to use certification marks, trademarks or other special designations to indicate compliance with these materials.

Software developed under the terms of this license may claim compliance or conformance with this specification if and only if the software compliance is of a nature fully matching the applicable compliance points as stated in the specification. Software developed only partially matching the applicable compliance points may claim only that the software was based on this specification, but may not claim compliance or conformance with this specification. In the event that testing suites are implemented or approved by Object Management Group, Inc., software developed using this specification may claim compliance or conformance with the specification only if the software satisfactorily completes the testing suites.

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

- **CORBA/IIOP**
- **Data Distribution Services**
- **Specialized CORBA**

IDL/Language Mapping Specifications

Modeling and Metadata Specifications

- **UML, MOF, CWM, XMI**
- **UML Profile**

Modernization Specifications

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

- **CORBAServices**
- **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 primarily defines the interface between the CMS and a Radar system within a modular combat system architecture for naval platforms. However, it is structured to aligned 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 any radar component within a combat system), as illustrated below. As such it has potential to provide the basis for specifications for other combat system sensors and subsystems.

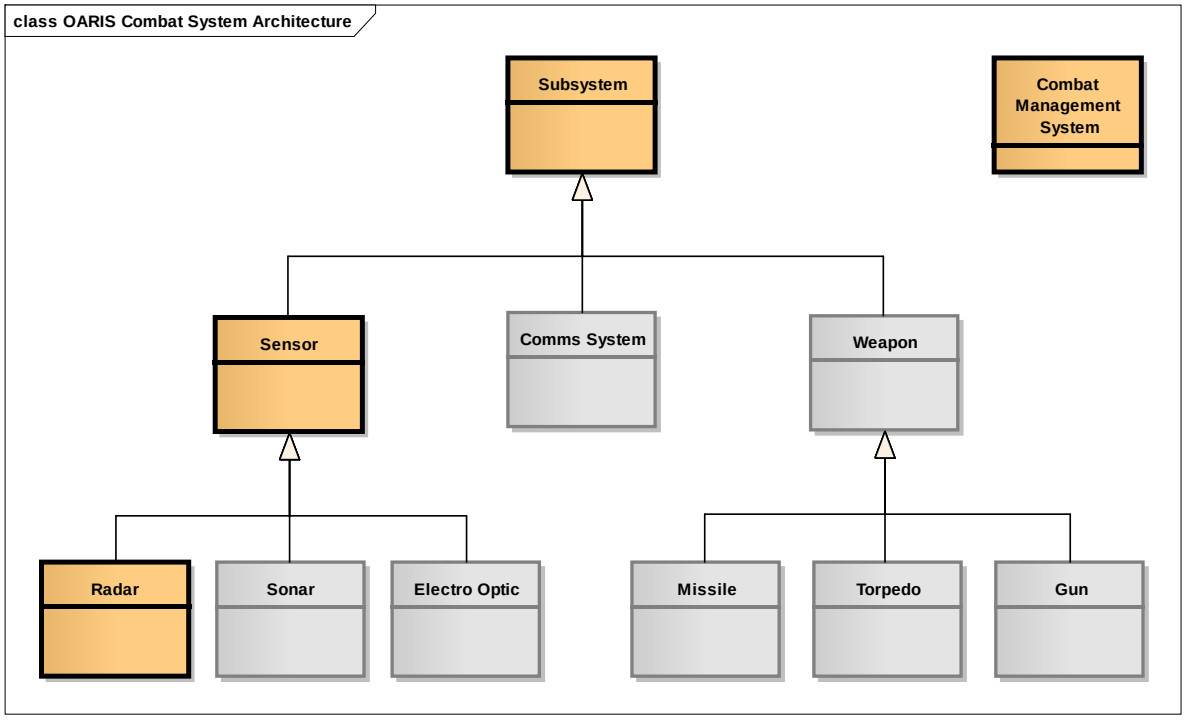

Figure 1.1 - The OARIS specification exploits specialisation and generalisation to promote modularity and extensibility

2 Conformance

In order to support utilization by a range of radars from simple navigation radars to complex multi-function radars the RFP defines the following compliance levels:

- Level 1
  The simplest radar operation providing 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
  In addition to basic operation (level 2), compliance with NNSI (Not supported in this version of the response.)
- Level 3G
  In addition to basic operation (level 2), compliance with METOC (Not supported in this version of the response.)

Radars 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 (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:

Compliance Level	Required Interfaces
1	Register Interest Track Reporting Plot Reporting
2	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
3A	Define Test Target Scenario Define Fault Scripts Control Simulation Control Fault Script Control Test Target Facility Control Recording Control Replay Provide Simulation Data
3B	Shutdown Restart Startup Manage Physical Configuration Perform Offline Test Receive Encyclopedic Data
3C	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
3D	Process Target Designation Provide Projectile Positional Information Perform Missile Downlink Perform Missile Uplink Kill Assessment Support Surface Engagement Perform Splash Plotting
3E	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

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.

- ALMAS (formal/2009-11-01)
- AMSM (formal/2010-11-02)
- CORBA (formal/2011-11-01,02,03)
- DDS (formal/2007-01-01)
- DIS (IEEE 1278.1–1995, IEEE 1278.1A–1998 and Enumeration and Bit-encoded values for use with IEEE 1278.1-1995)
- EVOT (formal/2008-08-01)
- HLA (IEEE 1516 2000-series and RPR-FOM 2.0)
- ISO 19111 (www.iso.org/)
- ISO 19115 (www.iso.org/)
- METOC RFP (c4i/08-12-02)
- NNSI RFP (c4i/07-12-01)
- Network Time Protocol (www.ntp.org)
- Precision Time Protocol (IEEE 1588 – http://www.ieee1588.com)
- SoaML (www.omg.org/spec/SoaML)

4 Terms and Definitions

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

- AB (Architecture Board)
- ALMAS (Alert Management Service)
- AMSM (Application Management and Status Monitoring)
- API (Application Programming Interface)

- ATC (Air Traffic Control)
- BC (Business Committee)
- BCQ (Business Committee Questionnaire)
- BoD (Board of Directors)
- CCM (CORBA Component Model)
- CMS (Combat Management System)
- CORBA (Common Object Request Broker Architecture)
- CSIV2 (Common Secure Interoperability Protocol Version 2)
- CWM (Common Warehouse Metamodel)
- DAIS (Data Acquisition from Industrial Systems)
- DDS (Data Distribution Service)
- EDOC (Enterprise Distributed Object Computing)
- EJB (Enterprise Java Bean)
- EVOT (Enhanced View of Time)
- FTF (Finalization Task Force)
- GE (Gene Expression)
- GIOP (General Inter-Orb Protocol)
- GLS (General Ledger Specification)
- IDL (Interface Definition Language)
- IFF (Interrogation, Friend or Foe)
- IIOP (Internet Inter-Orb Protocol)
- 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)
- MQS (MQSeries)
- 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)
- OTS (Object Transaction Service)
- PIDS (Personal Identification Service)
- PIM (Platform Independent Model)
- PSM (Platform Specific Model)
- P&P (Policies and Procedures of the OMG Technical Process)
- RFC (Request For Call)
- RFP (Request For Proposal)
- RM-ODP (Reference Model of Open Distributed Processing)
- RTF (Revision Task Force)
- SEC (Security Service)
- SOA (Service Oriented Architecture)
- SoaML (Service oriented architecture Modeling Language)
- SOLAS (Safety Of Life At Sea)
- SPEM (Software Process Engineering Metamodel)
- 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:

- BAE Systems
- Thales

The following companies supported this specification:

- Atlas Elektronik
- Cassidian
- DSTO
- John Hopkins University APL
- Selex ES
- US Navy

7 Open Architecture Radar Information Specification

7.1 Introduction

The specification is captured as an Enterprise Architect (EA) UML version 2.1 model, with this document being automatically generated as a report from the model.

7.1.1 Document Structure

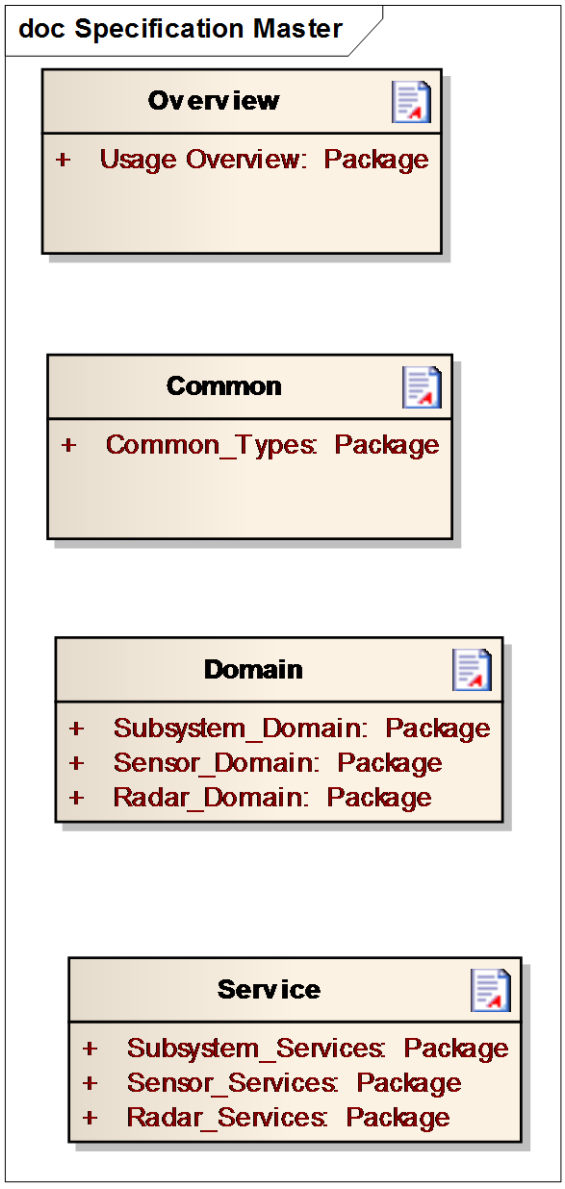

Figure 7.1 -Specification Master (Documentation diagram)

7.2 Usage Overview

Parent Package: Analysis Model (PIM)

The RFP defines a number of 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 NNSI) and Level 3G (compliance with METOC). These are not covered by this response.

Radars 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 (in addition to level 2)

The activity diagrams and the associated notes below show how the interfaces defined in 7.7 to 7.9 interact in order to support these compliance levels.

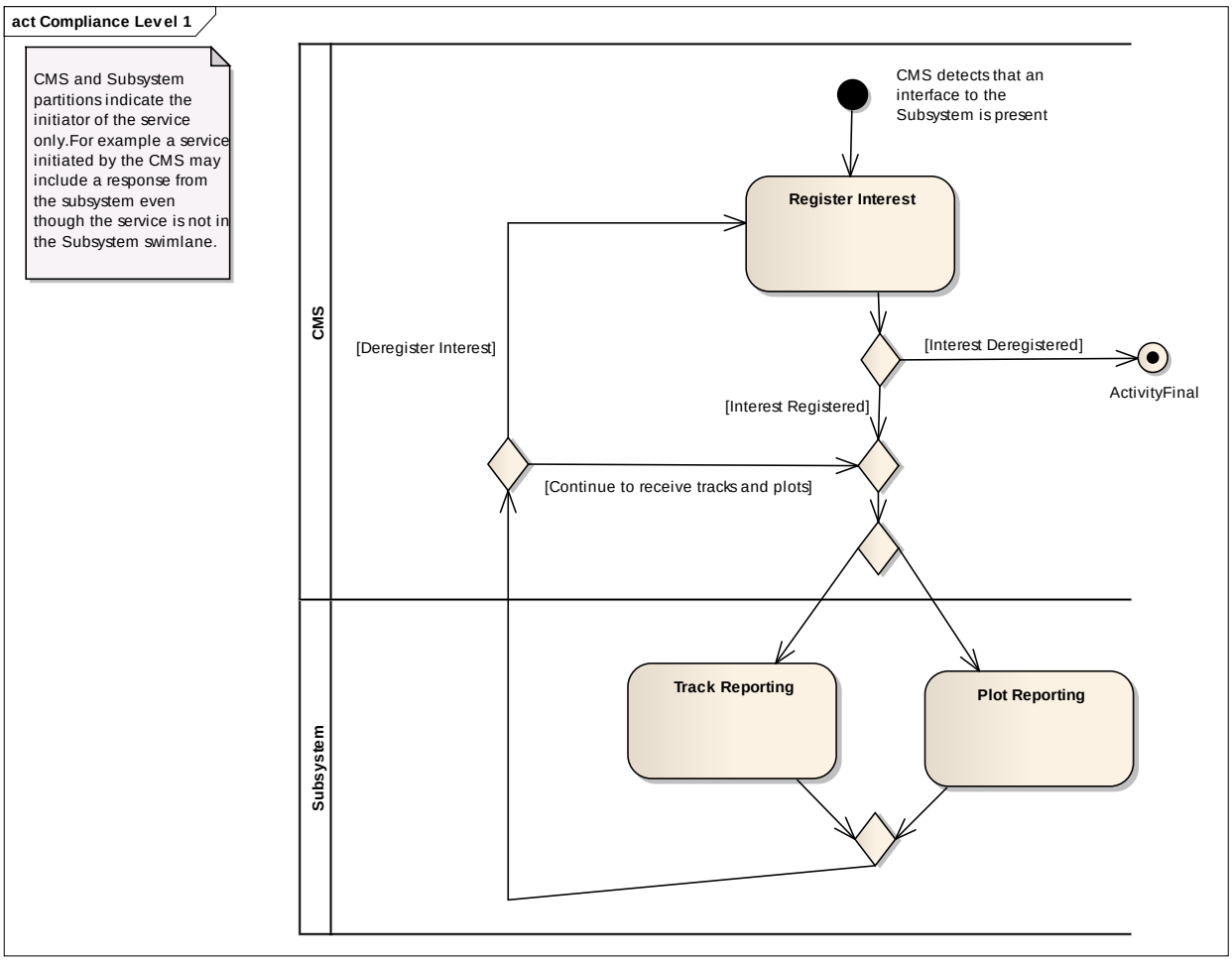

Figure 7.2 Compliance Level 1 (Activity diagram)

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.

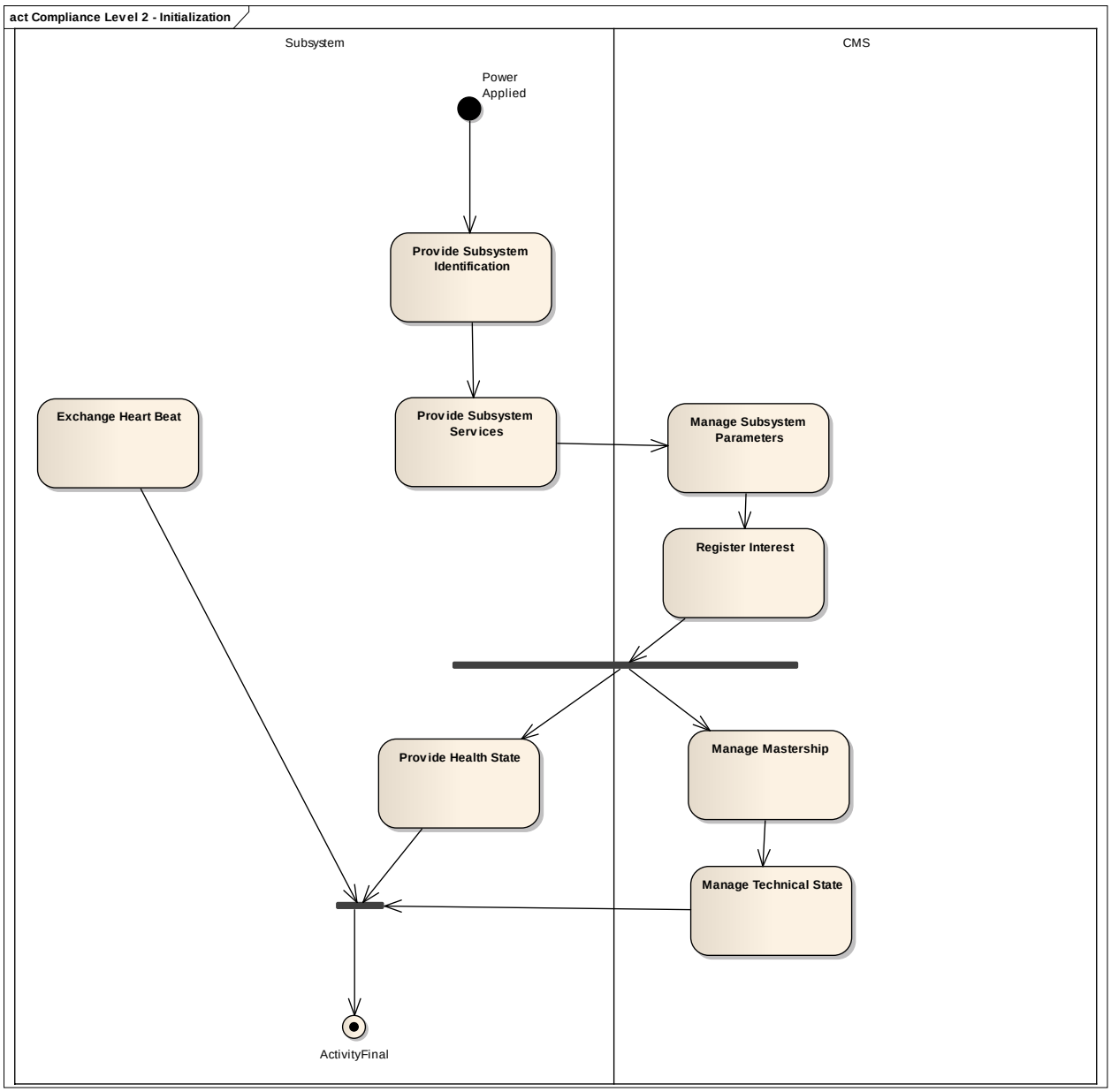

Figure 7.3 Compliance Level 2 - Initialization (Activity diagram)

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.

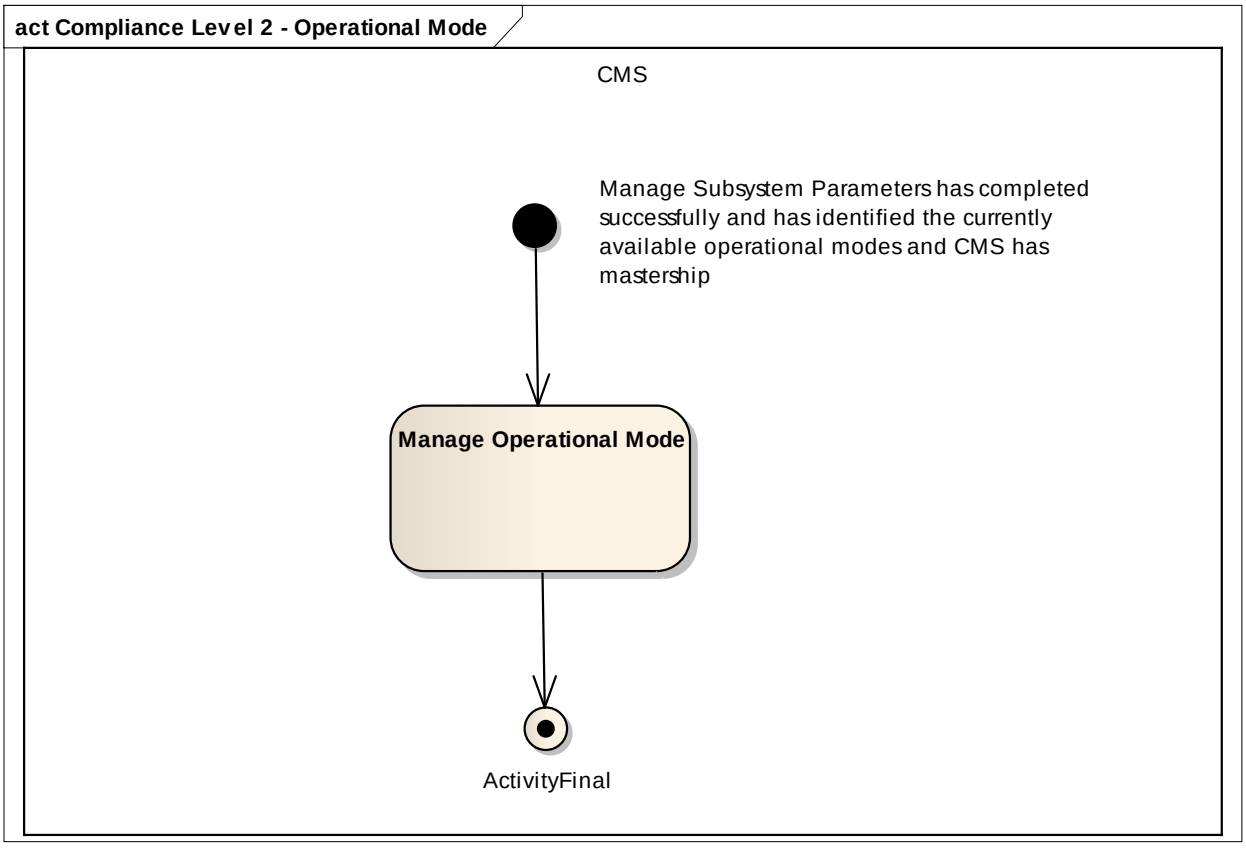

Figure 7.4 Compliance Level 2 - Operational Mode (Activity diagram)

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

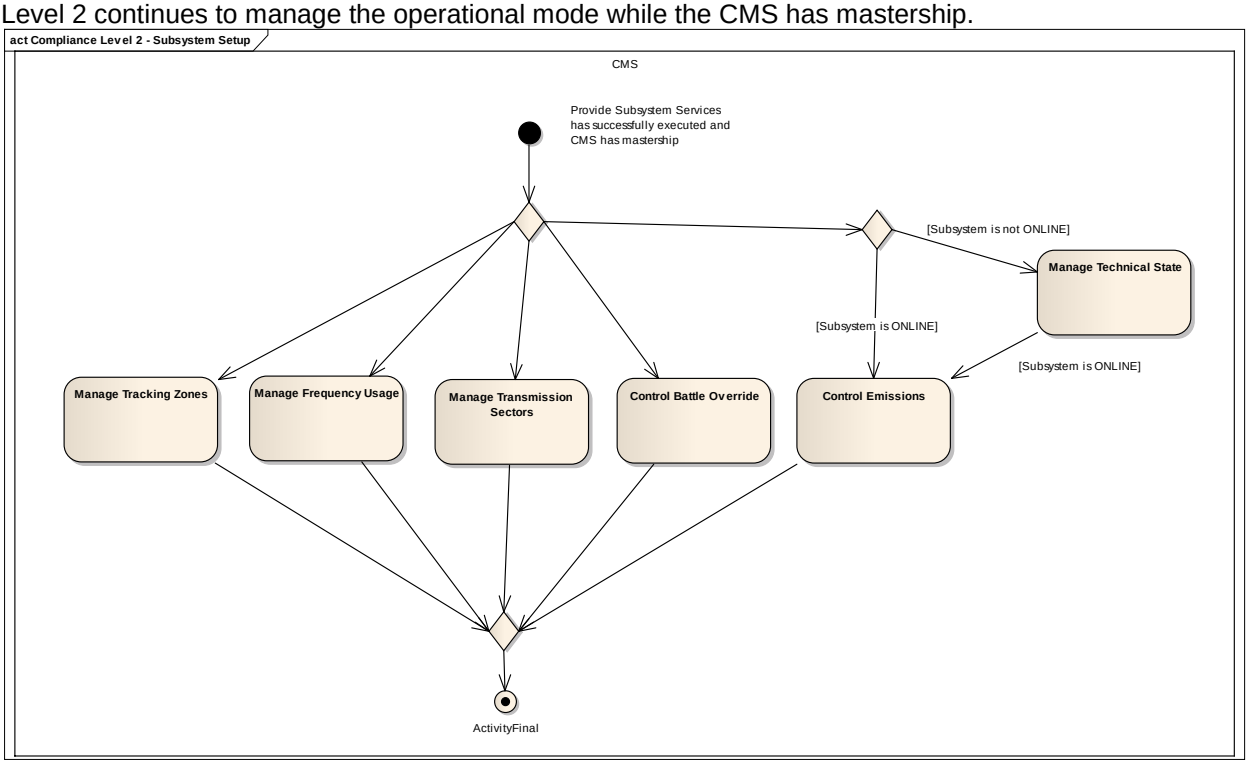

Figure 7.5 Compliance Level 2 - Subsystem Setup (Activity diagram)

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

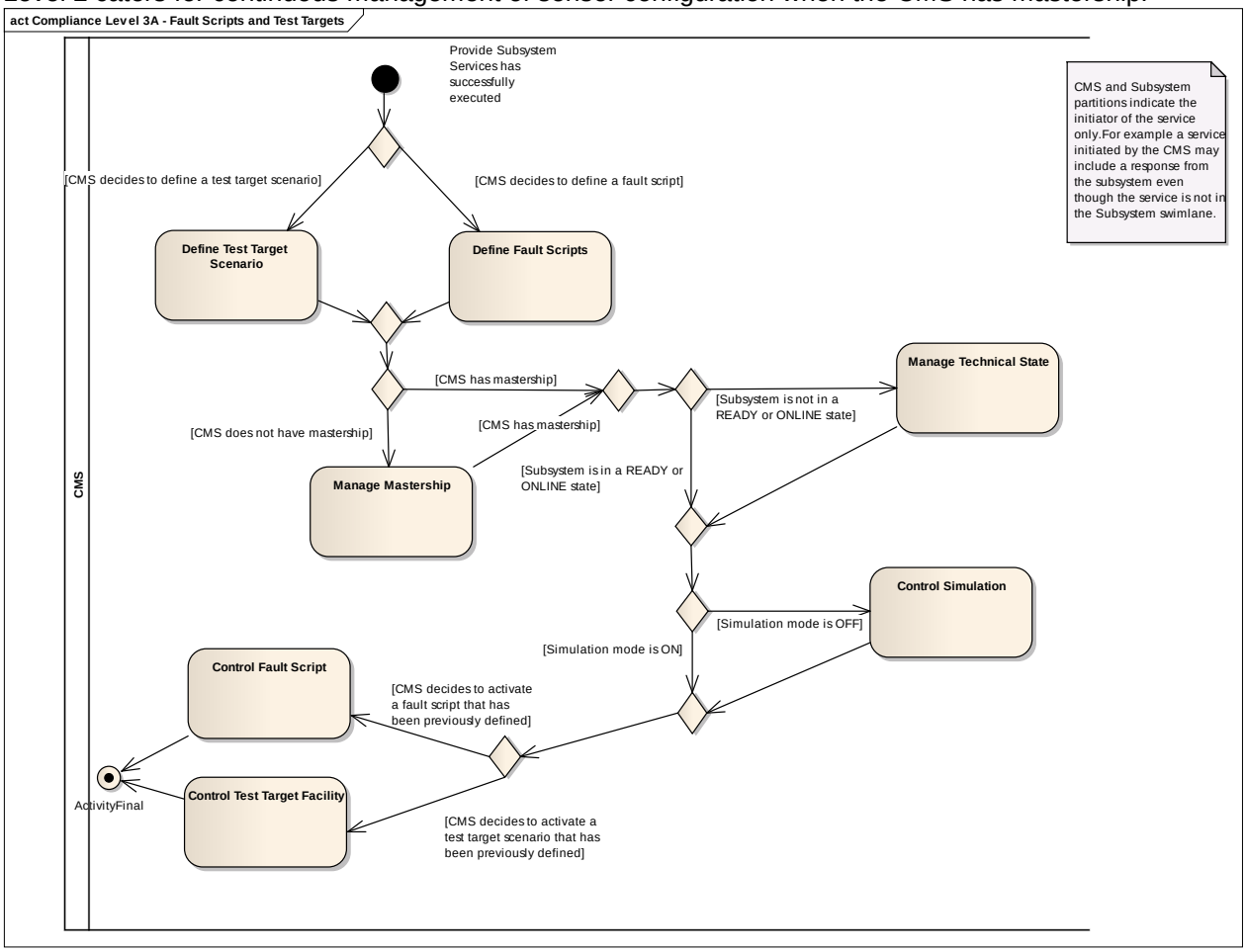

Figure 7.6 Compliance Level 3A - Fault Scripts and Test Targets (Activity diagram)

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

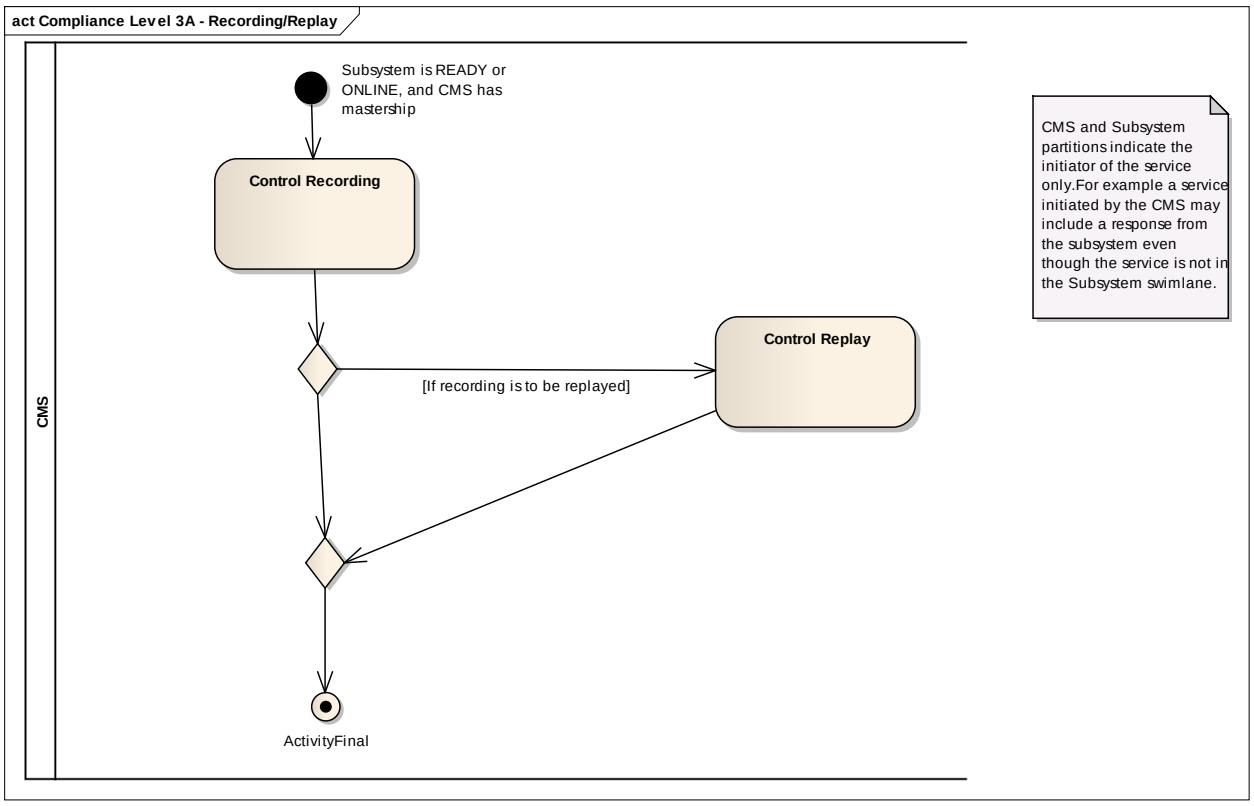

Figure 7.7 Compliance Level 3A - Recording/Replay (Activity diagram)

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

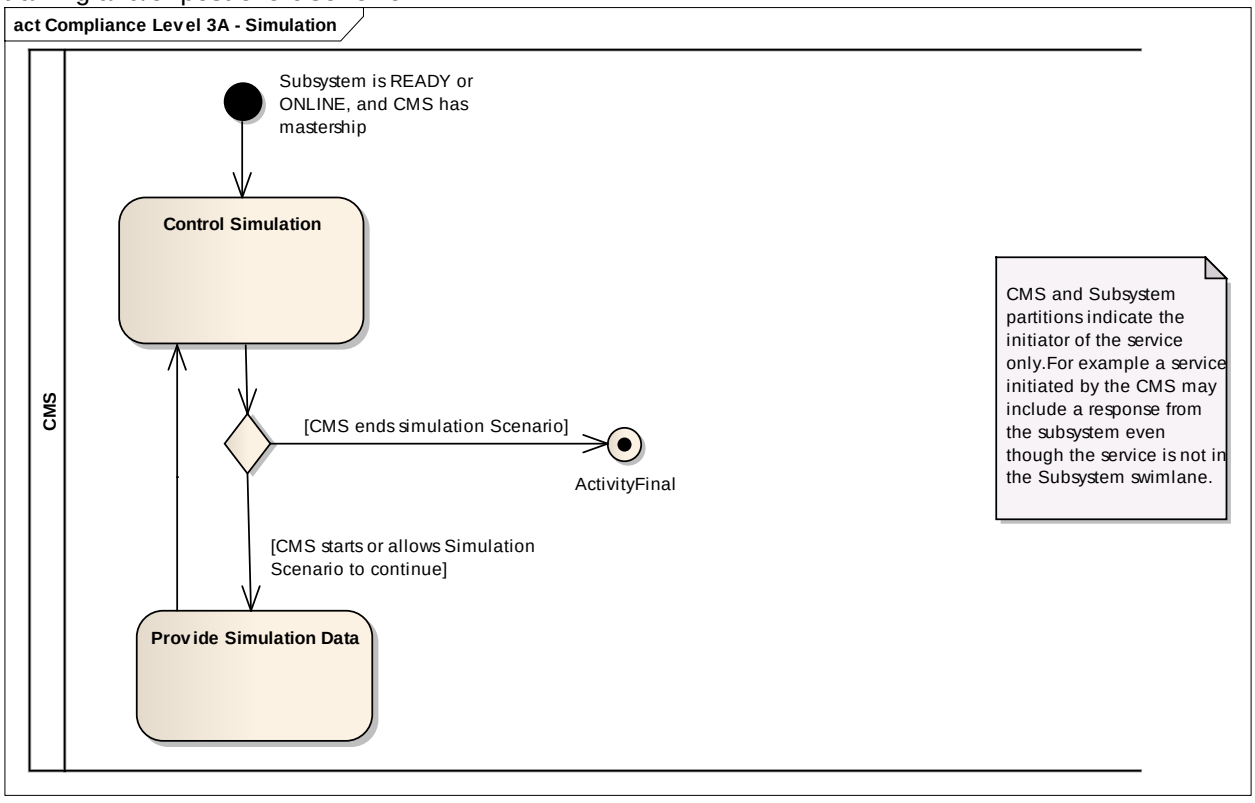

Figure 7.8 Compliance Level 3A - Simulation (Activity diagram)

The simulation interfaces are used to support training.

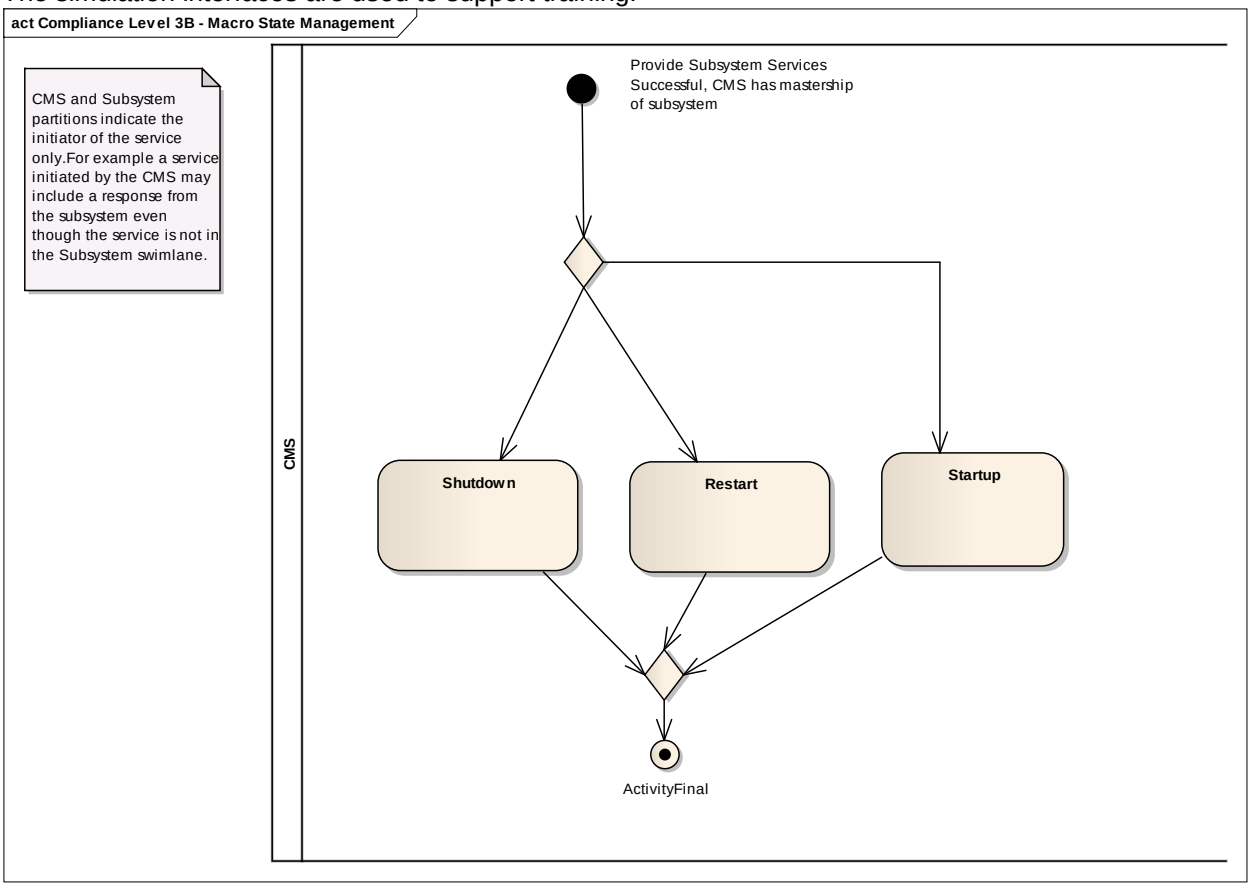

Figure 7.9 Compliance Level 3B - Macro State Management (Activity diagram)

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

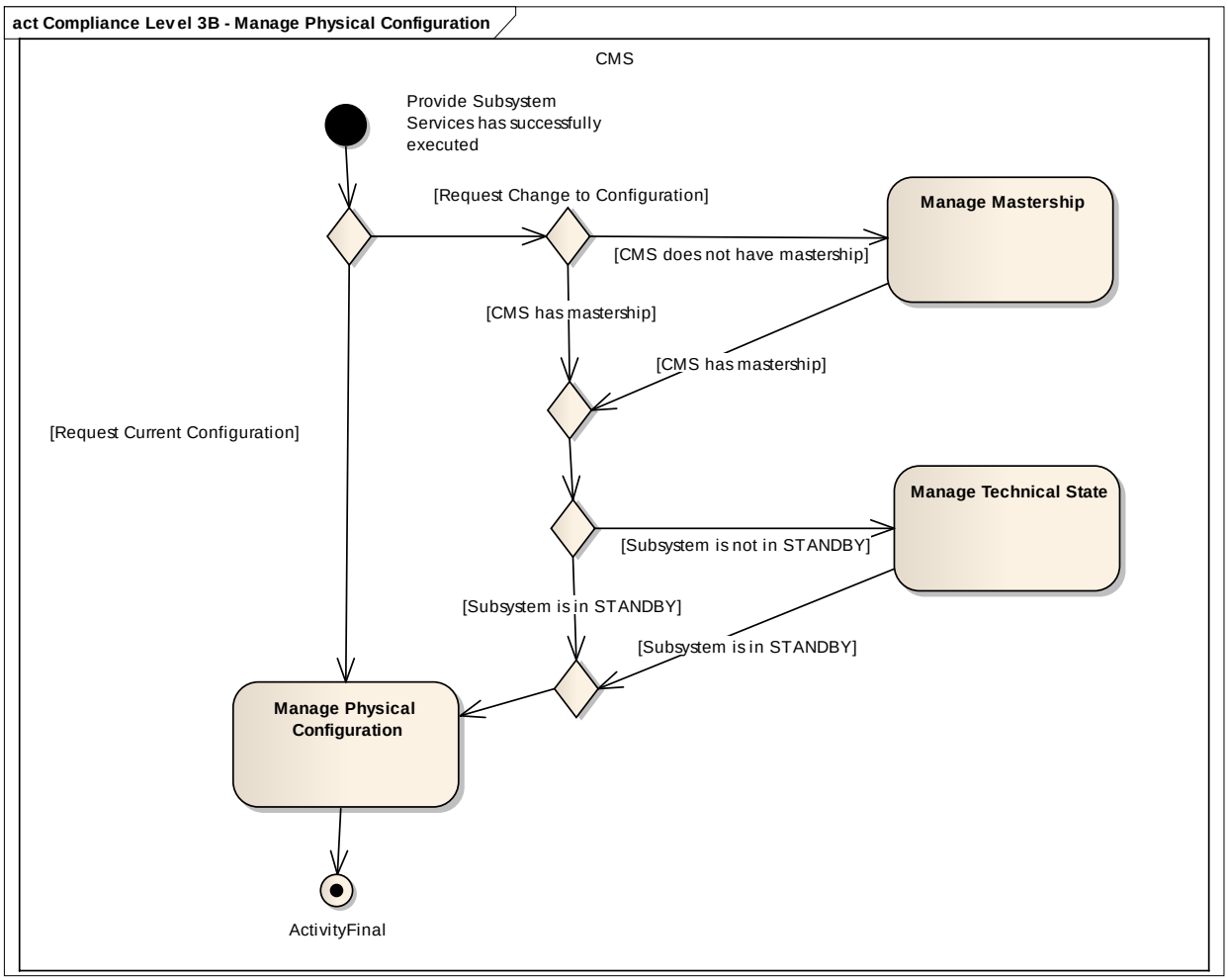

Figure 7.10 Compliance Level 3B - Manage Physical Configuration (Activity diagram)

These interfaces support more detailed control of the subsystem configuration.

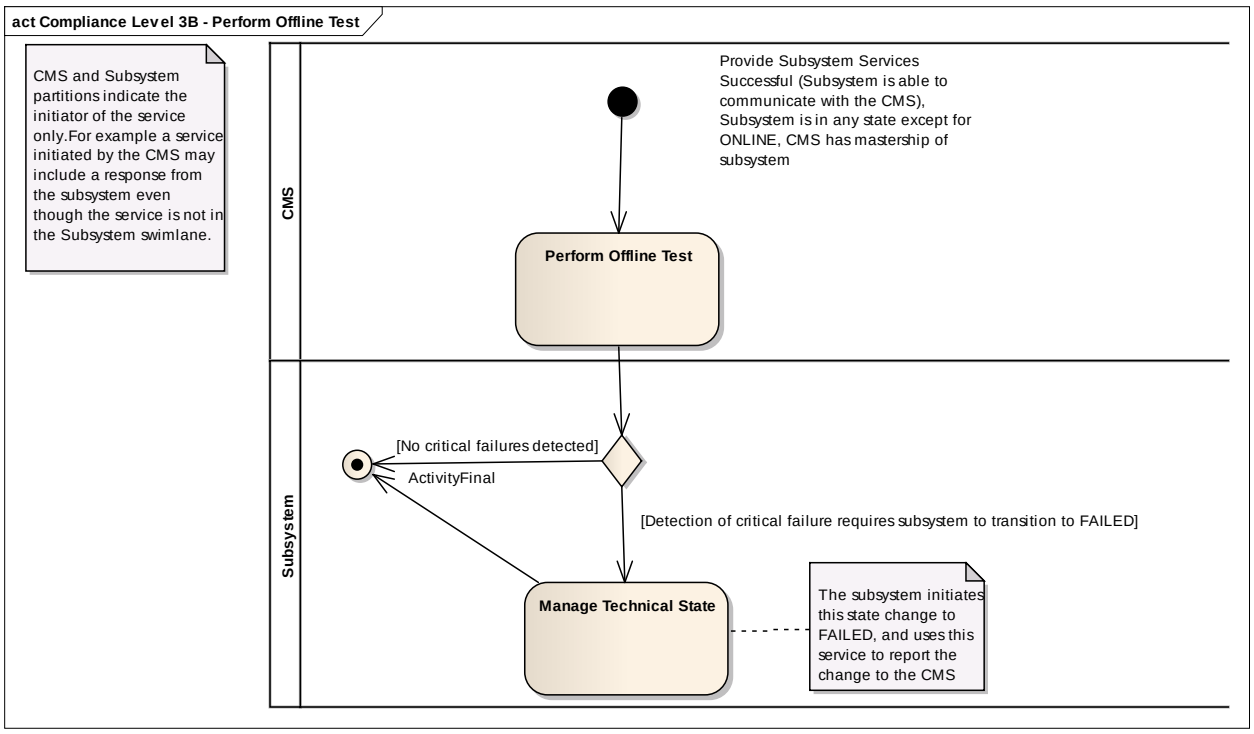

Figure 7.11 Compliance Level 3B - Perform Offline Test (Activity diagram)

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

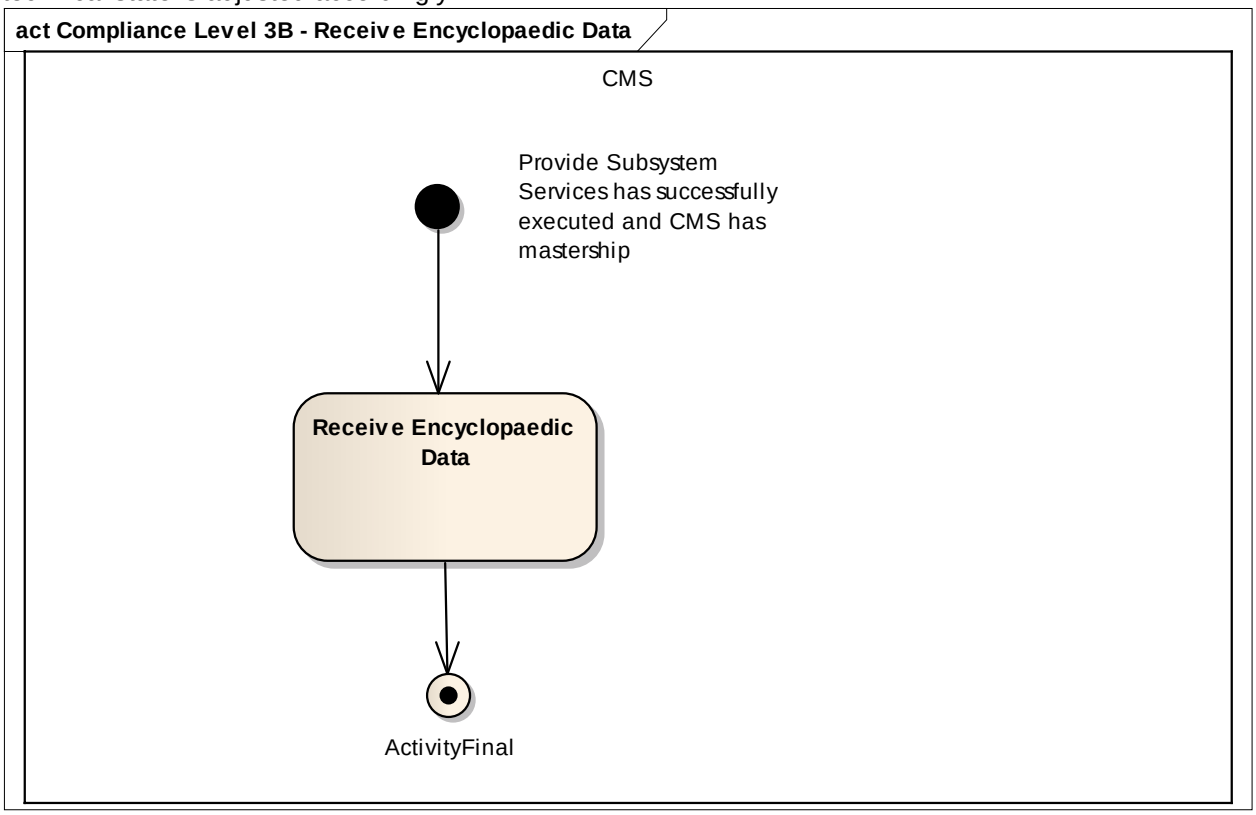

Figure 7.12 Compliance Level 3B - Receive Encyclopaedic Data (Activity diagram)

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

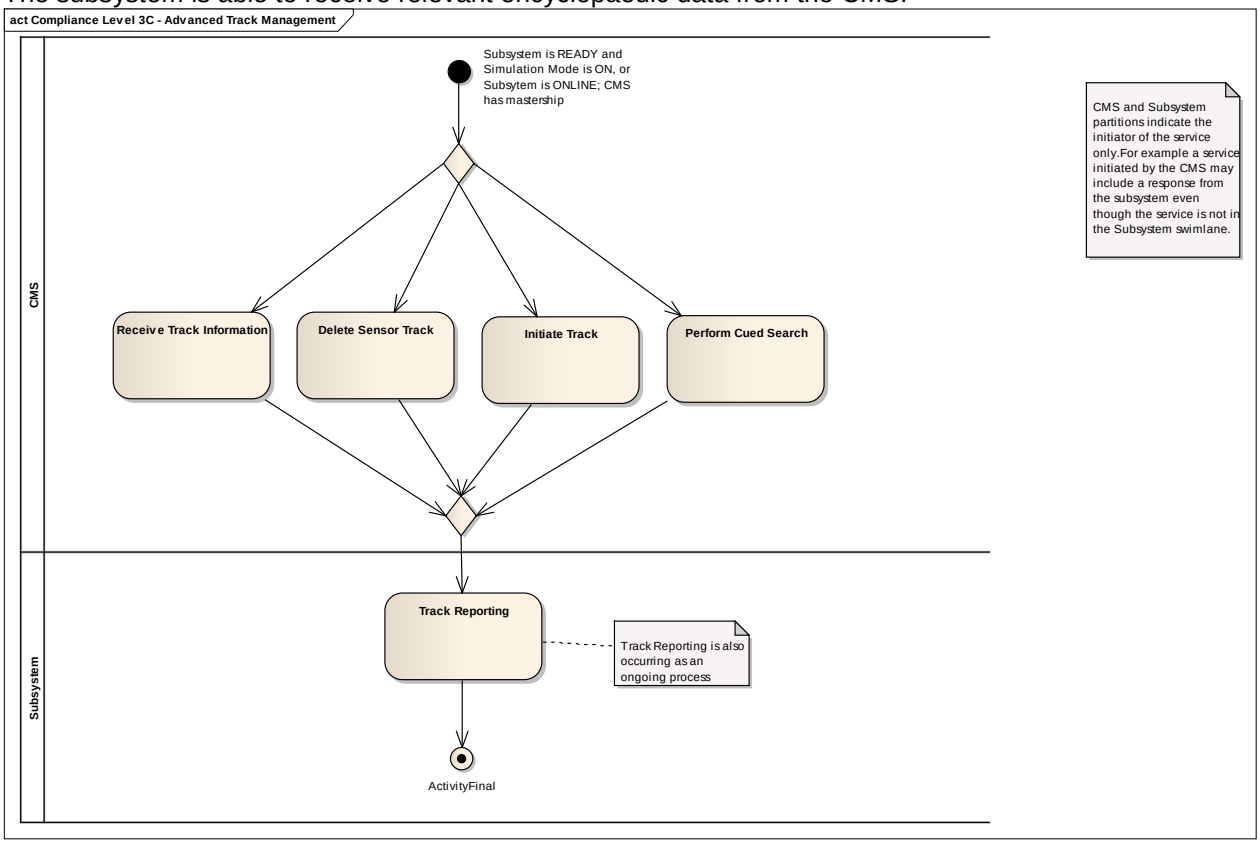

Figure 7.13 Compliance Level 3C - Advanced Track Management (Activity diagram)

The sensor supports detailed track management.

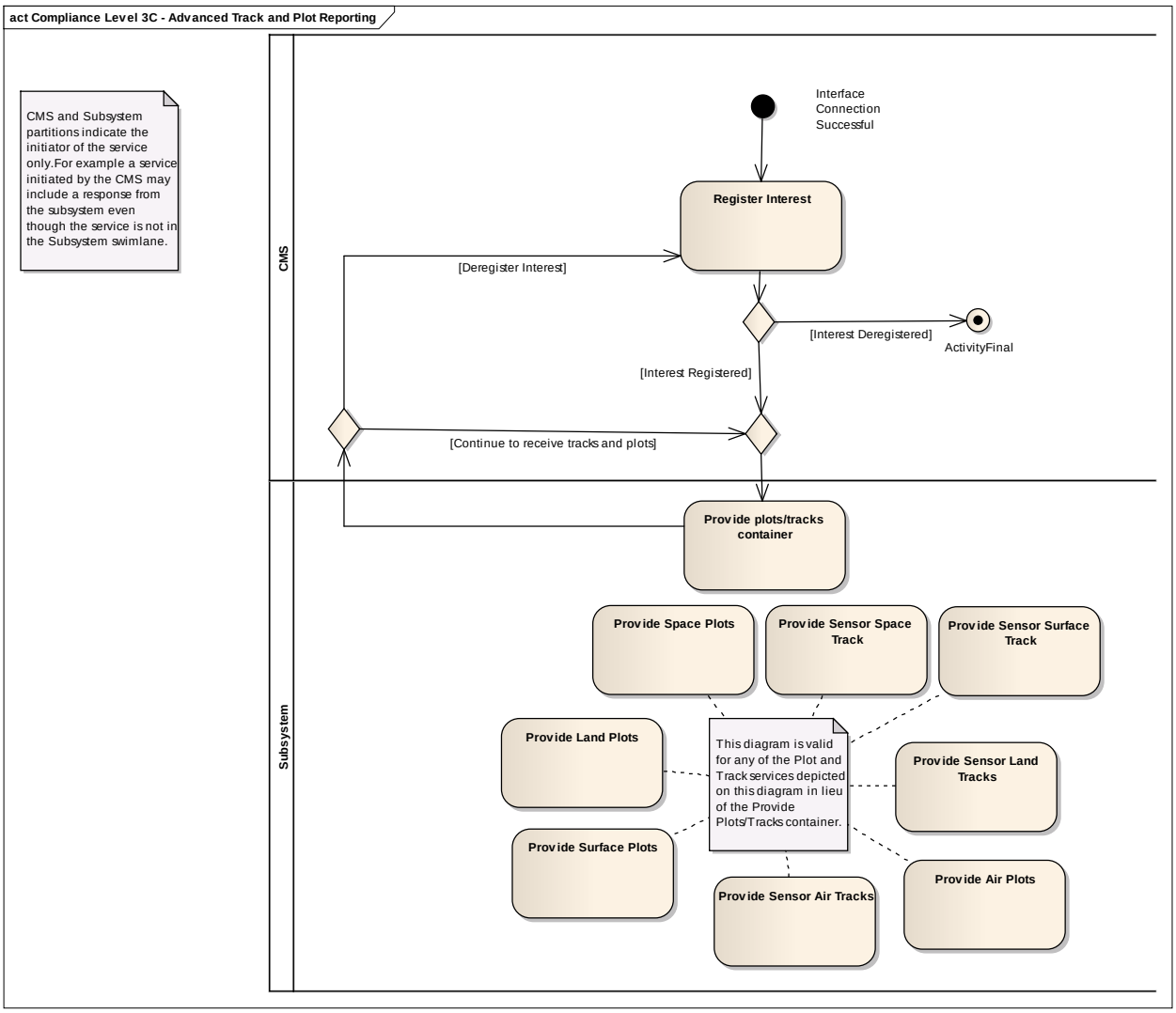

Figure 7.14 Compliance Level 3C - Advanced Track and Plot Reporting (Activity diagram)

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

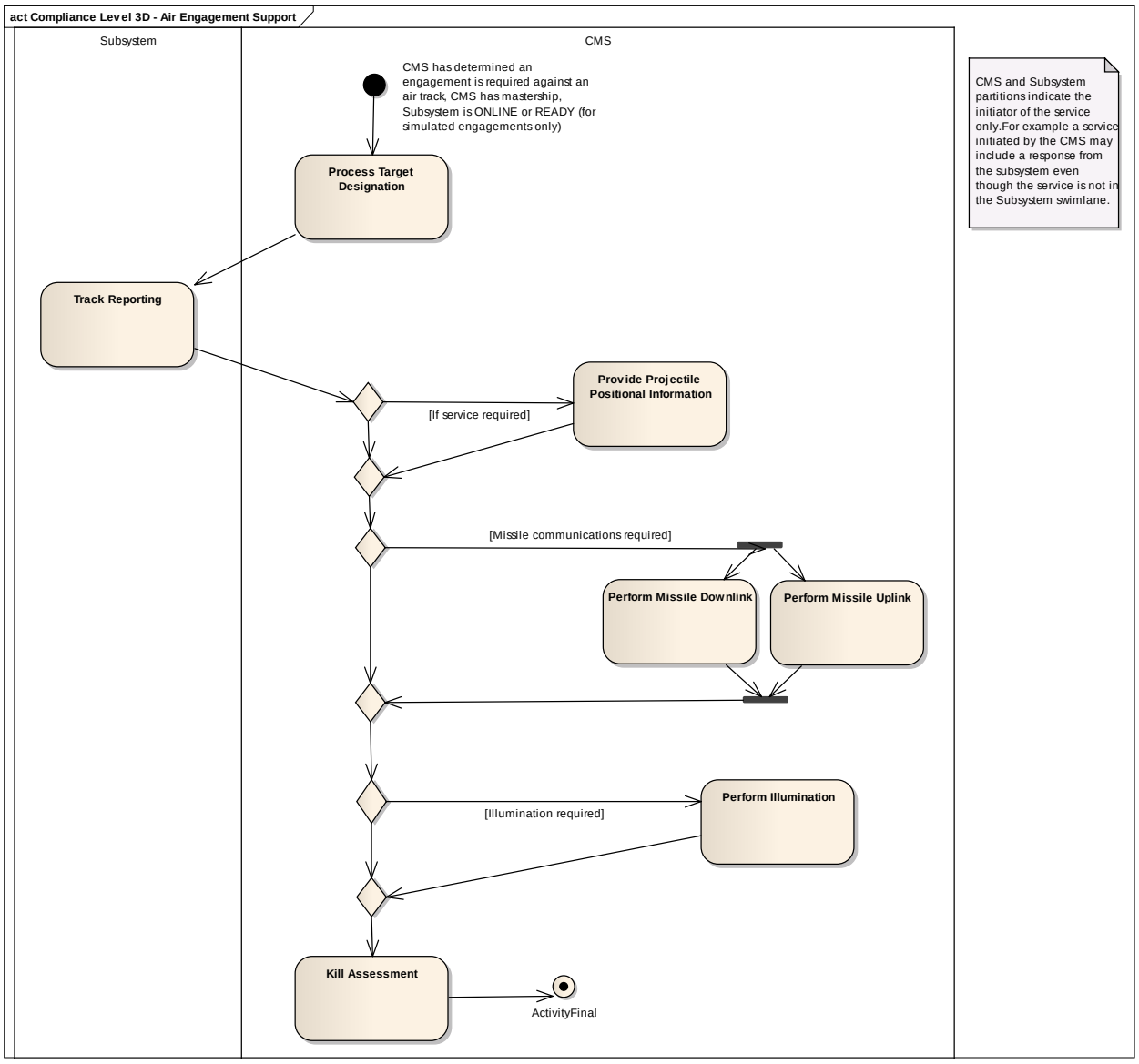

Figure 7.15 Compliance Level 3D - Air Engagement Support (Activity diagram)

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

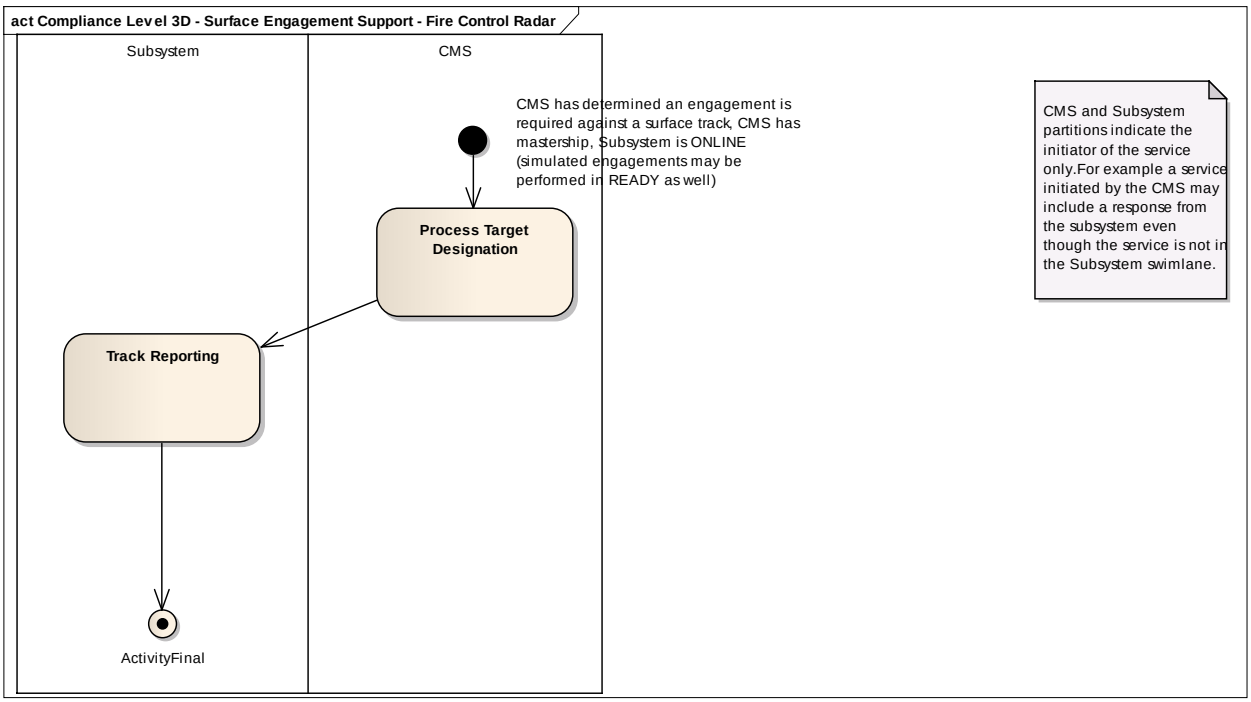

Figure 7.16 Compliance Level 3D - Surface Engagement Support - Fire Control Radar (Activity diagram)

This provides additional surface engagement support for fire control.

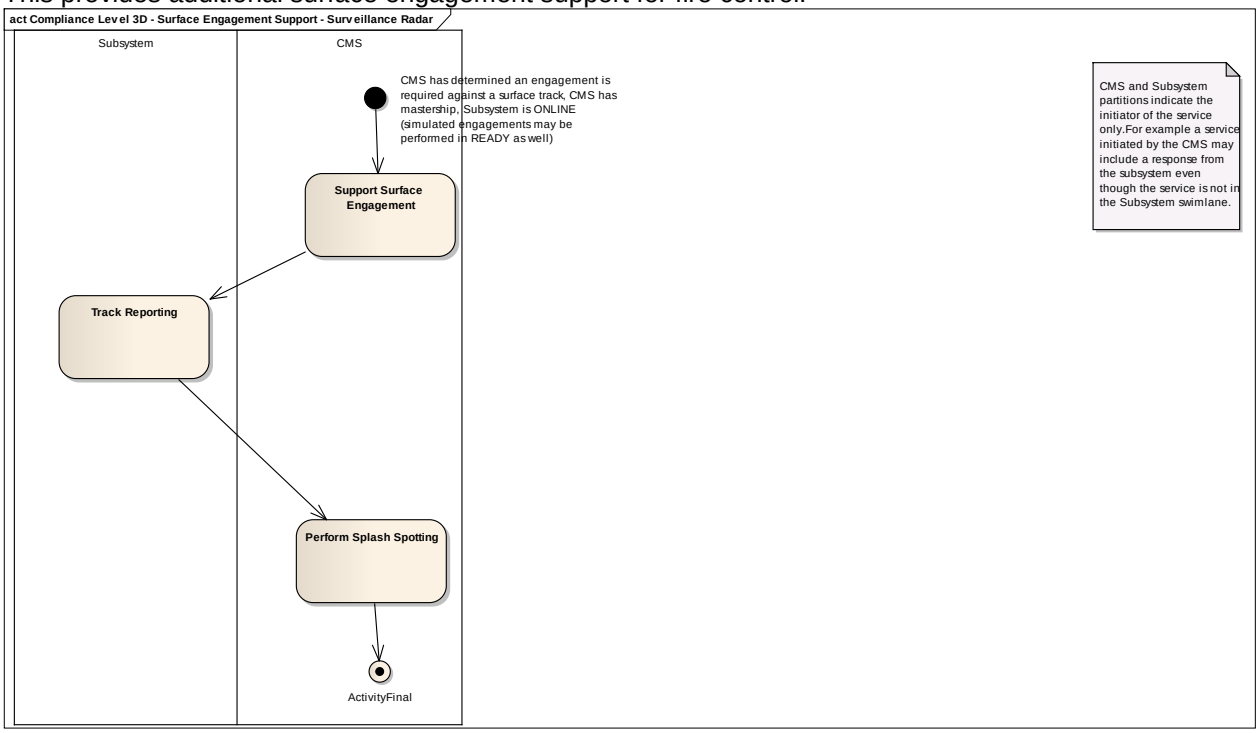

Figure 7.17 Compliance Level 3D - Surface Engagement Support - Surveillance Radar (Activity diagram)

This provides additional surface engagement support for surveillance purposes.

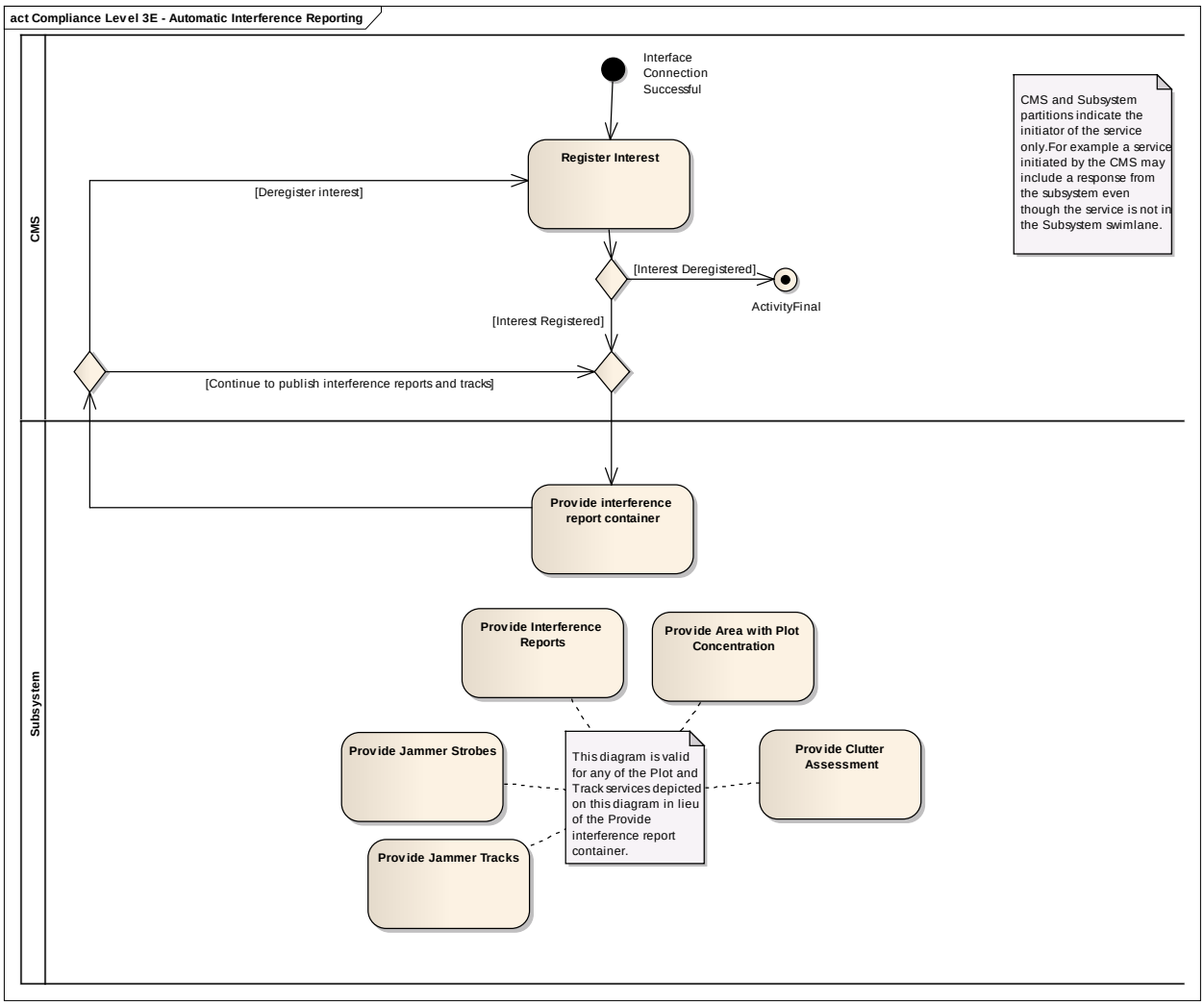

Figure 7.18 Compliance Level 3E - Automatic Interference Reporting (Activity diagram)

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

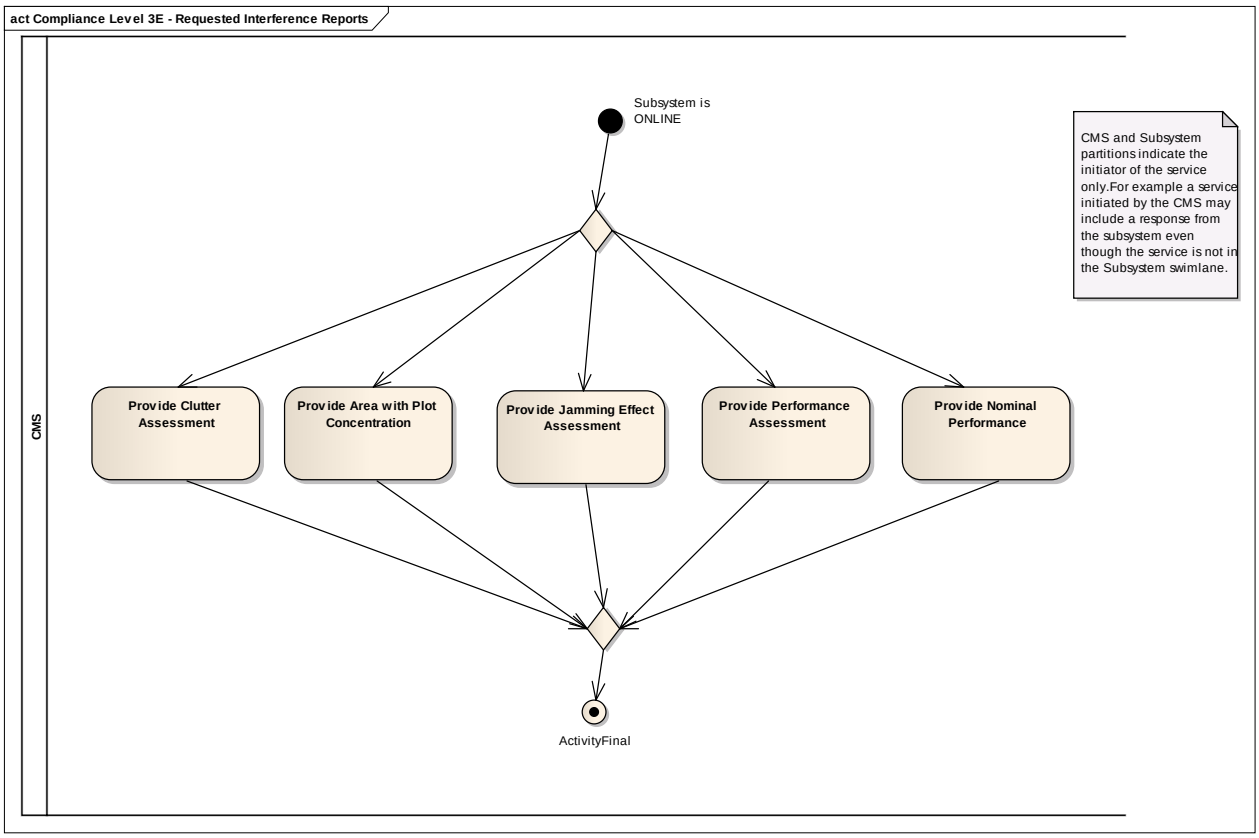

Figure 7.19 Compliance Level 3E - Requested Interference Reports (Activity diagram)

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

7.3 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.

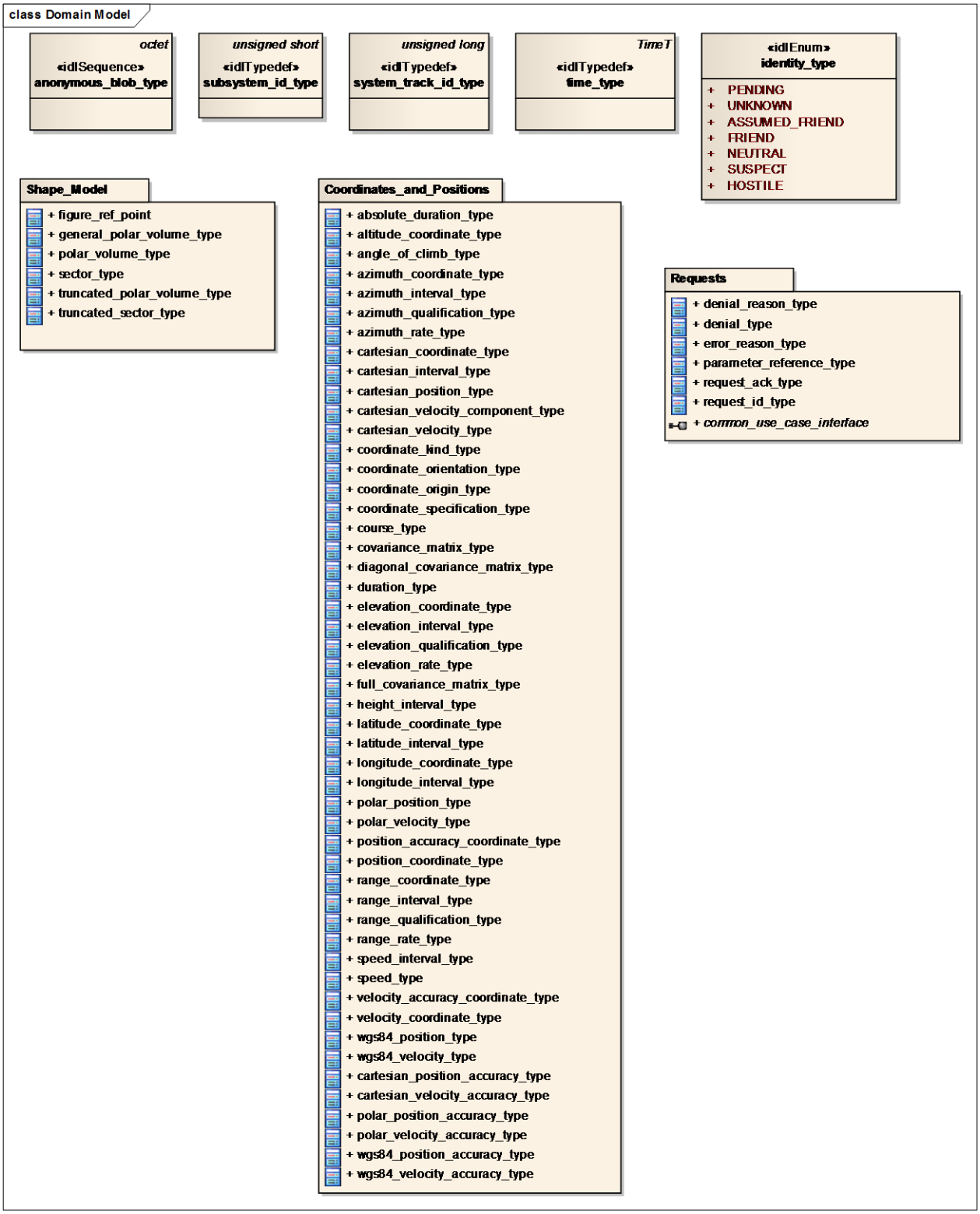

Figure 7.20 Domain Model (Logical diagram)

7.3.1 anonymous_blob_type

Type: IDLSequence octet
Package: Common_Types
Representation for a general binary type
Length = 1024

7.3.2 identity_type

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

Table 7.1 - Attributes of IDLEnum identity_type

Attribute	Notes
PENDING	
UNKNOWN	
ASSUMED_FRIEND	
FRIEND	
NEUTRAL	
SUSPECT	
HOSTILE	

7.3.3 subsystem_id_type

Type: IDLTypeDef unsigned short
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.
BaseType = unsigned short

7.3.4 system_track_id_type

Type: IDLTypeDef unsigned long
Package: Common_Types
System Track Identification

7.3.5 time_type

Type: IDLTypeDef TimeT
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

7.3.6 System_Track

Parent Package: Common_Types

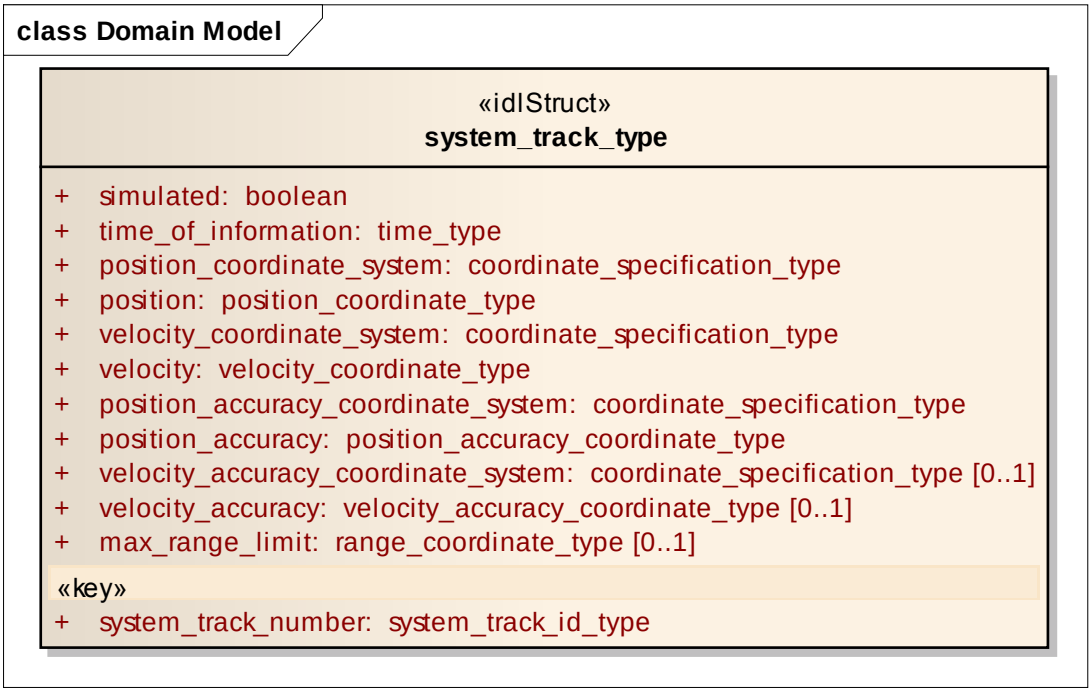

Figure 7.21 Domain Model (Logical diagram)

7.3.6.1 system_track_type

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

Table 7.2 - Attributes of IDLStruct system_track_type

Attribute	Notes
«key» **system_track_number** system_track_id_type	
simulated boolean	
time_of_information time_type	
position_coordinate_system coordinate_specification_type	
position position_coordinate_type	
velocity_coordinate_system coordinate_specification_type	
velocity velocity_coordinate_type	
position_accuracy_coordinate_system coordinate_specification_type	
position_accuracy position_accuracy_coordinate_type	
velocity_accuracy_coordinate_system coordinate_specification_type [0..1]	
velocity_accuracy velocity_accuracy_coordinate_type [0..1]	
max_range_limit range_coordinate_type [0..1]	

7.3.7 Coordinates_and_Positions

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

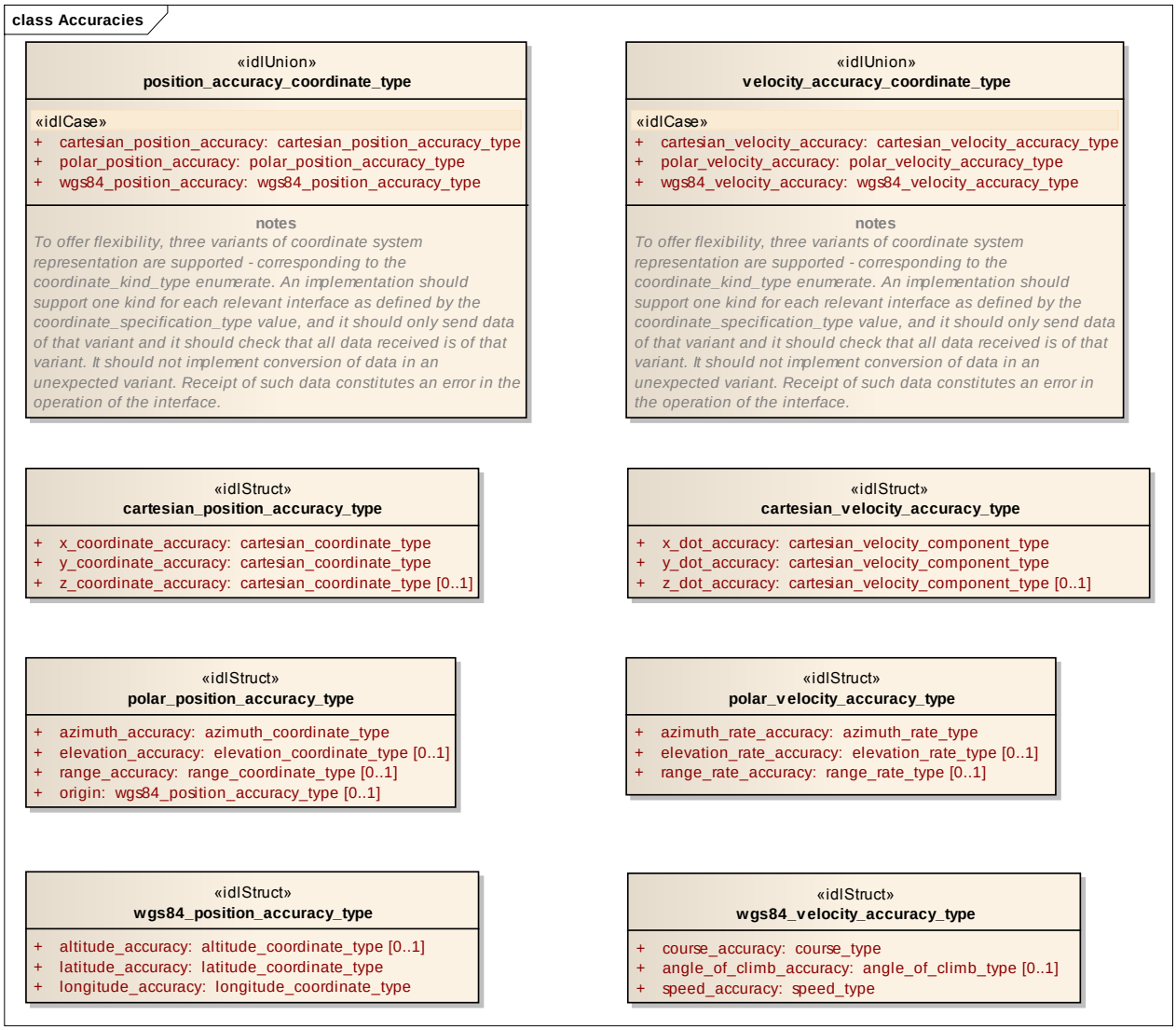

Figure 7.22 Accuracies (Logical diagram)

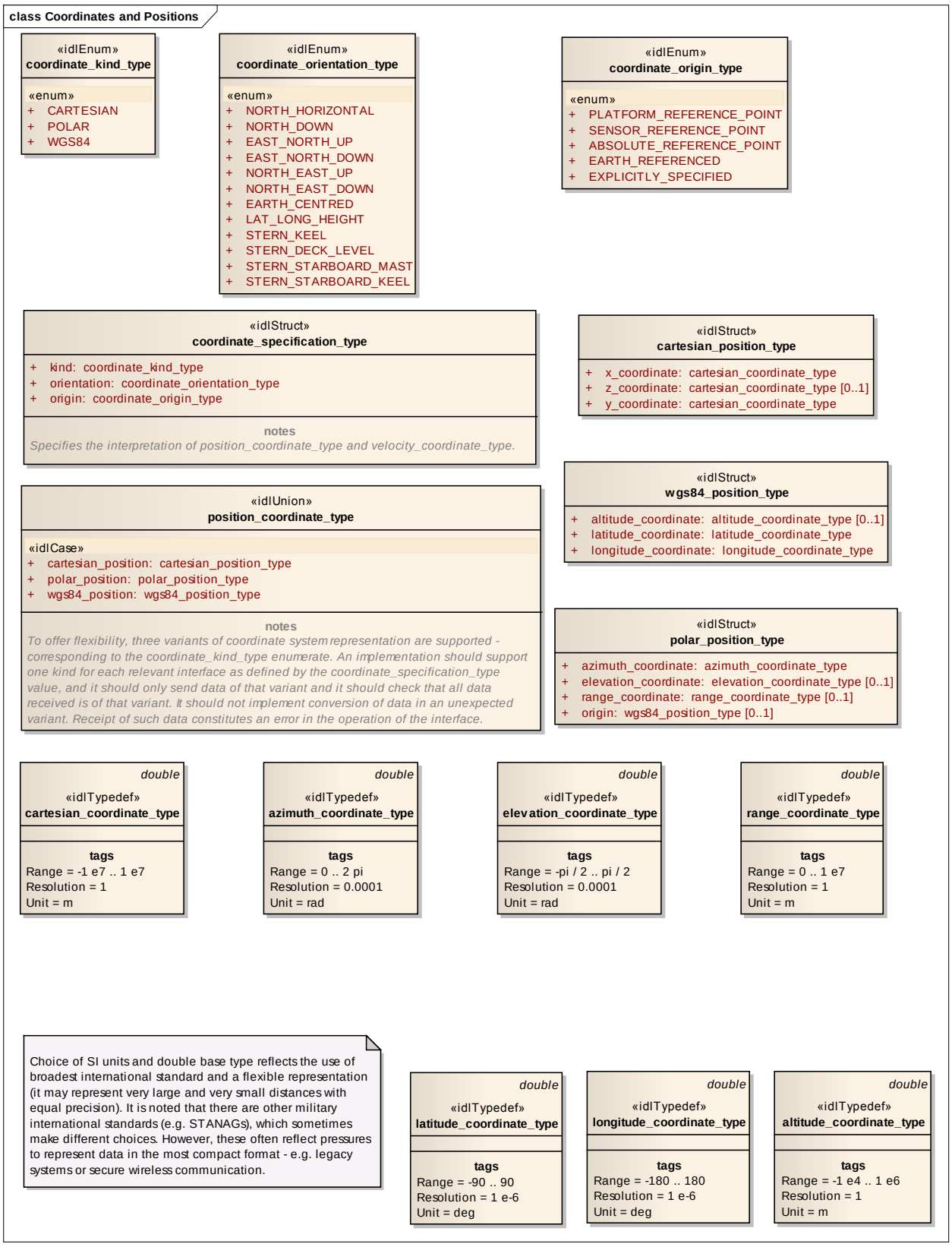

Figure 7.23 Coordinates and Positions (Logical diagram)

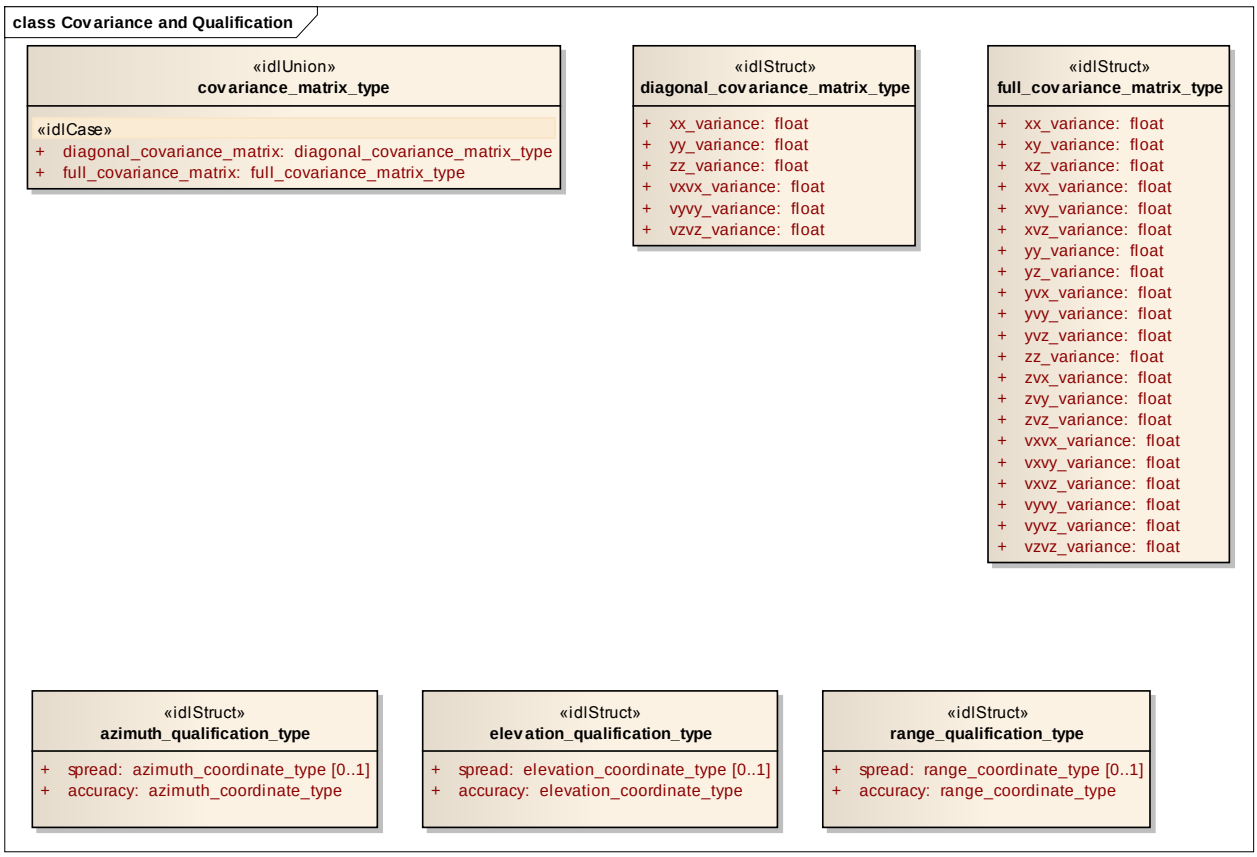

Figure 7.24 Covariance and Qualification (Logical diagram)

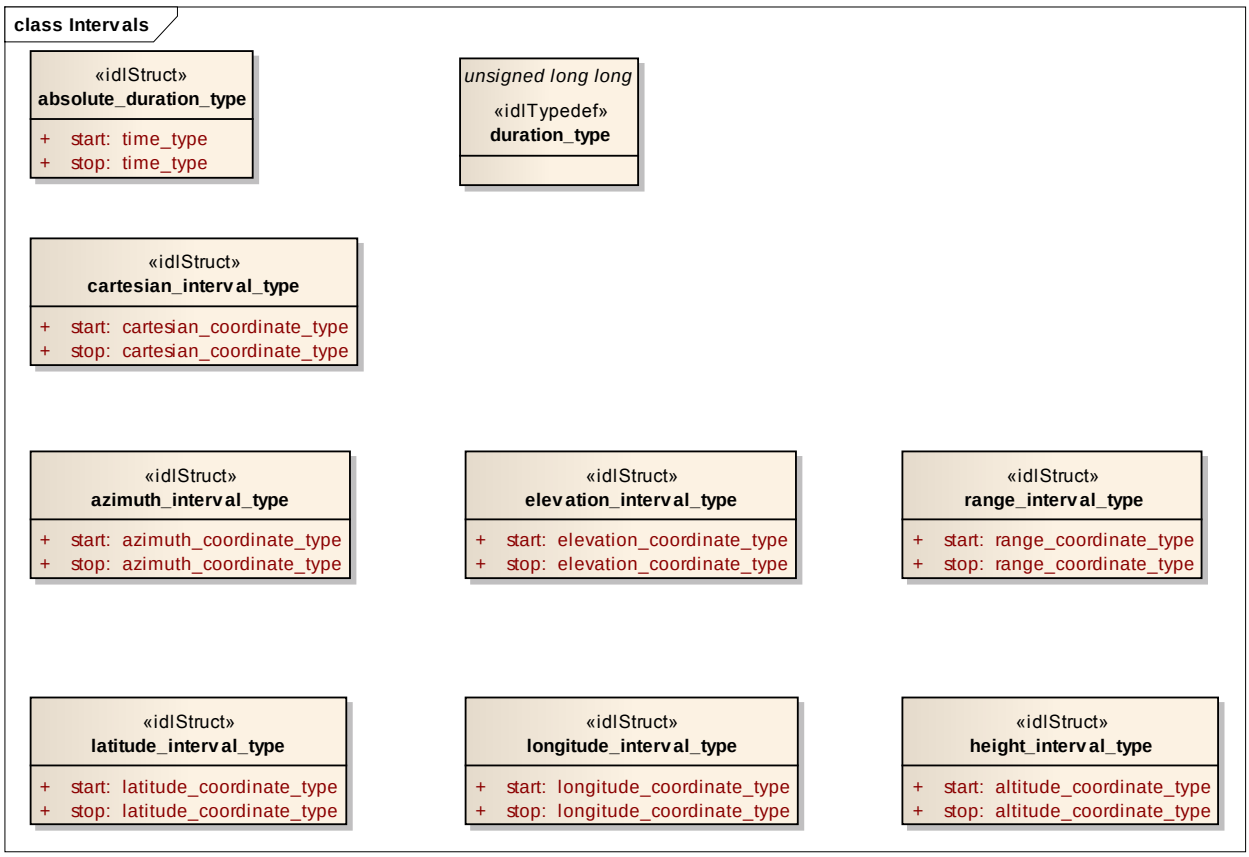

Figure 7.25 Intervals (Logical diagram)

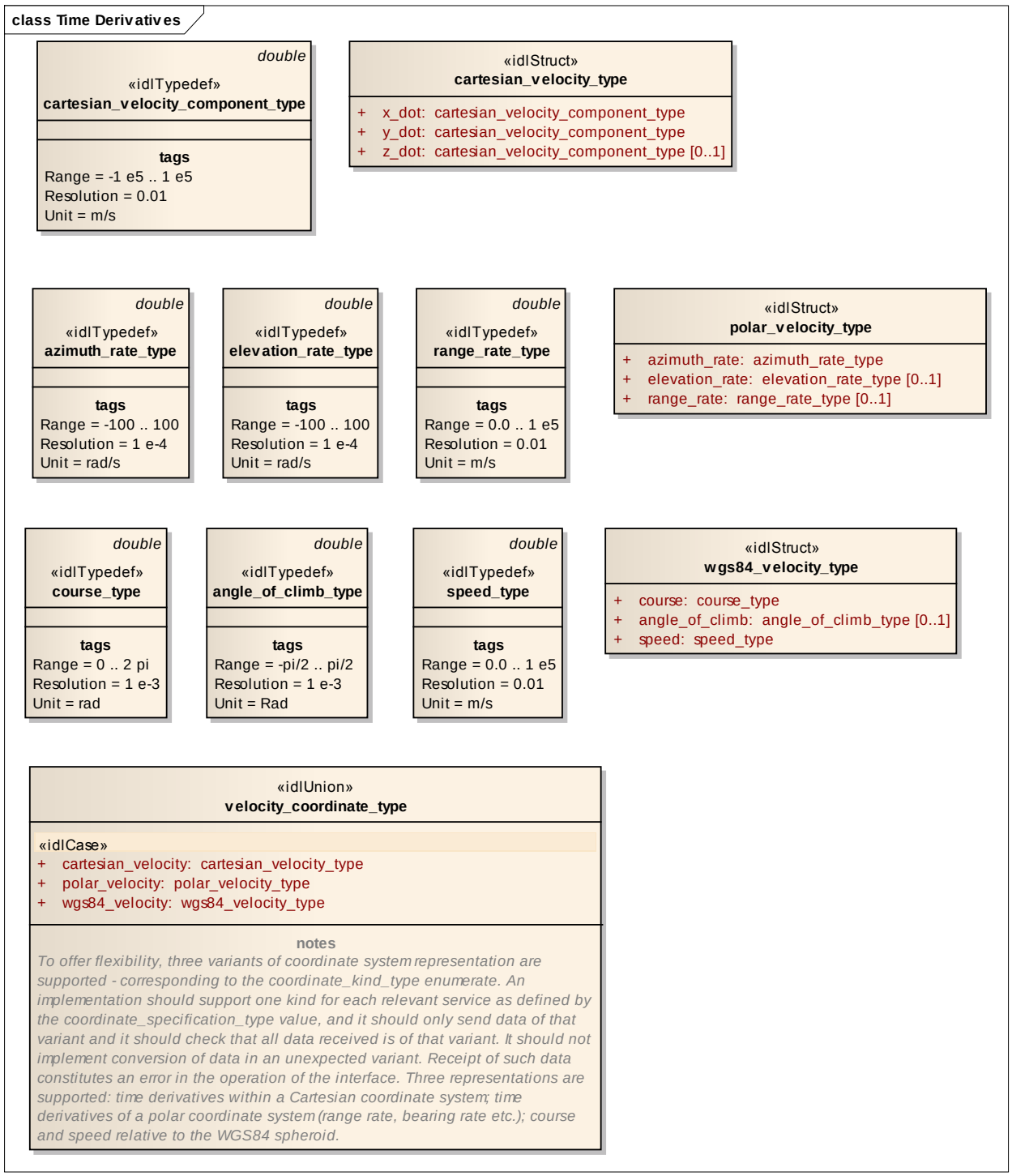

Figure 7.26 Time Derivatives (Logical diagram)

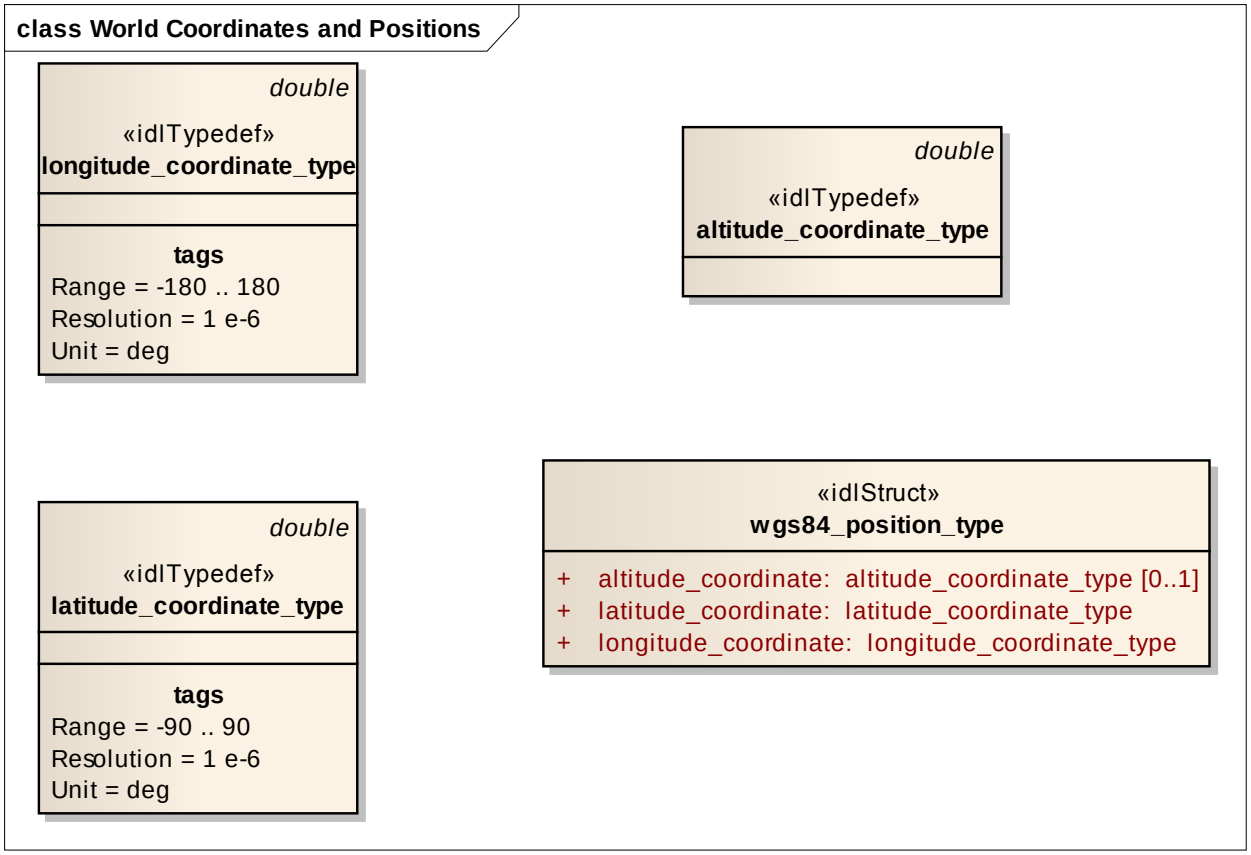

Figure 7.27 World Coordinates and Positions (Logical diagram)

7.3.7.1 absolute_duration_type

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

Table 7.3 - Attributes of IDLStruct absolute_duration_type

Attribute	Notes
start time_type	
stop time_type	

7.3.7.2 altitude_coordinate_type

Type: IDLTypeDef double
Package: Coordinates_and_Positions
For positive values, height above coordinate system ellipsoid, for negative values, depth below; measured in metres.
See diagram note on choice of SI units
Range = -1 e4 .. 1 e6
Resolution = 1
Unit = m

7.3.7.3 angle_of_climb_type

Type: IDLTypeDef double
Package: Coordinates_and_Positions

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

7.3.7.4 azimuth_coordinate_type

Type: IDLTypeDef double
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
Range = 0 .. 2 pi
Resolution = 0.0001
Unit = rad

7.3.7.5 azimuth_interval_type

Type: IDLStruct
Package: Coordinates_and_Positions

Table 7.4 - Attributes of IDLStruct azimuth_interval_type

Attribute	Notes
start azimuth_coordinate_type	
stop azimuth_coordinate_type	

7.3.7.6 azimuth_qualification_type

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

Table 7.5 - 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.

7.3.7.7 azimuth_rate_type

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

7.3.7.8 cartesian_coordinate_type

Type: IDLTypeDef double
Package: Coordinates_and_Positions
See diagram note on choice of SI units
Range = -1 e7 .. 1 e7
Resolution = 1
Unit = m

7.3.7.9 cartesian_interval_type

Type: IDLStruct
Package: Coordinates_and_Positions

Table 7.6 - Attributes of IDLStruct cartesian_interval_type

Attribute	Notes
start cartesian_coordinate_type	
stop cartesian_coordinate_type	

7.3.7.10 cartesian_position_type

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

Table 7.7 - Attributes of IDLStruct cartesian_position_type

Attribute	Notes
x_coordinate cartesian_coordinate_type	
z_coordinate cartesian_coordinate_type [0..1]	Optional as some sensors are 2D (horizontal plane or no elevation information)
y_coordinate cartesian_coordinate_type	

7.3.7.11 cartesian_velocity_component_type

Type: IDLTypeDef double
Package: Coordinates_and_Positions

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

7.3.7.12 cartesian_velocity_type

Type: IDLStruct
Package: Coordinates_and_Positions

Table 7.8 - Attributes of IDLStruct cartesian_velocity_type

Attribute	Notes
x_dot cartesian_velocity_component_type	
y_dot cartesian_velocity_component_type	
z_dot cartesian_velocity_component_type [0..1]	

7.3.7.13 coordinate_kind_type

Type: IDLEnum
Package: Coordinates_and_Positions

Table 7.9 - Attributes of IDLEnum coordinate_kind_type

Attribute	Notes
«enum» **CARTESIAN**	
«enum» **POLAR**	
«enum» **WGS84**	

7.3.7.14 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 7.10 - 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 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
«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)

Attribute	Notes
«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)

7.3.7.15 coordinate_origin_type

Type: IDLEnum
Package: Coordinates_and_Positions

Table 7.11 - 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.

7.3.7.16 coordinate_specification_type

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

Table 7.12 - Attributes of IDLStruct coordinate_specification_type

Attribute	Notes
kind coordinate_kind_type	
orientation coordinate_orientation_type	
origin coordinate_origin_type	

7.3.7.17 course_type

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

7.3.7.18 covariance_matrix_type

Type: IDLUnion
Package: Coordinates_and_Positions

Table 7.13 - Attributes of IDLUnion covariance_matrix_type

Attribute	Notes
«idlCase» **diagonal_covariance_matrix** diagonal_covariance_matrix_type	
«idlCase» **full_covariance_matrix** full_covariance_matrix_type	

7.3.7.19 diagonal_covariance_matrix_type

Type: IDLStruct
Package: Coordinates_and_Positions

Table 7.14 - Attributes of IDLStruct diagonal_covariance_matrix_type

Attribute	Notes
xx_variance float	
yy_variance float	
zz_variance float	
vxvx_variance float	
vyvy_variance float	
vzvz_variance float	

7.3.7.20 duration_type

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

7.3.7.21 elevation_coordinate_type

Type: IDLTypeDef double
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 units
Range = -pi / 2 .. pi / 2
Resolution = 0.0001
Unit = rad

7.3.7.22 elevation_interval_type

Type: IDLStruct

Package: Coordinates_and_Positions

Table 7.15 - Attributes of IDLStruct elevation_interval_type

Attribute	Notes
start elevation_coordinate_type	
stop elevation_coordinate_type	

7.3.7.23 elevation_qualification_type

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

Table 7.16 - 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.

7.3.7.24 elevation_rate_type

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

7.3.7.25 full_covariance_matrix_type

Type: IDLStruct
Package: Coordinates_and_Positions
Full covariance matrix

Table 7.17 - Attributes of IDLStruct full_covariance_matrix_type

Attribute	Notes
xx_variance float	
xy_variance float	
xz_variance float	
xvx_variance float	
xvy_variance float	
xvz_variance float	
yy_variance float	
yz_variance float	
yvx_variance float	
yvy_variance float	
yvz_variance float	
zz_variance float	
zvx_variance float	
zvy_variance float	
zvz_variance float	
vxvx_variance float	
vxvy_variance float	

Attribute	Notes
vxvz_variance float	
vyvy_variance float	
vyvz_variance float	
vzvz_variance float	

7.3.7.26 height_interval_type

Type: IDLStruct
Package: Coordinates_and_Positions

Table 7.18 - Attributes of IDLStruct height_interval_type

Attribute	Notes
start altitude_coordinate_type	
stop altitude_coordinate_type	

7.3.7.27 latitude_coordinate_type

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

7.3.7.28 latitude_interval_type

Type: IDLStruct
Package: Coordinates_and_Positions

Table 7.19 - Attributes of IDLStruct latitude_interval_type

Attribute	Notes
start latitude_coordinate_type	
stop latitude_coordinate_type	

7.3.7.29 longitude_coordinate_type

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

7.3.7.30 longitude_interval_type

Type: IDLStruct
Package: Coordinates_and_Positions

Table 7.20 - Attributes of IDLStruct longitude_interval_type

Attribute	Notes
start longitude_coordinate_type	
stop longitude_coordinate_type	

7.3.7.31 polar_position_type

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

Table 7.21 - Attributes of IDLStruct polar_position_type

Attribute	Notes
azimuth_coordinate azimuth_coordinate_type	
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 sensor provide no range information (e.g. most passive sensors)
origin wgs84_position_type [0..1]	Specifies 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.

7.3.7.32 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 7.22 - Attributes of IDLStruct polar_velocity_type

Attribute	Notes
azimuth_rate azimuth_rate_type	
elevation_rate elevation_rate_type [0..1]	Optional as some sensors provide no elevation information
range_rate range_rate_type [0..1]	Optional as some sensor provide no range information (e.g. most passive sensors)

7.3.7.33 position_accuracy_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.

Table 7.23 - Attributes of IDLUnion position_accuracy_coordinate_type

Attribute	Notes
«idlCase» **cartesian_position_accuracy** cartesian_position_accuracy_type	
«idlCase» **polar_position_accuracy** polar_position_accuracy_type	
«idlCase» **wgs84_position_accuracy** wgs84_position_accuracy_type	

7.3.7.34 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.
case type = coordinate_kind_type

Table 7.24 - Attributes of IDLUnion position_coordinate_type

Attribute	Notes
«idlCase» **cartesian_position** cartesian_position_type	
«idlCase» **polar_position** polar_position_type	
«idlCase» **wgs84_position** wgs84_position_type	

7.3.7.35 range_coordinate_type

Type: IDLTypeDef double
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
Range = 0 .. 1 e7
Resolution = 1
Unit = m

7.3.7.36 range_interval_type

Type: IDLStruct
Package: Coordinates_and_Positions

Table 7.25 - Attributes of IDLStruct range_interval_type

Attribute	Notes
start range_coordinate_type	
stop range_coordinate_type	

7.3.7.37 range_qualification_type

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

Table 7.26 - 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.

7.3.7.38 range_rate_type

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

7.3.7.39 speed_interval_type

Type: IDLStruct

Package: Coordinates_and_Positions
This class represents a range of speeds.

Table 7.27 - Attributes of IDLStruct speed_interval_type

Attribute	Notes
min speed_type	The minimum speed.
max speed_type	The maximum speed.

7.3.7.40 speed_type

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

7.3.7.41 velocity_accuracy_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.

Table 7.28 - Attributes of IDLUnion velocity_accuracy_coordinate_type

Attribute	Notes
«idlCase» **cartesian_velocity_accuracy** cartesian_velocity_accuracy_type	
«idlCase» **polar_velocity_accuracy** polar_velocity_accuracy_type	
«idlCase» **wgs84_velocity_accuracy** wgs84_velocity_accuracy_type	

7.3.7.42 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.
case type = coordinate_kind_type

Table 7.29 - Attributes of IDLUnion velocity_coordinate_type

Attribute	Notes
«idlCase» **cartesian_velocity** cartesian_velocity_type	
«idlCase» **polar_velocity** polar_velocity_type	
«idlCase» **wgs84_velocity** wgs84_velocity_type	

7.3.7.43 wgs84_position_type

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

Table 7.30 - Attributes of IDLStruct wgs84_position_type

Attribute	Notes
altitude_coordinate altitude_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	
longitude_coordinate longitude_coordinate_type	

7.3.7.44 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 7.31 - 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.

7.3.7.45 cartesian_position_accuracy_type

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

Table 7.32 - Attributes of IDLStruct cartesian_position_accuracy_type

Attribute	Notes
x_coordinate_accuracy cartesian_coordinate_type	
y_coordinate_accuracy cartesian_coordinate_type	
z_coordinate_accuracy cartesian_coordinate_type [0..1]	Optional as some sensors are 2D (horizontal plane or no elevation information)

7.3.7.46 cartesian_velocity_accuracy_type

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

Table 7.33 - Attributes of IDLStruct cartesian_velocity_accuracy_type

Attribute	Notes
x_dot_accuracy cartesian_velocity_component_type	
y_dot_accuracy cartesian_velocity_component_type	
z_dot_accuracy cartesian_velocity_component_type [0..1]	Optional as some sensors are 2D (horizontal plane or no elevation information)

7.3.7.47 polar_position_accuracy_type

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

Table 7.34 - Attributes of IDLStruct polar_position_accuracy_type

Attribute	Notes
azimuth_accuracy azimuth_coordinate_type	
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 sensor 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.

7.3.7.48 polar_velocity_accuracy_type

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

Table 7.35 - Attributes of IDLStruct polar_velocity_accuracy_type

Attribute	Notes
azimuth_rate_accuracy azimuth_rate_type	
elevation_rate_accuracy elevation_rate_type [0..1]	Optional as some sensors provide no elevation information
range_rate_accuracy range_rate_type [0..1]	Optional as some sensor provide no range information (e.g. most passive sensors)

7.3.7.49 wgs84_position_accuracy_type

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

Table 7.36 - Attributes of IDLStruct wgs84_position_accuracy_type

Attribute	Notes
altitude_accuracy altitude_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_accuracy latitude_coordinate_type	
longitude_accuracy longitude_coordinate_type	

7.3.7.50 wgs84_velocity_accuracy_type

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

Table 7.37 - Attributes of IDLStruct wgs84_velocity_accuracy_type

Attribute	Notes
course_accuracy course_type	
angle_of_climb_accuracy 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.

Attribute	Notes
speed_accuracy speed_type	

7.3.8 Shape_Model

Parent Package: Common_Types

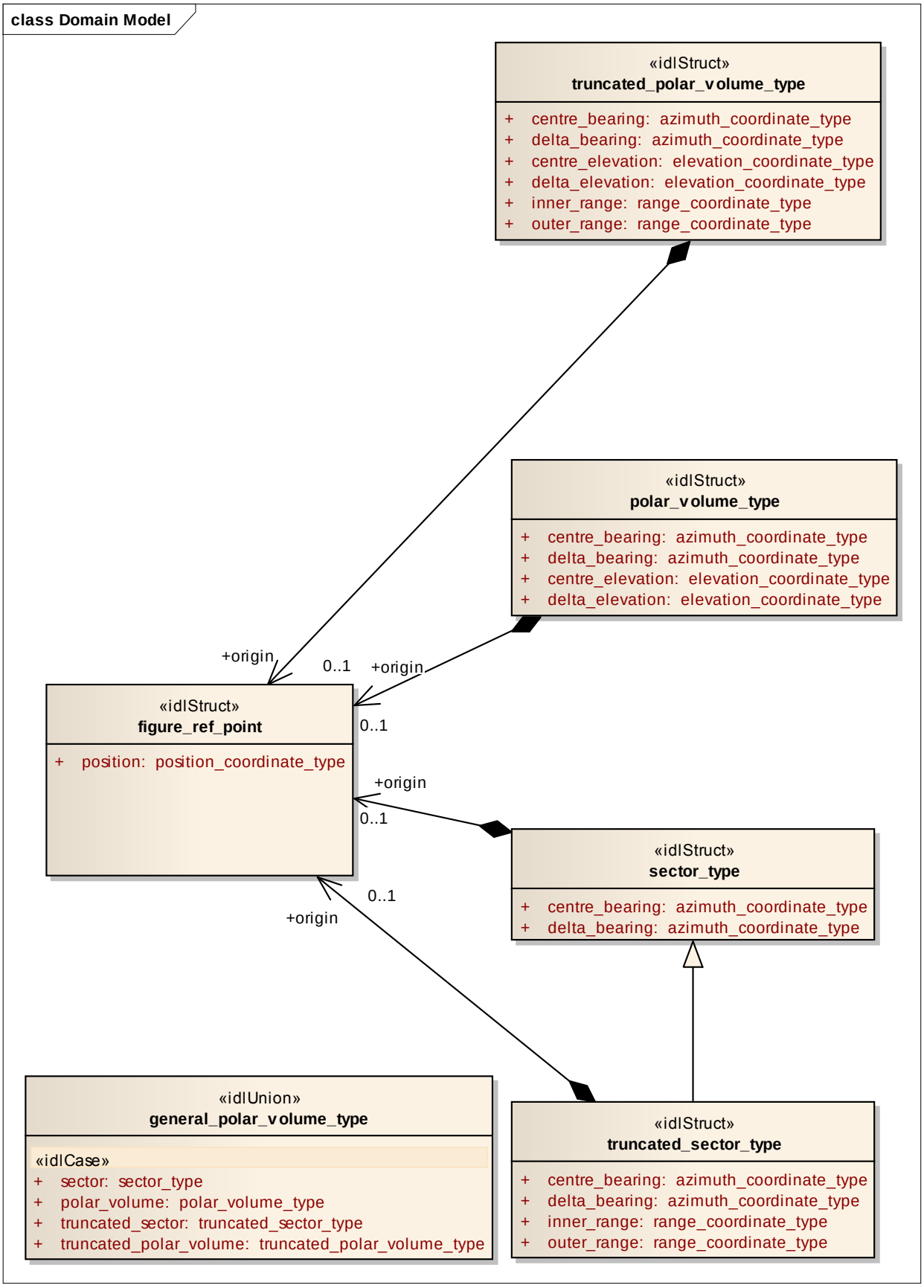
class Domain Model
«idlStruct»
truncated_polar_volume_type
+ centre_bearing: azimuth_coordinate_type
+ delta_bearing: azimuth_coordinate_type
+ centre_elevation: elevation_coordinate_type
+ delta_elevation: elevation_coordinate_type
+ inner_range: range_coordinate_type
+ outer_range: range_coordinate_type
«idlStruct»
polar_volume_type
+ centre_bearing: azimuth_coordinate_type
+ delta_bearing: azimuth_coordinate_type
+ centre_elevation: elevation_coordinate_type
+ delta_elevation: elevation_coordinate_type
+origin
0..1
+origin
«idlStruct»
figure_ref_point
+ position: position_coordinate_type
0..1
+origin
0..1
«idlStruct»
sector_type
+ centre_bearing: azimuth_coordinate_type
+ delta_bearing: azimuth_coordinate_type
0..1
+origin
«idlUnion»
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»
truncated_sector_type
+ centre_bearing: azimuth_coordinate_type
+ delta_bearing: azimuth_coordinate_type
+ inner_range: range_coordinate_type
+ outer_range: range_coordinate_type

Figure 7.28 Domain Model (Logical diagram)

7.3.8.1 figure_ref_point

Type: IDLStruct
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 7.38 - Attributes of IDLStruct figure_ref_point

Attribute	Notes
position position_coordinate_type	

7.3.8.2 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 7.39 - 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.

7.3.8.3 polar_volume_type

Type: IDLStruct
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 7.40 - Attributes of IDLStruct 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.

Attribute	Notes
delta_elevation elevation_coordinate_type	This attribute specifies the elevation delta on each side of a specified centre elevation line.

7.3.8.4 sector_type

Type: IDLStruct
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 7.41 - Attributes of IDLStruct 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.

7.3.8.5 truncated_polar_volume_type

Type: IDLStruct
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 7.42 - Attributes of IDLStruct 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.

7.3.8.6 truncated_sector_type

Type: IDLStruct sector_type
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 7.43 - Attributes of IDLStruct 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.

7.3.9 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.

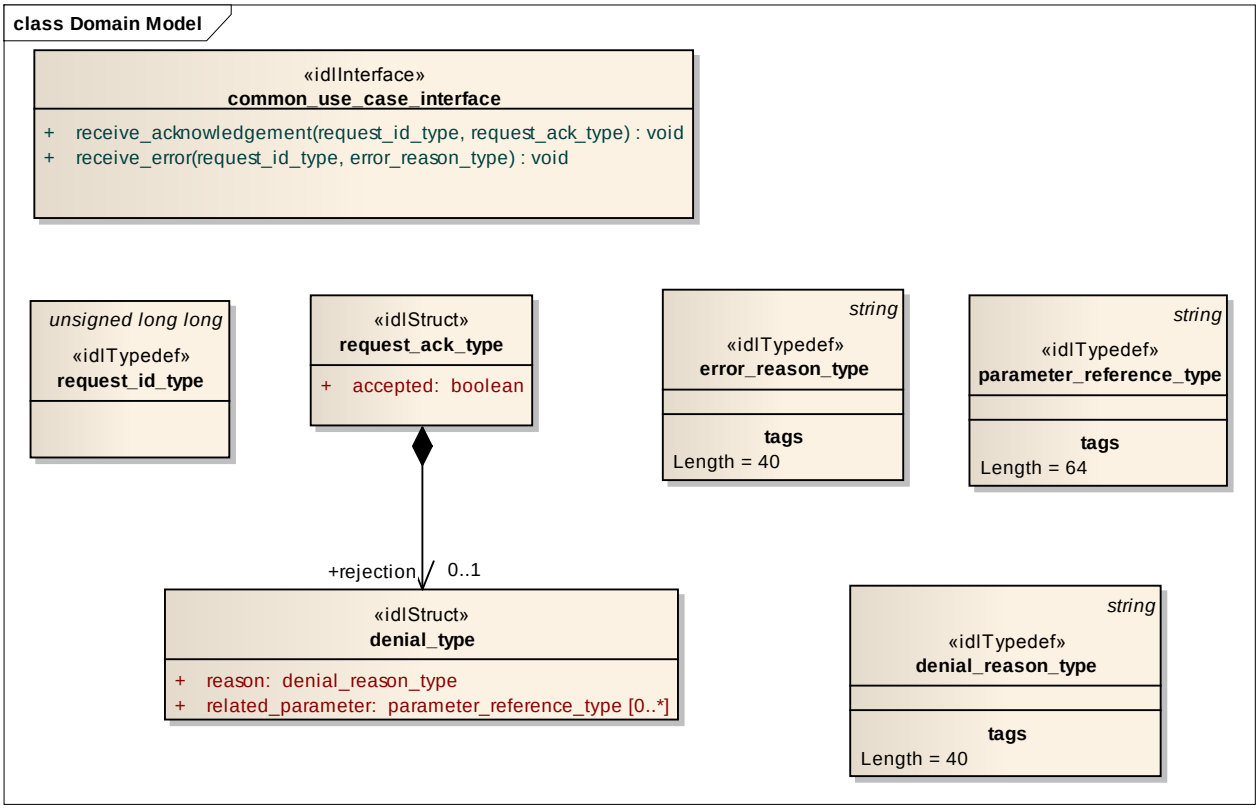

Figure 7.29 Domain Model (Logical diagram)

7.3.9.1 denial_reason_type

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

7.3.9.2 denial_type

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

Table 7.44 - Attributes of IDLStruct 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.

7.3.9.3 error_reason_type

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

7.3.9.4 parameter_reference_type

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

7.3.9.5 request_ack_type

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

Table 7.45 - Attributes of IDLStruct request_ack_type

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

7.3.9.6 request_id_type

Type: IDLTypeDef unsigned long long
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).

7.3.9.7 common_use_case_interface

Type: IDLInterface
Package: Requests

Interface which includes operations common to all CMS interfaces.

Table 7.220 - Methods of IDLInterface 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.	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**

7.4 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.

7.4.1 Encyclopaedic_Support

Parent Package: Subsystem_Domain

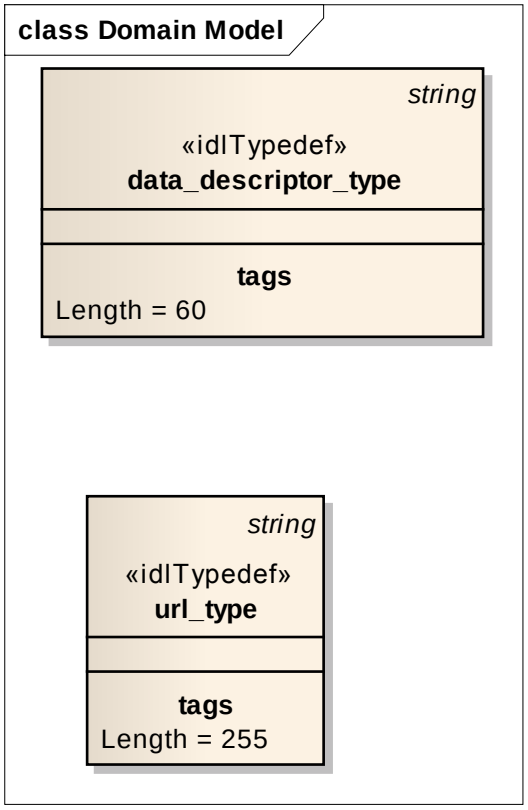

Figure 7.30 Domain Model (Logical diagram)

7.4.1.1 data_descriptor_type

Type: IDLTypeDef string
Package: Encyclopaedic_Support
Standard description of the encyclopaedic data set
Length = 60

7.4.1.2 url_type

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

7.4.2 Extended_Subsystem_Control

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

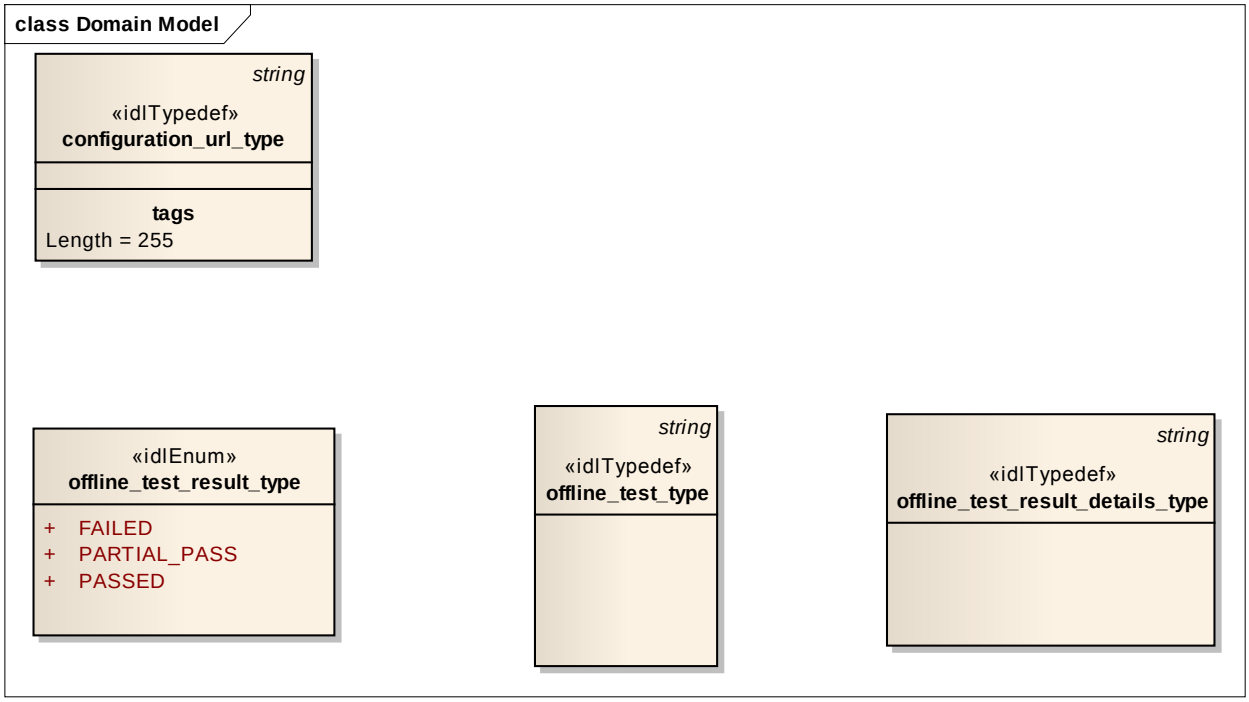

Figure 7.31 Domain Model (Logical diagram)

7.4.2.1 configuration_url_type

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

7.4.2.2 offline_test_result_details_type

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

7.4.2.3 offline_test_result_type

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

Table 7.46 - Attributes of IDLEnum 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.

7.4.2.4 offline_test_type

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

7.4.3 Recording_and_Replay

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

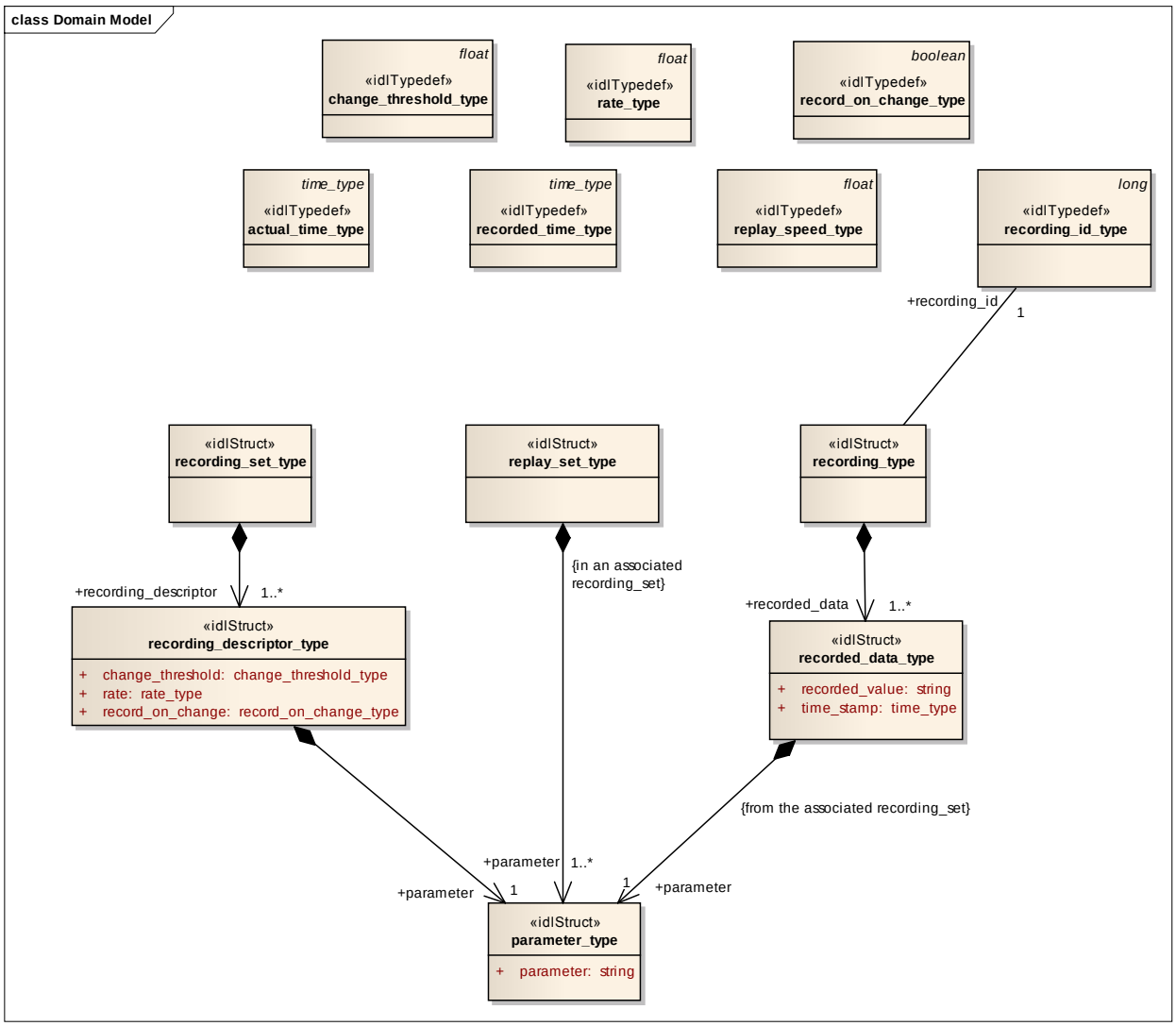

Figure 7.32 Domain Model (Logical diagram)

7.4.3.1 actual_time_type

Type: IDLTypeDef time_type
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.

7.4.3.2 change_threshold_type

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

7.4.3.3 parameter_type

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

Table 7.47 - Attributes of IDLStruct parameter_type

Attribute	Notes
parameter string	

7.4.3.4 rate_type

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

7.4.3.5 record_on_change_type

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

7.4.3.6 recorded_data_type

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

Table 7.48 - Attributes of IDLStruct recorded_data_type

Attribute	Notes
recorded_value string	This needs to reference allowable values defined by the possible recording parameters - see 'recording parameters'.
time_stamp time_type	

7.4.3.7 recorded_time_type

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

7.4.3.8 recording_descriptor_type

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

Table 7.49 - Attributes of IDLStruct 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.
record_on_change record_on_change_type	Indicates whether to record all changes greater than the change threshold or record at the specified rate.

7.4.3.9 recording_id_type

Type: IDLTypeDef long
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.

7.4.3.10 recording_set_type

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

7.4.3.11 recording_type

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

7.4.3.12 replay_set_type

Type: IDLStruct
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.

7.4.3.13 replay_speed_type

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

7.4.4 Simulation_Support

Parent Package: Subsystem_Domain

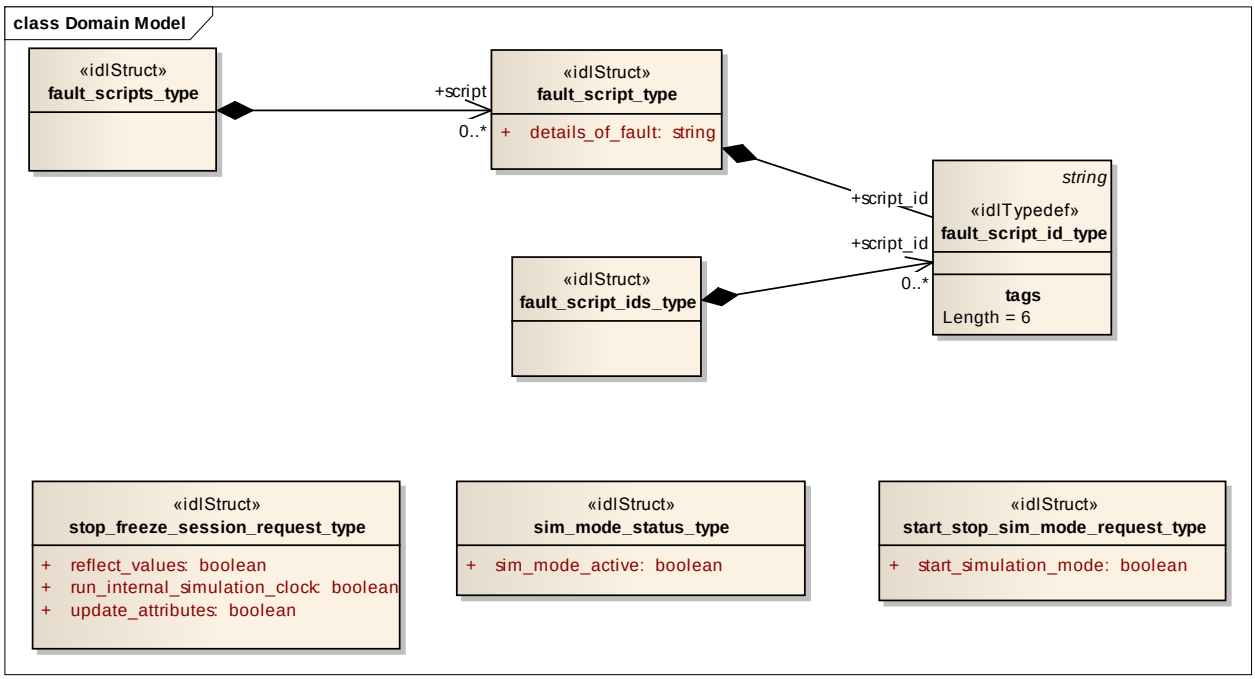

Figure 7.33 Domain Model (Logical diagram)

7.4.4.1 fault_script_id_type

Type: IDLTypeDef string
Package: Simulation_Support
Identifies a single fault script.
Length = 6

7.4.4.2 fault_script_ids_type

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

7.4.4.3 fault_script_type

Type: IDLStruct
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 7.50 - Attributes of IDLStruct fault_script_type

Attribute	Notes
details_of_fault string	A description of the fault, such as is interpretable during the simulation

7.4.4.4 fault_scripts_type

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

7.4.4.5 sim_mode_status_type

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

Table 7.51 - Attributes of IDLStruct sim_mode_status_type

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

7.4.4.6 start_stop_sim_mode_request_type

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

Table 7.52 - Attributes of IDLStruct start_stop_sim_mode_request_type

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

7.4.4.7 stop_freeze_session_request_type

Type: IDLStruct
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 7.53 - Attributes of IDLStruct 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.

7.4.5 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.

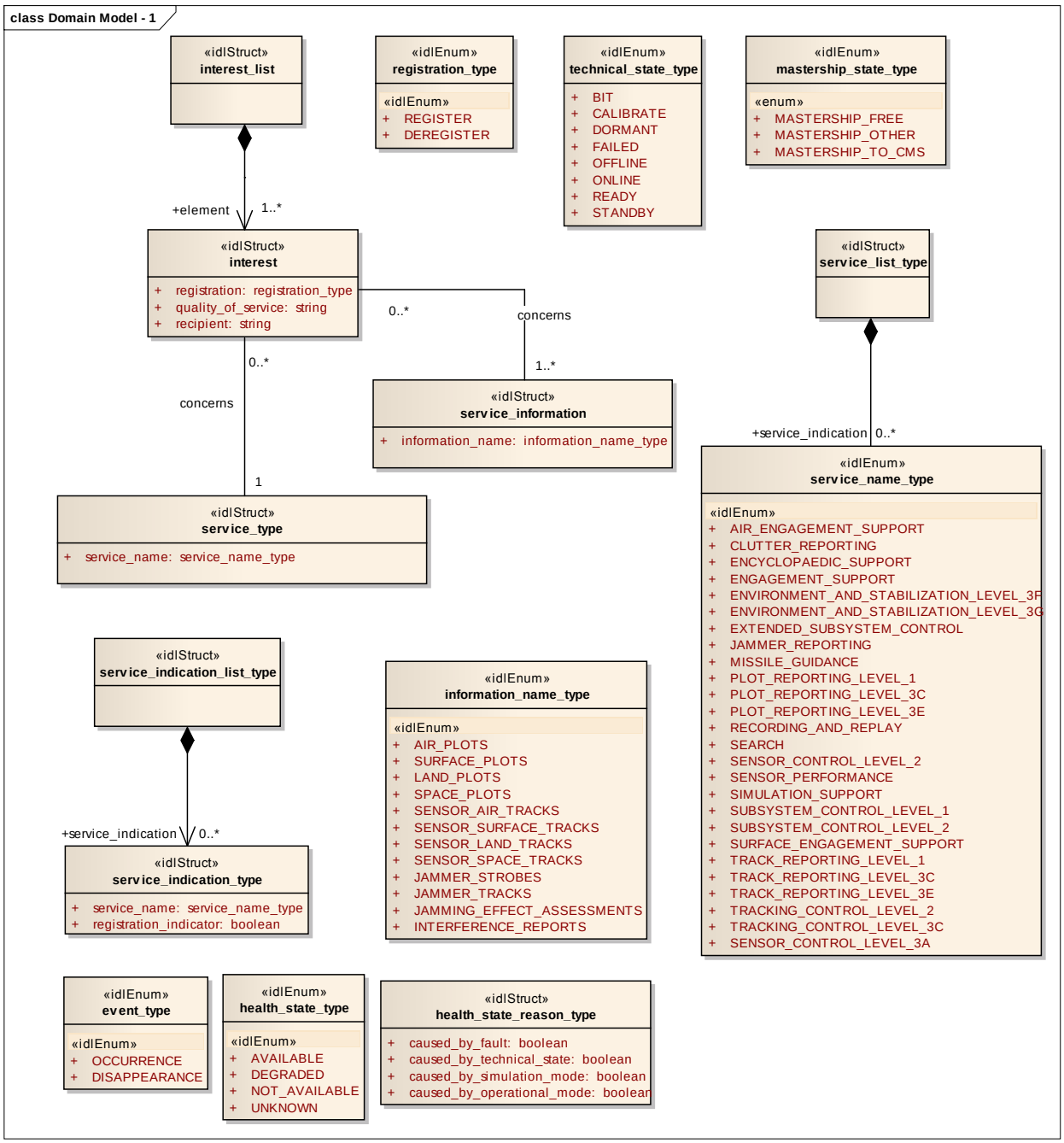

Figure 7.34 Domain Model - 1 (Logical diagram)

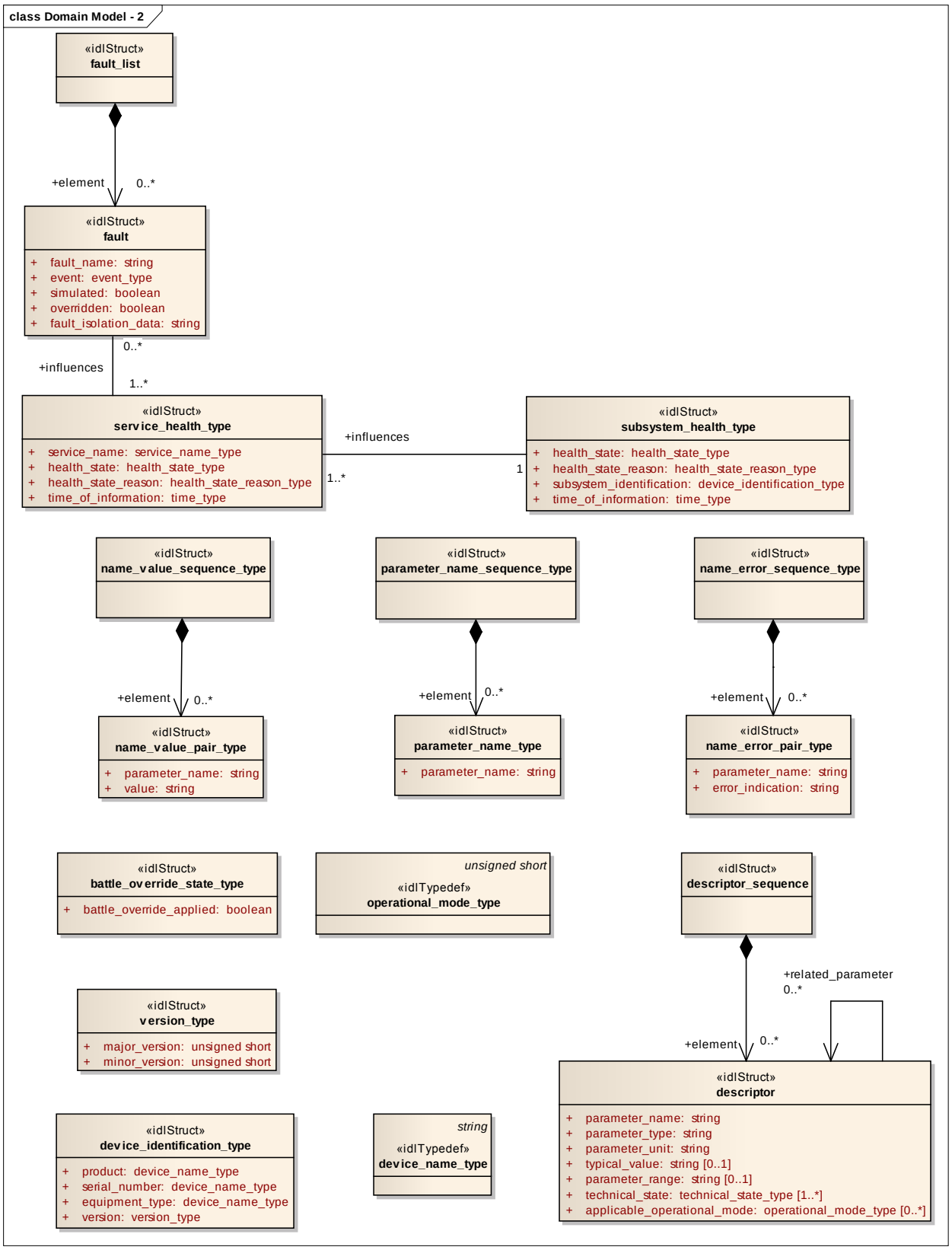

Figure 7.35 Domain Model - 2 (Logical diagram)

7.4.5.1 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 7.54 - Attributes of IDLEnum service_name_type

Attribute	Notes
«idlEnum» **AIR_ENGAGEMENT_SUPPORT**	
«idlEnum» **CLUTTER_REPORTING**	
«idlEnum» **ENCYCLOPAEDIC_SUPPORT**	
«idlEnum» **ENGAGEMENT_SUPPORT**	
«idlEnum» **ENVIRONMENT_AND_STABILIZATION_LEVEL_3F**	
«idlEnum» **ENVIRONMENT_AND_STABILIZATION_LEVEL_3G**	
«idlEnum» **EXTENDED_SUBSYSTEM_CONTROL**	
«idlEnum» **JAMMER_REPORTING**	
«idlEnum» **MISSILE_GUIDANCE**	
«idlEnum» **PLOT_REPORTING_LEVEL_1**	
«idlEnum» **PLOT_REPORTING_LEVEL_3C**	
«idlEnum» **PLOT_REPORTING_LEVEL_3E**	
«idlEnum» **RECORDING_AND_REPLAY**	
«idlEnum» **SEARCH**	
«idlEnum» **SENSOR_CONTROL_LEVEL_2**	
«idlEnum» **SENSOR_PERFORMANCE**	
«idlEnum» **SIMULATION_SUPPORT**	
«idlEnum» **SUBSYSTEM_CONTROL_LEVEL_1**	
«idlEnum» **SUBSYSTEM_CONTROL_LEVEL_2**	
«idlEnum» **SURFACE_ENGAGEMENT_SUPPORT**	
«idlEnum» **TRACK_REPORTING_LEVEL_1**	
«idlEnum» **TRACK_REPORTING_LEVEL_3C**	
«idlEnum» **TRACK_REPORTING_LEVEL_3E**	
«idlEnum» **TRACKING_CONTROL_LEVEL_2**	
«idlEnum» **TRACKING_CONTROL_LEVEL_3C**	
«idlEnum» **SENSOR_CONTROL_LEVEL_3A**	

7.4.5.2 battle_override_state_type

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

Table 7.55 - Attributes of IDLStruct battle_override_state_type

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

7.4.5.3 descriptor

Type: IDLStruct
Package: Subsystem_Control
Type for parameter descriptors.

Table 7.56 - Attributes of IDLStruct descriptor

Attribute	Notes
parameter_name string	parameter_name values are unique within the scope of a subsystem.
parameter_type string	
parameter_unit string	
typical_value string [0..1]	*optional*
parameter_range string [0..1]	*optional*
technical_state technical_state_type [1..*]	Technical state(s) in which this parameter may be modified.
applicable_operational_mode operational_mode_type [0..*]	

7.4.5.4 descriptor_sequence

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

7.4.5.5 device_identification_type

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

Table 7.57 - 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.
equipment_type device_name_type	This describes the general type of the equipment. Example: Air Surveillance Radar
version version_type	Version of the device.

7.4.5.6 device_name_type

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

7.4.5.7 event_type

Type: IDLEnum
Package: Subsystem_Control
Type of event

Table 7.58 - Attributes of IDLEnum event_type

Attribute	Notes
«idlEnum» **OCCURRENCE**	
«idlEnum» **DISAPPEARANCE**	

7.4.5.8 fault

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

Table 7.59 - Attributes of IDLStruct fault

Attribute	Notes
fault_name string	
event event_type	
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.

7.4.5.9 fault_list

Type: IDLStruct
Package: Subsystem_Control
A list of faults

7.4.5.10 health_state_reason_type

Type: IDLStruct
Package: Subsystem_Control
Reason for the health state

Table 7.60 - Attributes of IDLStruct health_state_reason_type

Attribute	Notes
caused_by_fault boolean	
caused_by_technical_state boolean	
caused_by_simulation_mode boolean	
caused_by_operational_mode boolean	

7.4.5.11 health_state_type

Type: IDLEnum
Package: Subsystem_Control
Encapsulation of health state

Table 7.61 - 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**	Indicates that the subsystem may not determine the health state of the service or subsystem (e.g. because BIT is not running).

7.4.5.12 information_name_type

Type: IDLEnum
Package: Subsystem_Control
Name of information

Table 7.62 - Attributes of IDLEnum information_name_type

Attribute	Notes
«idlEnum» **AIR_PLOTS**	
«idlEnum» **SURFACE_PLOTS**	
«idlEnum» **LAND_PLOTS**	
«idlEnum» **SPACE_PLOTS**	
«idlEnum» **SENSOR_AIR_TRACKS**	
«idlEnum» **SENSOR_SURFACE_TRACKS**	
«idlEnum» **SENSOR_LAND_TRACKS**	
«idlEnum» **SENSOR_SPACE_TRACKS**	
«idlEnum» **JAMMER_STROBES**	
«idlEnum» **JAMMER_TRACKS**	
«idlEnum» **JAMMING_EFFECT_ASSESSMENTS**	
«idlEnum» **INTERFERENCE_REPORTS**	

7.4.5.13 interest

Type: IDLStruct
Package: Subsystem_Control
Encapsulation of interest in service

Table 7.63 - Attributes of IDLStruct interest

Attribute	Notes
registration registration_type	
quality_of_service string	* optional *
recipient string	* optional *

7.4.5.14 interest_list

Type: IDLStruct
Package: Subsystem_Control
A list of interest

7.4.5.15 mastership_state_type

Type: IDLEnum
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 7.64 - Attributes of IDLEnum 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.

7.4.5.16 parameter_name_type

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

Table 7.65 - Attributes of IDLStruct parameter_name_type

Attribute	Notes

Attribute	Notes
parameter_name string	parameter_name values are unique within the scope of a subsystem.

7.4.5.17 name_error_pair_type

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

Table 7.66 - Attributes of IDLStruct name_error_pair_type

Attribute	Notes
parameter_name string	parameter_name values are unique within the scope of a subsystem.
error_indication string	

7.4.5.18 name_error_sequence_type

Type: IDLStruct
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).

7.4.5.19 parameter_name_sequence_type

Type: IDLStruct
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.

7.4.5.20 name_value_pair_type

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

Table 7.67 - Attributes of IDLStruct name_value_pair_type

Attribute	Notes
parameter_name string	parameter_name values are unique within the scope of a subsystem.
value string	

7.4.5.21 name_value_sequence_type

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

7.4.5.22 operational_mode_type

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

7.4.5.23 parameter_value_response_type

Type: IDLStruct
Package: Subsystem_Control

Response type for retrieving and modifying sequences of parameters.

Table 7.68 - Attributes of IDLStruct parameter_value_response_type

Attribute	Notes
request_id long	

7.4.5.24 registration_type

Type: IDLEnum
Package: Subsystem_Control
Type of registration

Table 7.69 - Attributes of IDLEnum registration_type

Attribute	Notes
«idlEnum» **REGISTER**	
«idlEnum» **DEREGISTER**	

7.4.5.25 service_type

Type: IDLStruct
Package: Subsystem_Control
Type of service

Table 7.70 - Attributes of IDLStruct service_type

Attribute	Notes
service_name service_name_type	Only registrable services are allowed

7.4.5.26 service_health_type

Type: IDLStruct
Package: Subsystem_Control
Health of service

Table 7.71 - Attributes of IDLStruct service_health_type

Attribute	Notes
service_name service_name_type	
health_state health_state_type	
health_state_reason health_state_reason_type	
time_of_information time_type	

7.4.5.27 service_indication_list_type

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

7.4.5.28 service_indication_type

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

Table 7.72 - 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.

7.4.5.29 service_information

Type: IDLStruct
Package: Subsystem_Control
Information about a service

Table 7.73 - Attributes of IDLStruct service_information

Attribute	Notes
information_name information_name_type	

7.4.5.30 service_list_type

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

7.4.5.31 subsystem_health_type

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

Table 7.74 - 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	
time_of_information time_type	

7.4.5.32 technical_state_type

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

Table 7.75 - Attributes of IDLEnum 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.

7.4.5.33 version_type

Type: IDLStruct
Package: Subsystem_Control
Version of the equipment

Table 7.76 - Attributes of IDLStruct version_type

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

7.4.5.34 Initial

Type: Initial State
Package: Subsystem_Control

7.5 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.

7.5.1 Clutter_Reporting

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

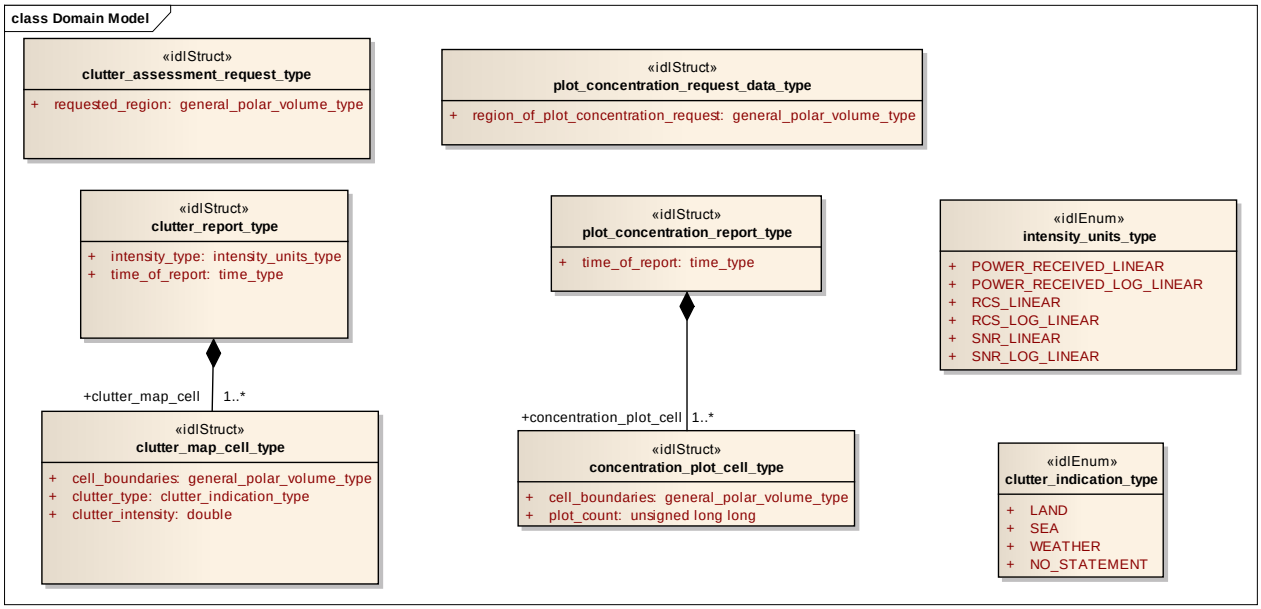

Figure 7.36 Domain Model (Logical diagram)

7.5.1.1 clutter_assessment_request_type

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

Table 7.77 - Attributes of IDLStruct clutter_assessment_request_type

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

7.5.1.2 clutter_indication_type

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

Table 7.78 - Attributes of IDLEnum clutter_indication_type

Attribute	Notes
LAND	
SEA	
WEATHER	
NO_STATEMENT	

7.5.1.3 clutter_map_cell_type

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

Table 7.79 - 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.

7.5.1.4 clutter_report_type

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

Table 7.80 - 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.

7.5.1.5 concentration_plot_cell_type

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

Table 7.81 - Attributes of IDLStruct 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 long	The number of plots generated within the cell.

7.5.1.6 intensity_units_type

Type: IDLEnum
Package: Clutter_Reporting
Units of the clutter intensity

Table 7.82 - Attributes of IDLEnum intensity_units_type

Attribute	Notes
POWER_RECEIVED_LINEAR	
POWER_RECEIVED_LOG_LINEAR	(e.g. dBm, dBW)
RCS_LINEAR	square meters
RCS_LOG_LINEAR	
SNR_LINEAR	
SNR_LOG_LINEAR	

7.5.1.7 plot_concentration_report_type

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

Table 7.83 - Attributes of IDLStruct plot_concentration_report_type

Attribute	Notes
time_of_report time_type	Time of the plot concentration report.

7.5.1.8 plot_concentration_request_data_type

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

Table 7.84 - 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.

7.5.2 Plot_Reporting

Parent Package: Sensor_Domain

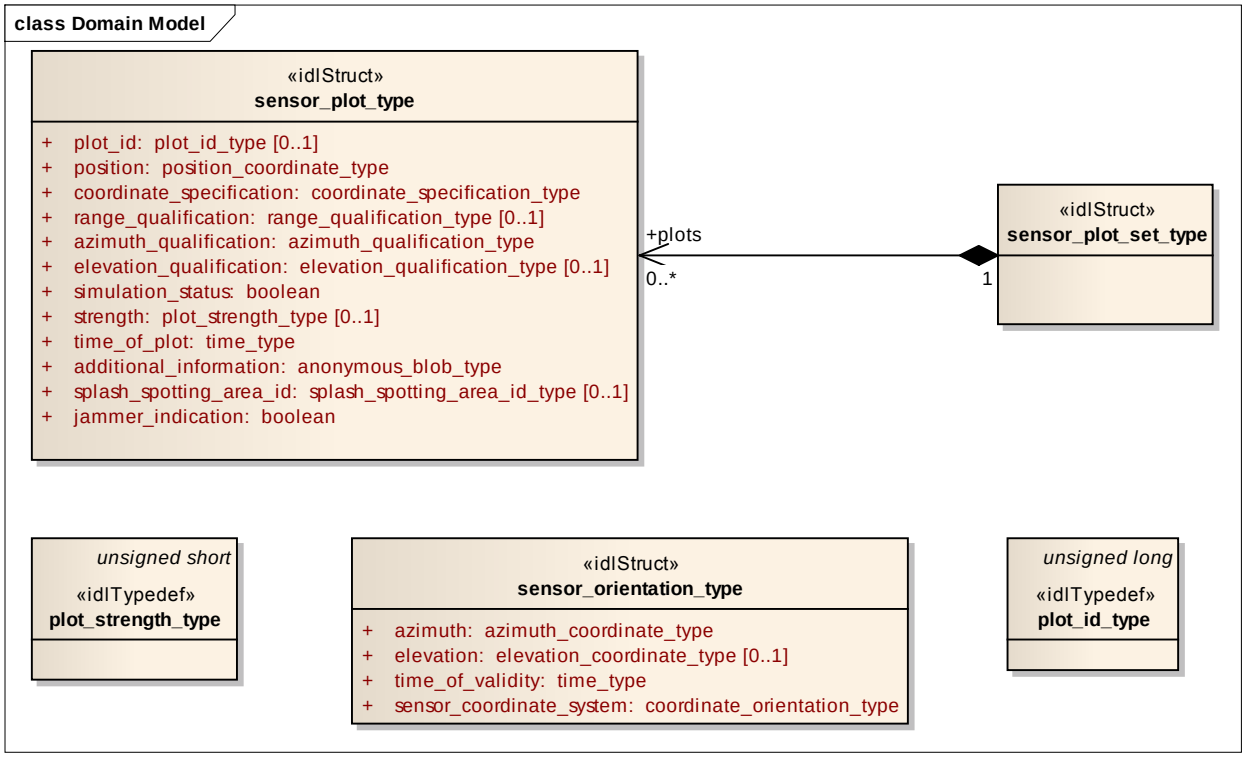

Figure 7.37 Domain Model (Logical diagram)

7.5.2.1 plot_id_type

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

7.5.2.2 plot_strength_type

Type: IDLTypeDef unsigned short
Package: Plot_Reporting
Strength of the plot. The precise semantics of this type are sensor subsystem specific, but a typical interpretation is as a signal to noise ratio in dB.

7.5.2.3 sensor_plot_set_type

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

7.5.2.4 sensor_plot_type

Type: IDLStruct
Package: Plot_Reporting
One plot from a sensor.

The additional_information attribute is used for characteristics of the plot that are specific to certain sensors, and therefore not in the general plot type, for example MTI or range rate.

Table 7.85 - Attributes of IDLStruct sensor_plot_type

Attribute	Notes
plot_id plot_id_type [0..1]	A unique identifier for the plot within the scope of the sensor. This attribute is optional as not all sensors need to provide such an identifier for each plot.
position position_coordinate_type	The position of the plot. This is the mean, central position. Note the qualification attributes, which give information on accuracy and spread estimates.
coordinate_specification coordinate_specification_type	This attribute defines the characteristics of the coordinate system used
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.
simulation_status boolean	If true, the plot is simulated. See also simulation support services within this standard.
strength plot_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.
time_of_plot time_type	The time at which the plot was measured.
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.

7.5.2.5 sensor_orientation_type

Type: IDLStruct

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 7.86 - Attributes of IDLStruct 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)
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.
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

7.5.3 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.

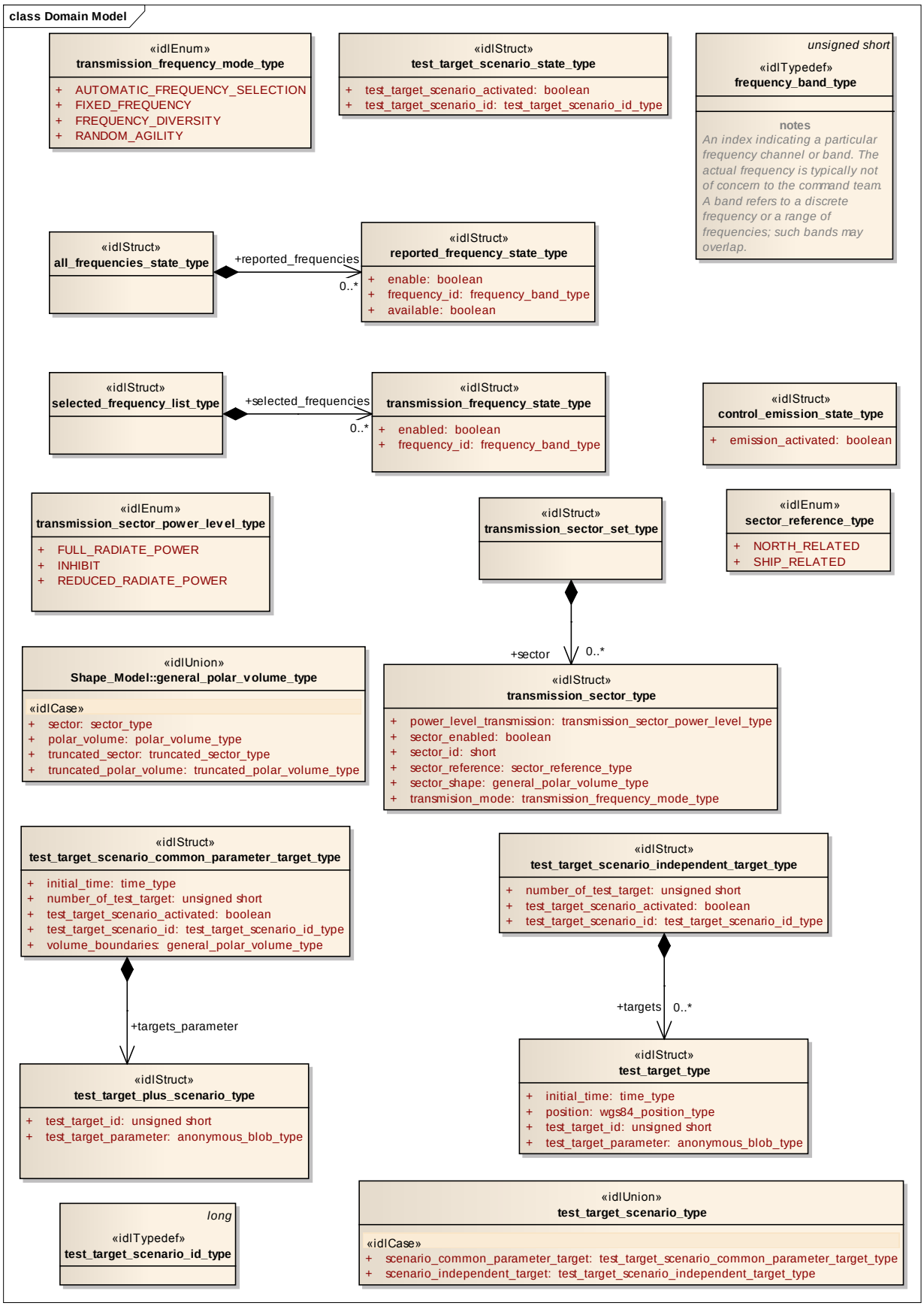
class Domain Model
«idlEnum»
transmission_frequency_mode_type
+ AUTOMATIC_FREQUENCY_SELECTION
+ FIXED_FREQUENCY
+ FREQUENCY_DIVERSITY
+ RANDOM_AGILITY
«idlStruct»
test_target_scenario_state_type
+ test_target_scenario_activated: boolean
+ test_target_scenario_id: test_target_scenario_id_type
unsigned short
«idlTypedef»
frequency_band_type
notes
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
0..*
«idlStruct»
reported_frequency_state_type
+ enable: boolean
+ frequency_id: frequency_band_type
+ available: boolean
«idlStruct»
selected_frequency_list_type
+selected_frequencies
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
+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
+targets_parameter
+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 7.38 Domain Model (Logical diagram)

7.5.3.1 selected_frequency_list_type

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

7.5.3.2 transmission_frequency_state_type

Type: IDLStruct
Package: Sensor_Control
State of frequency transmission

Table 7.87 - Attributes of IDLStruct 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.

7.5.3.3 all_frequencies_state_type

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

7.5.3.4 reported_frequency_state_type

Type: IDLStruct
Package: Sensor_Control
reported frequency state

Table 7.88 - 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.
available boolean	Indicates whether a transmission frequency is available or not available.

7.5.3.5 frequency_band_type

Type: IDLTypeDef unsigned short
Package: Sensor_Control
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.

7.5.3.6 transmission_frequency_mode_type

Type: IDLEnum
Package: Sensor_Control
The mode

Table 7.89 - Attributes of IDLEnum transmission_frequency_mode_type

Attribute	Notes
AUTOMATIC_FREQUENCY_SELECTION	The sensor always uses the same pre-selected frequency

Attribute	Notes
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.

7.5.3.7 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.

7.5.3.8 transmission_sector_type

Type: IDLStruct
Package: Sensor_Control
Sector for transmission

Table 7.90 - Attributes of IDLStruct 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.
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

7.5.3.9 transmission_sector_power_level_type

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

Table 7.91 - Attributes of IDLEnum transmission_sector_power_level_type

Attribute	Notes
FULL_RADIATE_POWER	
INHIBIT	
REDUCED_RADIATE_POWER	

7.5.3.10 sector_reference_type

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

Table 7.92 - Attributes of IDLEnum sector_reference_type

Attribute	Notes
NORTH_RELATED	
SHIP_RELATED	

7.5.3.11 control_emission_state_type

Type: IDLStruct
Package: Sensor_Control
Emission state

Table 7.93 - Attributes of IDLStruct control_emission_state_type

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

7.5.3.12 test_target_scenario_type

Type: IDLUnion
Package: Sensor_Control
Scenario for test targets

Table 7.94 - 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.

7.5.3.13 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 7.95 - 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.

7.5.3.14 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 7.96 - 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.
test_target_scenario_id test_target_scenario_id_type	A unique identifier for the test target scenario.

Attribute	Notes
volume_boundaries general_polar_volume_type	This indicates the area/volume boundaries where the test targets are distributed.

7.5.3.15 test_target_type

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

Table 7.97 - 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).

7.5.3.16 test_target_plus_scenario_type

Type: IDLStruct
Package: Sensor_Control
Test target with its scenario

Table 7.98 - Attributes of IDLStruct 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).

7.5.3.17 test_target_scenario_id_type

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

7.5.3.18 test_target_scenario_state_type

Type: IDLStruct
Package: Sensor_Control
scenario state

Table 7.99 - Attributes of IDLStruct 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.

7.5.4 Sensor_Performance

Parent Package: Sensor_Domain

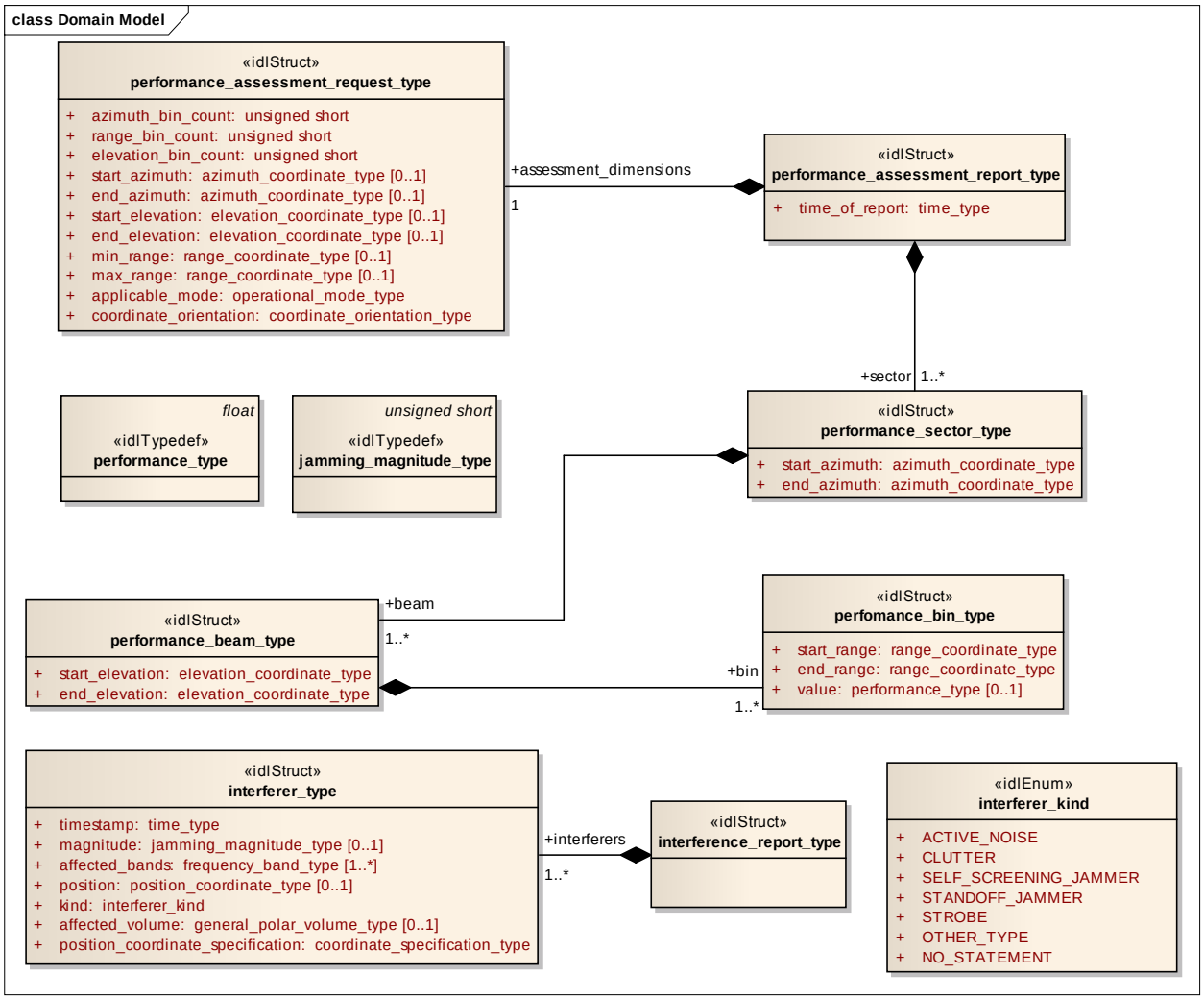

Figure 7.39 Domain Model (Logical diagram)

7.5.4.1 interference_report_type

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

7.5.4.2 interferer_kind

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

Table 7.100 - Attributes of IDLEnum interferer_kind

Attribute	Notes
ACTIVE_NOISE	Interference from active noise.
CLUTTER	Interference from clutter.

Attribute	Notes
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.

7.5.4.3 interferer_type

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

Table 7.101 - Attributes of IDLStruct 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.
position_coordinate_specification coordinate_specification_type	Specifies the coordinate system used to define the interferer.

7.5.4.4 jamming_magnitude_type

Type: IDLTypeDef unsigned short
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.

7.5.4.5 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 7.102 - 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.
value performance_type [0..1]	The assessed level of performance. If no value present, there is no performance data available for this bin.

7.5.4.6 performance_assessment_report_type

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

Table 7.103 - Attributes of IDLStruct performance_assessment_report_type

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

7.5.4.7 performance_assessment_request_type

Type: IDLStruct
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 7.104 - Attributes of IDLStruct performance_assessment_request_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.
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.
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.

7.5.4.8 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 7.105 - 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).

7.5.4.9 performance_sector_type

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

Table 7.106 - Attributes of IDLStruct 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).

7.5.4.10 performance_type

Type: IDLTypeDef float
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..

7.5.5 Track_Reporting

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

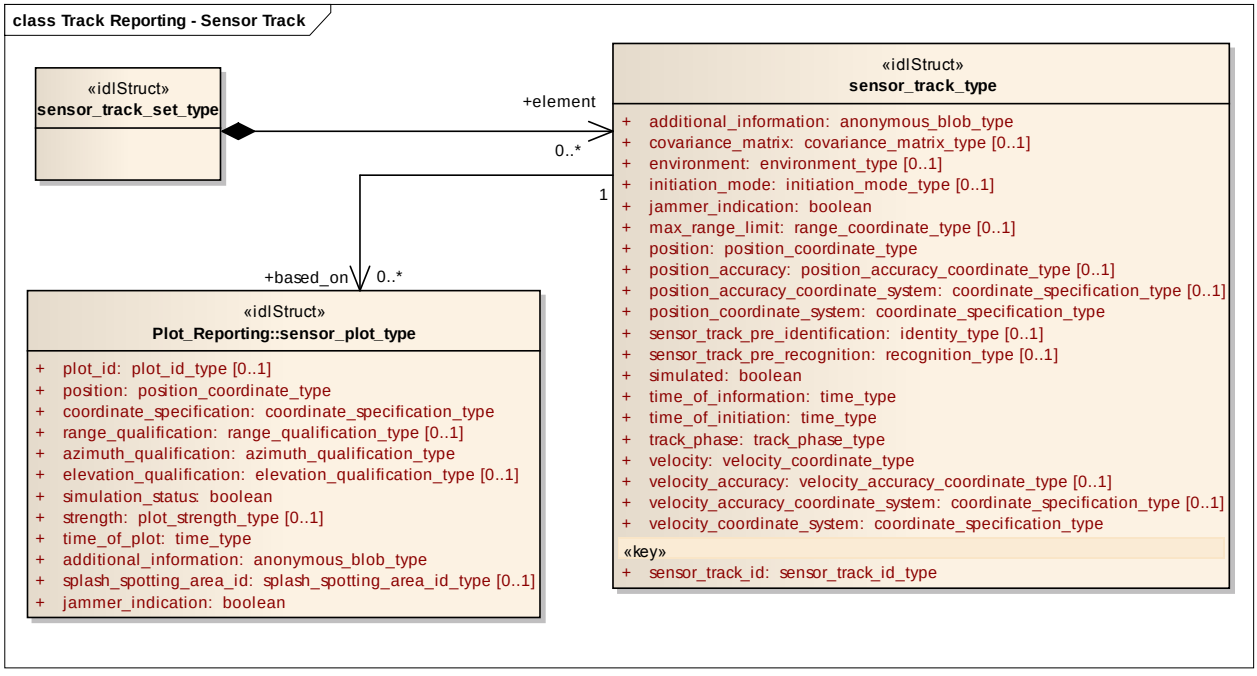

Figure 7.40 Track Reporting - Sensor Track (Logical diagram)

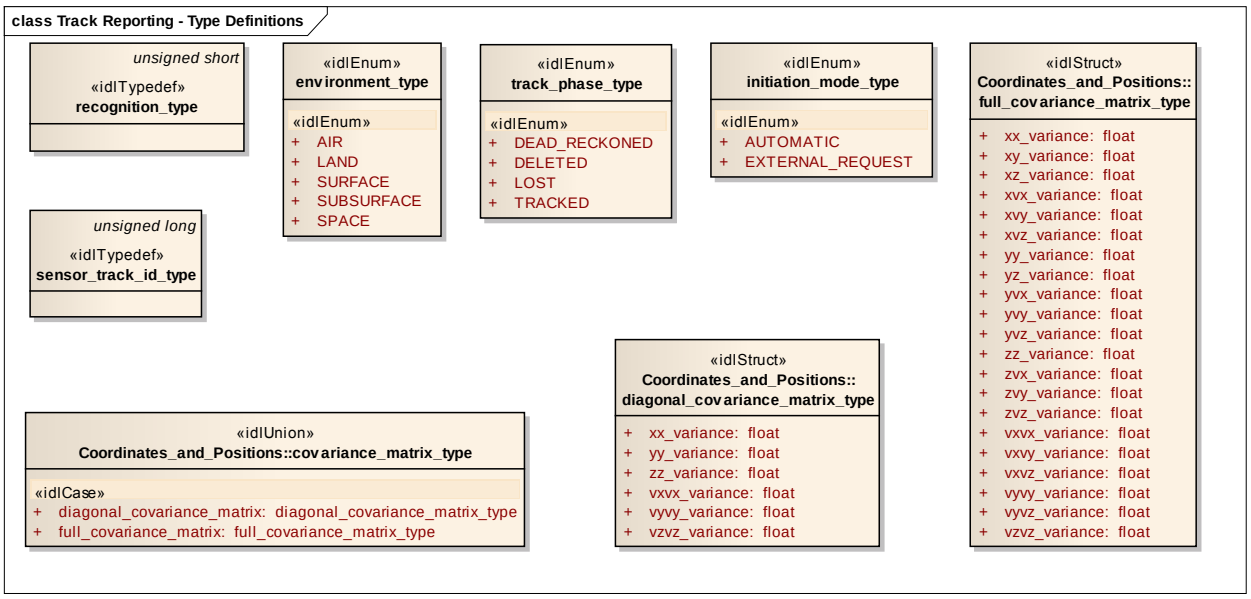

Figure 7.41 Track Reporting - Type Definitions (Logical diagram)

7.5.5.1 sensor_track_id_type

Type: IDLTypeDef unsigned long
Package: Track_Reporting
Sensor Track Identification

7.5.5.2 environment_type

Type: IDLEnum
Package: Track_Reporting
The sensor tracking environment

Table 7.107 - Attributes of IDLEnum environment_type

Attribute	Notes
«idlEnum» **AIR**	
«idlEnum» **LAND**	
«idlEnum» **SURFACE**	
«idlEnum» **SUBSURFACE**	
«idlEnum» **SPACE**	

7.5.5.3 initiation_mode_type

Type: IDLEnum
Package: Track_Reporting
Type of track initiation

Table 7.108 - Attributes of IDLEnum initiation_mode_type

Attribute	Notes
«idlEnum» **AUTOMATIC**	Automatic track initiation mode
«idlEnum» **EXTERNAL_REQUEST**	Track initation on external request (e.g. from CMS)

7.5.5.4 recognition_type

Type: IDLTypeDef unsigned short

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.

7.5.5.5 sensor_track_type

Type: IDLStruct
Package: Track_Reporting
Encapsulation of a sensor track

Table 7.109 - 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 initiatied)
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.
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	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.

Attribute	Notes
velocity_coordinate_system coordinate_specification_type	The coordinate system chosen by the sensor.

7.5.5.6 sensor_track_set_type

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

7.5.5.7 track_phase_type

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

Table 7.110 - Attributes of IDLEnum track_phase_type

Attribute	Notes
«idlEnum» **DEAD_RECKONED**	Track provided based on extrapolated position (dead-reckoned)
«idlEnum» **DELETED**	Track has been deleted.
«idlEnum» **LOST**	Track has been lost
«idlEnum» **TRACKED**	Regular update of new and existing track

7.5.6 Tracking_Control

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

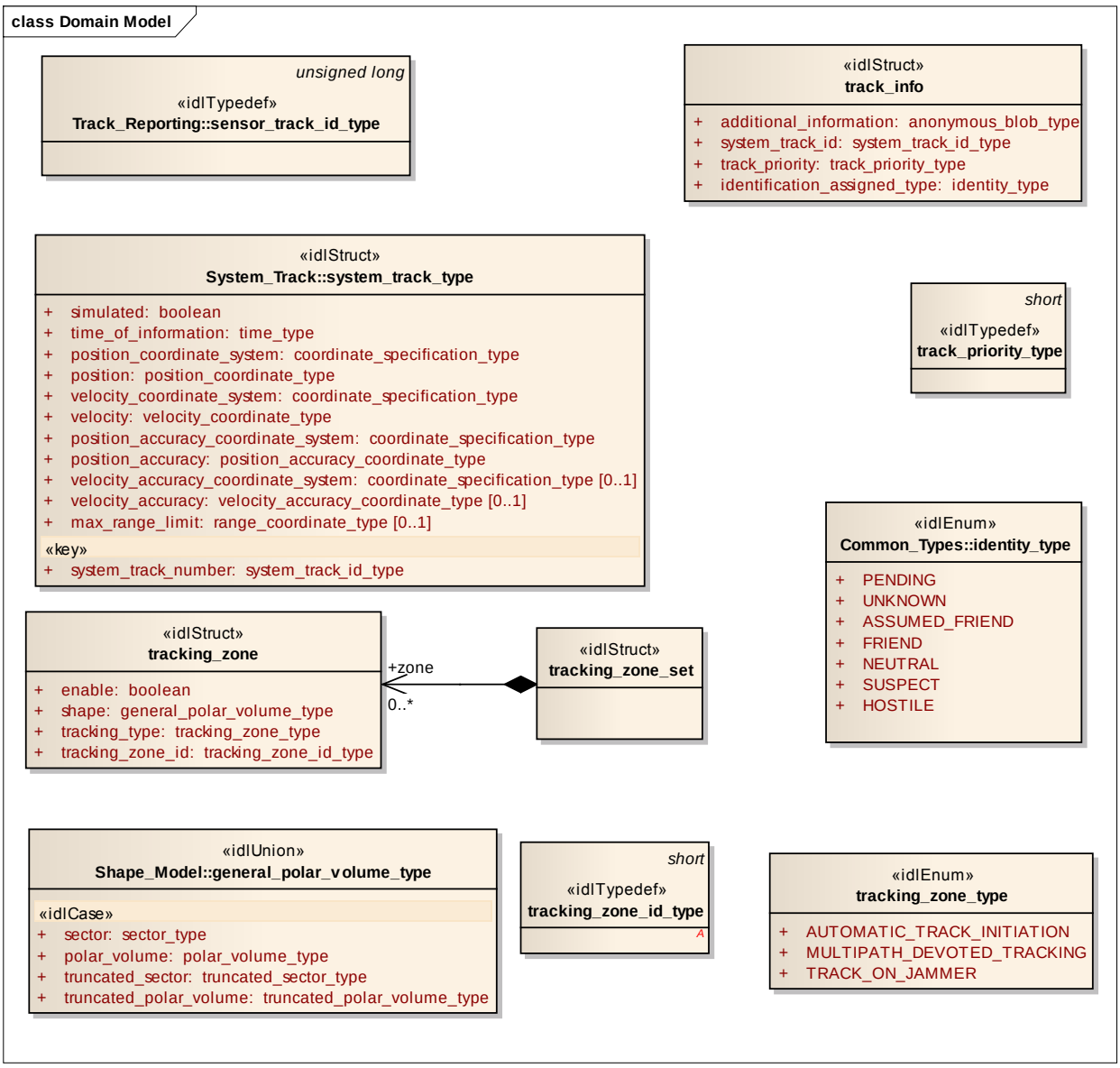

Figure 7.42 Domain Model (Logical diagram)

7.5.6.1 track_info

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

Table 7.111 - Attributes of IDLStruct track_info

Attribute	Notes

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	
track_priority track_priority_type	
identification_assigned_type identity_type	

7.5.6.2 track_priority_type

Type: IDLTypeDef short
Package: Tracking_Control
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).

Example of values:
1 Track While Scan (TWS)
2 Low Priority Target (LPT)
3 High Priority Target (HPT)

7.5.6.3 tracking_zone_set

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

7.5.6.4 tracking_zone

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

Table 7.112 - Attributes of IDLStruct tracking_zone

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_type	This indicates the tracking zone type.
tracking_zone_id tracking_zone_id_type	A unique identifier for the tracking zone.

7.5.6.5 tracking_zone_type

Type: IDLEnum
Package: Tracking_Control
Identifies the type of a tracking zone.

Table 7.113 - Attributes of IDLEnum tracking_zone_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.).

Attribute	Notes
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.

7.5.6.6 tracking_zone_id_type

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

7.6 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.

7.6.1 Air_Engagement_Support

Parent Package: Radar_Domain

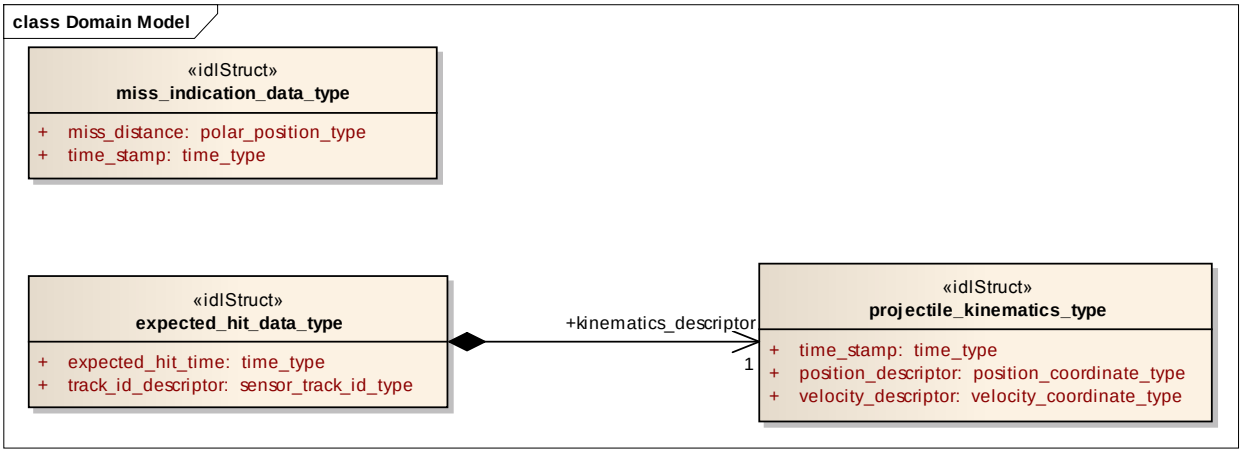

Figure 7.43 Domain Model (Logical diagram)

7.6.1.1 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 7.114 - Attributes of IDLStruct expected_hit_data_type

Attribute	Notes

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.

7.6.1.2 miss_indication_data_type

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

Table 7.115 - 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.

7.6.1.3 projectile_kinematics_type

Type: IDLStruct
Package: Air_Engagement_Support
Identifies the kinematics of the projectile that is expected to hit the target.

Table 7.116 - 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.

7.6.2 Engagement_Support

Parent Package: Radar_Domain

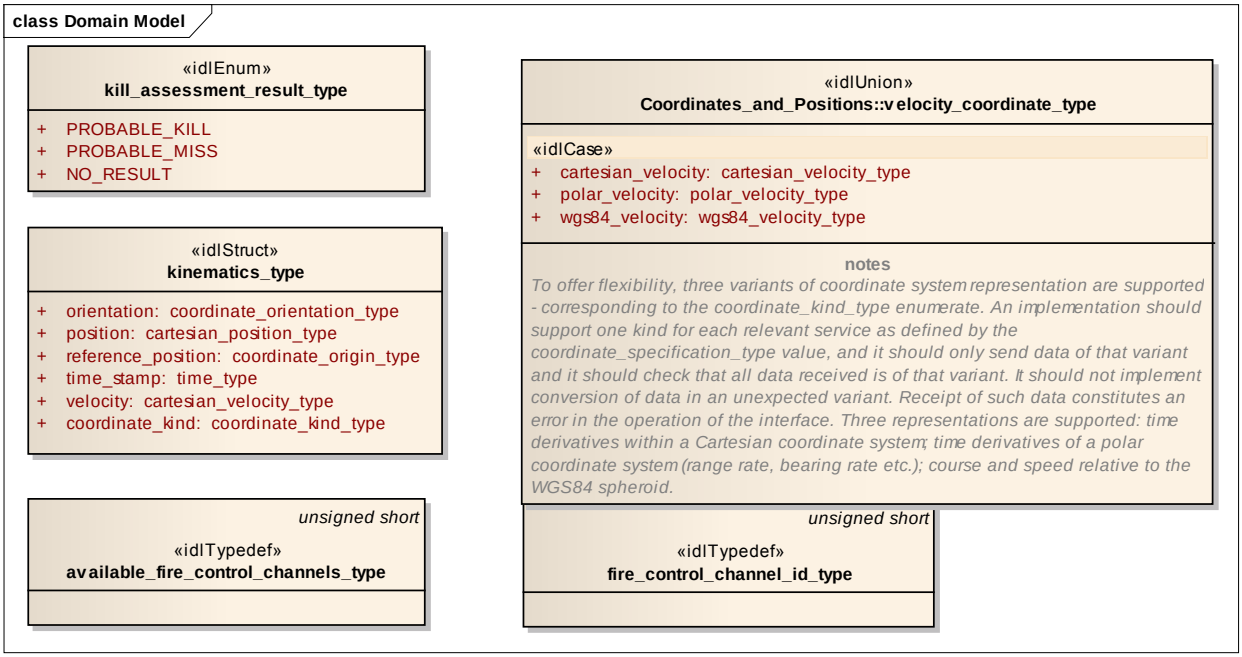

Figure 7.44 Domain Model (Logical diagram)

7.6.2.1 available_fire_control_channels_type

Type: IDLTypeDef unsigned short
Package: Engagement_Support
The number/amount of available fire control channels.

7.6.2.2 fire_control_channel_id_type

Type: IDLTypeDef unsigned short
Package: Engagement_Support
The fire control channel ID as assigned by the subsystem.

7.6.2.3 kill_assessment_result_type

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

Table 7.117 - Attributes of IDLEnum kill_assessment_result_type

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

7.6.2.4 kinematics_type

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

Table 7.118 - Attributes of IDLStruct kinematics_type

Attribute	Notes
orientation coordinate_orientation_type	
position cartesian_position_type	
reference_position coordinate_origin_type	
time_stamp time_type	
velocity cartesian_velocity_type	
coordinate_kind coordinate_kind_type	

7.6.3 Missile_Guidance

Parent Package: Radar_Domain

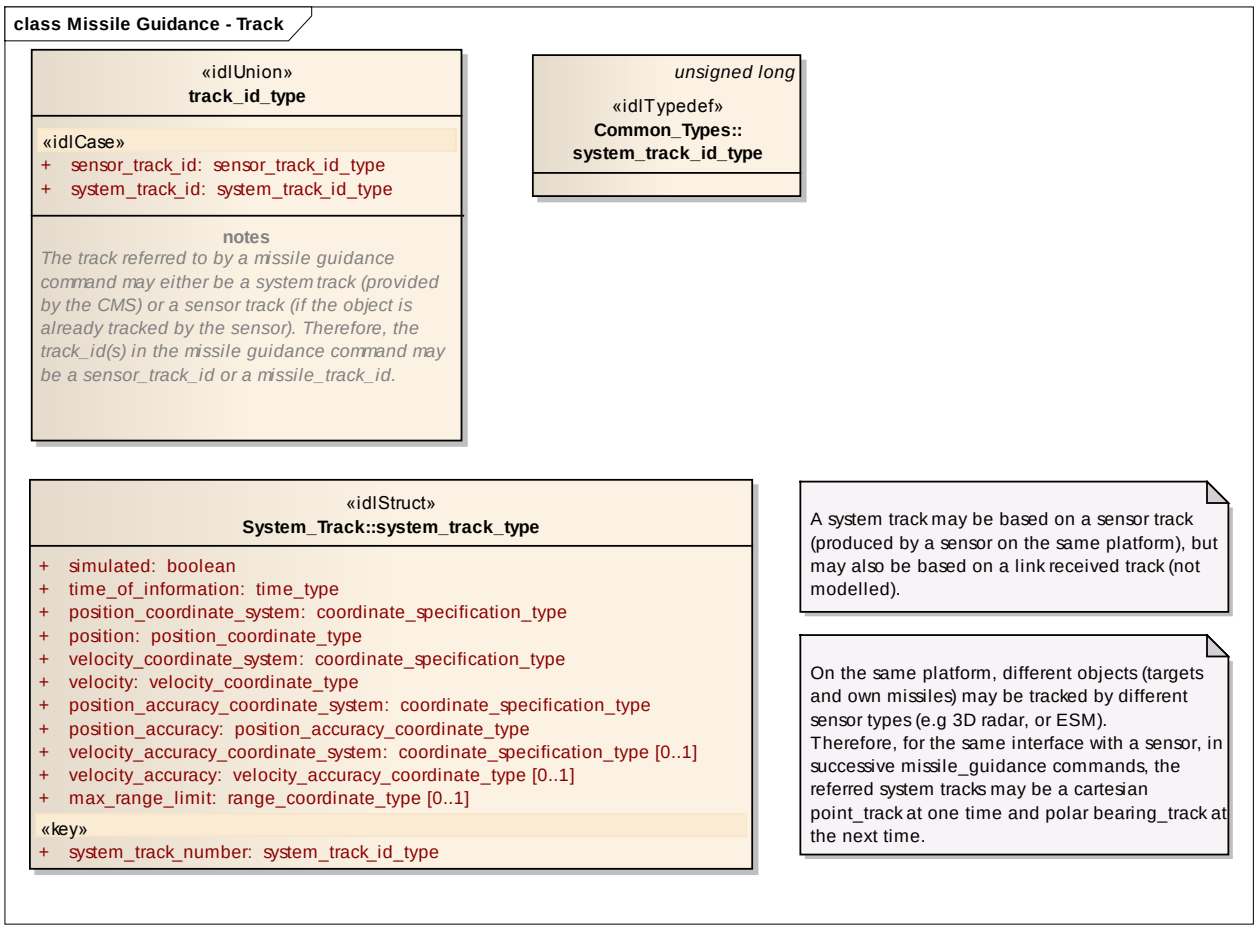

Figure 7.45 Missile Guidance - Track (Logical diagram)

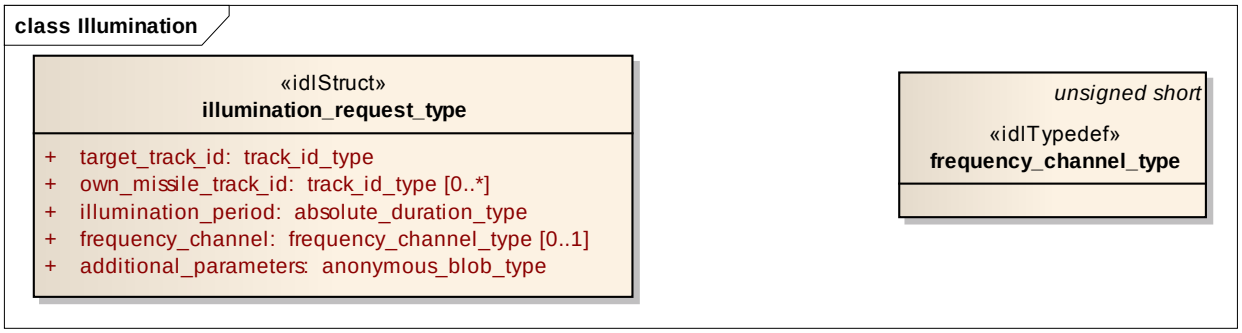

Figure 7.46 Illumination (Logical diagram)

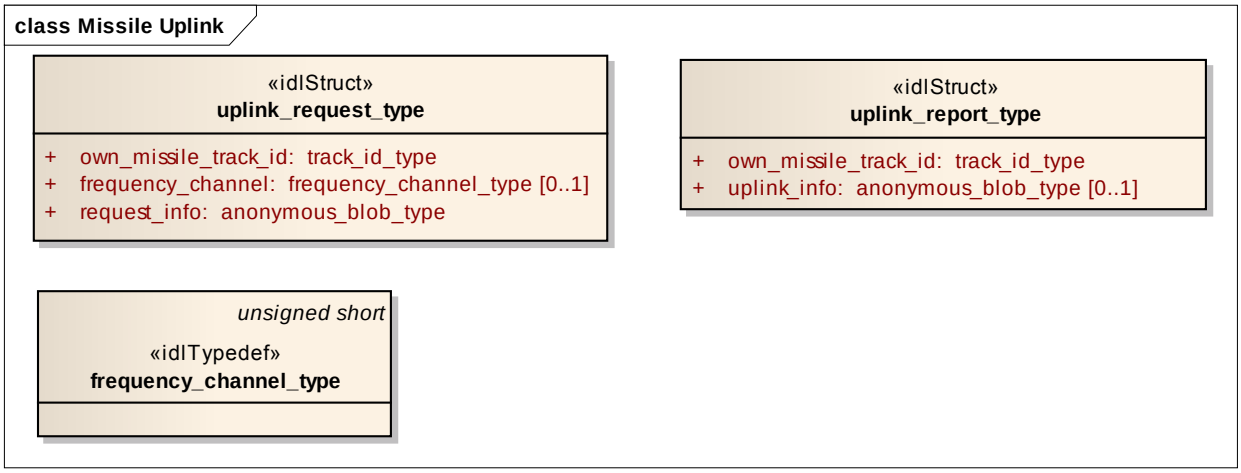

Figure 7.47 Missile Uplink (Logical diagram)

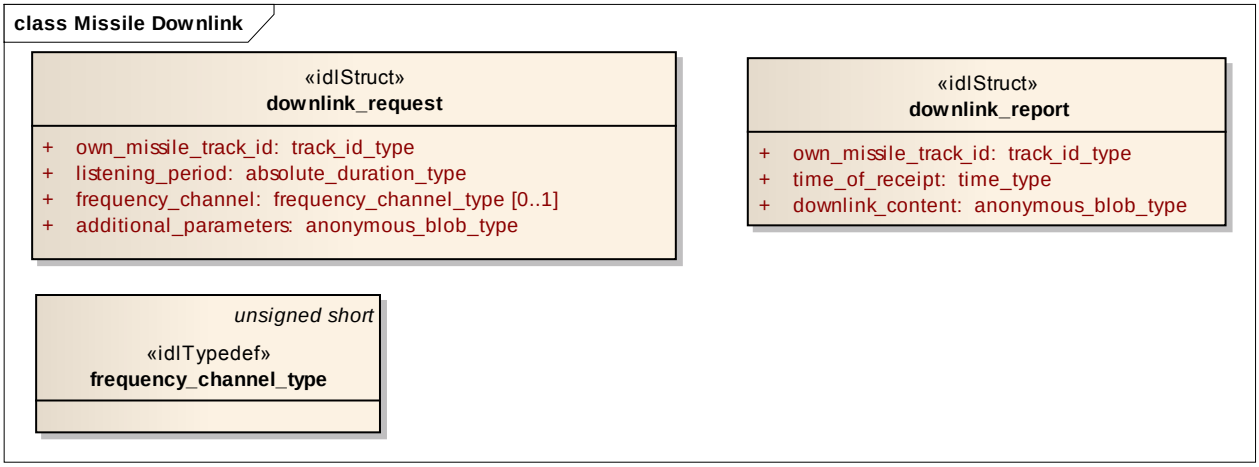

Figure 7.48 Missile Downlink (Logical diagram)

7.6.3.1 downlink_report

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

Table 7.119 - Attributes of IDLStruct downlink_report

Attribute	Notes
own_missile_track_id track_id_type	
time_of_receipt time_type	
downlink_content anonymous_blob_type	

7.6.3.2 downlink_request

Type: IDLStruct
Package: Missile_Guidance
request to downlink

Table 7.120 - Attributes of IDLStruct downlink_request

Attribute	Notes

Attribute	Notes
own_missile_track_id track_id_type	
listening_period absolute_duration_type	Start of period during which downlinks shall be received
frequency_channel frequency_channel_type [0..1]	
additional_parameters anonymous_blob_type	

7.6.3.3 frequency_channel_type

Type: IDLTypeDef unsigned short
Package: Missile_Guidance
A frequency channel identifies a specific radar frequency.

7.6.3.4 illumination_request_type

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

Table 7.121 - Attributes of IDLStruct illumination_request_type

Attribute	Notes
target_track_id track_id_type	
own_missile_track_id track_id_type [0..*]	
illumination_period absolute_duration_type	
frequency_channel frequency_channel_type [0..1]	
additional_parameters anonymous_blob_type	

7.6.3.5 track_id_type

Type: IDLUnion
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 7.122 - Attributes of IDLUnion track_id_type

Attribute	Notes
«idlCase» **sensor_track_id** sensor_track_id_type	
«idlCase» **system_track_id** system_track_id_type	

7.6.3.6 uplink_report_type

Type: IDLStruct
Package: Missile_Guidance
a report from uplink

Table 7.123 - Attributes of IDLStruct uplink_report_type

Attribute	Notes
own_missile_track_id track_id_type	
uplink_info anonymous_blob_type [0..1]	* optional *

7.6.3.7 uplink_request_type

Type: IDLStruct
Package: Missile_Guidance
a request to downlink

Table 7.124 - Attributes of IDLStruct uplink_request_type

Attribute	Notes

Attribute	Notes
own_missile_track_id track_id_type	
frequency_channel frequency_channel_type [0..1]	* optional *
request_info anonymous_blob_type	

7.6.4 Search

Parent Package: Radar_Domain

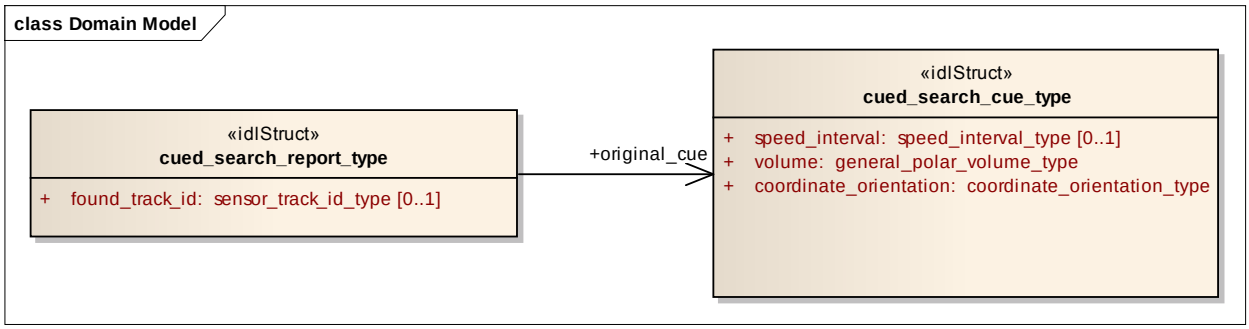

Figure 7.49 Domain Model (Logical diagram)

7.6.4.1 cued_search_cue_type

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

Table 7.125 - Attributes of IDLStruct 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.
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.

7.6.4.2 cued_search_report_type

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

Table 7.126 - Attributes of IDLStruct cued_search_report_type

Attribute	Notes
found_track_id sensor_track_id_type [0..1]	

7.6.5 Surface_Engagement_Support

Parent Package: Radar_Domain

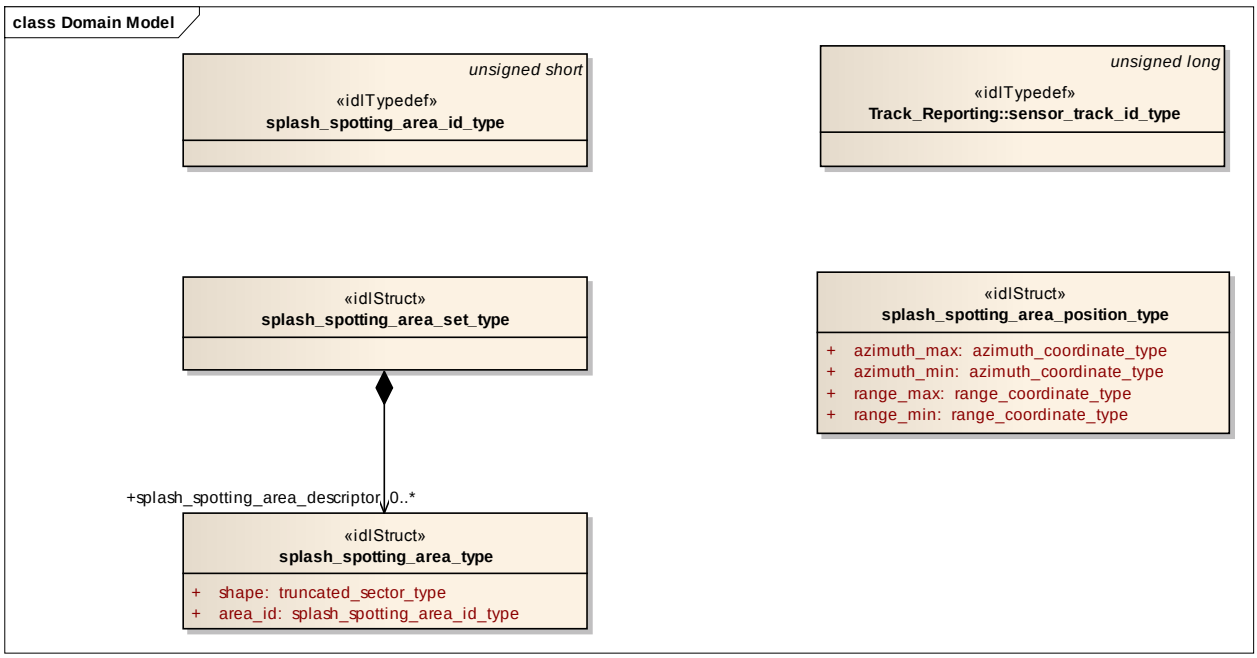

Figure 7.50 Domain Model (Logical diagram)

7.6.5.1 splash_spotting_area_id_type

Type: IDLTypeDef unsigned short
Package: Surface_Engagement_Support
the area ID assigned by the sensor.

7.6.5.2 splash_spotting_area_position_type

Type: IDLStruct
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 7.127 - Attributes of IDLStruct 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

7.6.5.3 splash_spotting_area_set_type

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

7.6.5.4 splash_spotting_area_type

Type: IDLStruct

Package: Surface_Engagement_Support
Definition of a single splash spotting area.

Table 7.128 - Attributes of IDLStruct 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.

7.7 Subsystem_Services

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

7.7.1 Encyclopaedic_Support

Parent Package: Subsystem_Services

7.7.1.1 Receive_Encyclopaedic_Data

Parent Package: Encyclopaedic_Support

7.7.1.1.1 Receive_Encyclopaedic_Data_CMS

Type: IDLInterface common_use_case_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 7.129 - Methods of IDLInterface 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

7.7.1.1.2 Receive_Encyclopaedic_Data_Sub

Type: IDLInterface
Package: Receive_Encyclopaedic_Data

Table 7.130 - Methods of IDLInterface 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.

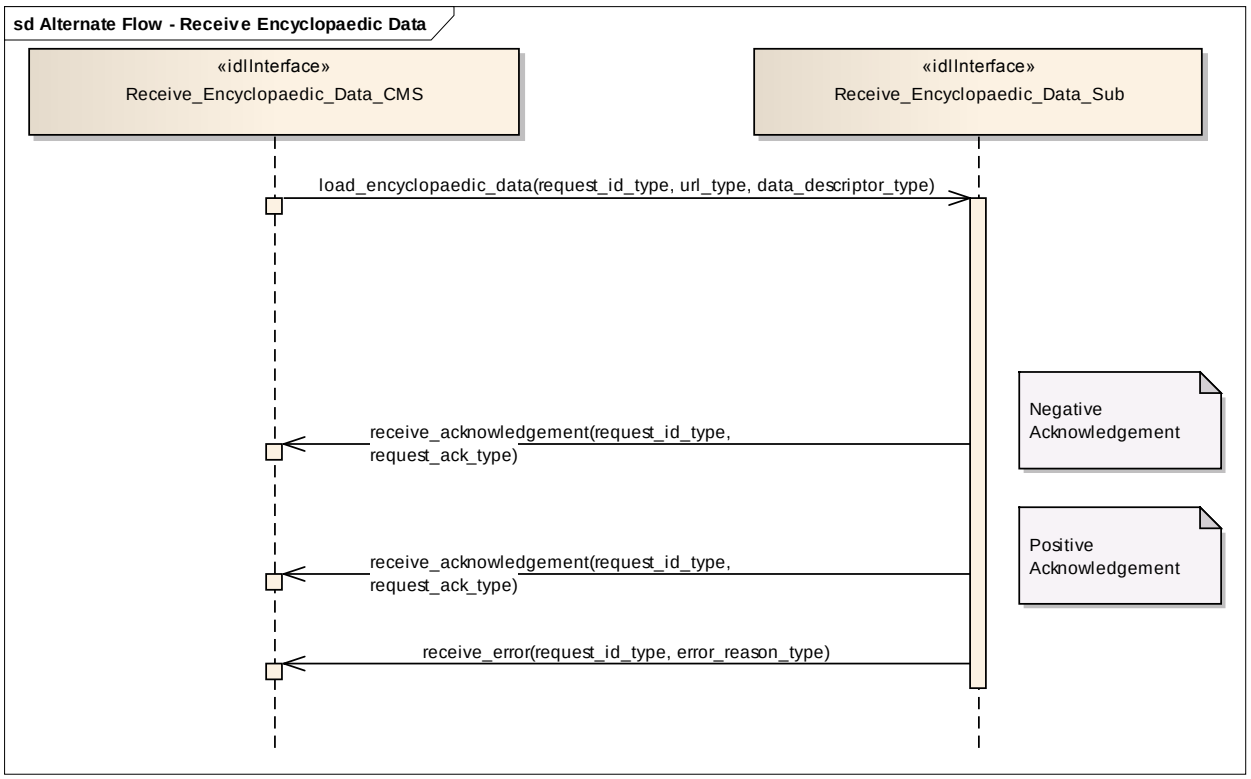

Figure 7.51 Alternate Flow - Receive Encyclopaedic Data (Sequence diagram)

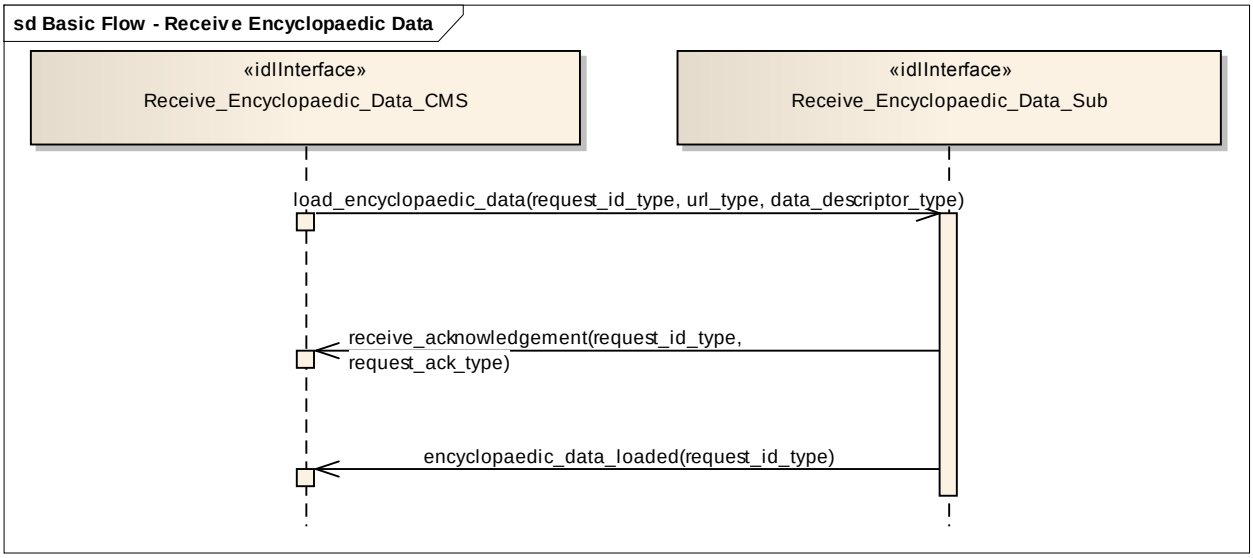

Figure 7.52 Basic Flow - Receive Encyclopaedic Data (Sequence diagram)

7.7.2 Extended_Subsystem_Control

Parent Package: Subsystem_Services

Contains interfaces for the Extended Subsystem Control service.

7.7.2.1 Manage Physical Configuration

Parent Package: Extended_Subsystem_Control
Contains operations and sequence diagrams for the Manage Physical Configuration interface.

7.7.2.1.1 Manage_Physical_Configuration_CMS

Type: IDLInterface common_use_case_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 7.131 - Methods of IDLInterface 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**

7.7.2.1.2 Manage_Physical_Configuration_Sub

Type: IDLInterface
Package: Manage Physical Configuration

Table 7.132 - Methods of IDLInterface 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	request_id_type **request_id**

	provide its current physical configuration data.	

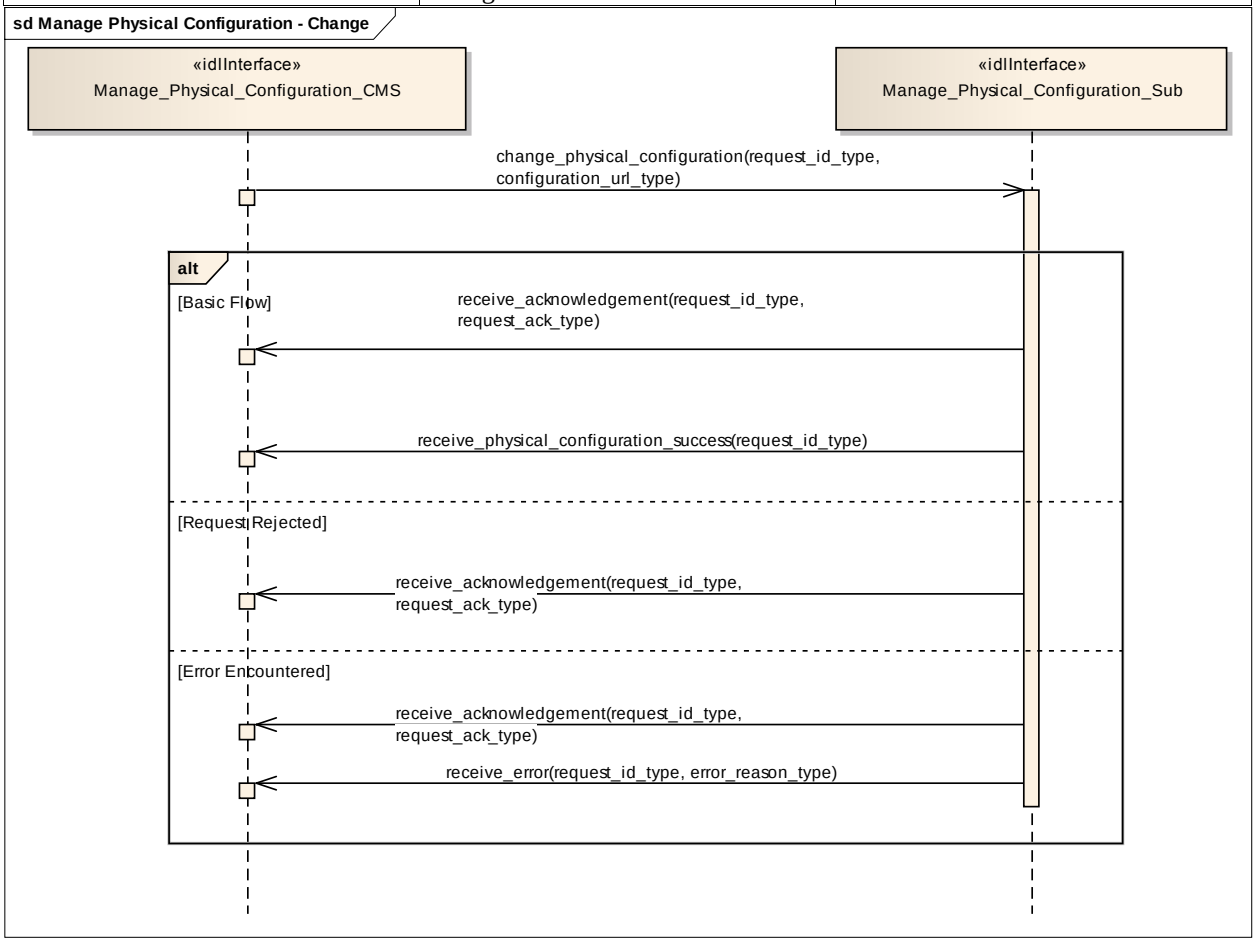

Figure 7.53 Manage Physical Configuration - Change (Sequence diagram)

Flow of events which depicts the CMS requesting that the subsystem changing its physical configuration data (also depicts alternate rejection and error paths).

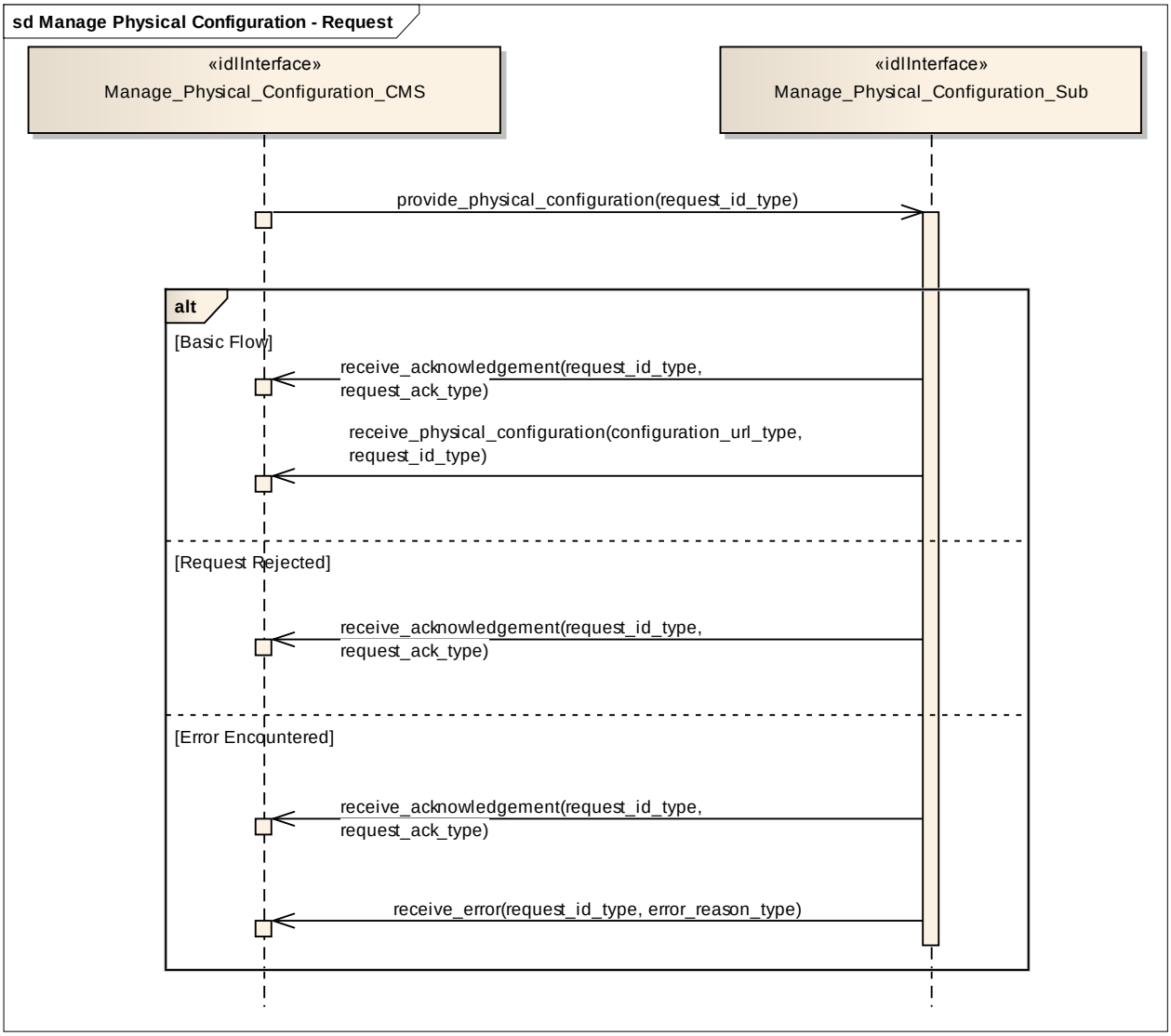

Figure 7.54 Manage Physical Configuration - Request (Sequence diagram)

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).

7.7.2.2 Perform Offline Test

Parent Package: Extended_Subsystem_Control
Contains the interface for offline testing.

7.7.2.2.1 Perform_Offline_Test_CMS

Type: IDLInterface common_use_case_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 7.133 - Methods of IDLInterface 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**

7.7.2.2.2 Perform_Offline_Test_Sub

Type: IDLInterface
Package: Perform Offline Test

Table 7.134 - Methods of IDLInterface 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**

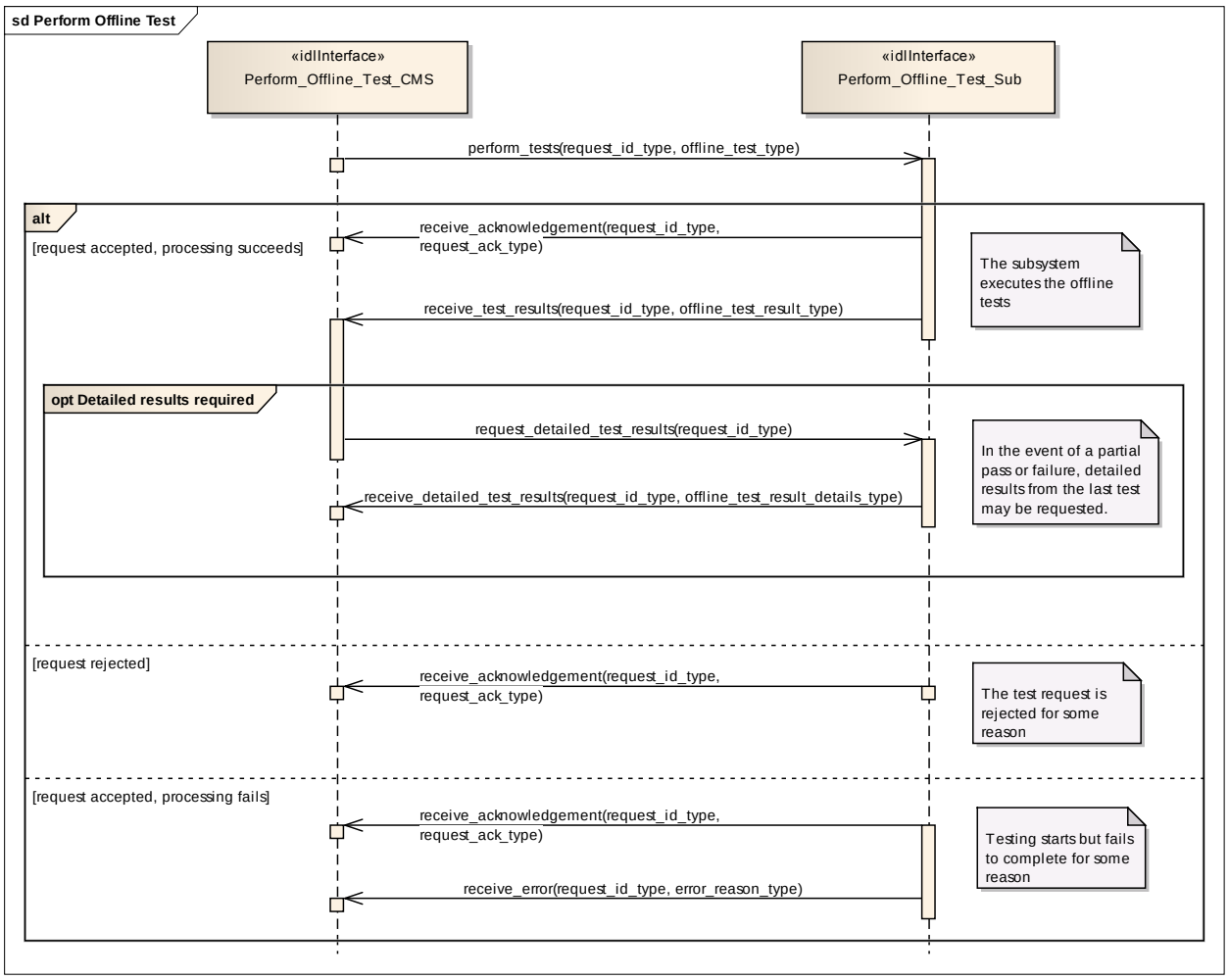

Figure 7.55 Perform Offline Test (Sequence diagram)

This shows the required sequential behaviour for Perform_Offline_Test, See diagram embedded notes for further explanation

7.7.2.3 Restart

Parent Package: Extended_Subsystem_Control
Contains operations and sequence diagrams for the Restart interface.

7.7.2.3.1 Restart_CMS

Type: IDLInterface common_use_case_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 7.135 - Methods of IDLInterface 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**

7.7.2.3.2 Restart_Sub

Type: IDLInterface
Package: Restart

Table 7.136 - Methods of IDLInterface 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**

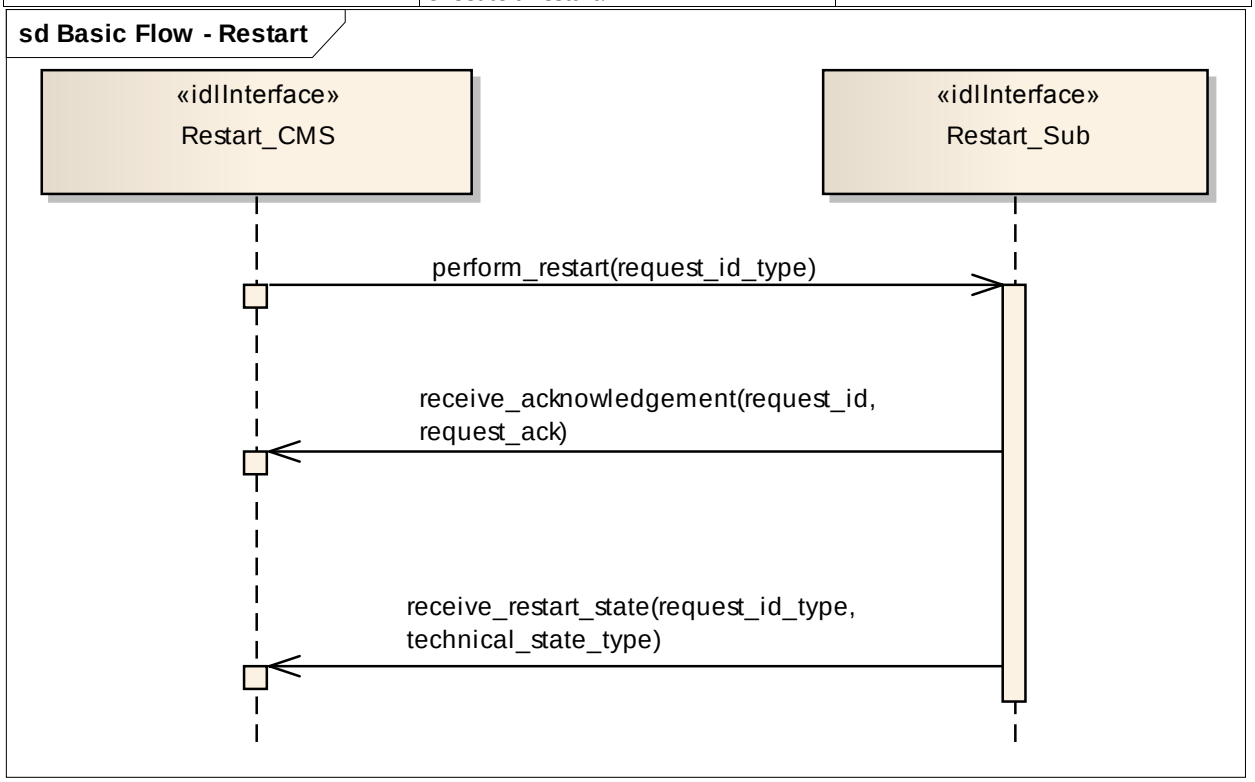

Figure 7.56 Basic Flow - Restart (Sequence diagram)

Basic flow for CMS requesting the subsystem to transition to STANDBY followed by a transition to READY.

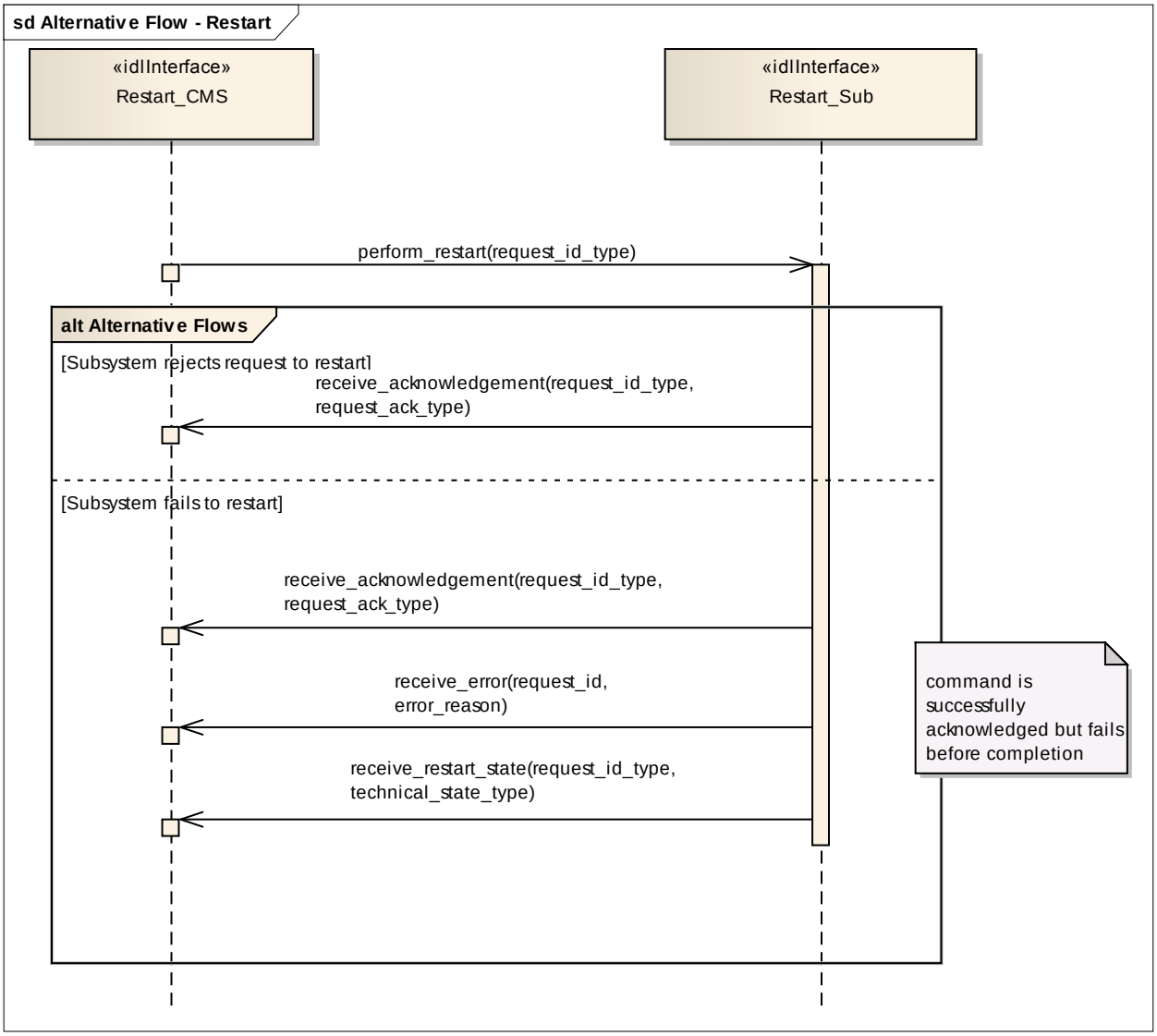

Figure 7.57 Alternative Flow - Restart (Sequence diagram)

Alternate flow for CMS requesting the subsystem to transition to STANDBY followed by a transition to READY (depicts rejection and error paths).

7.7.2.4 Shutdown

Parent Package: Extended_Subsystem_Control
Contains operations and sequence diagrams for the Shutdown interface.

7.7.2.4.1 Shutdown_CMS

Type: IDLInterface common_use_case_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 7.137 - Methods of IDLInterface 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**

7.7.2.4.2 Shutdown_Sub

Type: IDLInterface
Package: Shutdown

Table 7.138 - Methods of IDLInterface 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**

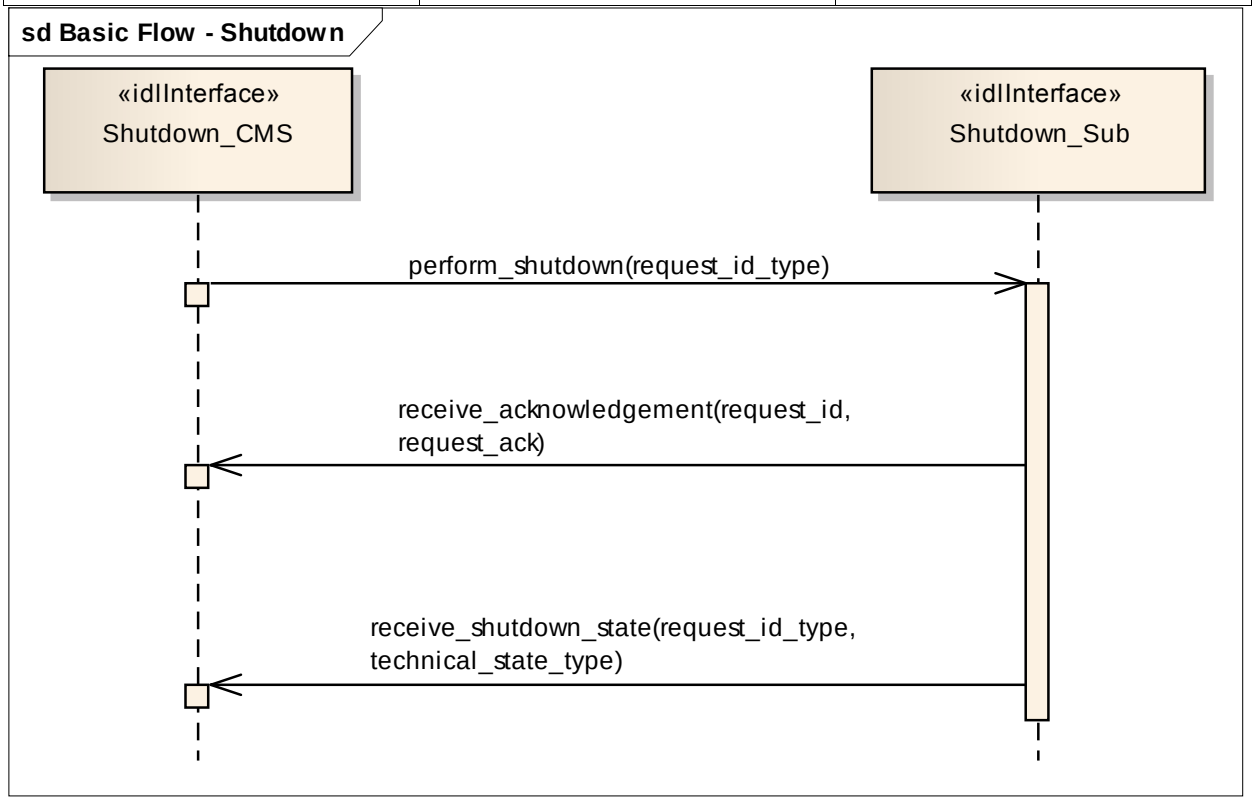

Figure 7.58 Basic Flow - Shutdown (Sequence diagram)

Basic flow for CMS requesting the subsystem to transition to STANDBY.

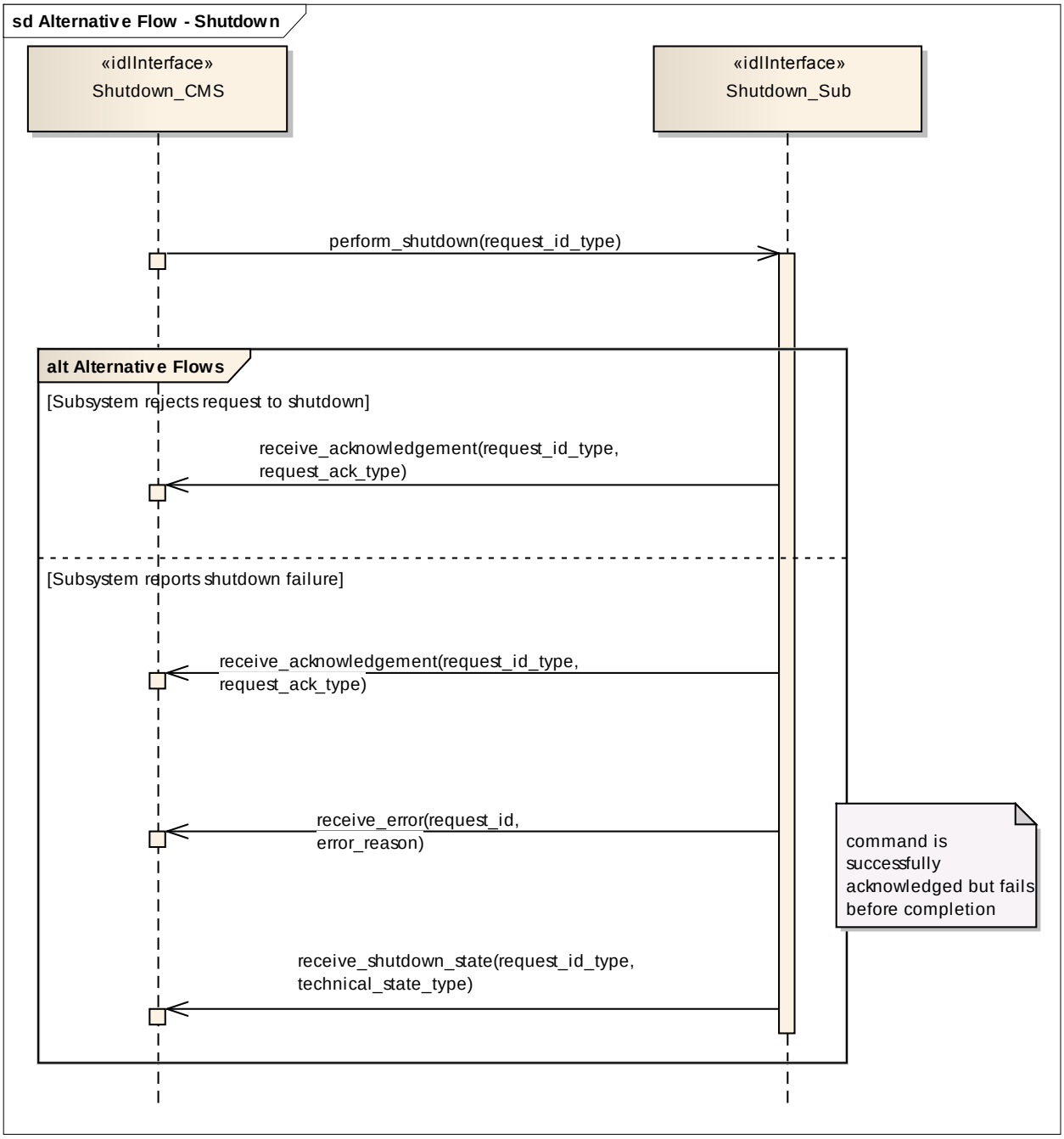

Figure 7.59 Alternative Flow - Shutdown (Sequence diagram)

Alternate flow for CMS requesting the subsystem to transition to STANDBY (depicts rejection and error paths).

7.7.2.5 Startup

Parent Package: Extended_Subsystem_Control

Contains operations and sequence diagrams for the Startup interface.

7.7.2.5.1 Startup_CMS

Type: IDLInterface common_use_case_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 7.139 - Methods of IDLInterface 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**

7.7.2.5.2 Startup_Sub

Type: IDLInterface
Package: Startup

Table 7.140 - Methods of IDLInterface 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**

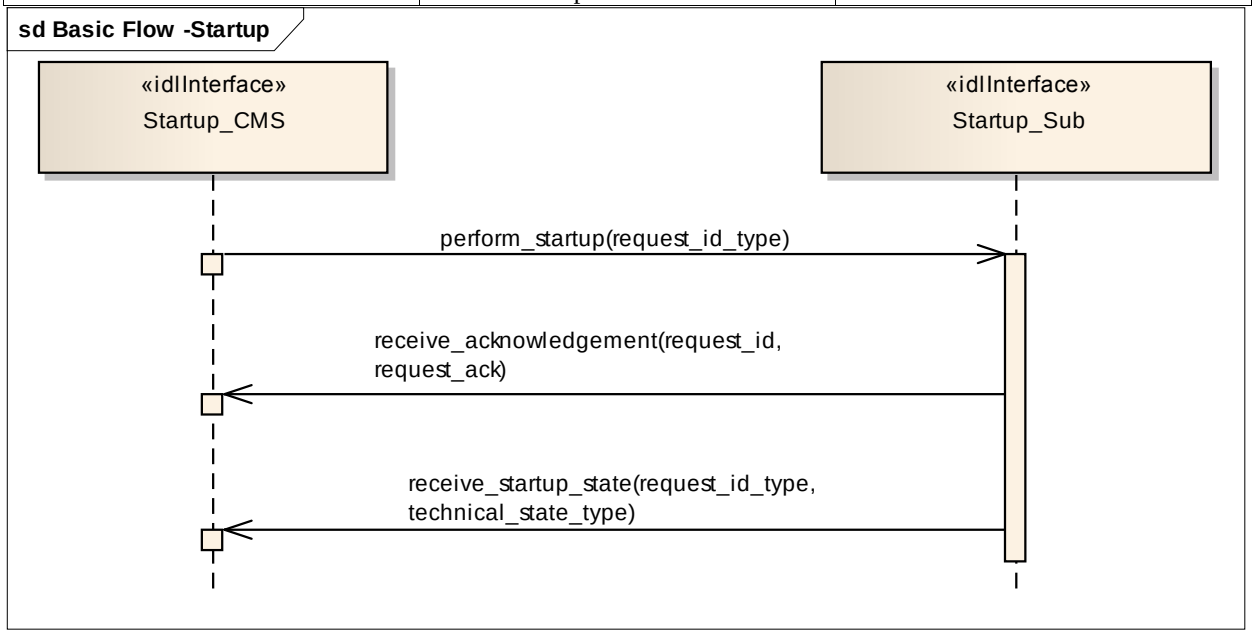

Figure 7.60 Basic Flow -Startup (Sequence diagram)

Basic flow for CMS requesting the subsystem to transition from STANDBY to READY.

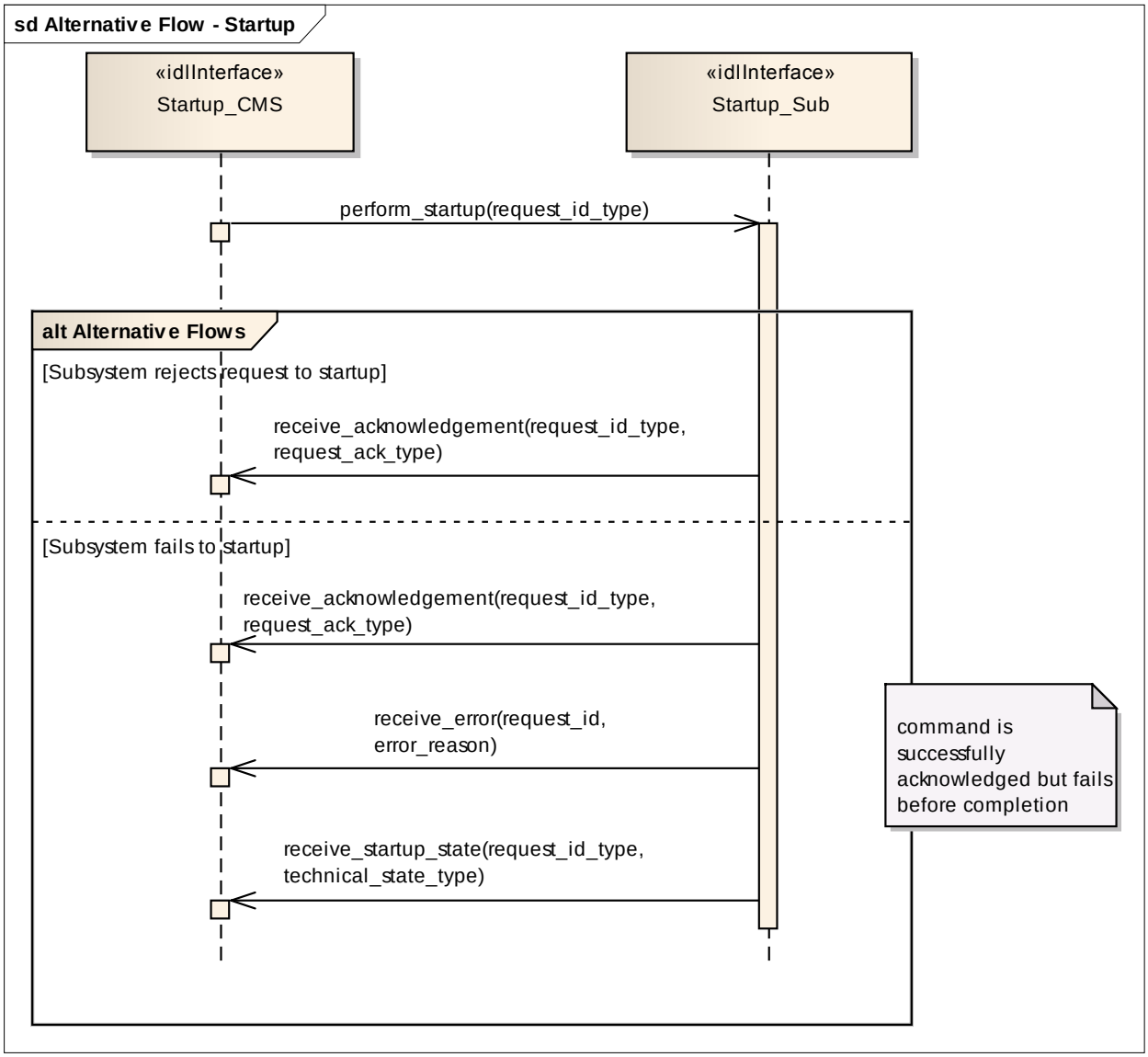

Figure 7.61 Alternative Flow - Startup (Sequence diagram)

Alternate flow for CMS requesting the subsystem to transition from STANDBY to READY (depicts rejection and error paths).

7.7.3 Recording_and_Replay

Parent Package: Subsystem_Services
Contains the interfaces controlling recording and replay.

7.7.3.1 Control_Recording

Parent Package: Recording_and_Replay
Contains the interface controlling the recording of information.

7.7.3.1.1 Control_Recording_CMS

Type: IDLInterface common_use_case_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.

7.7.3.1.2 Control_Recording_Sub

Type: IDLInterface
Package: Control_Recording

Table 7.141 - Methods of IDLInterface 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**

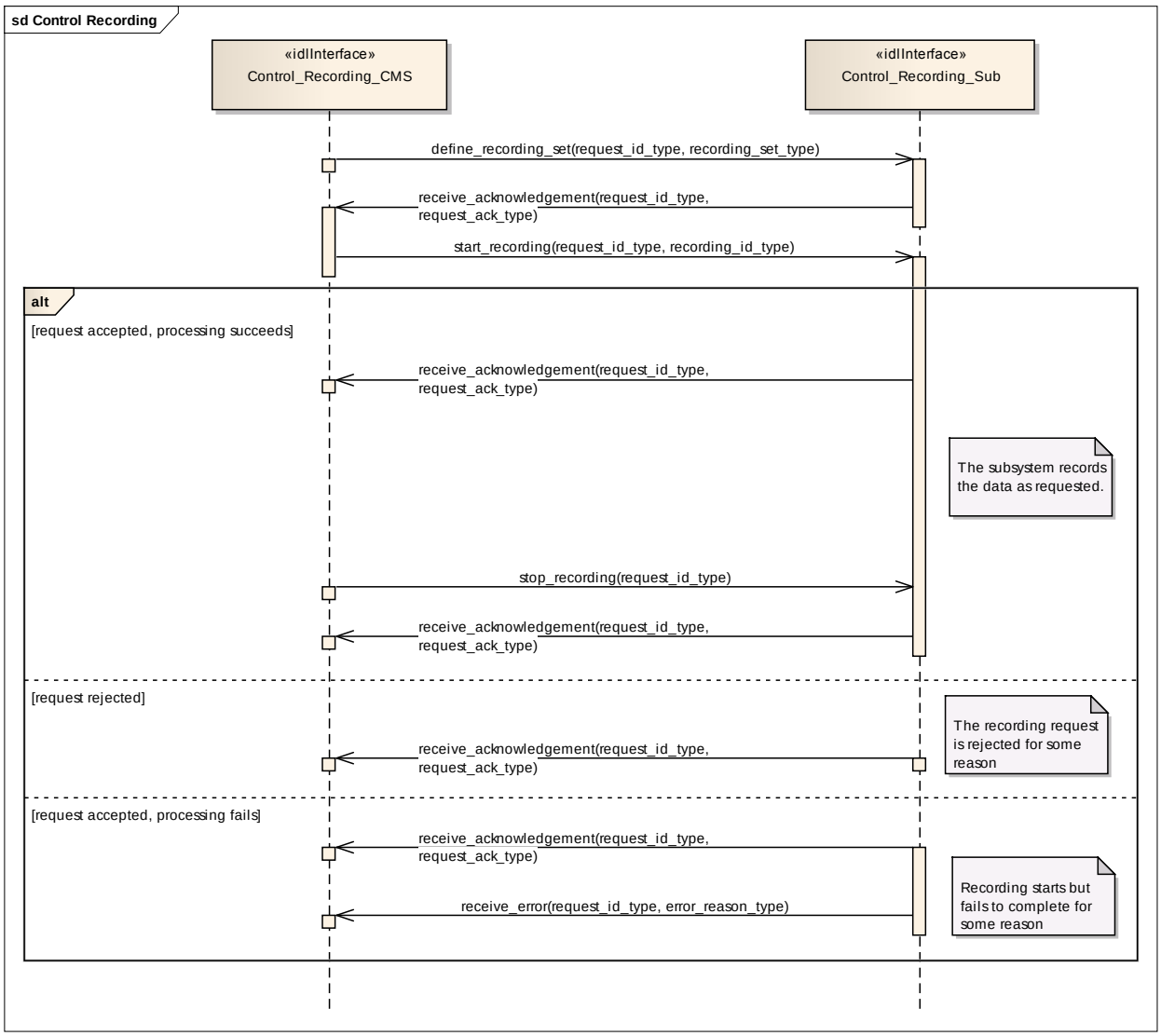

Figure 7.62 Control Recording (Sequence diagram)

This shows the required sequential behaviour for Control_Recording, See diagram embedded notes for further explanation.

7.7.3.2 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.

7.7.3.2.1 Control_Replay_CMS

Type: IDLInterface common_use_case_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 7.142 - Methods of IDLInterface 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.

7.7.3.2.2 Control_Replay_Sub

Type: IDLInterface
Package: Control_Replay

Table 7.143 - Methods of IDLInterface 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**

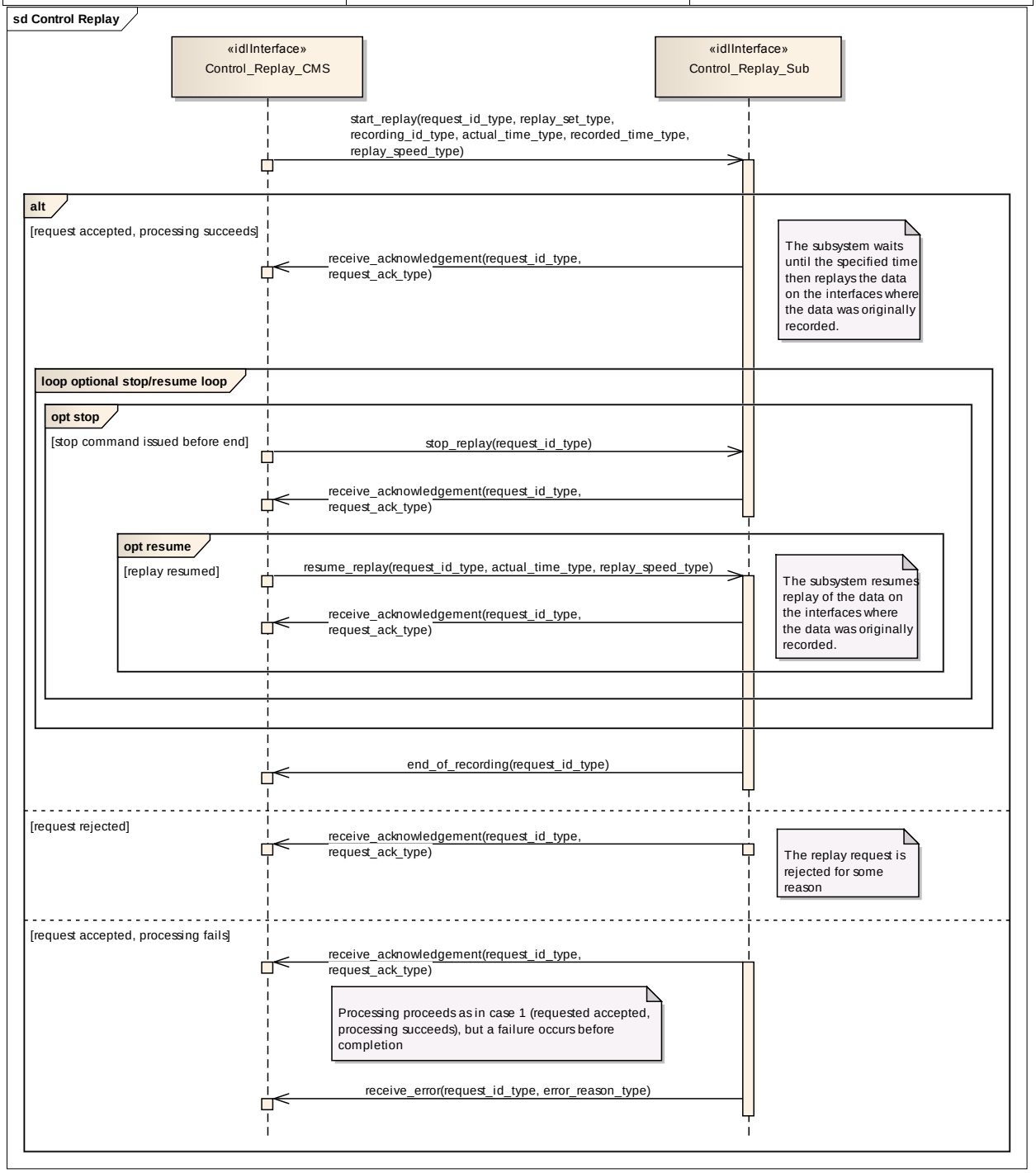

Figure 7.63 Control Replay (Sequence diagram)

This shows the required sequential behaviour for Control_Replay using real_time mode, See diagram embedded notes for further explanation.

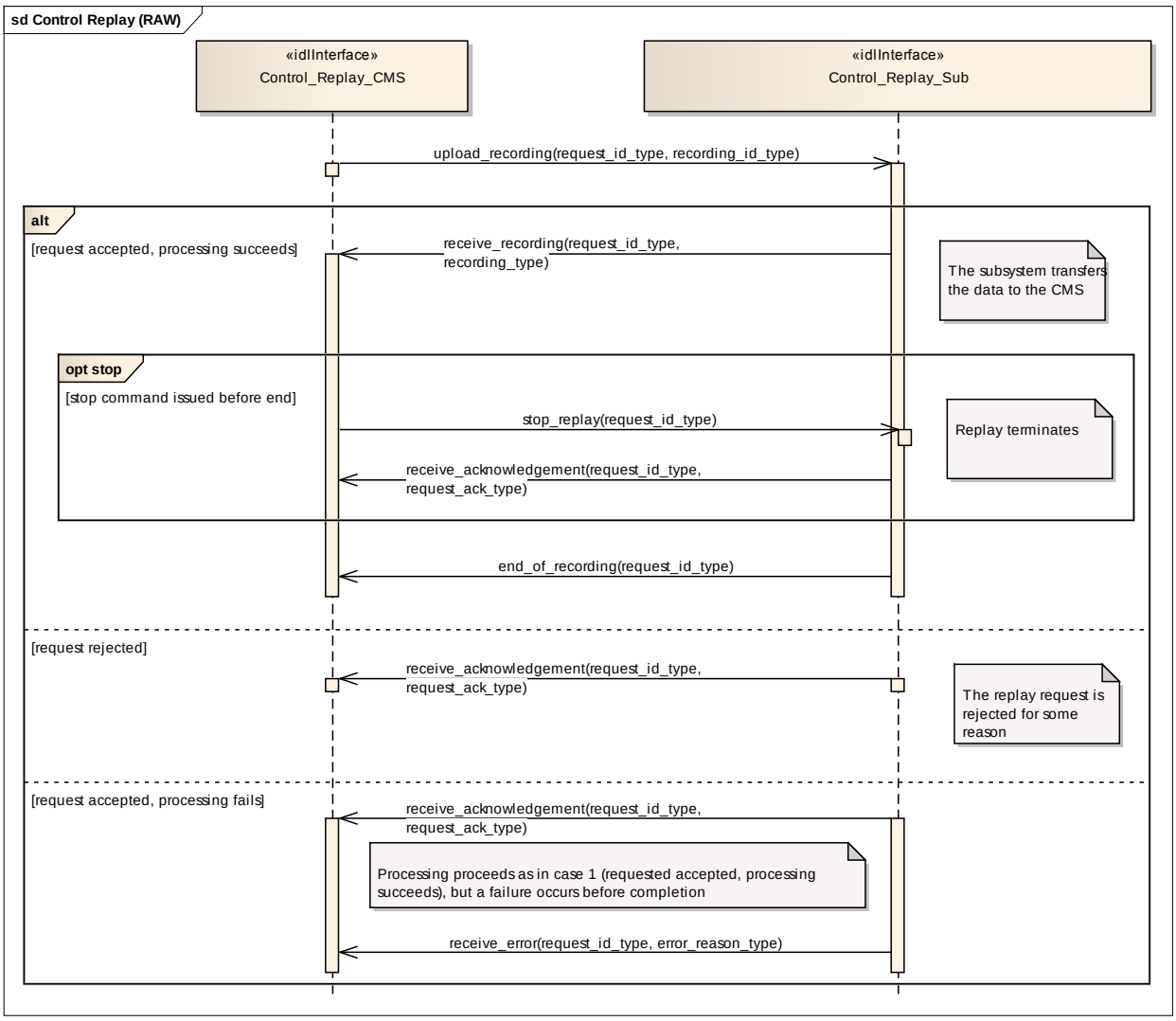

Figure 7.64 Control Replay (RAW) (Sequence diagram)

This shows the required sequential behaviour for Control_Replay using raw mode, See diagram embedded notes for further explanation.

7.7.4 Simulation_Support

Parent Package: Subsystem_Services

7.7.4.1 Define_Simulation_Scenario

Parent Package: Simulation_Support

7.7.4.1.1 Define_Simulation_Scenario_CMS

Type: IDLInterface
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 7.144 - Methods of IDLInterface 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**

7.7.4.1.2 Define_Simulation_Scenario_Sub

Type: IDLInterface
Package: Define_Simulation_Scenario

Table 7.145 - Methods of IDLInterface 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**

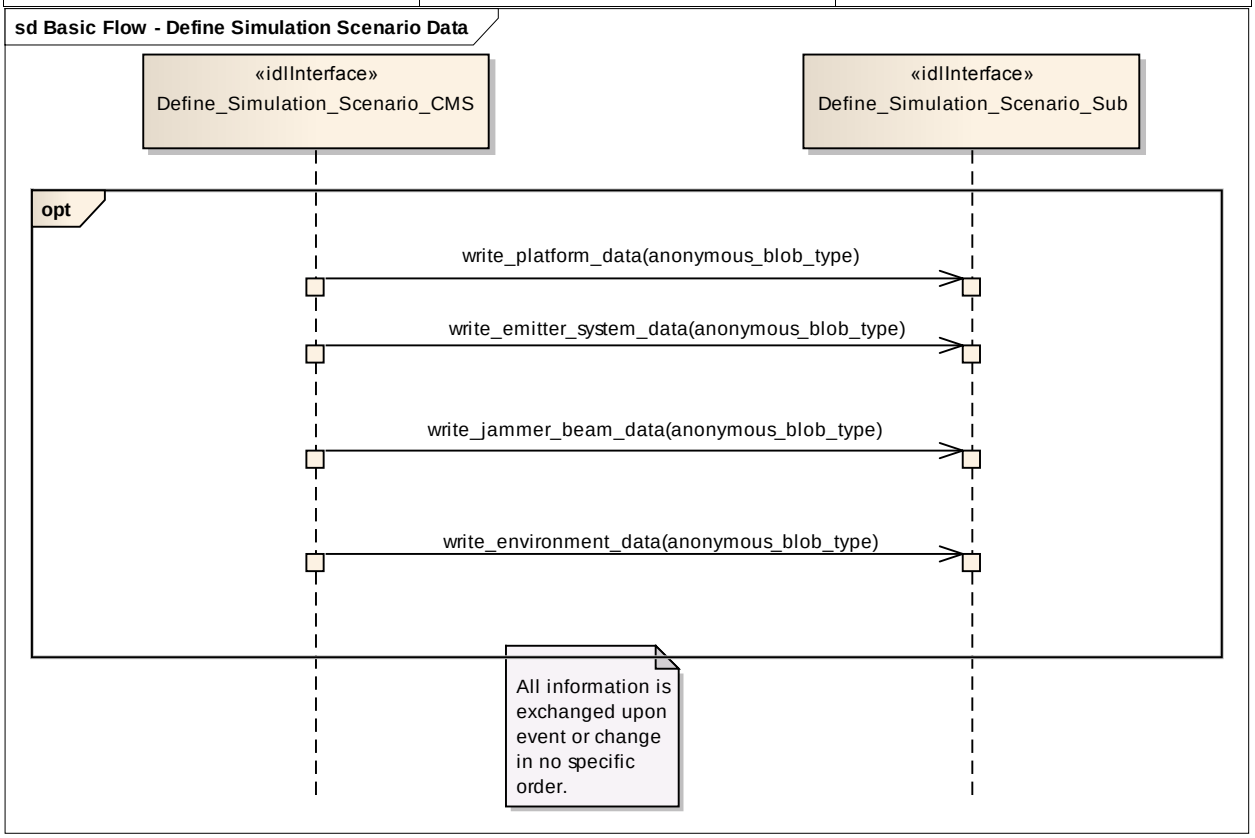

Figure 7.65 Basic Flow - Define Simulation Scenario Data (Sequence diagram)

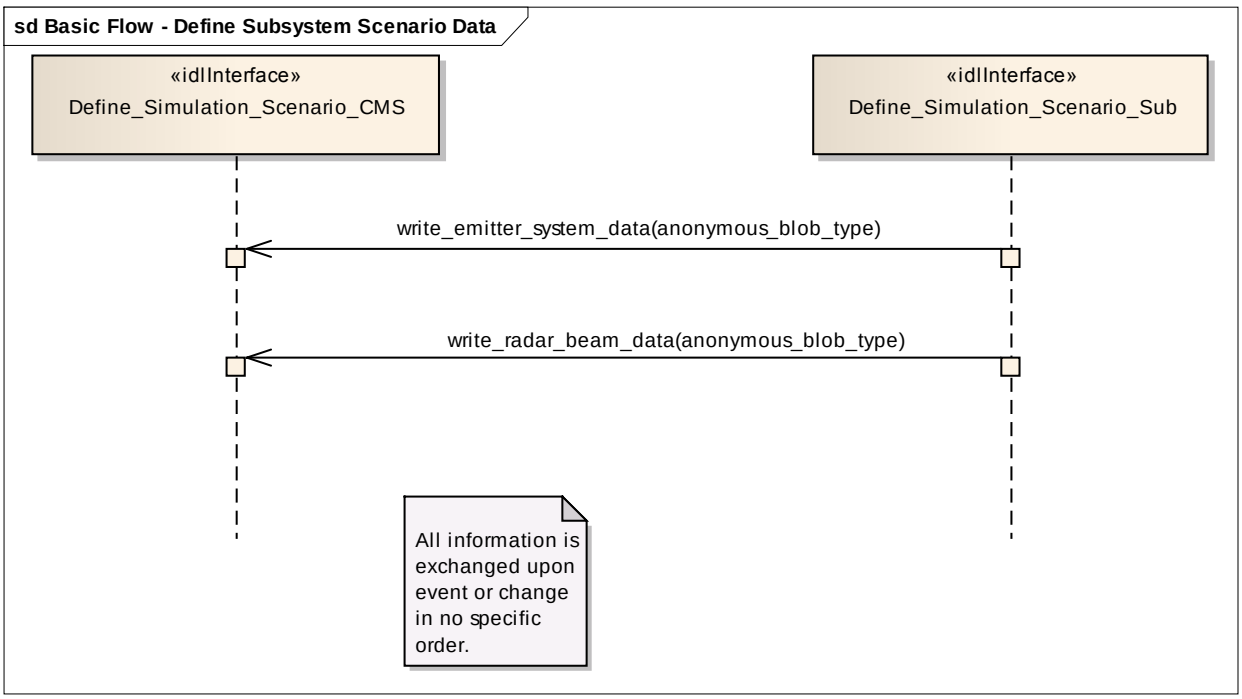

Figure 7.66 Basic Flow - Define Subsystem Scenario Data (Sequence diagram)

7.7.4.2 Control_Simulation

Parent Package: Simulation_Support

7.7.4.2.1 Control_Simulation_CMS

Type: IDLInterface common_use_case_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 7.146 - Methods of IDLInterface Control_Simulation_CMS

Method	Notes	Parameters
sim_mode_status()	Receive the status and mode of simulation.	request_id_type **request_id** sim_mode_status_type **the_status**

7.7.4.2.2 Control_Simulation_Sub

Type: IDLInterface common_use_case_interface
Package: Control_Simulation

Table 7.147 - Methods of IDLInterface 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_id_type **request_id** start_stop_sim_mode_request_type **the_request**

	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**

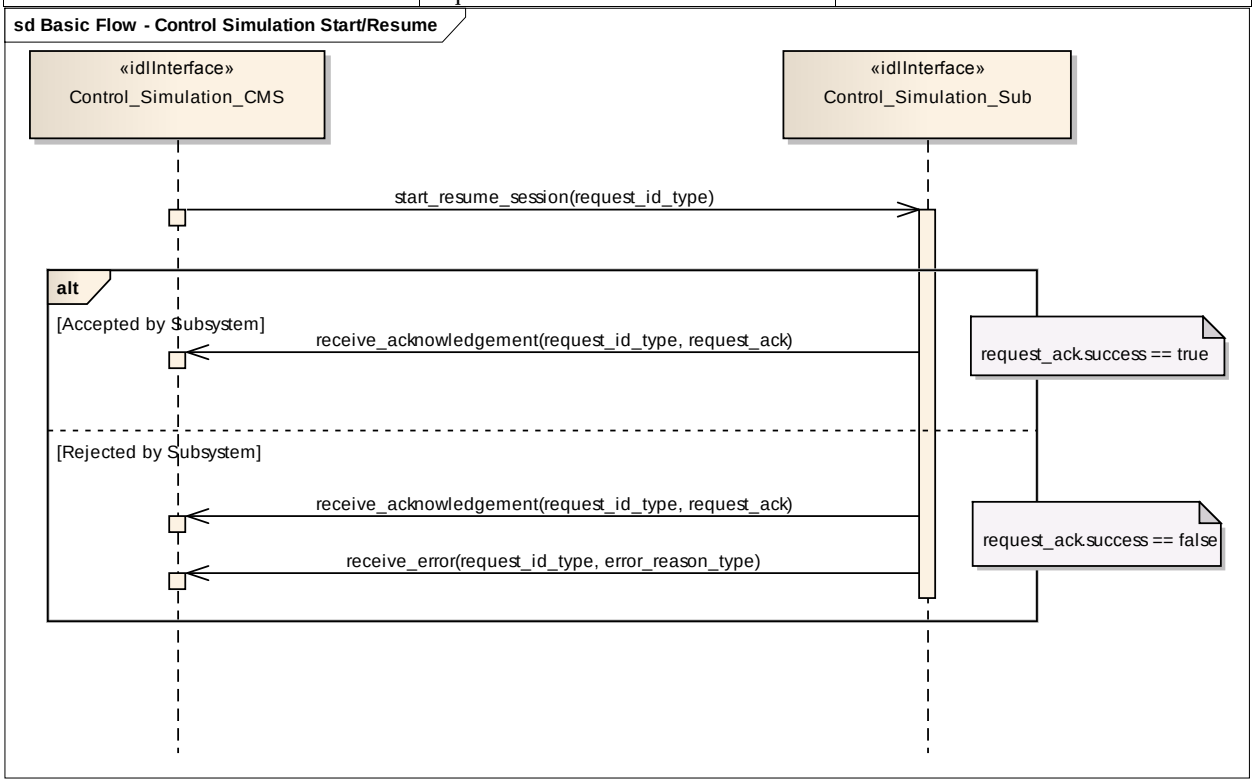

Figure 7.67 Basic Flow - Control Simulation Start/Resume (Sequence diagram)

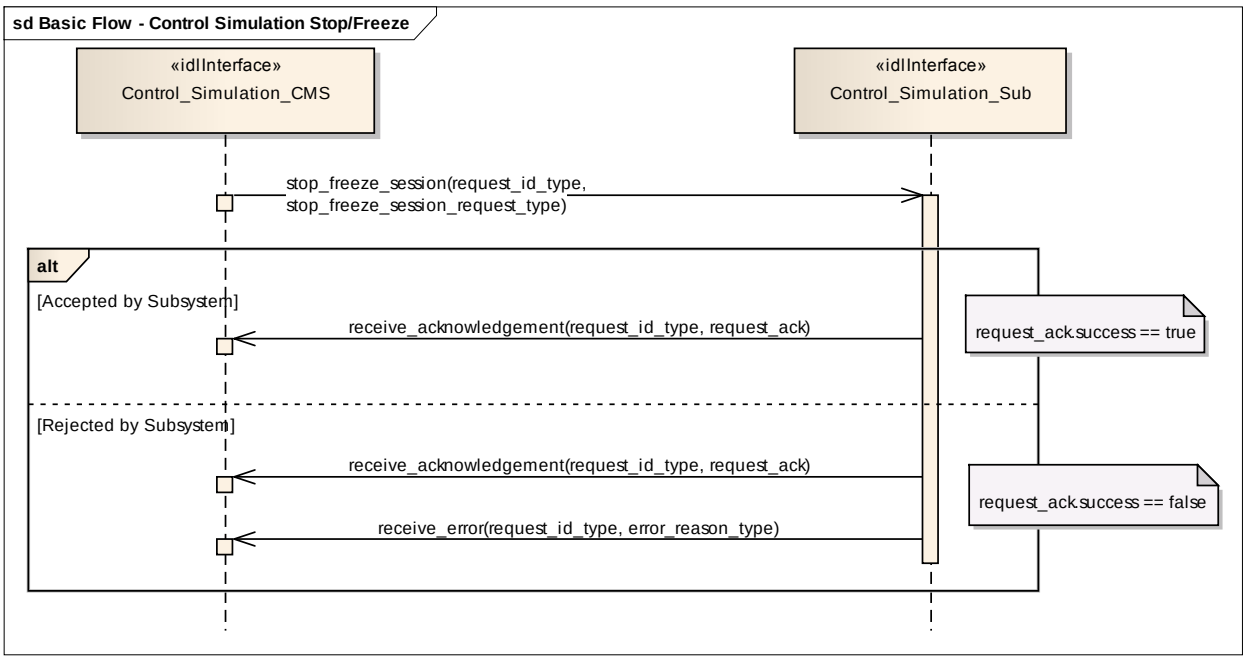

Figure 7.68 Basic Flow - Control Simulation Stop/Freeze (Sequence diagram)

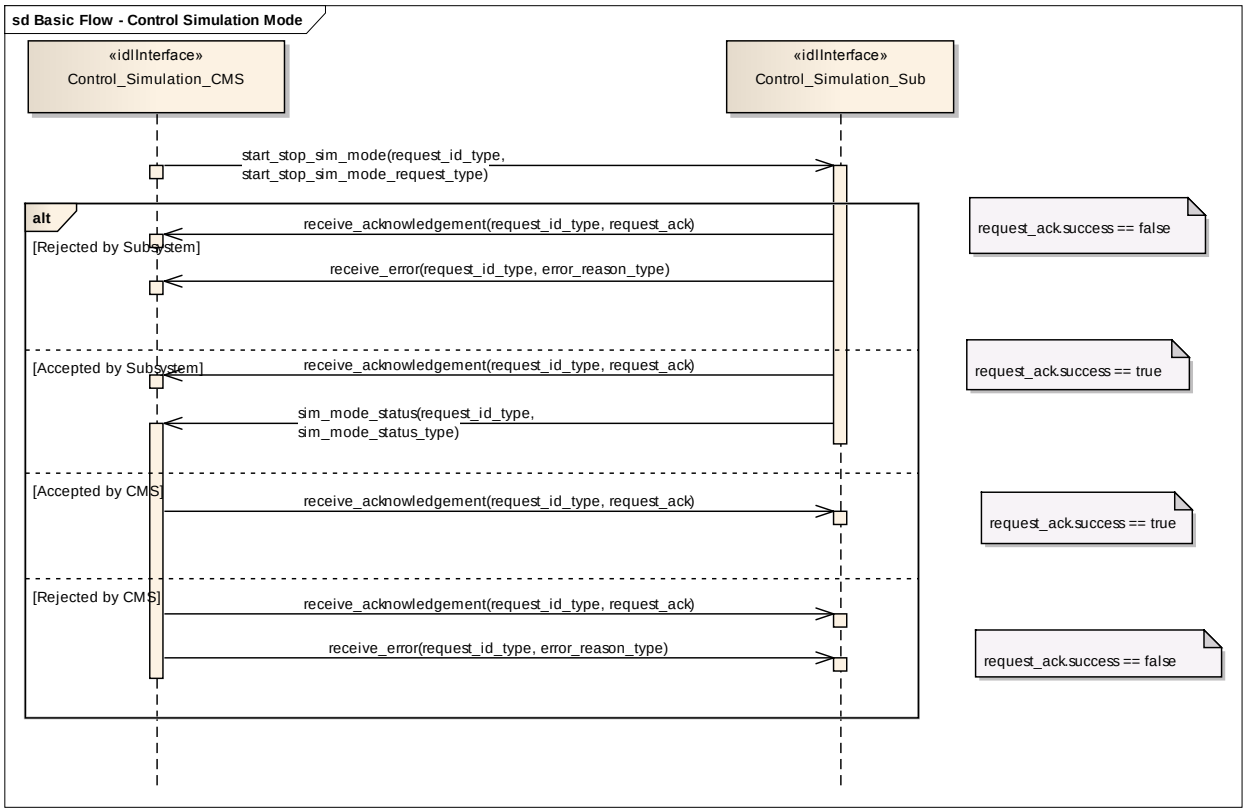

Figure 7.69 Basic Flow - Control Simulation Mode (Sequence diagram)

7.7.4.3 Define_Fault_Scripts

Parent Package: Simulation_Support

7.7.4.3.1 Define_Fault_Scripts_CMS

Type: IDLInterface common_use_case_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 7.148 - Methods of IDLInterface 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

7.7.4.3.2 Define_Fault_Scripts_Sub

Type: IDLInterface
Package: Define_Fault_Scripts

Table 7.149 - Methods of IDLInterface Define_Fault_Scripts_Sub

Method	Notes	Parameters
add_fault_scripts()	Adds the given fault scripts to the subsystem's simulation.	request_id_type **request_id** 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

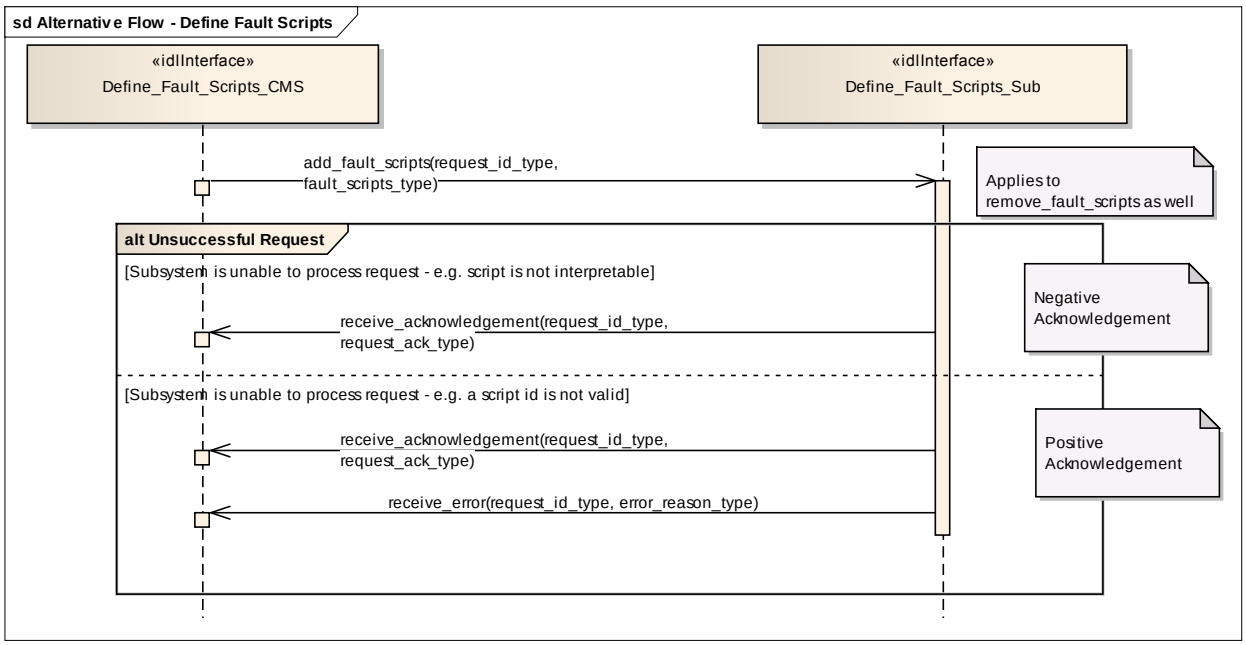

Figure 7.70 Alternative Flow - Define Fault Scripts (Sequence diagram)

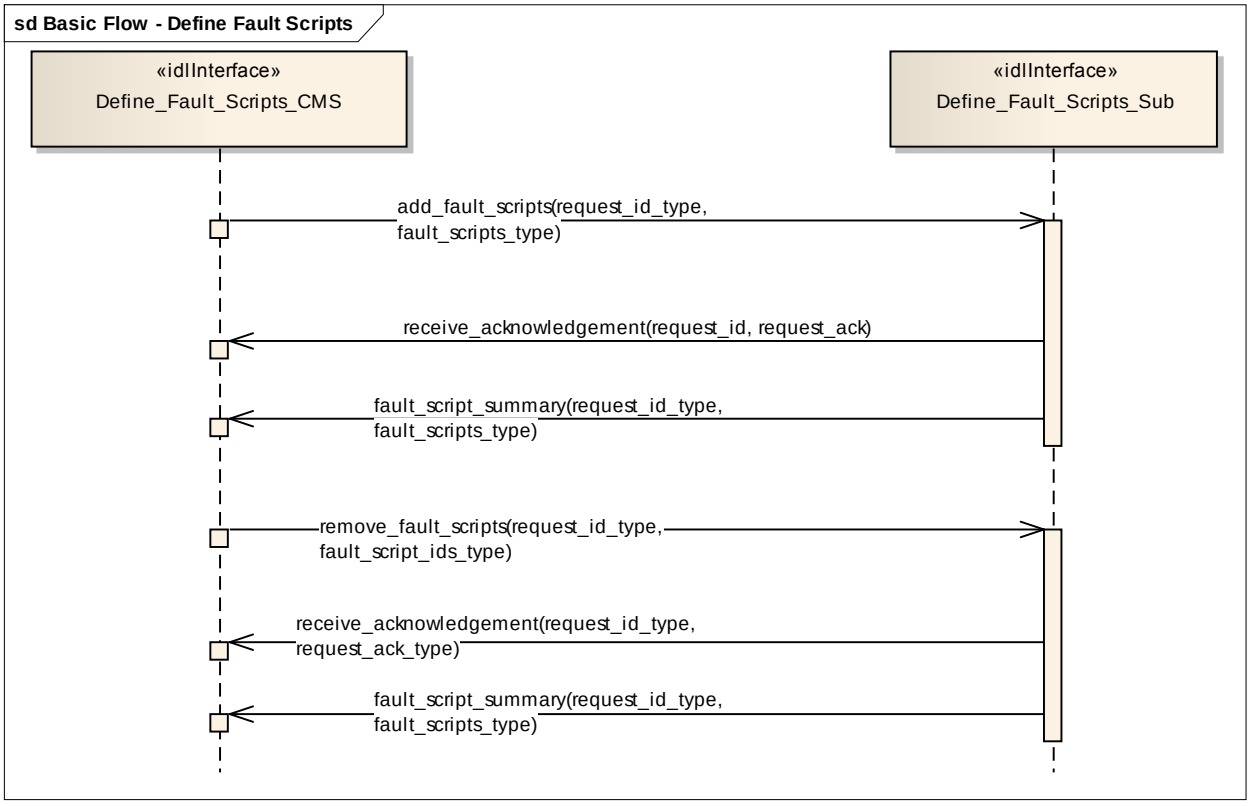

Figure 7.71 Basic Flow - Define Fault Scripts (Sequence diagram)

7.7.4.4 Control_Fault_Scripts

Parent Package: Simulation_Support

7.7.4.4.1 Control_Fault_Scripts_CMS

Type: IDLInterface common_use_case_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 scripts 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

7.7.4.4.2 Control_Fault_Scripts_Sub

Type: IDLInterface
Package: Control_Fault_Scripts

Table 7.150 - Methods of IDLInterface 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

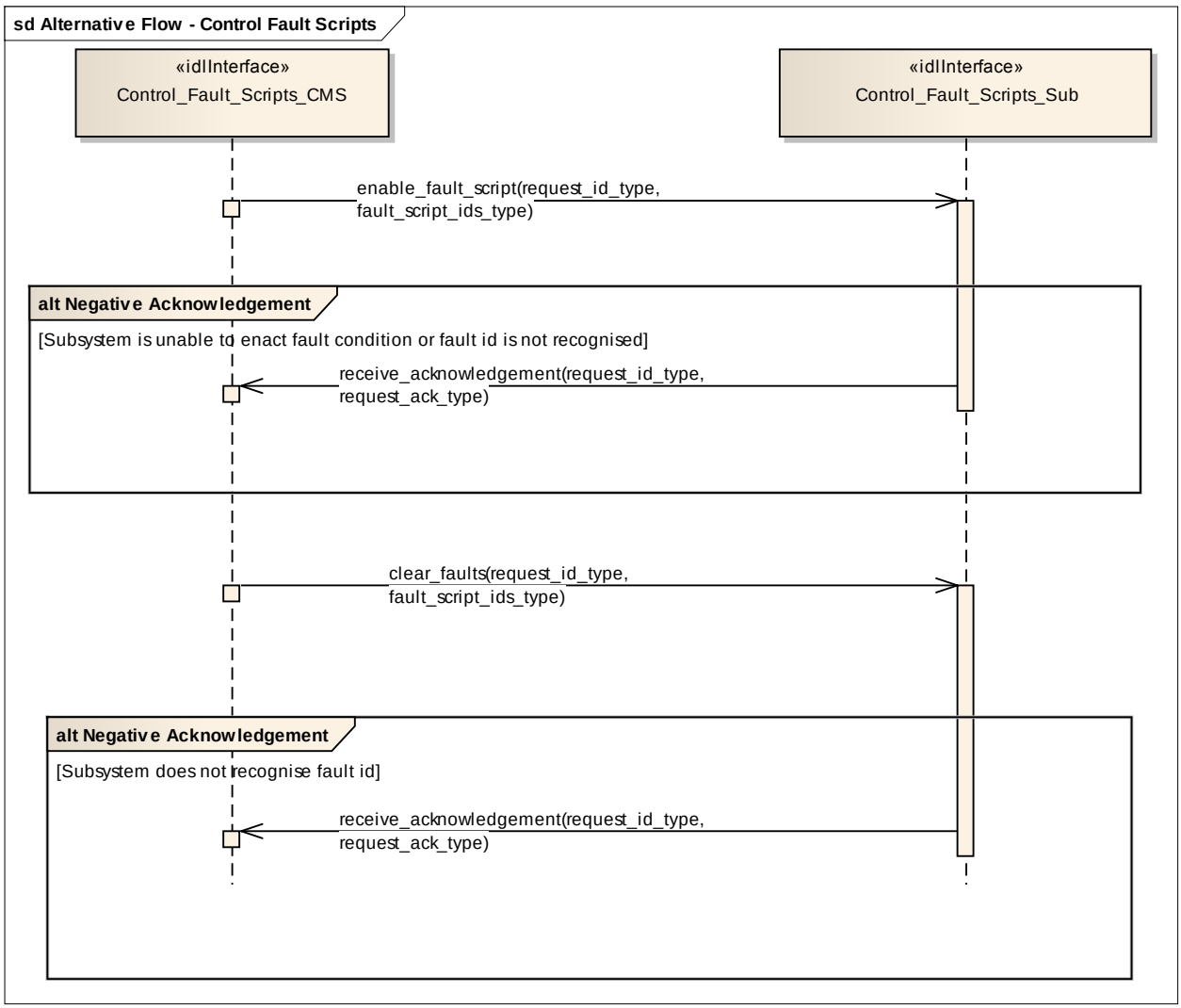

Figure 7.72 Alternative Flow - Control Fault Scripts (Sequence diagram)

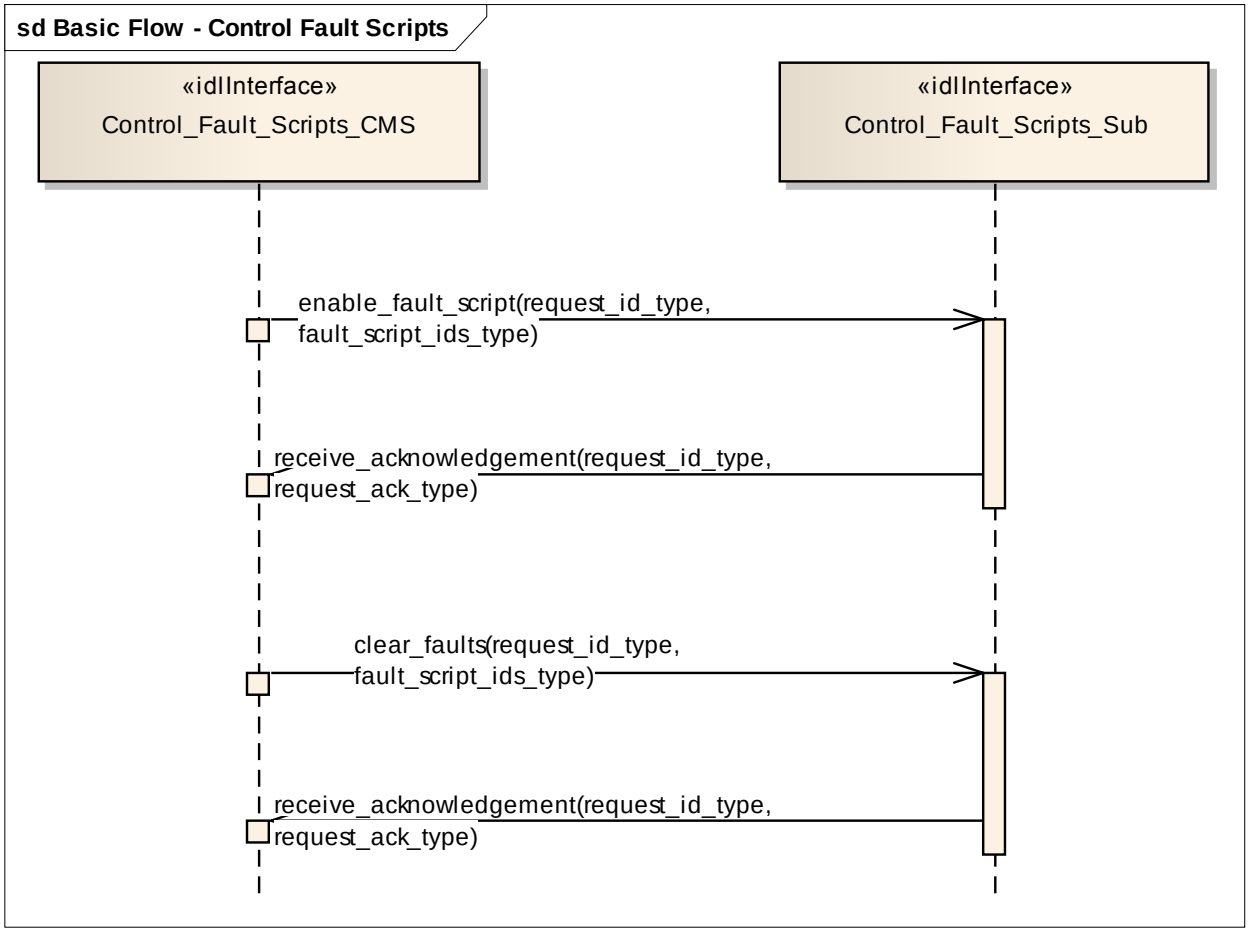

Figure 7.73 Basic Flow - Control Fault Scripts (Sequence diagram)

7.7.5 Subsystem_Control

Parent Package: Subsystem_Services
Contains interfaces for the Subsystem Control service.

7.7.5.1 Manage Technical State

Parent Package: Subsystem_Control
Contains operations and sequence diagrams for the Manage Technical State interface.

7.7.5.1.1 Manage_Technical_State_CMS

Type: IDLInterface common_use_case_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 7.151 - Methods of IDLInterface 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**

7.7.5.1.2 Manage_Technical_State_Sub

Type: IDLInterface
Package: Manage Technical State

Table 7.152 - Methods of IDLInterface 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**

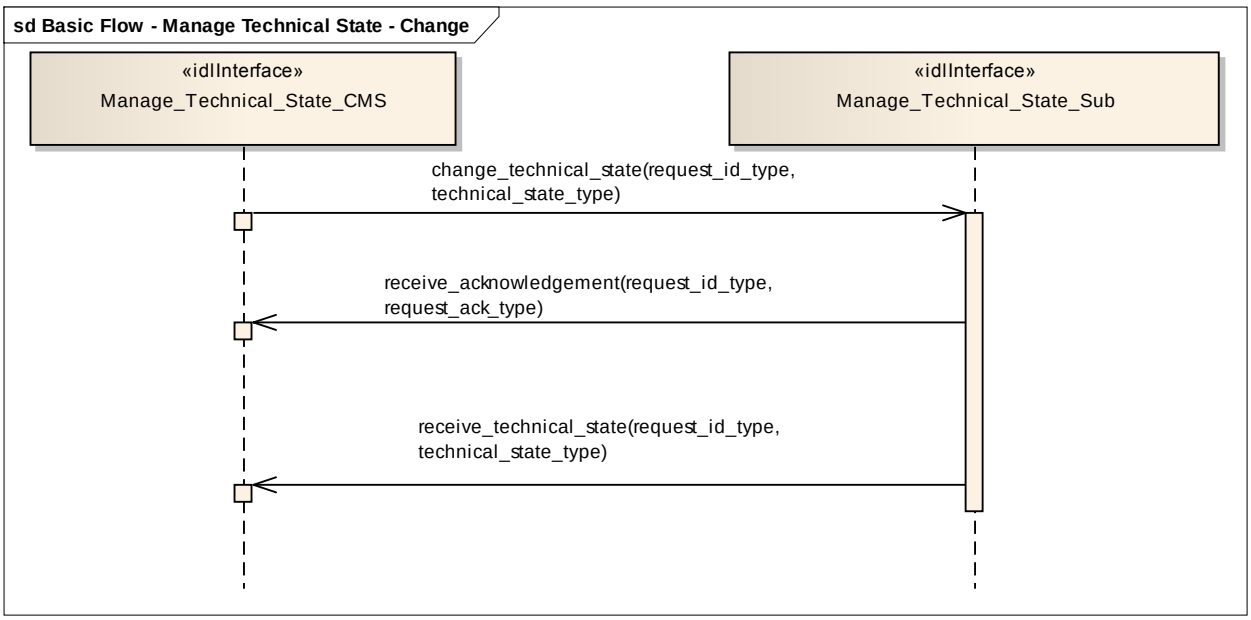

Figure 7.74 Basic Flow - Manage Technical State - Change (Sequence diagram)

Flow of events which depicts the CMS requesting that the subsystem changing its current technical state.

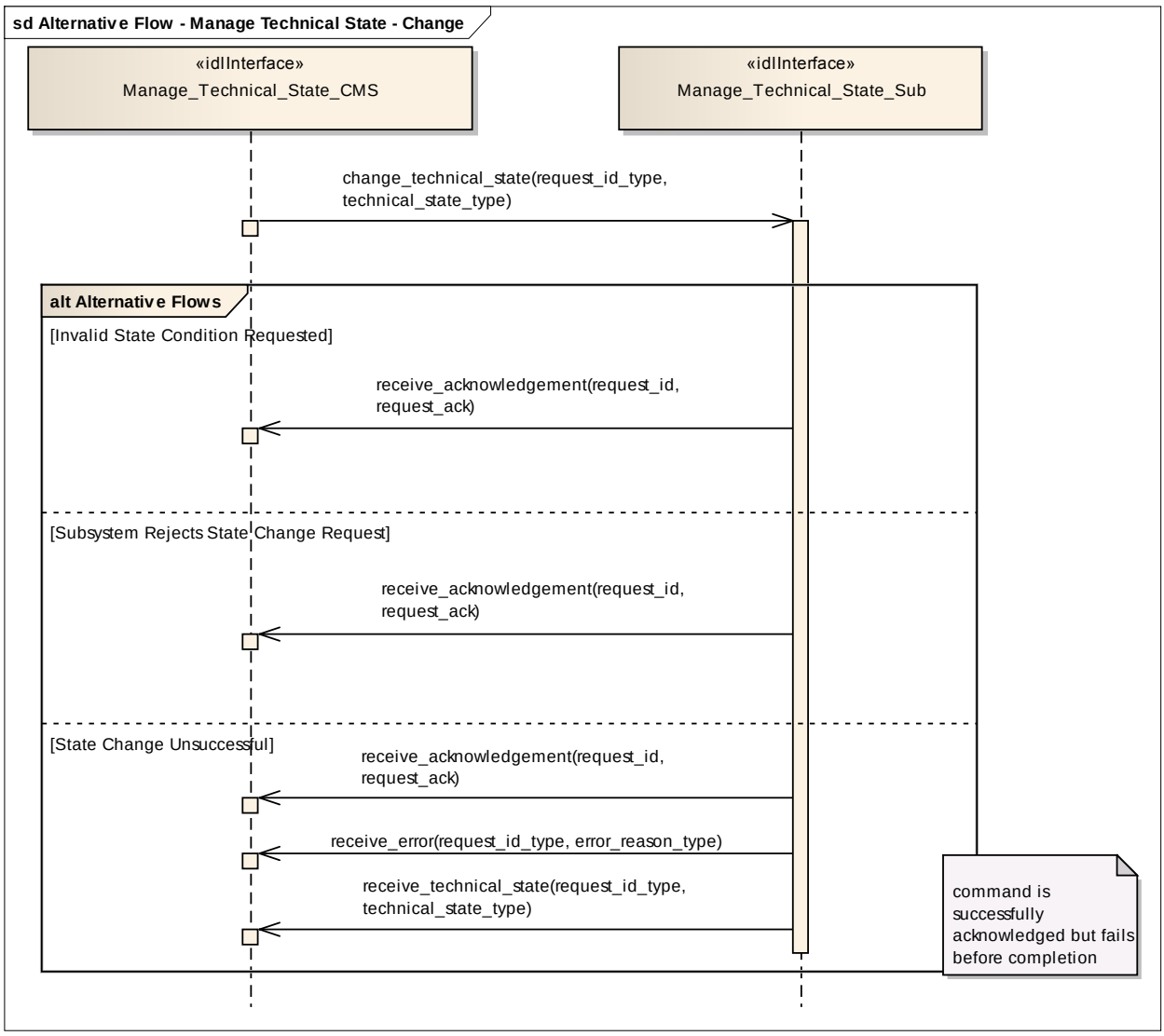

Figure 7.75 Alternative Flow - Manage Technical State - Change (Sequence diagram)

Alternate flow depicting rejection and error cases for a CMS requesting the subsystem to change its Technical State.

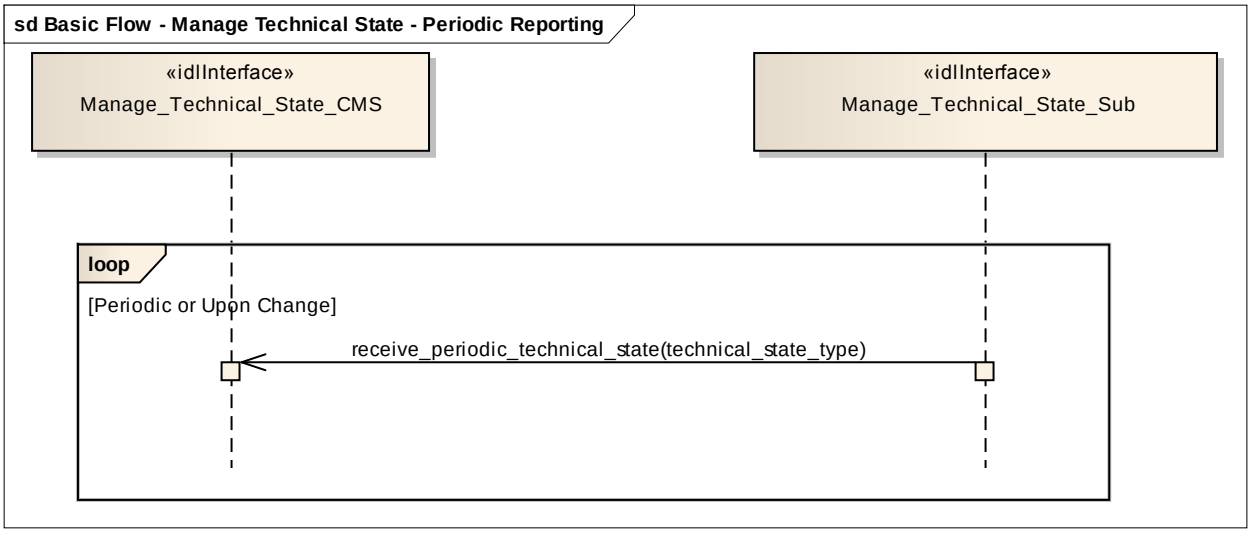

Figure 7.76 Basic Flow - Manage Technical State - Periodic Reporting (Sequence diagram)

Flow of events which depicts a subsystem that periodically reports its technical state (without the need for a CMS request).

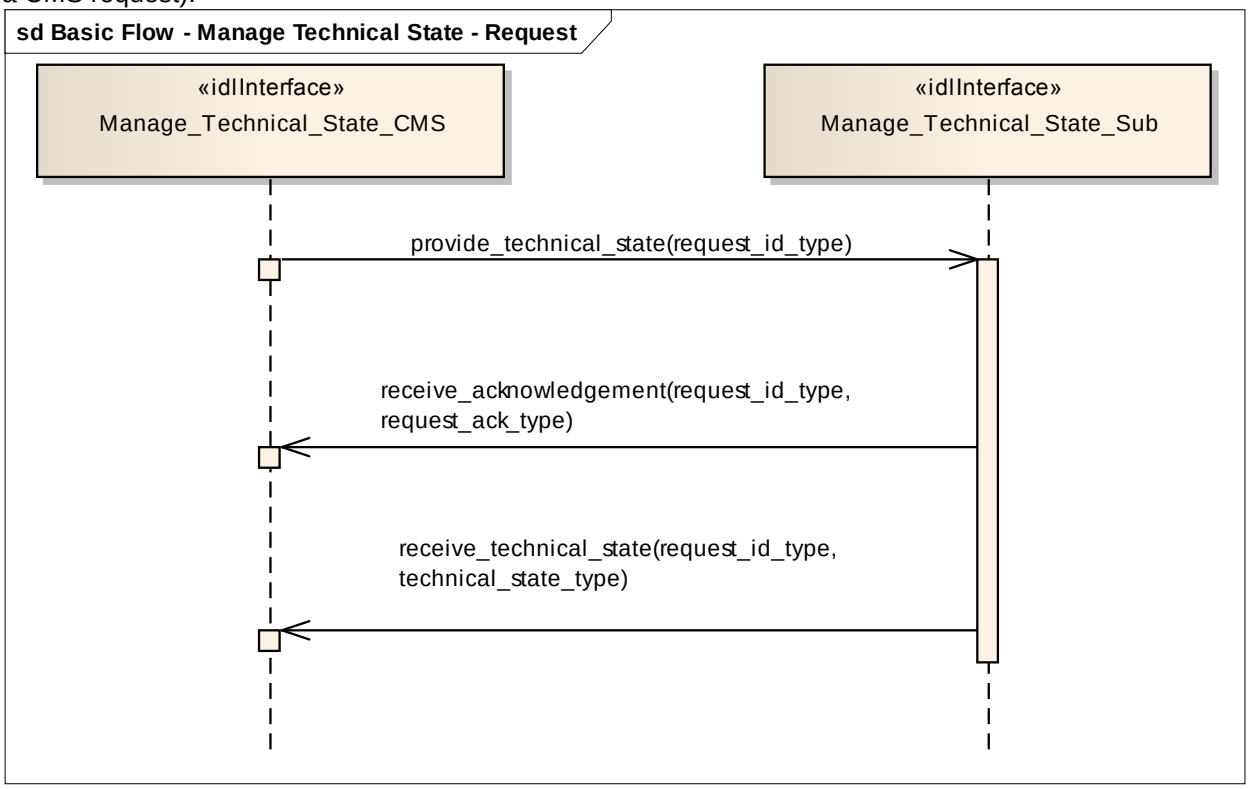

Figure 7.77 Basic Flow - Manage Technical State - Request (Sequence diagram)

Flow of events which depicts the CMS requesting that the subsystem report on its current technical state.

7.7.5.2 Heartbeat_Signal

Parent Package: Subsystem_Control

7.7.5.2.1 Heartbeat_Signal_CMS

Type: IDLInterface
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 7.153 - Methods of IDLInterface 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.

7.7.5.2.2 Heartbeat_Signal_Sub

Type: IDLInterface
Package: Heartbeat_Signal

Table 7.154 - Methods of IDLInterface 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.

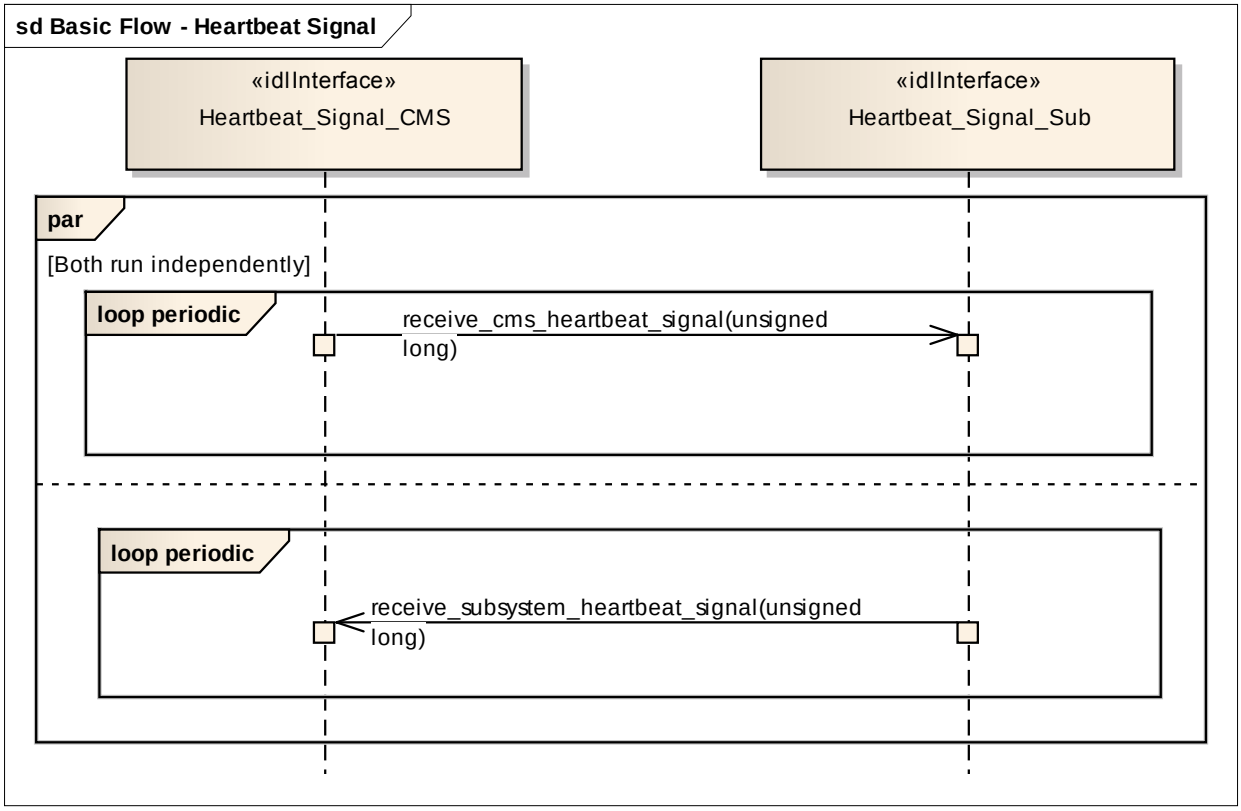

Figure 7.78 Basic Flow - Heartbeat Signal (Sequence diagram)

7.7.5.3 Provide_Subsystem_Identification

Parent Package: Subsystem_Control

7.7.5.3.1 Provide_Subsystem_Identification_CMS

Type: IDLInterface common_use_case_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 7.155 - Methods of IDLInterface 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**

7.7.5.3.2 Provide_Subsystem_Identification_Sub

Type: IDLInterface common_use_case_interface
Package: Provide_Subsystem_Identification

Table 7.156 - Methods of IDLInterface 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**

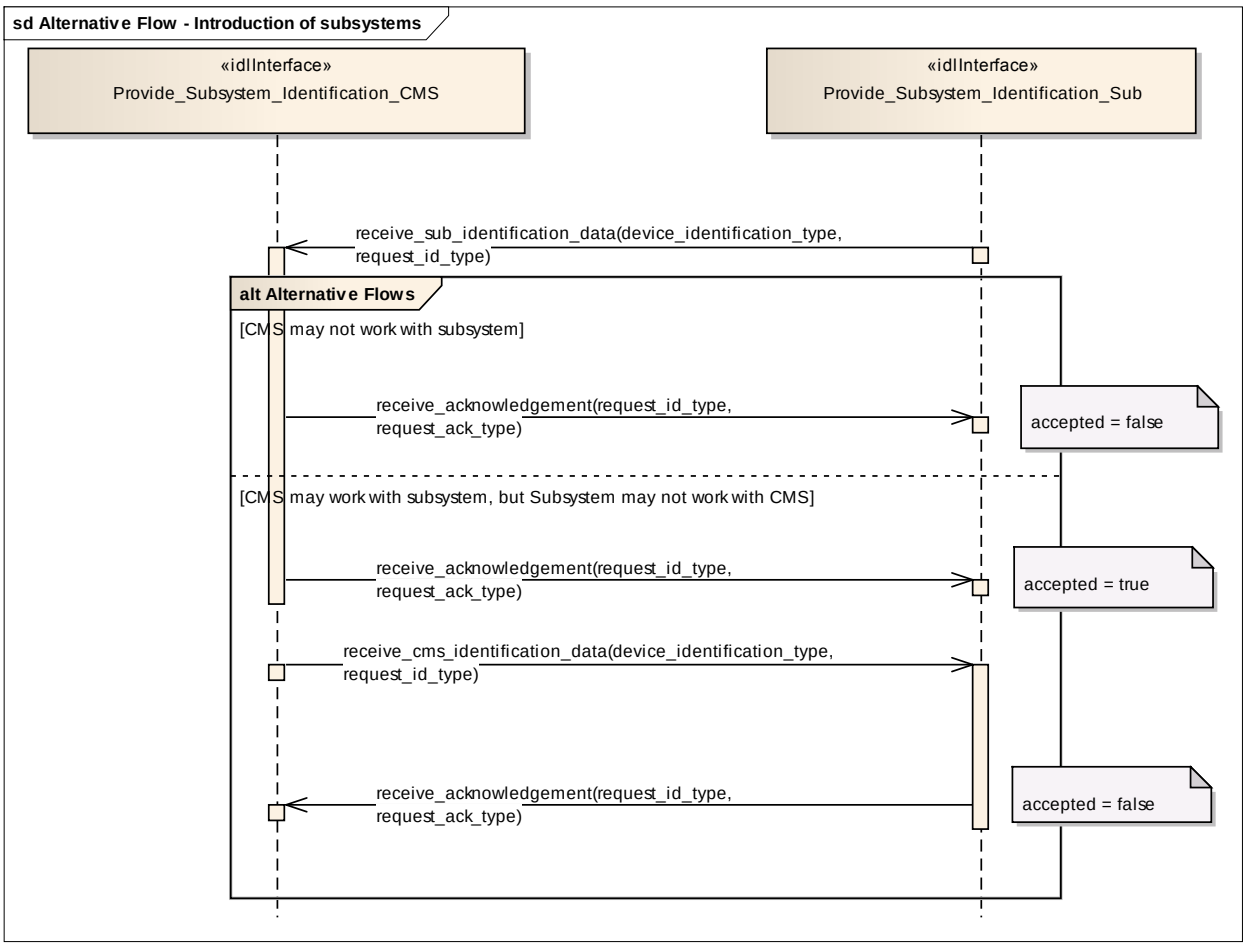

Figure 7.79 Alternative Flow - Introduction of subsystems (Sequence diagram)

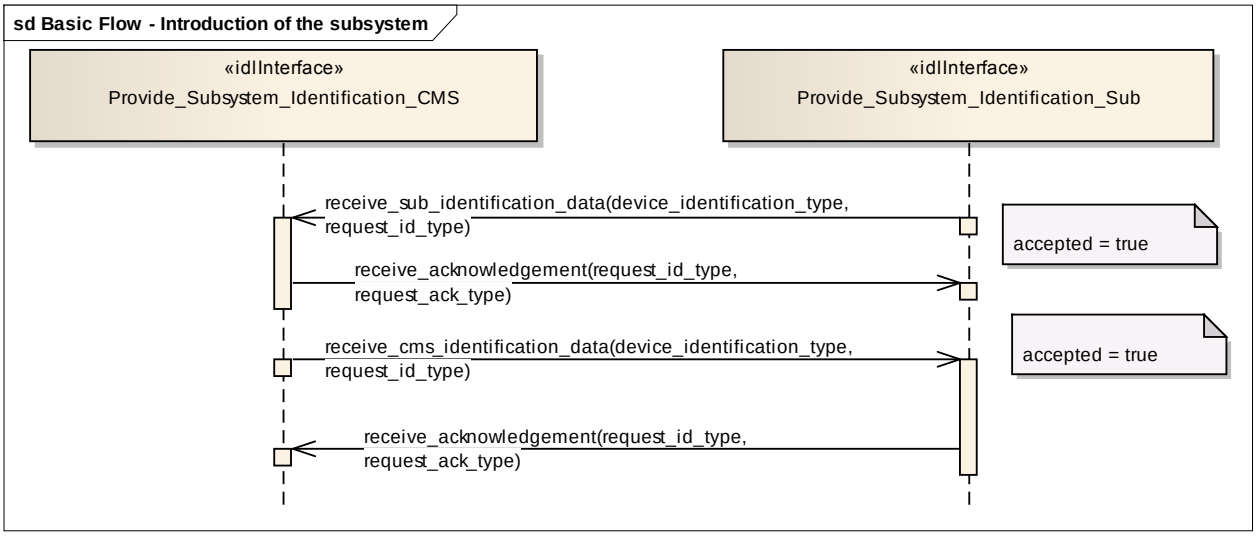

Figure 7.80 Basic Flow - Introduction of the subsystem (Sequence diagram)

7.7.5.4 Provide_Health_State

Parent Package: Subsystem_Control

7.7.5.4.1 Provide_Health_State_CMS

Type: IDLInterface common_use_case_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 or READY.
Post-condition: CMS awareness CMS is aware of the health state of the subsystem and/or its services.

Table 7.157 - Methods of IDLInterface Provide_Health_State_CMS

Method	Notes	Parameters
report_fault()	Report a fault to CMS	fault **the_fault**
report_service_health()	Report health of service	request_id_type **request_id** service_health_type **health** fault_list **the_fault_list**
report_subsystem_health()	Report health of subsystem	request_id_type **request_id** subsystem_health_type **health**

7.7.5.4.2 Provide_Health_State_Sub

Type: IDLInterface
Package: Provide_Health_State

Table 7.158 - Methods of IDLInterface 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**

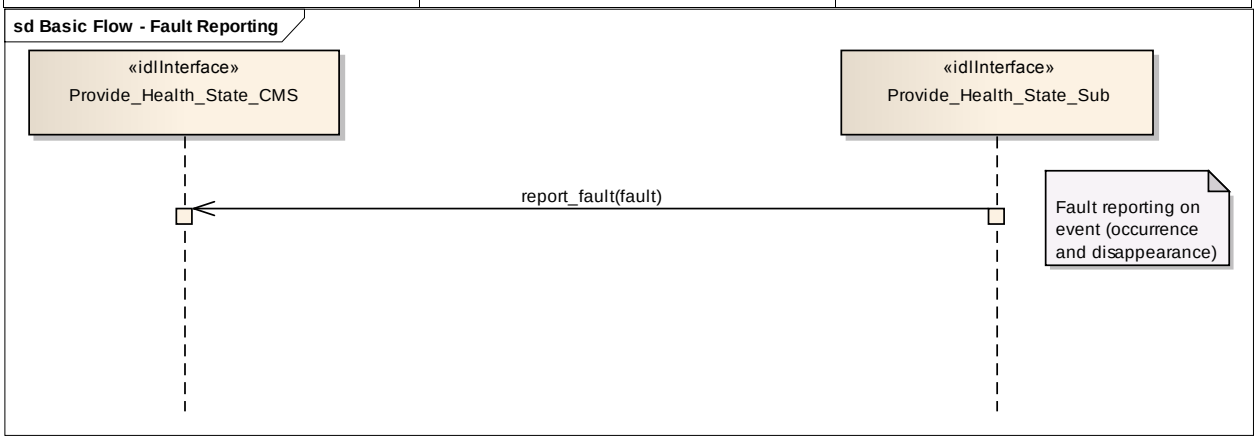

Figure 7.81 Basic Flow - Fault Reporting (Sequence diagram)

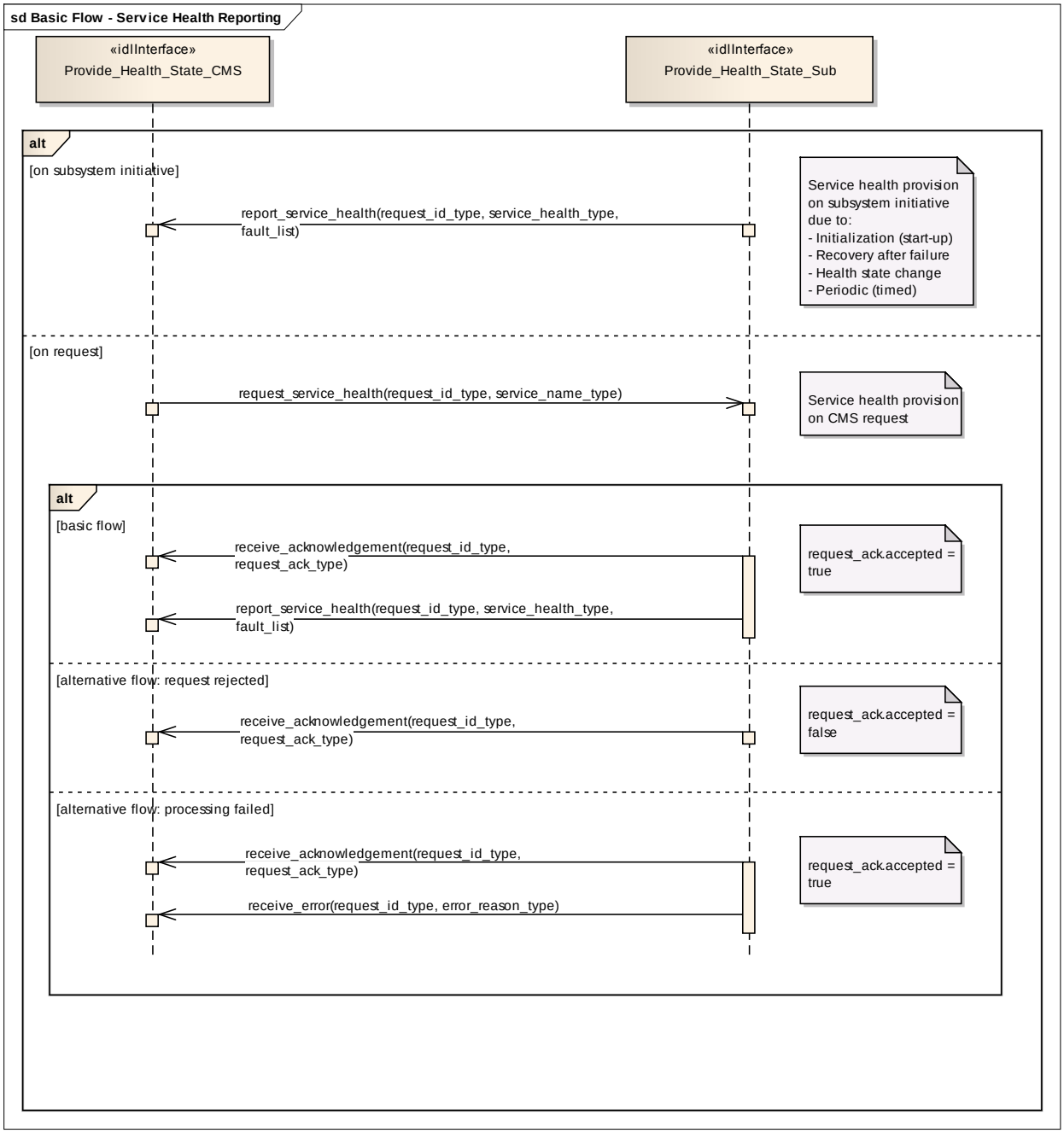

Figure 7.82 Basic Flow - Service Health Reporting (Sequence diagram)

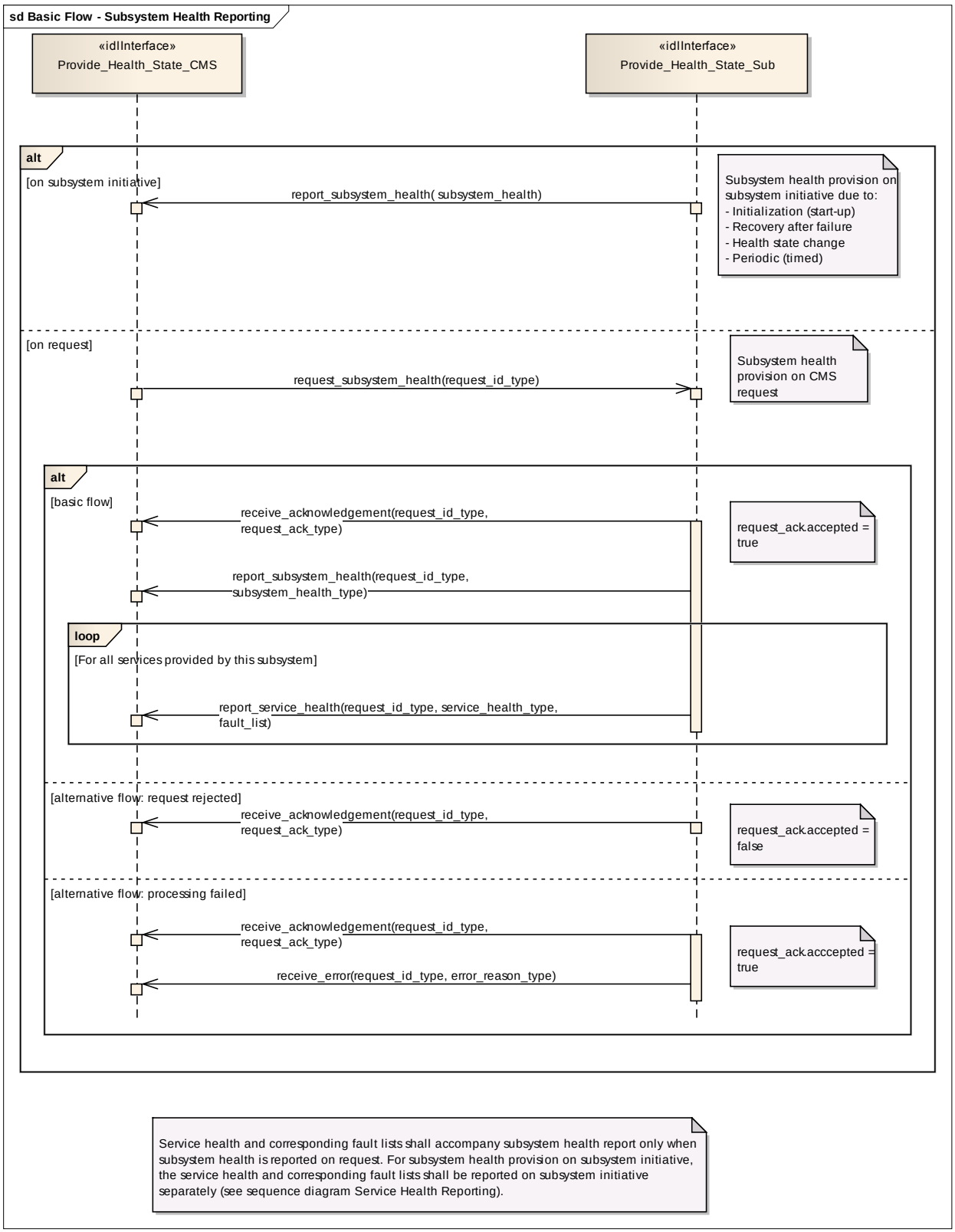

Figure 7.83 Basic Flow - Subsystem Health Reporting (Sequence diagram)

7.7.5.5 Manage_Operational_Mode

Parent Package: Subsystem_Control

7.7.5.5.1 Manage_Operational_Mode_CMS

Type: IDLInterface common_use_case_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 7.159 - Methods of IDLInterface 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**

7.7.5.5.2 Manage_Operational_Mode_Sub

Type: IDLInterface
Package: Manage_Operational_Mode

Table 7.160 - Methods of IDLInterface 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**

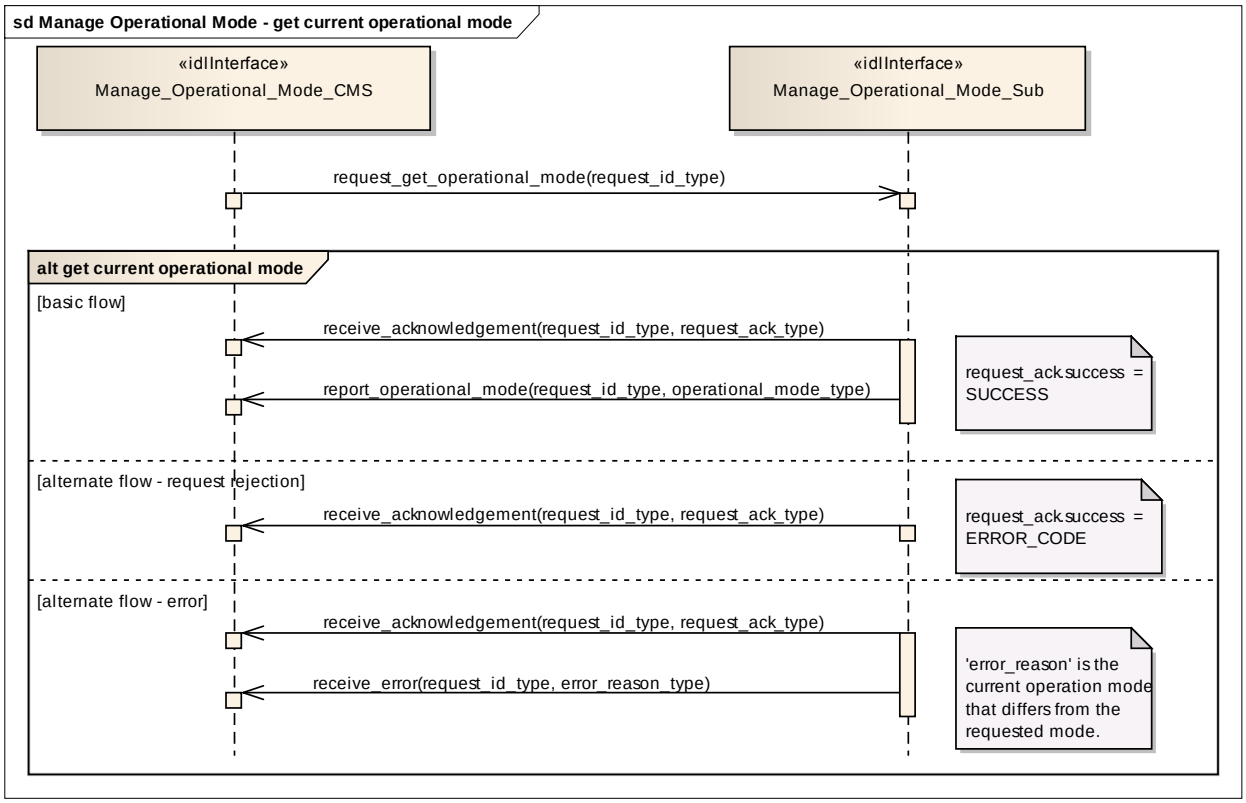

Figure 7.84 Manage Operational Mode - get current operational mode (Sequence diagram)

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".

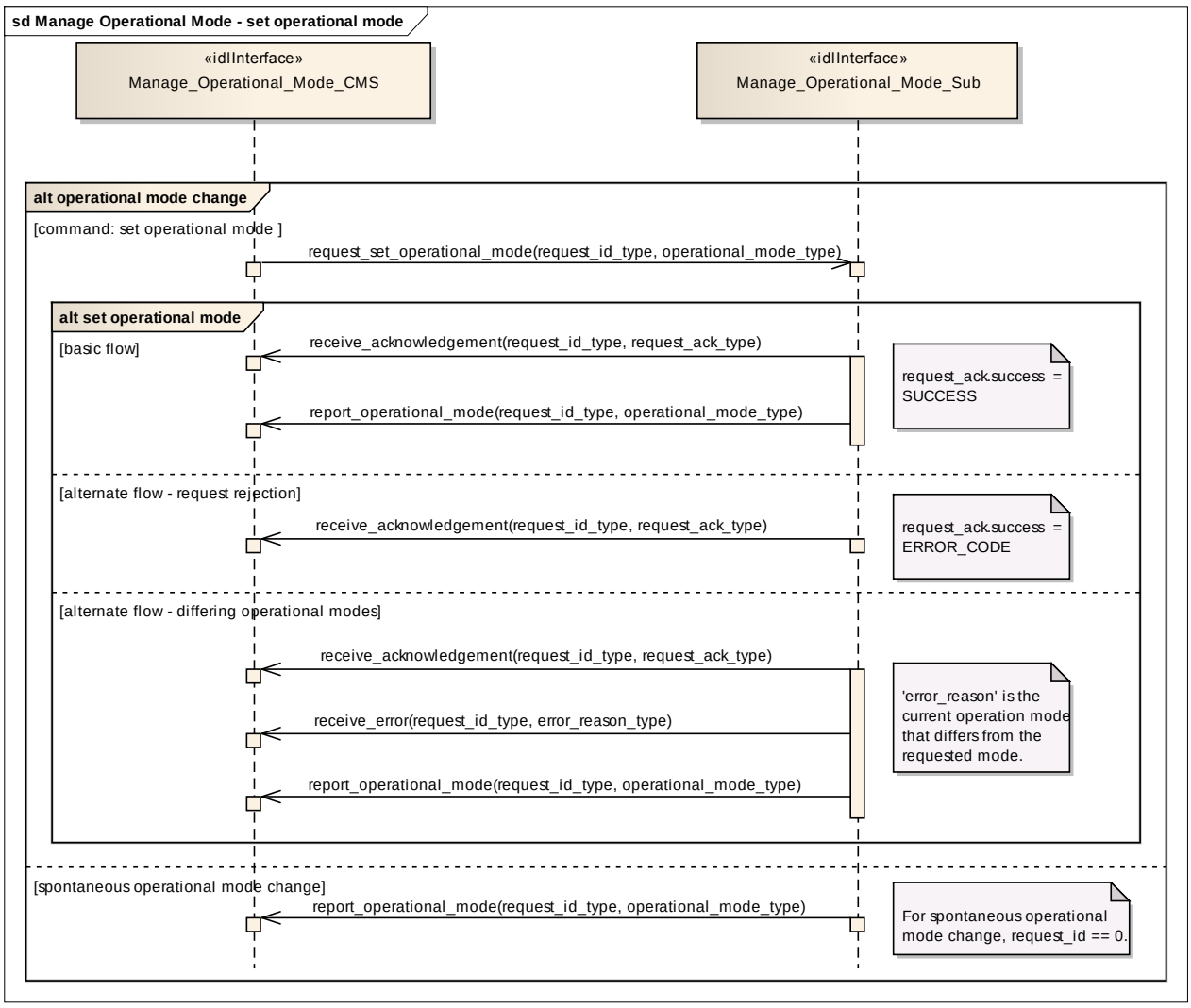

Figure 7.85 Manage Operational Mode - set operational mode (Sequence diagram)

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".

7.7.5.6 Control_Battle_Override

Parent Package: Subsystem_Control

This package contains interfaces for the Control Battle Override service.

7.7.5.6.1 Control_Battle_Override_CMS

Type: IDLInterface common_use_case_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 7.161 - Methods of IDLInterface 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**

7.7.5.6.2 Control_Battle_Override_Sub

Type: IDLInterface
Package: Control_Battle_Override

Table 7.162 - Methods of IDLInterface 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 7.86 Basic Flow - Control Battle Override - Set/Reset (Sequence diagram)

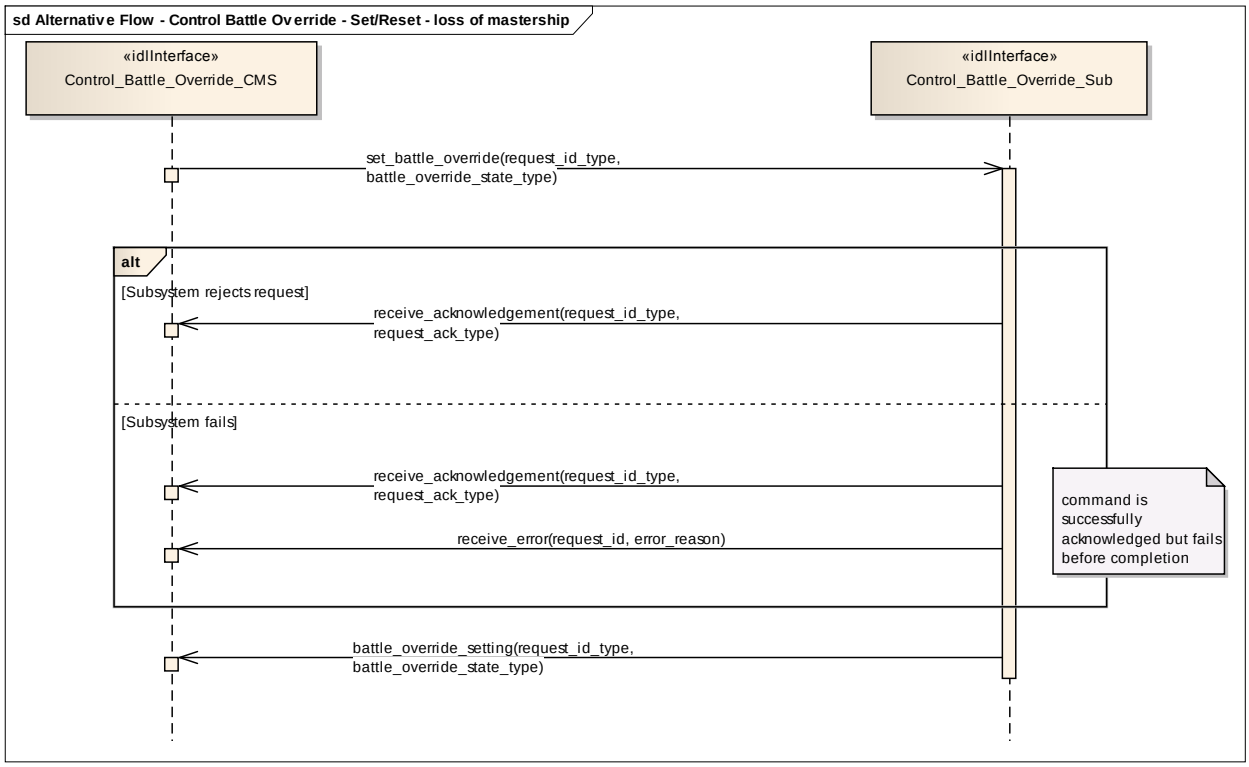

Figure 7.87 Alternative Flow - Control Battle Override - Set/Reset - loss of mastership (Sequence diagram)

7.7.5.7 Manage_Subsystem_Parameters

Parent Package: Subsystem_Control

7.7.5.7.1 Manage_Subsystem_Parameters_CMS

Type: IDLInterface common_use_case_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 7.163 - Methods of IDLInterface 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 **the_descriptor_sequence** name_error_sequence_type **the_name_error_set**

7.7.5.7.2 Manage_Subsystem_Parameters_Sub

Type: IDLInterface
Package: Manage_Subsystem_Parameters

Table 7.164 - Methods of IDLInterface 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**

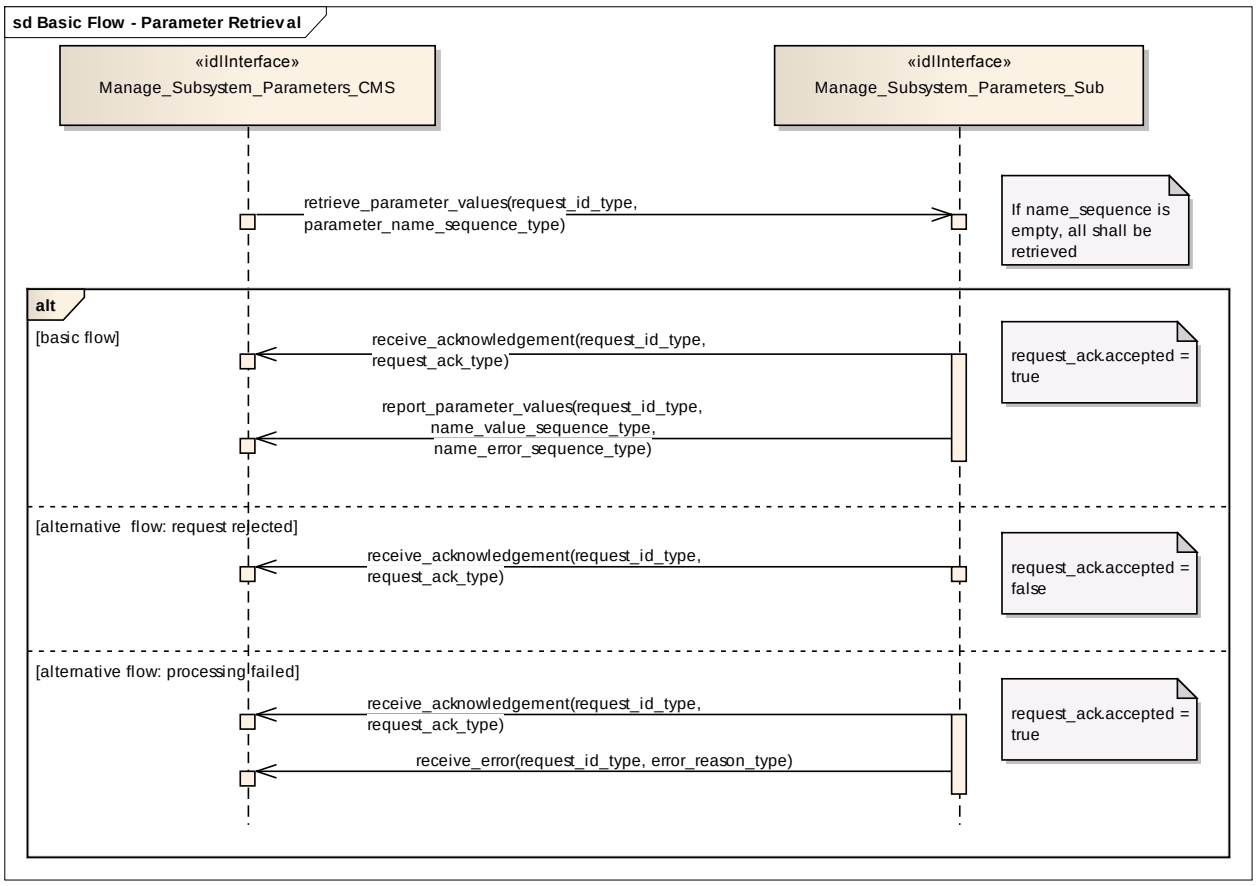

Figure 7.88 Basic Flow - Parameter Retrieval (Sequence diagram)

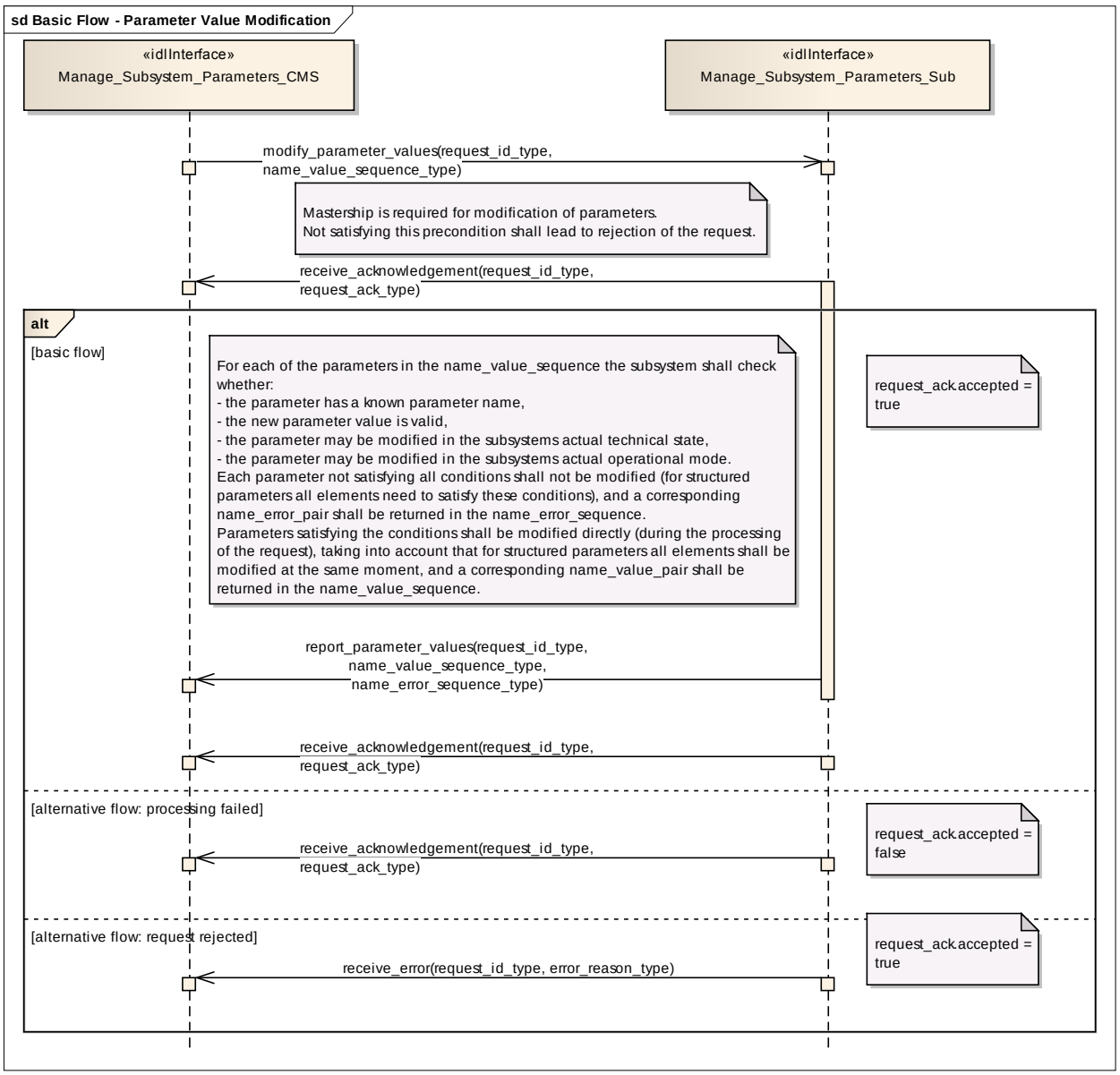

Figure 7.89 Basic Flow - Parameter Value Modification (Sequence diagram)

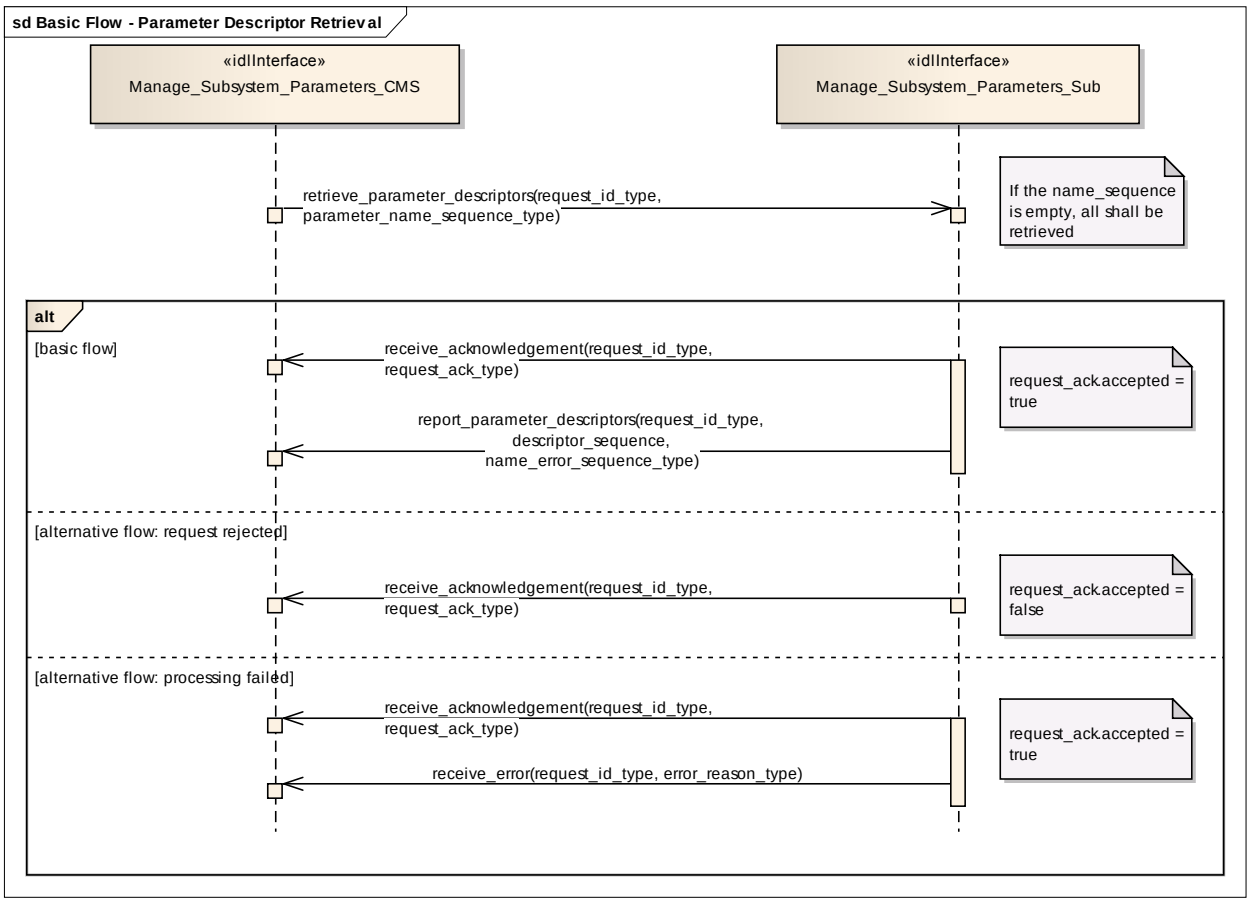

Figure 7.90 Basic Flow - Parameter Descriptor Retrieval (Sequence diagram)

7.7.5.8 Provide_Subsystem_Services

Parent Package: Subsystem_Control

7.7.5.8.1 Provide_Subsystem_Services_CMS

Type: Interface common_use_case_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 7.165 - 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**

7.7.5.8.2 Provide_Subsystem_Services_Sub

Type: Interface common_use_case_interface
Package: Provide_Subsystem_Services

Table 7.166 - 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**

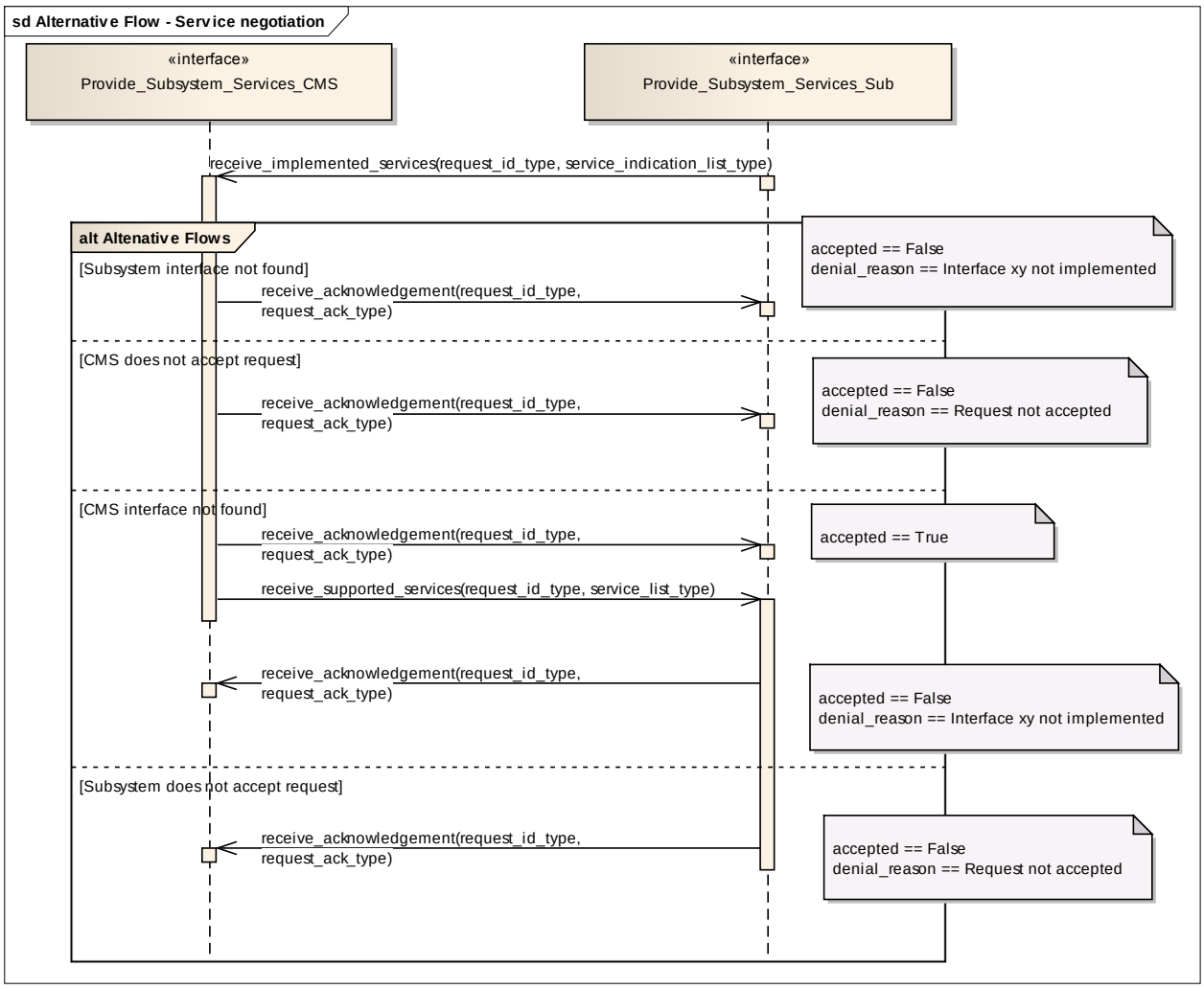

Figure 7.91 Alternative Flow - Service negotiation (Sequence diagram)

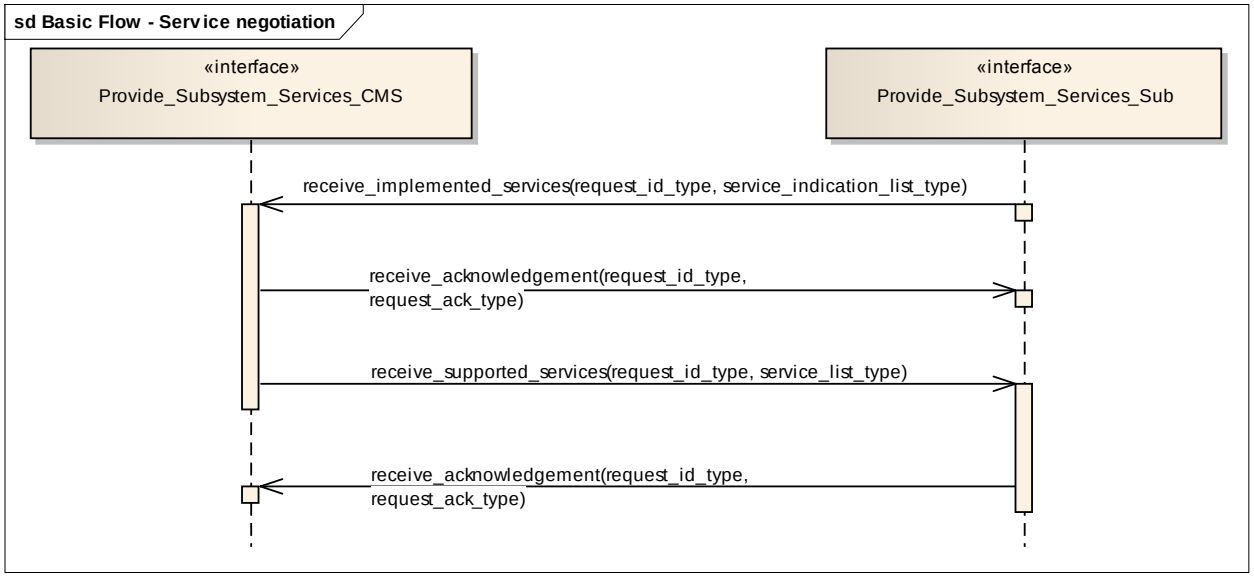

Figure 7.92 Basic Flow - Service negotiation (Sequence diagram)

7.7.5.9 Manage_Mastership

Parent Package: Subsystem_Control
This package contains interfaces for the Manage Mastership service.

7.7.5.9.1 Manage_Mastership_CMS

Type: IDLInterface common_use_case_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 7.167 - Methods of IDLInterface 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**

7.7.5.9.2 Manage_Mastership_Sub

Type: IDLInterface
Package: Manage_Mastership

Table 7.168 - Methods of IDLInterface 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.
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.

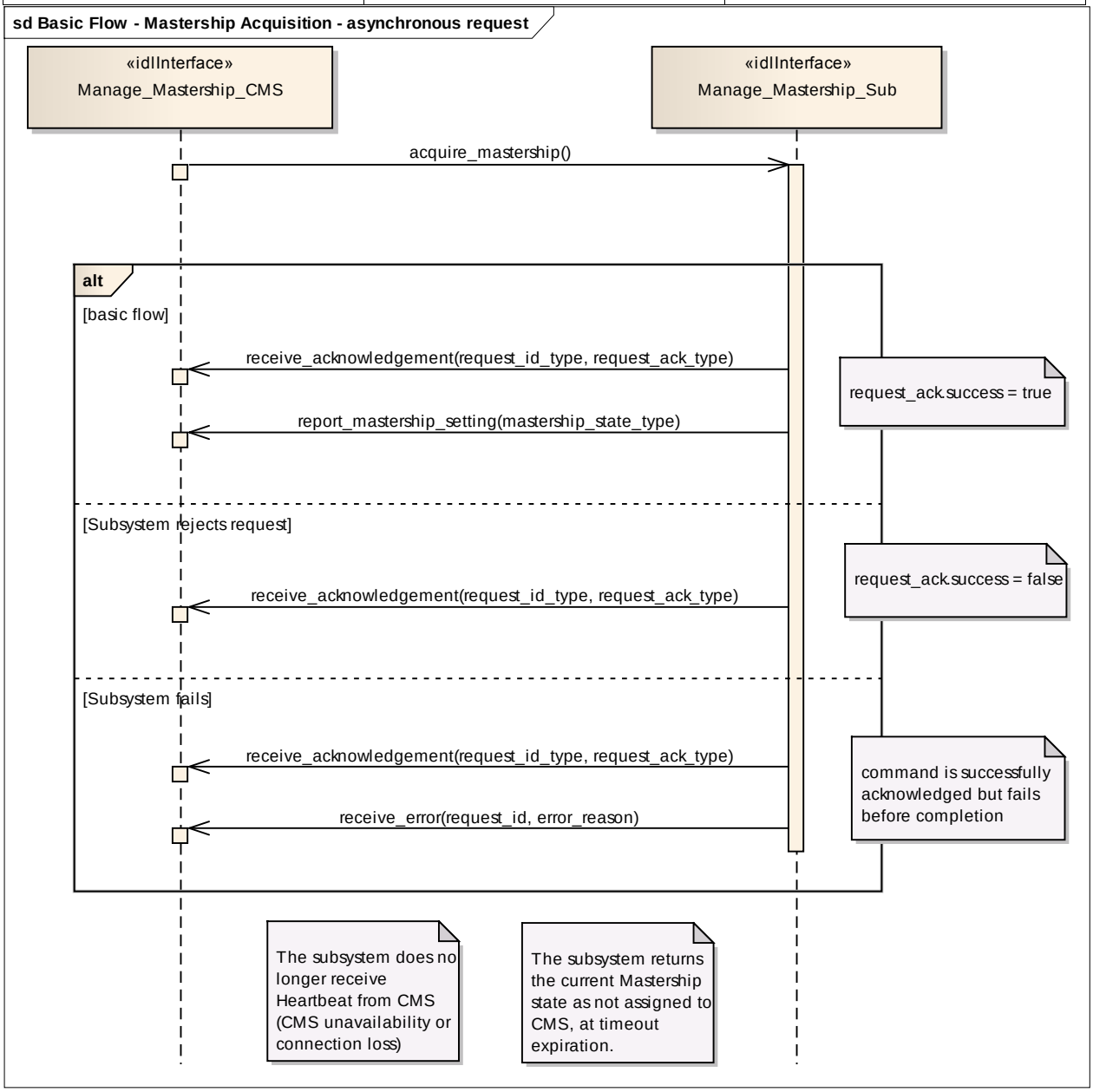

Figure 7.93 Basic Flow - Mastership Acquisition - asynchronous request (Sequence diagram)

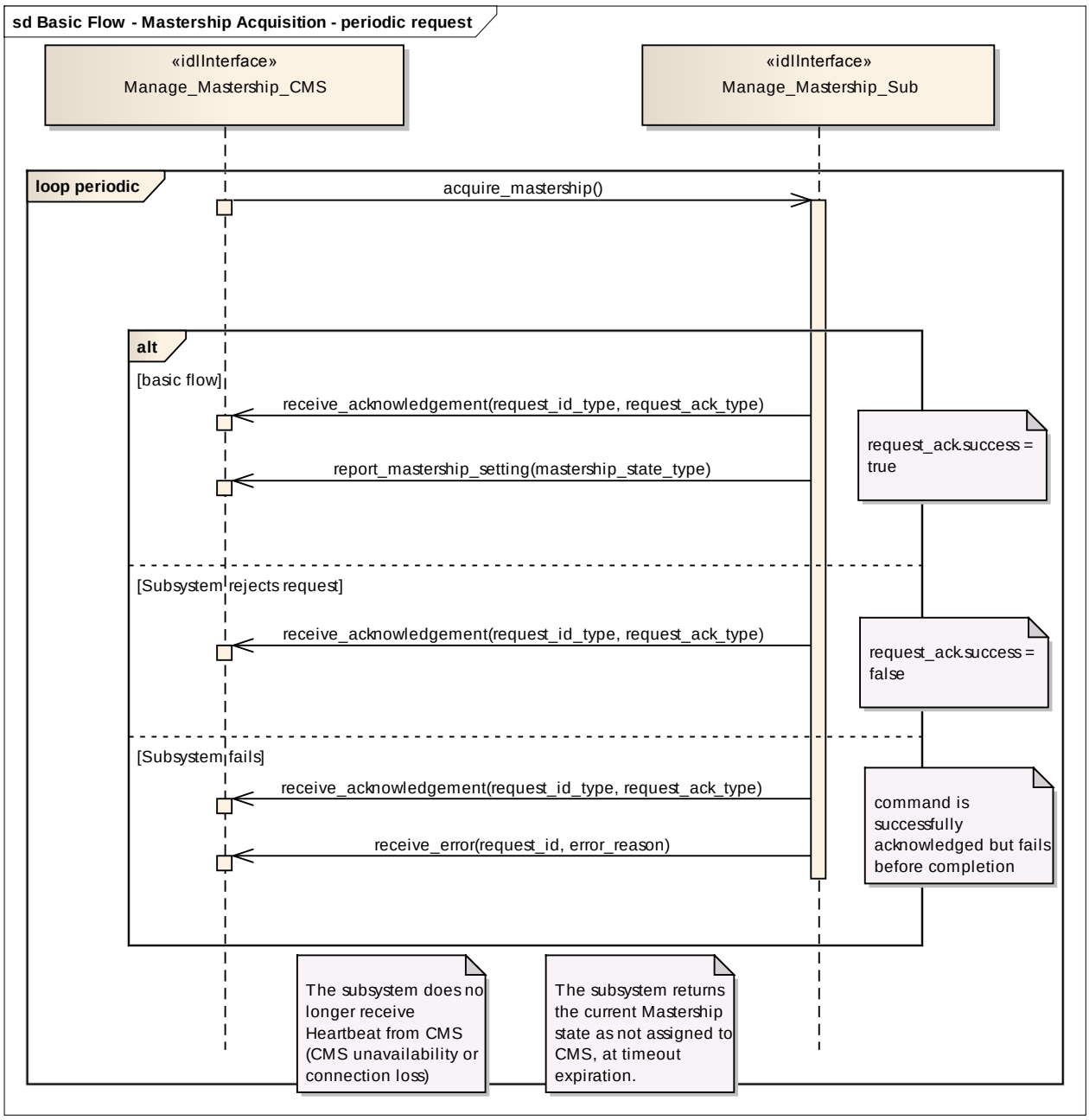

Figure 7.94 Basic Flow - Mastership Acquisition - periodic request (Sequence diagram)

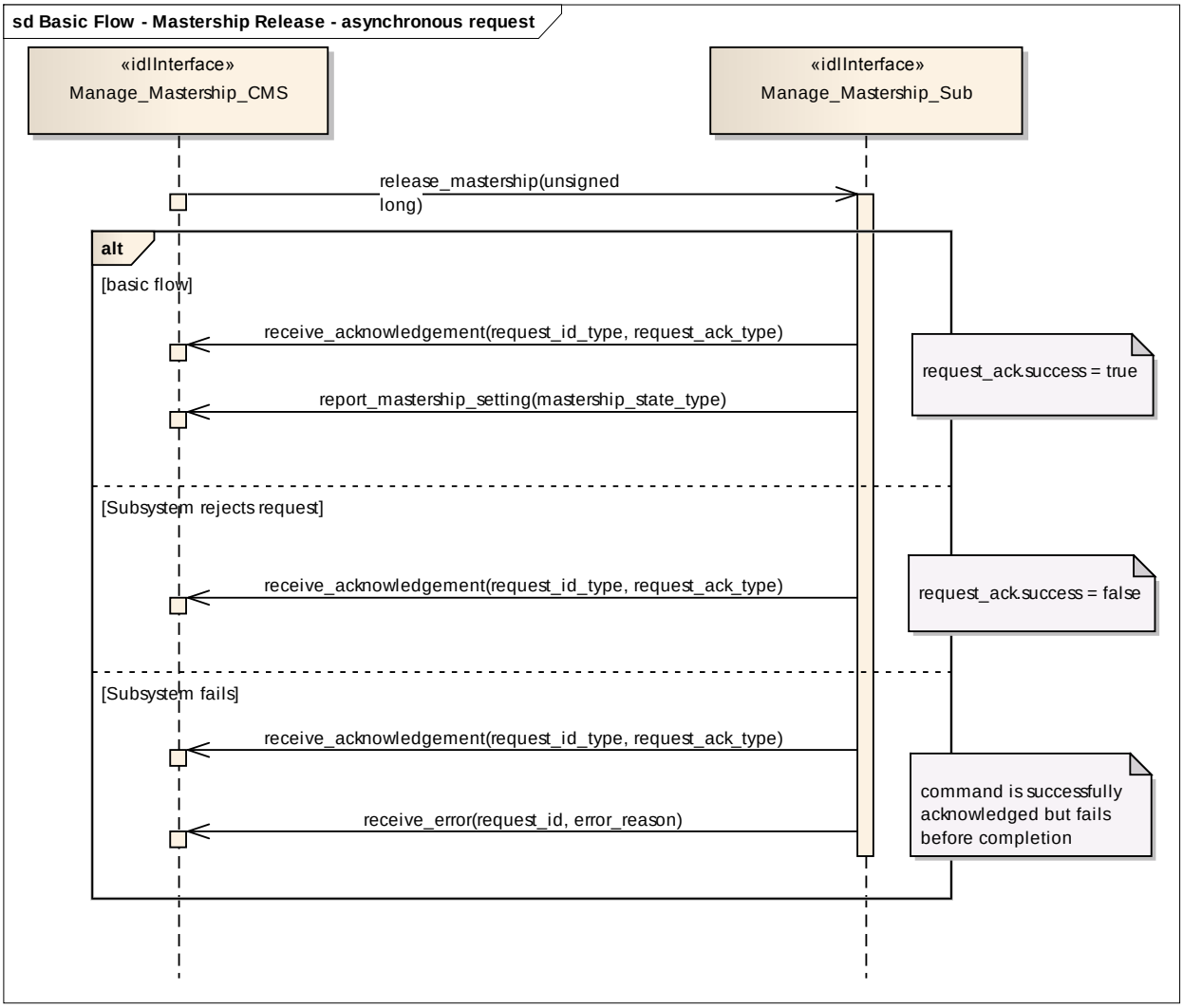

Figure 7.95 Basic Flow - Mastership Release - asynchronous request (Sequence diagram)

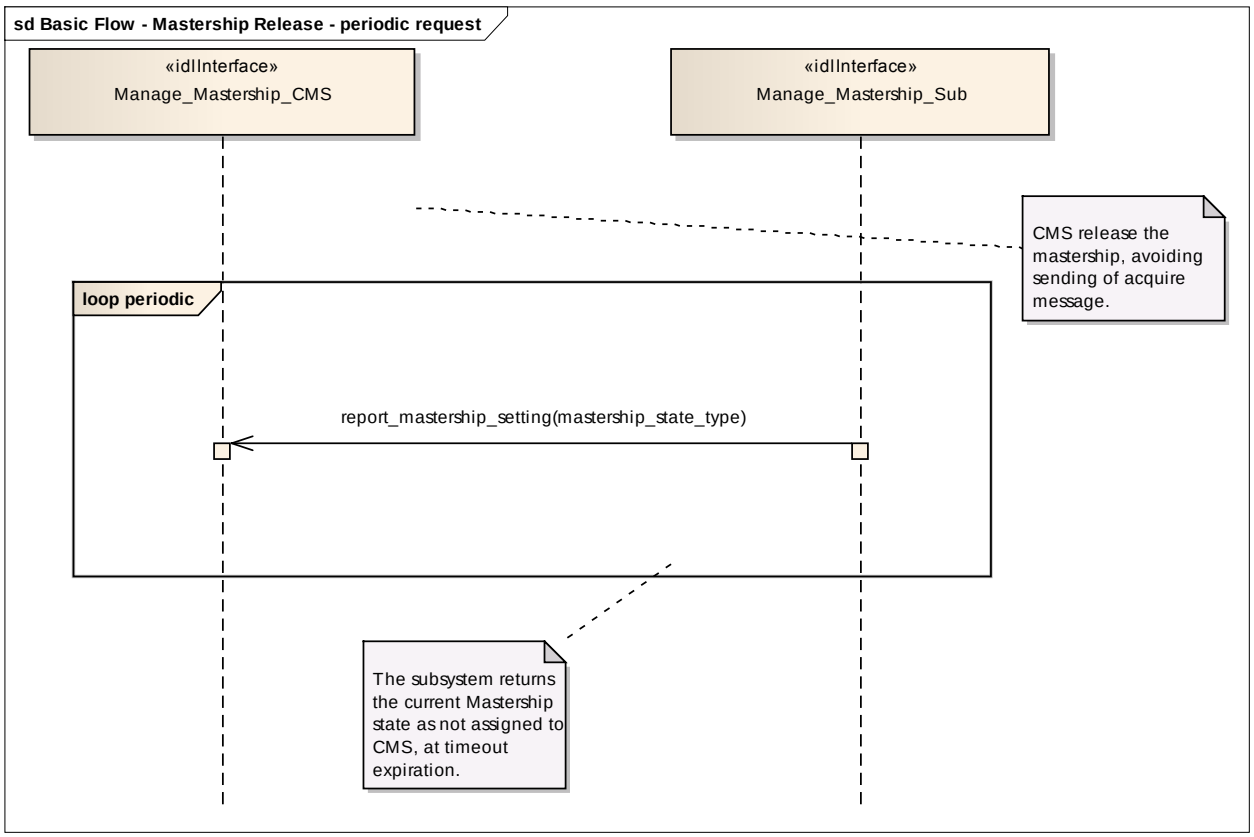

Figure 7.96 Basic Flow - Mastership Release - periodic request (Sequence diagram)

7.7.5.10 Register_Interest

Parent Package: Subsystem_Control

7.7.5.10.1 Register_Interest_CMS

Type: IDLInterface common_use_case_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 7.169 - Methods of IDLInterface Register_Interest_CMS

Method	Notes	Parameters
confirm_registration()	Confirm registration of interest	request_id_type **request_id**

7.7.5.10.2 Register_Interest_Sub

Type: IDLInterface
Package: Register_Interest

Table 7.170 - Methods of IDLInterface Register_Interest_Sub

Method	Notes	Parameters
register_interest()	Register interest in the service	request_id_type **request_id** interest_list **the_interest_list**

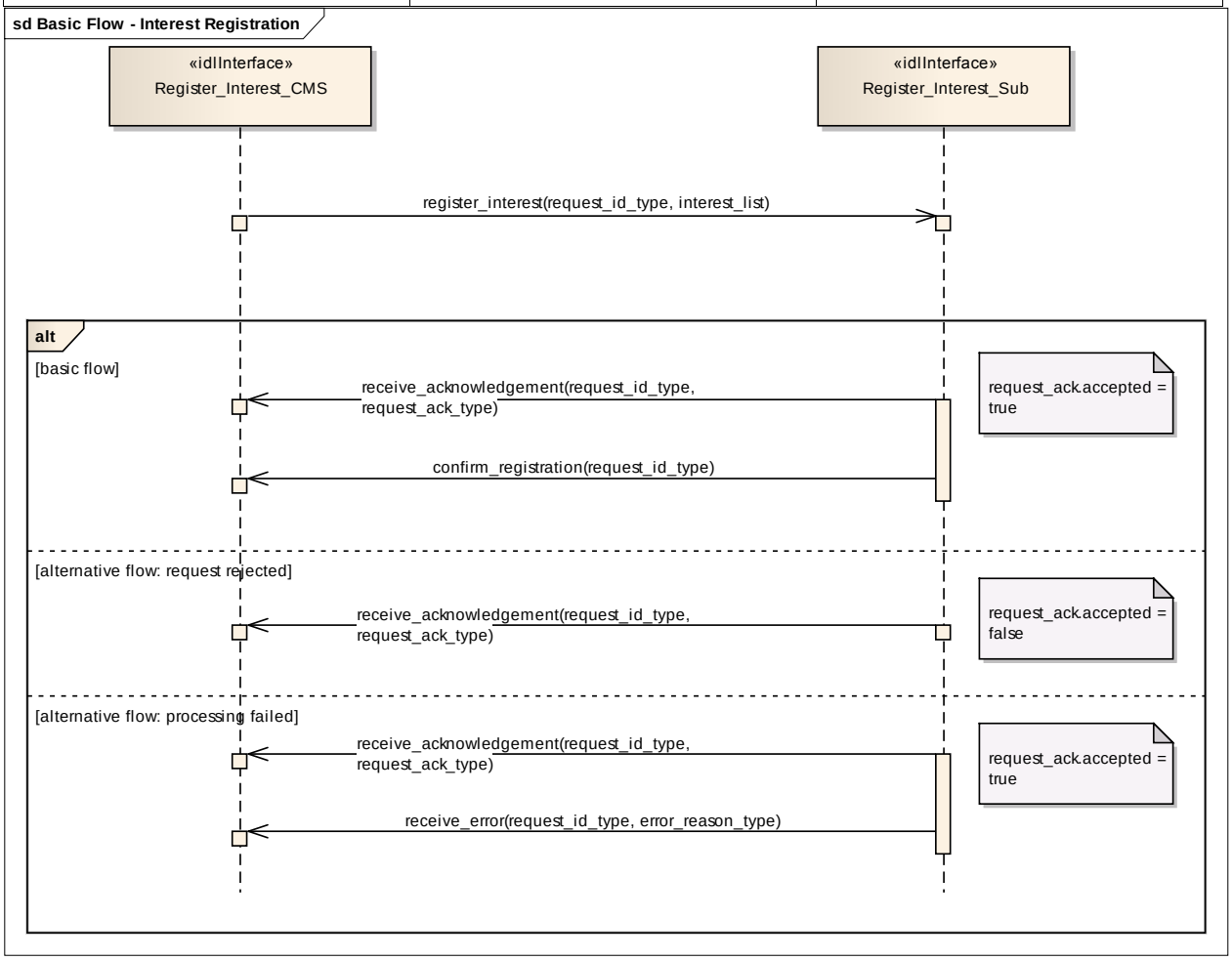

Figure 7.97 Basic Flow - Interest Registration (Sequence diagram)

7.8 Sensor_Services

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

7.8.1 Clutter_Reporting

Parent Package: Sensor_Services
Contains interfaces for the Clutter Reporting service.

7.8.1.1 Provide Area with Plot Concentration

Parent Package: Clutter_Reporting
Contains operations and sequence diagrams for the Provide Area with Plot Concentration interface.

7.8.1.1.1 Provide_Plot_Concentration_CMS

Type: IDLInterface common_use_case_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 7.171 - Methods of IDLInterface 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**

7.8.1.1.2 Provide_Plot_Concentration_Sub

Type: IDLInterface
Package: Provide Area with Plot Concentration

Table 7.172 - Methods of IDLInterface 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**

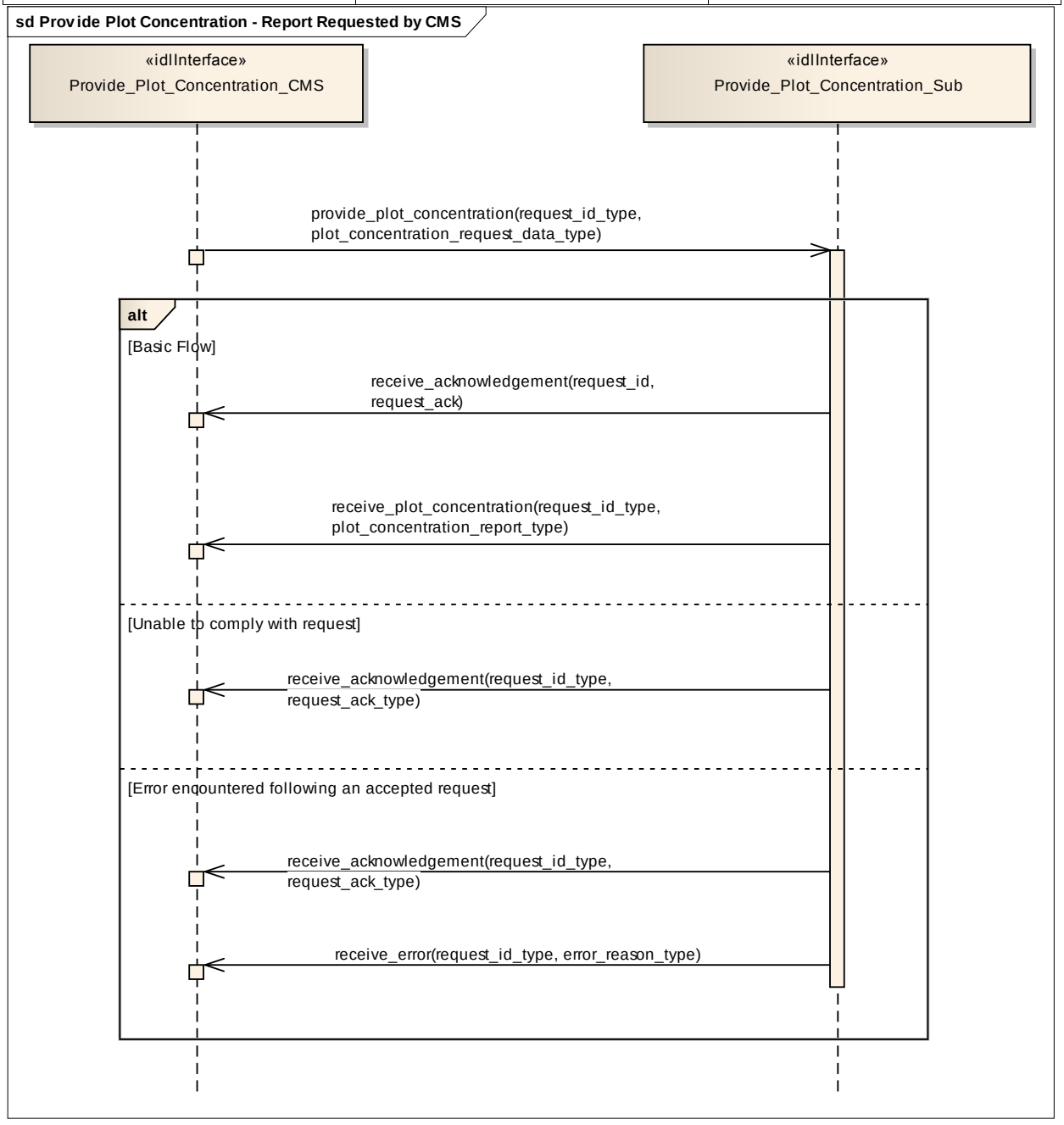

Figure 7.98 Provide Plot Concentration - Report Requested by CMS (Sequence diagram)

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).

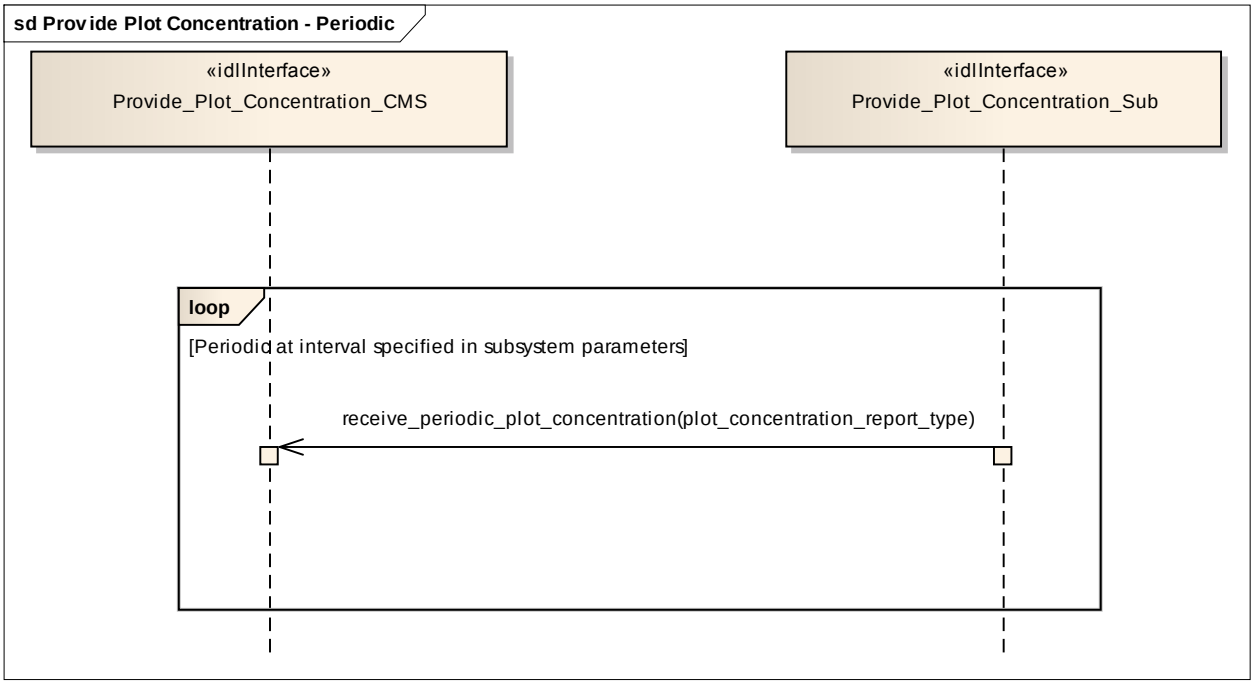

Figure 7.99 Provide Plot Concentration - Periodic (Sequence diagram)

Flow of events which depicts a subsystem that periodically reports plot concentration reports (without the need for a CMS request).

7.8.1.2 Provide Clutter Assessment

Parent Package: Clutter_Reporting
Contains operations and sequence diagrams for the Provide Clutter Assessment interface.

7.8.1.2.1 Provide_Clutter_Assessment_CMS

Type: IDLInterface common_use_case_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 7.173 - Methods of IDLInterface 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	clutter_report_type **clutter_report**

	from the subystem.	

7.8.1.2.2 Provide_Clutter_Assessment_Sub

Type: IDLInterface
Package: Provide Clutter Assessment

Table 7.174 - Methods of IDLInterface 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**

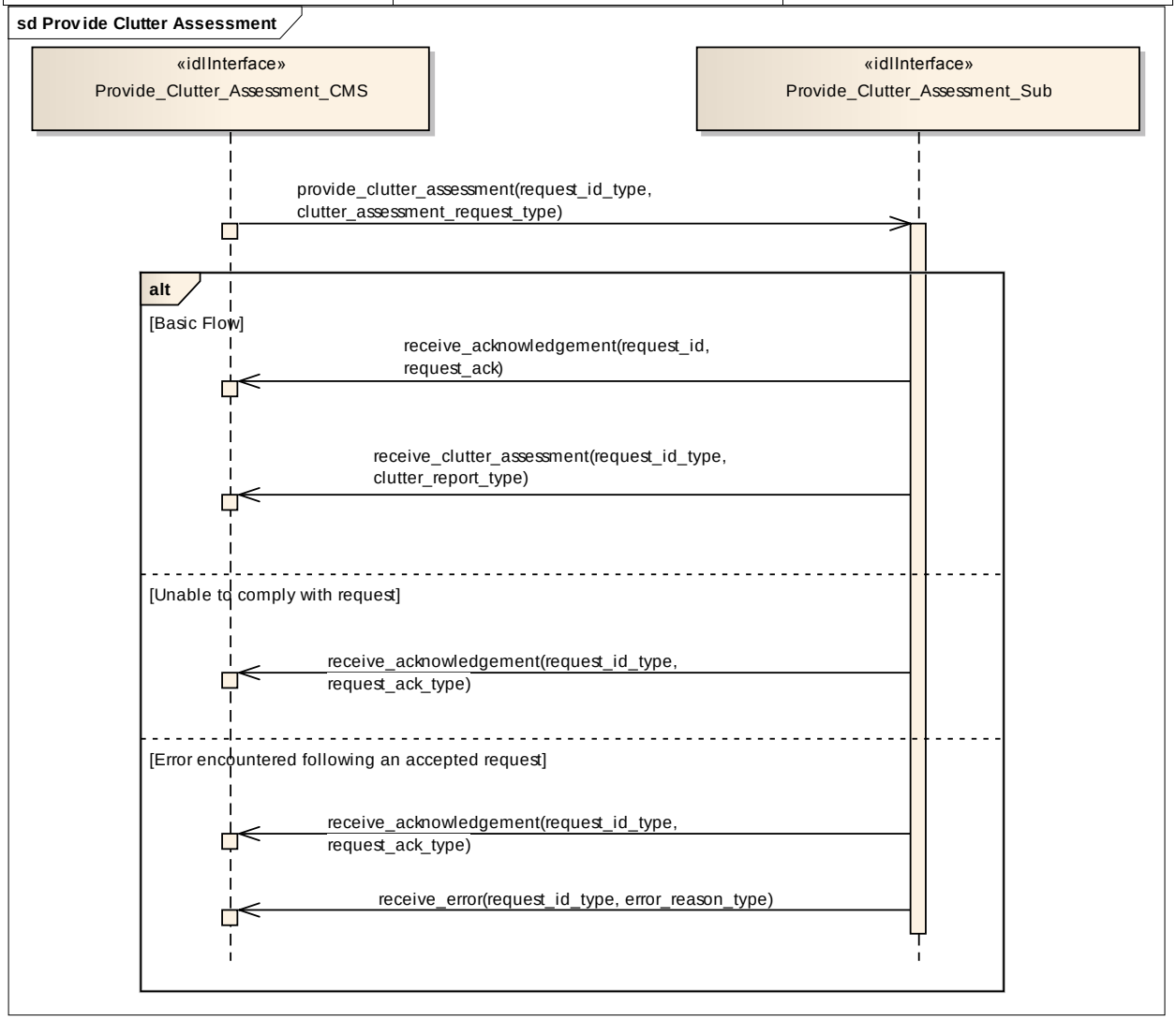

Figure 7.100 Provide Clutter Assessment (Sequence diagram)

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).

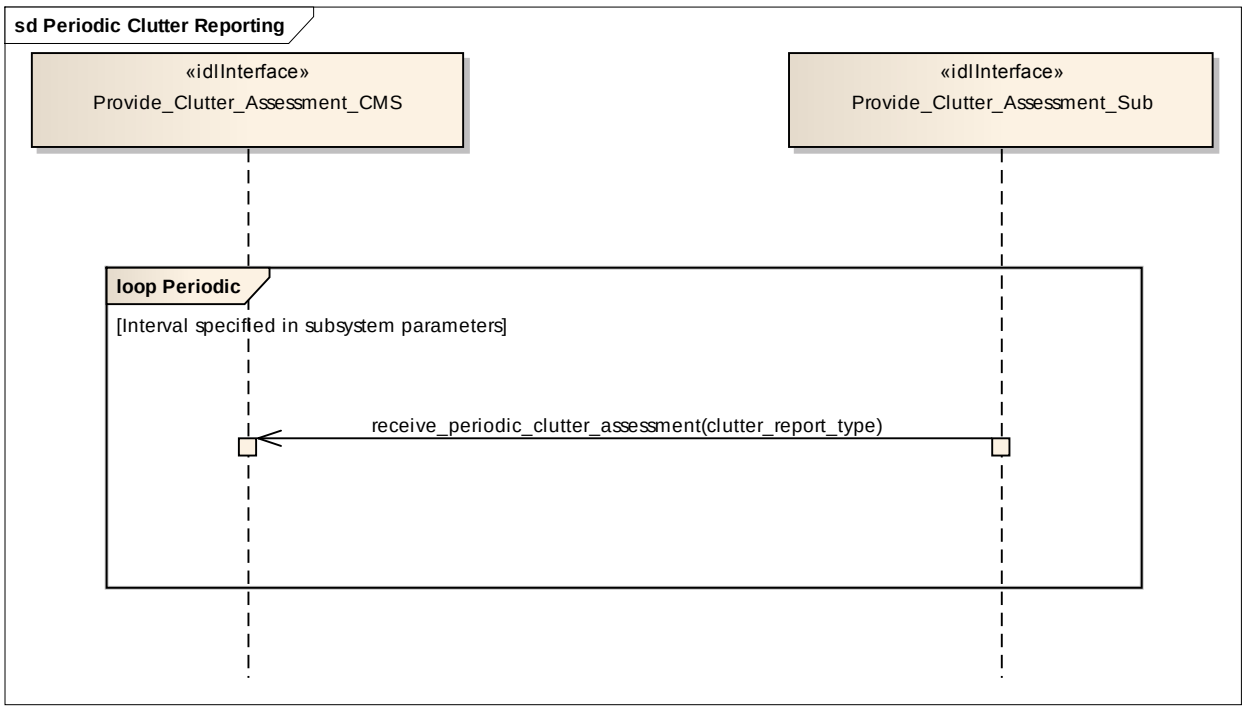

Figure 7.101 Periodic Clutter Reporting (Sequence diagram)

Flow of events which depicts a subsystem that periodically reports a clutter assessment (without the need for a CMS request).

7.8.2 Plot_Reporting

Parent Package: Sensor_Services

7.8.2.1 Provide_Plots

Parent Package: Plot_Reporting

7.8.2.1.1 Provide_Plots_CMS

Type: IDLInterface
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 7.175 - Methods of IDLInterface 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

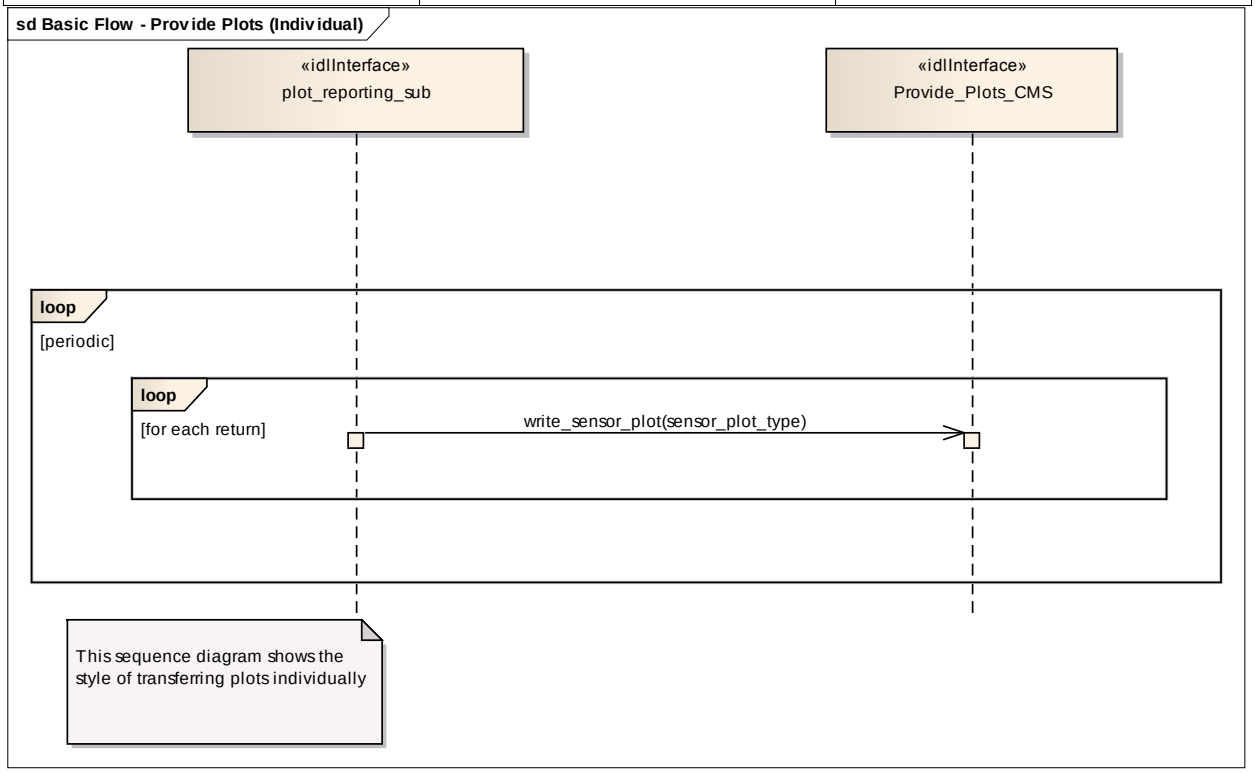

Figure 7.102 Basic Flow - Provide Plots (Individual) (Sequence diagram)

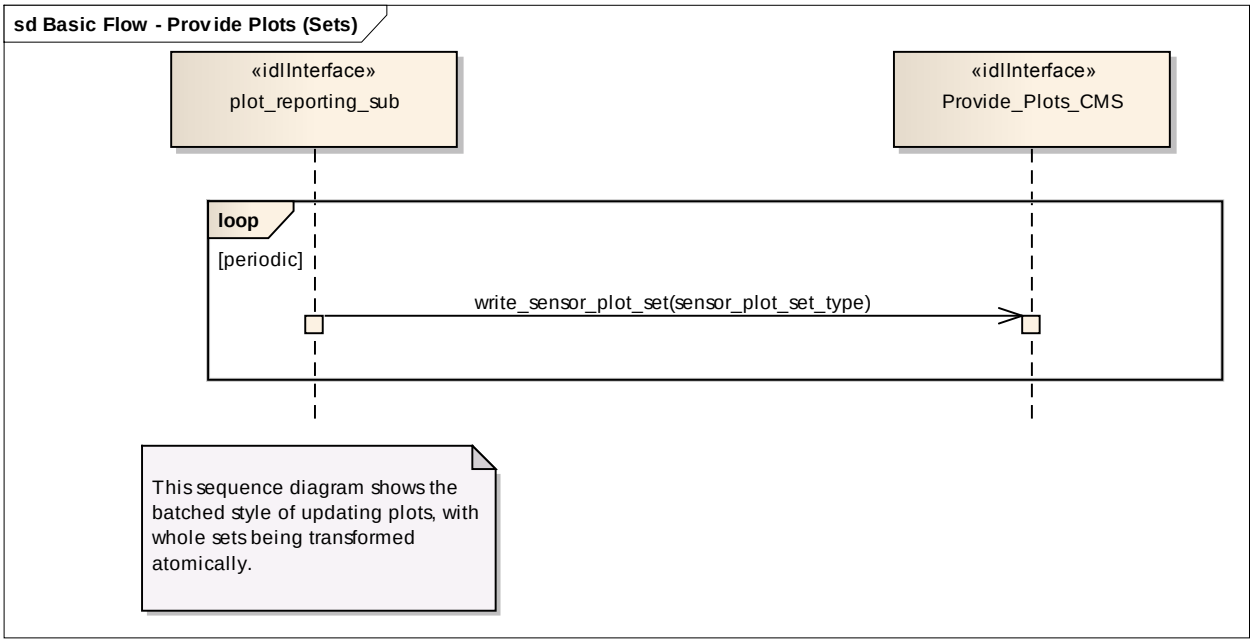

Figure 7.103 Basic Flow - Provide Plots (Sets) (Sequence diagram)

7.8.2.2 Provide_Sensor_Orientation

Parent Package: Plot_Reporting

7.8.2.2.1 Provide_Sensor_Orientation_CMS

Type: IDLInterface
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 7.176 - Methods of IDLInterface 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

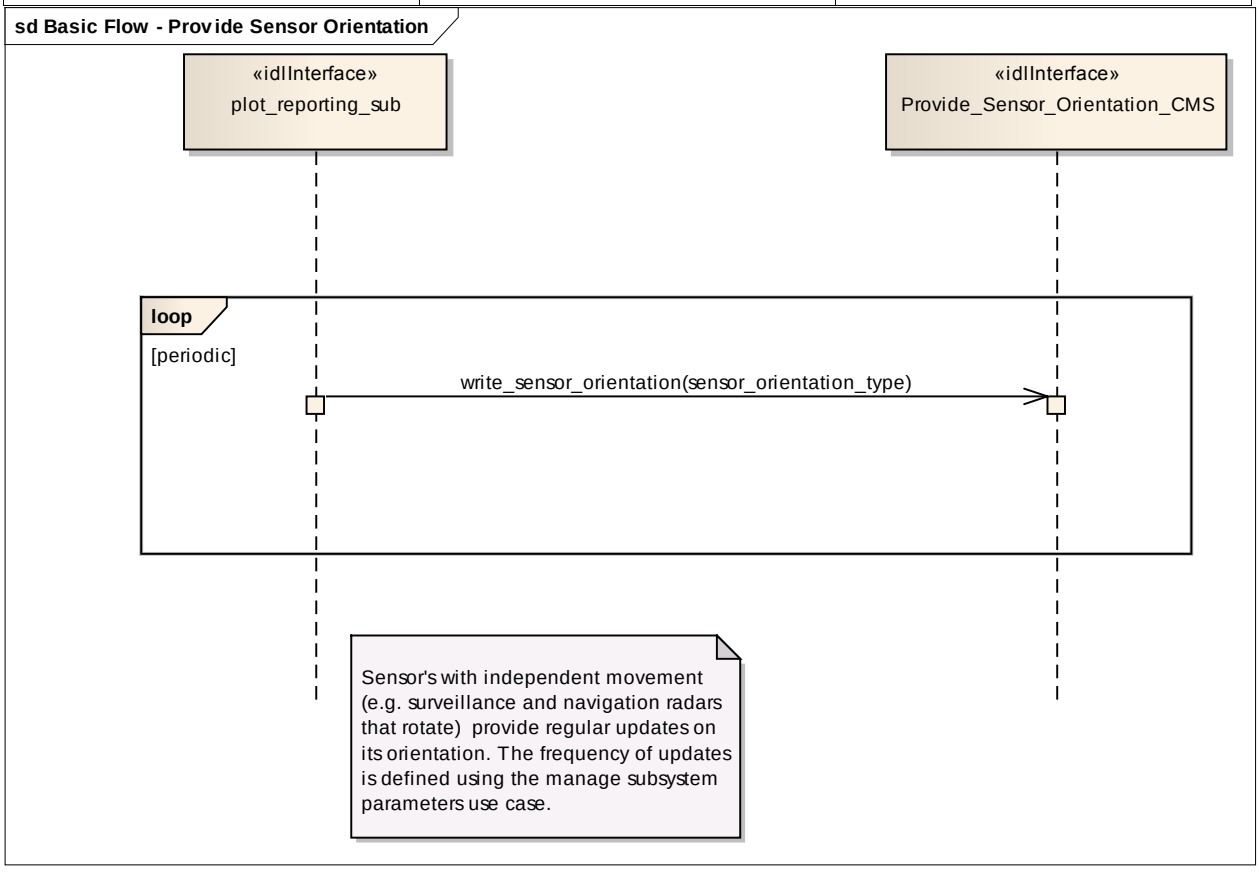

Figure 7.104 Basic Flow - Provide Sensor Orientation (Sequence diagram)

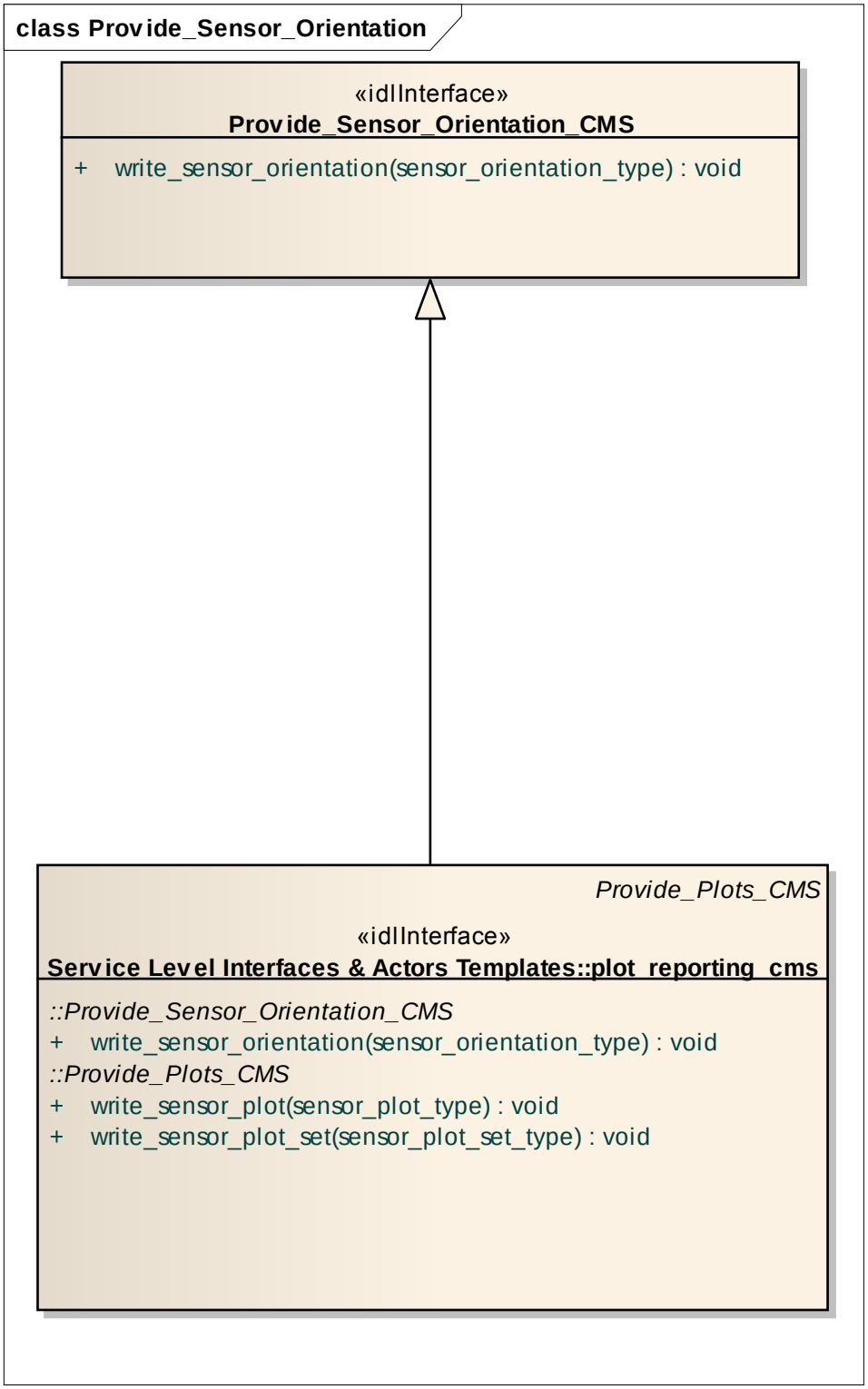

Figure 7.105 Provide_Sensor_Orientation (Logical diagram)

7.8.3 Sensor_Control

Parent Package: Sensor_Services
This package contains interfaces for the Sensor Control service.

7.8.3.1 Manage_Frequency_Usage

Parent Package: Sensor_Control
This package contains interfaces for the Manage Frequency Usage service.

7.8.3.1.1 Manage_Frequency_Usage_CMS

Type: IDLInterface common_use_case_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 7.177 - Methods of IDLInterface 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**

7.8.3.1.2 Manage_Frequency_Usage_Sub

Type: IDLInterface
Package: Manage_Frequency_Usage
This is the Subsystem interface for managing frequency usage.

Table 7.178 - Methods of IDLInterface 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**

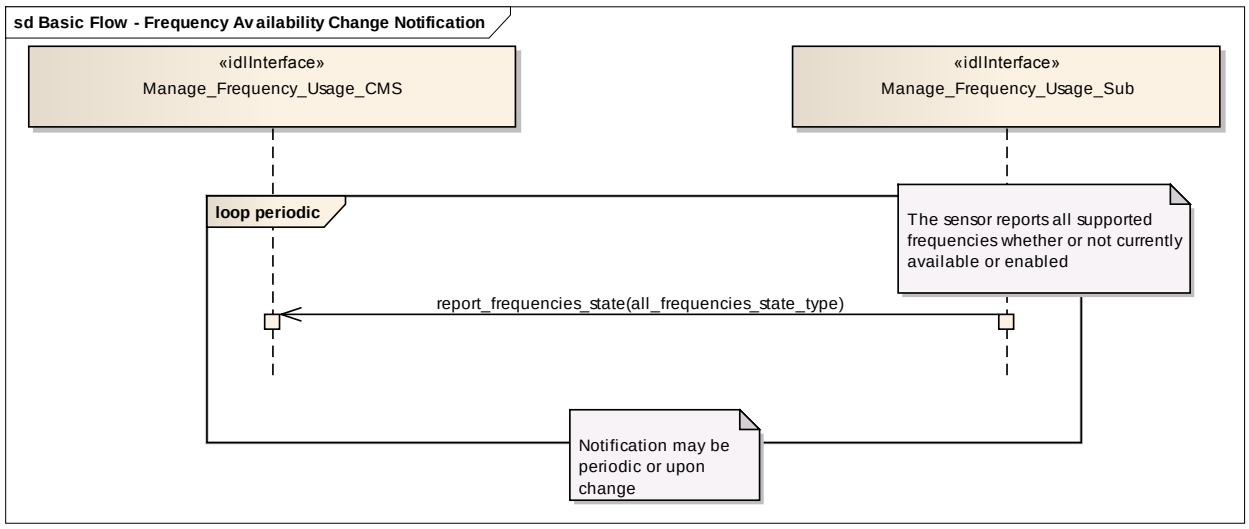

Figure 7.106 Basic Flow - Frequency Availability Change Notification (Sequence diagram)

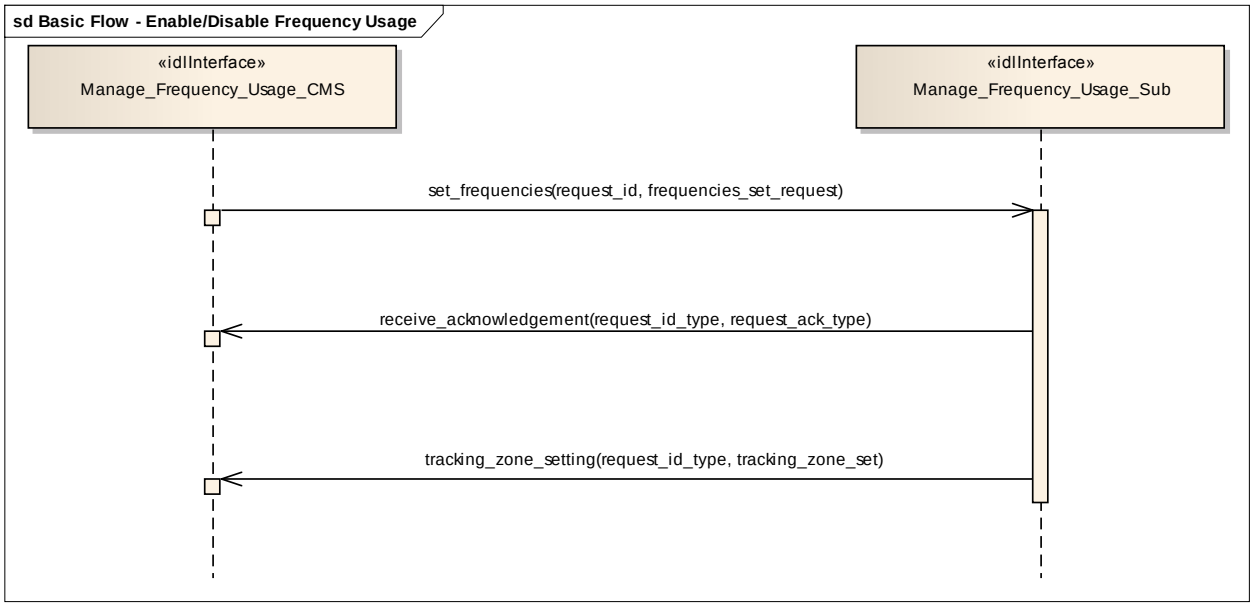

Figure 7.107 Basic Flow - Enable/Disable Frequency Usage (Sequence diagram)

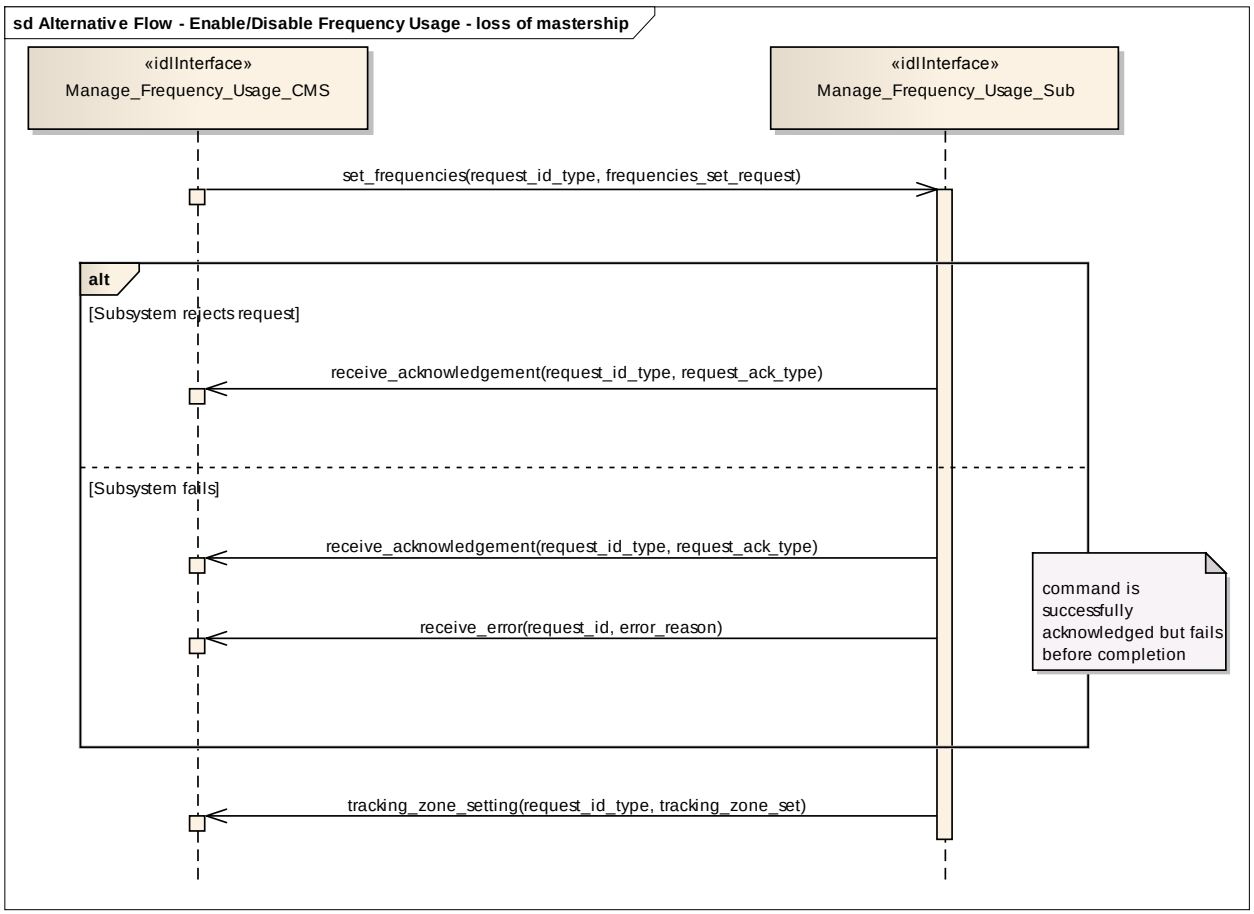

Figure 7.108 Alternative Flow - Enable/Disable Frequency Usage - loss of mastership (Sequence diagram)

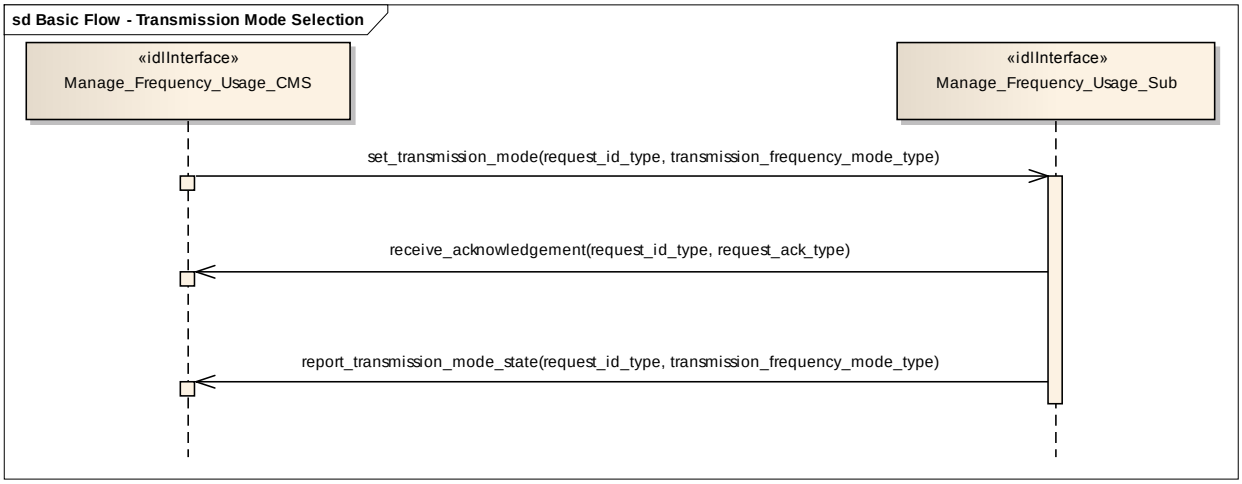

Figure 7.109 Basic Flow - Transmission Mode Selection (Sequence diagram)

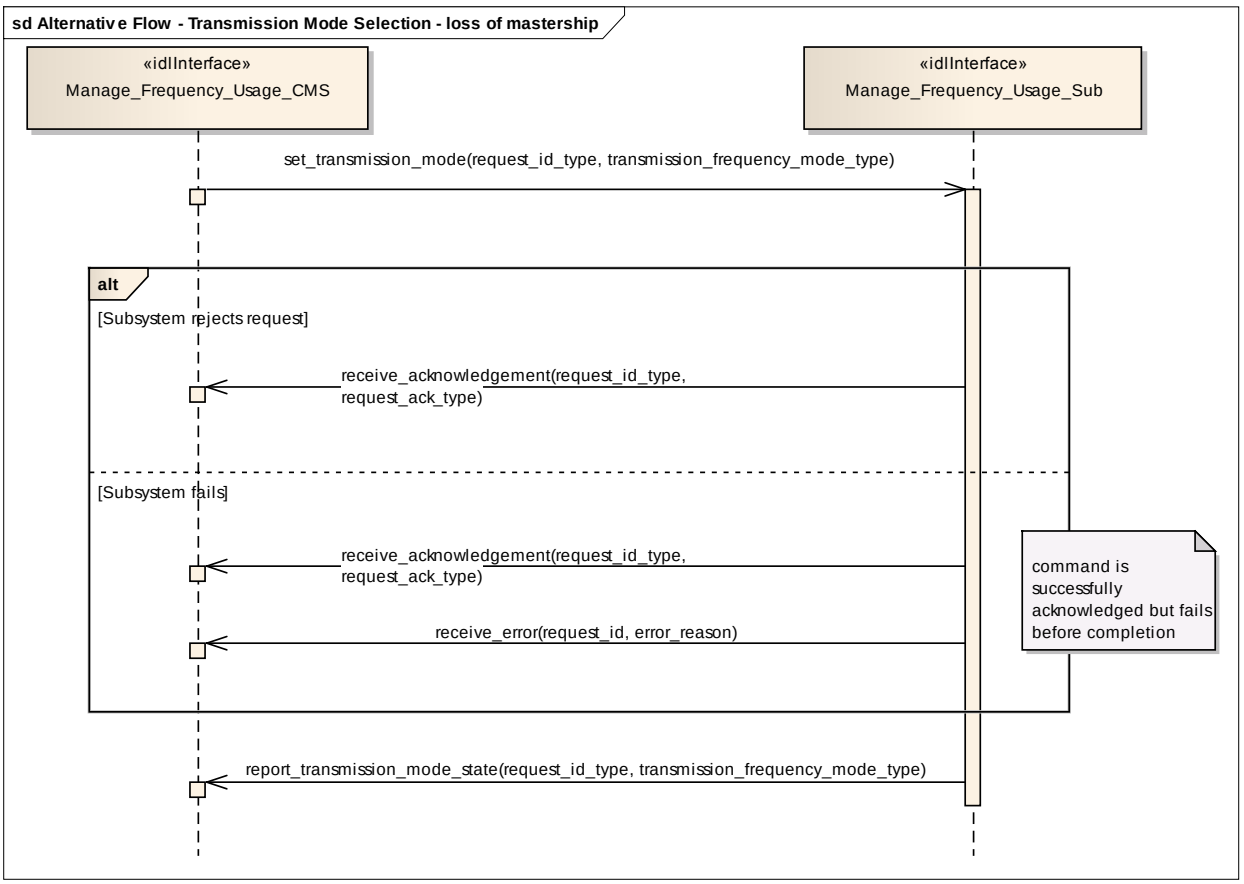

Figure 7.110 Alternative Flow - Transmission Mode Selection - loss of mastership (Sequence diagram)

7.8.3.2 Manage_Transmission_Sectors

Parent Package: Sensor_Control
This package contains interfaces for the Manage Transmission Sectors service.

7.8.3.2.1 Manage_Transmission_Sectors_CMS

Type: IDLInterface common_use_case_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 7.179 - Methods of IDLInterface 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**

7.8.3.2.2 Manage_Transmission_Sectors_Sub

Type: IDLInterface
Package: Manage_Transmission_Sectors
This is the Subsystem interface for managing transmission sectors.

Table 7.180 - Methods of IDLInterface 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**

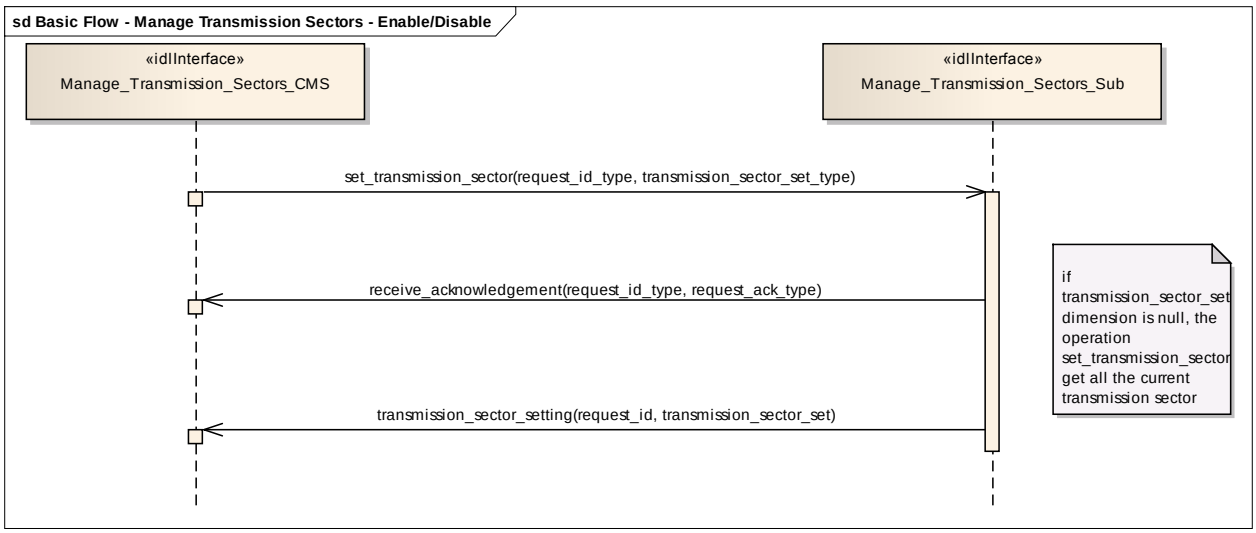

Figure 7.111 Basic Flow - Manage Transmission Sectors - Enable/Disable (Sequence diagram)

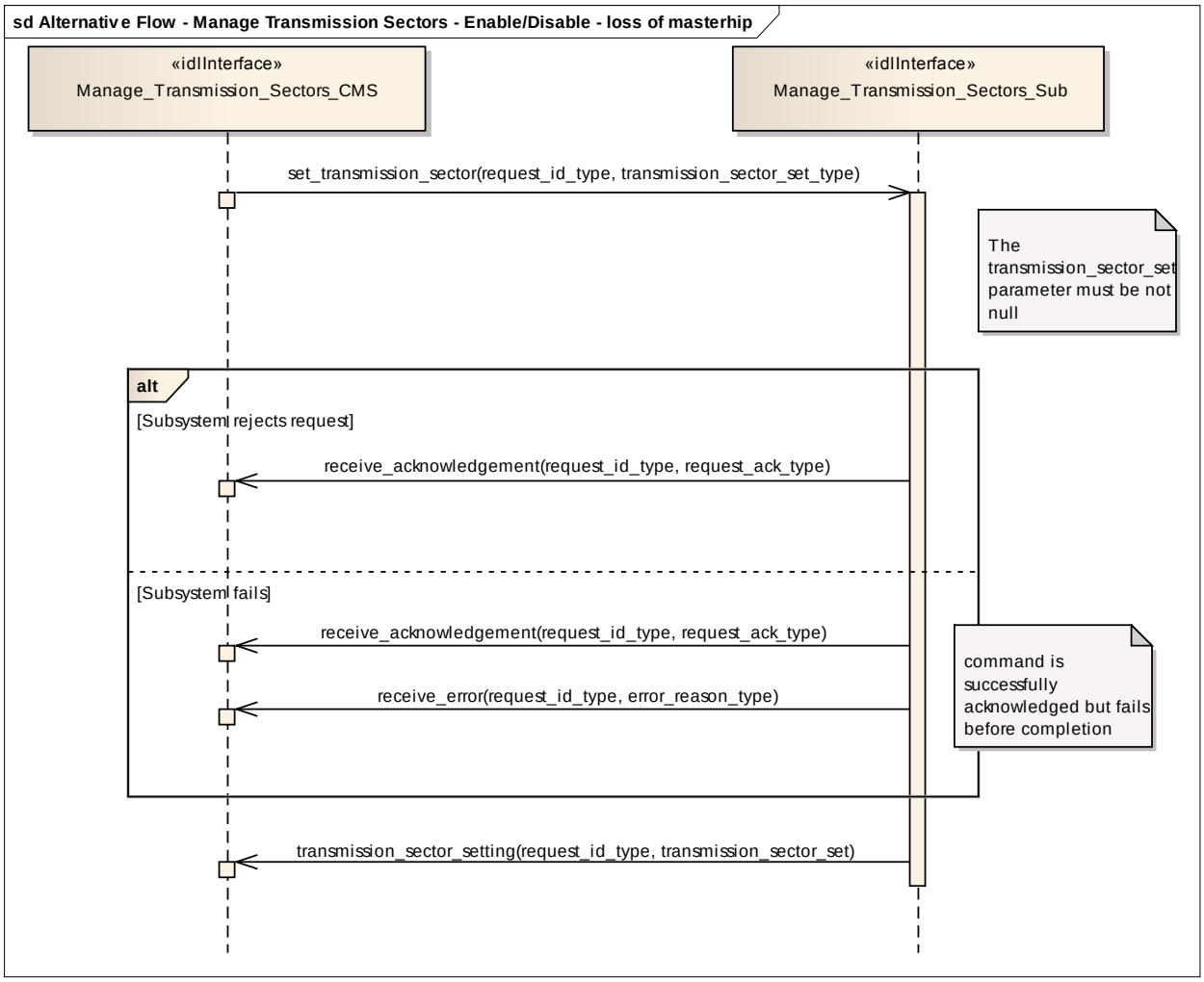

Figure 7.112 Alternative Flow - Manage Transmission Sectors - Enable/Disable - loss of masterhip (Sequence diagram)

7.8.3.3 Control_Emissions

Parent Package: Sensor_Control
This package contains interfaces for the Control Emissions service.

7.8.3.3.1 Control_Emissions_CMS

Type: IDLInterface common_use_case_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 7.181 - Methods of IDLInterface 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**

7.8.3.3.2 Control_Emissions_Sub

Type: IDLInterface
Package: Control_Emissions
This is the Subsystem interface for controlling emissions.

Table 7.182 - Methods of IDLInterface 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**

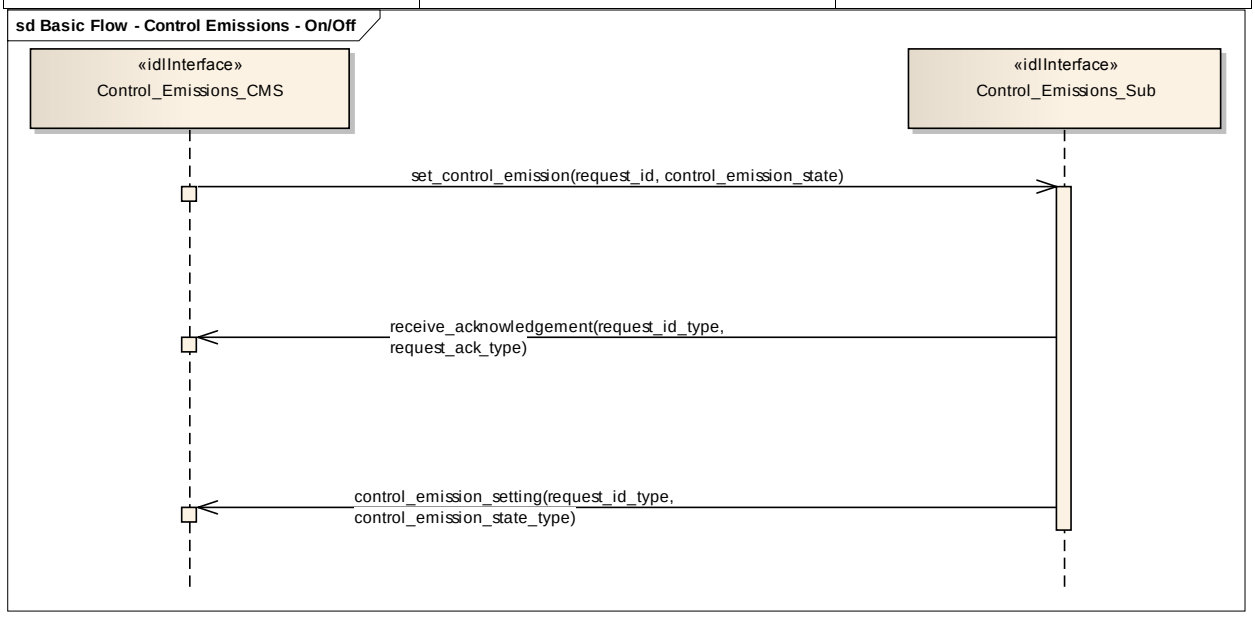

Figure 7.113 Basic Flow - Control Emissions - On/Off (Sequence diagram)

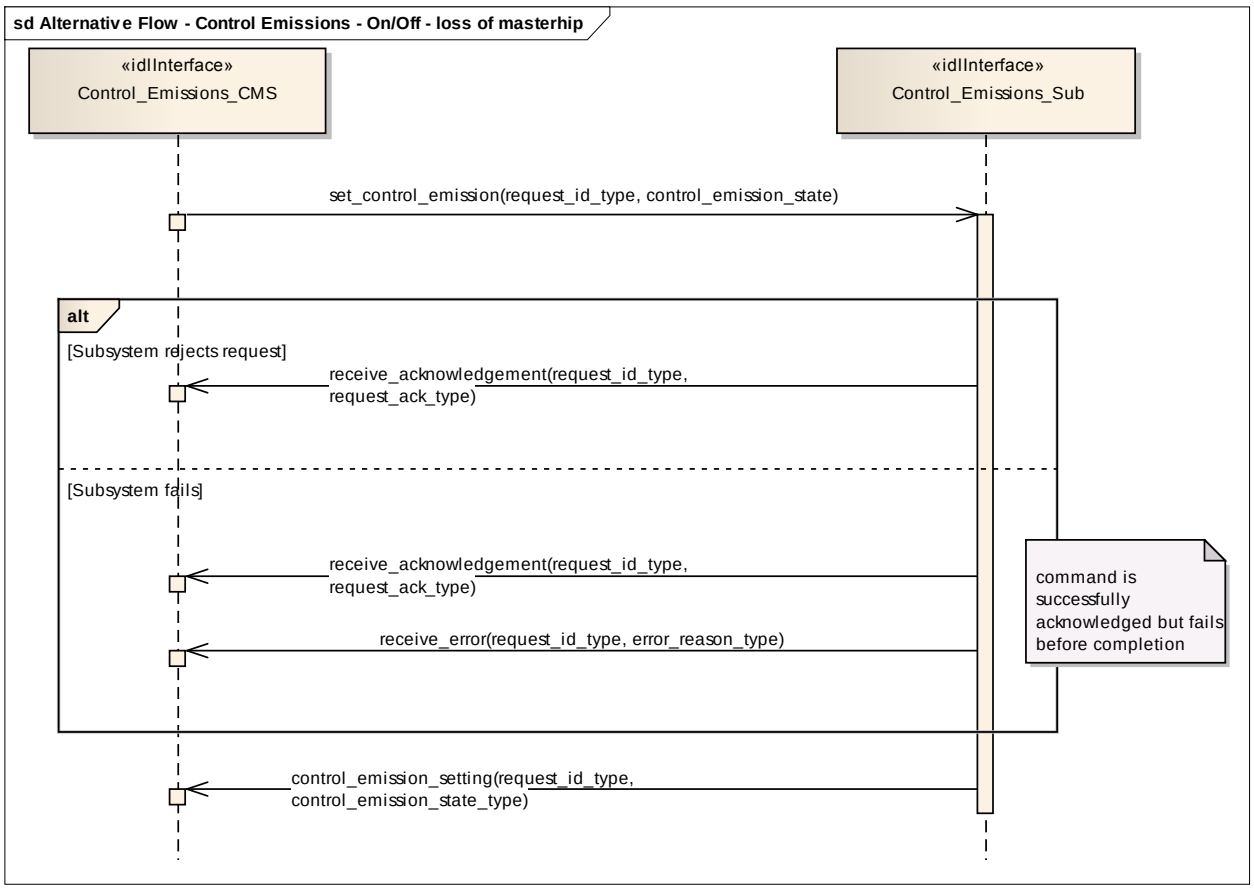

Figure 7.114 Alternative Flow - Control Emissions - On/Off - loss of masterhip (Sequence diagram)

7.8.3.4 Define_Test_Target_Scenario

Parent Package: Sensor_Control
This package contains interfaces for the Define Test Target Scenario service.

7.8.3.4.1 Define_Test_Target_Scenario_CMS

Type: IDLInterface common_use_case_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 7.183 - Methods of IDLInterface 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**

7.8.3.4.2 Define_Test_Target_Scenario_Sub

Type: IDLInterface
Package: Define_Test_Target_Scenario

This is the Subsystem interface for defining test target scenarios.

Table 7.184 - Methods of IDLInterface 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**

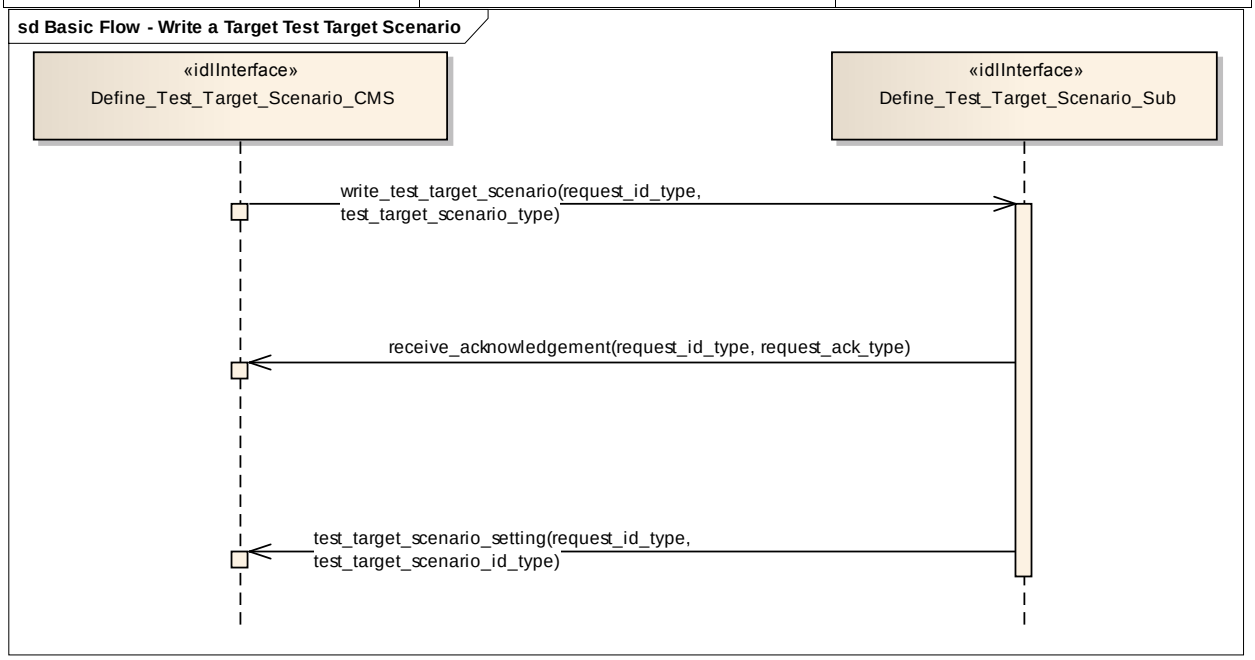

Figure 7.115 Basic Flow - Write a Target Test Target Scenario (Sequence diagram)

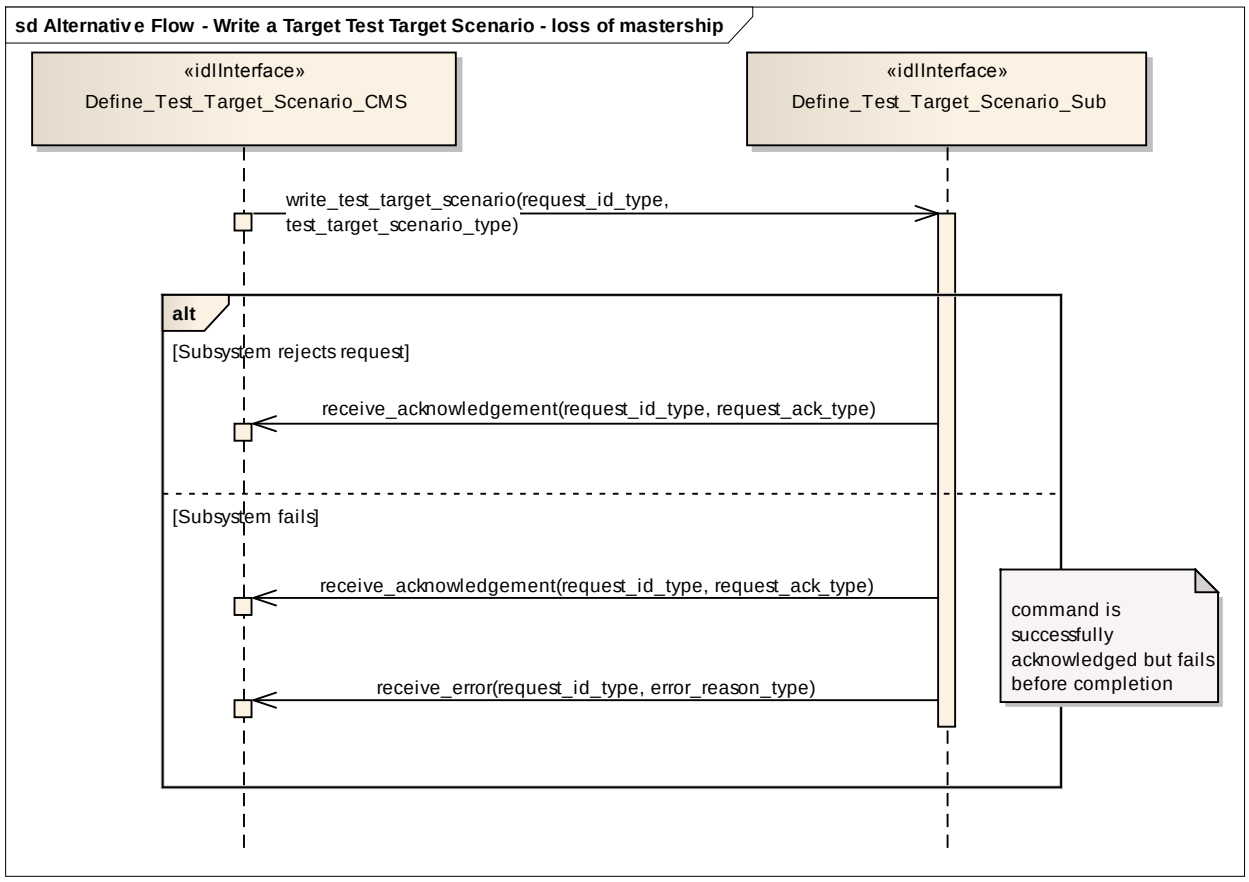

Figure 7.116 Alternative Flow - Write a Target Test Target Scenario - loss of mastership (Sequence diagram)

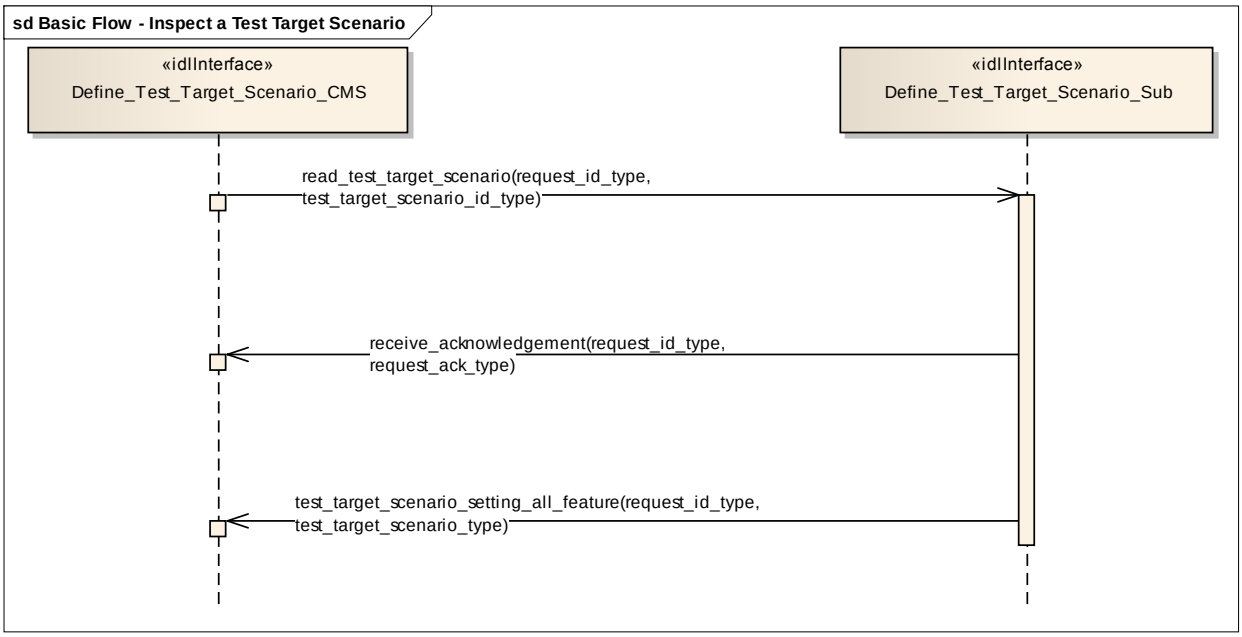

Figure 7.117 Basic Flow - Inspect a Test Target Scenario (Sequence diagram)

7.8.3.5 Test_Target_Facility

Parent Package: Sensor_Control
This package contains interfaces for the Test Target Facility service.

7.8.3.5.1 Test_Target_Facility_CMS

Type: IDLInterface common_use_case_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 7.185 - Methods of IDLInterface 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**

7.8.3.5.2 Test_Target_Facility_Sub

Type: IDLInterface

Package: Test_Target_Facility
This is the Subsystem interface for testing target facilities.

Table 7.186 - Methods of IDLInterface 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 7.118 Basic Flow - Activate/Deactivate Test Target Facility (Sequence diagram)

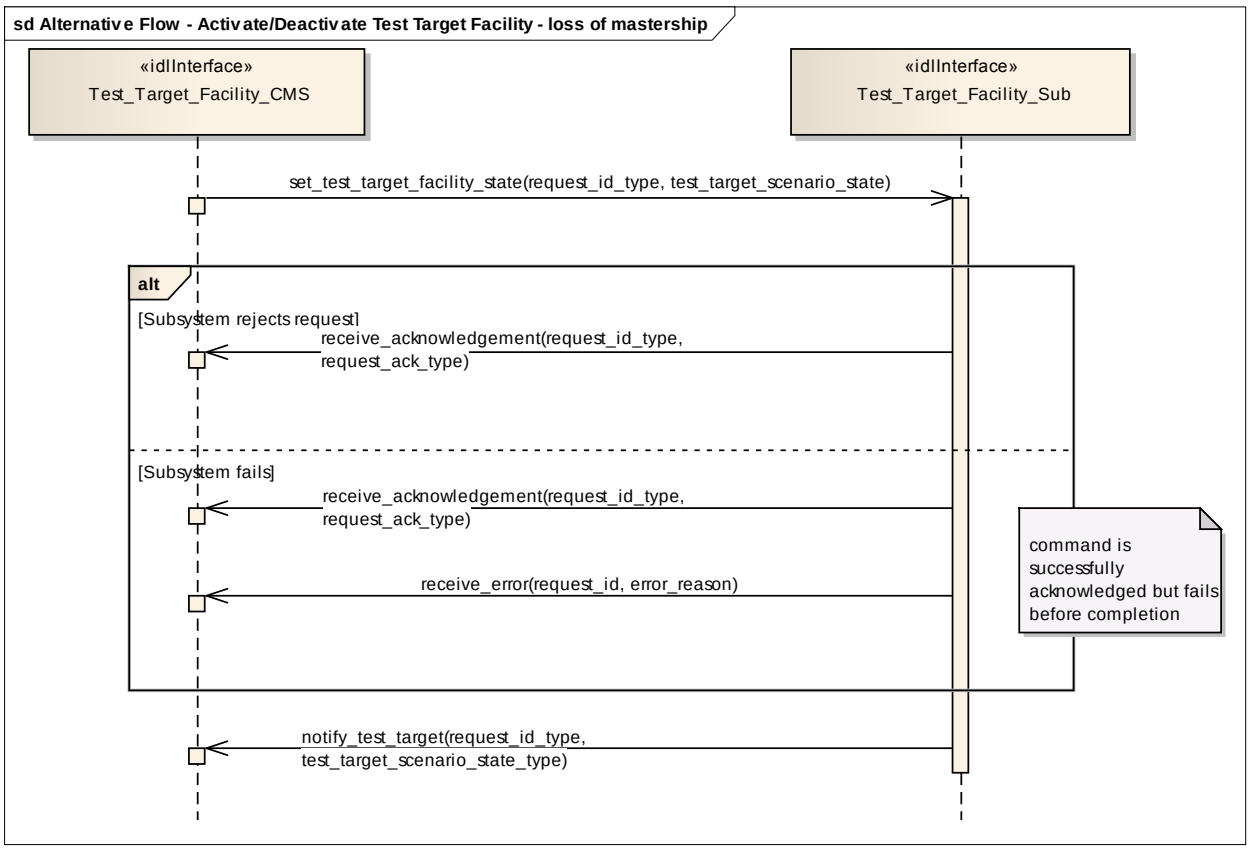

Figure 7.119 Alternative Flow - Activate/Deactivate Test Target Facility - loss of mastership (Sequence diagram)

7.8.4 Sensor_Performance

Parent Package: Sensor_Services

7.8.4.1 Provide_Interference_Reports

Parent Package: Sensor_Performance

7.8.4.1.1 Provide_Interference_Reports_CMS

Type: IDLInterface common_use_case_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 7.187 - Methods of IDLInterface 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

7.8.4.1.2 Provide_Interference_Reports_Sub

Type: IDLInterface
Package: Provide_Interference_Reports

Table 7.188 - Methods of IDLInterface 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 interference reports.	request_id_type **request_id** The unique identifier for this request. This is referenced in acknowledgement 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

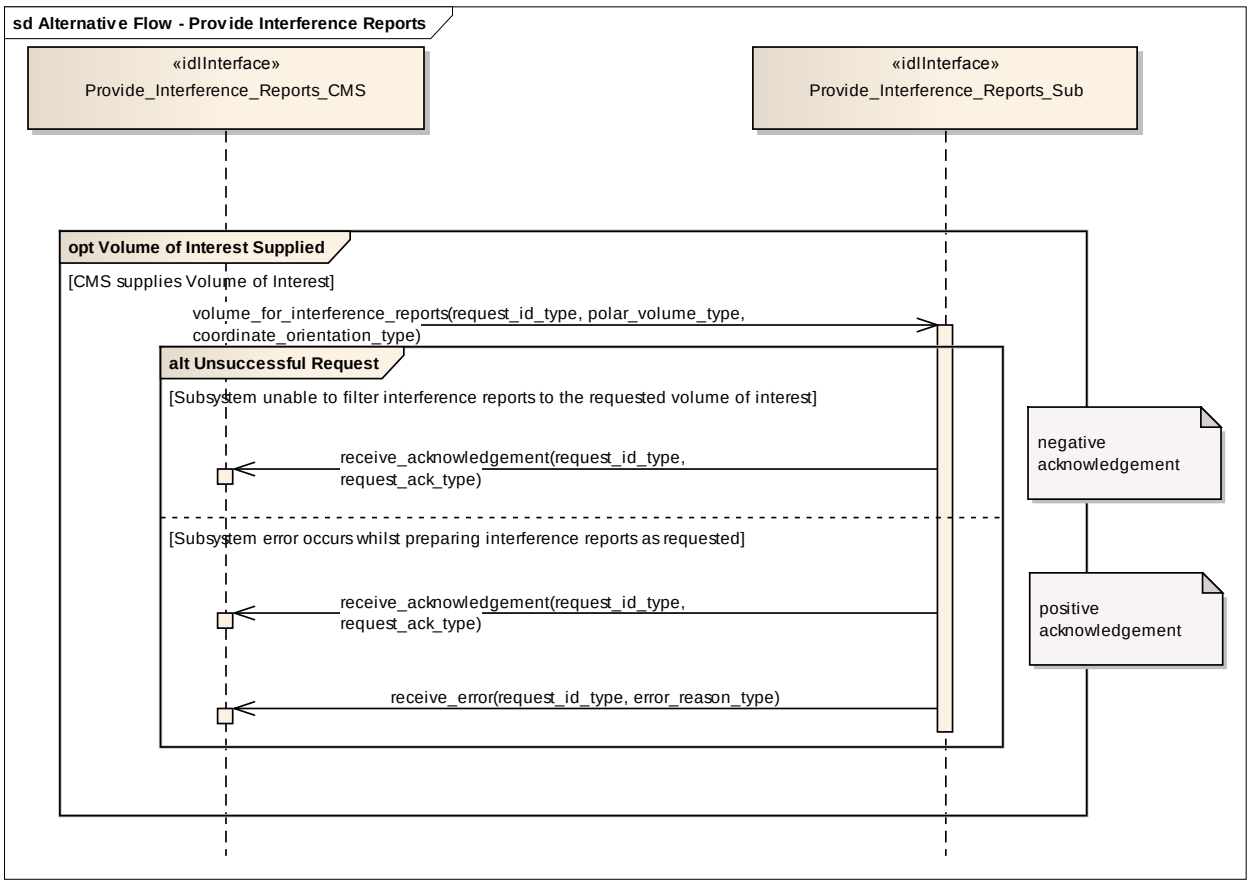

Figure 7.120 Alternative Flow - Provide Interference Reports (Sequence diagram)

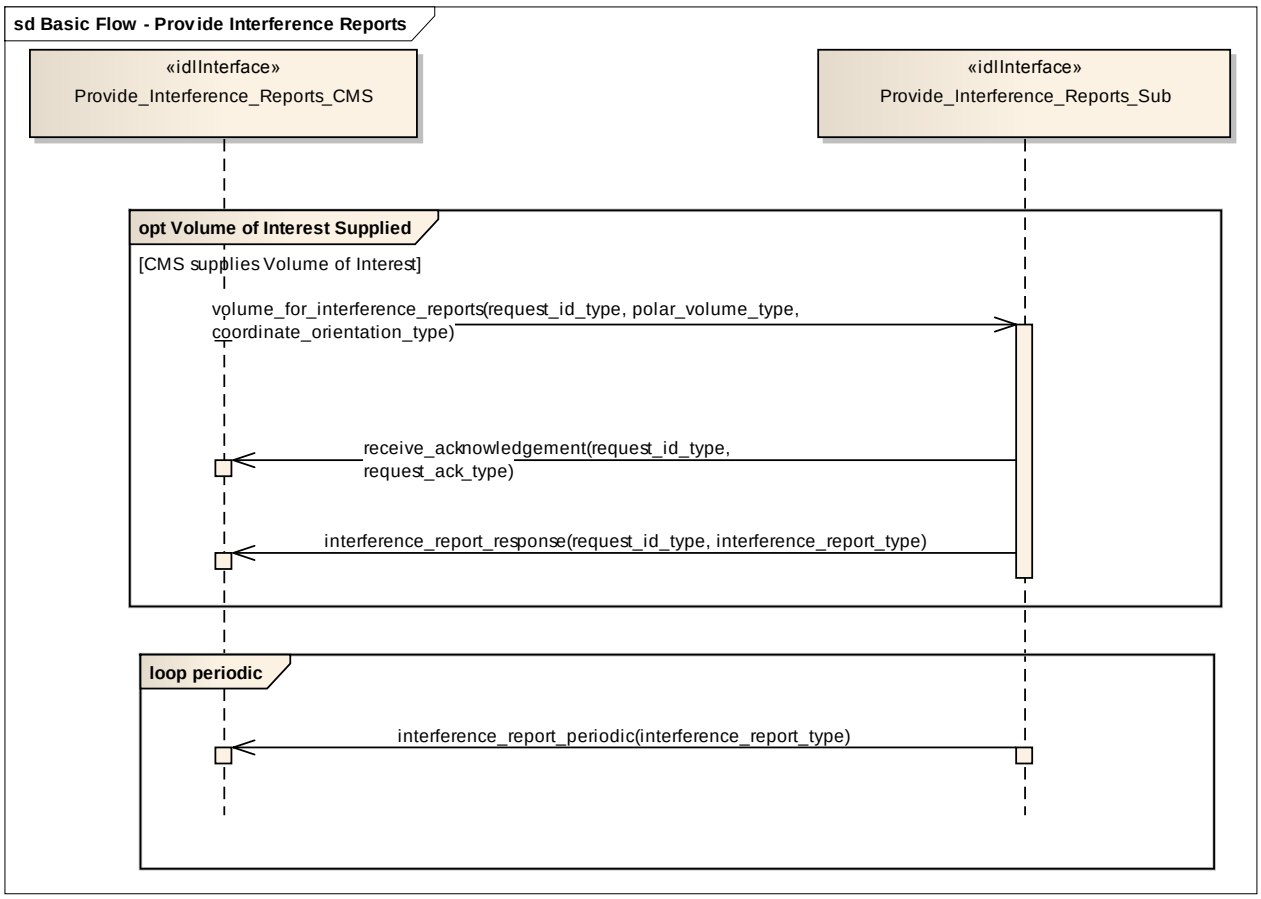

Figure 7.121 Basic Flow - Provide Interference Reports (Sequence diagram)

7.8.4.2 Provide_Nominal_Performance

Parent Package: Sensor_Performance

7.8.4.2.1 Provide_Nominal_Performance_CMS

Type: IDLInterface common_use_case_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 7.189 - Methods of IDLInterface 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

7.8.4.2.2 Provide_Nominal_Performance_Sub

Type: IDLInterface
Package: Provide_Nominal_Performance
Subsystem interface for provision of nominal performance assessment.

Table 7.190 - Methods of IDLInterface 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_request_type **request** The details of the performance request

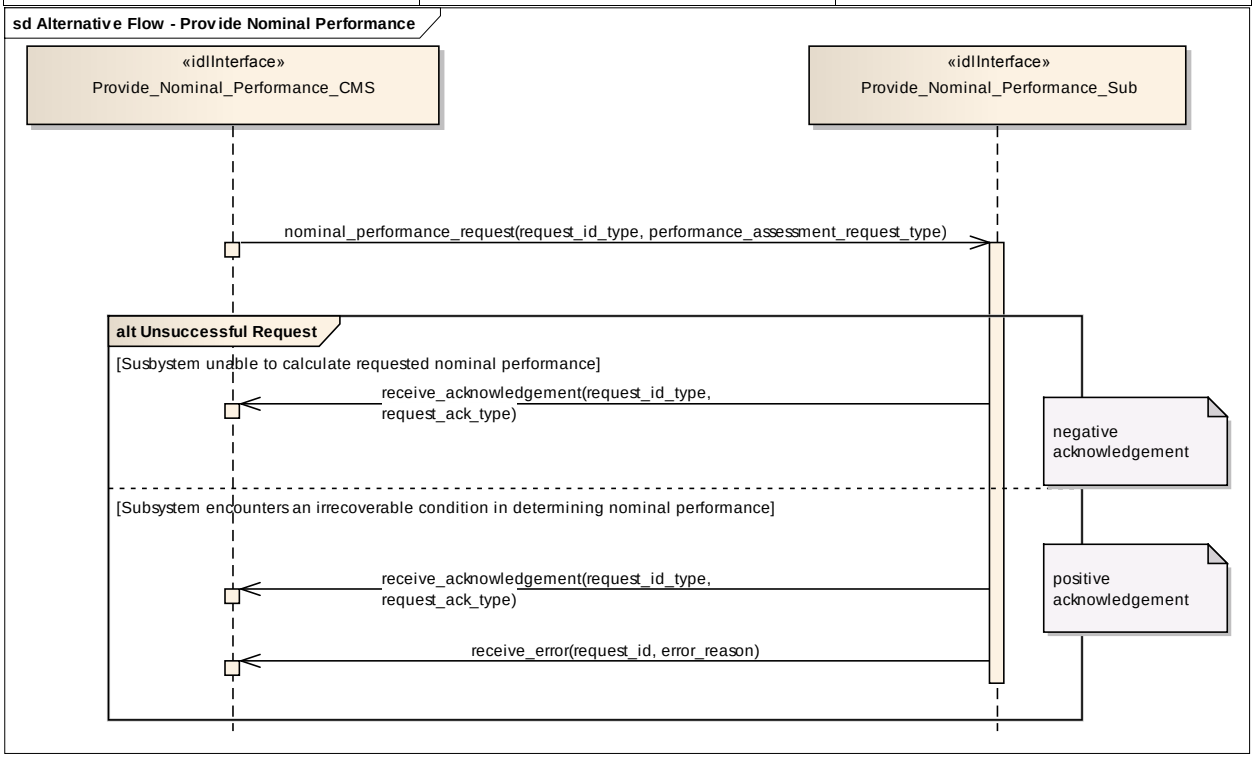

Figure 7.122 Alternative Flow - Provide Nominal Performance (Sequence diagram)

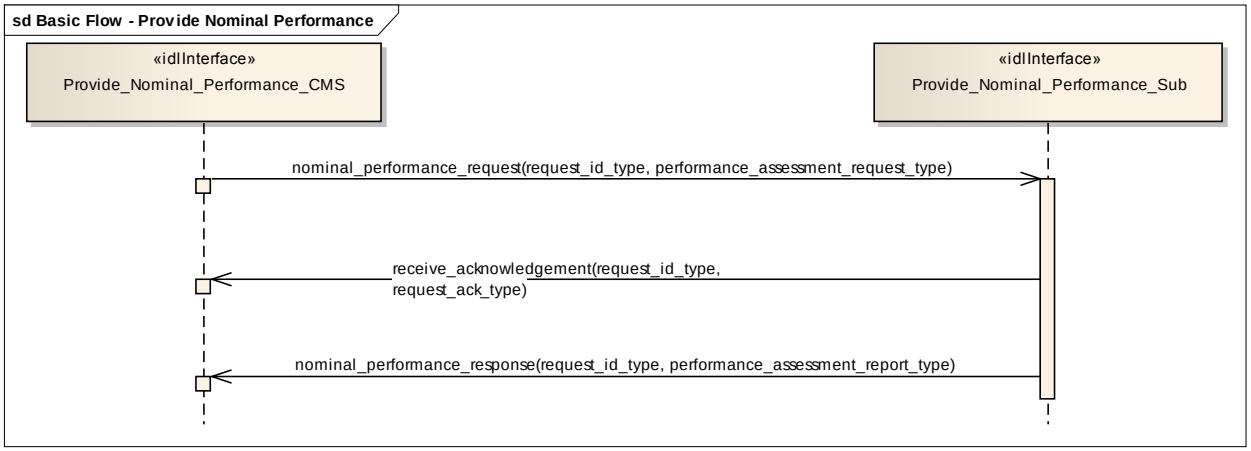

Figure 7.123 Basic Flow - Provide Nominal Performance (Sequence diagram)

7.8.4.3 Provide_Performance_Assessment

Parent Package: Sensor_Performance

7.8.4.3.1 Provide_Performance_Assessment_CMS

Type: IDLInterface common_use_case_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

able 7.191 - Methods of IDLInterface 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

7.8.4.3.2 Provide_Performance_Assessment_Sub

Type: IDLInterface
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 7.192 - Methods of IDLInterface 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_request_type **request** Details of the assessment

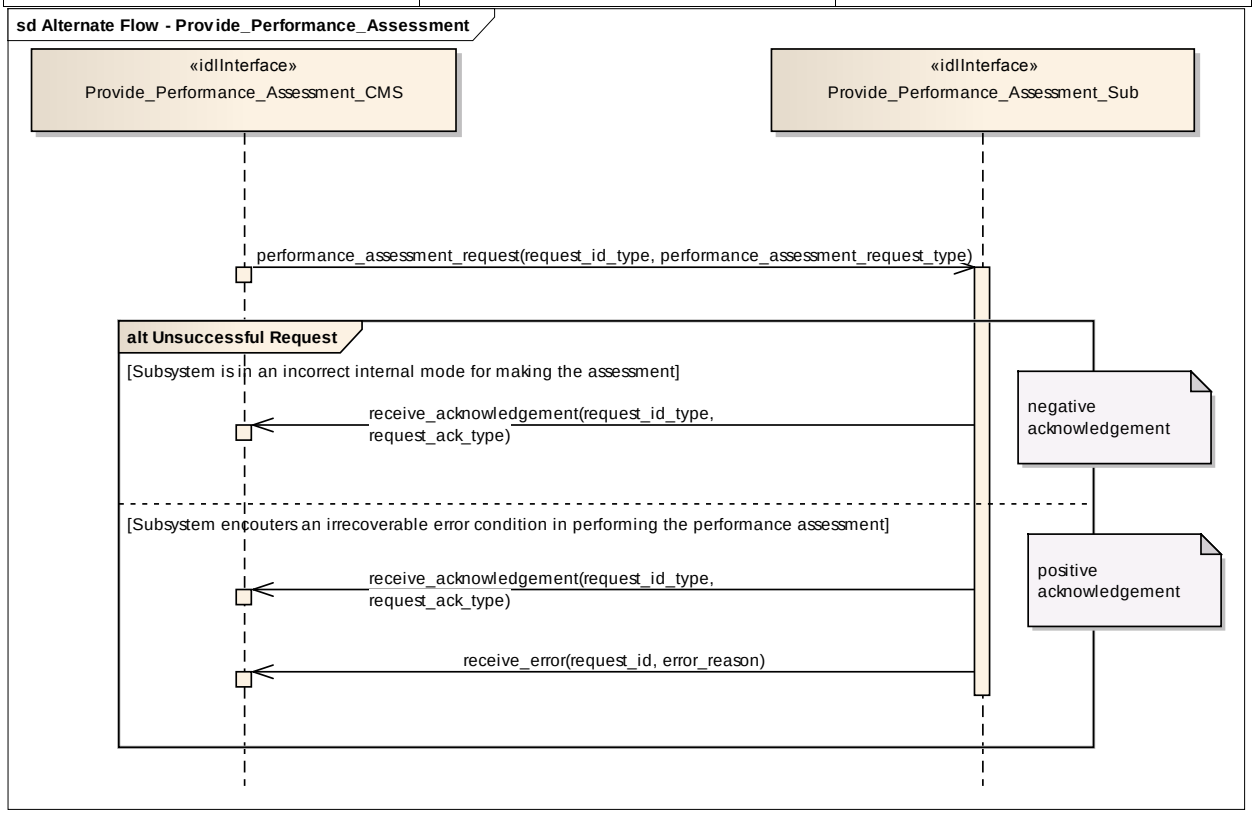

Figure 7.124 Alternate Flow - Provide_Performance_Assessment (Sequence diagram)

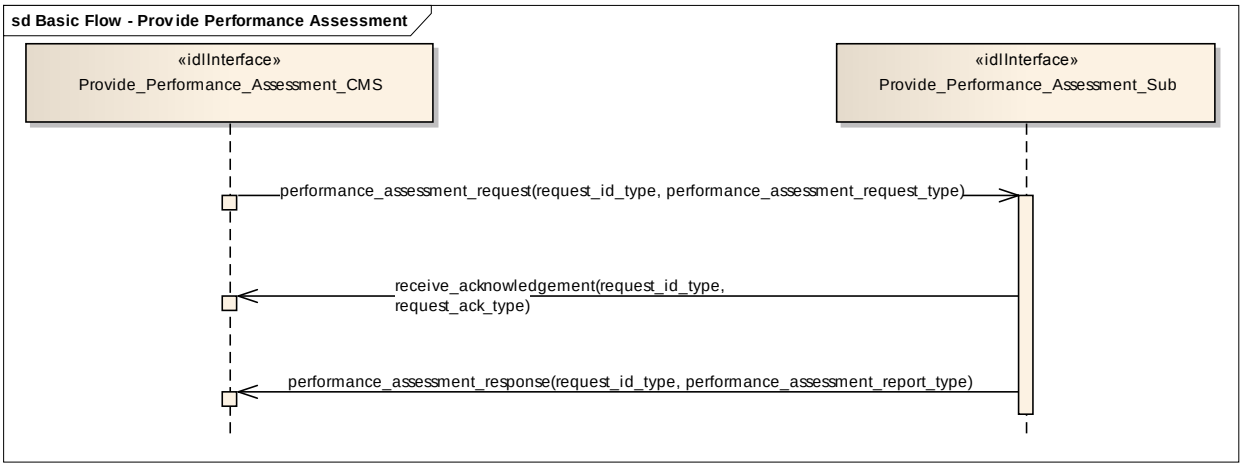

Figure 7.125 Basic Flow - Provide Performance Assessment (Sequence diagram)

7.8.4.4 Provide_Jammer_Assessment

Parent Package: Sensor_Performance

7.8.4.4.1 Provide_Jammer_Assessment_CMS

Type: IDLInterface common_use_case_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 7.193 - Methods of IDLInterface Provide_Jammer_Assessment_CMS

Method	Notes	Parameters
jammer_assessment_response()		request_id_type **request_id** performance_assessment_report_typ

		e **report**

7.8.4.4.2 Provide_Jammer_Assessment_Sub

Type: IDLInterface
Package: Provide_Jammer_Assessment

Table 7.194 - Methods of IDLInterface Provide_Jammer_Assessment_Sub

Method	Notes	Parameters
jammer_assessment_request()		request_id_type **request_id** performance_assessment_request_type **jammer_assessment_request**

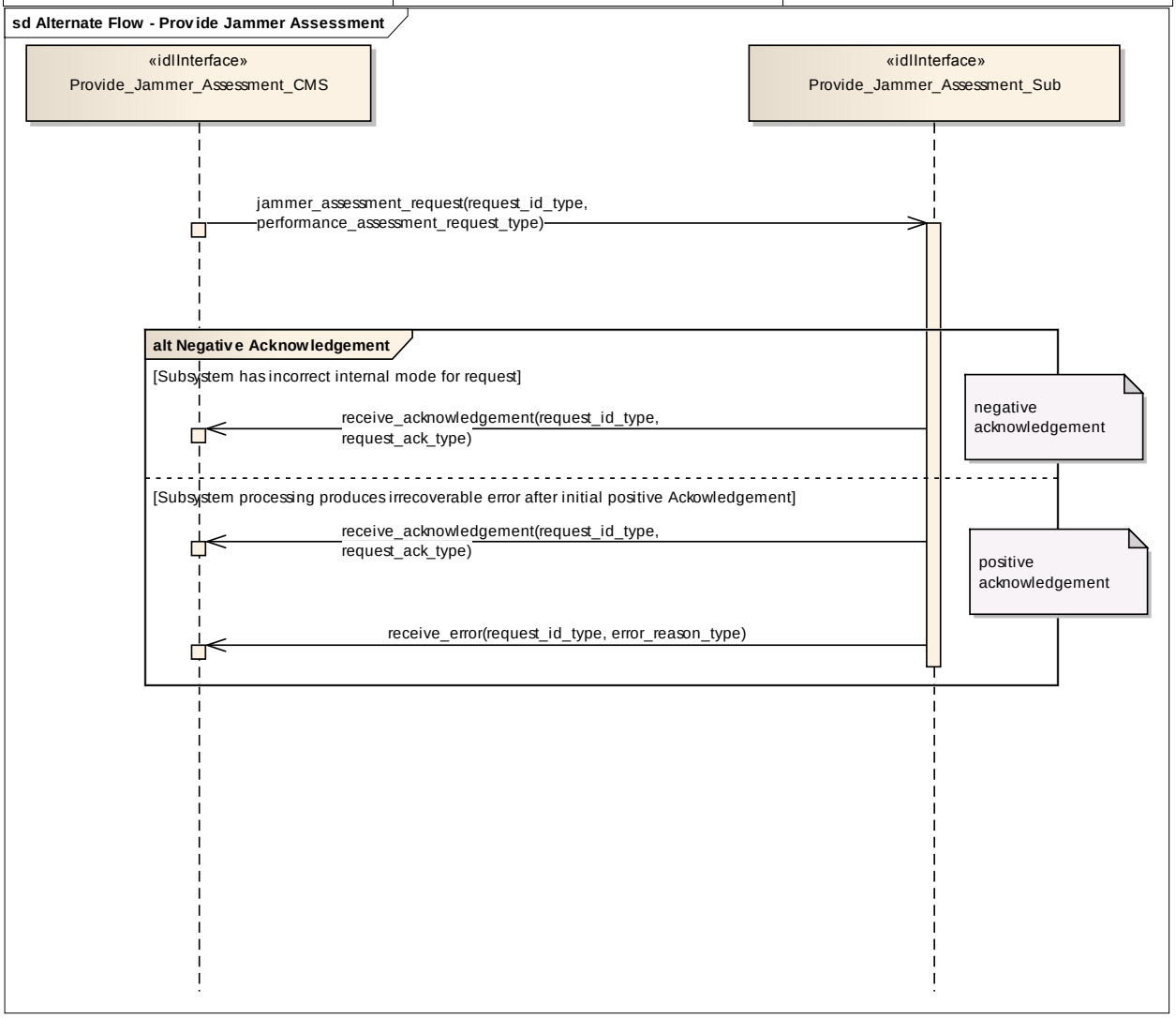

Figure 7.126 Alternate Flow - Provide Jammer Assessment (Sequence diagram)

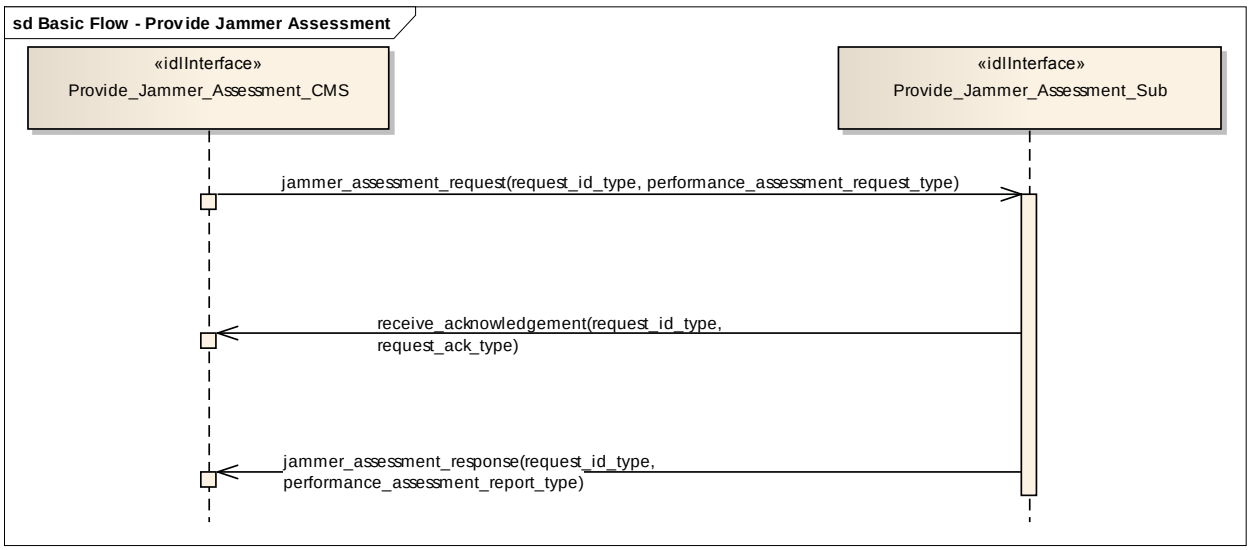

Figure 7.127 Basic Flow - Provide Jammer Assessment (Sequence diagram)

7.8.5 Track_Reporting

Parent Package: Sensor_Services

7.8.5.1 Provide_Sensor_Tracks

Parent Package: Track_Reporting

7.8.5.1.1 Provide_Sensor_Tracks_CMS

Type: IDLInterface common_use_case_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 7.195 - Methods of IDLInterface 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**

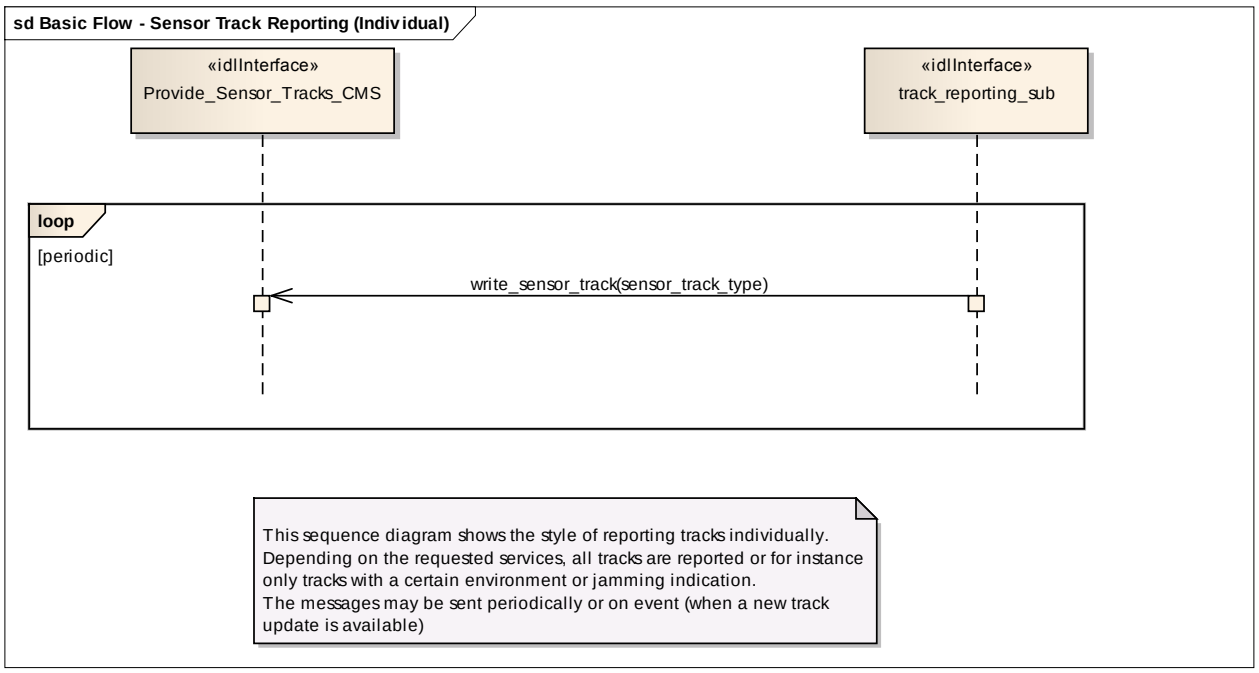

Figure 7.128 Basic Flow - Sensor Track Reporting (Individual) (Sequence diagram)

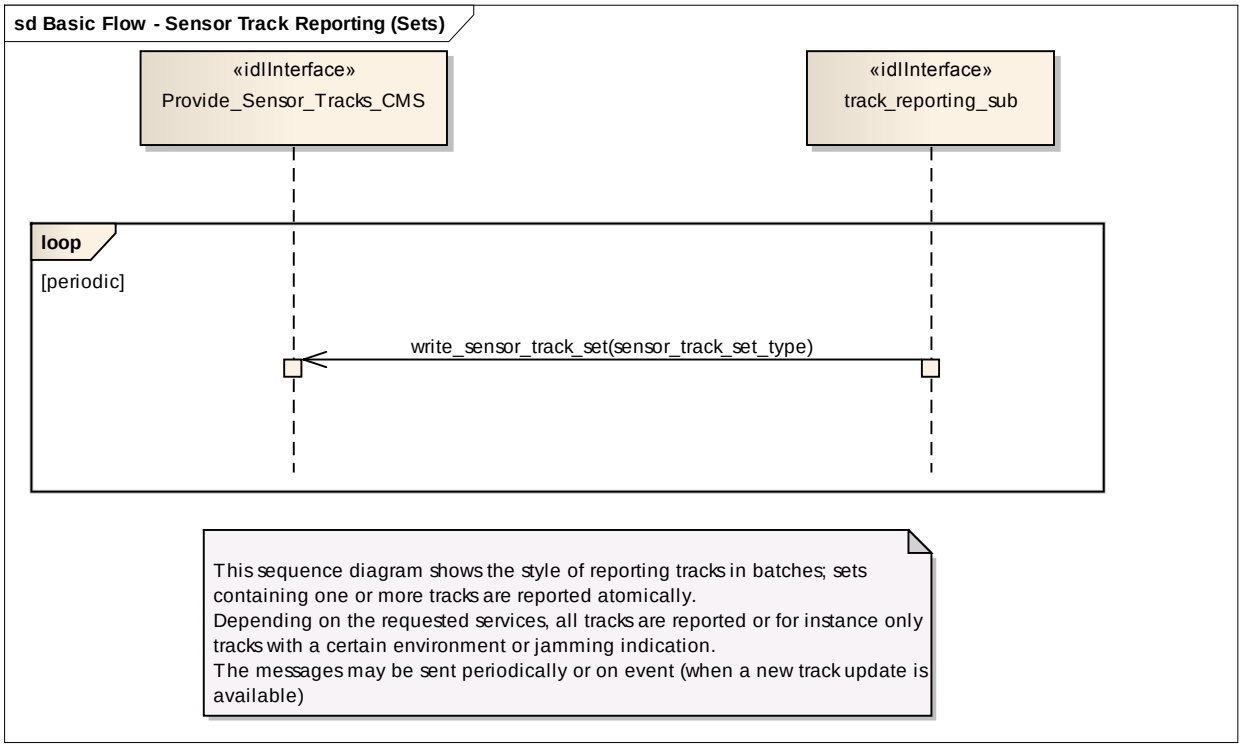

Figure 7.129 Basic Flow - Sensor Track Reporting (Sets) (Sequence diagram)

7.8.6 Tracking_Control

Parent Package: Sensor_Services

This package contains interfaces for the Tracking Control service.

7.8.6.1 Delete_Sensor_Track

Parent Package: Tracking_Control
This package contains interfaces for the Delete Sensor Track service.

7.8.6.1.1 Delete_Sensor_Track_CMS

Type: IDLInterface common_use_case_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.

7.8.6.1.2 Delete_Sensor_Track_Sub

Type: IDLInterface
Package: Delete_Sensor_Track
This is the Subsystem interface for deleting sensor tracks.

Table 7.196 - Methods of IDLInterface 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**

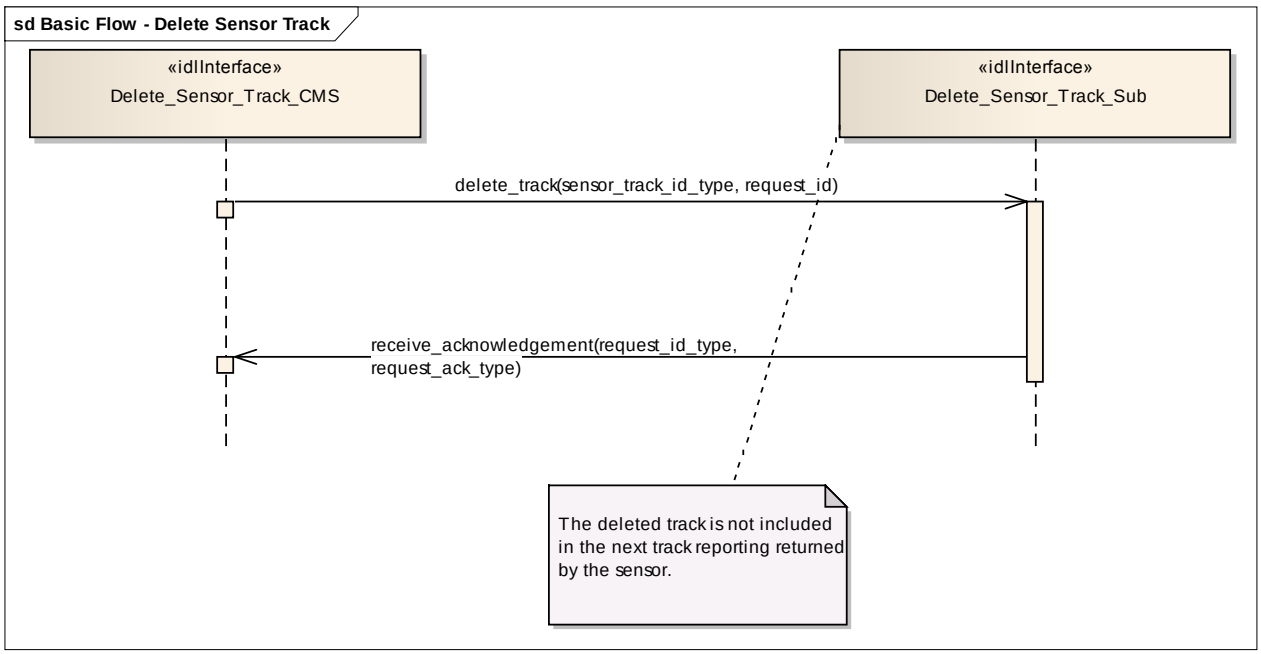

Figure 7.130 Basic Flow - Delete Sensor Track (Sequence diagram)

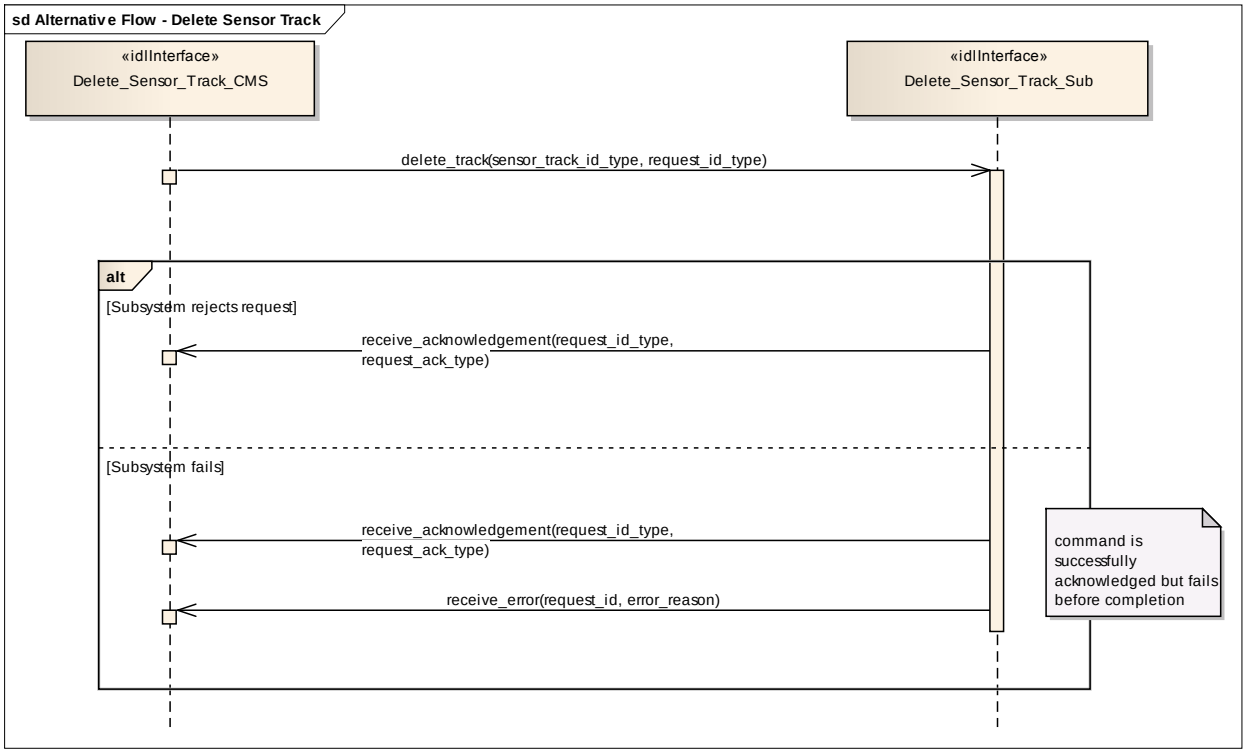

Figure 7.131 Alternative Flow - Delete Sensor Track (Sequence diagram)

7.8.6.2 Receive_Track_Information

Parent Package: Tracking_Control

This package contains interfaces for the Receive Track Information service.

7.8.6.2.1 Receive_Track_Information_CMS

Type: IDLInterface common_use_case_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 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.

7.8.6.2.2 Receive_Track_Information_Sub

Type: IDLInterface
Package: Receive_Track_Information
This is the Subsystem interface for receiving track information.

Table 7.197 - Methods of IDLInterface 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 **trackInfo**

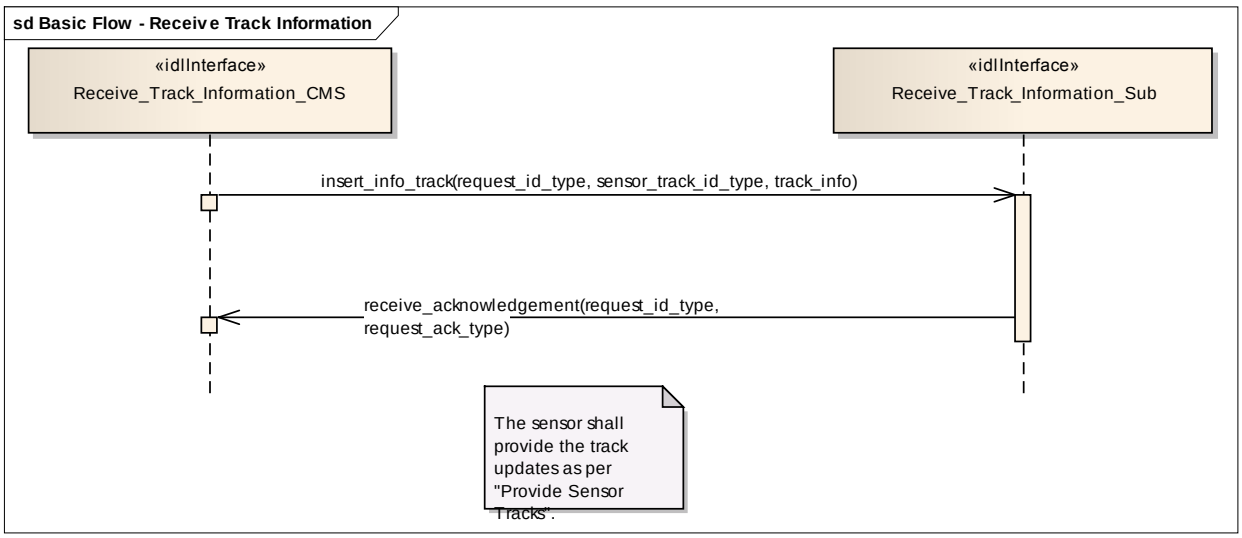

Figure 7.132 Basic Flow - Receive Track Information (Sequence diagram)

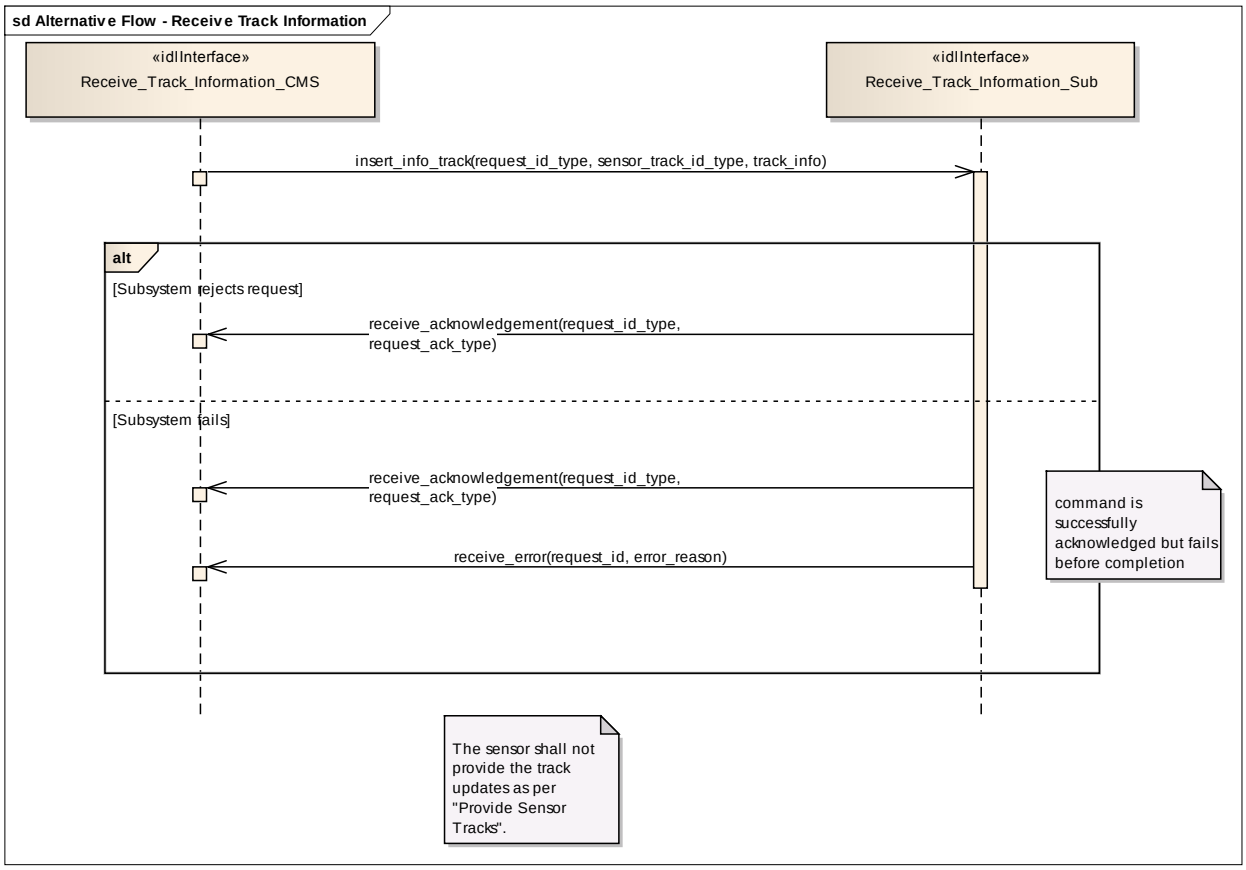

Figure 7.133 Alternative Flow - Receive Track Information (Sequence diagram)

7.8.6.3 Initiate_Track

Parent Package: Tracking_Control

This package contains interfaces for the Initiate Track service.

7.8.6.3.1 Initiate_Track_CMS

Type: IDLInterface common_use_case_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 7.198 - Methods of IDLInterface 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**

7.8.6.3.2 Initiate_Track_Sub

Type: IDLInterface
Package: Initiate_Track
This is the Subsystem interface for initiating tracks.

Table 7.199 - Methods of IDLInterface Initiate_Track_Sub

Method	Notes	Parameters
initiate_track()	Method used by the CMS to send an "externally designated track	request_id_type **request_id** system_track_type **track_info**

	initiation request", specifying a timed position and kinematic.	

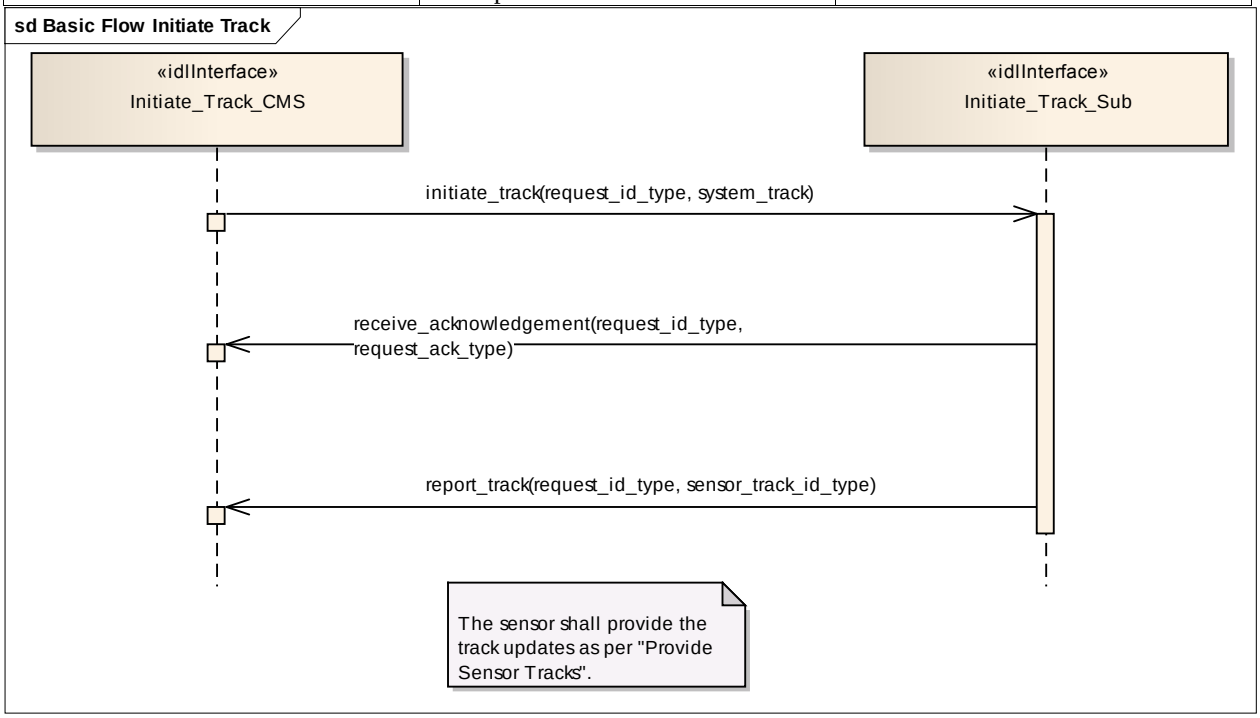

Figure 7.134 Basic Flow Initiate Track (Sequence diagram)

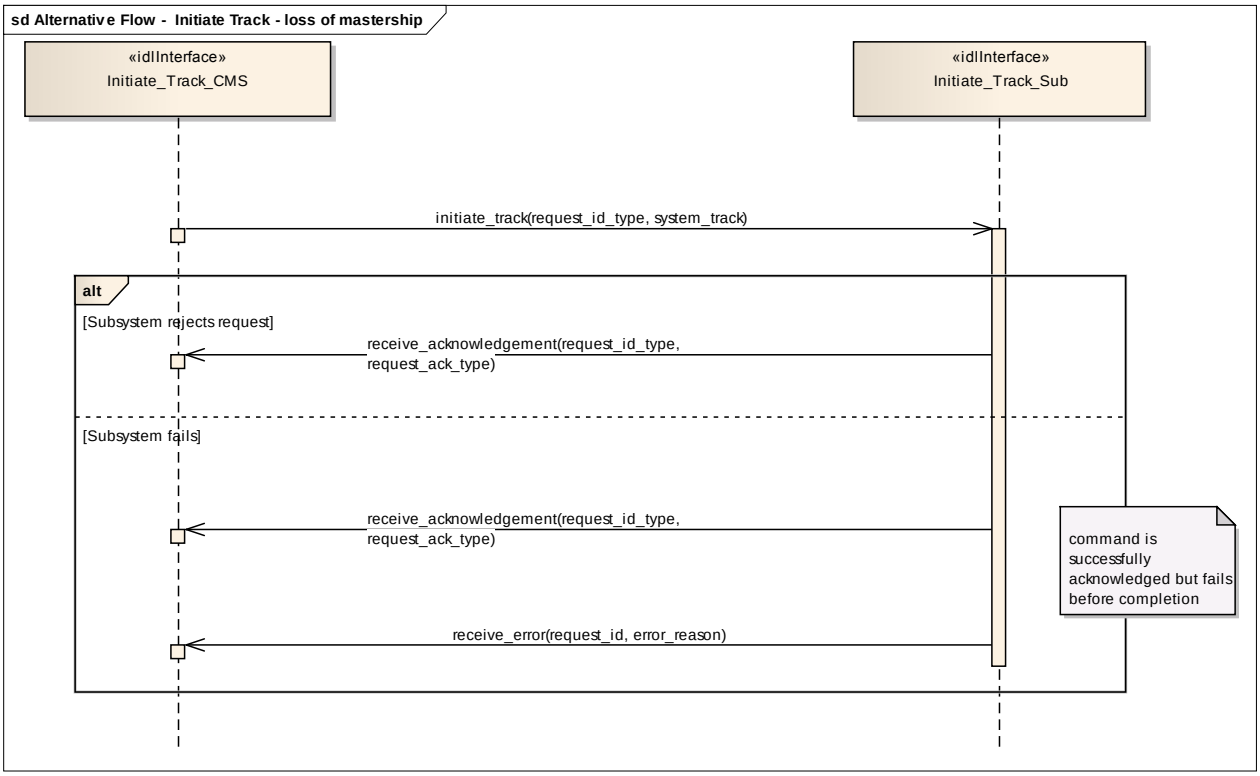

Figure 7.135 Alternative Flow - Initiate Track - loss of mastership (Sequence diagram)

7.8.6.4 Manage_Tracking_Zones

Parent Package: Tracking_Control
This package contains interfaces for the Manage Tracking Zones service.

7.8.6.4.1 Manage_Tracking_Zones_CMS

Type: IDLInterface common_use_case_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 7.200 - Methods of IDLInterface 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 **setting_message**

7.8.6.4.2 Manage_Tracking_Zones_Sub

Type: IDLInterface
Package: Manage_Tracking_Zones
This is the Subsystem interface for managing tracking zones.

Table 7.201 - Methods of IDLInterface 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 **zone**

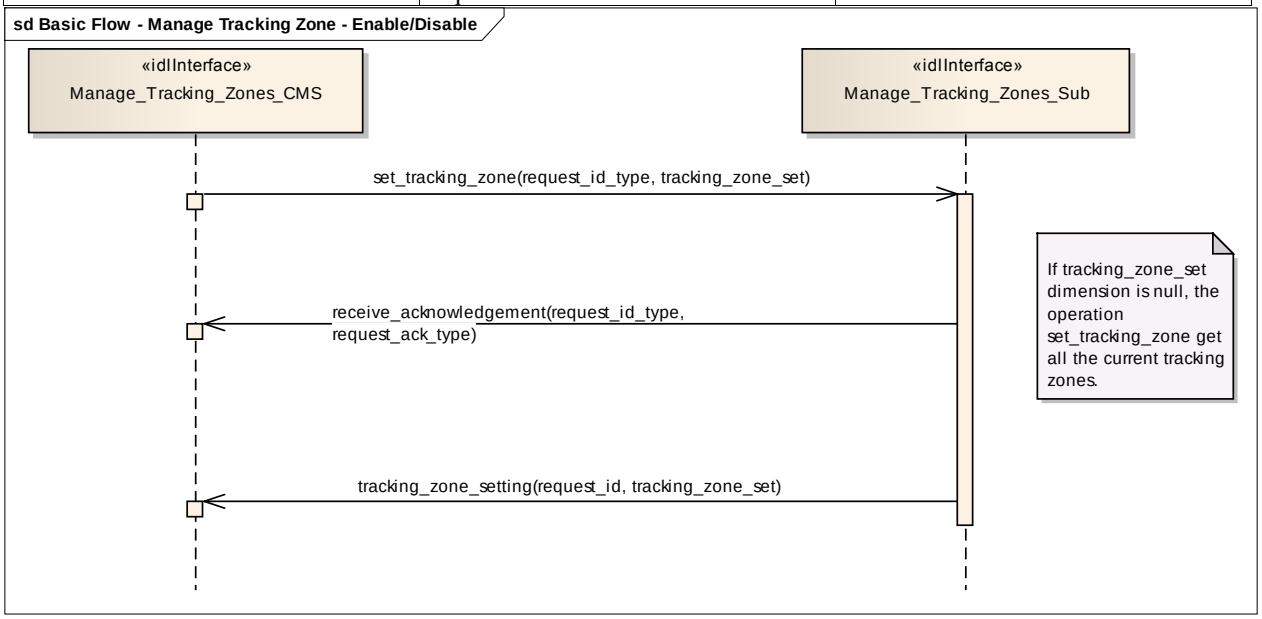

Figure 7.136 Basic Flow - Manage Tracking Zone - Enable/Disable (Sequence diagram)

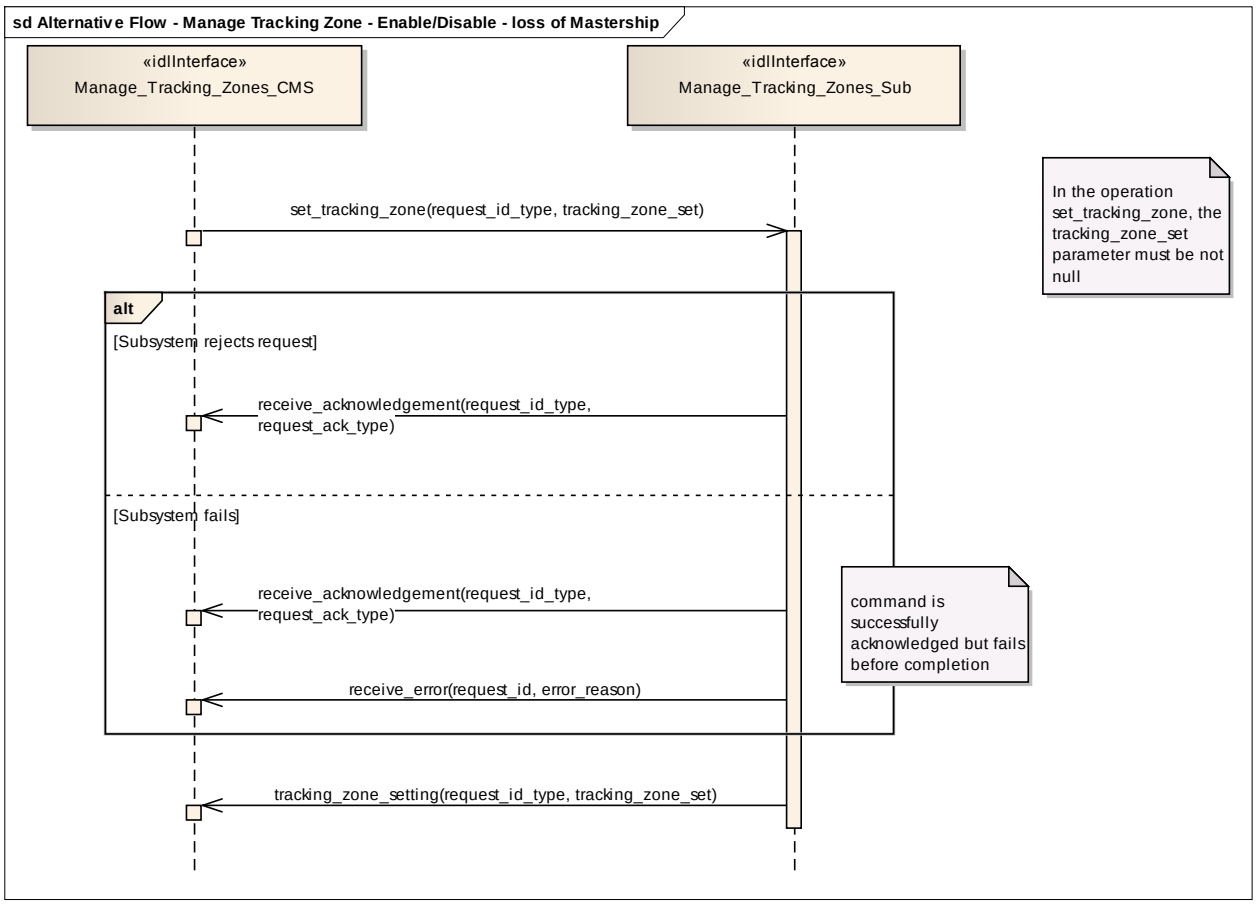

Figure 7.137 Alternative Flow - Manage Tracking Zone - Enable/Disable - loss of Mastership (Sequence diagram)

7.9 Radar_Services

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

7.9.1 Air_Engagement_Support

Parent Package: Radar_Services

7.9.1.1 Provide_Projectile_Positional_Information

Parent Package: Air_Engagement_Support

7.9.1.1.1 Provide_Projectile_Positional_Information_CMS

Type: IDLInterface common_use_case_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 7.202 - Methods of IDLInterface 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**

7.9.1.1.2 Provide_Projectile_Positional_Information_Sub

Type: IDLInterface
Package: Provide_Projectile_Positional_Information

Table 7.203 - Methods of IDLInterface 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**

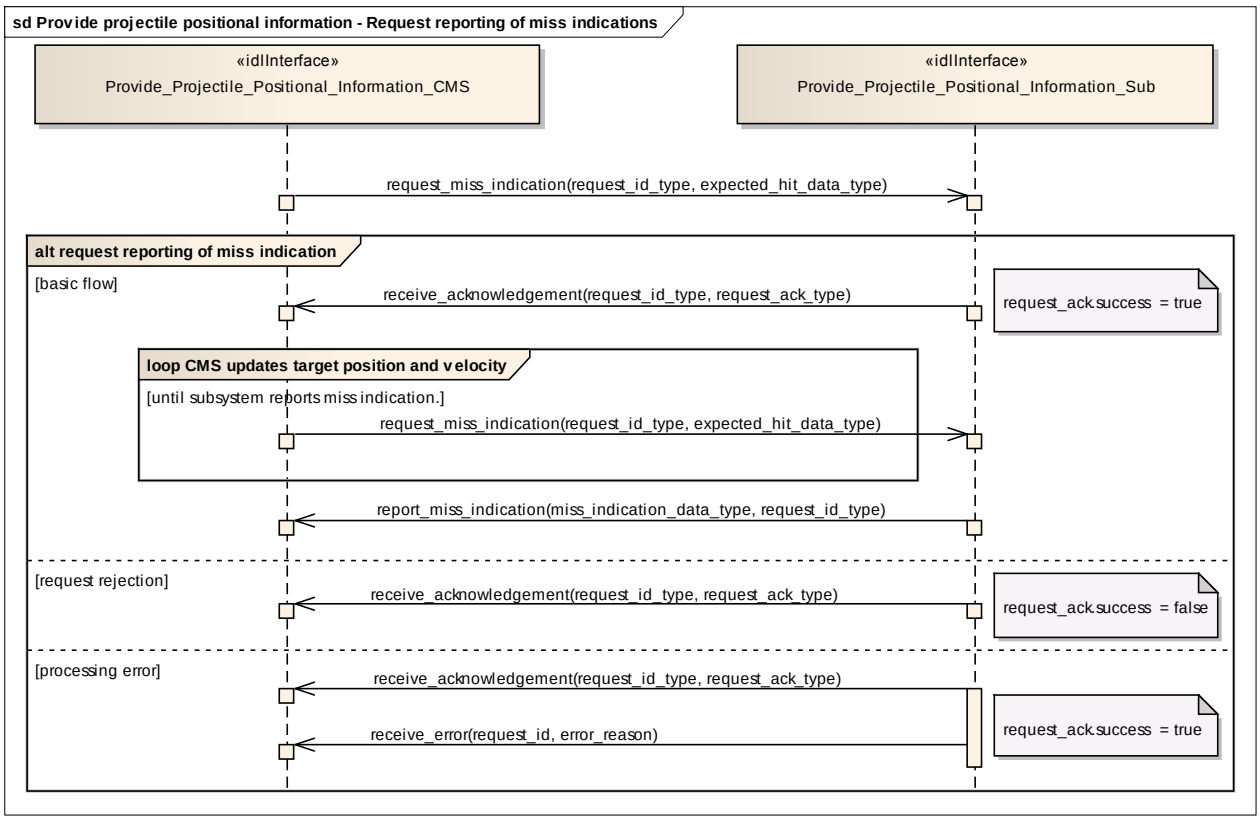

Figure 7.138 Provide projectile positional information - Request reporting of miss indications (Sequence diagram)

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'.

7.9.2 Engagement_Support

Parent Package: Radar_Services

7.9.2.1 Process_Target_Designation

Parent Package: Engagement_Support

7.9.2.1.1 Process_Target_Designation_CMS

Type: IDLInterface common_use_case_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 7.204 - Methods of IDLInterface 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**

7.9.2.1.2 Process_Target_Designation_Sub

Type: IDLInterface
Package: Process_Target_Designation

Table 7.205 - Methods of IDLInterface 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**

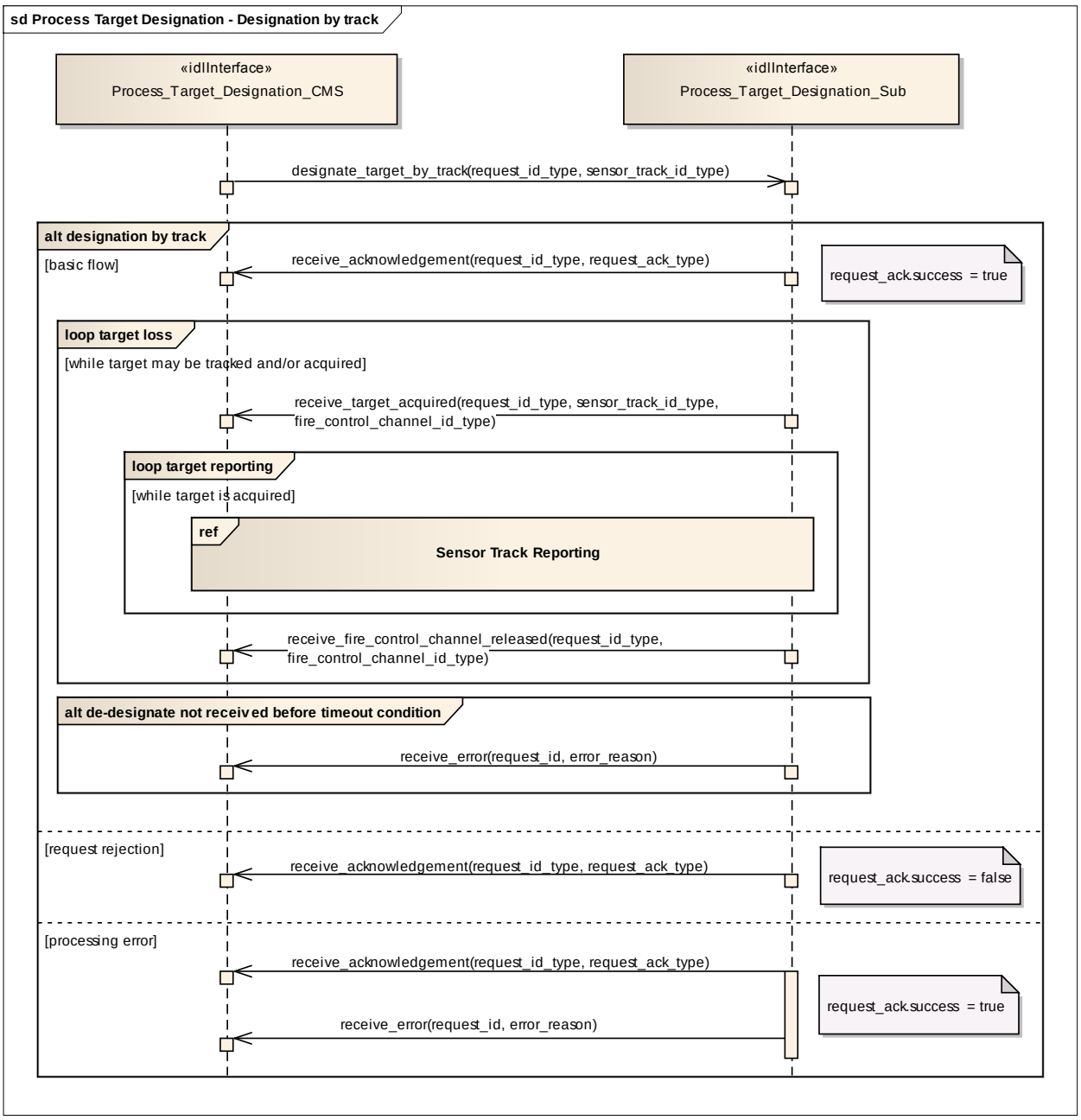

Figure 7.139 Process Target Designation - Designation by track (Sequence diagram)

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".

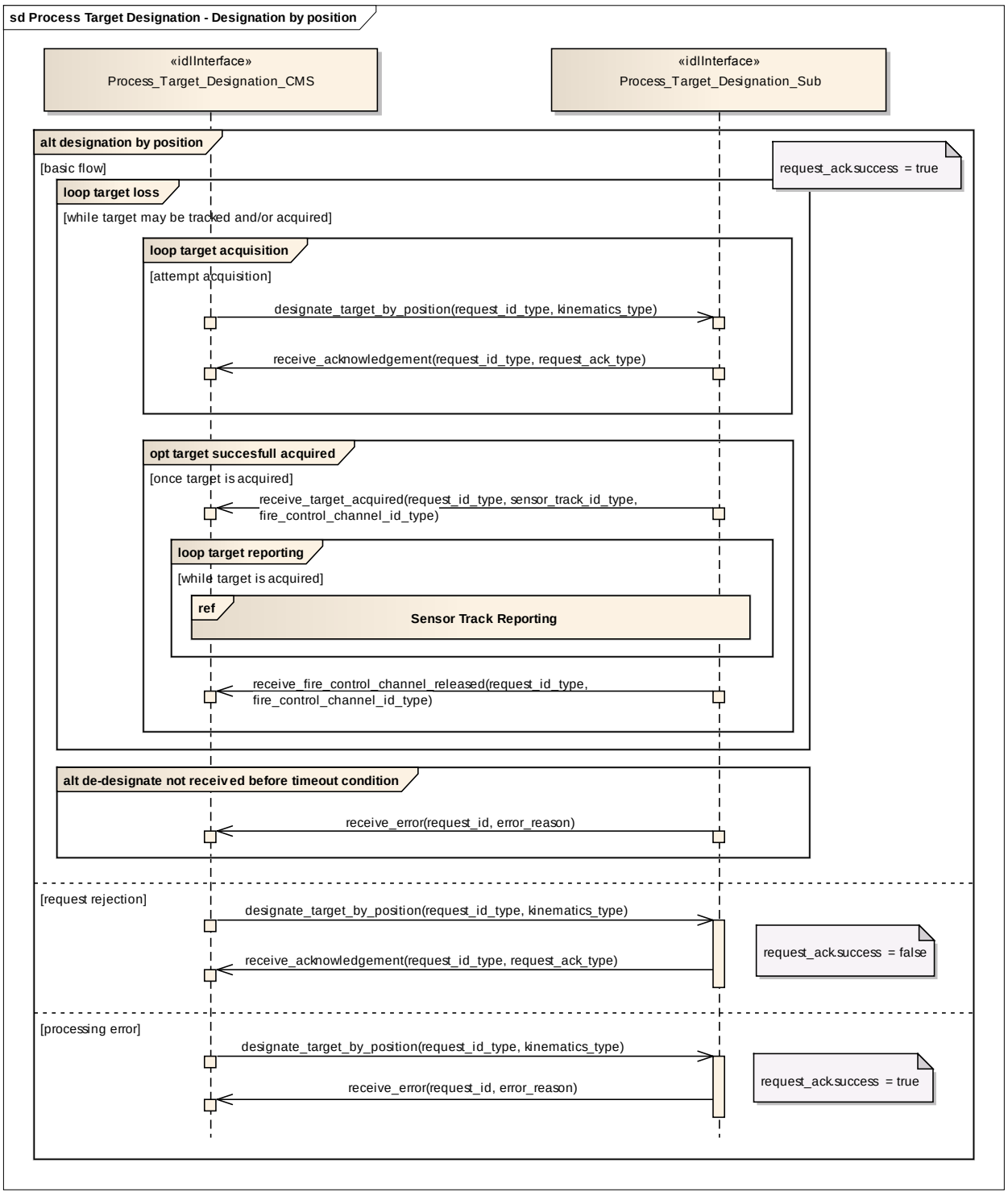

Figure 7.140 Process Target Designation - Designation by position (Sequence diagram)

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".

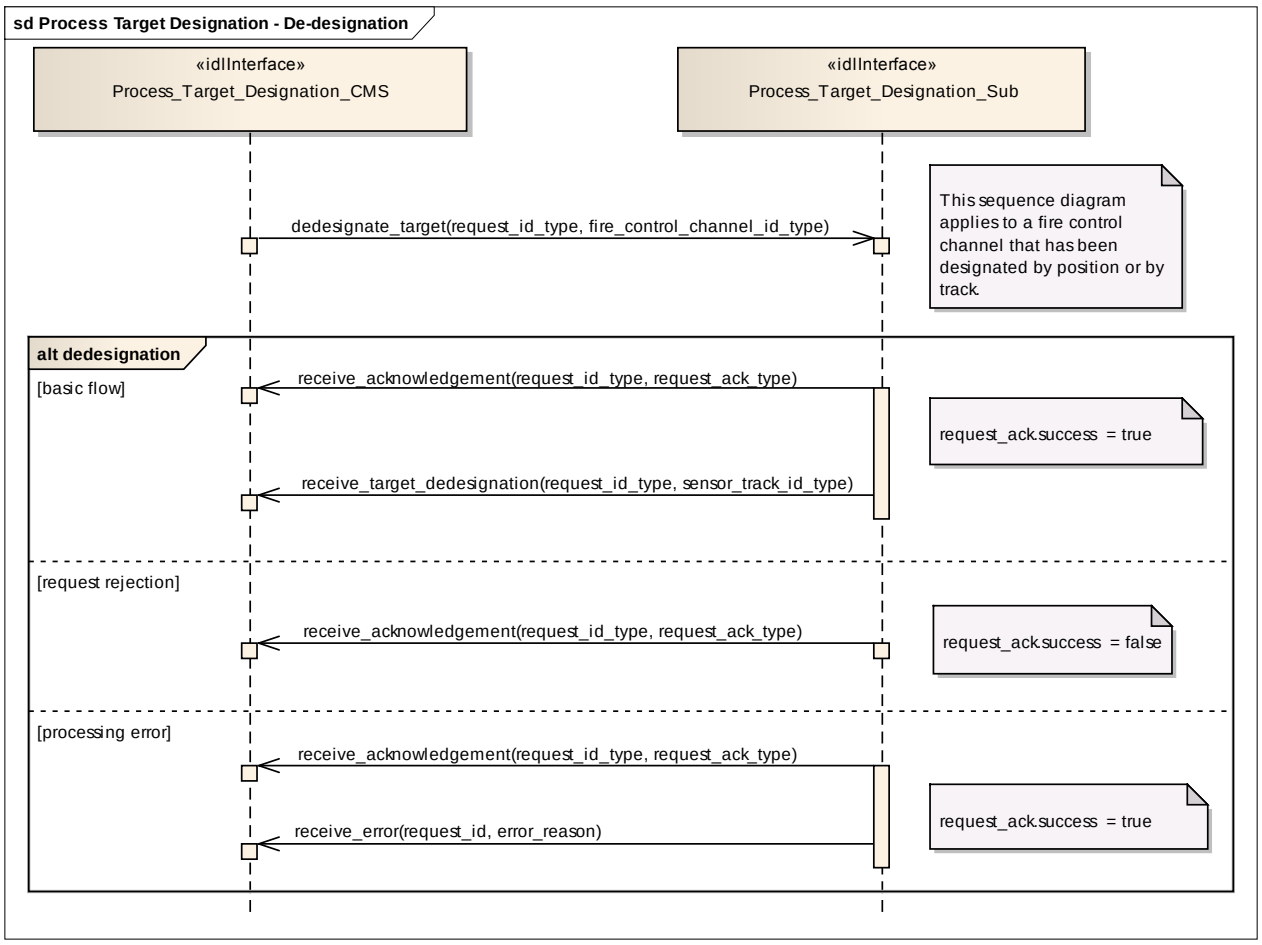

Figure 7.141 Process Target Designation - De-designation (Sequence diagram)

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.

7.9.2.2 Support_Kill_Assessment

Parent Package: Engagement_Support

7.9.2.2.1 Support_Kill_Assessment_CMS

Type: IDLInterface common_use_case_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 7.206 - Methods of IDLInterface 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**

7.9.2.2.2 Support_Kill_Assessment_Sub

Type: IDLInterface
Package: Support_Kill_Assessment

Table 7.207 - Methods of IDLInterface 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**

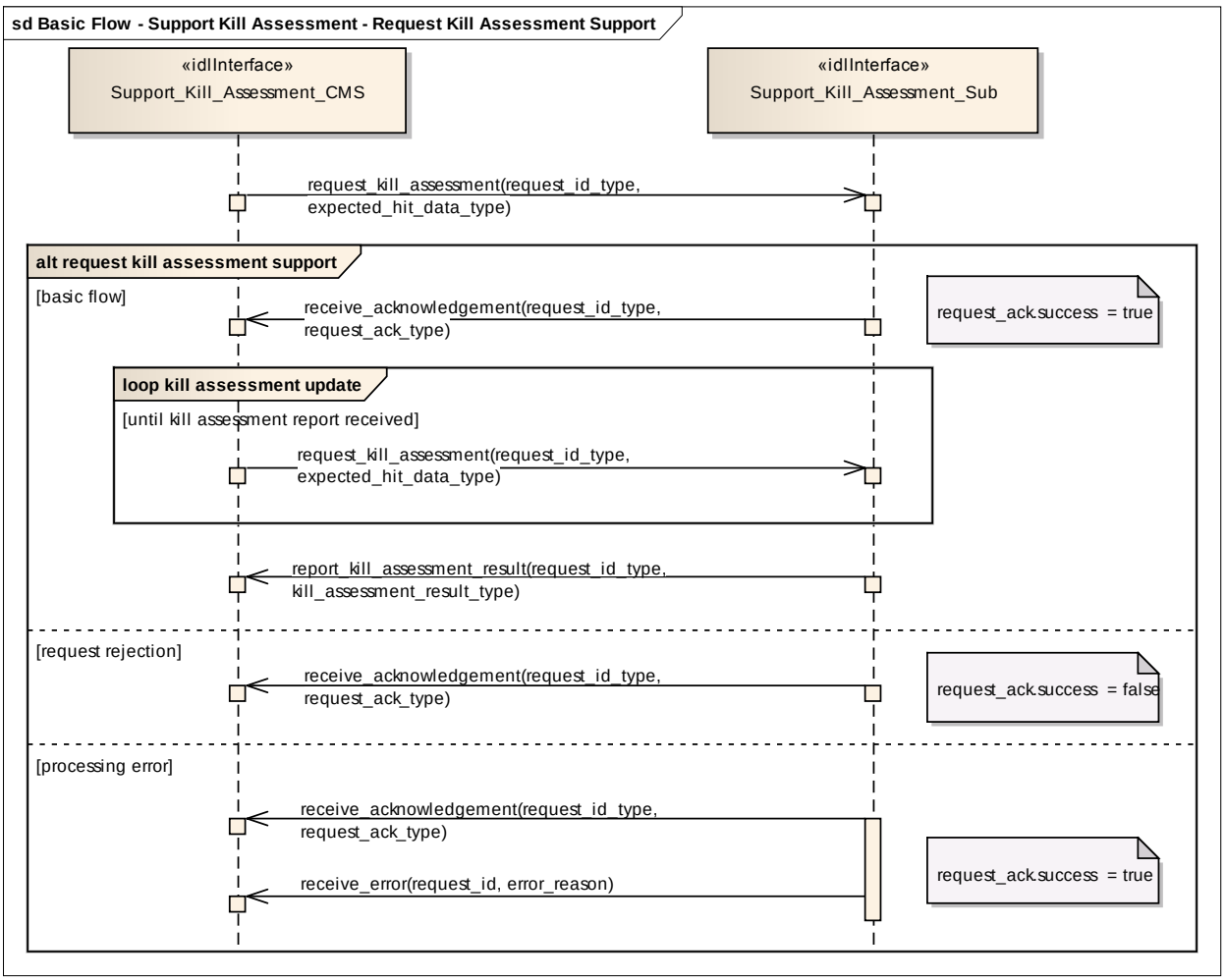

Figure 7.142 Basic Flow - Support Kill Assessment - Request Kill Assessment Support (Sequence diagram)

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".

7.9.2.3 Support_Surface_Target_Engagement

Parent Package: Engagement_Support

7.9.2.3.1 Support_Surface_Target_Engagement_CMS

Type: IDLInterface common_use_case_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 7.208 - Methods of IDLInterface 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**

7.9.2.3.2 Support_Surface_Target_Engagement_Sub

Type: IDLInterface
Package: Support_Surface_Target_Engagement

Table 7.209 - Methods of IDLInterface 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_contr	Request to the subsystem to report	request_id_type **RequestID**

ol_channels()	the available fire control channels.	

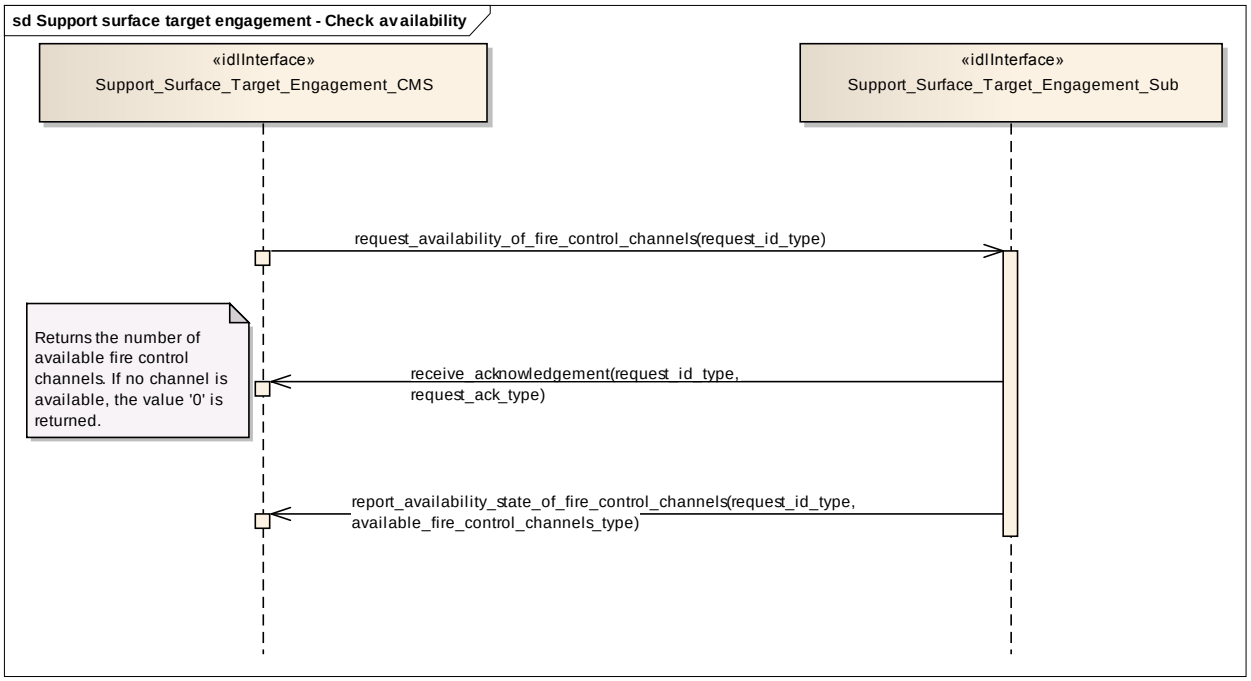

Figure 7.143 Support surface target engagement - Check availability (Sequence diagram)

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".

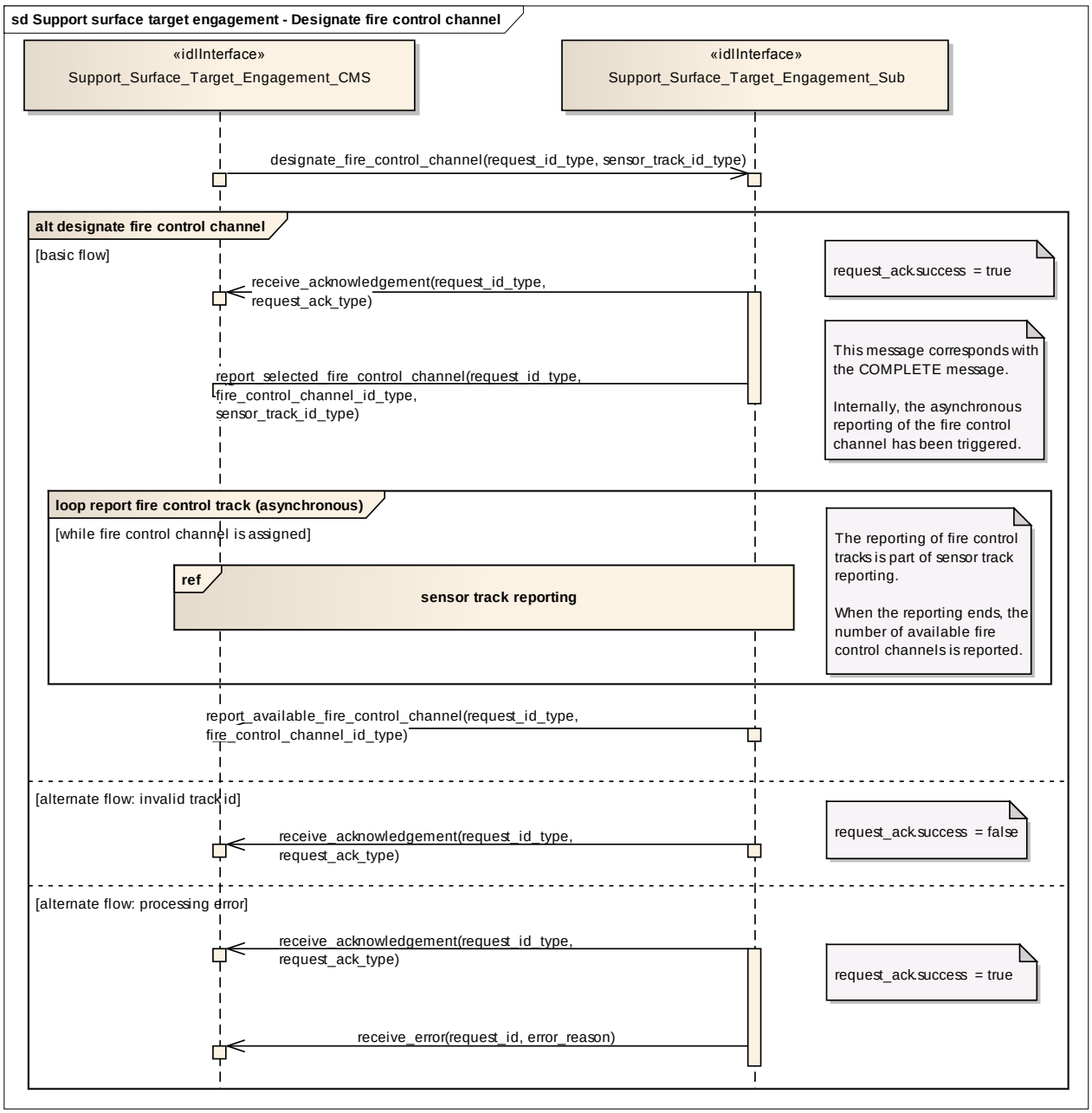

Figure 7.144 Support surface target engagement - Designate fire control channel (Sequence diagram)

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".

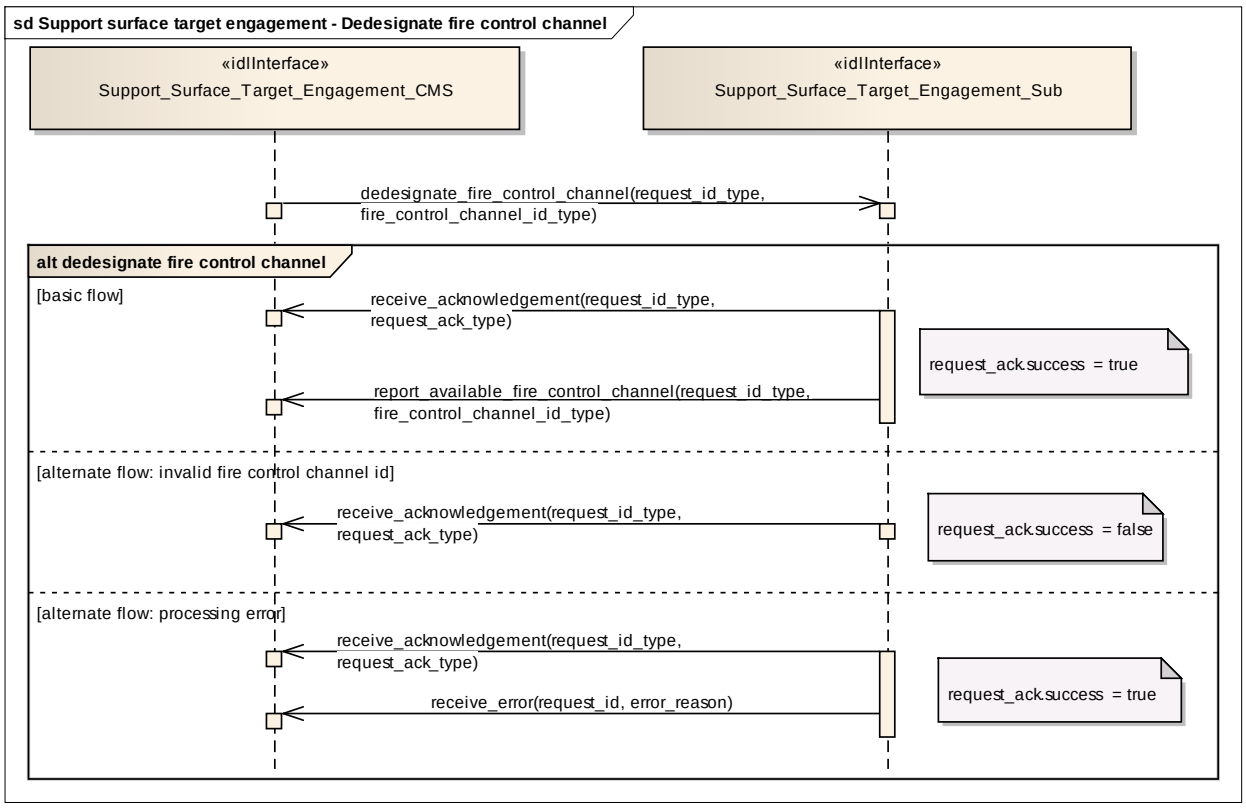

Figure 7.145 Support surface target engagement - Dedesignate fire control channel (Sequence diagram)

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".

7.9.3 Missile_Guidance

Parent Package: Radar_Services

7.9.3.1 Perform_Illumination

Parent Package: Missile_Guidance

7.9.3.1.1 Perform_Illumination_CMS

Type: IDLInterface common_use_case_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 7.210 - Methods of IDLInterface Perform_Illumination_CMS

Method	Notes	Parameters
complete()		request_id_type **request_id**

7.9.3.1.2 Perform_Illumination_Sub

Type: IDLInterface
Package: Perform_Illumination

Table 7.211 - Methods of IDLInterface Perform_Illumination_Sub

Method	Notes	Parameters
request_illumination()		request_id_type **request_id** illumination_request_type **request**
provide_track()		system_track_type **track**

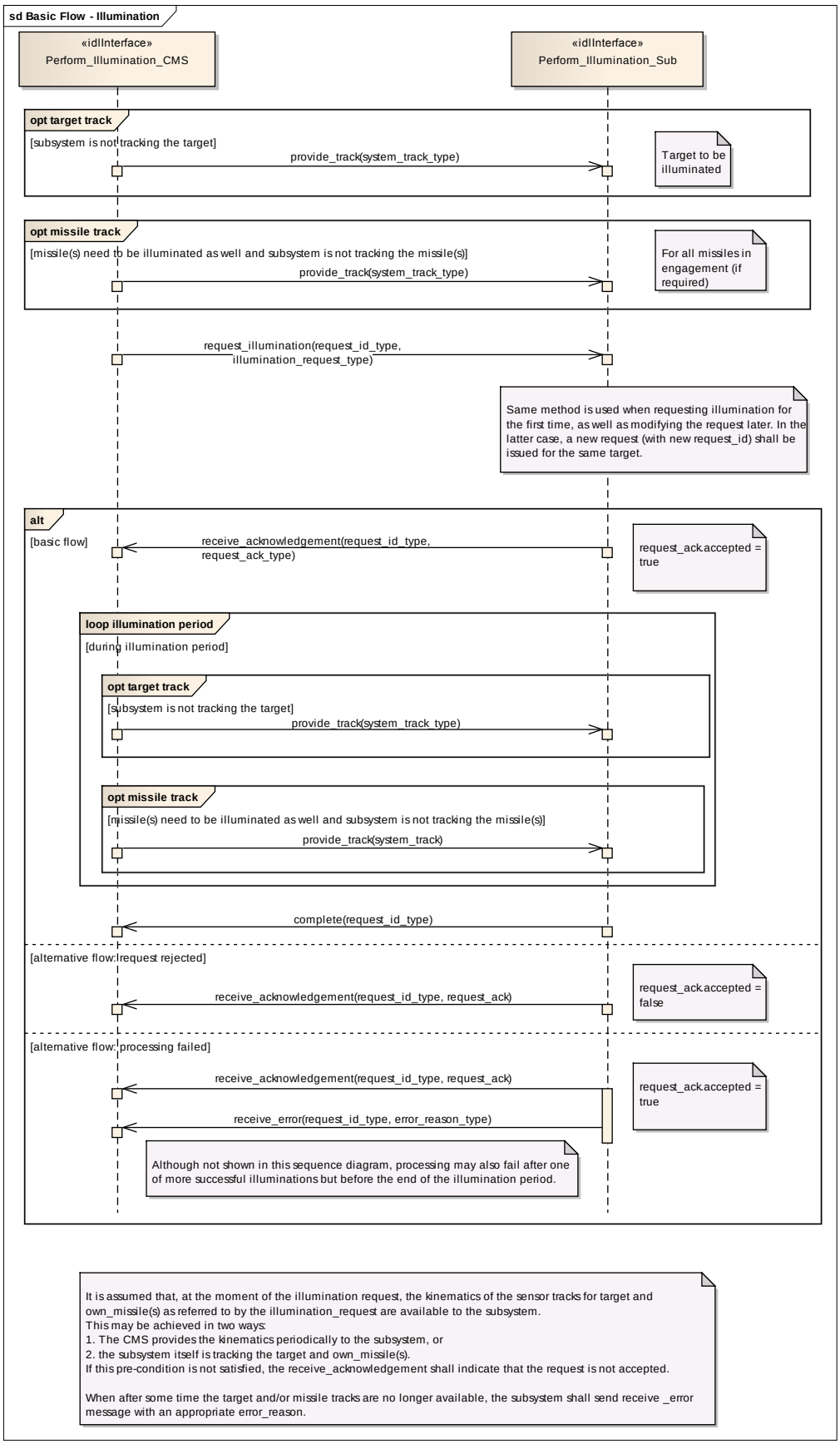
sd Basic Flow - Illumination
«idlInterface»
Perform_Illumination_CMS
«idlInterface»
Perform_Illumination_Sub
opt target track
[subsystem is not tracking the target]
provide_track(system_track_type)
Target to be illuminated
opt missile track
[missile(s) need to be illuminated as well and subsystem is not tracking the missile(s)]
provide_track(system_track_type)
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_type)
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 7.146 Basic Flow - Illumination (Sequence diagram)

7.9.3.2 Perform_Missile_Downlink

Parent Package: Missile_Guidance

7.9.3.2.1 Perform_Missile_Downlink_CMS

Type: IDLInterface common_use_case_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[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: 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 7.212 - Methods of IDLInterface Perform_Missile_Downlink_CMS

Method	Notes	Parameters
report_downlink()		request_id_type **request_id** downlink_report **the_downlink_info**
complete()		request_id_type **request_id**

7.9.3.2.2 Perform_Missile_Downlink_Sub

Type: IDLInterface
Package: Perform_Missile_Downlink

Table 7.213 - Methods of IDLInterface Perform_Missile_Downlink_Sub

Method	Notes	Parameters
request_downlink()		request_id_type **request_id** downlink_request **request**
provide_track()		system_track_type **track**

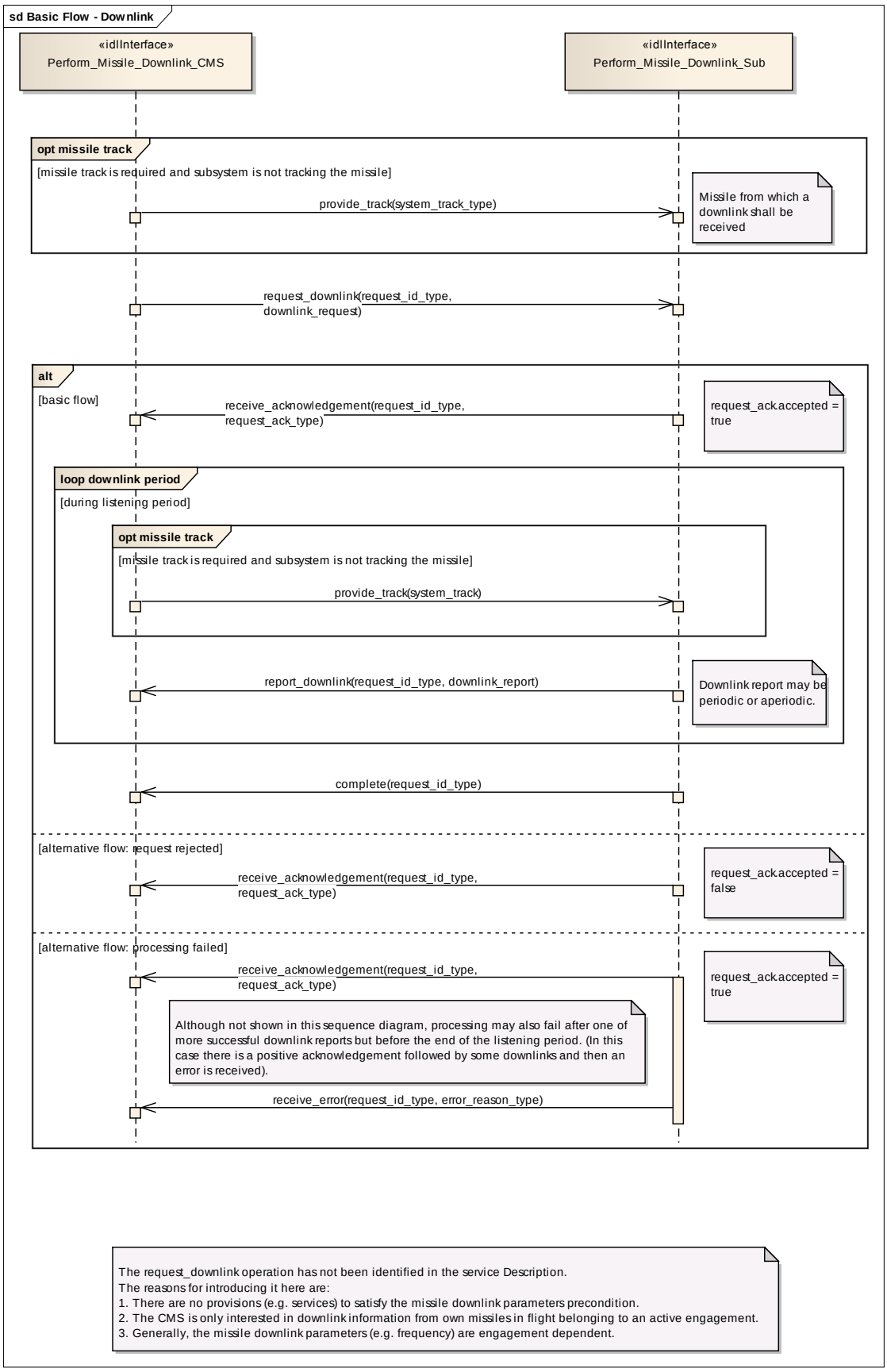
sd Basic Flow - Downlink
«idlInterface»
Perform_Missile_Downlink_CMS
«idlInterface»
Perform_Missile_Downlink_Sub
opt missile track
[missile track is required and subsystem is not tracking the missile]
provide_track(system_track_type)
Missile from which a downlink shall be received
request_downlink(request_id_type, downlink_request)
alt
[basic flow]
receive_acknowledgement(request_id_type, request_ack_type)
request_ack.accepted = true
loop downlink period
[during listening period]
opt missile track
[missile track is required and subsystem is not tracking the missile]
provide_track(system_track)
report_downlink(request_id_type, downlink_report)
Downlink report may be periodic or aperiodic.
complete(request_id_type)
[alternative flow: request rejected]
receive_acknowledgement(request_id_type, request_ack_type)
request_ack.accepted = false
[alternative flow: processing failed]
receive_acknowledgement(request_id_type, request_ack_type)
request_ack.accepted = true
Although not shown in this sequence diagram, processing may also fail after one of more successful downlink reports but before the end of the listening period. (In this case there is a positive acknowledgement followed by some downlinks and then an error is received).
receive_error(request_id_type, error_reason_type)
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 7.147 Basic Flow - Downlink (Sequence diagram)

7.9.3.3 Perform_Missile_Uplink

Parent Package: Missile_Guidance

7.9.3.3.1 Perform_Missile_Uplink_CMS

Type: IDLInterface common_use_case_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 7.214 - Methods of IDLInterface Perform_Missile_Uplink_CMS

Method	Notes	Parameters
report_uplink_completed()		request_id_type **request_id** uplink_report_type **report**

7.9.3.3.2 Perform_Missile_Uplink_Sub

Type: IDLInterface
Package: Perform_Missile_Uplink

Table 7.215 - Methods of IDLInterface Perform_Missile_Uplink_Sub

Method	Notes	Parameters
request_uplink()		request_id_type **request_id** uplink_request_type **request**
provide_track()		system_track_type **track**

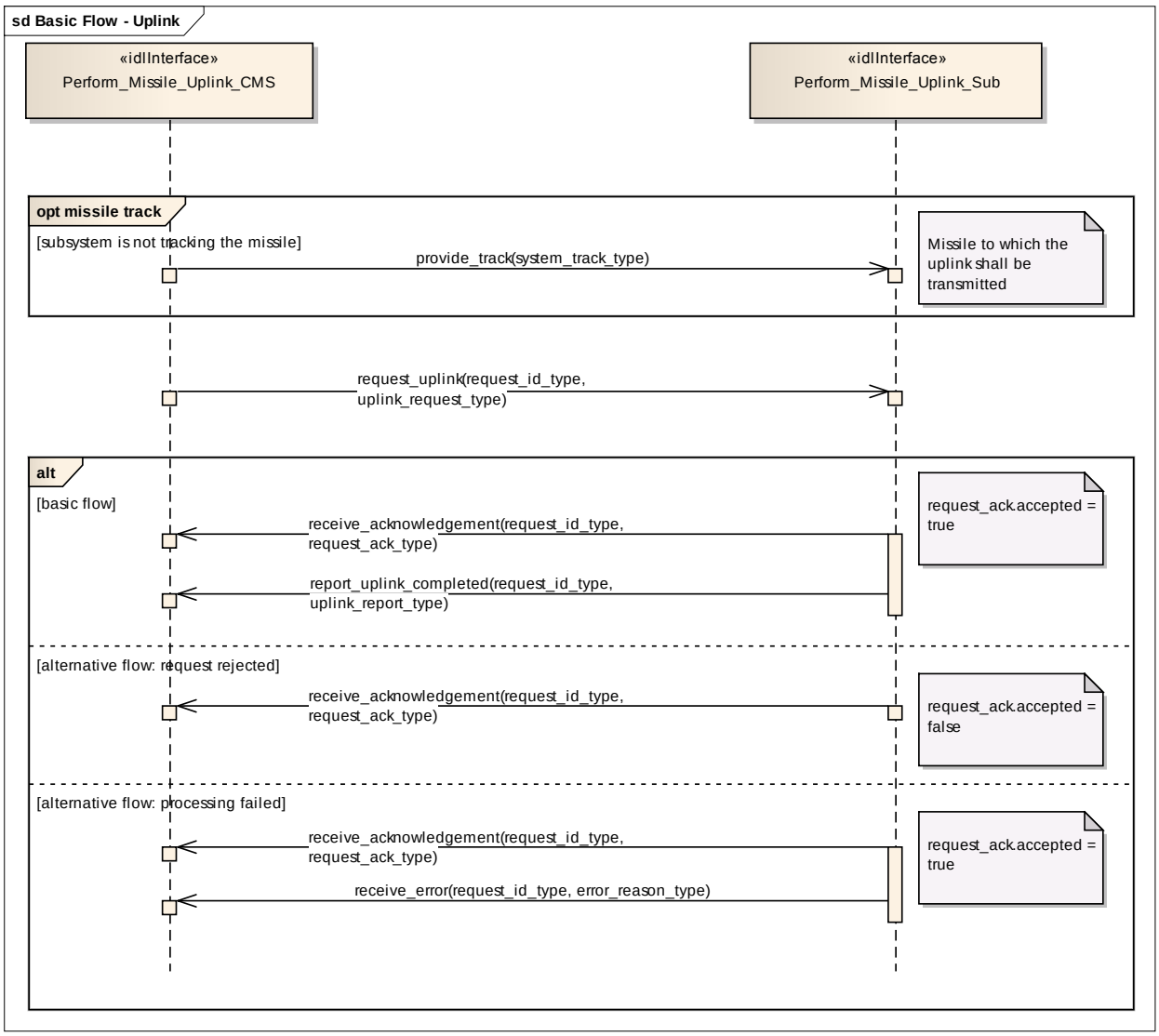

Figure 7.148 Basic Flow - Uplink (Sequence diagram)

7.9.4 Search

Parent Package: Radar_Services

7.9.4.1 Perform_Cued_Search

Parent Package: Search

7.9.4.1.1 Perform_Cued_Search_CMS

Type: IDLInterface common_use_case_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 7.216 - Methods of IDLInterface 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.

7.9.4.1.2 Perform_Cued_Search_Sub

Type: IDLInterface
Package: Perform_Cued_Search
The Subsystem Search Interface.

Table 7.217 - Methods of IDLInterface 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.

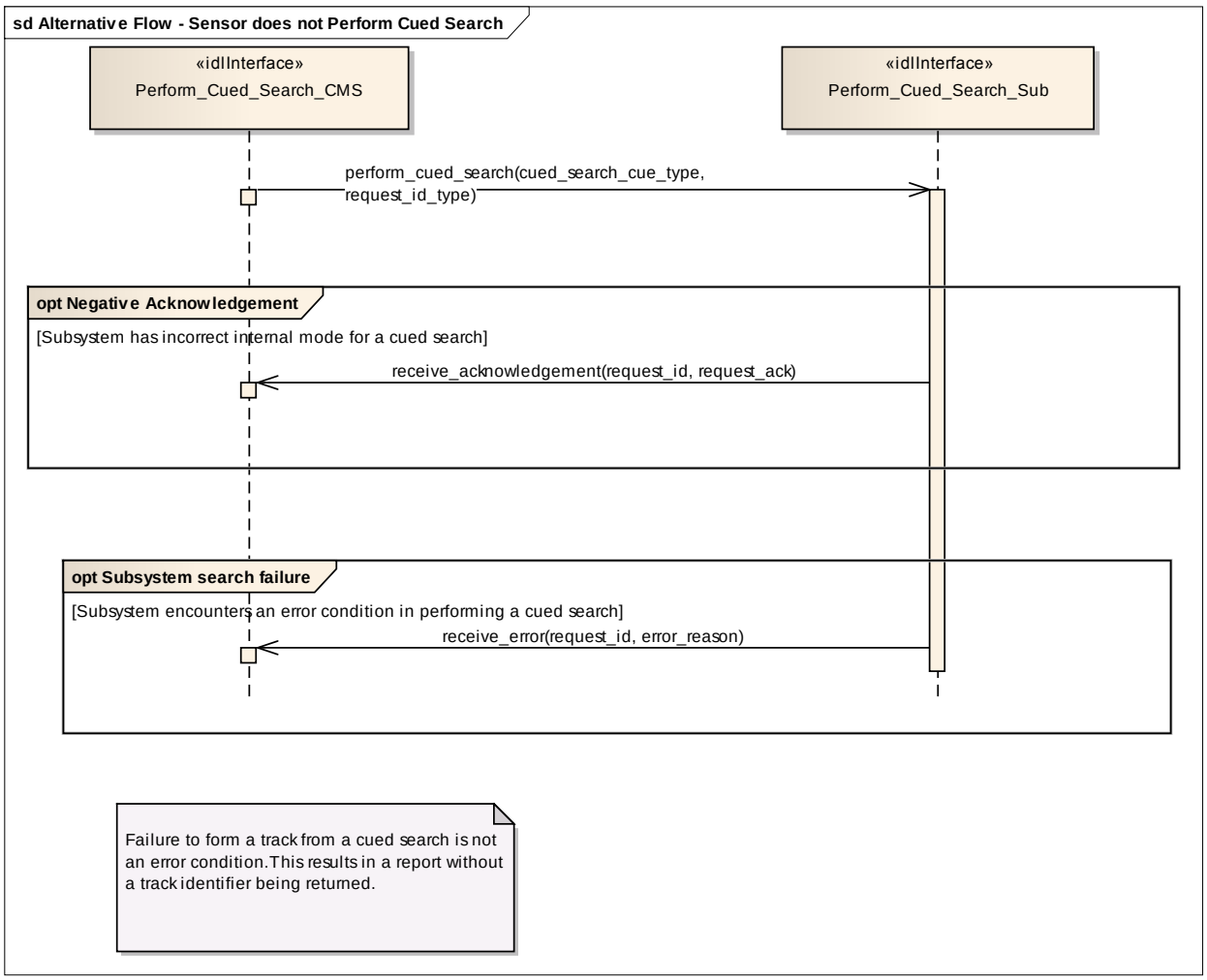

Figure 7.149 Alternative Flow - Sensor does not Perform Cued Search (Sequence diagram)

Figure 7.150 Basic Flow - Perform Cued Search (Sequence diagram)

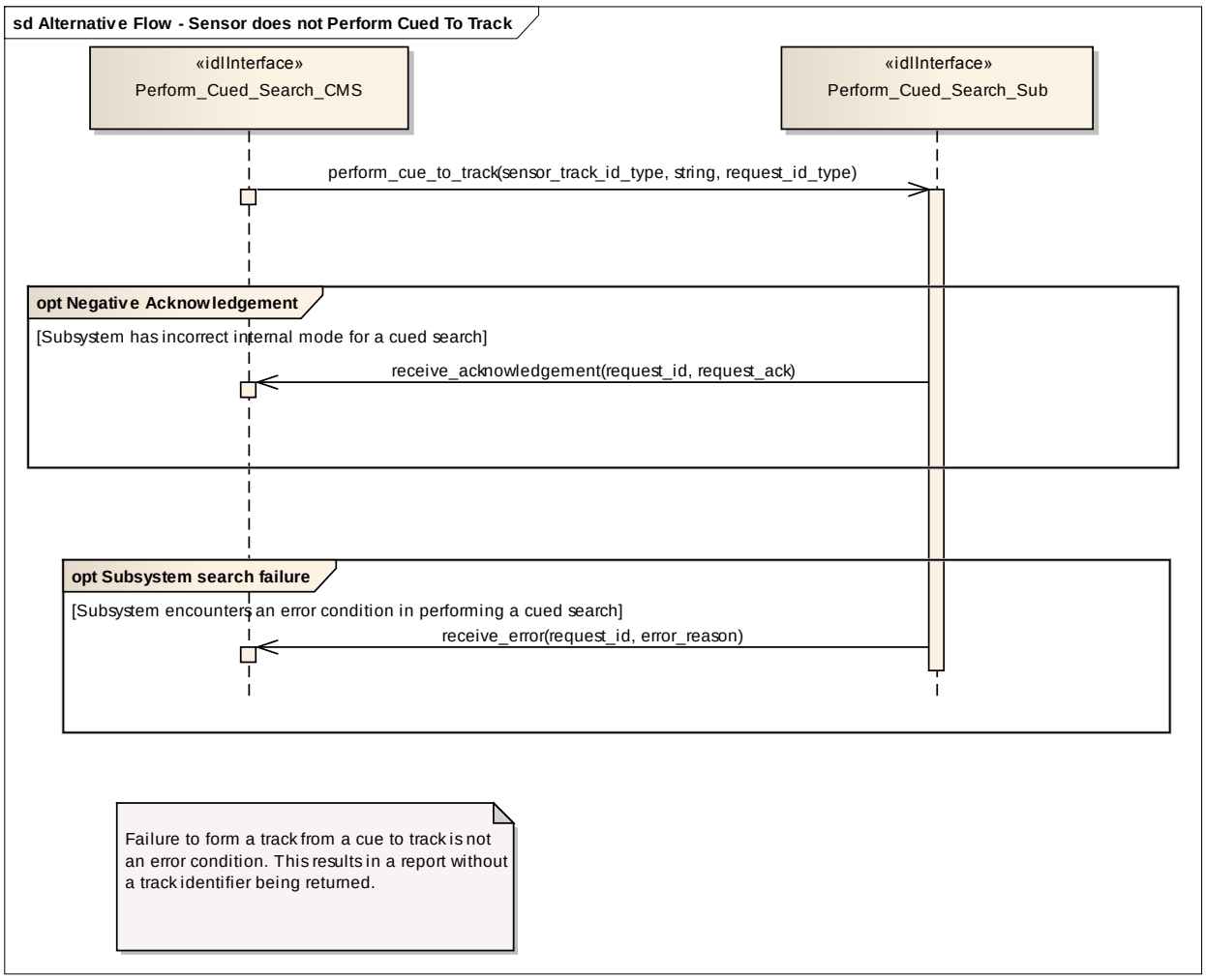

Figure 7.151 Alternative Flow - Sensor does not Perform Cued To Track (Sequence diagram)

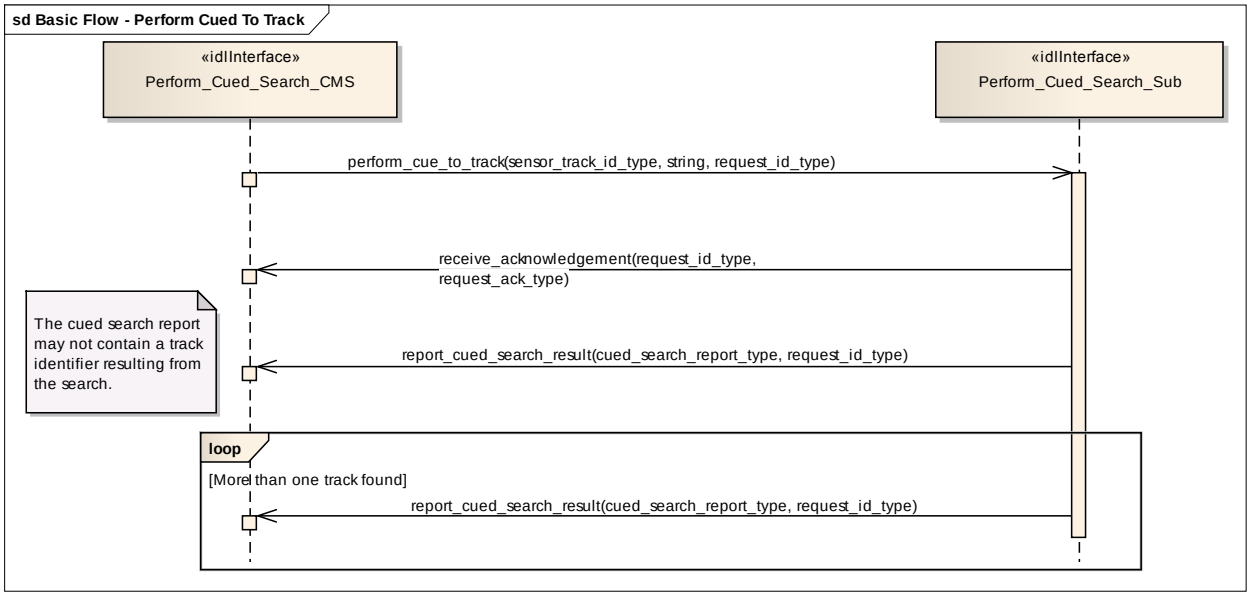

Figure 7.152 Basic Flow - Perform Cued To Track (Sequence diagram)

7.9.5 Surface_Engagement_Support

Parent Package: Radar_Services

7.9.5.1 Perform_Splash_Spotting

Parent Package: Surface_Engagement_Support

7.9.5.1.1 Perform_Splash_Spotting_CMS

Type: IDLInterface common_use_case_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 7.218 - Methods of IDLInterface Perform_Splash_Spotting_CMS

Method	Notes	Parameters
confirm_reposition_splash_splotting_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_area_position()	Via this method, the request for a new splash spotting area based on a position is confirmed by the	request_id_type **RequestID** splash_spotting_area_id_type **SplashSpottingAreaID**

	subsystem.	
receive_splash_splotting_area_track()	Via this method, the request for a new splash spotting area based on a track is confirmed by the subsystem.	request_id_type **RequestID** 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**

7.9.5.1.2 Perform_Splash_Spotting_Sub

Type: IDLInterface
Package: Perform_Splash_Spotting

Table 7.219 - Methods of IDLInterface 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**

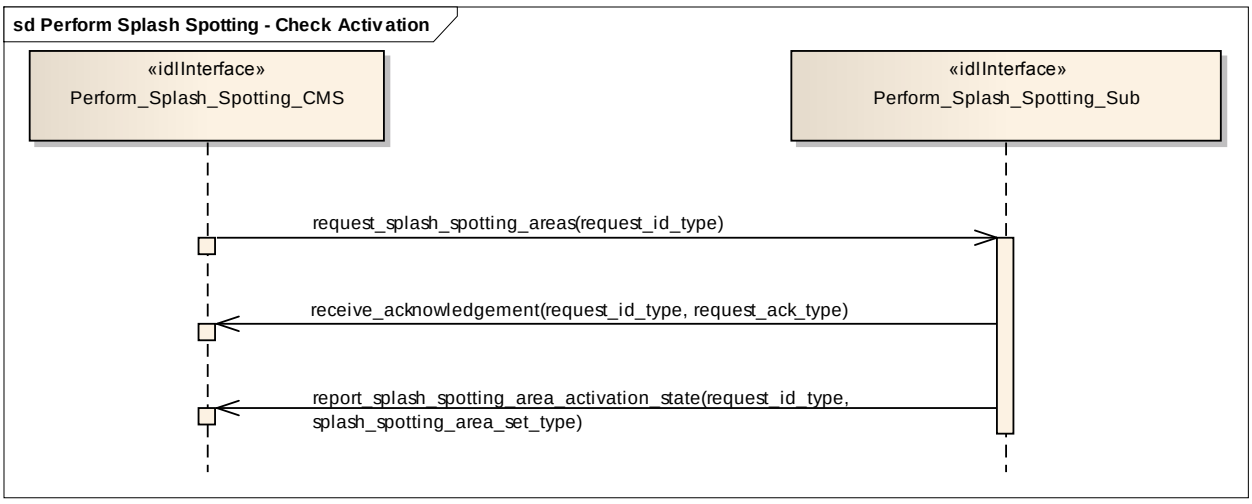

Figure 7.153 Perform Splash Spotting - Check Activation (Sequence diagram)

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".

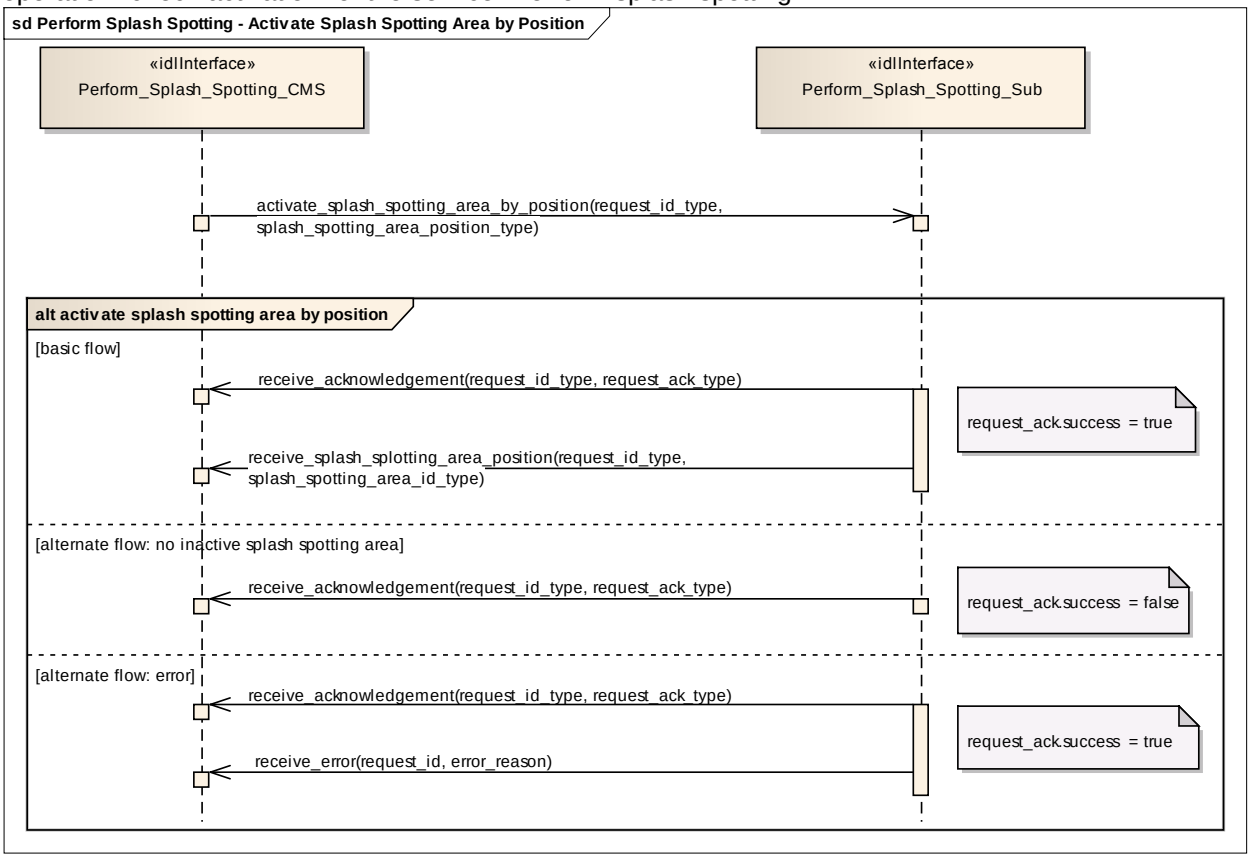

Figure 7.154 Perform Splash Spotting - Activate Splash Spotting Area by Position (Sequence diagram)

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".

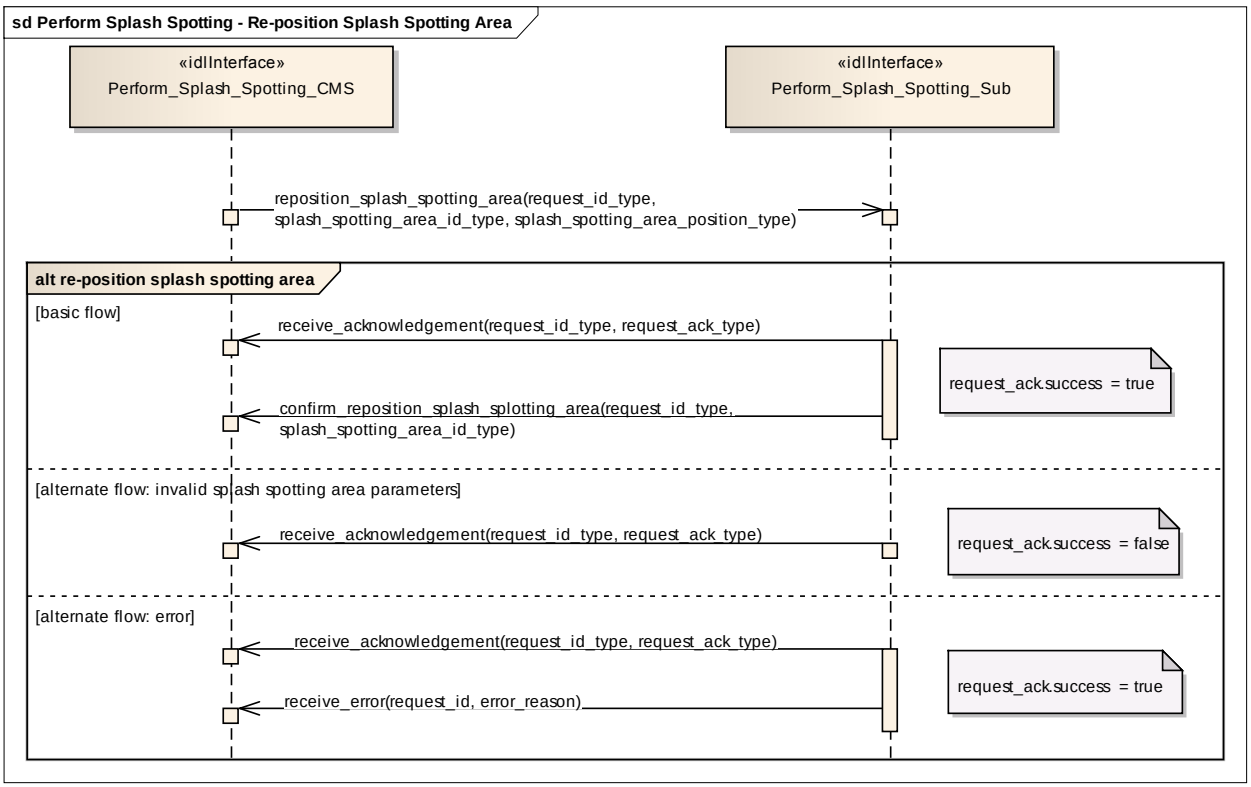

Figure 7.155 Perform Splash Spotting - Re-position Splash Spotting Area (Sequence diagram)

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".

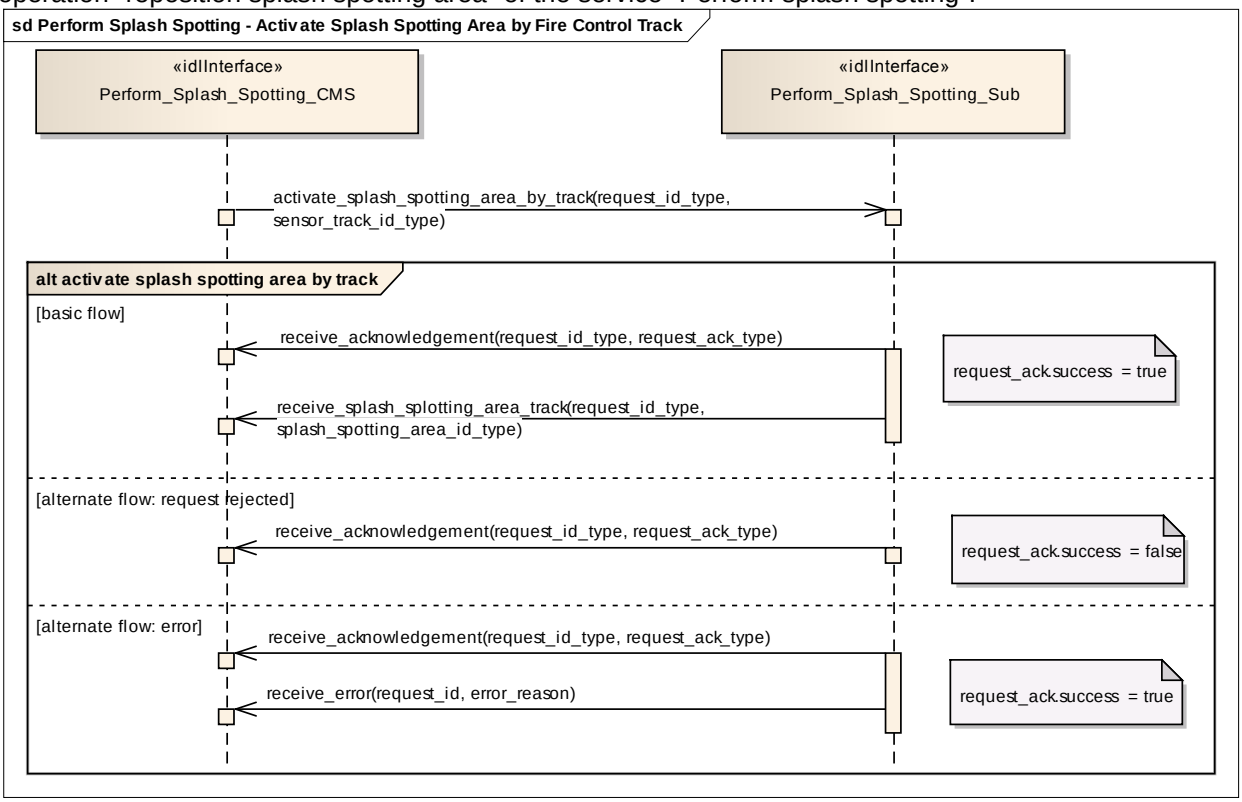

Figure 7.156 Perform Splash Spotting - Activate Splash Spotting Area by Fire Control Track (Sequence diagram)

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".

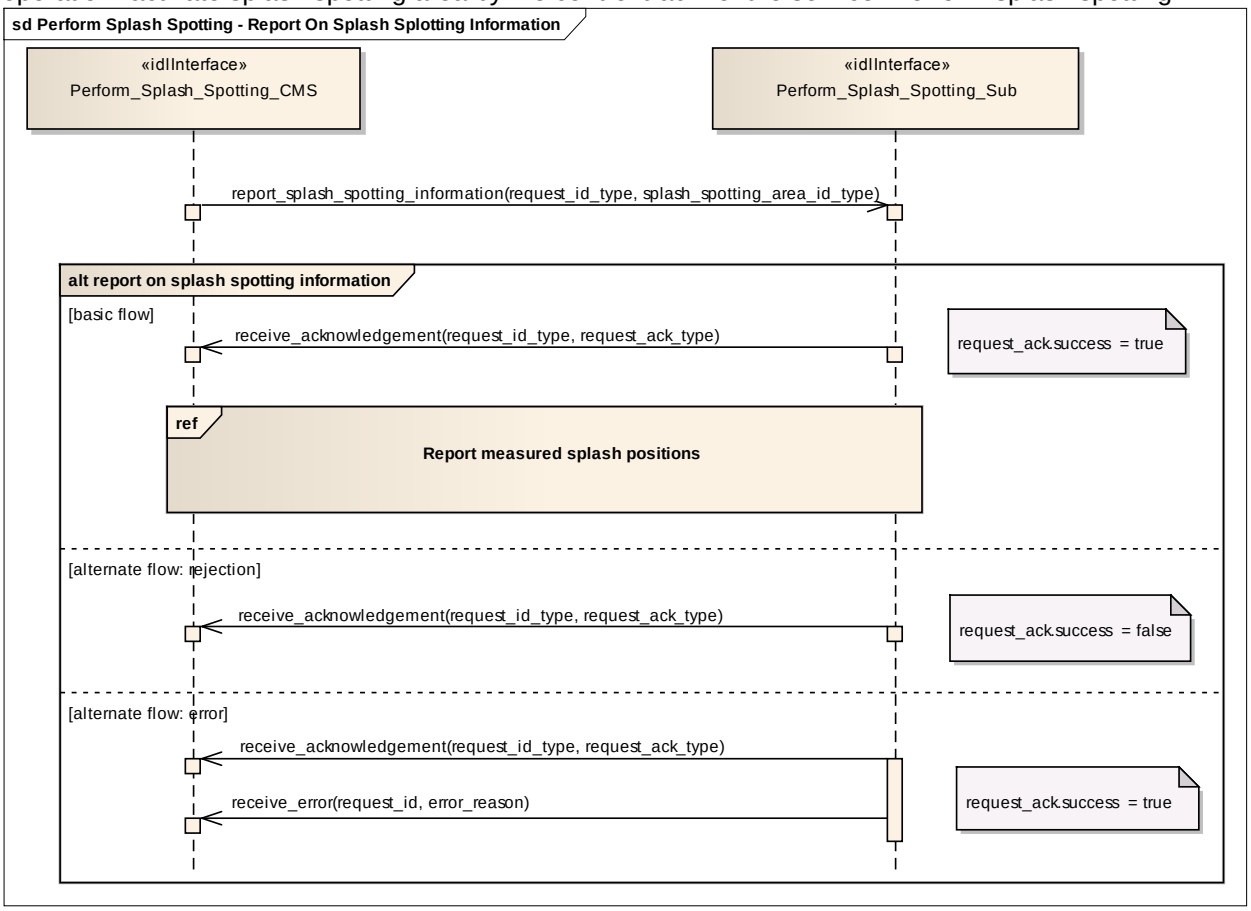

Figure 7.157 Perform Splash Spotting - Report On Splash Splotting Information (Sequence diagram)

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".

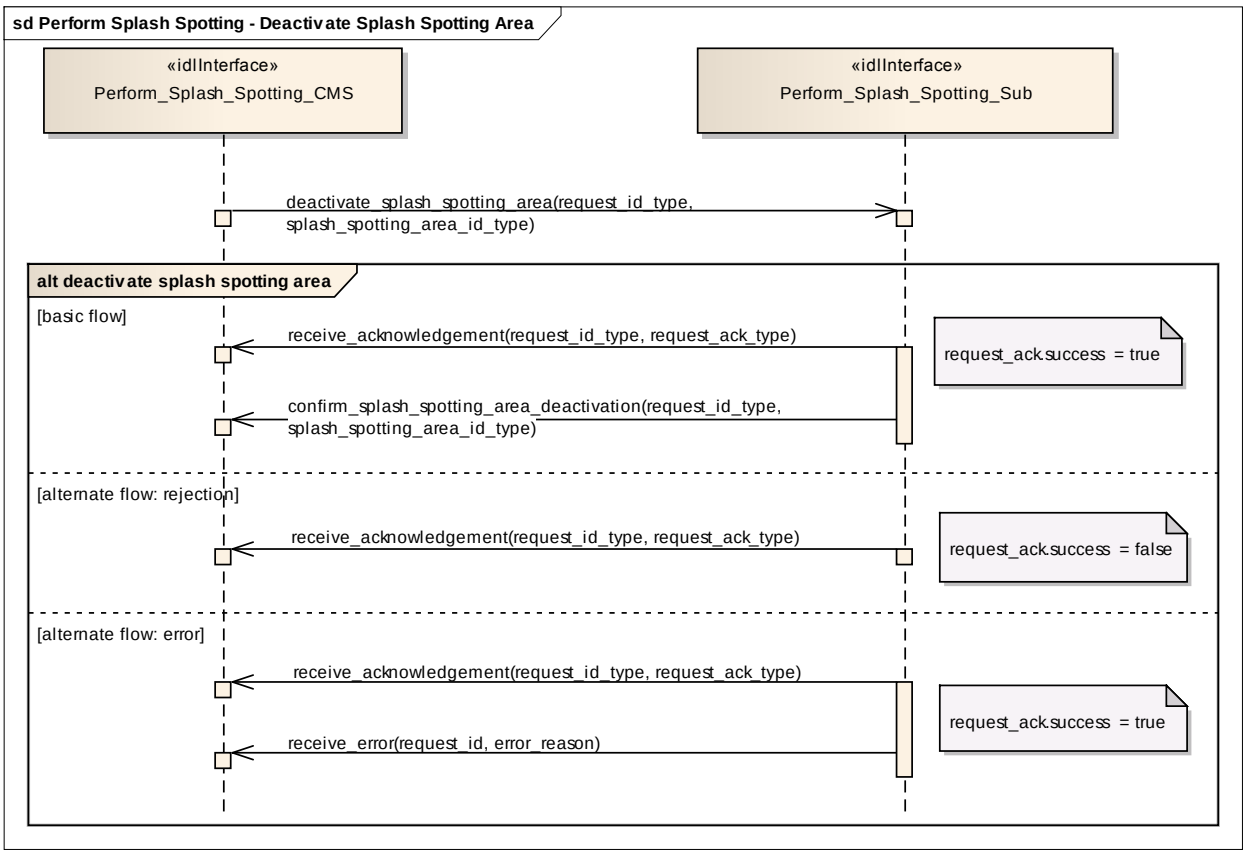

Figure 7.158 Perform Splash Spotting - Deactivate Splash Spotting Area (Sequence diagram)

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:

- The PIM attributes are mapped to IDL attributes;
- Optional 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;
- Specialization / Generalization PIM relationships are mapped to IDL unions. Additional data classes are introduced for 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.

To robustly and efficiently ensure that the data exchanged between a particular subsystem and a CMS is recognised correctly, topic samples pertaining to a particular subsystem are published on the partition corresponding to the name used in the Subsystem Identification use case. 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.