ALert MAnagement Service (ALMAS)

Version 1.4 beta

OMG Document Number: formal/24-01-03 [smsc/24-01-02]

Standard document URL: https://www.omg.org/spec/ALMAS

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 companies 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, 9C Medway Rd, PMB 274, Milford, MA 01757, U.S.A.

TRADEMARKS

CORBA®, CORBA logos®, FIBO®, Financial Industry Business Ontology®, FINANCIAL INSTRUMENT GLOBAL IDENTIFIER®, IIOP®, IMM®, Model Driven Architecture®, MDA®, Object Management Group®, OMG®, OMG Logo®, SoaML®, SOAML®, SysML®, UAF®, Unified Modeling Language®, UML®, UML Cube Logo®, VSIPL®, and XMI® are registered trademarks of the Object Management Group, Inc.

For a complete list of trademarks, see: https://www.omg.org/legal/tm_list.htm. 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 https://www.omg.org, under Specifications, Report a Bug/Issue.

Table of Contents

Preface

About the Object Management Group

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 Meta-model); and industry-specific standards for dozens of vertical markets.

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

OMG Specifications

As noted, OMG specifications address middleware, modeling and vertical domain frameworks. All OMG Formal Specifications are available from this URL: *https://www.omg.org/spec*

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
9C Medway Road, PMB 274
Milford, MA 01757
USA

Tel: +1-781-444-0404
Fax: +1-781-444-0320

Email: *pubs@omg.org*

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

Issues

The reader is encouraged to report and technical or editing issues/problems with this specification to: https://www.omg.org/report_issue.htm

1 Scope

The domain of naval Combat Management Systems is characterized by a huge variety of underlying computing platforms, with different and often incompatible means of managing and reporting alerts. Standards-based alert management services are essential for interoperable and open systems. This specification is a standard for ALert MAnagement Service (ALMAS) in CMS systems, consisting of a standard alerts data model and a model for an alert delivery and lifecycle management service.

2 Conformance

This specification provides a level of conformance for a minimalist, basic ALMAS system; a fully functional ALMAS system; plus, additional levels for specialized extensions. In addition, conformant ALMAS implementation must conform to one or more of the middleware platform specific models presented in Chapters 8, 9, 10 and 11 of this document in addition to conforming to the XML Alert template data model and the XML initialization PSMs as presented in sections 7.1to 7.3 of this document.

There are three distinct roles in the ALMAS abstracted by the interface classes defined in the ALMAS Management package: Producer, Manager and Receiver. Conformance recognises that a conforming application will be performing only one of these roles and therefore only a subset of the interface will be applicable. Accordingly, conformance is defined in terms of the PIM interface methods that are applicable to each of these roles.

Level	Producer	Manager	Receiver
1	Invokes: RaiseAlertFromData CancelAlert	Implements: RaiseAlertFromData CancelAlert Invokes: AlertDataNotification	Implements: AlertDataNotification
2	Implements: ALMASNotificationListener interface Invokes: ALMASConfiguration interface and any of the ALMASProducer and ALMASManager interface methods	Implements: ALMASConfiguration, ALMASManager, ALMASResponder and ALMASProducer interfaces Invokes: ALMASNotificationListener and ALMASReceiver interfaces	Implements: ALMASReceiver interface Invokes: Any of the ALMASResponder and ALMASManager interface methods
3A	N/A	Level 2 plus Implements: ALMASResponderExtensions interface	Level 2 plus Invokes: ALMASResponderExtensions interface
3B	Level 2 plus	Level 2 plus	N/A

	Invokes: RemoveAlertsWithDynamicMessageData	Implements: RemoveAlertsWithDynamicMessageData	
3C	Level 2 plus Invokes: AttachCategorizationRule & DetachCategorizationRule	Level 2 plus Implements: AttachCategorizationRule & DetachCategorizationRule	N/A

Level 3A groups together extensions for mulit-language support; 3B is an extension for a more complex alert lifecycle from a producer's perspective; and 3C is an extension for raising alerts indirectly through categorization rules.

Alternatively, configuration may be performed centrally by a system function, in which case applications with the Producer role do not invoke the ALMASConfiguration interface.

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.

Table 3-1: Normative References

Title (Acronym)	Version / Date	Organization	Reference / URL
Data Distribution Service (DDS)	1.4 / March 2015	OMG	formal/2015-04-10 https://www.omg.org/spec/DDS/1.4/PDF
Interface Definition Language (IDL)	4.2 / January 2018	OMG	formal/2018-01-05 https://www.omg.org/spec/IDL/4.2/PDF
Graph Query Language (GraphQL)	June 2018	Facebook	https://spec.graphql.org/June2018/
OASIS Common Alerting Protocol (OASIS)	1.0 / 2004	OASIS	https://www.oasis-open.org/committees/download.php/6334/oasis-200402-cap-core-1.0.pdf
Microsoft Component Object Model (COM)	2009	Microsoft	https://www.docs.microsoft.com/en-us/windows/win32/com/component-object-model--com--portal
Common Object Request Broker Architecture (CORBA)	3.0.3 / March 2004	OMG	https://www.omg.org/spec/CORBA
Extensible Markup Language (XML)	1.0 / 2008	W3C	https://www.w3.org/TR/xml/

Open Telemetry (OTEL)	1.38.0	CNCF	https://opentelemetry.io/docs/specs/otel/

4 Terms and definitions

4.1 General Definitions

Architecture Board (AB) - The OMG plenary that is responsible for ensuring the technical merit and MDA-compliance of RFPs and their submissions.

Board of Directors (BoD) - The OMG body that is responsible for adopting technology.

Component Object Model (COM) – A platform-independent, distributed, object-oriented system for creating binary software components that can interact.

Common Object Request Broker Architecture (CORBA) - An OMG distributed computing platform specification that is independent of implementation languages.

Common Warehouse Metamodel (CWM) - An OMG specification for data repository integration.

CORBA Component Model (CCM) - An OMG specification for an implementation language independent distributed component model.

Data-Centric Publish-Subscribe (DCPS) – The DDS specification describing the application interfaces and communication semantics for distributed application communication and integration.

Data Distribution Service (DDS) – a middleware protocol and API standard for data-centric connectivity.

Data Local Reconstruction Layer (DLRL) – describes a high-level interface to DDS that allows a simple integration of the DDS Service into the application layer.

Graph Query Language (GraphQL) – a query language for Application Programmable Interfaces and a runtime for fulfilling those queries with your existing data.

Interface Definition Language (IDL) - An OMG and ISO standard language for specifying interfaces and associated data structures.

Internet Assigned Numbers Authority (IANA) – a standards organization that oversees global Internet Protocol (IP) address allocation and other internet Protocol-related symbols and numbers.

Letter of Intent (LOI) - A letter submitted to the OMG BoD's Business Committee signed by an officer of an organization signifying its intent to respond to the RFP and confirming the organization's willingness to comply with OMG's terms and conditions, and commercial availability requirements.

Mapping - Specification of a mechanism for transforming the elements of a model conforming to a particular metamodel into elements of another model that conforms to another (possibly the same) metamodel.

Metadata - Data that represents models. For example, a UML model; a CORBA object model expressed in IDL; and a relational database schema expressed using CWM.

Metamodel - A model of models.

Meta Object Facility (MOF) - An OMG standard, closely related to UML, that enables metadata management and language definition.

Model - A formal specification of the function, structure and/or behavior of an application or system.

Model Driven Architecture (MDA) - An approach to IT system specification that separates the specification of functionality from the specification of the implementation of that functionality on a specific technology platform.

Normative – Provisions that one must conform to in order to claim compliance with the standard. (as opposed to non-normative or informative which is explanatory material that is included in order to assist in understanding the standard and does not contain any provisions that must be conformed to in order to claim compliance).

Normative Reference – References that contain provisions that one must conform to in order to claim compliance with the standard that contains said normative reference.

Platform - A set of subsystems/technologies that provide a coherent set of functionality through interfaces and specified usage patterns that any subsystem that depends on the platform can use without concern for the details of how the functionality provided by the platform is implemented.

Platform Independent Model (PIM) - A model of a subsystem that contains no information specific to the platform, or the technology that is used to realize it.

Platform Specific Model (PSM) - A model of a subsystem that includes information about the specific technology that is used in the realization of it on a specific platform, and hence possibly contains elements that are specific to the platform.

Quality of Service (QOS) – in the context of DDS, a rich set of characteristics that define the behavior of the DDS systems (such as reliability, liveliness, durability, etc.)

Request for Information (RFI) - A general request to industry, academia, and any other interested parties to submit information about a particular technology area to one of the OMG's Technology Committee subgroups.

Request for Proposal (RFP) - A document requesting OMG members to submit proposals to the OMG's Technology Committee. Such proposals must be received by a certain deadline and are evaluated by the issuing task force.

Task Force (TF) - The OMG Technology Committee subgroup responsible for issuing an RFP and evaluating submission(s).

Technology Committee (TC) - The body responsible for recommending technologies for adoption to the BoD. There are two TCs in OMG – Platform TC (PTC), that focuses on IT and modeling infrastructure related standards; and Domain TC (DTC), that focus on domain specific standards.

Unified Modeling Language (UML) - An OMG standard language for specifying the structure and behavior of systems. The standard defines an abstract syntax and a graphical concrete syntax.

UML Profile - A standardized set of extensions and constraints that tailors UML to particular use.

XML Metadata Interchange (XMI) - An OMG standard that facilitates interchange of models via XML documents.

4.2 Definitions Specific to this Document

The RFP prompting this response defined the following set of standard terminology which will henceforth be used within this document:

- An ***event*** is an occurrence that has been detected by the system whose happening must be reported to other members of the system, including human operators.
- An ***alert*** is an entity of observation regarding an event (or sequence of related events) to be reported (directly or indirectly) to an appropriate set of actors.
- ***Alert clients*** are the entities within the system that raise, modify, receive, process, or handle alerts generated by ALMAS.
- An ***alert template*** is a generic definition of a type of alert which can be raised, e.g., 'collision warning' – it requires instantiation to create an alert.
- An ***instance*** of an alert is a specifically raised alert e.g., 'collision warning with track number 111, bearing 020, range 2nm'

In addition to the general terms defined above, the RFP indicates that there is an expectation that the ALMAS standard will include three main alert categories, as follows:

- Alerts which require no actor action or acknowledgement. This collection of alert templates are generally ***informative*** or routine alerts, they are usually of lower priority / urgency and require some action by ALMAS to be removed.
- Alerts which require acknowledgement by actor(s). This collection of ***acknowledgement*** alert templates is usually more urgent alerts where at least one actor must indicate acknowledgement to ALMAS that the alert has been received.
- Alerts which require both acknowledgement and action confirmation by actor(s). This collection of ***action*** alert templates is frequently used for important or critical events where not only is acknowledgement of the receipt required, but also confirmation that the required action has been taken[1].

5 Acronyms and Abbreviations

CMS	(Naval) Combat Management System
CORBA	Common Object Request Broker Architecture
DCOM	Distributed Component Object Model
HTTP	HyperText Transfer Protocol
OMG	Object Management Group
RFP	Request For Proposal
UML	Unified Modelling Language
XML	eXtensible Mark-up Language

[1] Definition of the required action is not within the scope of ALMAS.

6 Platform Independent Model (PIM)

The PIM has been split into three packages as follows:

- ALMAS Client Callbacks: The interface to be implemented by system components that wish to be notified of ALMAS events such as alerts created, deleted, etc.
- ALMAS Data Model: The structures and their relationships used in an ALMAS system.
- ALMAS Management: Components of the ALMAS system responsible for setting up ALMAS and alert lifecycle oversight.

These are described below, note that ALMAS Categorization is an optional PIM for attaching event-based categorization rules to alerts defined in the core parts of this specification.
Section 6.5 describes ALMAS dynamic behavior: alert state transition as well as the interactions between ALMAS Receivers, Producers, and the ALMAS System itself.

6.1 ALMAS Client Callbacks

ALMAS Client Callbacks are the interfaces to be implemented by system components that wish to be notified of ALMAS events such as alerts created, deleted, etc. There are two classes in this package. In order to be plugged into the ALMAS system, a client must implement one of these interfaces, and register with the Alert Manager.

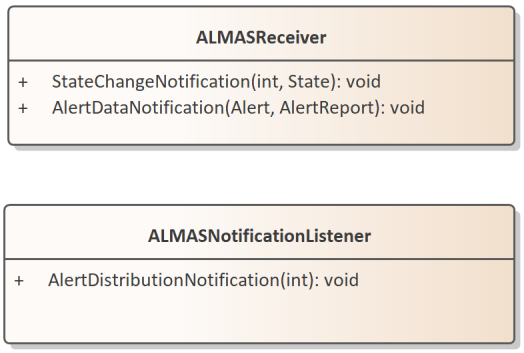

Figure 6-1: PIM class diagram for ALMAS Clients

6.1.1 ALMASNotificationListener

Class provided by registering notification listeners for receipt of alert distribution notifications.

6.1.1.1 Operation

Name	Type	Summary
AlertDistributionNotification (int)	public void [Parameters] AlertID: int	This is called as soon as an alter requiring confirmation has been received by the ALMAS system. This callback is generated when the alert is in the Raised state. (Note: StateChangeNotification callbacks are also generated for this event.) The onward distribution is notified through additional StateChangeNotification callbacks.

6.1.2 ALMASReceiver

Class provided by registering alert receivers for provision of the notification callbacks. Only clients that implement this interface and register as receivers can access active alert data. Clients can only register if they are built against the

ALMAS interface; therefore, no runtime security control is required in this context. Note: The ALMASResponder interface is used to notify ALMAS of "progress" in satisfying the received alert.

6.1.2.1 Operation

Name	Type	Summary
StateChangeNotification (int, Enumeration)	public void [Parameters] AlertID: int, NewState: Enumeration	Indicates a change of state of an alert to a receiver who has registered for this alert's state change notifications. These states are the same states as used in CurrentState for an Alert. This callback is generated for an alert's initial state and all subsequent state transitions. it is not generated when an alert instance is deleted. (Note: instances can only be deleted by the ALMAS system from a non-current state).
AlertDataNotification (Alert, AlertReport)	public void [Parameters] AlertInfo: Alert, Report: AlertReport	Provides notifications of new and modified alert data.

6.2 ALMAS Data Model

The classes described in this section provide the definition of the contents of Alerts, Alert Templates, and Receivers for ALMAS. The two primary concepts in this data model are of an Alert Template and an Alert. The Alert Template describes the static description of a pre-defined class of alerts, while an Alert contains the specific attributes of a "live" Alert within the ALMAS system. Both utilize the AlertData class to describe many of their field attributes and values.

Note that the constraint called 'alert_data' in the figure below is defined as follows:

"context a: Alert inv: if ((a.alert_data.Category = Information) or (a.alert_data.Category = Warning))) then
(a.CurrentState <> Handled)"

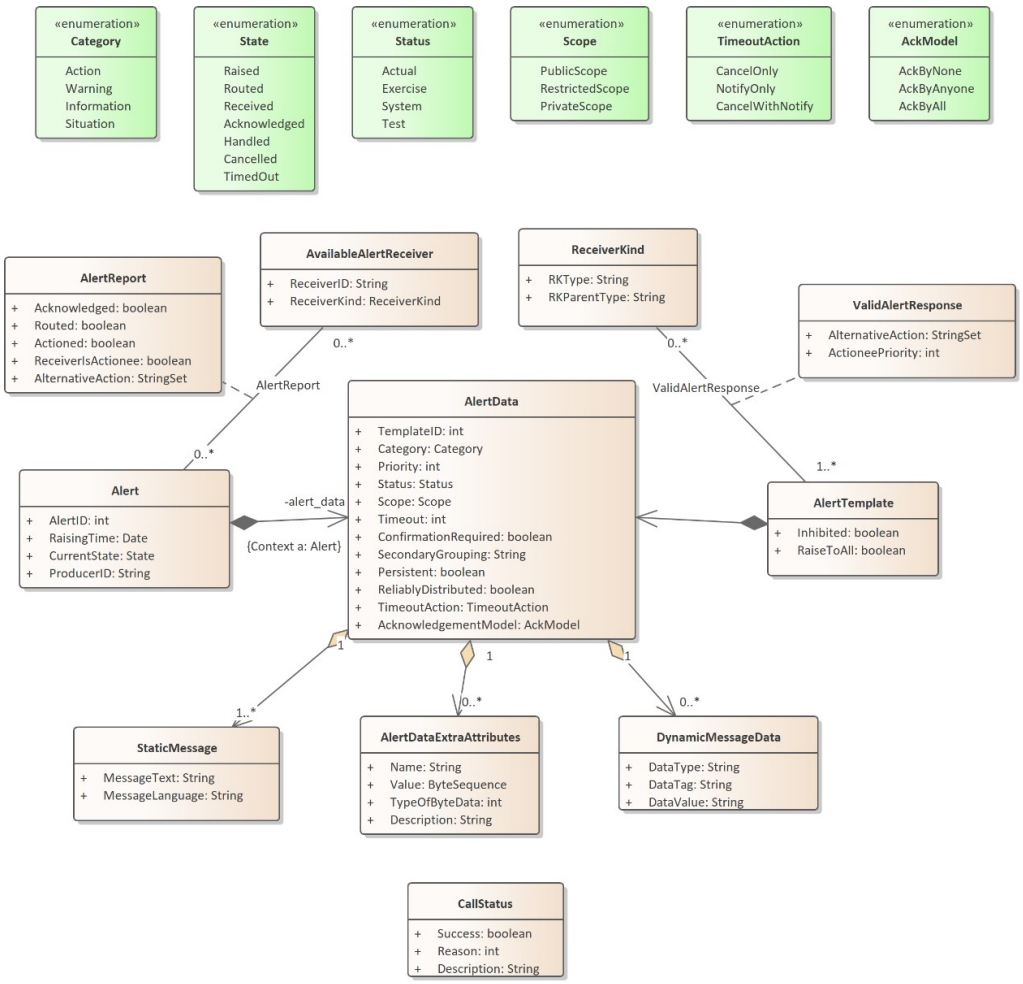

Figure 6-2: PIM class diagram for ALMAS Data Model

6.2.1 Alert

An active alert within ALMAS. The Alert class provides the main entity that ALMAS uses for tracking the state of an alert. The specific data such as message and other attributes for an active alert is provided in the AlertData class which is a member attribute of the Alert.

6.2.1.1 Attribute

Name	Type	Summary
AlertID	public int	The instance id for the specific instance of the alert.
RaisingTime	public Date	The time at which the alert was raised.
CurrentState	public State	Holds the current state of the alert, valid states are determined by the category of the alert, {Raised, Routed, Received, Acknowledged, Handled, Cancelled, Timed_Out}. Note that Handled is not a valid state for Information and Warning Alerts.
ProducerID	public String	The producer freetext ID - corresponds to CAP source

6.2.2 AlertData

This represents the set of data shared between the alert template and alert classes. All fields have default values which can be changed when alerts are raised/updated. This may be set up through the use of templates as specified through the XML PSM, which initialises AlertTemplate and its associated classes.

6.2.2.1 Attribute

Name	Type	Summary
TemplateID	public int	A unique identifier for template which owns this alert data (or that was used to create the alert if this is referenced from Alert). Valid range from 1 upwards.
Category	public Enumeration	This enumeration can take the value Action / Warning / Information / Situation
Priority	public int	Alert priority as an integer value in the range 1-99. The priority is open for client use and not intended for interpretation by ALMAS.
Status	public Status	Corresponds to the OASIS CAP Status field.
Scope	public Scope	Corresponds to CAP scope.
Timeout	public int	Specifies the time, in seconds, required to elapse before the alert will timeout and perform its default timeout action. 0 implies there is no timeout.
ConfirmationRequired	public boolean	This is set if confirmation of receipt is required, that it has been distributed. If this is set to true, the producer has registered for receipt of the distribution notification.
SecondaryGrouping	public String	This is an additional field to support client specific filtering mechanisms.
Persistent	public boolean	Indicates whether the alert data is required to be persistent in the event of a system restart
ReliablyDistributed	public boolean	A flag which, when true, indicates that the alert should have guaranteed delivery.
TimeoutAction	public TimeoutAction	When the alert times-out, ALMAS acts according to this attribute.
AcknowledgementModel	public AckModel	Sets the conditions upon which the alert state can transition to 'acknowledged'.This has the options of {none, anyone, all}

6.2.3 AlertDataExtraAttributes

This is a class representing items of alert data that are specific to particular clients, that require supporting in order to fulfil possible requirements of an alert management system (such as images, screen locations or other binary data), but are not general enough to be defined explicitly as data types in an ALMAS. Effectively ALMAS provides blind delivery of the information provided by this class to the alert receiver without any knowledge as to its intended meaning and behaviour. The extra attributes are configured via the ALMAS Alert definition xml PSM specified in section 7.1. If defined in the Alert definition XML provided to ALMAS, then ALMAS shall support the definition, receipt, storage and passing of this data to receivers as part of a standard implementation.

6.2.3.1 Attribute

Name	Type	Summary
Name	public String	Name of the client specific attribute
Value	public ByteSequence	Contents as a byte sequence. (Note: strings are not null-terminated).
TypeOfByteData	public int	Valid values for this are: 0 = string (UTF-8) 1 = Integer8 2 = Integer16 3 = Integer32 4 = Float32 5 = Float64 6 = bytes (private format) 7 = bytes (defined by media type)
Description	public String	This field is used to provide an indication of the content e.g., 'image (jpg), URL, track object ID, … When the TypeOfByteData is 7, this is set to the media type / subtype tree as defined by IANA.

6.2.4 AlertReport

This provides the delivery message of an Alert to an ALMASReceiver. It contains the Alert and the current status information. This will contain details of whether the instance has been acknowledged by this receiver etc. and will also be completed with respect to any dynamic message data.

6.2.4.1 Attribute

Name	Type	Summary
Acknowledged	public boolean	Identified whether the alert has been acknowledged by this receiver.
Routed	public boolean	Identified whether the alert can be confirmed to have been routed as per the 'routed' alert substate.
Actioned	public boolean	Identified whether the alert has been actioned by this receiver.
ReceiverIsActionee	public boolean	Indicates that this receiver is the chosen actionee for this alert.
AlternativeAction	public StringSet	Provides means by which an alternative action outside of the scope of ALMAS can be distributed with the alert via ALMAS.

6.2.5 AlertTemplate

An AlertTemplate specifys the generic characteristics of a specific alert type "at rest" (e.g., the general characteristics of a collision warning alert). This includes the category of alert, such as Action etc. An AlertTemplate uses an associated AlertData object to specify the contents of the template. An AlertTemplate can be used to specify the properties of commonly used within a system. At the time of raising an Alert from a template, the user/system provides the relevant instance data of that alert. It is an error to specify RaiseToAll and to define either ReceiverKind instances or a Secondary Grouping in the AlertData instance.

6.2.5.1 Attribute

Name	Type	Summary
Inhibited	public boolean	The inhibition status of that alert type. If this is 'true' then attempts to raise an alert of that type will fail.
RaiseToAll	public boolean	Indicates that the alert should be raised to all available receivers rather than specified ones.

6.2.6 AvailableAlertReceiver

The class used to identify a receiver of alerts. A registered receiver of alerts. The AvailableAlertReceiver is registered with ALMAS through the ALMASResponder API. The AvailableAlertReceiver is directly associated with an ALMASReceiver through the ReceiverID attribute, which is provided at registration time to ALMAS using the RegisterReceiver method.

6.2.6.1 Attribute

Name	Type	Summary
ReceiverID	public String	Unique identifier for the receiver.
ReceiverKind	public ReceiverKind	The kind of the receiver as an explicit attribute link to the Receiver Kind class.

6.2.7 CallStatus

This is the ALMAS a general-purpose success/failure descriptor class used throughout ALMAS. If Success, then the other parameters are not applicable.

6.2.7.1 Attribute

Name	Type	Summary
Success	public boolean	Flag indicating pass/fail status
Reason	public int	Enumerated reason correlating to the “Call Status” 0 = Success 1 = Not Accepted 2 = Malformed Alert 3 = Timeout/delivery 4 = Requested Service Unavailable 5+ = Other
Description	public String	Additional String data further describing status

6.2.8 DynamicMessageData

Since Alerts often have variable data fields, the DynamicMessageData class provides the means for inserting variable content into the Alert's MessageText during runtime. Replacement values for the DataTag are treated as strict string substitution within the MessageText of the StaticMessage associated with the Alert. This is used to capture the triplet of data tag type, tag position in the alert message and the value that this tag in the template message text should be replaced with. Note: if the text specified in the StaticMessage contains multiple replacement points then an equal number of DynamicMessageData objects are required for full substitution. It is an error to specify StaticMessage and DynamicDataMessage instance collections with different sets of substitution tags for the same alert template. To substitute language locale specific dynamic data, define and supply distinct language locale specific tags when raising alerts. (I.e., for each language and placeholder combination, supply a DynamicMessageData instance with a unique DataTag to match a placeholder in exactly one StaticMessage instance).

6.2.8.1 Attribute

Name	Type	Summary
DataType	public String	The type of related object e.g., freetext, track, vehicle, position, etc.
DataTag	public String	This identifies the insertion point for the related object in the MessageText associated with the Alert. Tags are alphanumeric so to match StaticMessage text "xxxxx %number% yyyyyyy zzzz", a DataTag with the value "number" is required. It is a case sensitive, alphanumeric string.
DataValue	public String	The value of the object instantiation. Given a type of string to be general enough to support free text and track/vehicle id's alike.

6.2.9 ReceiverKind

The descriptor of an alert receiver. This could for example be an operator role. ReceiverKind objects are used in many places in ALMAS including the specification of what operators/clients will receive which Alerts.

- These are used to show all possible receivers of an Alert, when used in an AlertTemplate.
- These are used during runtime to identify the actual receivers for an active alert.

6.2.9.1 Attribute

Name	Type	Summary
RKType	public String	String identifier of the kind of receiver, for example the role of a receiving operator.
RKParentType	public String	The hierarchical parent receiver kind name that this one "belongs to". This is used by ALMAS to resolve cases where a specific RK is not available, but handing is required by an appropriate receiver. Note that a lack of a Parent is indicated by an empty string.

6.2.10 StaticMessage

Provides the default message text for an alert as a tuplet of the actual static text and the language in which the text is provided. An AlertData object has a StaticMessage instance for each language supported by a particular ALMAS. If the StaticMessage requires runtime updating, then use data tags as specified in DynamicMessageData. To support the runtime substitution of different text for different languages, data tags must have different values in each of the languages to define the substitution uniquely.

6.2.10.1 Attribute

Name	Type	Summary
MessageText	Public String	This is a text string, which in an Alert or AlertTemplate is only partially completed. With the MessageText being "xxxxx %number% yyyyyyy zzzz" in an Alert or AlertTemplate, and with a DynamicMessageData with DataTag having the value “number” and DataValue having the value "123" then the resulting MessageText in response to GetFilledMessageText will be "xxxxx 123 yyyyyyy zzzz". All substitution points are of the form "(start | non-alphanumeric%(tag)%(end | non-alphanumeric)", where start and end denote the start and end of the MessageText string respectively and tags are case sensitive, alphanumeric strings ("number" in the above) which should correspond to a DataTag in an associated DynamicMessageData.
MessageLanguage	public String	The message ‘Locale’

6.2.11 ValidAlertResponse

The ValidAlertResponse is the association class that specifies the list of actions that a particular ReceiverKind (e.g., “role”) can take in response to an Alert of an AlertTemplate type. It also specifies the priority for being chosen as the actionee of that ReceiverKind among all ReceiverKinds associated with that AlertTemplate.

The set of alternative action strings can be used by the system to provide a constraind set of “command-response” options to the client. For example, ValidAlertResponses for an “Engagement Request Alert” might include “WILCO”, “CANTCO”, etc.

6.2.11.1 Attribute

Name	Type	Summary
AlternativeAction	public StringSet	The 'names' of alternative actions available to the relevant actor.
ActioneePriority	public int	The priority of the ReceiverKind as actionee for a specifc alert as described by its template. The highest priority actionee for an action alert should be chosen as the current actionee for the alert. This will then flow into the ReceiverIsActionee field of the AlertReport.

6.2.12 Category

The categories of alerts in terms of the expectation placed on the operator receiving the alert; i.e., generically, why has the alert been received and what type of implicit or explicit response is expected.

6.2.12.1Attribute

Name	Summary
Action	An explicit input to the system is expected as a result of receiving the alert. The alert persists until it is

	cancelled due to the condition to which it relates no longer being present (due either to explicit operator action relating to the alert or action external to the ALMAS system).
Warning	The receiver may decide to take an explicit action in mitigation to the condition to which the warning relates. The alert does not persist according to the underlying condition that the alert warns about.
Information	The receiver is expected to take account of this information in subsequent decisions. The alert does not persist according to the underlying condition that the alert informs about.
Situation	The receiver is expected to take account of the new state of the situation in subsequent decisions. The alert persists until it is cancelled due to the condition to which it relates no longer being present (due either to explicit operator action relating to the alert or action external to the ALMAS system).

6.2.13 State

The states between which an alert transitions in its lifetime.

6.2.13.1 Attribute

Name	Summary
Raised	The alert has been created by the alert producer.
Routed	The alert has been routed to the receivers, but reception has not been confirmed by sufficient receivers to enter the received state.
Received	The alert has been received by sufficient receivers.
Acknowledged	All necessary acknowledgements have been made.
Handled	The alert ends its lifetime through being handled.
Cancelled	The alert ends its lifetime through being cancelled by the producer.
TimedOut	The alert ends its lifetime through being timed-out.

6.2.14 Status

The status of the entities with regards to the mode of use of ALMAS in comparison to the mode of use of receivers and producers.

6.2.14.1 Attribute

Name	Summary
Actual	Actionable by all targeted recipients.
Exercise	Actionable only by designated exercise participants.
System	For entities that support alert network internal functions.
Test	Technical testing only, all recipients disregard

6.2.15 Scope

This class models the scope of the alert's dissemination.

6.2.15.1 Attribute

Name	Summary
PublicScope	unrestricted dissemination
RestrictedScope	dissemination restricted to known functions
PrivateScope	dissemination restricted to specified addresses

6.2.16 TimeoutAction

This class models the possible behaviors when an alert is timed-out.

6.2.16.1 Attribute

Name	Summary
CancelOnly	The alert is just cancelled (the alert instance's lifetime ends).
NotifyOnly	The alert manager is notified.
CancelWithNotify	The alert is cancelled (the alert instance's lifetime ends) and the alert manager is notified.

6.2.17 AckModel

This class models the conditions upon which an alert state can transition to 'acknowledged'.

6.2.17.1 Attribute

Name	Summary
AckByNone	No acknowledgement is required.
AckByAnyone	Any single acknowledgement is sufficient.
AckByAll	The alert must be acknowledged by all recipients.

6.3 ALMAS Management

This section describes the classes responsible for raising, routing, maintaining the state of, and destroying alerts through their lifecycle. ALMAS uses a collection of specialized component interfaces for maintaining state, data, and lifecycle of Alerts. In general, systems that utilize ALMAS will interact during runtime primarily through the ALMAS Producer, Responder, and Notification Listener classes. The ALMAS Manager interface is utilized more at system startup.

Deleting alert instances is under the control of ALMAS itself as part of its lifecycle management, and not at the request of its users. In more detail:

- Any alert is removed when cancelled. Note that Situation alerts are only removed when cancelled.
- Information and Warning alerts are removed when the required number of acknowledgements (as identified in the AlertData AcknowledgementModel attribute) are given or (if a timeout is defined) when the timeout is expired.
- Action alerts are removed when HandleAlert is called by the Receiver identified as the Actionee in its AlertReport.

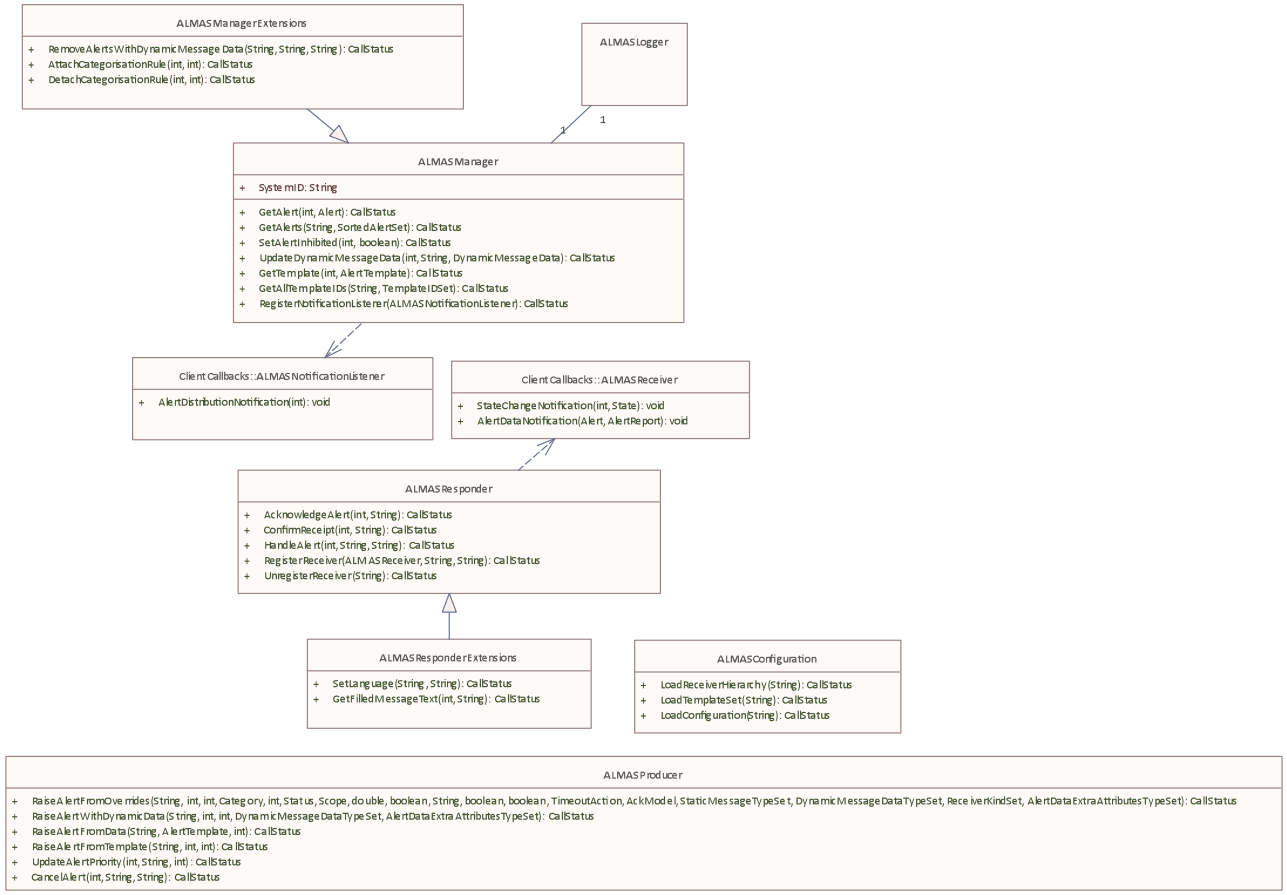

Figure 6-3: PIM class diagram for ALMAS Management

This package provides the main API to the ALMAS service.

6.3.1 ALMASConfiguration

Provides an API by which systems can configure ALMAS to behave in a more tailored manner in order to satisfy very specific requirements. There are three categories of configuration file that can be used by ALMAS: the receiver hierarchy, templates, and configuration information. The string filename is expected to resolve to either a local file accessible to ALMAS, or a URL accessible to ALMAS. The returned CallStatus object from each of the methods provides an indication of success/failure and any additional relevant rationale describing that status. The effect of invoking the ALMASConfiguration interface does not persist beyond the lifetime of the application. Clients must invoke the operations on the interface for each execution lifetime of the Manage application.

6.3.1.1 Operation

Name	Type	Summary
LoadReceiverHierarchy (String)	public CallStatus [Parameters] Filename: String	Loads the receiver hierarchy as provided by the client via xml conforming to the relevant xml schema document. The specification of the ReceiverHierarchy file format can be found in section 7.3. If invoked multiple times for an application lifetime, the semantics are additive but disjoint; it is an error to define a receiver in a hierarchy file that has already been defined.
LoadTemplateSet (String)	public CallStatus [Parameters] Filename: String	Loads a template set into the ALMAS database. Multiple calls to this method result in the union of the new templates with the existing templates in ALMAS. It is an error to refer to ReceiverKind instances in the template file that have not been previously defined in a loaded hierarchy file. It is also an error to duplicate an existing template id or to mismatch tags between static messages and dynamic data. These are permitted implementation-specific error conditions: to use an unsupported data type for dynamic message data, to use an out of range value for type of byte data, to omit a static message for a particular language or to exceed capacity limits. The specification of the template file format can be found in section 7.1.
LoadConfiguration (String)	public CallStatus [Parameters] Filename: String	Loads the ALMAS configuration file as provided by the client. The specification of the configuration file format can be found in section 7.2.

6.3.2 ALMASLogger

The ALMASLogger interface provides a logging mechanism to record Alert information created by the system. All API methods are logged by conformant implementations; the mechanism to do so is defined by each of the PSM sections later in this document

6.3.3 ALMASManager

The ALMASManager interface provides the minimal set of APIs necessary to track ALMAS activity. Additionally, the ALMASManager provides the interface in ALMAS for retrieving Alerts and AlertTemplates and registering for the notification of delivery of Alerts. Note that the registration of receivers is done via the ALMAS Responder class.

Note: The methods found in the ALMASProducer interface allow the system to update the status or attributes of an alert during runtime. The ALMAS Manager resolves dynamic message data and each recipient's language selection that each Alert instance contains exactly one static message and no dynamic message data. A component implementing the ALMASManager, ALMASProducer and ALMASResponder interfaces is able to flexibly coordinate the distribution of alerts and responses between producers and responders.

6.3.3.1 Attribute

Name	Type	Summary

SystemID	private String	Provides a field for specifying the current instance of ALMAS. Corresponds to CAP sender

6.3.3.2 Operation

Name	Type	Summary
GetAlert (int, Alert)	public CallStatus [Parameters] AlertID: int, out Alert: Alert	Retrieves data for a specific raised alert from ALMAS given the passed AlertID. Assumes the requestor knows the AlertID to retrieve. This operation retrieves the current data for an alert that is already known to the client.
GetAlerts (String, SortedAlertSet)	public CallStatus [Parameters] Filter: String, out AlertSet: SortedAlertSet	Retrieves a set of all alert instances within ALMAS that satisfy the filter. The filter string provided will be compared with the value in the AlertData SecondaryGrouping field. All matches will be returned in the Set.
SetAlertInhibited (int, boolean)	public CallStatus [Parameters] TemplateID: int, Inhibition: boolean	Sets the inhibition status of a specific alert template to suppress or allow the raising of all alerts of that template. Whilst set to inhibited, the ALMAS Manager fails attempts to raise an alert using that template.
UpdateDynamicMessageData (int, String, DynamicMessageData)	public CallStatus [Parameters] AlertID: int, ObjectValue: String, OldData: DynamicMessageData	Indicates a change to the value of a related object for the provided alert ID. OldData is necessary in order to clearly indicate which dynamic message data should be changed.
GetTemplate (int, AlertTemplate)	public CallStatus [Parameters] TemplateID: int, out Template: AlertTemplate	Retrieves an existing alert template from ALMAS by providing the template ID.
GetAllTemplateIDs (String, TemplateIDSet)	public CallStatus [Parameters] Filter: String, out TemplateIDs: TemplateIDSet	Retrieves all Alert Template IDs, or if the Filter string is non-null, it returns those which satisfy the Filter. The filter string provided will be compared with the value in the AlertData SecondaryGrouping field. All matches will be returned in the Set.
RegisterNotificationListener (ALMASNotificationListener)	public CallStatus [Parameters] Handle: ALMASNotificationListener	Registers a new Notification Listener for receipt of the alert distribution notifications.

6.3.4 ALMASManagerExtensions

This class contains optional extensions to the alert manager functionality. These extensions may or may not be implemented in simple ALMAS implementations.

6.3.4.1 Operation

Name	Type	Summary
RemoveAlertsWithDynamicMessageData (String, String)	public CallStatus [Parameters] CancellerID: String,DataType: String,DataValue: String	Indicates to ALMAS that a specific object has been removed from the system, and therefore all associated alerts are no longer valid. These will then be deleted from ALMAS.
AttachCategorisationRule (int, int)	public CallStatus [Parameters] TemplateID: int, RuleID: int	Associates an event categorisation rule with an AlertTemplate
DetachCategorisationRule (int, int)	public CallStatus [Parameters] TemplateID: int, RuleID: int	Disassociates an event categorisation rule from an AlertTemplate

6.3.5 ALMAS Producer

Provides the API by which system components producing alerts can create and update alerts that are generated. A CallStatus object will be returned to indicate whether the request has been accepted by ALMAS. If a system wished to track the lifecycle of the alert, they must implement the NotificationListener functionality to receive updates.

Four mechanisms by which alerts can be raised are provided by the ALMASProducer interface class. Three variants RaiseAlertFromTemplate, RaiseAlertWithDynamicData and RaiseAlertFromOverrides allow the system to raise an alert by simply specifying the alert ID, template ID and their own ProducerID; with dynamic data allows the specification of the intentionally variable data to supplement the template alert definition; from overrides also allows the over-ride of any placeholders that may be present in the ‘Message’ attribute of the alert data class associated with that template.

The raiser may also optionally override any of the following parameters: Message, MessageLanguage, Category, Status, Scope, Timeout, ConfirmationRequired, AlertReceiverSet, Priority, TimeoutAction and AcknowledgementModel.

The RaiseAlertFromData method allows the raiser to specify a completely new alert with no basis on any existing templates. Systems using ALMAS may not wish to support alert templates depending on their size, complexity, and level of alert usage, in which case that system can always use RaiseAlertFromData without need to instantiate any templates at any point during operation.

The status or attributes of an alert can be updated during runtime by calling the UpdateAlert method found in the ALMASProducer interface. The ALMASProducer then works with the ALMAS system to ensure state and data is properly maintained in the system.

6.3.5.1 Operation

Name	Type	Summary
RaiseAlertFromOverrides (String, int, int, Category, int, Status, Scope, double, boolean, String, boolean, boolean, TimeoutAction, AckModel, StaticMessageTypeSet, DynamicMessageDataTypeSet, ReceiverKindTypeSet, AlertDataExtraAttributesTypeSet)	public CallStatus [Parameters] ProducerID: String, TemplateID: int, out AlertID: int, Category: Category, Priority: int, Status: Status, Scope: Scope, Timeout: double, ConfirmationRequired: boolean, SecondaryGrouping: String, Persistent: boolean, ReliablyDistributed: boolean, TimeoutAction: TimeoutAction, AcknowledgementModel: AckModel, StaticMessages: StaticMessageTypeSet, DynamicMessages: DynamicMessageDataTypeSet, AlertReceivers: ReceiverKindTypeSet, ExtraAttributes: AlertDataExtraAttributesTypeSet	This will cause an alert based on a known alert template to be created and raised. ProducerID, TemplateID and the out parameter AlertID are mandatory, all other parameters are optional. Return parameter indicates success or failure reason. The operation fails if the template is inhibited, or the hierarchy does not define the receiver kinds or static and dynamic tags are mismatched. The following are permitted implementation-specific failure cases: unsupported data type for dynamic message data, omitted language in static messages, capacity exceeded.
RaiseAlertWithDynamicData (String, int, int, DynamicMessageDataTypeSet, AlertDataExtraAttributesTypeSet)	public CallStatus [Parameters] ProducerID: String, TemplateID: int, out AlertID: int, DynamicMessages: DynamicMessageDataTypeSet, ExtraAttributes: AlertDataExtraAttributesTypeSet	This will cause an alert based on a known alert templateto be created and raised, whilst only specifying the dynamic data content that differs from the template definition. All parameters are mandatory. Return parameter indicates success or failure reason. The operation fails if the template is inhibited.
RaiseAlertFromData (String, AlertTemplate, int)	public CallStatus [Parameters] ProducerID: String, AlertInfo: AlertTemplate, out AlertID: int	Raise an alert not present in the ALMAS template database. A temporary AlertTemaplate is created (whose TemplateID is ignored), to facilitate the creation. Return parameter indicates success or failure reason. The operation fails if the hierarchy does not define the receiver kinds or static and dynamic tags are mismatched. The following are permitted implementation-specific failure cases: unsupported data type for dynamic message data, value for type of byte data out of range, omitted language in static messages, capacity exceeded.
RaiseAlertFromTemplate (String, int, int)	public CallStatus [Parameters] ProducerID: String, TemplateID: int, out AlertID: int	Raise an alert without any of the optional parameters for optimal use in the normal case. The operation fails if the template is inhibited.

Name	Type	Summary
UpdateAlertPriority (int, String, int)	public CallStatus [Parameters] AlertID: int, ProducerID: String, Priority: int	Updated the priority of existing alert instances that have previously been raised.
CancelAlert (int, String, String)	public CallStatus [Parameters] AlertID: int, CancellerID: String, CancellationReason: String	Cancel a specific alert within ALMAS Return parameter indicates success or failure reason.

6.3.6 ALMASResponder

Provides the API for systems to respond to and provide feedback to ALMAS about alerts received. Embedded in this class are the methods to register and un-register your system-specific receiver.

The system notifies ALMAS through this interface of significant events that have occurred to change the state of an alert.

6.3.6.1 Operation

Name	Type	Summary
AcknowledgeAlert (int, String)	public CallStatus [Parameters] AlertID: int, ReceiverID: String	Indication from an alert receiver that they have acknowledged receipt of the alert and no longer require distribution of its information.
ConfirmReceipt (int, String)	public CallStatus [Parameters] AlertID: int, ReceiverID: String	Confirmation by an alert receiver that they have successfully received the alert to ensure reliable distribution. The ReceiverID field enables action & situation alerts to transition when sufficient confirmations have been received. 'Sufficient' is the 'actionee' for action alerts, and anyone for situation alerts. It can also be used for logging purposes.
HandleAlert (int, String, String)	public CallStatus [Parameters] AlertID: int, ReceiverID: String, AlternativeAction: String	Indication from an Alert Receiver that they have performed the appropriate action required by an Action alert and that the alert can therefore be removed from ALMAS as no longer applicable. An Alternative Action has been performed if that parameter is non-null.
RegisterReceiver (ALMASReceiver, String, String)	public CallStatus [Parameters] ReceiverHandler: ALMASReceiver, ReceiverID: String, RKType: String	This registers a receiver with ALMAS, the parameters are ReceiverHandle (for callback); ReceiverID (for use in all other methods, including UnregisterReceiver) and RKType to provide link to RK hierarchy. It is an error to refer to an RKType that has not been previously defined in a loaded hierarchy file.
UnregisterReceiver (String)	public CallStatus [Parameters] ReceiverID: String	Removes a registered receiver from ALMAS, indicating that they are no longer available for receipt of alert data.

6.3.7 ALMASResponderExtensions

Optional extensions to the alert responder functionality.

6.3.7.1 Operation

Name	Type	Summary
SetLanguage (String, String)	public CallStatus [Parameters] ReceiverID: String, Language: String	Sets the language that this specific receiver should see their message text displayed in where appropriate. This method fails (Requested Service Unavailable) if there is no support for the language.
GetFilledMessageText (int, String)	public CallStatus [Parameters] AlertID: int, out MessageText: String	Returns the message text post related info substitutions. This is an optional helper function as the client could derive this itself.

6.4 Alert Categorisation

The Alert Categorisation PIM allows the expression of Event-Condition-Action rules which can guide automatic triggering of alerts. This represents an optional part of the specification, as it is also possible to trigger alerts through the ALMAS API. The Categorisation PIM allows for the implementation of monitoring components (agents) which can trigger alerts based on different events taking place in the system, such as time events or changes in the internal state of the system.

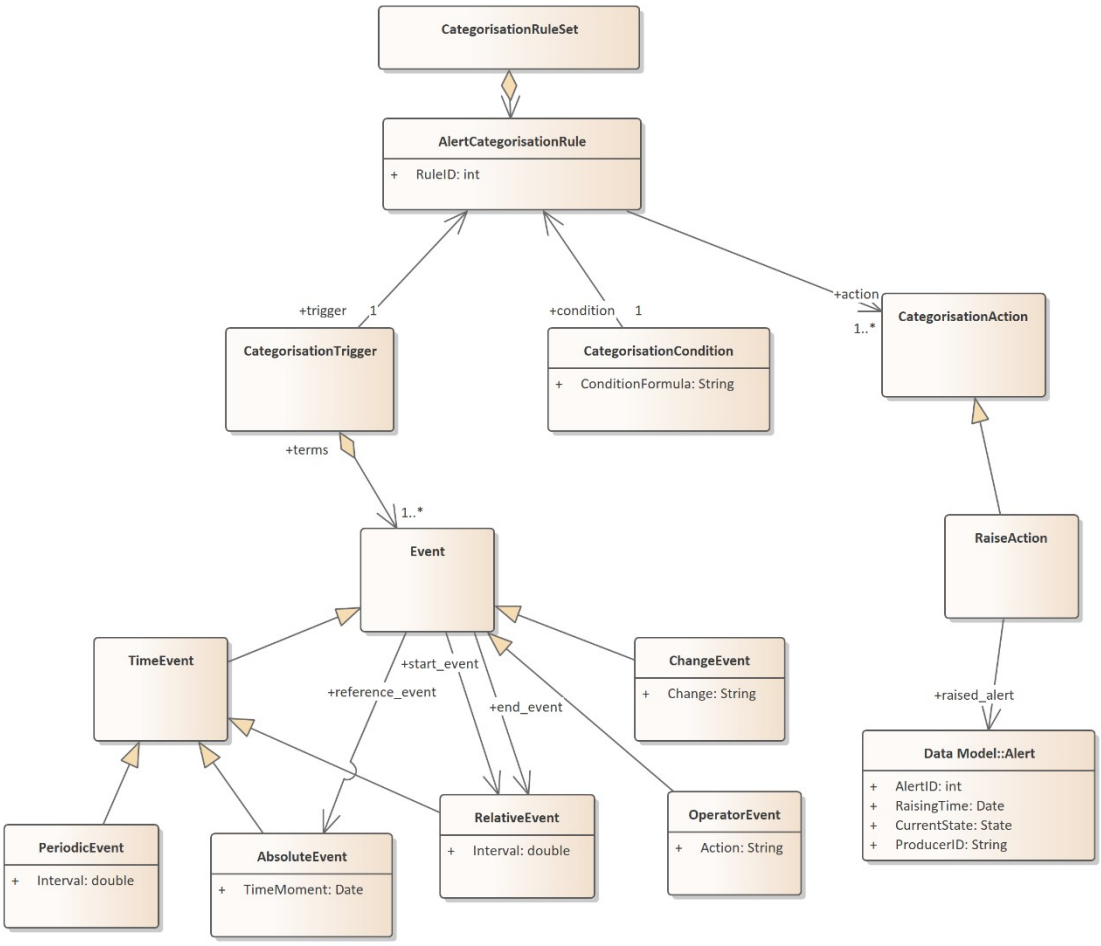

Figure 6-4: Alert Categorisation Platform Independent Model

6.4.1 AbsoluteEvent

Represents an event taking place once at a specific time moment.

6.4.1.1 Attribute

Name	Type	Summary
TimeMoment	public Date	The time of the trigger event

6.4.2 AlertCategorisationRule

Alert Categorisation Rule represents an Event-Condition-Action rule guiding the categorisation. On Event being triggered, a Condition is evaluated. If it evaluates to true, the corresponding Categorisation Action is executed.

6.4.2.1 Attribute

Name	Type	Summary
RuleID	public int	The rule identifier

6.4.3 CategorisationAction

Categorisation Action represents the action to be executed when an event has occurred, and the conditions required have been fulfilled.

6.4.4 CategorisationCondition

The Categorisation Condition represents the condition part of the Event, Condition Action rule.

6.4.4.1 Attribute

Name	Type	Summary
ConditionFormula	public String	The condition formula

6.4.5 CategorisationRuleSet

This is the set of Event, Condition Action rules which apply to this ALMAS system.

6.4.6 CategorisationTrigger

The Categorisation Trigger represents the Event which is able to be observed by ALMAS that can trigger categorisation.

6.4.7 ChangeEvent

One type of event such as enter/leave area, change of generic data value, etc.

6.4.7.1 Attribute

Name	Type	Summary
Change	public String	The change which is required

6.4.8 Event

General class of Event, used within the Categorisation Trigger.

6.4.9 OperatorEvent

Operator initiated events, for example operator changing a role.

6.4.9.1 Attribute

Name	Type	Summary
Action	public String	The operator action required

6.4.10 PeriodicEvent

Represents a relative event, i.e., an event taking place at a specific (time) interval after another event.

6.4.10.1 Attribute

Name	Type	Summary
Interval	public double	The condition formula

6.4.11 RaiseAction

A kind of Categorisation Action which raises an alert. Other categorisation actions could be added.

6.4.12 RelativeEvent

Represents a periodic event taking place between start_event and end_event at a specific periodicity (interval).

6.4.12.1 Attribute

Name	Type	Summary
Interval	public double	Time interval after the reference_interval event at which the RelativeEvent is to take place.

6.4.13 Time Event

A timeout event, which can be absolute, relative, or periodic.

6.5 Dynamic behaviour

6.5.1 Action Situation Alert State Model

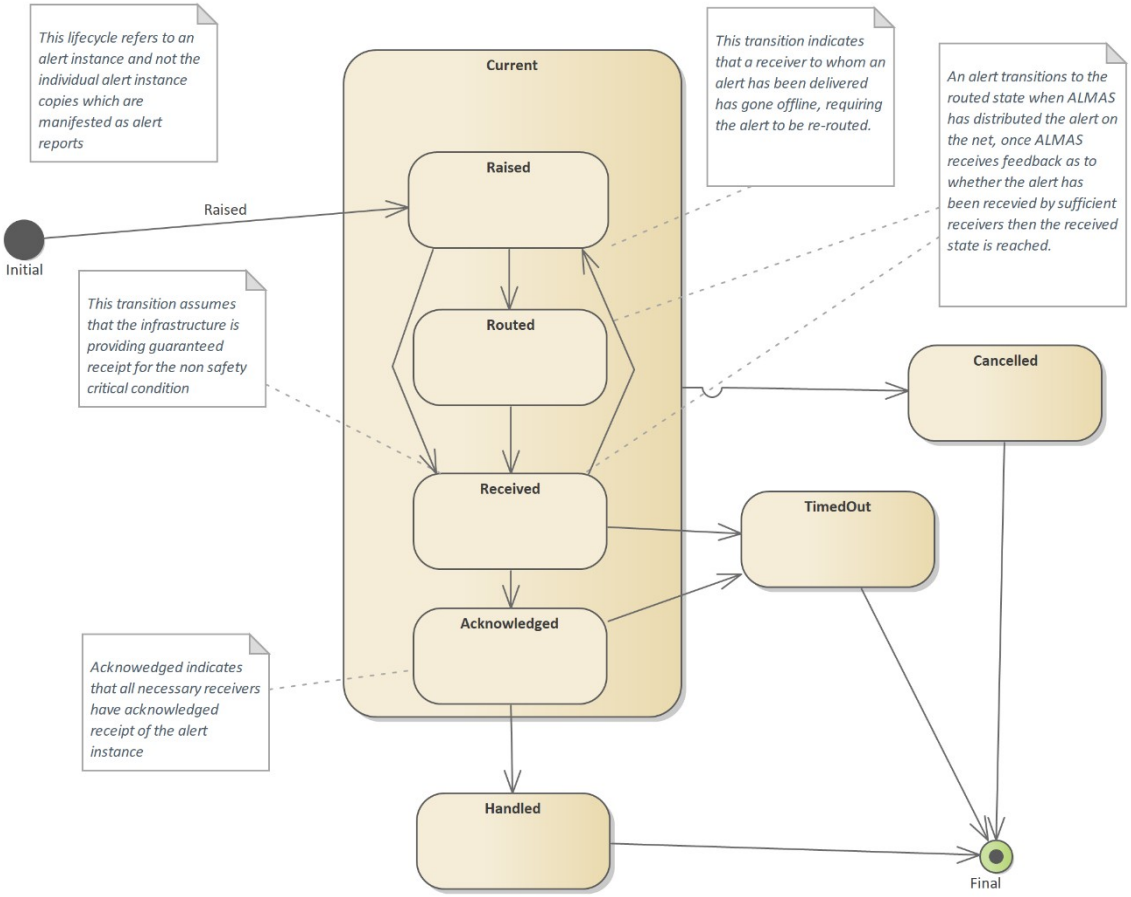

Figure 6-5: Action/Situation Alert Lifecycle

6.5.2 Information Warning Alert State Model

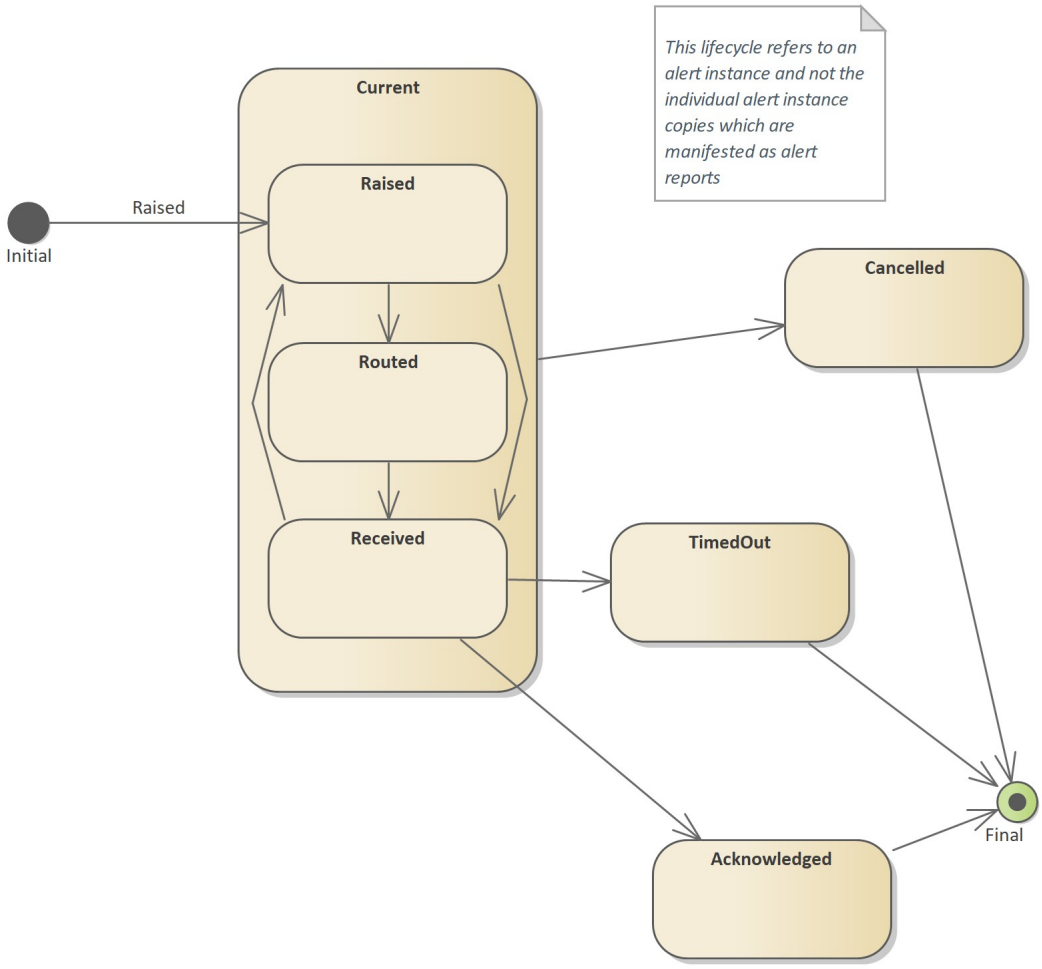

Figure 6-6: Information/Warning Alert Instance Lifecycle

6.5.3 Alert Registration and Creation

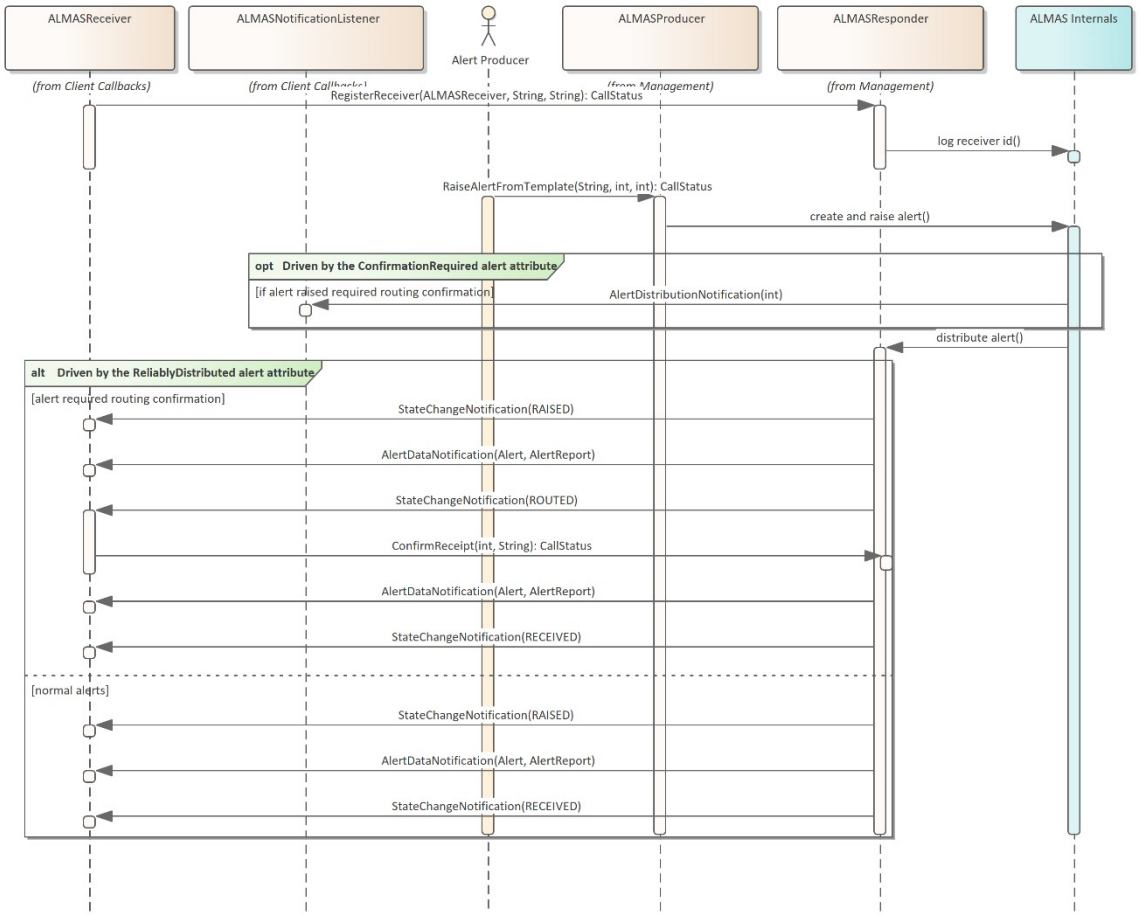

Figure 6-7: Alert Registration and Creation Sequence Diagram

The above sequence diagram shows the interaction with the ALMAS service from several user perspectives.

First it indicates the receiver registration interactions (shown as threads 1 and 2 in the figure).

Second it shows the alert raising interactions from an alert producer, with an illustration of the additional callback made if the alert requires routing confirmation (thread 3 up to 3.1.1).

Interactions 3.1.2 through 3.1.6 are indications of the internal activities but are not requirements upon the internals (hence shown under the fictional class ALMAS System Internals).

Finally, interactions 3.1.6.1-4 and 3.1.6.5-7 are two possible interaction from ALMAS back to the alert receiver, depending upon the ReliablyDistributed attribute of the alert. In the case of this attribute being TRUE then 3.1.6.1-4 are executed, otherwise 3.1.6.5-7 are executed.

This page intentionally left blank.

7 XML Platform Specific Model

7.1 The Template Alert Data specification file

The Template Alert Data specification file is an xml schema document which specifies the ontology of the alert template data to be loaded into an ALMAS by the LoadTemplateSet method. Use of this is therefore effectively optional but any

client that wishes to make use of templates may do so by supplying corresponding valid xml for loading into the system. There are no API methods in this PSM and therefore there is no logging mechanism associated with this PSM.

7.2 The ALMAS configuration file

The ALMAS configuration file is an xml schema document specifying some client specific attributes to allow an ALMAS to be more flexible to a client's specific needs from their ALMAS implementation. This should allow for greater interoperability and usability. It is loaded by use of the LoadConfiguration method.

7.3 The Receiver Hierarchy configuration file

The receiver hierarchy configuration file specifies the structure of the relationships between alert receivers to allow for resilience processing in the event of receiver non-availability. If an alert requires routing to a specific receiver who is not available, then the receiver Hierarchy file specifies a parent receiver in place of the higher-priority receiver originally specified.
Search progresses iteratively up the hierarchy until an available receiver is found in place of the original one.

The receiver hierarchy is loaded via the LoadReceiverHierarchy method.

7.4 The ALMAS categorisation rule file

The categorization rule file is an xml schema document which specifies the categorization rules which can be attached to (or detached from) alerts by means of AttachCategorisationRule method in ALMAS Manager. The configuration file is read by an ALMAS implementation at startup but attaching/detaching of rules to alerts can be done dynamically at runtime using those methods.

8 OMG CORBA/IDL Platform Specific Model

8.1 Rationale

The objective of this PSM is to normalize the CORBA/IDL structures and interfaces. This PSM aims to support the entire PIM interface.

In order for this interface to be reasonably compatible with the DDS PSM, also provided, the data model part is separated from the functional interface model.

All attributes, methods and associations are mapped to IDL elements. As a general rule, therefore, classes with methods are mapped to CORBA/IDL interfaces, classes without methods are mapped to structs, attributes are mapped to CORBA/IDL attributes, associations, and compositions to read only attributes and methods to methods which deal with

errors through CORBA exceptions. Typedef declarations are introduced for UML int attributes mapped to an IDL long, sequences for UML zero-to-many attributes or compositions and to map a PIM date to a CORBA TimeT.

Subscribe methods and indication classes are also mapped within a client IDL file which has to be implemented by clients in order to receive indications (i.e., callbacks) from ALMAS.

The invocation of API methods is logged using the Open Telemetry (OTEL) standard by the implementation of the API method.

9 DDS/DCPS Platform Specific Model

9.1 Rationale

The approach in this PSM is to compare it to the CORBA PSM and highlight differences as necessary. In the DDS PSM two (not exclusive) ways are provided for modeling the management module:

- DCPS-only mapping, in which interfaces are modeled as topics (singletons) and methods as pairs of (request- and reply) topics.
- DLRL mapping, which models classes and methods more directly. The mapping is based on information provided by PrismTech on DLRL data modeling. This entails following when compared to the CORBA PSM:
    - use of valuetypes instead of interfaces – note that a valuetype which is to be distributed by DLRL must inherit from DDS::ObjectRoot
    - there must be an XML-based mapping from DLRL to DCPS. This mapping is not provided in the submission as it is expected that the default DLRL-DCPS mapping is used.

A DCPS-only implementation will use only DCPS-only mapping, while a DLRL implementation will use a combination of DCPS and DLRL mappings.

If DDS_XTYPES is defined topic keys are defined using the @key annotation defined by the DDS_XTYPES specification. Otherwise an alternate #pragma keylist mechanism is used for compatibility with earlier versions of this specification.

The invocation of API methods is logged using the Open Telemetry (OTEL) standard by the implementation of the API method or by a DDS Logging tool using the DDS RTPS protocol.

9.1.1 DCPS level mapping

A generic response topic is used for responses to all method calls; note that this does not provide return values, but just the error code.

Return values are implemented in DCPS by publication of an appropriate topic.

In terms of mapping the PIM-level methods on DCPS, following rules are applied:

- Wherever possible, PIM-level methods are mapped to subscriptions or publications of respective DDS topics. This means that even though these methods cannot be found in the DDS PSM IDL, they can be executed on the PSM level by simply calling the required function from the DDS API. For example, the method GetAlert in ALMAS Manager can therefore be implemented by a DDS read of the Alert topic, with attached condition to receive only the Alert with the ID we are interested in.
- In all other cases, so-called “control topics” are used (such as also applied in the AMSM specification). The names of the topics identify the method which they realize. The control topics also include an identifier of the request (assumed to be uniquely generated by the calling application). The responses to methods are modelled as instances of topic ALMAS_Response, which includes the error code (return_type on the PIM level) and the request identifier (which then can be used to relate the response to the request). In case a method has output parameters other than return_type, these are obtained by reading the relevant topic.

Also, ALMAS_RegisterReceiver and ALMAS_UnregisterReceiver are mapped to DCPS built-in API methods and so are omitted from the IDL for this PSM. It is assumed that request IDs are generated by the producer and that they are unique to an individual ALMAS producer. Topic instances are post-fixed with the Producer ID so that they are unique to a producer. The caller is responsible for finding the instance of topic ALMAS_Response that corresponds to their request. This is in alignment with the approach taken in AMSM.

9.2 DCPS

9.2.1 ALMAS Client

The ALMAS client module is not required in the DDS PSM since this is all available through the use of the standard DDS mechanisms and the topics already defined for ALMAS_StateType and ALMAS_Alert.

9.2.2 ALMAS Management

Parameters of the operation RaiseAlertFromOverrides are implicitly defined as being optional in the PIM; in this PSM they are explicitly marked as optional using an IDL annotation.

The AlertID out parameter in PIM methods RaiseAlertFromOverrides, RaiseAlertWithDynamicData, RaiseAlertFromData and RaiseAlertFromTemplate is mapped to the ALMAS_CreatedAlert topic type so that producers are aware of the alert id for alerts they have raised in order to cancel them as appropriate.

The following table provides explanation of the mapping of methods in the ALMAS Management module. Only those methods which are mapped directly to DDS level constructs are listed in the table, all methods which are mapped on “control topics” are listed in the subsequent IDL file.

Class (PIM level)	Method	DDS mapping
ALMAS Manager	GetAlert(int, Alert)	DDS read with query condition
ALMAS Manager	GetAlerts(String, SortedAlertSet)	DDS read with query condition
ALMAS Manager	GetTemplate(int)	DDS read with query condition.
ALMAS Manager	GetAllTemplateIDs(String, TempalteIDSet)	DDS read with query condition.
ALMAS Manager	RegisterNotificationListener(ALMAS Notification Listener)	Creation of a new DDS Listener.

9.2.3 DCPS topics QoS

ALMAS topics share the same values for most of the DDS QoS (cf. [DDS]):

QoS	Value
USER_DATA	<unspecified>
TOPIC_DATA	<unspecified>

GROUP_DATA	<unspecified>
PRESENTATION	<unspecified>
DEADLINE	Period = infinite
LATENCY_BUDGET	duration = <unspecified>
OWNERSHIP	EXCLUSIVE
OWNERSHIP_STRENGTH	<unspecified>
LIVELINESS	kind = AUTOMATIC / lease_duration = <unspecified>
TIME_BASED_FILTER	<unspecified>
PARTITION	<unspecified>
TRANSPORT_PRIORITY	value=0
DESTINATION_ORDER	BY_SOURCE_TIMESTAMP
HISTORY	kind = KEEP_LAST / depth = 1
RESOURCE_LIMITS	All unlimited.
ENTITY_FACTORY	<unspecified>
WRITER_DATA_LIFECYCLE	<unspecified>
READER_DATA_LIFECYCLE	<unspecified>

The other QoS (DURABILITY, RELIABILITY and LIFESPAN) will be allocated with the following principle:

- As for the "Control topics" (both requests and responses), they have DURABILITY equals to VOLATILE, RELIABILITY set to RELIABLE and LIFESPAN.duration defined by the implementation:

  DURABILITY VOLATILE

  RELIABILITY kind = RELIABLE

  LIFESPAN Implementation dependant

- Others topics have DURABILITY to TRANSIENT, RELIABILITY set to RELIABLE and LIFESPAN.duration to infinite:

  DURABILITY TRANSIENT

  RELIABILITY kind = RELIABLE

  LIFESPAN duration = infinite

9.3 DLRL

9.3.1 ALMAS Client

The ALMAS client module is not required in the DDS PSM since this is all available through the use of the standard DDS mechanisms and the topics already defined for ALMAS_StateType and ALMAS_Alert (i.e., through the DCPS mapping).

This page intentionally left blank.

10 COM IDL Platform Specific Model

10.1 Rationale

The objective of this PSM is to normalize the structures and interfaces required for a COM implementation of the standard. This PSM aims to support the entire PIM interface.

In order for this interface to be reasonably compatible with the other PSMs provided in this document, the data model part is separated from the functional interface part.

All attributes, methods and associations are mapped to COM IDL elements. As a general rule, therefore, classes with methods are mapped to COM interfaces, classes without methods are mapped to structs, attributes are mapped to interface read/write methods. All return parameters and exceptions are mapped to method out parameters with the COM HRESULT returned from all interface methods.

Subscribe methods and indication classes are also mapped within a client IDL file which has to be implemented by clients in order to receive indications (i.e., callbacks) from ALMAS.

The invocation of API methods is logged using the Open Telemetry (OTEL) standard by the implementation of the API method.

This page intentionally left blank.

11 GraphQL Platform Specific Model

11.1 Rationale

The GraphQL PSM defines a single schema definition file for the ALMAS Data Model, ALMAS Management and ALMAS Client Callbacks packages defined by the PIM. Classes from the ALMAS Dtaa Model of the PIM are mapped to GraphQL types within the schema.

The detailed rules for the MDA code generation from the ALMAS Data Model PIM to the GraphQL PSM schema are as follows:

- The PIM attributes are mapped to GraphQL attributes;
- PIM attributes with multiplicity 1 are mapped to non-nullable GraphQL attributes
- Collections in the PIM are mapped to GraphQL arrays;
- Aggregation and compositions are mapped to GraphQL attributes;
- Association classes follow the equivalent mapping as the CORBA and DDS PSMs

The schema supports GraphQL clients for interfaces defined in the ALMAS Management and ALMAS Client Callbacks PIM packages. Mutations are used to invoke PIM interface methods; queries and subscriptions are used to process those invocations.

The PSM method for connecting to other components is through the underlying HTTPS web service connection. Web-sockets are used for subscription callbacks.

Specific rules for the MDA code generation from the Service Model PIM to the GraphQL PSM IDL are as follows:

- Interface method (in) parameters and return values (including out parameters) in the Service Model are each mapped to a (query) type, an input type and update type; these are for queries, mutations, and subscriptions respectively.
- To invoke a method an ALMAS client makes a mutation (with method parameter type) and subscribes or queries for the response (with the return value type).
- To process a method an ALMAS client queries or subscribes for the method parameter type and makes a mutation with the response (with the return value type).
- As per the DDS PSM, each of the parameter and return value types contain a request id; the instigator is responsible for allocating unique request ids in the scope of the ALMAS system; the processing component is responsible for labelling responses with the received request id so that the instigator can locate the corresponding response.
- The GraphQL schema Query type supports queries for any combination of interface methods in the Service Model.
- The GraphQL schema Mutation type supports invocation of single or multiple instances of any combination of interface methods in the Service Model.
- The GraphQL schema Subscription type supports subscription for any combination of interface methods in the Service Model.
- The following methods are mapped to the return (out parameter) type with a GraphQL filter condition in the query: GetAlert, GetAlerts, GetTemplate and GetAllTemplateIDs
- RegisterNotificationListener is mapped to a GraphQL subscription.
- Parameters of the operation RaiseAlertFromOverrides are implicitly defined as being optional in the PIM; in this PSM they are explicitly marked as optional using an IDL annotation.

The invocation of API methods is logged using the Open Telemetry (OTEL) standard by the implementation of the API method.